Research Report

I. Title of Report

Reproducibility of Script-Based Workflows: A Case Study and Demonstration

II. Investigators:

Qiwen Wang
2016 Summer SPIN Intern
Undergraduate Student
Mathematics and Computer Science

SPIN Mentor: Bertram Ludaescher

III. Abstract/Synopsis

In this project we experimented with a number of technologies and tools that can improve the reproducibility of script-based workflows. We also used several existing usage case written in python script to test the reproducibility of the workflow tools and study their benefits and limitations. We first tried out YesWorkflow and NoWorkflow on the use case. We can conclude that YesWorkflow is a convenient tool to capture prospective provenance, but can also capture some extent of retrospective provenance. NoWorkflow can capture very detail retrospective provenance. Furthermore, we also examined a new developed tool: YesWorkflow-NoWorkflow Bridge which can capture the provenance that can't be achieved by either one.

III. Methodology

1. LIGO Use Case

   LIGO script refers to the python script that comples some signal processing tasks on strain time-series data associated with the LIGO GW150914 data released from the LIGO Open Science Center. The detail description of LIGO script is over here. The script reads 4 hdf5 files that represent H1 and L1 with durations of 32 and 4096 seconds around GW150914, sampled at 16384 and 4096 Hz. Then it does signal processing tasks like create spectrogram, filter data, downsampling, bandpassing, and generate wavefiles. It also outputs some visualization for the result using the python library matplotlib.

2. YesWorkflow

   YesWorkflow is a scientific workflow management system. In YesWorkflow, user can mark the script with YesWorkflow annotation in the comment. Some sample YesWorkflow annotations are @begin @in @desc @param @uri. The good thing about YesWorkflow is that the tags are very self-explanatory, and it is language independent, that means user can use YesWorkflow in python, MATLAB, R, etc.

   In this project, I first went over the YesWorkflow tutorial on the github page. I learnt about the meaning of every tag, and how to mark a simple script with YesWorkflow and output the graph with different views, i.e. process view, data view, and combined view.

   Then I marked our use case LIGO script with YesWorkflow, from a very high level markup to a markup that has detail description to every code block, and finally finished the markup that has a detail overview to every function and some URI template that allow

Then I marked our use case LIGO script with YesWorkflow, from a very high level markup to a markup that has detail description to every code block, and finally finished the markup that has a detail overview to every function and some URI template that allow users to trace the retrospective provenance later.

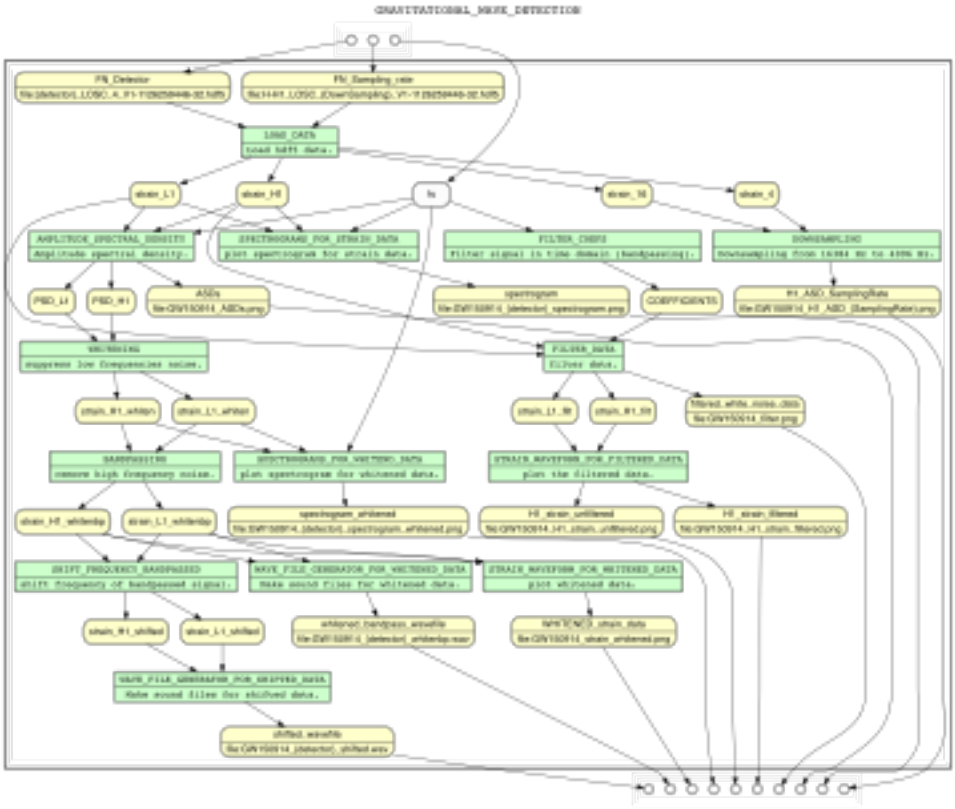
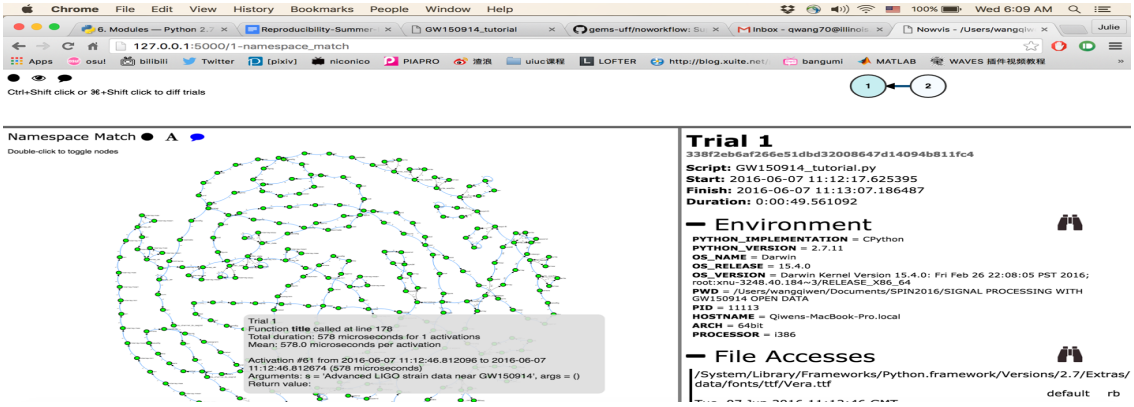


Figure1: A Combined View of YesWorkflow From Ligo

3. NoWorkflow

NoWorkflow automatically captures provenance from Python scripts without requiring any modifications to the script, unlike YesWorkflow that needs to annotate the script with special tags. User can run the script for any Python file, and for each trial(run), NoWorkflow captures definition, deployment and execution provenance and stores it in a SQL database, so that user can use it to do further analysis.

I also go over the NoWorkflow tutorial on Github. It has an organized interface that can show the running log and the dataflow of the script. It also utilizes IPython and Jupyter to interactively visualize the workflow and query for the provenance. When I try to apply NoWorkflow on LIGO script, it generates a huge graph that includes all the low level details, such as which python built-in function is being used with what parameter.



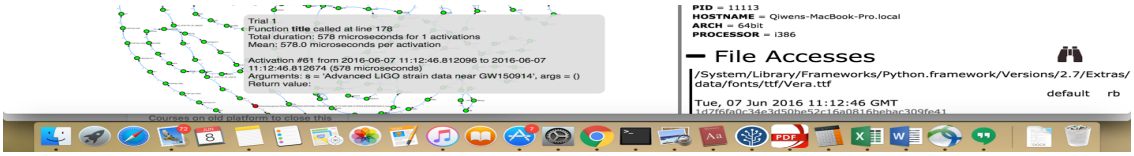


Figure2: Provenance Visualization from NoWorkflow of LIGO

4. Query on YesWorkflow

   YesWorkflow and NoWorkflow use deductive database – SWI Prolog and XSB Prolog as their query language, and they have already developed some good queries for asking questions for prospective and retrospective provenance. However, as this kind of deductive databases are not so popular these days, and people tend to use relational database systems, like SQL, I tried to write SQLite queries that can answer the same existing questions for YesWorkflow. Since I had no previous experience with database and query language, I learnt the basis of deductive database and relational database. In this way, I'm able to read and interpret XSB Prolog in YesWorkflow into SQLite query.

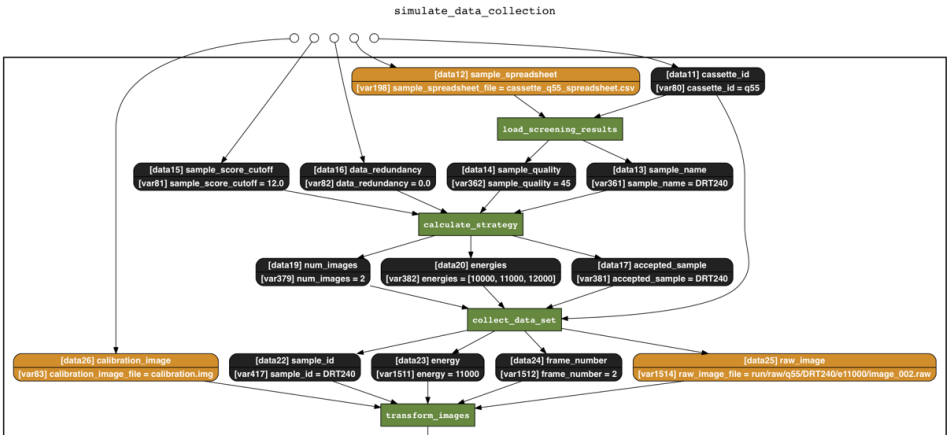
5. Other Use Case

   Beside the use case LIGO, I also picked up some other use case related to machine learning and data science – LedDetector, a python script that takes a video as the input, and output the number and positions; simulate_data_collection, a python script widely used in YesWorkflow experiment that collect diffraction data from high quality crystals in a cassette. I used YesWorkflow and NoWorkflow on it, and tested similar SQLite queries as I wrote for LIGO script, to test if these queries are robust and can work for all the script.

6. uery on YesWorkflow-NoWorkflow Bridge

   The yw-noworkflow explores how the complementary provenance information provided by YesWorkflow and NoWorkflow can be used together to yield visualizations and query results that neither can provide on its own. As the previous section introduced, YesWorkflow captures prospective provenance by user defined YW annotation, and NoWorkflow uses python library to capture retrospective provenance. YesWorkflow-NoWorkflow Bridge gathers retrospective provenance being captured from NoWorkflow but defined inside the YesWorkflow block. From the connection between YesWorkflow input & output, and the actual names of the parameter, we can get the specific value used to produce the specific output. I mainly worked on solving these two questions on simulate_data_collection here:

   1. hat Python variables carries values of DataName into the BlockName workflow step?
   2. hat values are emitted by the DataName output of the BlockName step?



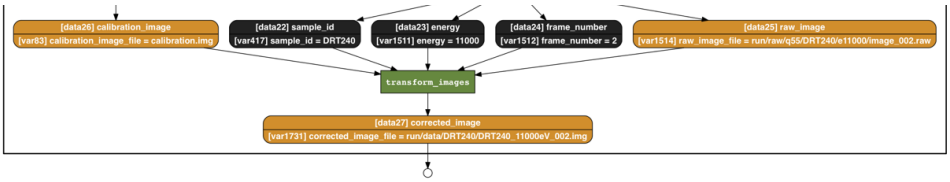


Figure 3: Provenance Visualization from YW-NW of simulate_data_collection

7. evelop a Deliverable Product

The aim of this project is to explore the reproducibility of different kinds of workflow. In order to let users who don't have experience with YesWorkflow and NoWorkflow before, I wrote a YesWorkflow-NoWorkflow Bridge readme file, that guides the users to go through the basis of this bridge, understand the high level idea of what every command is doing, and reproduce the result, the graphics and query results by themselves. In this way, they can apply YesWorkflow-NoWorkflow Bridge to their own python script, and ask provenance related questions to their script.

IV. Results

In this section, I will discuss the results from the Methods section.

1. sWorkflow queries and results:

Here are parts of SQLite query for YesWorkflow, and there answer for LIGO script.

English question:
EQ1: What source files SF were YW annotations extracted from?
SQLite query:
```
SELECT source_path FROM Extractfacts_extract_source;
```
Result:
```
./GW150914_tutorial.py
```

English question:
EQ2: What are the names N of all program blocks?
SQLite query:
```
SELECT value FROM extractfacts_annotation WHERE LOWER(keyword)
== '@begin';
```
Result:
```
GRAVITATIONAL_WAVE_DETECTION
LOAD_DATA
AMPLITUDE_SPECTRAL_DENSITY
WHITENING
BANDPASSING
STRAIN_WAVEFORM_FOR_WHITENED_DATA
SPECTROGRAMS_FOR_STRAIN_DATA
SPECTROGRAMS_FOR_WHITEND_DATA
FILTER_COEFS
FILTER_DATA
STRAIN_WAVEFORM_FOR_FILTERED_DATA
WAVE_FILE_GENERATOR_FOR_WHITENED_DATA
SHIFT_FREQUENCY_BANDPASSED
WAVE_FILE_GENERATOR_FOR_SHIFTED_DATA
DOWNSAMPLING
```

There are twenty more queries related to YesWorkflow, and can be checked out in LIGO repository.

There are twenty more queries related to YesWorkflow, and can be checked out in LIGO repository.

2. YesWorkflow-NoWorkflow Bridge queries and results

English question:

```
YW-NW-Q1:  What Python variables carries values of DataName
into the BlockName workflow step?
```

SQLite query:

```
      SELECT variable_id,variable_name,variable_value
 FROM yw.yw_flow yf, nw_variable_for_yw_in_port_10 vip
  WHERE yf.data_name = :DataName AND yf.sink_program_name =
:BlockName
   AND yf.sink_port_id = vip.port_id;
```

Result:

```
Variable ID: 255. Variable Name: sample_name, Variable Value
'DRT110'
Variable ID: 361. Variable Name: sample_name, Variable Value
'DRT240'
Variable ID: 2511. Variable Name: sample_name, Variable Value
'DRT322'
```

English question:

```
YW-NW-Q2:  What values are emitted by the DataName output of
the BlockName step?
```

SQLite query:

```
      SELECT variable_id, variable_name,variable_value
 FROM yw.yw_step_output yo, nw_variable_for_yw_out_port vop
 WHERE vop.port_id = yo.port_id
 AND yo.program_name = :BlockName AND yo.data_name = :DataName;
```

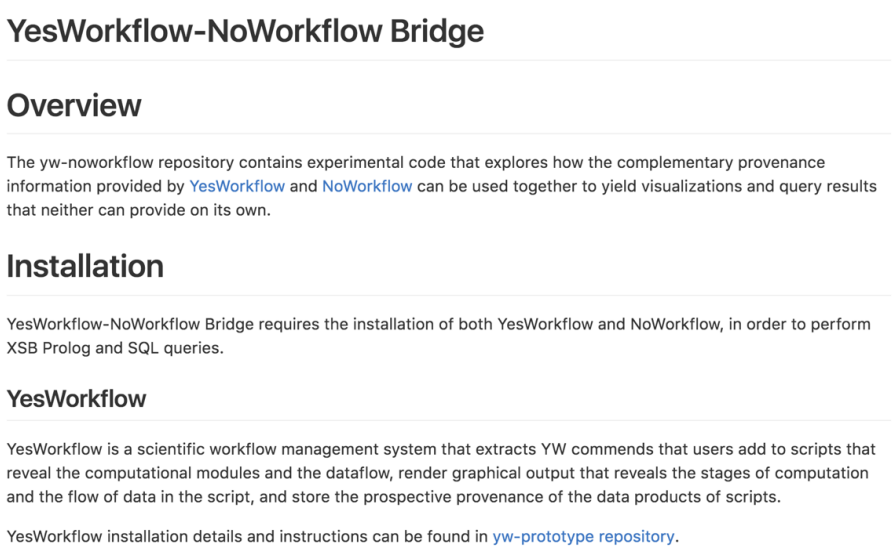
Result:

```
Variable ID: 708. Variable Name: corrected_image_file,
Variable Value 'run/data/DRT240/DRT240_10000eV_001.img'
Variable ID: 1048. Variable Name: corrected_image_file,
Variable Value 'run/data/DRT240/DRT240_10000eV_002.img'
Variable ID: 1390. Variable Name: corrected_image_file,
Variable Value 'run/data/DRT240/DRT240_11000eV_001.img'
Variable ID: 1730. Variable Name: corrected_image_file,
Variable Value 'run/data/DRT240/DRT240_11000eV_002.img'
Variable ID: 2072. Variable Name: corrected_image_file,
Variable Value 'run/data/DRT240/DRT240_12000eV_001.img'
Variable ID: 2412. Variable Name: corrected_image_file,
Variable Value 'run/data/DRT240/DRT240_12000eV_002.img'
Variable ID: 2857. Variable Name: corrected_image_file,
Variable Value 'run/data/DRT322/DRT322_10000eV_001.img'
Variable ID: 3197. Variable Name: corrected_image_file,
Variable Value 'run/data/DRT322/DRT322_10000eV_002.img'
Variable ID: 3539. Variable Name: corrected_image_file,
Variable Value 'run/data/DRT322/DRT322_11000eV_001.img'
Variable ID: 3879. Variable Name: corrected_image_file,
Variable Value 'run/data/DRT322/DRT322_11000eV_002.img'
```

3. Deliverable Product under YW-NW repository

The deliverable product of YW-NW is a readme markdown file for Github. It consists of the introduction of YW-NW, the prerequisite to use YW-NW, software to install on the computer, and the step by step tutorial of how to use YesWorkflow-NoWorkflow Bridge. People can check out the Yw-Noworkflow repository.

YesWorkflow-NoWorkflow Bridge



YesWorkflow-NoWorkflow Bridge

Overview

The yw-noworkflow repository contains experimental code that explores how the complementary provenance information provided by YesWorkflow and NoWorkflow can be used together to yield visualizations and query results that neither can provide on its own.

Installation

YesWorkflow-NoWorkflow Bridge requires the installation of both YesWorkflow and NoWorkflow, in order to perform XSB Prolog and SQL queries.

YesWorkflow

YesWorkflow is a scientific workflow management system that extracts YW commends that users add to scripts that reveal the computational modules and the dataflow, render graphical output that reveals the stages of computation and the flow of data in the script, and store the prospective provenance of the data products of scripts.

YesWorkflow installation details and instructions can be found in yw-prototype repository.

Figure 4: A Screenshot of the readme file

V. Conclusion

In this internship, I observed different kinds of tool that can enhance the reproducibility of the scientific workflow. That includes: YesWorkflow, NoWorkflow, YesWorkflow-NoWorkflow Bridge, MakeFile, ReproZip, Docker, etc. I also wrote queries in SQLite that can answer various kinds of scientific workflow related question. For the detail of the use cases in this project, you can check out the following Github Repository: Yw-Noworkflow, LIGO, LedHighliter.

X. References

T. McPhillips, T. Song, T. Kolisnik, S. Aulenbach, K. Belhajjame, R.K. Bocinsky, Y. Cao, J. Cheney, F. Chirigati, S. Dey, J. Freire, C. Jones, J. Hanken, K.W. Kintigh, T.A. Kohler, D. Koop, J.A. Macklin, P. Missier, M. Schildhauer, C. Schwalm, Y. Wei, M. Bieda, B. Ludäscher (2015). **YesWorkflow: A User-Oriented, Language-Independent Tool for Recovering Workflow Information from Scripts**. *International Journal of Digital Curation* **10**, 298-313.

T. McPhillips, S. Bowers, K. Belhajjame, B. Ludäscher (2015). **Retrospective Provenance Without a Runtime Provenance Recorder**. *7th USENIX Workshop on the Theory and Practice of Provenance (TaPP'15)*.

PIMENTEL, J. F. N.; FREIRE, J.; BRAGANHOLO, V.; MURTA, L. G. P.; Tracking and Analyzing the Evolution of Provenance from Scripts. In: International Provenance and Annotation Workshop (IPAW), 2016, McLean, Virginia.

PIMENTEL, J. F. N.; DEY, S.; MCPHILLIPS, T.; BELHAJJAME, K.; KOOP, D.; MURTA, L. G. P.; BRAGANHOLO, V.; LUDÄSCHER B.; Yin & Yang: Demonstrating Complementary Provenance from noWorkflow & YesWorkflow. In: International Provenance and Annotation Workshop (IPAW), 2016, McLean, Virginia.