FACILITATING REPRODUCIBILITY AFTER THE FACT

Fernando Chirigati
ViDA – Visualization and Data Analysis Lab
NYU Polytechnic School of Engineering

NEW YORK UNIVERSITY

Reproducibility? What? Why?

No need for motivation here.

Reproducibility may be hard. Why?

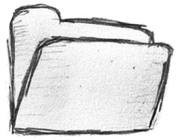
Cultural Change

Potential Lack of Attribution

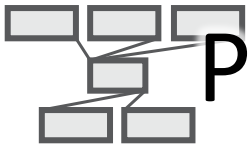
Legal Barriers

Burdensome

Too many dependencies!



DATA



PROVENANCE

ENVIRONMENT

From pixshark.com

Too many different platforms!

From intelerad-insider.com

Too much to do, too little time!

*"authors have complained that the process **requires too much work for the benefit derived**"*

Bonnet et al., SIGMOD Record 2011

***"Insufficient time** is the main reason why scientists do not make their data and experiment available and reproducible."*

Carol Tenopir, Beyond the PDF 2 Conference

***"77%** claim that they do not have **time to document and clean up the code**."*

Victoria Stodden, Survey of the Machine Learning Community – NIPS 2010

***"**It would require **huge amount of effort** to make our code work with the latest versions of these tools."*

Collberg et al., Repeatability and Benefaction in Computer Systems Research, University of Arizona TR 14-04

From galleryhip.com

Planning for Reproducibility

Scientific Workflow Systems (VisTrails, Taverna, Kepler, …)

Virtual Machines and Containers (VirtualBox, Vagrant, Docker, …)

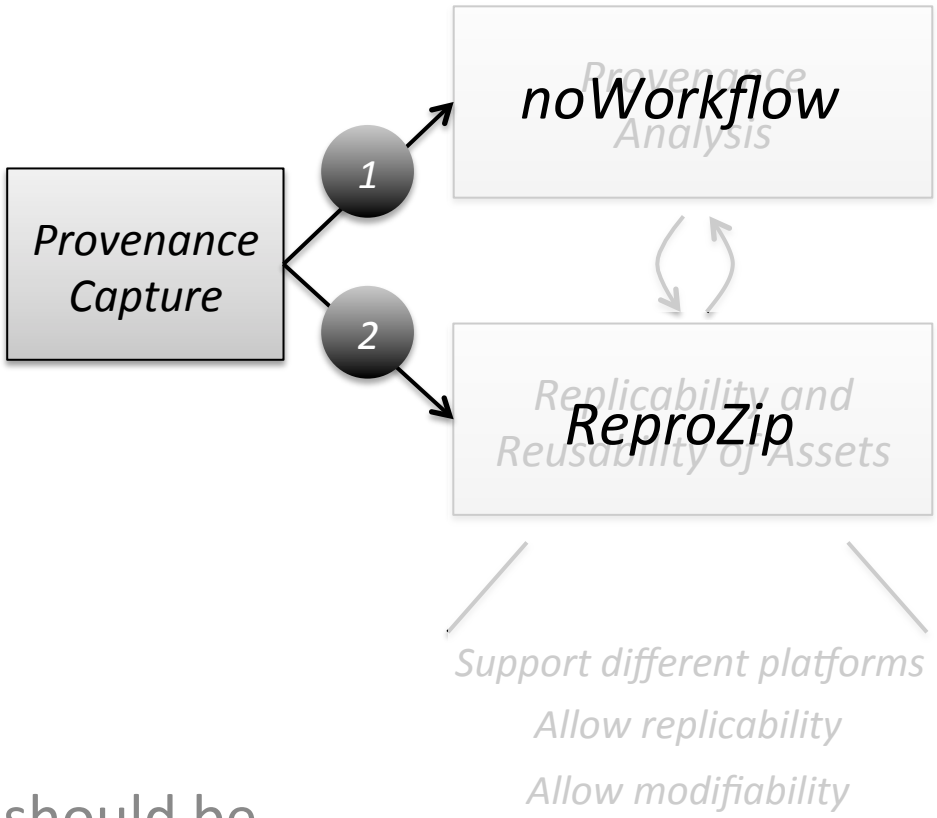
Configuration Management Tools (Chef, Puppet, …)

… and many others !

But what about *reproducibility after the fact*?

Again, time-consuming and error-prone!

Reproducibility After the Fact



Process should be
simple, ***automatic***, and ***non-intrusive***!

noWorkflow

Capturing and Analyzing Provenance of Scripts

Joint work with: João Felipe Pimentel **(UFF)**
Leonardo Murta **(UFF)**
Vanessa Braganholo **(UFF)**
David Koop **(UMass-Dartmouth)**
Juliana Freire **(NYU)**

NEW YORK UNIVERSITY

Instituto de computação

Provenance for Scripts!

Goal: provenance analysis for scripts

Challenge: *transparent* and *non-intrusive* provenance capture

Scientific Workflows – VisTrails, Taverna, … ***intrusive***
OS-Level Tools – ES3, Burrito, and Pass [1,2,3] ***different provenance level***
Provenance API for Python [4] ***intrusive***
VCR [5] ***intrusive***
Sumatra [6] ***intrusive***
ProvenanceCurious [7] ***non-transparent***
Tariq et al. – LLVM compiler framework [8] ***no dynamic information***

noWorkflow

Transparently captures the *provenance* of a script
- *Language-independent approach*
- *Language-dependent solution (Python)*

Is non-intrusive: no need for user-defined annotations, instrumented environment, or other requirements

Provides different methods for provenance analysis
- *Visualization of Trials*
- *Evolution Graph*
- *Diff Analysis*
- *Querying*
- *IPython Notebook*

How does noWorkflow work?

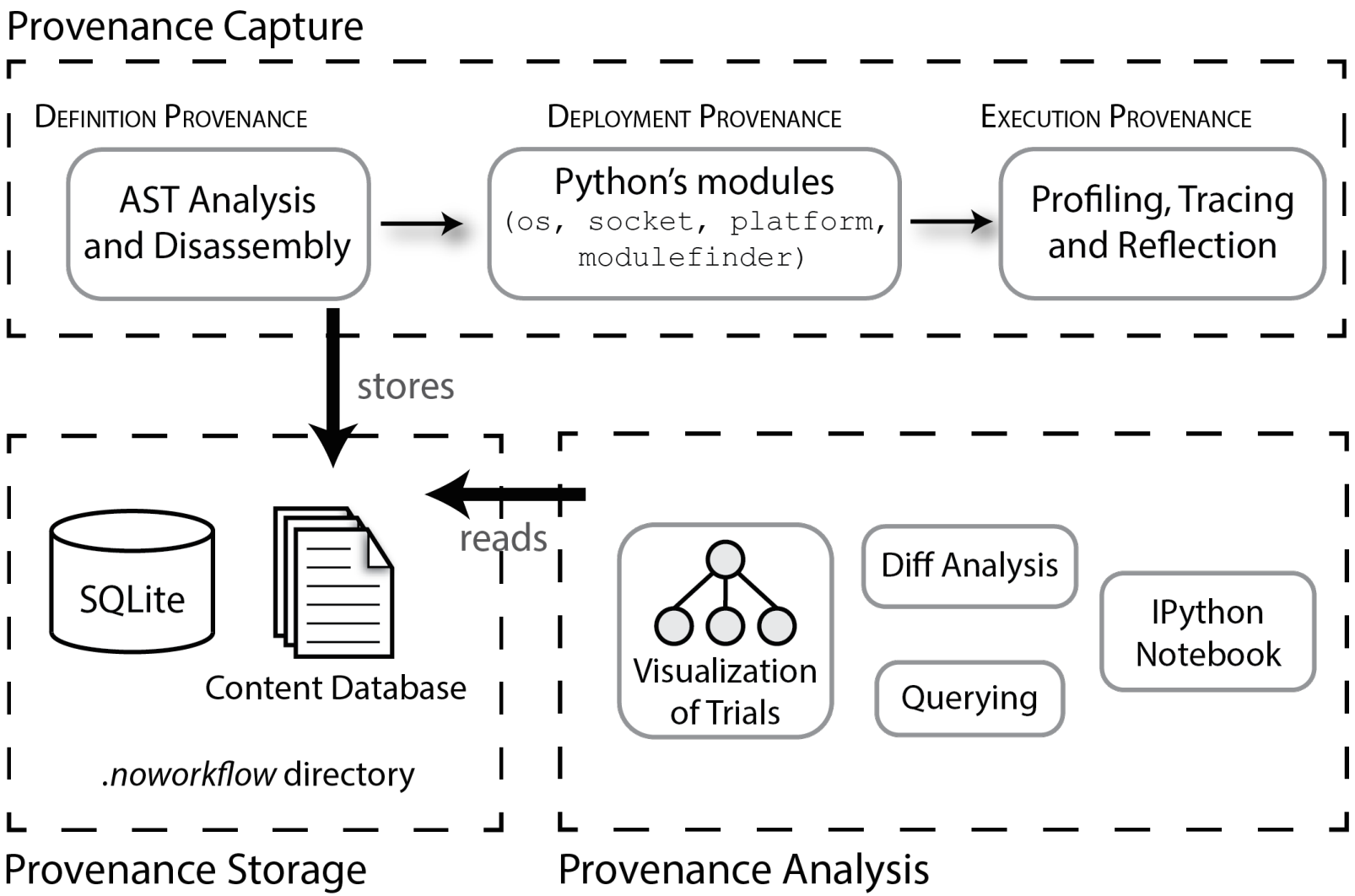
Instead of running

```
$ python my_script.py
```

users run

```
$ now run my_script.py
```

That’s it.


Provenance Capture
Definition Provenance
AST Analysis and Disassembly
Deployment Provenance
Python's modules
(os, socket, platform, modulefinder)
Execution Provenance
Profiling, Tracing and Reflection
stores
reads
SQLite
Content Database
.noworkflow directory
Visualization of Trials
Diff Analysis
Querying
IPython Notebook
Provenance Storage
Provenance Analysis

Provenance Analysis

```
$ now list
[now] trials available in the provenance store:
  Trial 1: generateClassifier.py
           with code hash 6d4ca1fe98be349b9c39dcf91a71e4df9c3af1fb
           ran from 2015-05-19 03:41:25.729536 to None
  Trial 2: generateClassifier.py
           with code hash 6d4ca1fe98be349b9c39dcf91a71e4df9c3af1fb
           ran from 2015-05-19 07:12:09.224764 to 2015-05-19
07:33:59.112497
  Trial 3: performRecognition.py
           with code hash 3d64107ce1efa86336d699bea1d67d0dd28f7cf5
           ran from 2015-05-19 07:36:03.621202 to 2015-05-19
07:36:08.242297
  Trial 4: performRecognition.py
           with code hash 2f3e9a35b72f5194fa0692f8fbb5da22cb229096
           ran from 2015-05-19 07:52:18.722780 to 2015-05-19
07:52:20.822903
```

Provenance Analysis

```
$ now show -f 2
[now] trial information:
  Id: 2
  Inherited Id: 1
  Script: generateClassifier.py
  Code hash: 6d4ca1fe98be349b9c39dcf91a71e4df9c3af1fb
  Start: 2015-05-19 07:12:09.224764
  Finish: 2015-05-19 07:33:59.112497
[now] this trial accessed the following files:
  Name: digits_cls.pkl
  Mode: wb
  Buffering: default
  Content hash before: a69ddffb759ef5708f53cdf55c91883405a256fa
  Content hash after: 562b9c739d970134400827cc357aa28bd24f6859
  Timestamp: 2015-05-19 07:33:59.087665
  Function: dump ->  ... -> open
...
```

Provenance Analysis

```
$ now show –f 3
[now] trial information:
  Id: 3
  Inherited Id: 1
  Script: performRecognition.py
  Code hash: 3d64107ce1efa86336d699bea1d67d0dd28f7cf5
  Start: 2015-05-19 07:36:03.621202
  Finish: 2015-05-19 07:36:08.242297
[now] this trial accessed the following files:
  Name: digits_cls.pkl
  Mode: rb
  Buffering: default
  Content hash before: 562b9c739d970134400827cc357aa28bd24f6859
  Content hash after: 562b9c739d970134400827cc357aa28bd24f6859
  Timestamp: 2015-05-19 07:36:07.405609
  Function: load ->  ... -> open
  ...
```

Provenance Analysis

```
$ now diff 2 4
[now] trial diff:
  finish changed from 2015-05-19 07:33:59.112497 to 2015-05-19
07:52:20.822903
  parent_id changed from 1 to 3
  script changed from generateClassifier.py to performRecognition.py
  code_hash changed from 6d4ca1fe98be349b9c39dcf91a71e4df9c3af1fb to
2f3e9a35b72f5194fa0692f8fbb5da22cb229096
  start changed from 2015-05-19 07:12:09.224764 to 2015-05-19
07:52:18.722780
  duration changed from 1309887733 to 2100123
```

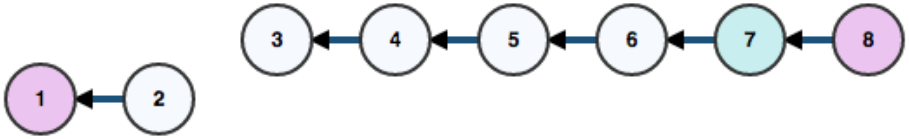
Provenance Analysis

```
$ now export 4
%
% FACT: activation(trial_id, id, name, start, finish,
caller_activation_id).
%

:- dynamic(activation/6).
activation(4, 210224, '/home/fchirigati/digitRecognition/
performRecognition.py', 1432021938.836802, 1432021940.822519, nil).
activation(4, 210225, '/usr/local/lib/python2.7/dist-packages/numpy/
__init__.py', 1432021938.933919, 1432021939.084865, 210224).
activation(4, 210226, '/usr/local/lib/python2.7/dist-packages/
sklearn/activation(4, 210230, 'load', 1432021940.480701,
1432021940.611564, 210224).
activation(4, 210231, 'imread', 1432021940.611702, 1432021940.650233,
210224).
activation(4, 210232, 'cvtColor', 1432021940.650314,
1432021940.657686, 210224).
```

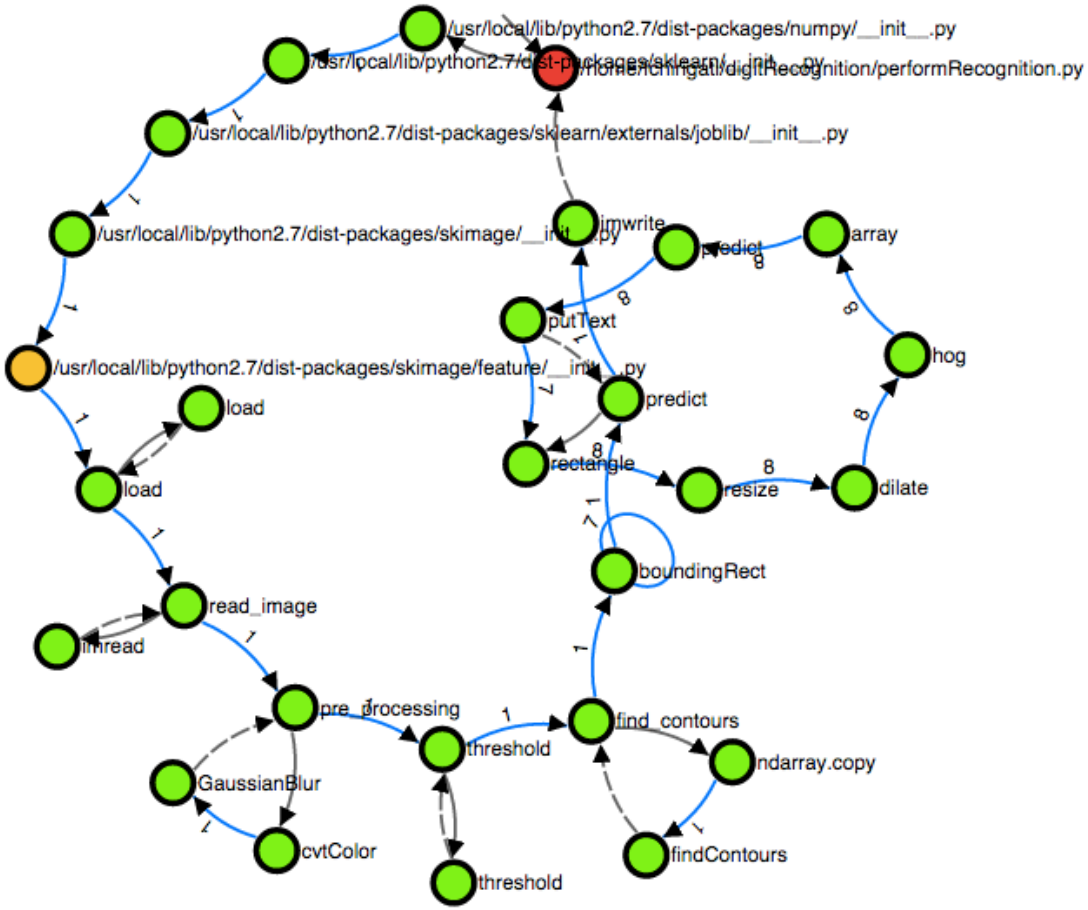
All Scripts
All Statuses
Reload

Ctrl-click to diff trials



Exact Match | Combined

Ctrl-click to toggle nodes



Trial 7

2546d07cc916831cc63d44a7aa48ef534f55905c

Script: performRecognition.py
Start: 2015-05-19 08:13:21.983213
Finish: 2015-05-19 08:13:24.216981

Environment

PYTHON_IMPLEMENTATION = CPython
PYTHON_VERSION = 2.7.6
OS_NAME = Linux
PWD = /home/fchirigati/digitRecognition
PID = 14841
HOSTNAME = fchirigati-ubuntu
ARCH = 64bit
PROCESSOR = x86_64

File Accesses

/usr/local/lib/python2.7/dist-packages/skimage/data/orb_descriptor_positions.txt
default U
2015-05-19 08:13:23.765053
3fc36db34fc1354f357a7ab4e767bda394fab826
3fc36db34fc1354f357a7ab4e767bda394fab826
/usr/local/lib/python2.7/dist-packages/skimage/feature/__init__.py -> ... -> open

digits_cls.pkl
default rb
2015-05-19 08:13:23.797647
562b9c739d970134400827cc357aa28bd24f6859
562b9c739d970134400827cc357aa28bd24f6859
load -> load -> ... -> open

/usr/local/lib/python2.7/dist-packages/noworkflow/now/prov_execution/profiler.py
default rU
2015-05-19 08:13:23.996066
3e82ed0c8ed209c4c452c63facc51dd1060be7c0
3e82ed0c8ed209c4c452c63facc51dd1060be7c0

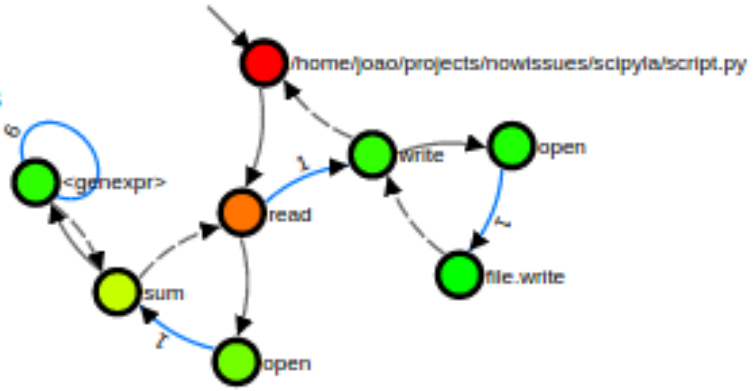
Provenance Analysis

In [1]:
```
%load_ext noworkflow
%now_set_default graph_width=430 graph_height=150
nip = %now_ip
```

In [2]:
```
dry = 0
trial = %now_run --name ipython_script script.py $dry
trial
```

Out[2]:

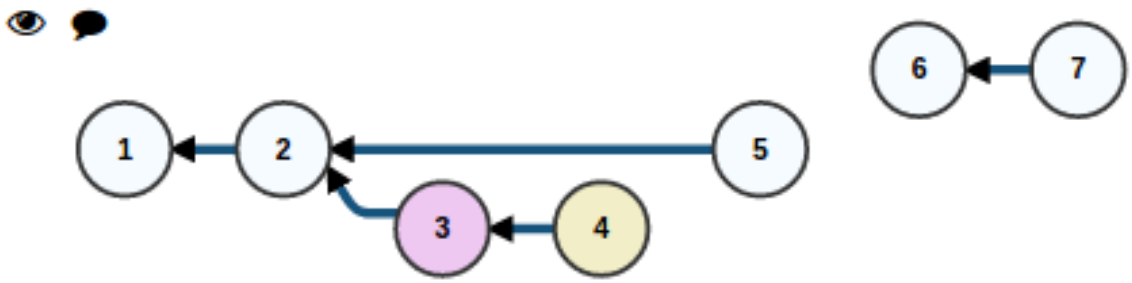
Trial 6. Ctrl-click to toggle nodes



Provenance Analysis

```
In [7]: nip.History()
```

Out[7]:



Provenance Analysis

In [8]:
```
%%now_prolog --result result {trial.id}
duration({trial.id}, read, X)
```

In [9]:
```
for match in result:
    print(match['X'])
```

```
0.00296902656555
```

In [10]:
```
%%now_sql
SELECT DISTINCT script FROM trial
```

Out[10]:

script
script.py
ipython_script

Future Work

Caching capabilities

Automatic identification of flaws in the execution

Connection between OS- and script-level provenance

Replicability feature

Try it!

Website: *https://github.com/gems-uff/noworkflow*

L. Murta, V. Braganholo, F. Chirigati, D. Koop, and J. Freire: *noWorkflow: Capturing and Analyzing Provenance of Scripts*. In Provenance and Annotation of Data and Processes, vol. 8628, Lecture Notes in Computer Science (LNCS), pp. 71-83, Springer International Publishing, 2015

Send your feedback and interesting use cases!

ReproZip

Packing Experiments for Reproducibility

Joint work with: Rémi Rampin
Dennis Shasha
Juliana Freire

NEW YORK UNIVERSITY

Creating Executable Packages!

Goal: creation of executable packages for reproducing experiments

Challenges: *transparent*, *non-intrusive,* and *language-independent* provenance capture; support to multiple platforms

CDE – Code, Data, and Environment [9]
PTU – Provenance-To-Use [10]
CARE – Comprehensive Archiver for Reproducible Execution [11]

Linux-only

limited interfaces for varying the experiment

ReproZip

Automatically and systematically captures the *provenance* of an existing experiment

Language-independent approach and solution

Creates a self-contained *reproducible package* from captured provenance

Extracts package in another environment, *independent* of the operating system

Provides *easy-to-use* interfaces for replicating and varying the original configuration of the experiment

How does ReproZip work?

ReproZip is a packaging tool

From reputablemoving.com

From wykop.pl

Packing Experiments

AUTHORS

Computational Environment ***E*** (Linux)

Experiment

Packing Experiments

AUTHORS

Computational Environment ***E*** (Linux)

Experiment

Executing

reprozip

Packing Experiments

Computational Environment ***E*** (Linux)

Experiment

Executing

reprozip

Capturing Provenance

ptrace + SQLite

Experiment Provenance

Data

Input files, output files, parameters, …

Workflow

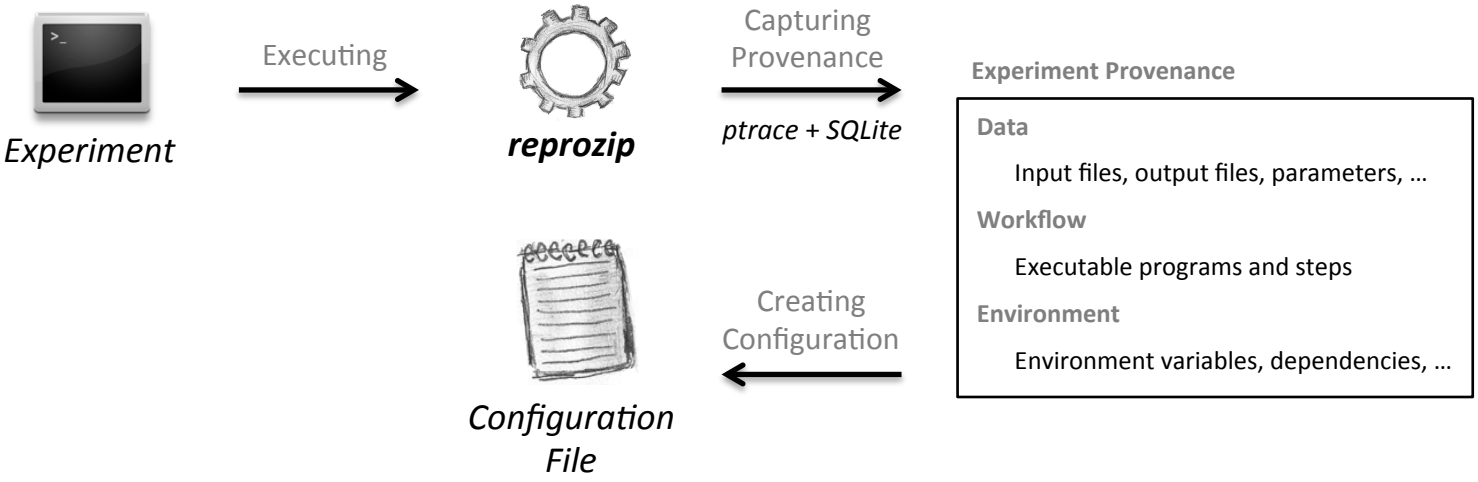
Executable programs and steps

Environment

Environment variables, dependencies, …

Packing Experiments

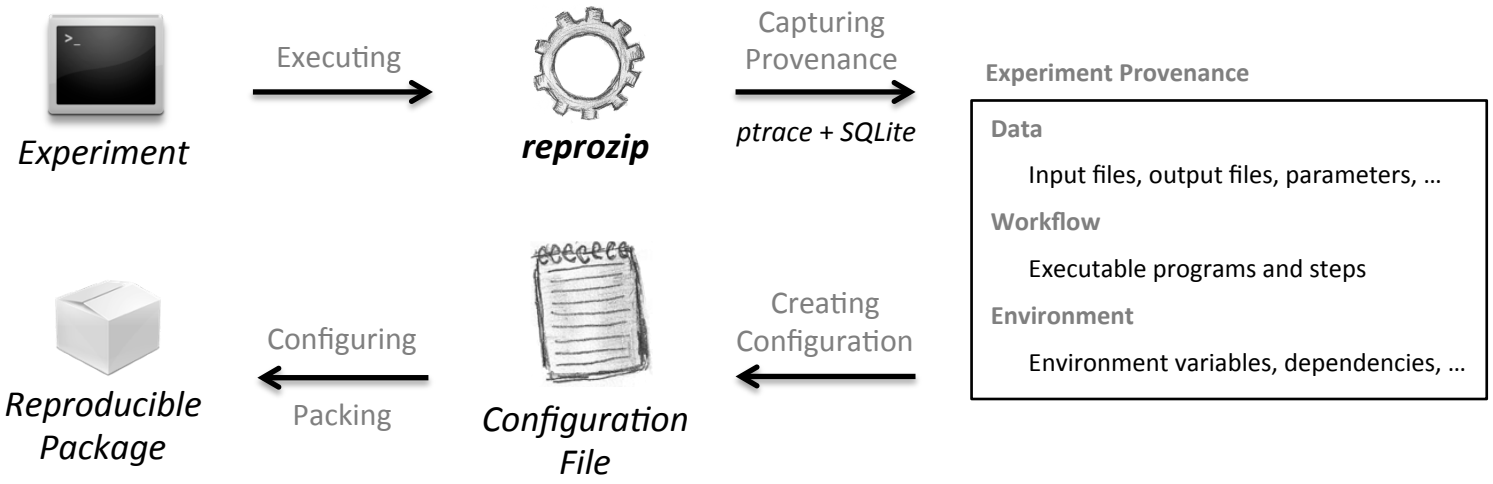
Computational Environment ***E*** (Linux)



Packing Experiments

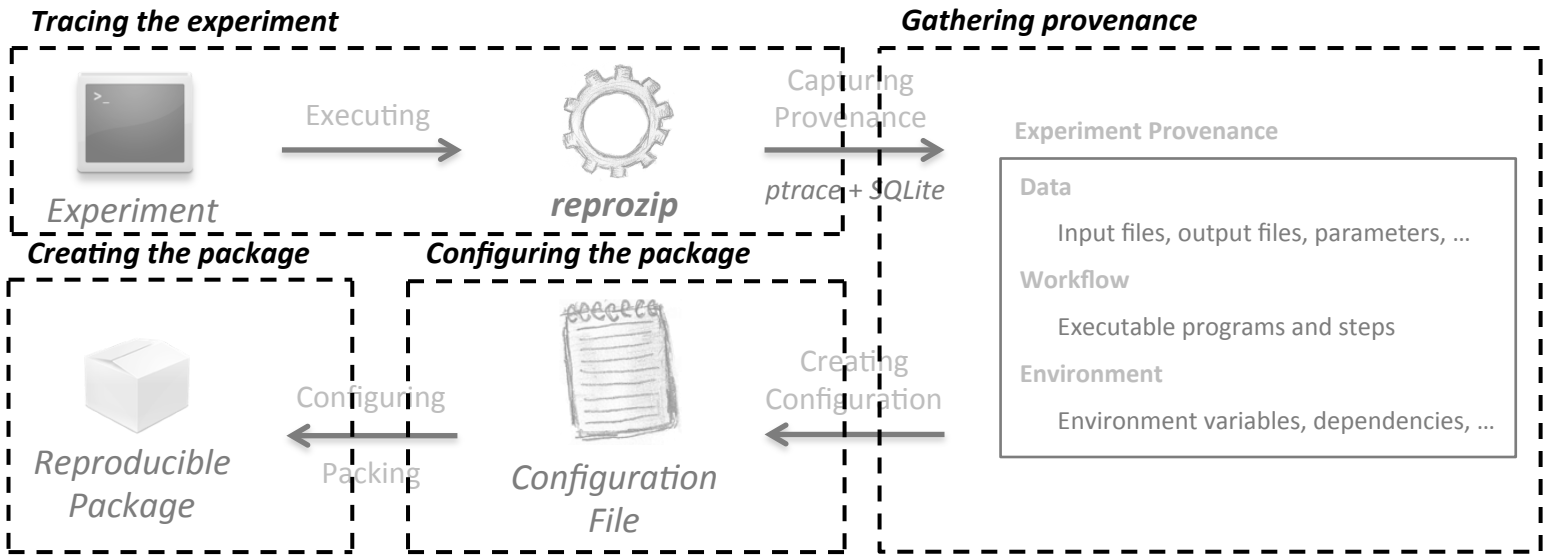
AUTHORS

Computational Environment ***E*** (Linux)

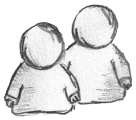


Packing Experiments

Computational Environment ***E*** (Linux)



Unpacking Experiments

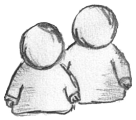


REVIEWERS
READERS

Computational Environment ***E'*** (potentially different than ***E***)

Reproducible Package

Unpacking Experiments



REVIEWERS
READERS

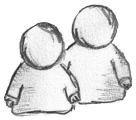
Computational Environment ***E'*** (potentially different than ***E***)

Reproducible Package

Extracting

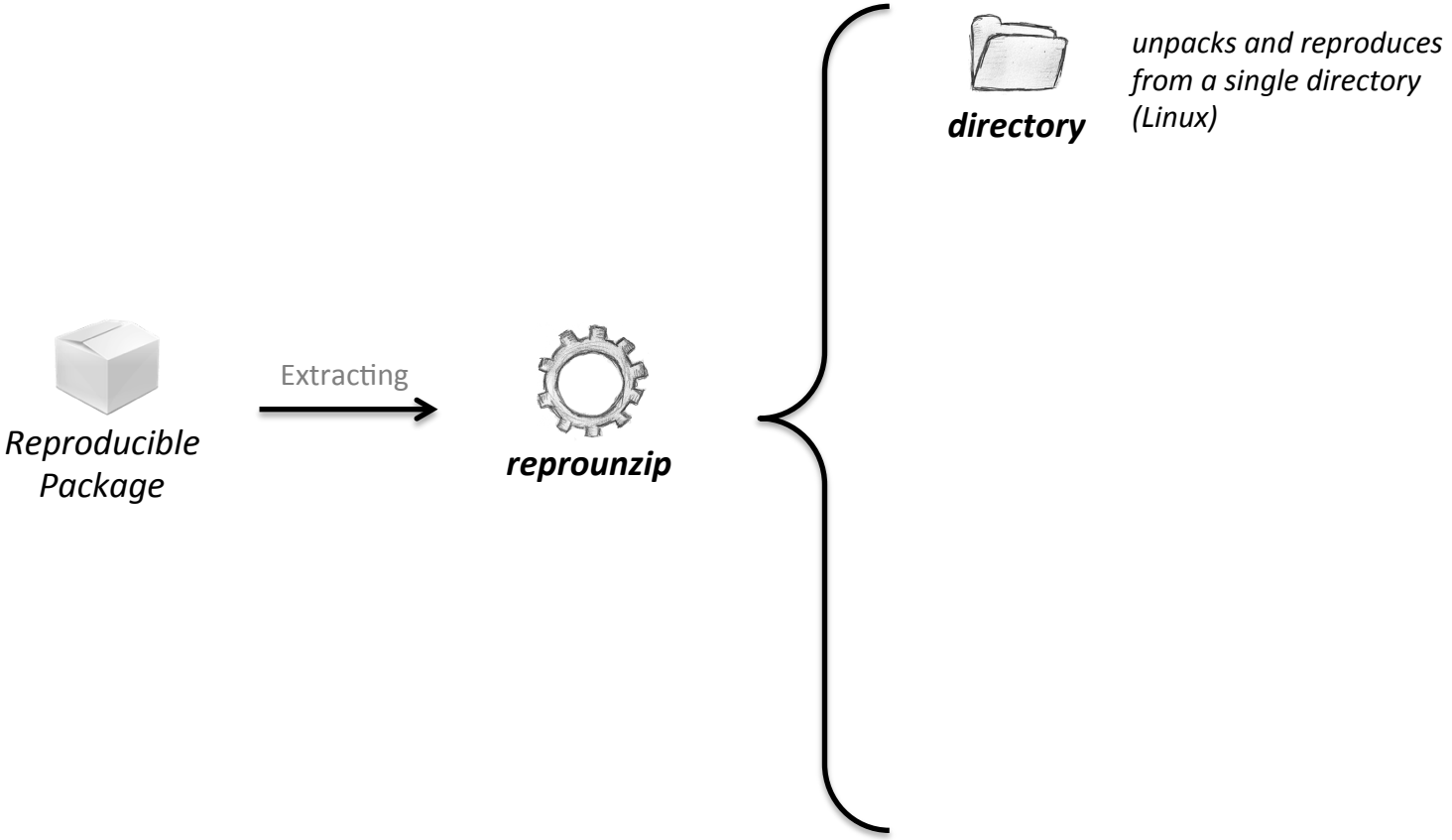
reprounzip

Unpacking Experiments

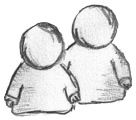


REVIEWERS
READERS

Computational Environment ***E'*** (potentially different than ***E***)



Unpacking Experiments



REVIEWERS
READERS

Computational Environment ***E'*** (potentially different than ***E***)

Reproducible Package

Extracting

reprounzip

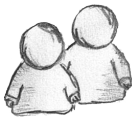
directory

unpacks and reproduces from a single directory (Linux)

chroot

unpacks in a single directory and builds a full system environment (Linux)

Unpacking Experiments



REVIEWERS
READERS

Computational Environment ***E'*** (potentially different than ***E***)

Reproducible Package

Extracting

reprounzip

directory

unpacks and reproduces from a single directory (Linux)

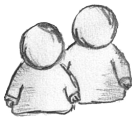
chroot

unpacks in a single directory and builds a full system environment (Linux)

vagrant

unpacks in a virtual machine using Vagrant (Linux, Mac OS X, Windows)

Unpacking Experiments



REVIEWERS
READERS

Computational Environment ***E'*** (potentially different than ***E***)

Reproducible Package

Extracting

reprounzip

directory

unpacks and reproduces from a single directory (Linux)

chroot

unpacks in a single directory and builds a full system environment (Linux)

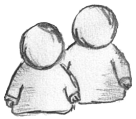
vagrant

unpacks in a virtual machine using Vagrant (Linux, Mac OS X, Windows)

docker

unpacks in a Docker container (Linux, Mac OS X, Windows)

Unpacking Experiments



REVIEWERS
READERS

Inspecting a reproducible package:

info, *showfiles*, *graph*

Running an unpacker:

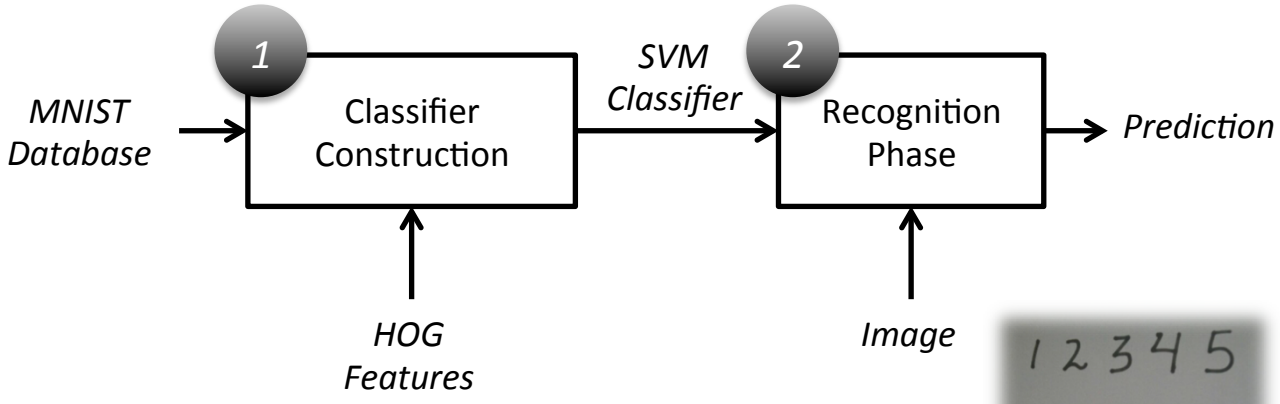
setup, *run*, *destroy*, *upload*, *download*

Natively installing required software dependencies:

installpkgs

Example

Predicting the Value of a Handwriting Digit from an Image



Main Dependencies

- *scikit-learn*
- *scikit-image*
- *opencv*

Demo

News!

ReproZip …

… has been adopted in the Bonneau Lab (NYU)

http://bonneaulab.bio.nyu.edu/

… will be used by the ACM SIGMOD 2015 Reproducibility Review

http://db-reproducibility.seas.harvard.edu/

… will be used by the Information Systems journal (Reproducibility Section)

http://www.journals.elsevier.com/information-systems/

… is being used for enabling automatic version upgrades of complex systems (work submitted to TaPP'15 by Dennis Shasha and colleagues)

Wrap-Up: Main Advantages

Automatically **captures** experimental steps

Preserves experiment in a package (**longevity**)

Allows configuration of what should (not) be included in the package

Allows reproducibility of graphical tools

Allows experiment to be reproduced using the same configuration (**replicability**)

Allows users to change command line parameters and input files (**modifiability**)

Experiments can be ported from Linux to Mac OS X and Windows (**portability**)

Wrap-Up: Limitations

Only packs experiments in Linux distros (yet…)

Only detects software packages in Debian-based environments (yet…)

Does not guarantee reproducibility of distributed applications

Does not allow reproducibility of *non-deterministic* processes

Does not save *state*

Future Work

Creating reproducible packages in Mac OS X

Identifying software packages in other systems

Proprietary software

Improvements in the provenance graph generation

Creation of dataflows (increases *modifiability*) – *ongoing work*

Reproducibility of distributed applications – *ongoing work*

Integration with other tools (noWorkflow, Liquid Version Climber, …)

Try it!

Website: *http://vida-nyu.github.io/reprozip/*

GitHub: *https://github.com/ViDA-NYU/reprozip*

Mailing lists: *reprozip-users@vgc.poly.edu*

reprozip-dev@vgc.poly.edu

F. Chirigati, D. Shasha, and J. Freire: *Packing Experiments for Sharing and Publication*. In Proceedings of the 2013 International Conference on Management of Data (SIGMOD), pp. 977-980, 2013

F. Chirigati, D. Shasha, and J. Freire: *ReproZip: Using Provenance to Support Computational Reproducibility*. In Proceedings of the 5th USENIX conference on Theory and Practice of Provenance (TaPP), 2013

Send your feedback and interesting use cases!

Thanks!

Questions?

NEW YORK UNIVERSITY

References

[1] Frew, J., Metzger, D., Slaughter, P.: Automatic capture and reconstruction of computational provenance. Concurrency and Computation: Practice and Experience 20(5), 485–496 (2008)

[2] Guo, P.J., Seltzer, M.: BURRITO: Wrapping Your Lab Notebook in Computational Infrastructure. In: TaPP. pp. 7–7 (2012)

[3] Muniswamy-Reddy, K.K., Holland, D.A., Braun, U., Seltzer, M.: Provenance-aware storage systems. In: USENIX. pp. 4–4 (2006)

[4] Bochner, C., Gude, R., Schreiber, A.: A Python Library for Provenance Recording and Querying. In: IPAW. pp. 229–240 (2008)

[5] Gavish, M., Donoho, D.: A Universal Identifier for Computational Results. Procedia Computer Science 4, 637–647 (2011)

[6] Davison, A.: Automated Capture of Experiment Context for Easier Reproducibility in Computational Research. Computing in Science Engineering 14(4), 48–56 (2012)

[7] Huq, M.R., Apers, P.M.G., Wombacher, A.: ProvenanceCurious: a tool to infer data provenance from scripts. In: EDBT. pp. 765–768 (2013)

[8] Tariq, D., Ali, M., Gehani, A.: Towards automated collection of application-level data provenance. In: TaPP. pp. 1–5 (2012)

[9] Guo, P.: CDE: run any Linux application on-demand without installation. In Proceedings of LISA'11. USENIX Association, Berkeley, CA, USA, 2-2 (2011)

[10] Pham, Q., Malik, T., and Foster, I.: Using provenance for repeatability. In Proceedings of TaPP '13. USENIX Association, Berkeley, CA, USA, Article 2 (2013)

[11] Janin, Y., Vincent, C., and Duraffort, R.: CARE, the comprehensive archiver for reproducible execution. In Proceedings of TRUST '14. ACM, New York, NY, USA, Article 1 (2014)