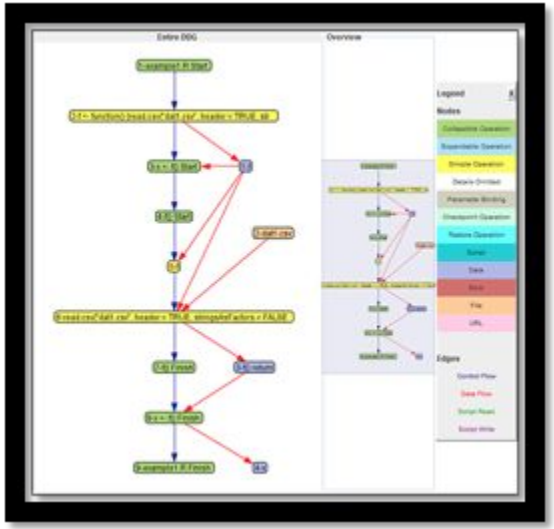
L-Vis: Analyses on My Mind

Visualizing Language-Level Provenance

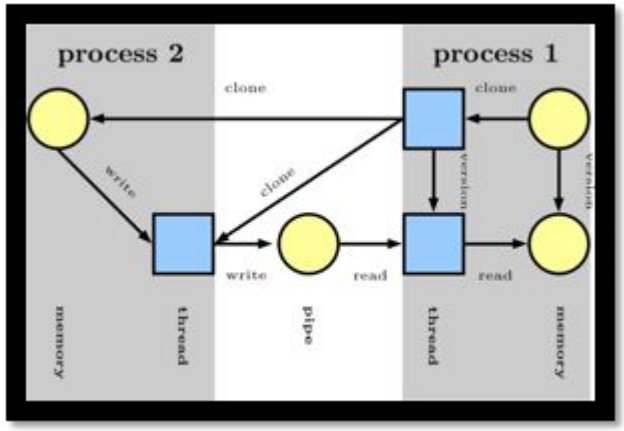
Francis Nguyen and Joseph Wonsil

Digital Provenance

Application



Language



System

Application-Level Provenance in Visualization

- Depicts **workflows** (a series of tasks) and **processes**
- Useful in deriving **current state** and **possible choices** or **explored options**
- Can track data-flow through a system, but tends to focus on interaction history or task history
- Useful in visual analytics!

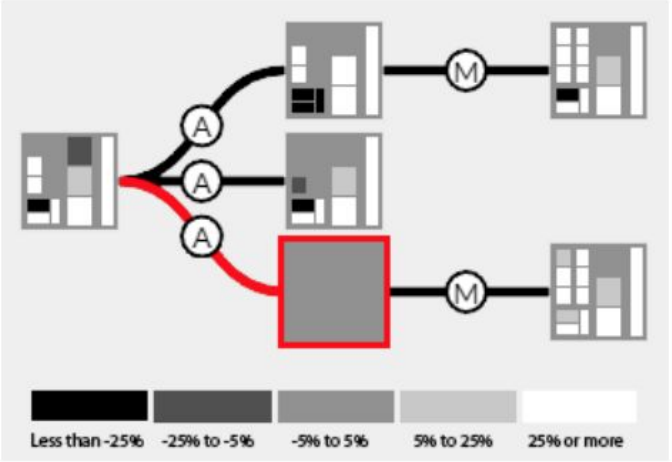
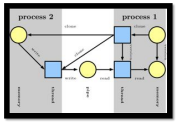


Fig 5. Provenance Tree from *Walch et al. 2018.* LightGuider: Guiding Interactive Lighting Design using Suggestions, Provenance, and Quality Visualization

System-Level Provenance



- System level collects **execution traces** at the level of the operating system
- Useful for security and verification (look at deltas in provenance)

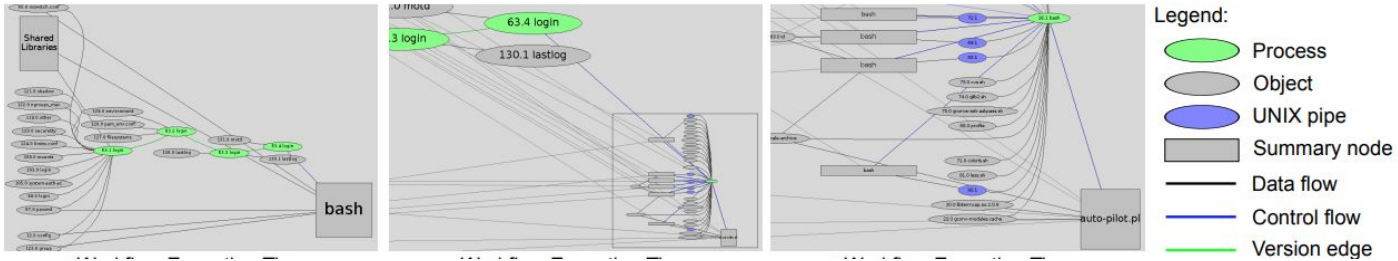
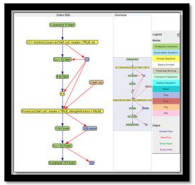


Fig 1. Macko & Seltzer. Provenance Map Orbiter: Interactive Exploration of Large Provenance Graphs.

Our focus: **Level-Language (LL) Provenance**



- Useful for reproducibility in scientific analysis
- Visualization space is relatively unexplored
- Has a *manageable eco-system to leverage (containR, RDT, Dataverse)
- We wanted to make our lives easier when inheriting “grad student code”

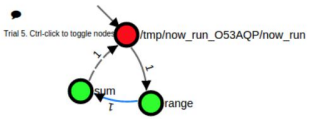
*Still painful but more accessible than system-level provenance tools

```
In [1]: %load_ext noworkflow
        %now_set_default graph.width=392 graph.height=150

In [2]: trial = %now_run script1.py --name tapp
        trial.id
Out[2]: 4

In [3]: size = 5

In [4]: %%now_run --name tapp --out=out_var $size
        import sys
        l = range(int(sys.argv[1]))
        c = sum(l)
        print(c)
Out[4]:
```




```
In [5]: out_var
Out[5]: '10\n'
```

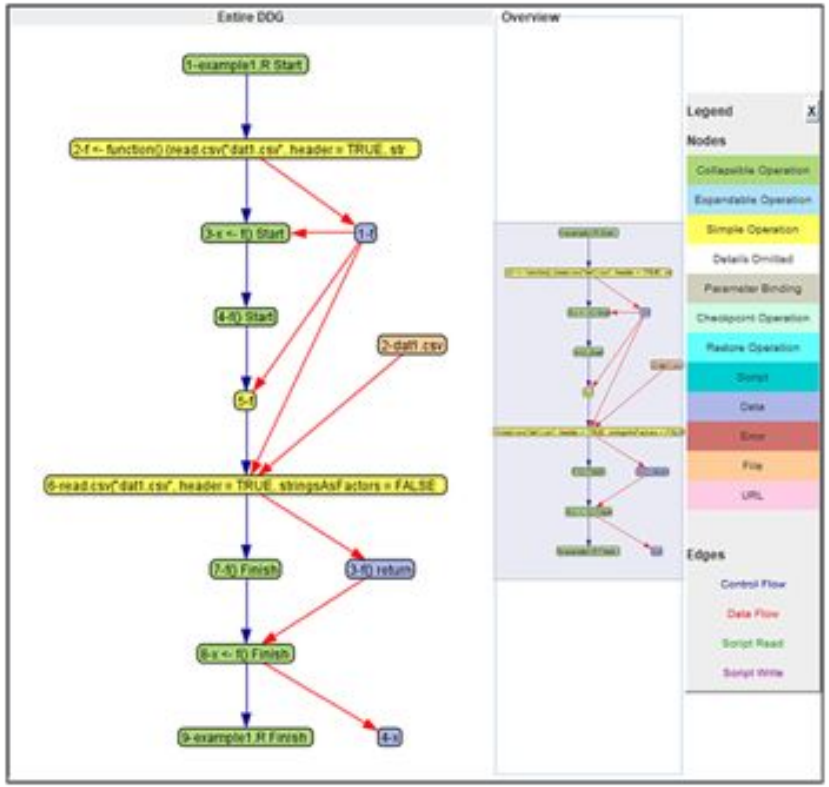
Figure 5. Provenance collection in notebook using noWorkflow

Pimental et al. 2015. Collecting and Analyzing Provenance on Interactive Notebooks: when IPython meets noWorkflow

Provenance Visualization

- If you haven’t noticed yet, the canonical representation of provenance are network graphs — node-link diagrams
- Great for showing structure and relationships (contingent on layout algo)
  - Often this is what you want
- But of course some problems follow….

Issue: Scale

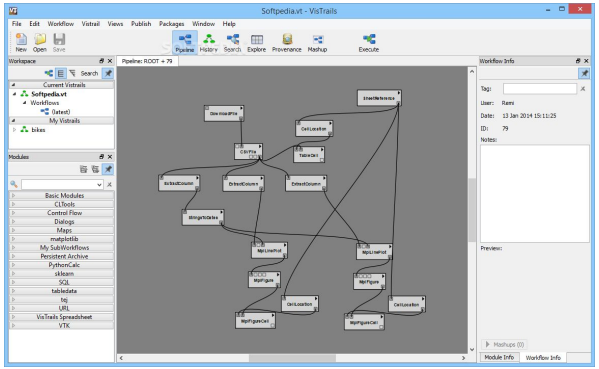


Two Line R Script Provenance Graph

12 Nodes
8 Edges

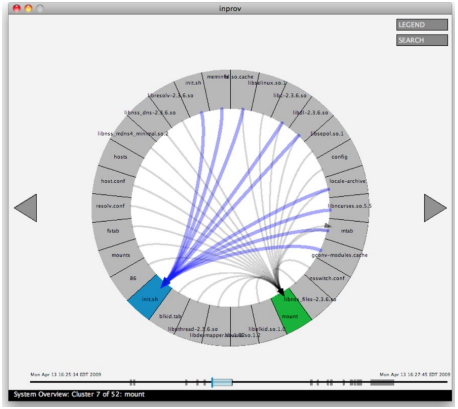
Issue: Assessment Robustness of Visual Encodings

- Only a handful of studies quantitatively measure the usefulness of provenance visualization
  - Corlissen et al. 2010, Macko et al. 2011, Borkin et al. 2013
- Do we know network graphs are the best idiom?
  - Are some layout algorithms preferred in networks? Why? For what tasks?



VisTrails

https://www.softpedia.com/get/Programming/Other-Programming-Files/VisTrails.shtml

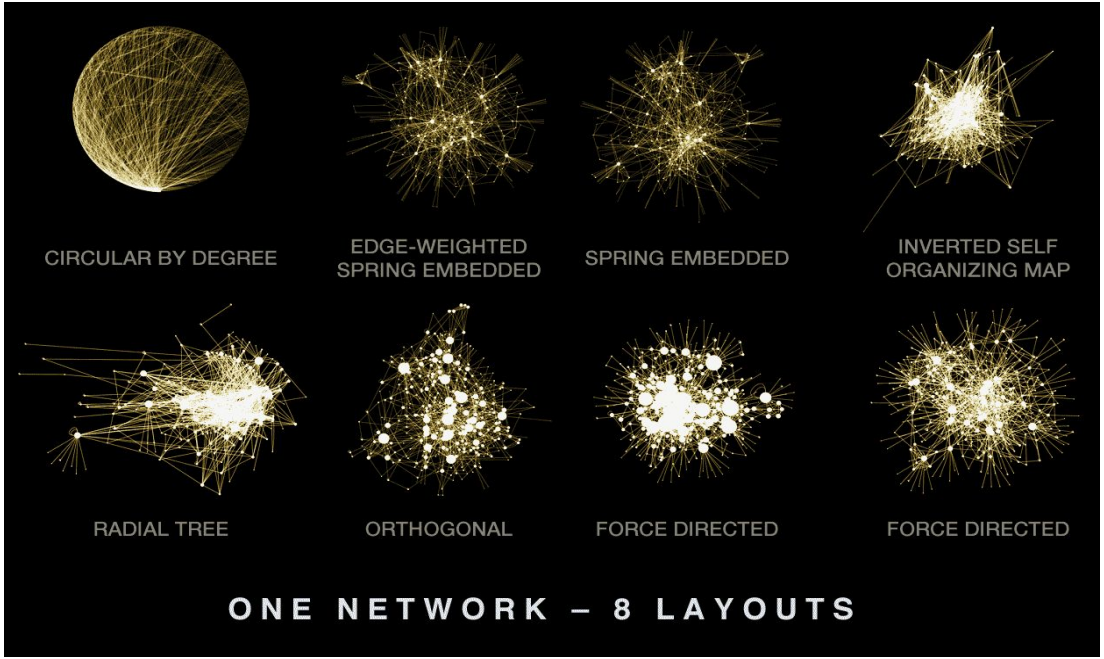


InProv

Borkin et al. 2013

Issue: Network layouts

- Apparent features are by-products of layout algorithm



Krzywinski et al. 2010

http://egweb.bcgsc.ca/img/networklayouts.png

Issue: Novel Usage

- Prov tools might visualize provenance to examine differences
- Often allow users to explore the provenance graph
- Visualize “for the sake” of it
- Focus has been on **collection, not usage** in meaningful ways

- We are interested in the lens of **program comprehension**
  - How do provenance visualizations aid cognition of programs?
  - How can it facilitate the development of mental models of code?

Past Solutions/Existing Approaches

- Only select “relevant nodes”
- Graph summarization nodes — bigger nodes represent clusters
- Allow navigation via semantic zoom in network-graphs
- Different graph layout algorithms based on different metrics
    - Time-based layouts, unsupervised clustering, MCL clustering

Task Abstraction

We conducted brief informal interviews for what our vis should support in addition to corroborating information from related work.

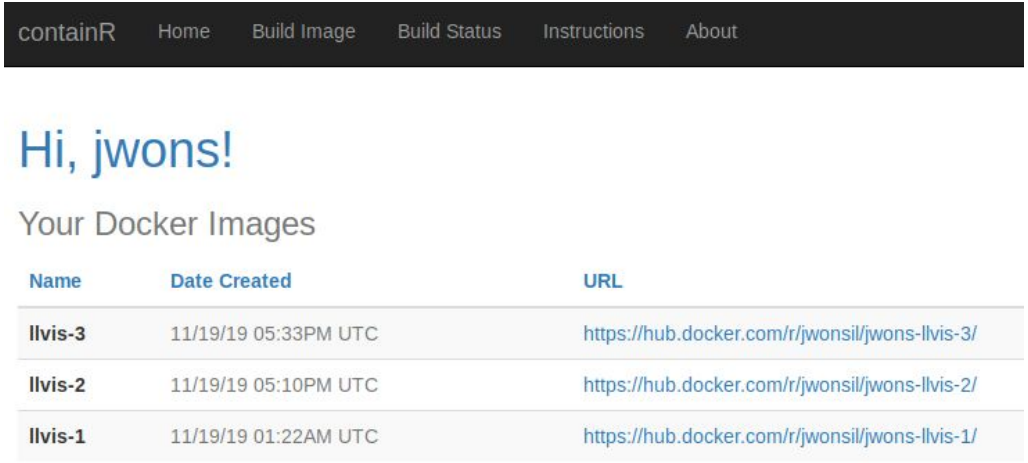
1. Reverse engineering of design patterns. *(Follow crash nodes)*
2. Navigate multiple overviews of the system architecture at various levels of abstraction. *(Multiple views & data abstraction)*
3. Investigate specific contexts. (*Semantic zoom*)
4. Support goal-directed, hypothesis-driven comprehension. For example, the cause of the bug is **x**.
5. View paths or relationships that led to the current focus. *(Graph layout)*
6. Understand syntactic and semantic relationships between variables and functions. *(Graph layout)*

Scenario

- Final-year PhD student has scripts/analyses on a dataset — upload data/scripts to containR
- New graduate student joins the lab & needs to edit code!
- containR and provenance work!
  - The old scripts still run exactly as they did before because they are running in a container.
- L-Vis helps the student learn **how** the old plots were created
  - Allows them to cleanly insert new code into the analysis*
  - Uses the **detail view** of the plot node in order to view all the code-snippets and variables related to generating the plot
- Dynamic what-if analysis
  - To begin building a mental model of the plot generation code, the new student uses the **detail view** and **changes variables** in order to see how the plot changes dynamically*

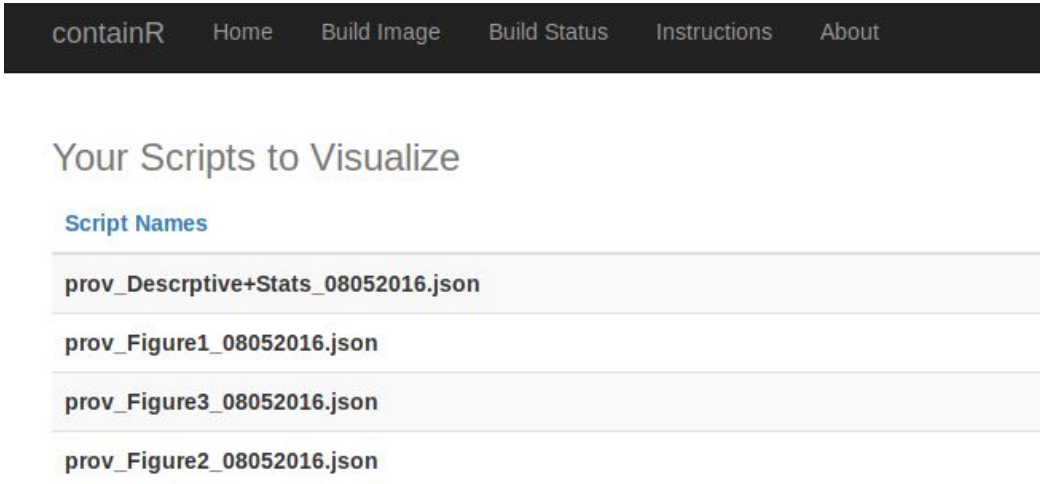
Platform: containR

- Website for increasing scientific repeatability
- ‘Containerize’ R analyses, includes provenance
- Can also be used for archiving other analyses from Dataverse
- Not currently supported. We are updating, extending, and running locally



containR in our scenario

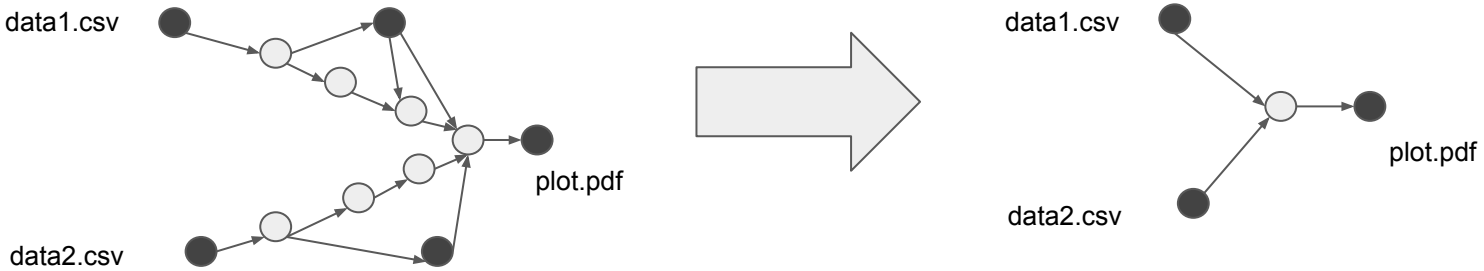
- Users choose container
- Visualization is started based on provenance
- Users can toggle between scripts in their analysis



Our solution

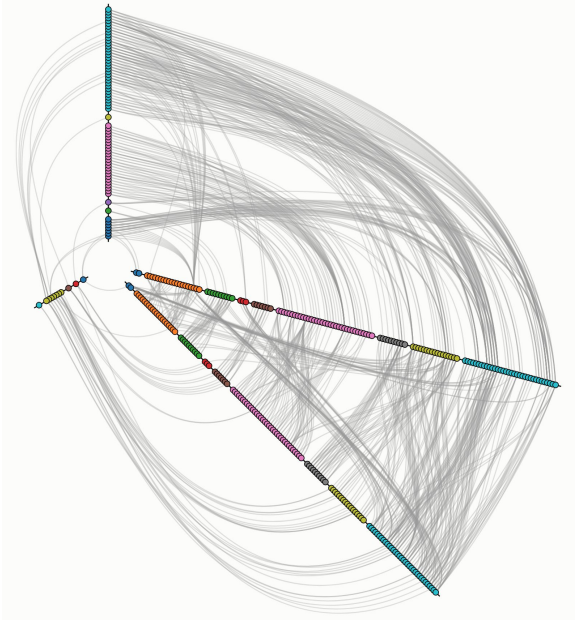
Data abstraction — Filter

- We choose to filter all nodes and explicitly show **crash nodes** - these are procedures where two (or more) different data nodes are both used and a new data node is created.
- Through visualizing crash nodes, the focus of the visualization becomes the path of data through the scripts and how different inputs may interact / depend on each other.

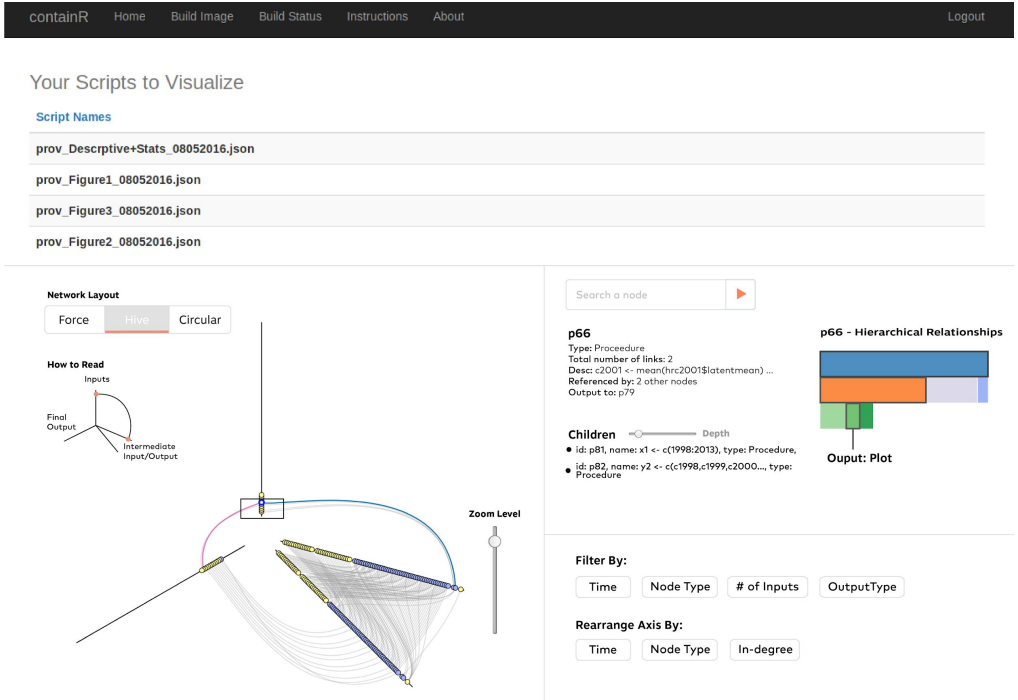


Our solution

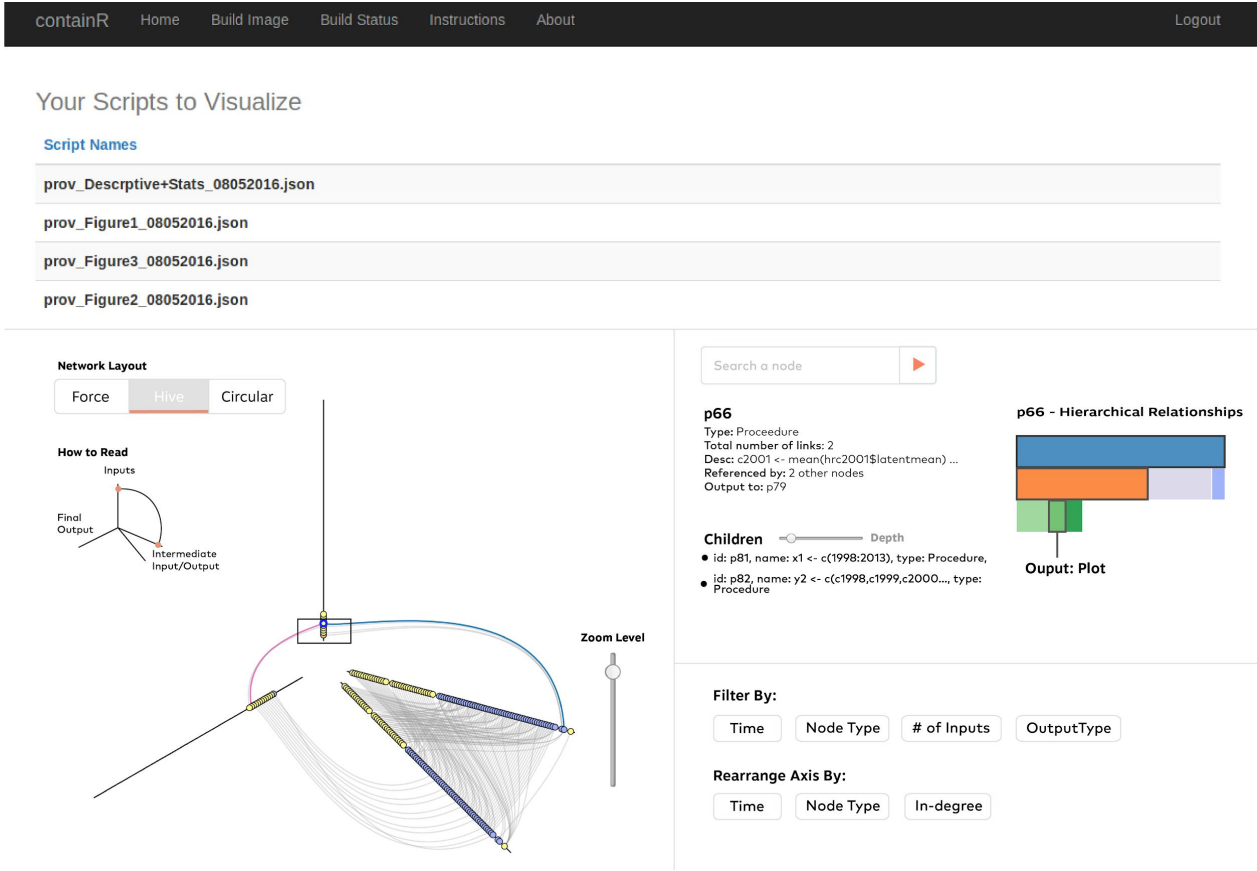
Exploratory visualizations — Graph Layouts



Hive Plots —
Arrange Data on axis meaningfully



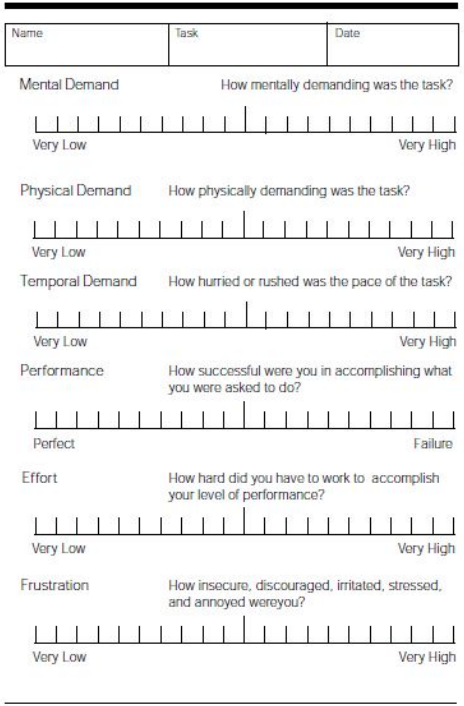
- Allow users to zoom to specific nodes
- Change layouts if used to other network layouts
- Detailed node view
- Will eventually include better references to analysis code
- Perceptually meaningful!



Evaluation

- (informal) User study to compare L-Vis to other LL-Prov tools
  - DDG Explorer versus L-Vis
  - L-Vis + script versus just script
  - In a formal study would probably assess efficiency and accuracy in performance
- NASA-TLX for assessing workload
- Performance evaluation for interactive responsiveness
- Case Studies

Yay NASA!



NASA Task Load Index

Hart and Staveland's NASA Task Load Index (TLX) method assesses work load on five 7-point scales. Increments of high, medium and low estimates for each point result in 21 gradations on the scales.

Name	Task	Date

Mental Demand — How mentally demanding was the task?
Very Low — Very High

Physical Demand — How physically demanding was the task?
Very Low — Very High

Temporal Demand — How hurried or rushed was the pace of the task?
Very Low — Very High

Performance — How successful were you in accomplishing what you were asked to do?
Perfect — Failure

Effort — How hard did you have to work to accomplish your level of performance?
Very Low — Very High

Frustration — How insecure, discouraged, irritated, stressed, and annoyed wereyou?
Very Low — Very High

In the future…

- Finish the prototype and add it to the containR workflow
- Optional filters of data to toggle traditional network layouts
- Design study to derive more formal tasks from users and get iterative feedback
- (more formal) Quantitative study to compare L-Vis to other LL-prov tools

L-Vis: Analyses on my Mind