

L-Vis: Visualizing Language-Level Provenance for Program Comprehension

Joe Wonsil and Francis Nguyen

December 10th, 2019

CPSC 547 Final Project

Driving Scenario: Grad Student Code

- A new graduate student inherits (messy) code for data analysis and must understand how to use and alter the scripts.
- Goal: build mental model of unfamiliar code.
 - Useful information can include **library dependencies and their usage**, understanding how **variables change over time** and the **relationship between inputs and outputs** of the analysis code.
- L-Vis visualizes relationships in the code structure of R scripts to help users understand unfamiliar code.

Requirements & Tasks

We conducted unstructured interviews to understand how program comprehension and visualization interact.

- **Participants:** software engineering professor, peer CS graduate students, and introductory CS undergrads.

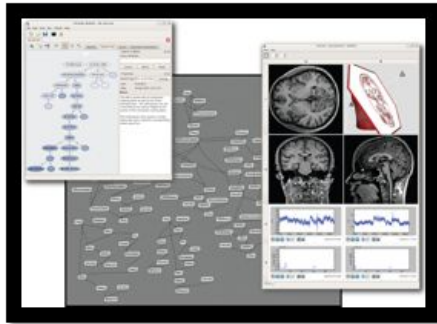
Their responses helped define tasks L-Vis should consider in its design.

- **Locate:** Identify parts of code an external library affects.
- **Locate:** Display affected code if a user changes a variable's value.
- **Present:** Highlighting the flow of inputs to output through a script.

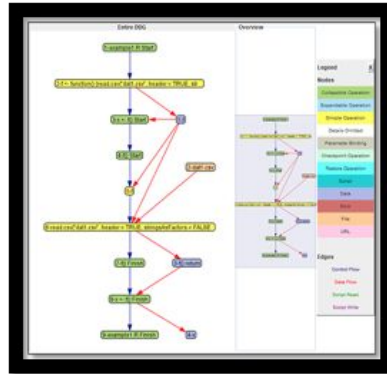
These tasks can be fulfilled by **provenance**.

Provenance is an object's history represented as a directed acyclic graph (DAG), which consists of **nodes** and **edges**.

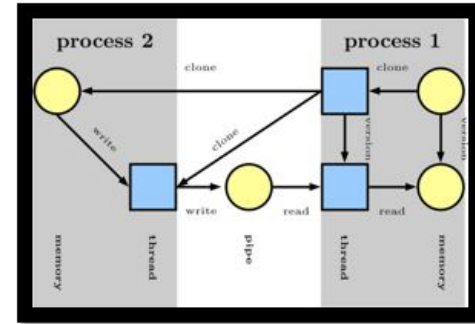
- This conceptual specification is defined by the **W3C PROV Data Model**¹ (a standardized model agnostic of level and implementation).



Application



Language



System

1. Khalid Belhajjame, Reza B'Far, James Cheney, Sam Coppens, Stephen Cresswell, Yolanda Gill, Paul Groth, Graham Klyne, Timothy Lebo, Jim McCusker, and et al. PROV-DM: The PROV Data Model.

Focus: **Language-Level Provenance (LL-prov)**

Scale: Line by line source-code level. For L-Vis, the language is **R**.

LL-Prov contains information about a past execution including:

- External library dependencies.
- Function calls.
- How inputs interact to create outputs.

RDataTracker (RDT)¹ is an R package that collects LL-Prov.

- Generates PROV-JSON² file of nodes/edges which is a serialization of the PROV Data Model.
- PROV-JSON is the input to L-Vis.

1. B.S. Lerner and E.R. Boose. Rdatatracker and ddg explorer. In Ludäscher B., Plale B. (eds) Provenance and Annotation of Data and Processes (IPAW 2014), volume 8628, pages 288–290, 2015

2. Trung Dong Huynh, Michael O. Jewell, Amir Sezavar Keshavarz, Danius T. Michaelides, Huanjia Yang, and Luc Moreau. The PROV-JSON serialization, 2013.

Our Tool: L-Vis

L-Vis: Visualizing Language-Level Provenance

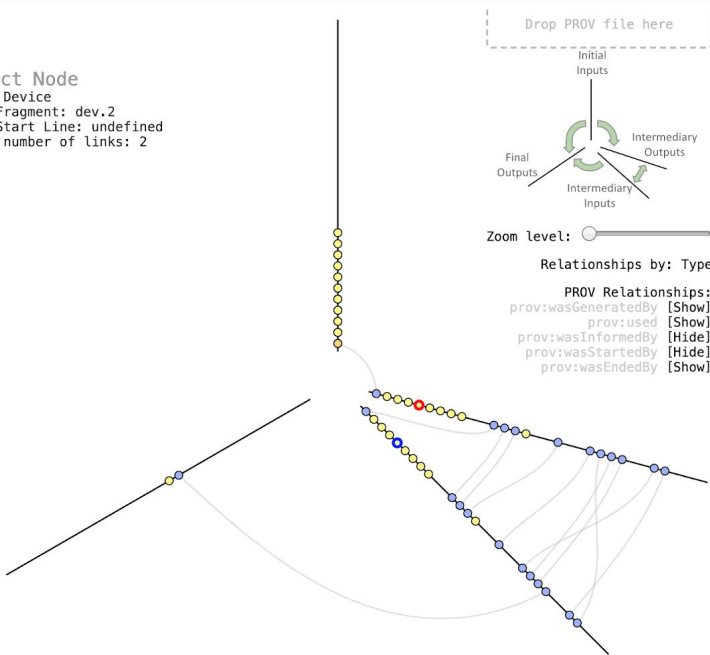
R Script Editor

```
1
2 # =====
3 # Paper title: It's All About Political Incentives: Democracy and t
4 # Authors: Patrick Bayer (Glasgow) and Johannes Urpelainen (Columbi
5 # Journal of Politics
6
7 # Last modified: November 7, 2015
8
9 # Purpose: R code creates the lineplot in Figure 2(a)
10 # =====
11
12 rm(list=ls())
13
14 year <- seq(1990,2012,1)
15
16 demo <- c(1,1,1,2,3,0,0,1,1,3,0,3,4,5,2,2,2,8,3,2,2,1,0)
17 auto <- c(0,0,0,0,0,0,0,0,0,0,0,2,0,0,2,1,1,1,2,2,1,1)
18
19
20 # =====
21 pdf("lineplot.pdf", height=6, width=6)
22 # =====
23
24 par(mar=c(4,4,4,4))
25
26 # line plot
27 plot(x=year,y=cumsum(demo),type="l",xaxp=c(1990,2012,22),cex.axis=.
28       ylab="FIT adoption (cumulative)",xlab="Year",ylim=c(0,50),yaxp=
29       main=c("FIT Adoption over Time, 1990-2012"),las=3)
30 lines(x=year,y=cumsum(auto))
31 text(x=2010,y=49,labels=c("Democratic
32 countries"),cex=0.9)
33 text(x=2010,y=16,labels=c("Autocratic
34 countries"),cex=0.9)
35
36 # =====
37 dev.off()
38 # =====
39
```

d5

Object Node

Type: Device
Code Fragment: dev.2
Code Start Line: undefined
Total number of links: 2

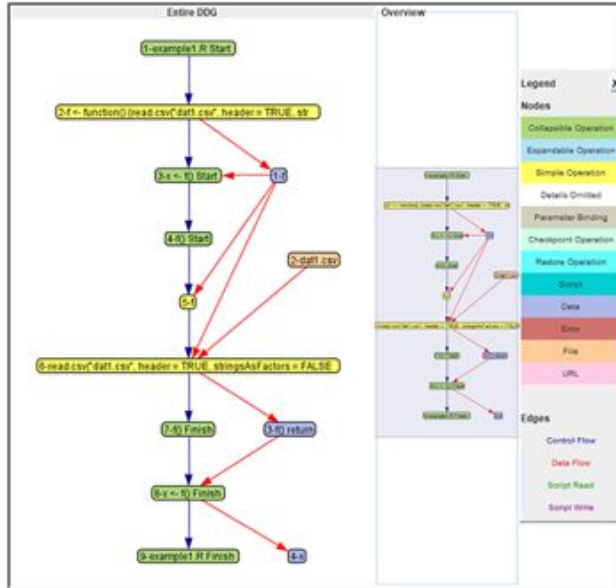


Prov-JSON

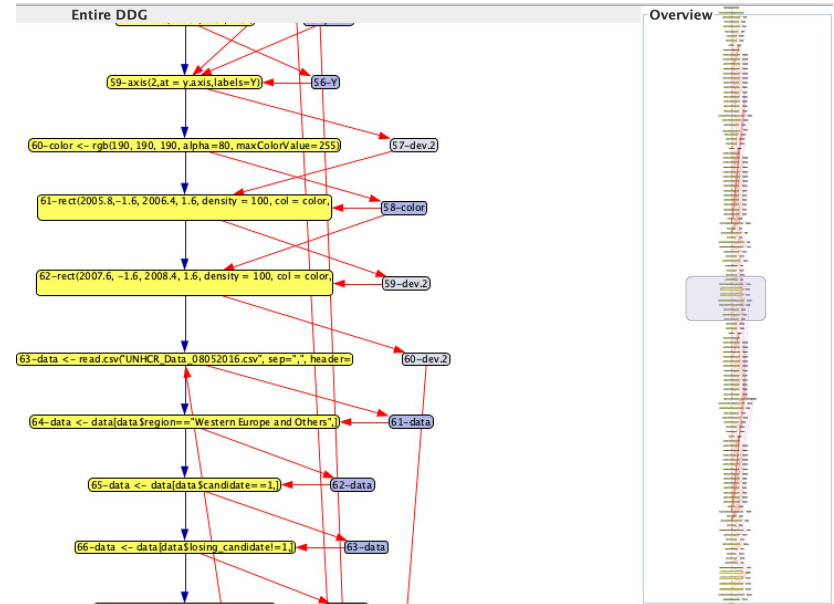
```
{
  "root": {
    "prefix": {
      "prov": {
        "string": "https://www.w3.org/ns/prov#"
      },
      "rdt": {
        "string": "https://github.com/End-to-end-provenance/extendedProvJson/blob/master/format.md"
      }
    }
  },
  "agent": {
    "rdt:trial": {
      "rdt:tool.name": "string \"rdtLite\"",
      "rdt:tool.version": "string \"1.2\"",
      "rdt:json.version": "string \"2.3\"",
      "rdt:args.names": {
        0: "string \"overwrite\"",
        1: "string \"details\"",
        2: "string \"snapshot.size\"",
        3: "string \"save.debug\""
      },
      "rdt:args.values": {
        0: "string \"TRUE\"",
        1: "string \"TRUE\"",
        2: "string \"0\"",
        3: "string \"FALSE\""
      },
      "rdt:args.types": {
        0: "string \"logical\"",
        1: "string \"logical\"",
        2: "string \"numeric\"",
        3: "string \"logical\""
      }
    }
  },
  "activity": {
    "rdt:tpl": {
      "rdt:name": "string \"lineplot.R\"",
      "rdt:type": "string \"Start\"",
      "rdt:elapsedTime": "string \"0.19\"",
      "rdt:scriptNum":
    }
  }
}
```


Issue: Scale

Images example of existing LL-Prov visualization tool: DDG Explorer.



2 lines of code
12 Nodes, 8 Edges



180 lines of code
267 Nodes, 435 Edges

Data Abstraction — Nodes / Edges

NODES

Prov Data Model

PROV-JSON

Fully Abstracted

“Entities”



Data node
Library node
Function node



Objects:

Variables, external dependencies, files (I/O)

“Activities”



Procedure node



Actions:

Executed code segment, typically a single line

EDGES

Indicate action chronology, object creation, and object usage

“Node (a) occurred before Node (b)”

Data Abstraction Cont.

Fully Abstracted

Objects:

Yellow Nodes

Variables, external dependencies, files (I/O)

Actions:

Blue Nodes

Executed Code Segments

p3

Action Node

Type: Operation

Code Fragment: year <- seq(1990,2012,1)

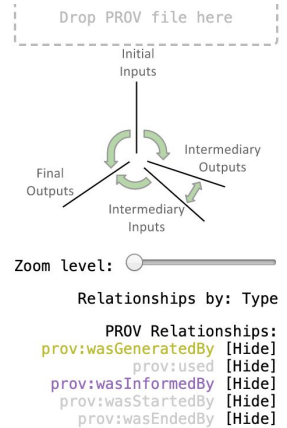
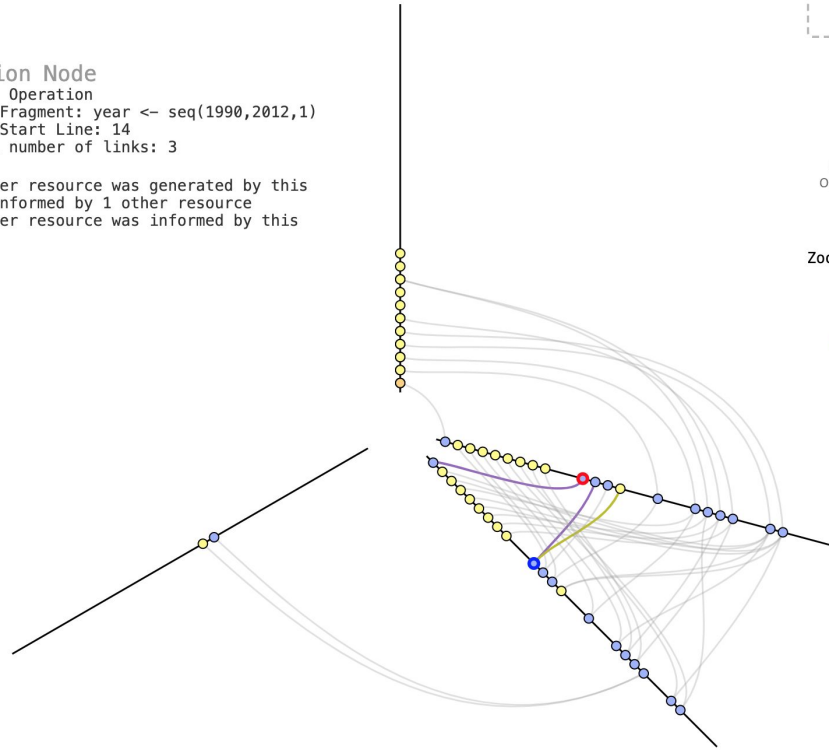
Code Start Line: 14

Total number of links: 3

1 other resource was generated by this

Was informed by 1 other resource

1 other resource was informed by this



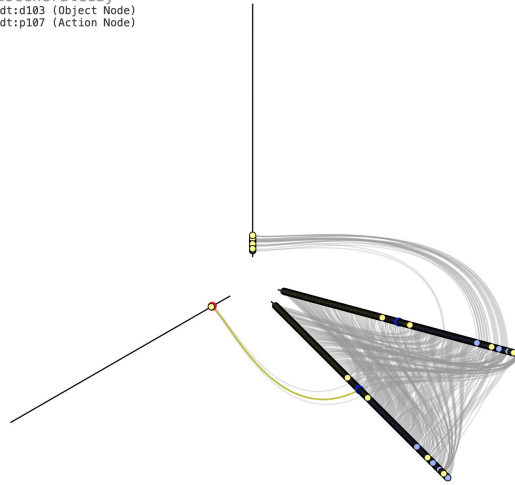
Data Abstraction Cont.

Derived Attribute

Crash Nodes:

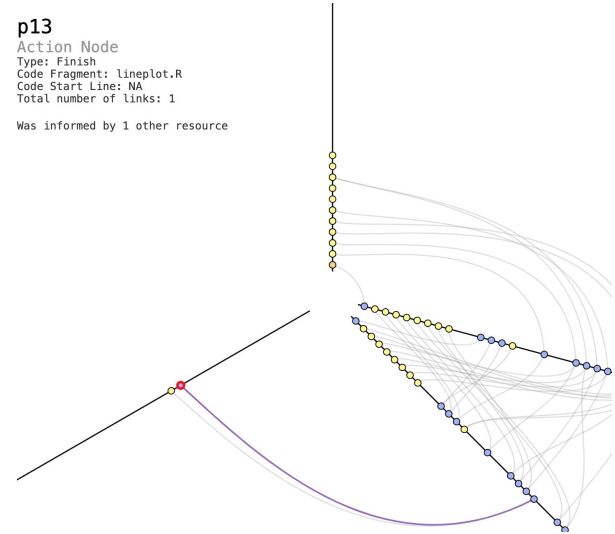
An action that produces a new object by using two or more existing objects

```
prov:wasGeneratedBy
Source: rdt:d103 (Object Node)
Target: rdt:p107 (Action Node)
```



Without crash nodes

```
p13
Action Node
Type: Finish
Code Fragment: lineplot.R
Code Start Line: NA
Total number of links: 1
Was informed by 1 other resource
```



With crash nodes

L-Vis Demo

Limitations

- Resources are limited. Not enough pixels, but also direction and hierarchy along hive plot axes have limitations.
- Are these the most elegant encodings? User study is necessary to evaluate effectiveness of idioms.
- L-Vis uses PROV-JSON as its input. Trade-Off: More specificity in schema, but code needs to be updated if schema is updated.

Future Work

In the very near future (for CPSC 508):

- Fully integrate L-Vis into existing reproducibility tool containR
- (Less rigorous) Qualitative study with our peers to evaluate L-Vis

On the horizon:

- Additional visualization idioms: Allow users to toggle traditional network layouts or radial layouts
- User interviews to derive tasks from users and get iterative feedback
- Quantitative study to compare L-Vis to other LL-prov tools

Questions?

L-Vis: Visualizing Language-Level Provenance

