Provenance Gathering from Scripts: Challenges and Opportunities

Leonardo Gresta Paulino Murta
leomurta@ic.uff.br

Instituto de Computação
Laboratório de Engenharia de Sistemas e Informação

IC / UFF

Instituto de Computação / Universidade Federal Fluminense

IC/UFF

UFF

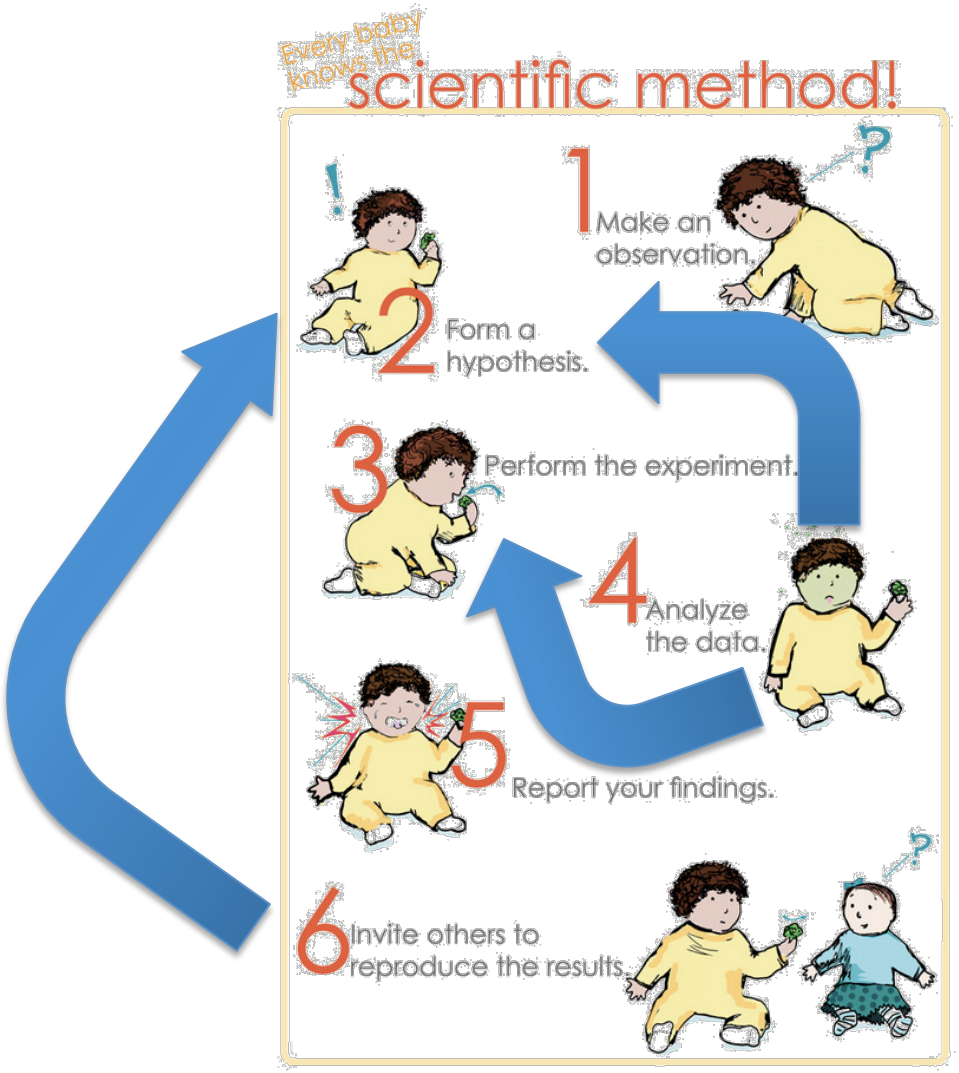
- 3.180 professors
- 66.186 students

IC

- 65 professors
- 90 PhD students
- 110 MS students
- 500 BS (CC) students
- 500 BS (IS) students
- 2.000 TSC students

It all starts with the Scientific Method

In fact, it is not linear



© 2017 Nerdy Baby

In the beginning

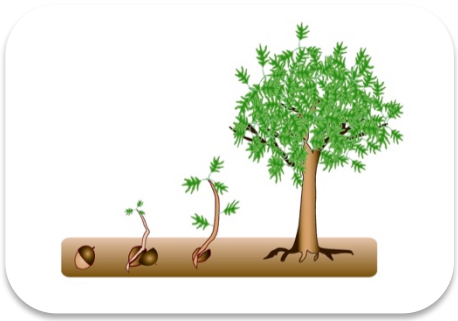
In-vivo

In-vitro

However in-vivo and in-vitro experiments are...

Costly

Risky



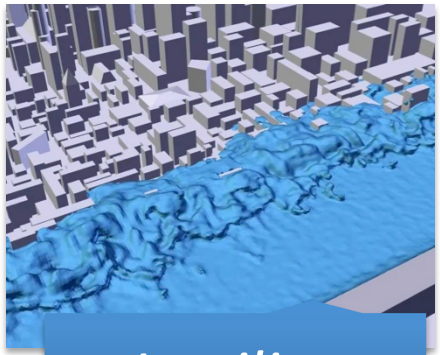
Slow

From in-vivo to *in-silico*

In-vivo

In-vitro

In-virtuo



In-silico

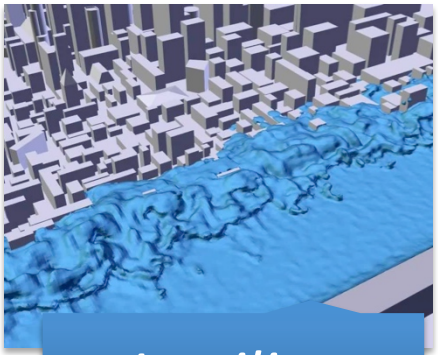
Travassos and Barros "Contributions of in virtuo and in silico experiments for the future of empirical studies in software engineering." WSESE 2003

Dealing with produced data

In-vivo

In-vitro

In-virtuo

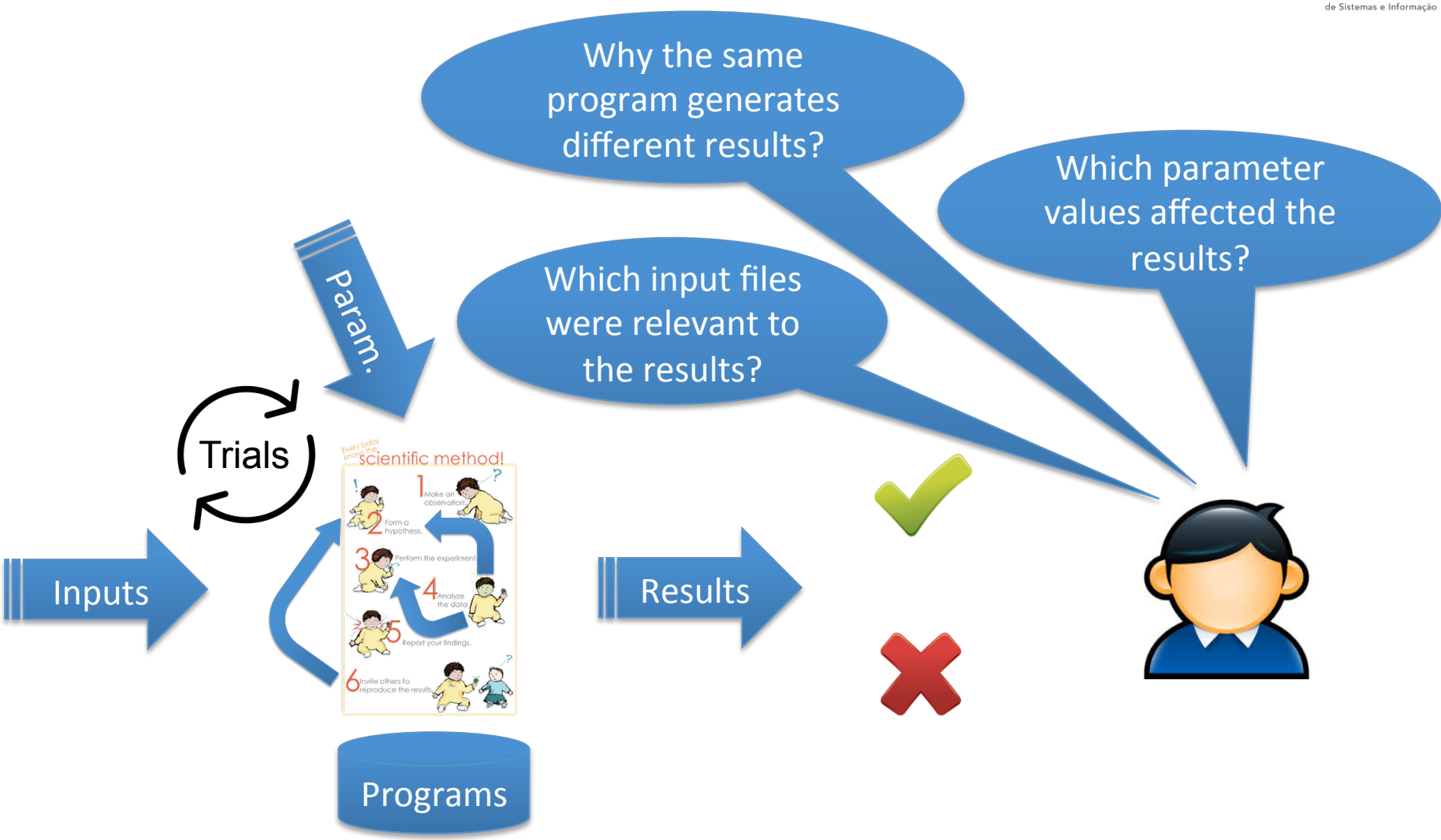


In-silico

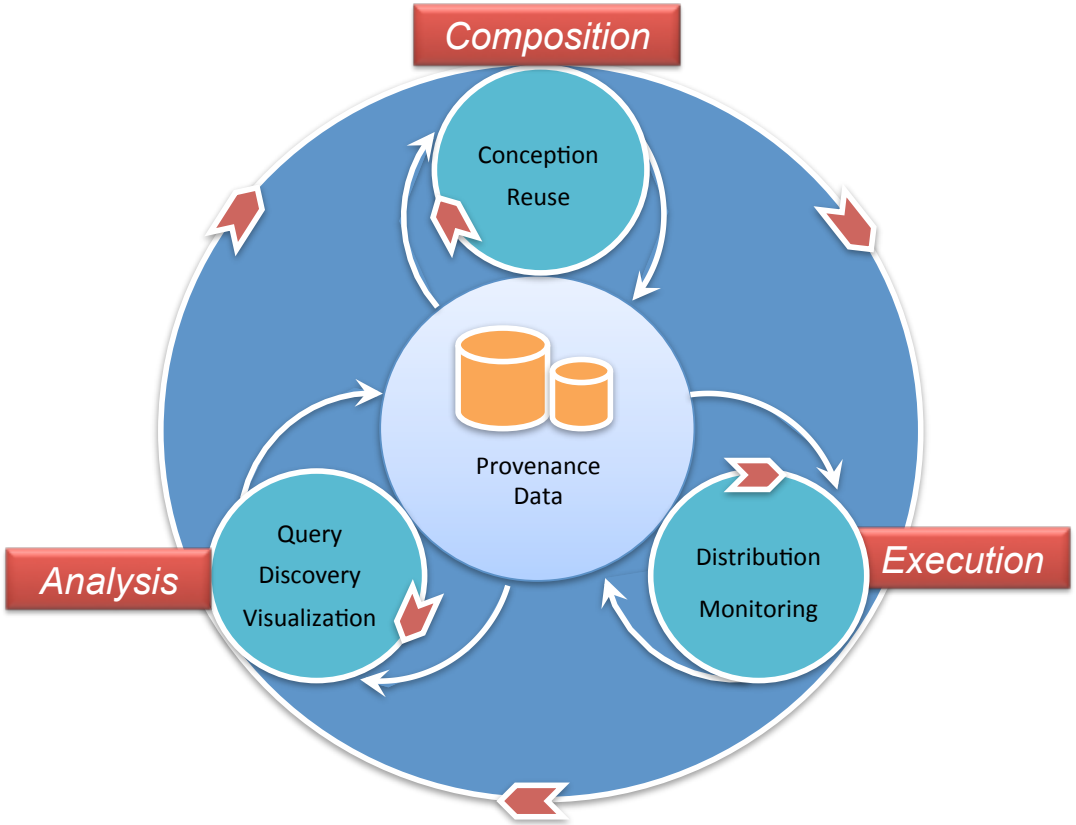
Data volume

Travassos and Barros "Contributions of in virtuo and in silico experiments for the future of empirical studies in software engineering." WSESE 2003

How to reason from the data?



Provenance is the key!



Mattoso et al. "Towards supporting the life cycle of large scale scientific experiments." IJBPIM 5(1) 2010

Provenance

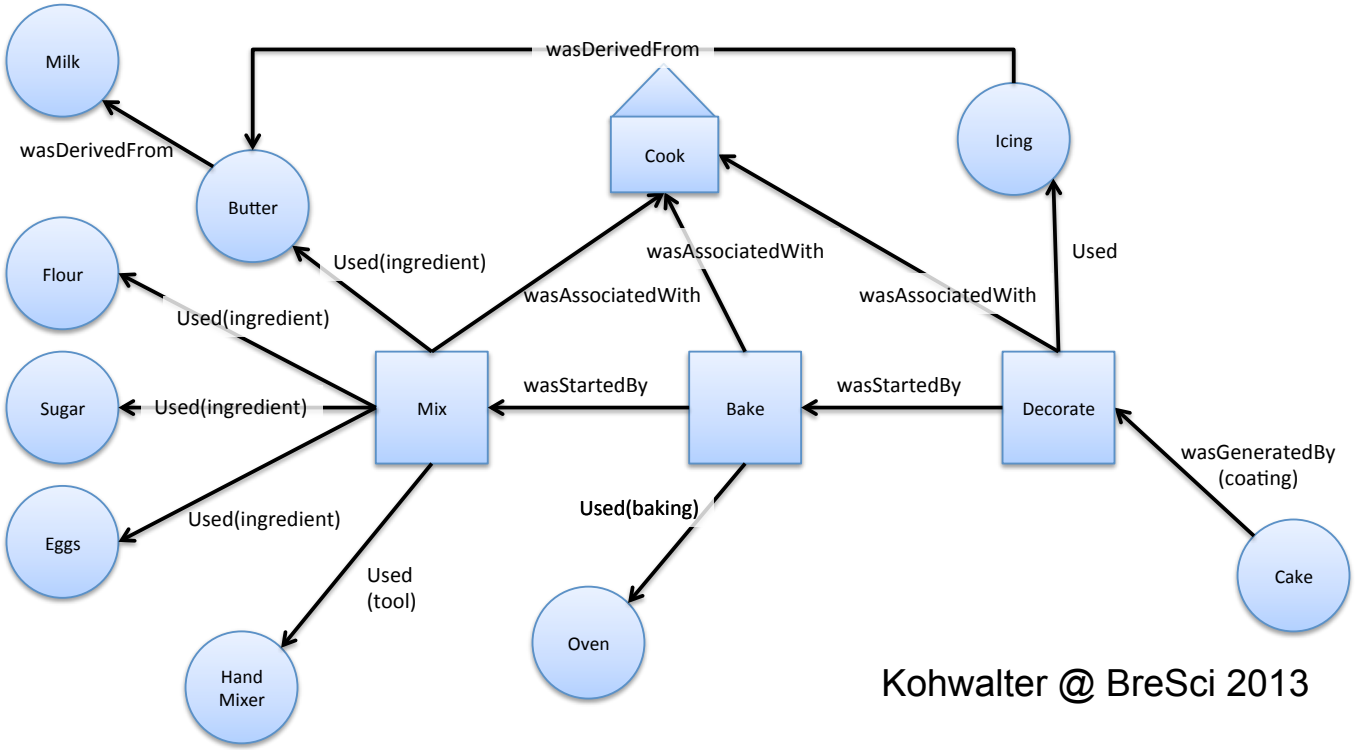
"A **record of ownership** of a **work of art** or an **antique**, used as a guide to **authenticity or quality**."

New Oxford American Dictionary.

"Provenance is defined as a **record** that describes the **people**, **institutions**, **entities**, and **activities** involved in **producing**, **influencing**, or **delivering** a piece of **data** or a thing."

Moreau and Missier "PROV-DM: The PROV data model." W3C Recommendation 2013.

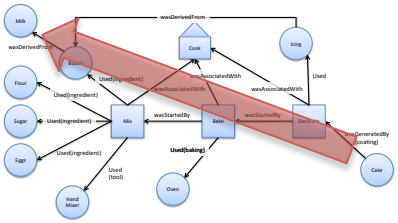
Example of Provenance



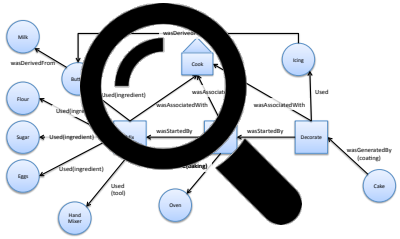
"Life can only be understood backwards; but it must be lived forwards."

Søren Kierkegaard, 1843.

Provenance is useful for…

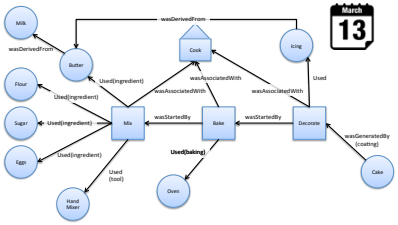


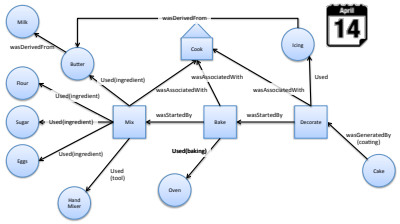
Interpreting and Understanding results

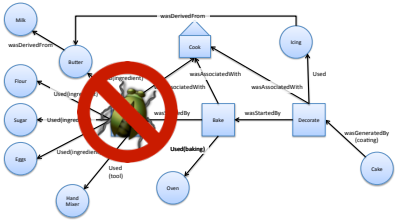


Auditing

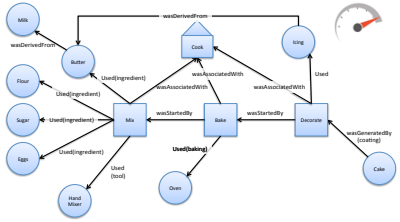
Reproducibility







Debugging

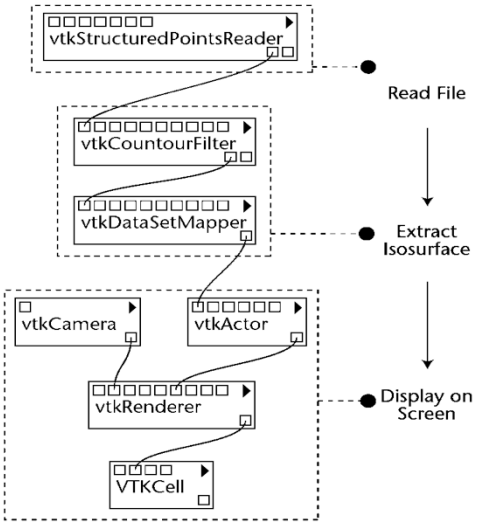


Caching

Among others applications (attribution, reuse, trustworthiness, sharing, …)

Provenance Gathering


```
1  import vtk
2
3  data = vtk.vtkStructuredPointsReader()
4  data.setFileName("../../../examples/data/head.120.vtk")
5
6  contour = vtk.vtkContourFilter()
7  contour.SetInput(0, data.GetOutput())
8  contour.SetValue(0, 67)
9
10 mapper = vtk.vtkPolyDataMapper()
11 mapper.SetInput(contour.GetOutput())
12 mapper.ScalarVisibilityOff()
13
14 actor = vtk.vtkActor()
15 actor.SetMapper(mapper)
16
17 cam = vtk.vtkCamera()
18 cam.SetViewUp(0,0,-1)
19 cam.SetPosition(745,-453,369)
20 cam.SetFocalPoint(135,135,150)
21 cam.ComputeViewPlaneNormal()
22
23 ren = vtk.vtkRenderer()
24 ren.AddActor(actor)
25 ren.SetActiveCamera(cam)
26 ren.ResetCamera()
27
28 renwin = vtk.vtkRenderWindow()
29 renwin.AddRenderer(ren)
30
31 style = vtk.vtkInteractorStyleTrackballCamera()
32 iren = vtk.vtkRenderWindowInteractor()
33 iren.SetRenderWindow(renwin)
34 iren.SetInteractorStyle(style)
35 iren.Initialize()
36 iren.Start()
```



Manual	General-purpose Script	SWfMS
• Error prone • Time consuming • Unviable for complex experiments	• Complex programming model • Primitive execution logs (if any)	• Visual programming interface • Controlled execution environment • Support for parallel execution • Built-in provenance support

Freire et al. "Provenance for computational tasks: A survey." CS&E 10(3) 2008

So, let's use SWfMS! ☺

Among many others…

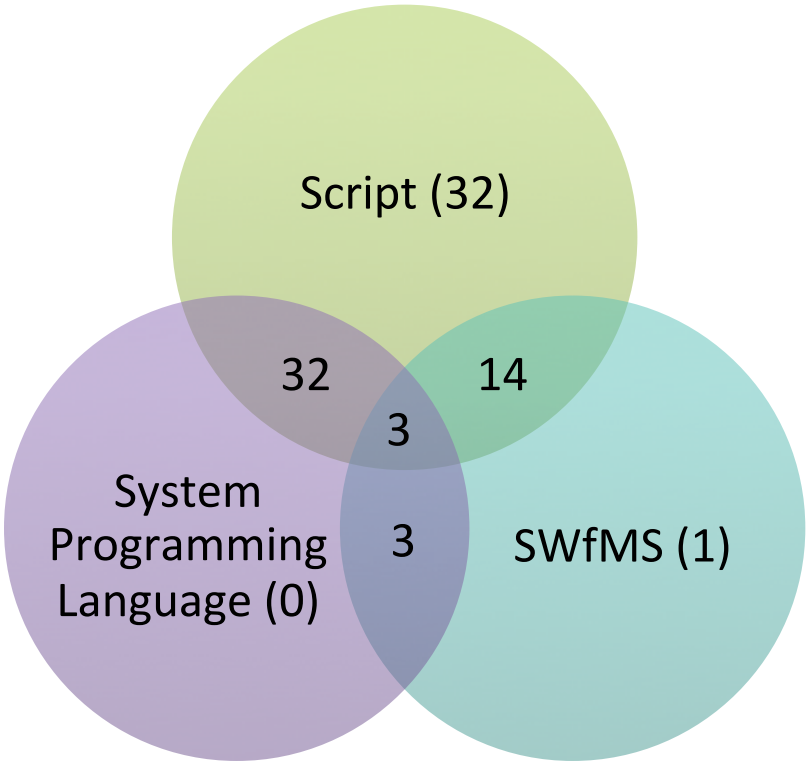
However, lots of people still use scripts ☹

Instituto de Computação

ESI Laboratório de Engenharia de Sistemas e Informação

95%

of the respondents* have scripts among their preferred/more often used tools to run experiments



*Survey sent to AMC@UvA (Olabarriaga), UFRJ (Mattoso), DATAONE (newsletter), DBBras (mailing list), FIOCRUZ (Davila), USP (Traina), INRIA-Montpellier (Zenith group), LNCC (Ocana), PW 2016 TPC, SciPyLA (Telegram), Software Carpentry (mailing list), U. Nantes (Gaignard), UPENN (Davidson), receiving 85 answers.

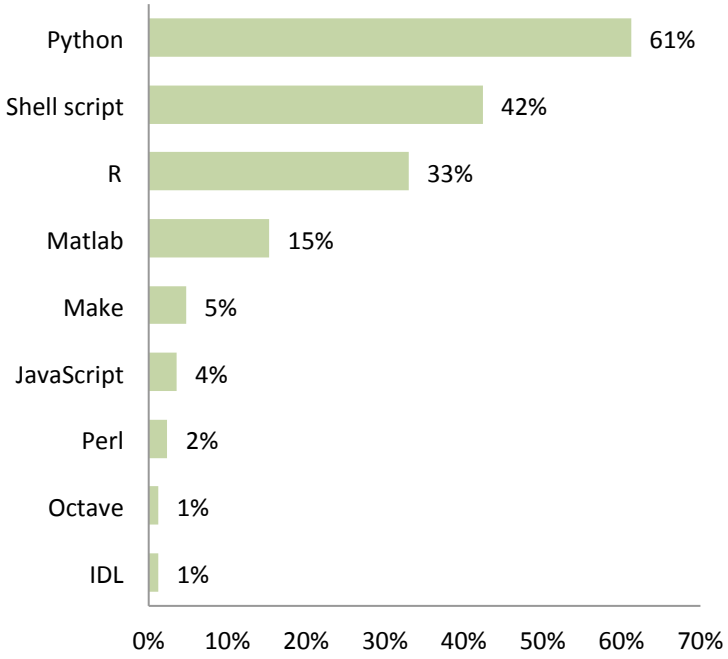
However, lots of people still use scripts ☹

Instituto de Computação

ESI Laboratório de Engenharia de Sistemas e Informação

61%

of the respondents* have Python among their preferred/more often used tools to run experiments



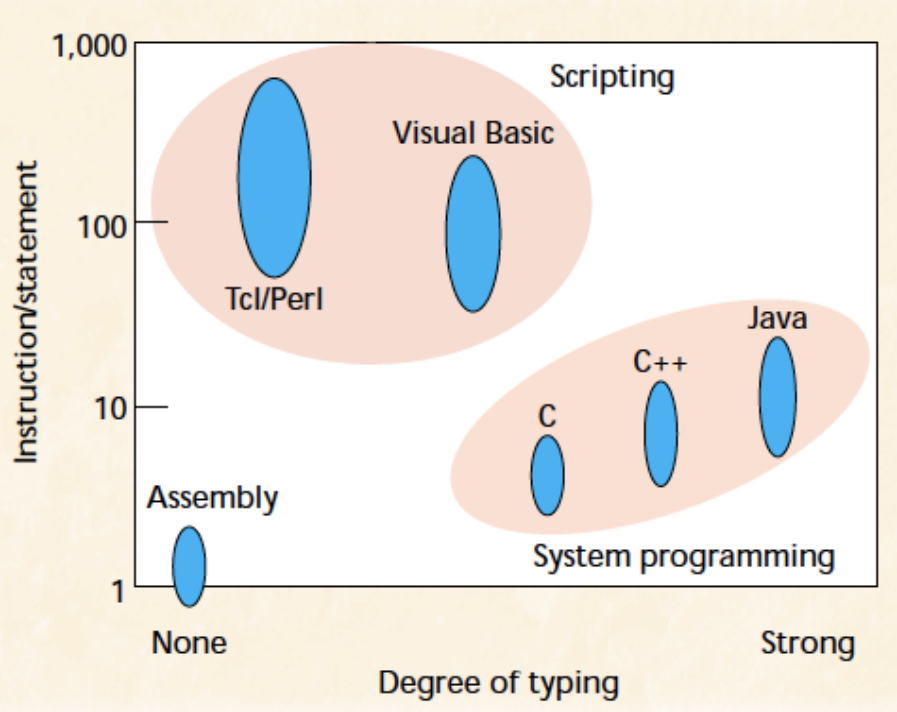
*Survey sent to AMC@UvA (Olabarriaga), UFRJ (Mattoso), DATAONE (newsletter), DBBras (mailing list), FIOCRUZ (Davila), USP (Traina), INRIA-Montpellier (Zenith group), LNCC (Ocana), PW 2016 TPC, SciPyLA (Telegram), Software Carpentry (mailing list), U. Nantes (Gaignard), UPENN (Davidson), receiving 85 answers.

But what exactly are Scripts?

- There is no robust definition in the literature!
- Our to-be-improved definition:
  - "A script is a program conceived for gluing components, which may have been written in different programming languages"
- Actually, it does not matter much…
  - "When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck" (James Whitcomb Riley)

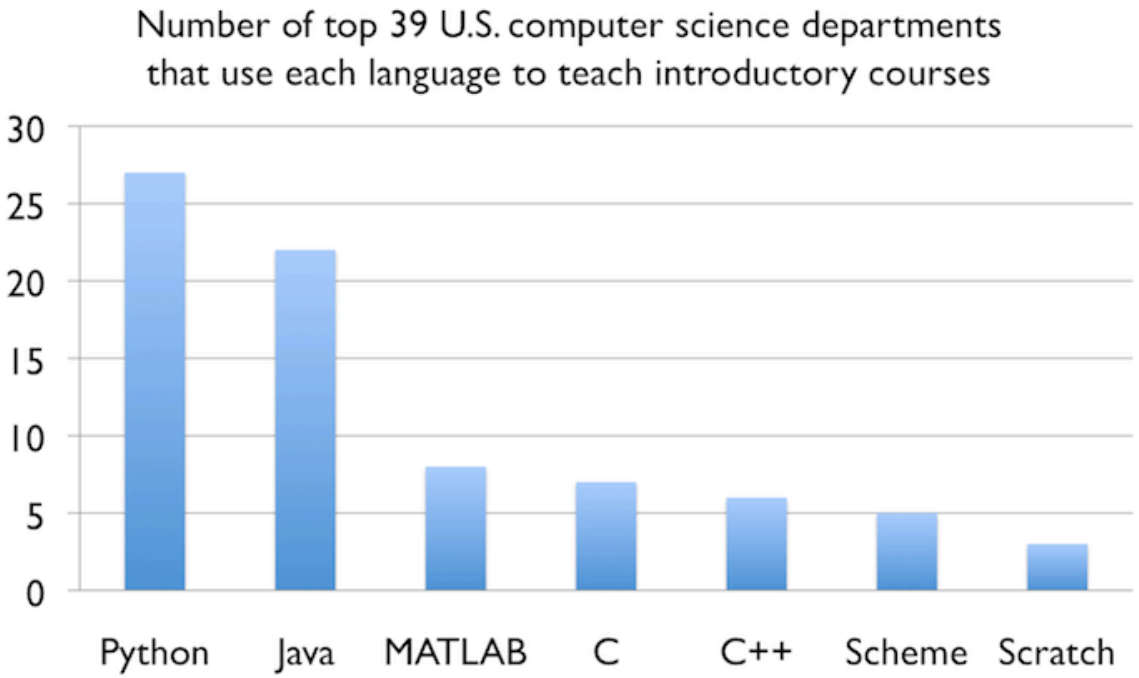
Scripts are high-level programs

- Everything is Object
- Multiparadigm
- Typeless (dynamically-typed)
- Interpreted
- Automatic memory management
- Extensive component library



Ousterhout “Scripting: Higher level programming for the 21st century.” Computer 31(3) 1998

Scripts are easy to learn



Guo “Python is Now the Most Popular Introductory Teaching Language at Top U.S. Universities”, BLOG@CACM July 2014

Scripts are easy to code

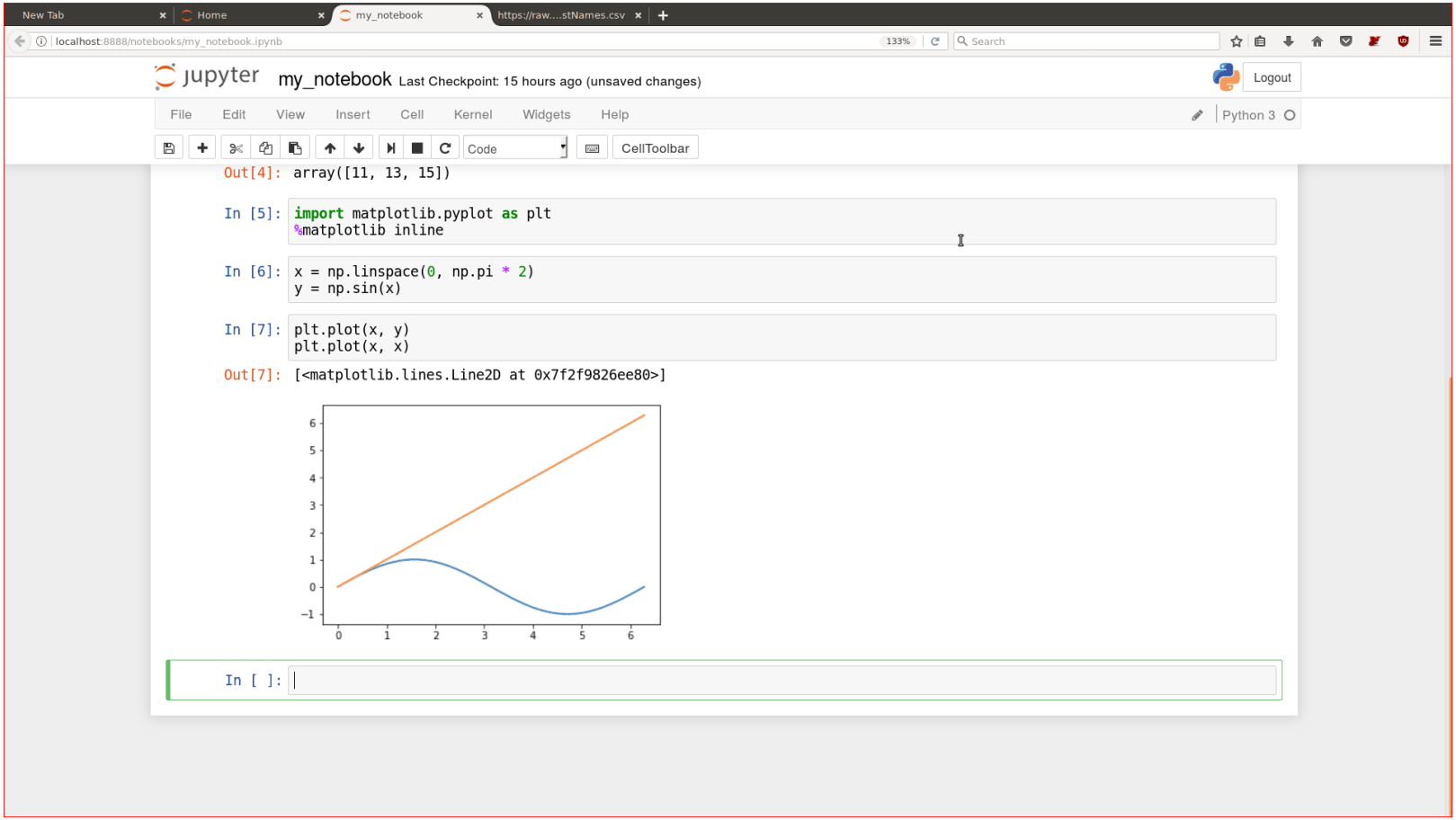
Application (contributor)	Comparison	Code ratio*	Effort ratio**	Comments
Database application (Ken Corey)	C++ version: 2 months Tcl version: 1 day		60	C++ version implemented first; Tcl version had more functionality
Computer system test and installation (Andy Belsey)	C test application: 272,000 lines, 120 months C FIS application: 90,000 lines, 60 months Tcl/Perl version: 7,700 lines, 8 months	47	22	C version implemented first; Tcl/Perl version replaced both C applications
Database library (Ken Corey)	C++ version: 2-3 months Tcl version: 1 week		8-12	C++ version implemented first
Security scanner (Jim Graham)	C version: 3,000 lines Tcl version: 300 lines	10		C version implemented first; Tcl version had more functionality
Display oil well production curves (Dan Schenck)	C version: 3 months Tcl version: 2 weeks		6	Tcl version implemented first
Query dispatcher (Paul Healy)	C version: 1,200 lines, 4-8 weeks Tcl version: 500 lines, 1 week	2.5	4-8	C version implemented first, uncommented; Tcl version had comments, more functionality
Spreadsheet tool	C version: 1,460 lines Tcl version: 380 lines	4		Tcl version implemented first
Simulator and GUI (Randy Wang)	Java version: 3,400 lines, 3-4 weeks Tcl version: 1,600 lines, <1 week	2	3-4	Tcl version had 10 to 20 percent more functionality and was implemented first

* Code ratio is the ratio of lines of code for the two implementations (<1 means the system programming language required more lines).

** Effort ratio is the ratio of development times. In most cases the two versions were implemented by different people.

Ousterhout “Scripting: Higher level programming for the 21st century.” Computer 31(3) 1998

Scripts are interactive



http://n-s-f.github.io/2017/03/25/r-to-python.html

Scripting vs System Programming

Instituto de Computação

ESI Laboratório de Engenharia de Sistemas e Informação

Scripting languages

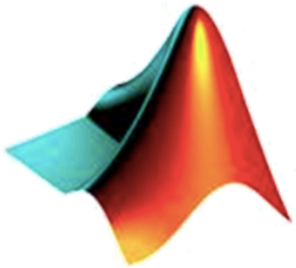
- Programming in the small
- Casual Programmers
- Gluing existing code
- Exploratory prototyping

System programming languages

- Programming in the large
- Professional Programmers
- Programming from scratch
- Industrial-strength development

Loui “In praise of scripting: Real programming pragmatism." Computer 41(7) 2008

Some popular scripting languages



MATLAB

programming republic of

Perl

Scripts-based initiatives for science miss provenance

Nevertheless, some tools have emerged for this end

Astro-WISE	Becker and Chambers	Bochner, Gude, and Schreiber	CPL	CXXR	Datatrack
ES3	ESSW	IncPy	Lancet	Magni	Michaelides et al.
noWorkflow	Provenance Curious	pypet	RDataTracker	Sacred	SisGExp
StarFlow	Sumatra	Tariq, Ali, and Gehani	Variolite	VCR	versuchung
	WISE	YesWorkflow	YW*NW		

noWorkflow

not only Workflow

- **Transparently** capture **provenance from Python scripts** at **fine grain**
- Consider multiple executions (**trials**)
- Provide **visualizations** for provenance analysis
- Allows **querying** provenance in different languages

noWorkflow

What?

vs

How?

Installing and running

Instead of running

```
$ python experiment.py
```

Install noWorkflow (once)

```
$ pip install noworkflow[all]
```

And run

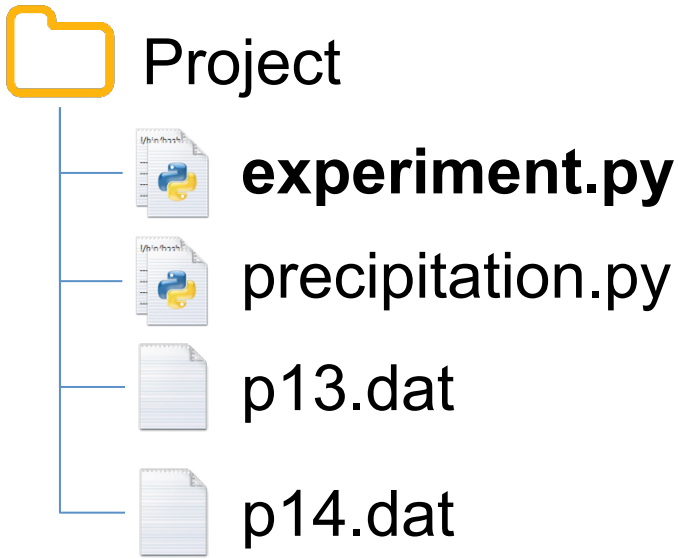
```
$ now run experiment.py
```

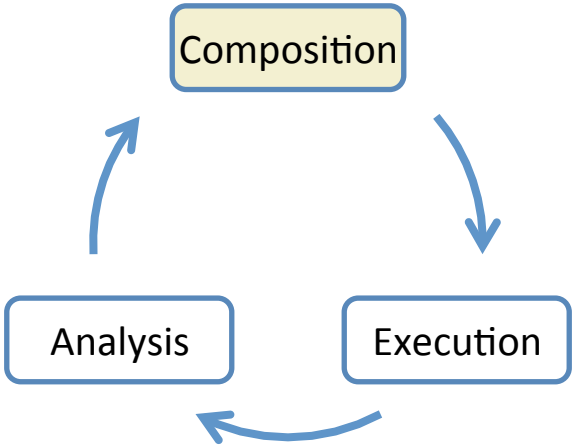
New experiment!

http://www.ifaketext.com/

1st iteration

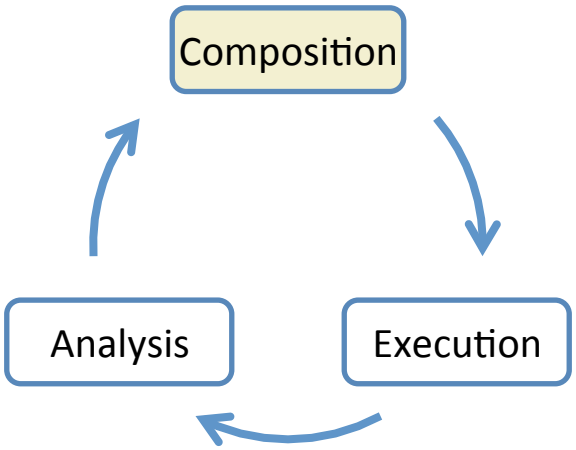
- H_1: “The precipitation of each month remains constant across years”





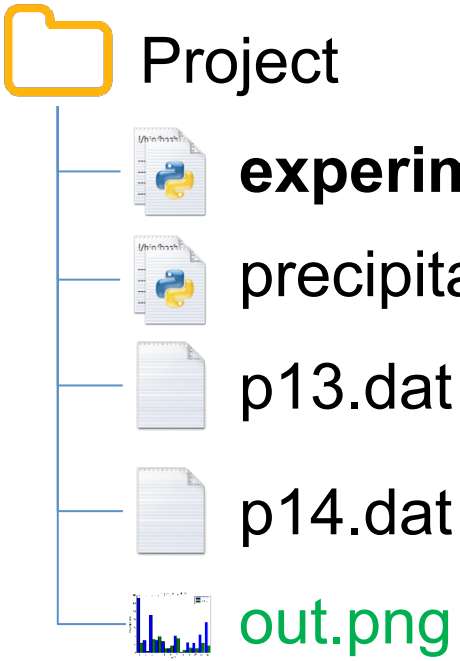
1st iteration – experiment.py

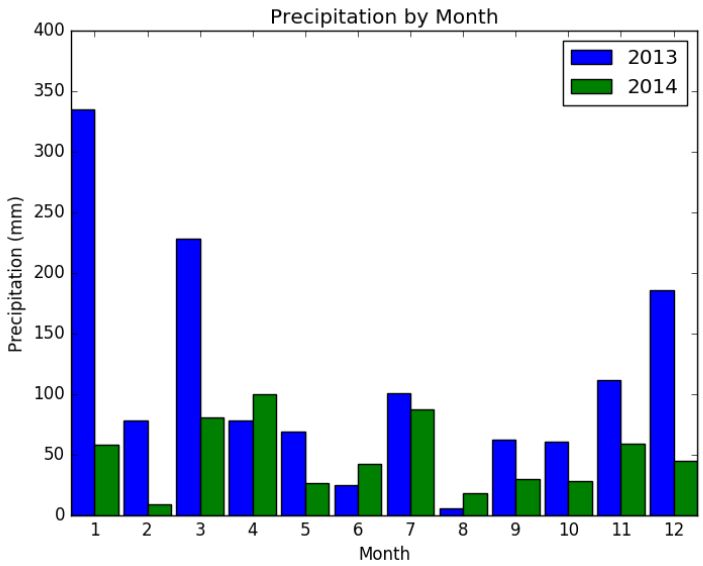
```
 1| import numpy as np
 2| from precipitation import read, sum_by_month
 3| from precipitation import create_bargraph
 4|
 5| months = np.arange(12) + 1
 6|
 7| d13, d14 = read("p13.dat"), read("p14.dat")
 8|
 9| prec13 = sum_by_month(d13, months)
10| prec14 = sum_by_month(d14, months)
11|
12| create_bargraph("out.png", months,
13|     ["2013", "2014"],
14|     prec13, prec14)
```

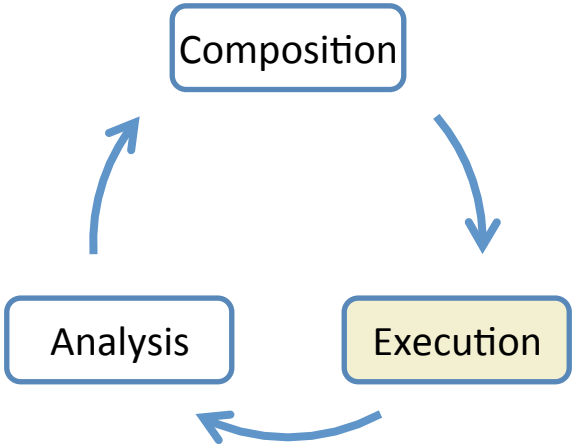


1st iteration – execution

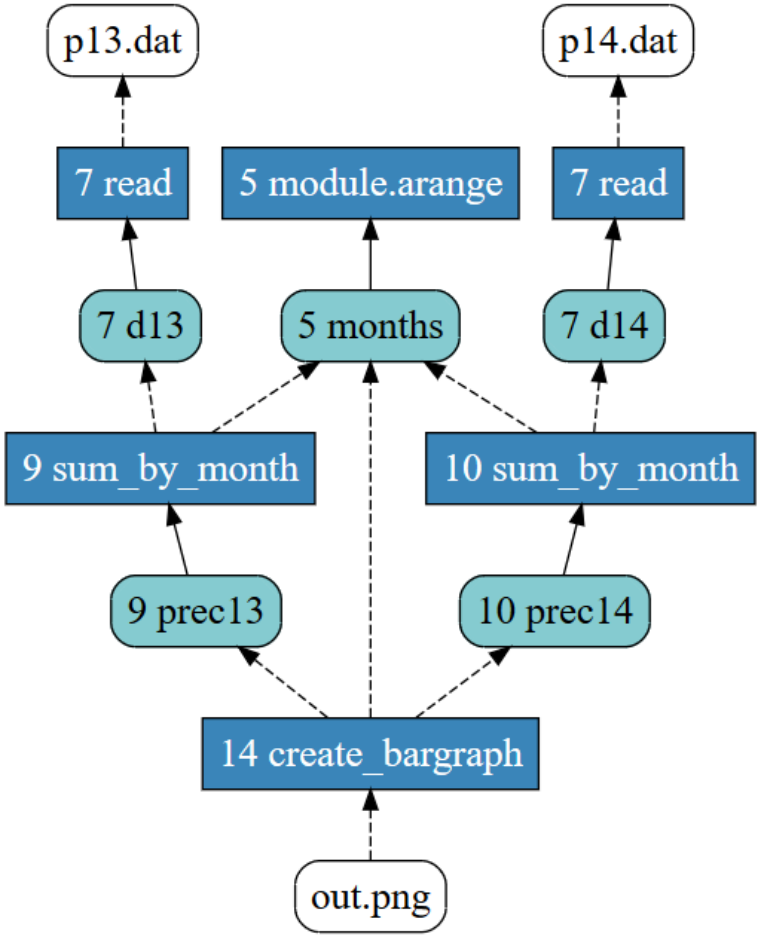
$ now run -e Tracker experiment.py



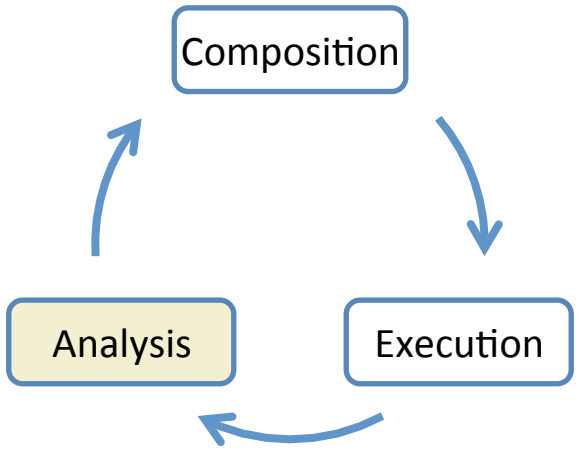




1st iteration – provenance




```
 1| import numpy as np
 2| from precipitation import read, sum_by_month
 3| from precipitation import create_bargraph
 4|
 5| months = np.arange(12) + 1
 6|
 7| d13, d14 = read("p13.dat"), read("p14.dat")
 8|
 9| prec13 = sum_by_month(d13, months)
10| prec14 = sum_by_month(d14, months)
11|
12| create_bargraph("out.png", months,
13|    ["2013", "2014"],
14|    prec13, prec14)
```

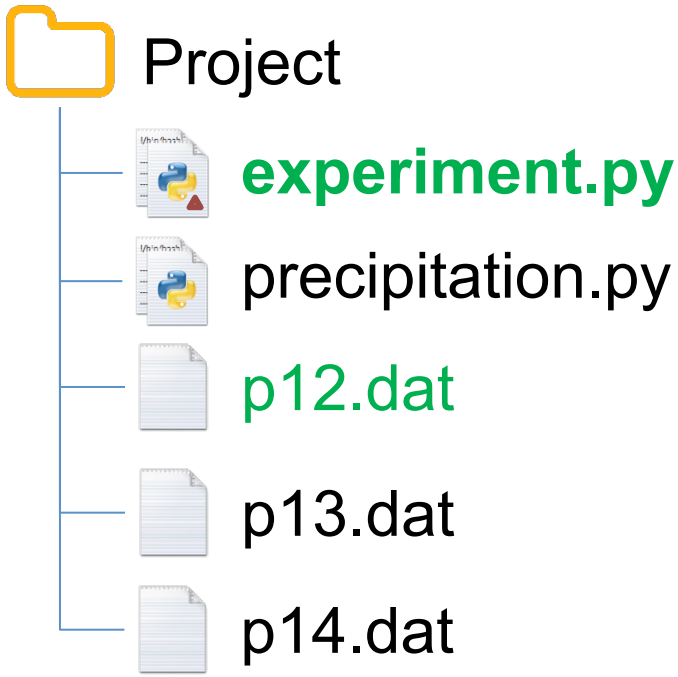


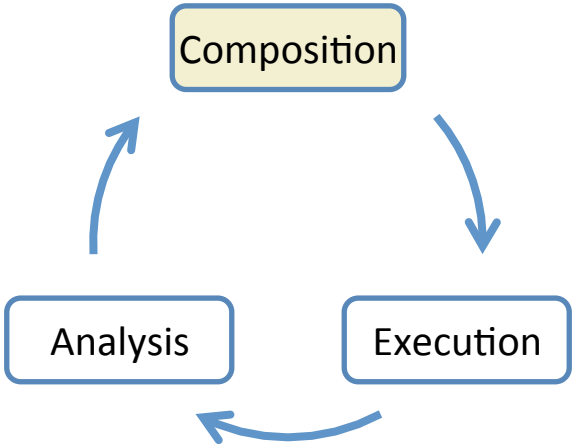
New hypothesis!

http://www.ifaketext.com/

2nd Iteration

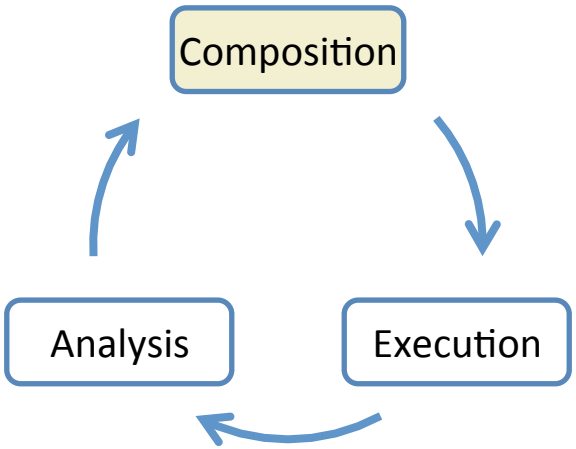
- H_2: “The precipitation for each month remains constant across years if there is no drought”





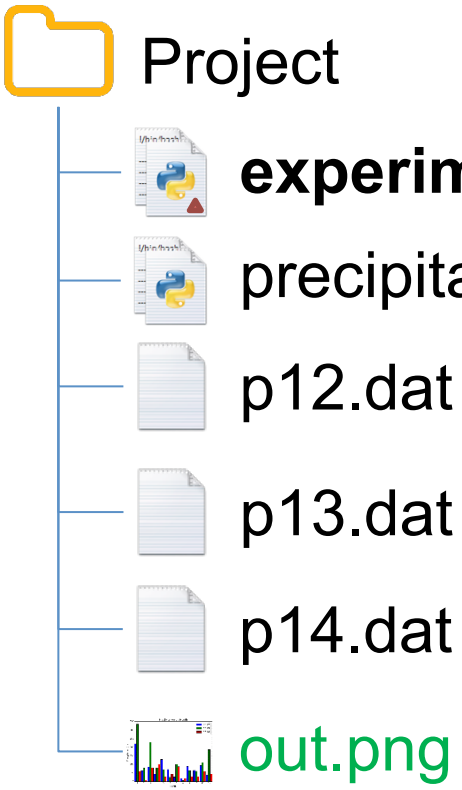
2nd Iteration – experiment.py

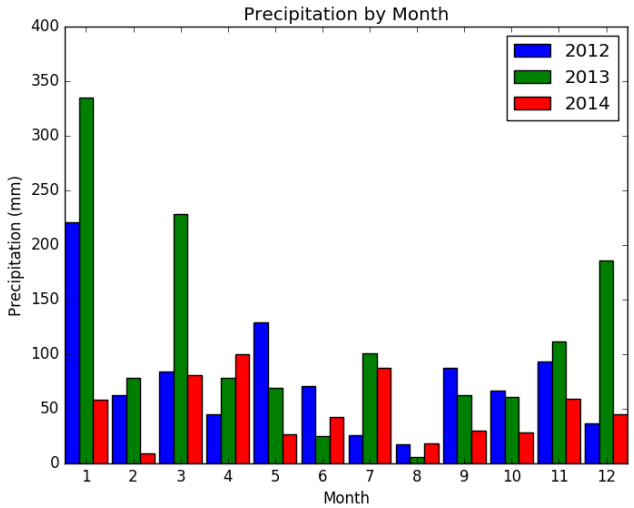
```
 1| import numpy as np
 2| from precipitation import read, sum_by_month
 3| from precipitation import create_bargraph
 4|
 5| months = np.arange(12) + 1
 6| d12 = read("p12.dat")
 7| d13, d14 = read("p13.dat"), read("p14.dat")
 8| prec12 = sum_by_month(d12, months)
 9| prec13 = sum_by_month(d13, months)
10| prec14 = sum_by_month(d14, months)
11|
12| create_bargraph("out.png", months,
13|     ["2012", "2013", "2014"],
14|     prec12, prec13, prec14)
```

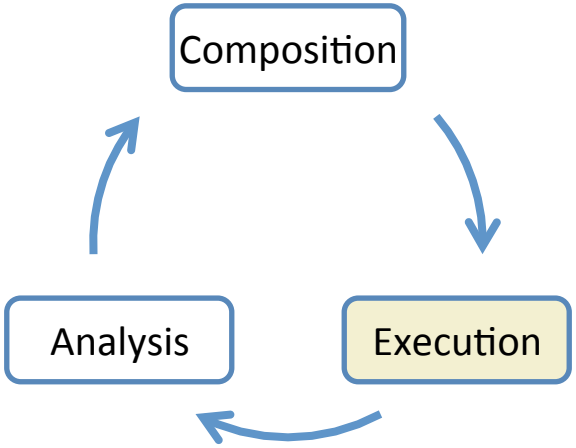


2nd Iteration – execution

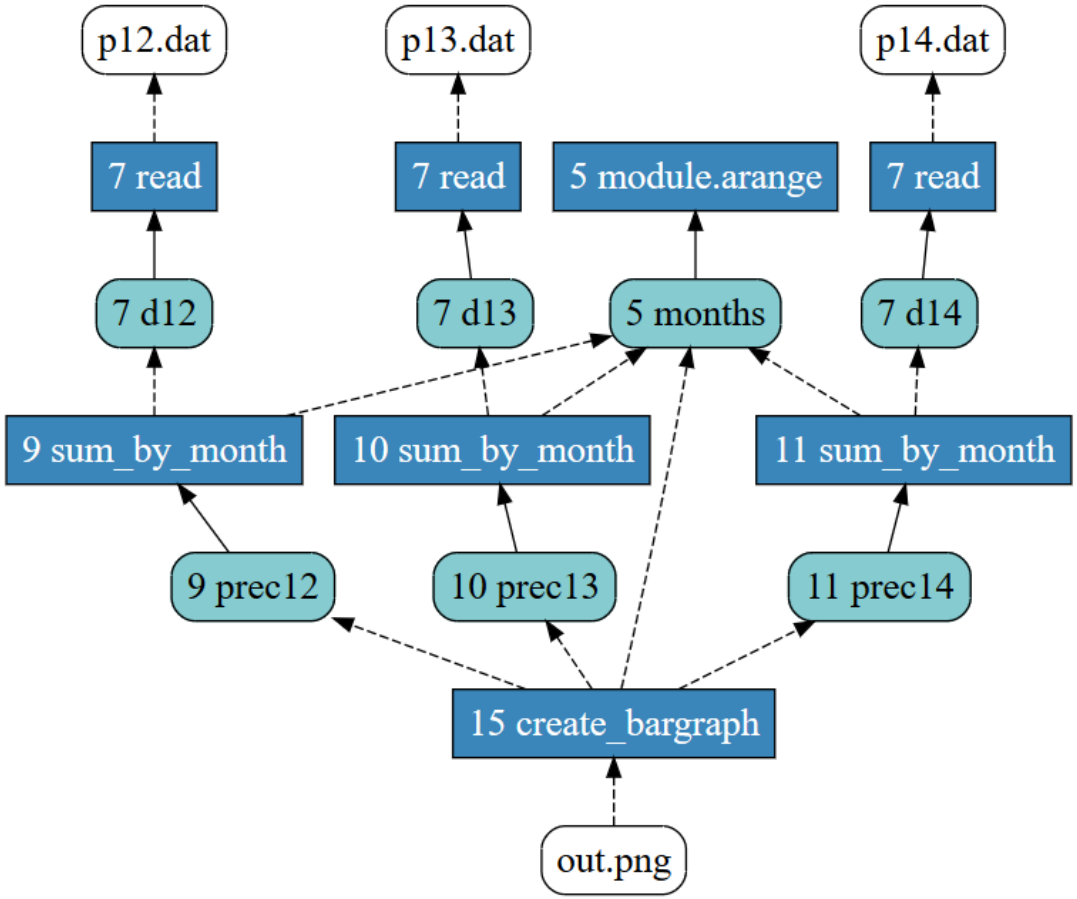
$ now run -e Tracker experiment.py

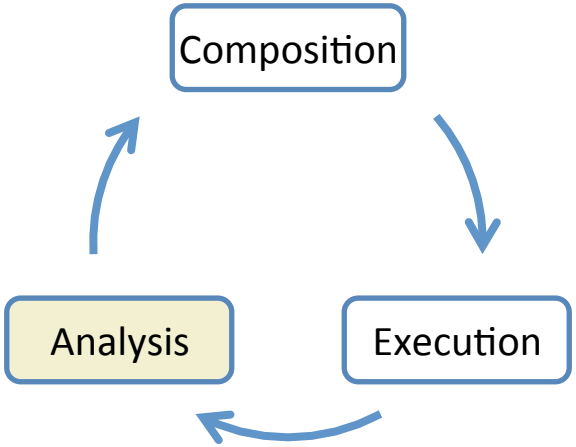






2nd iteration – provenance





More provenance analyses!

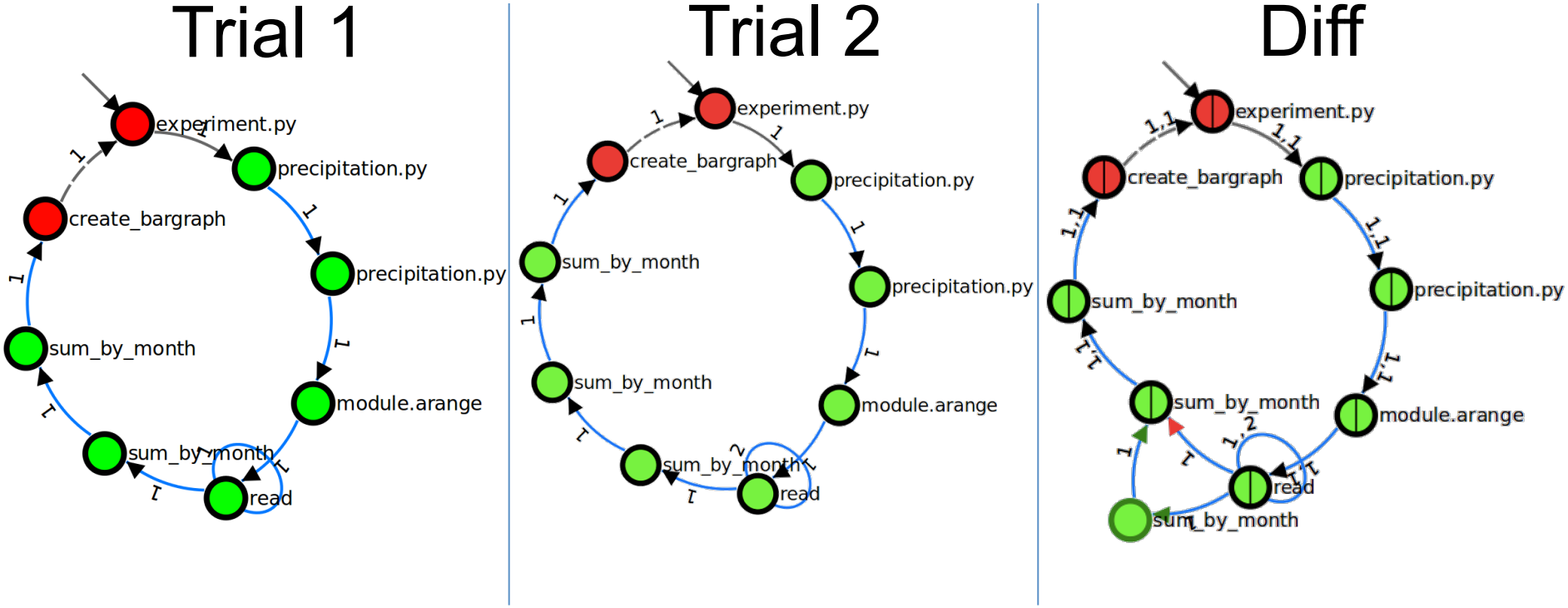
http://www.ifaketext.com/

Textual Diff

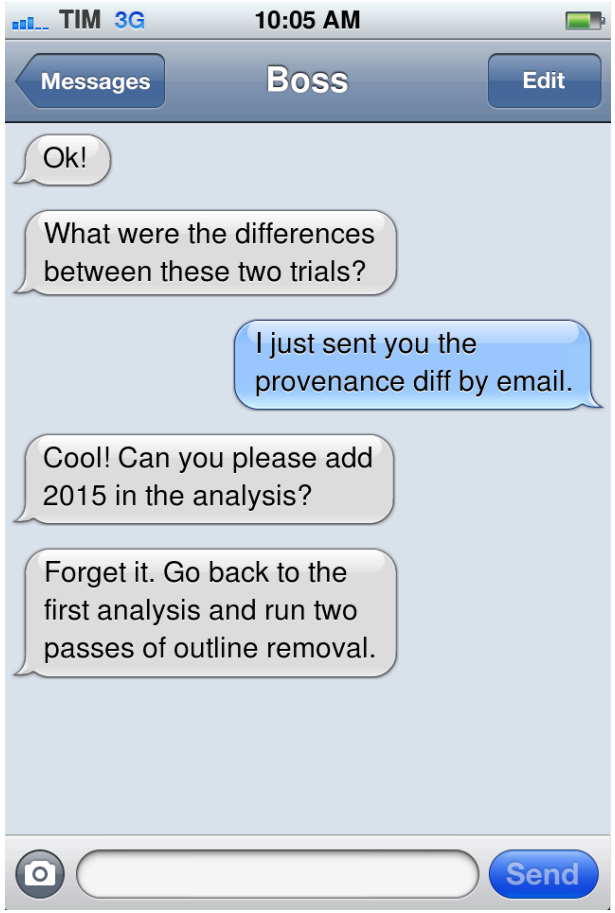
```
$ now diff 1 2 -f -brief

[now] trial diff:
  Start changed from 2016-05-30 7:33:26.105716
                  to 2016-05-30 7:55:26.276369
  Finish changed from 2016-05-30 7:34:27.729060
                   to 2016-05-30 7:56:24.863268
  Duration text changed from 0:01:01.623344 to 0:00:58.586899
  Code hash changed from a66f3052414673feed5e49812e6940a92bba7679
                     to ff62d0f369315fbc209c39379ccf93437725fa31
  Parent id changed from <None> to 1

[now] Brief file access diff
[Additions]   | [Removals]          | [Changes]
(r) p12.dat   | (wb) out.png (new)  |
(wb) out.png  |                     |
```

Activations Diff

- Visualization tool: now vis



After some other requests…

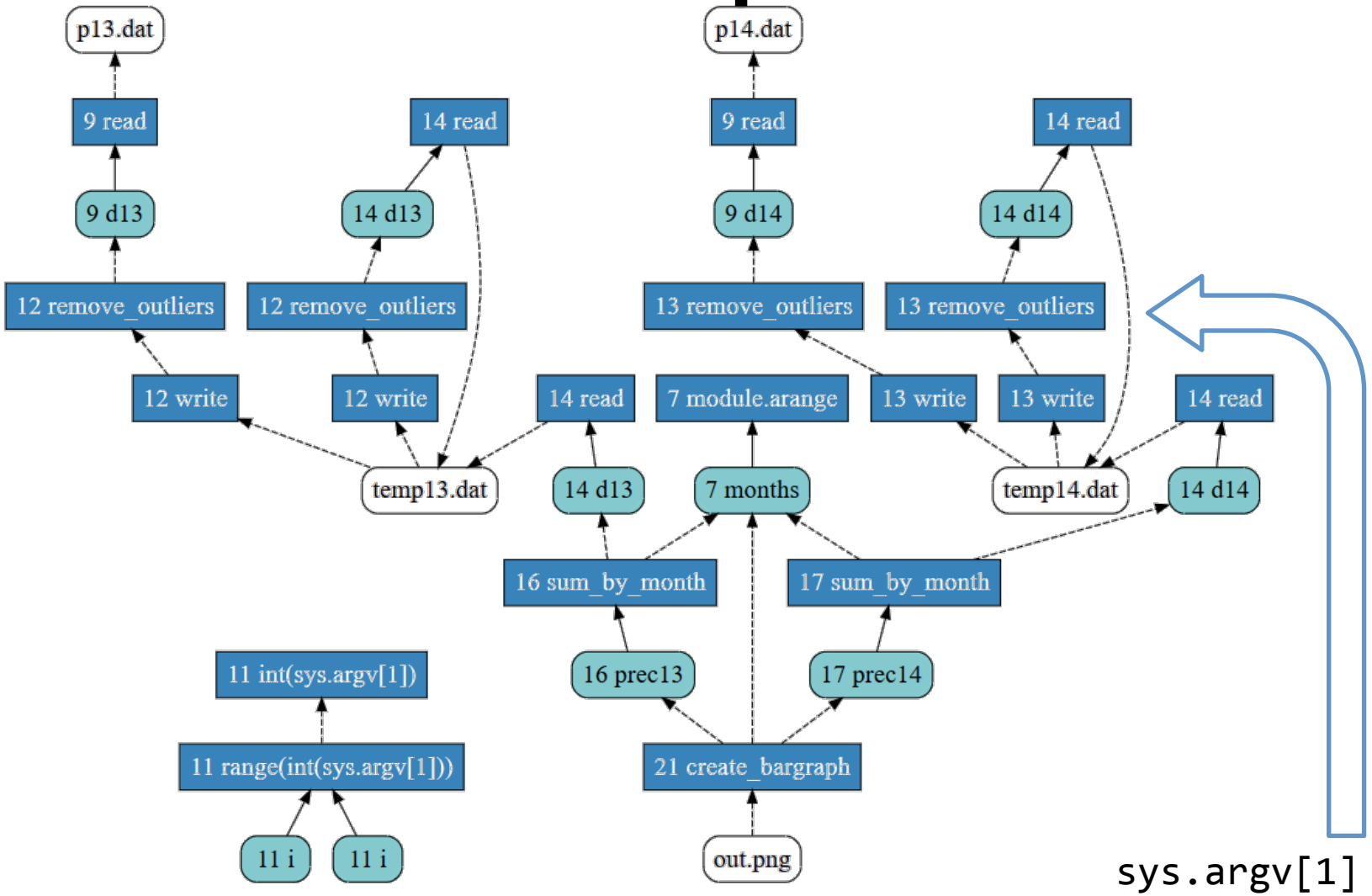


http://www.ifaketext.com/

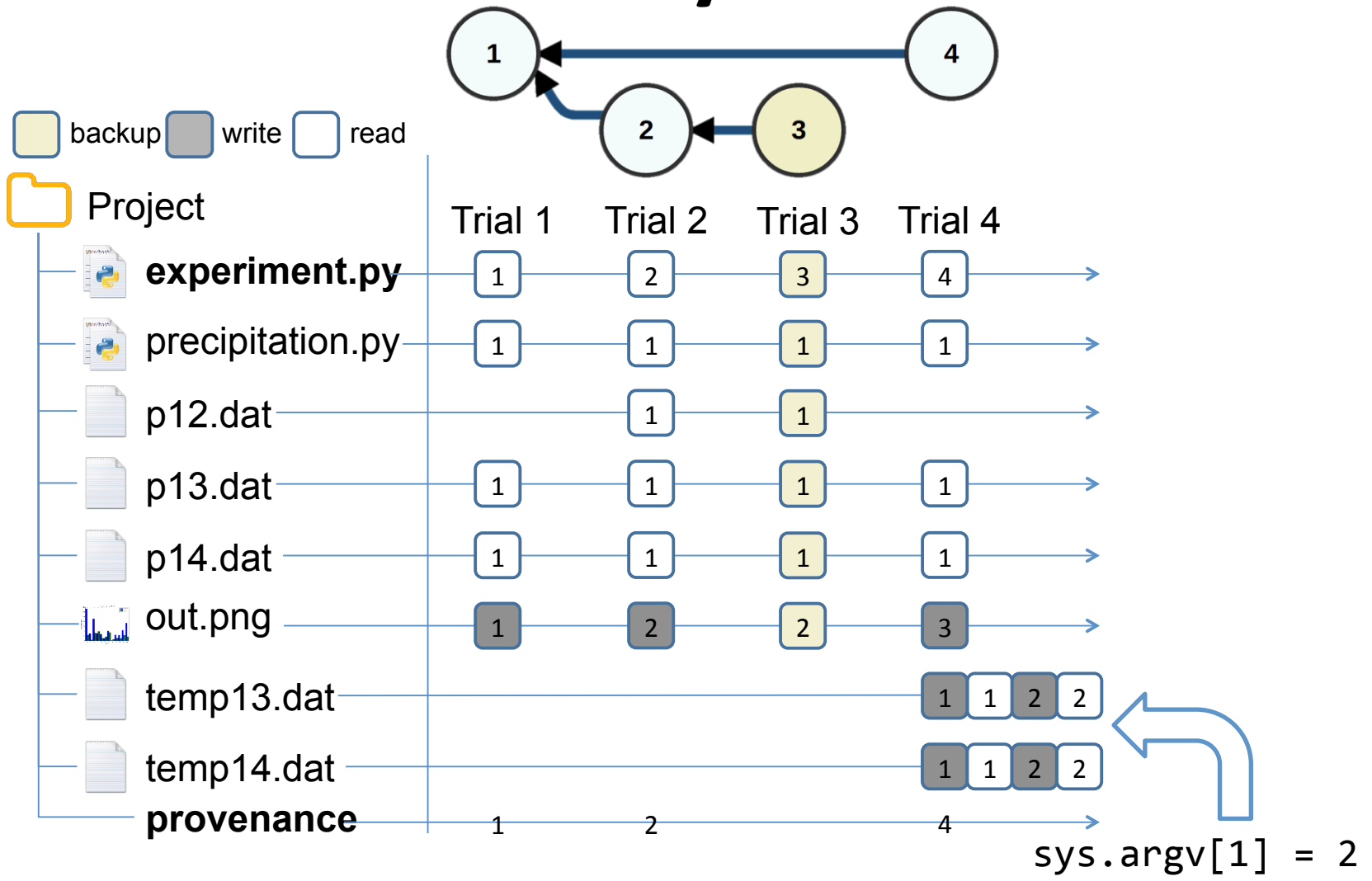
4th iteration – experiment.py

```
 1| import sys
 5| from precipitation import write, remove_outliers
 7| months = np.arange(12) + 1
 9| d13, d14 = read("p13.dat"), read("p14.dat")
10|
11| for i in range(int(sys.argv[1])):
12|    write("temp13.dat",remove_outliers(d13), 2013)
13|    write("temp14.dat",remove_outliers(d14), 2014)
14|    d13,d14=read("temp13.dat"), read("temp14.dat")
15|
16| prec13 = sum_by_month(d13, months)
17| prec14 = sum_by_month(d14, months)
19| create_bargraph("out.png", months,
20|                 ["2013", "2014"],
21|                 prec13, prec14)
```

4th iteration – provenance with temp files



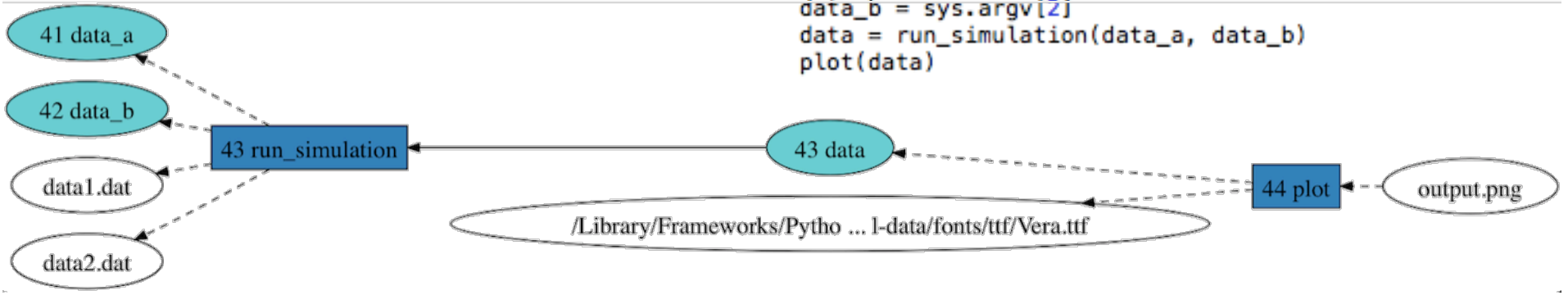
4th iteration – evolution history



Other Examples – blackbox gathering

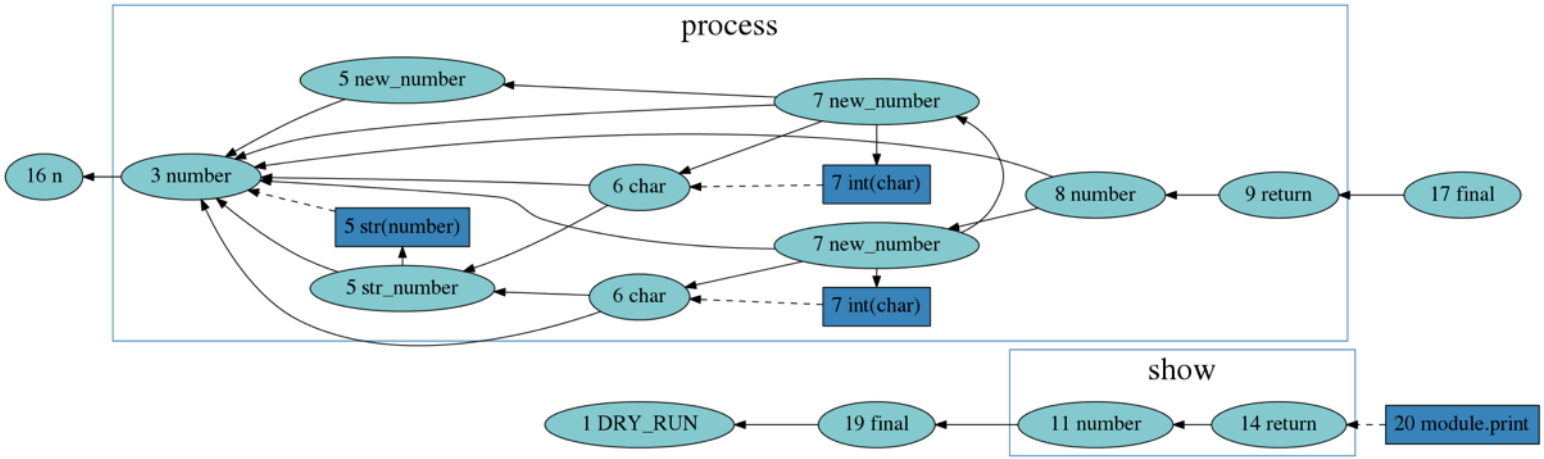
```python
import csv
import sys
import matplotlib.pyplot as plt
from simulator import simulate


def run_simulation(data_a, data_b):
    a = csv_read(data_a)
    b = csv_read(data_b)
    data = simulate(a, b)
    return data


def csv_read(f):
    reader = csv.reader(open(f, "rU"), delimiter=":")
    data = []
    for row in reader:
        data.append(row)
    return data
```

```python
def extract_column(data, column):
        col_data = []
        for row in data:
            col_data.append(float(row[column]))
        return col_data


def plot(data):
    # GetTemperature
    t = extract_column(data, 0)
    # GetPrecipitation
    p = extract_column(data, 1)
    plt.scatter(t, p, marker="o")
    plt.xlabel("Temperature")
    plt.ylabel("Precipitation")
    plt.savefig("output.png")


# Main Program
data_a = sys.argv[1]
data_b = sys.argv[2]
data = run_simulation(data_a, data_b)
plot(data)
```

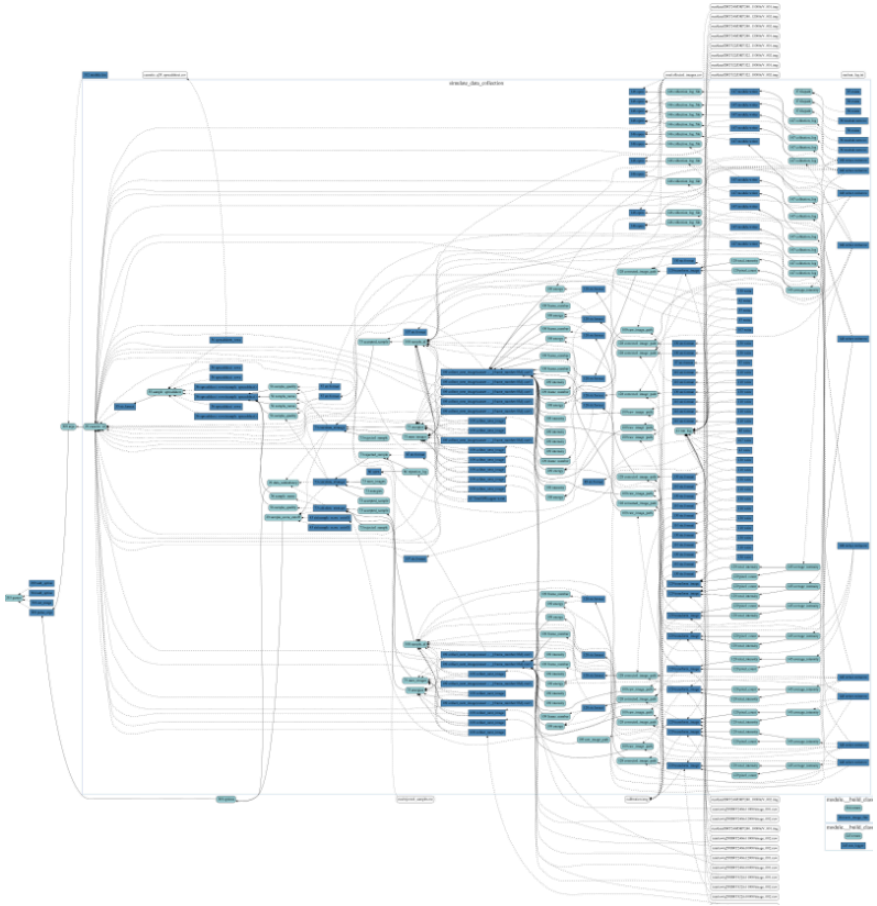


Other Examples – whitebox gathering

```
DRY_RUN = True

def process(number):
    while number >= 10:
        new_number, str_number = 0, str(number)
        for char in str_number:
            new_number += int(char) ** 2
        number = new_number
    return number

def show(number):
    if number not in (1, 7):
        return "unhappy number"
    return "happy number"

n = 10
final = process(n)
if DRY_RUN:
    final = 1
print(show(final))
```



Other Examples – fine-grained gathering



- Consequence of an unstructured script
- Not appropriate for visual analysis
- However, provenance can still be queried using **Prolog** export option of noWorkflow!

https://github.com/gems-uff/yin-yang-demo

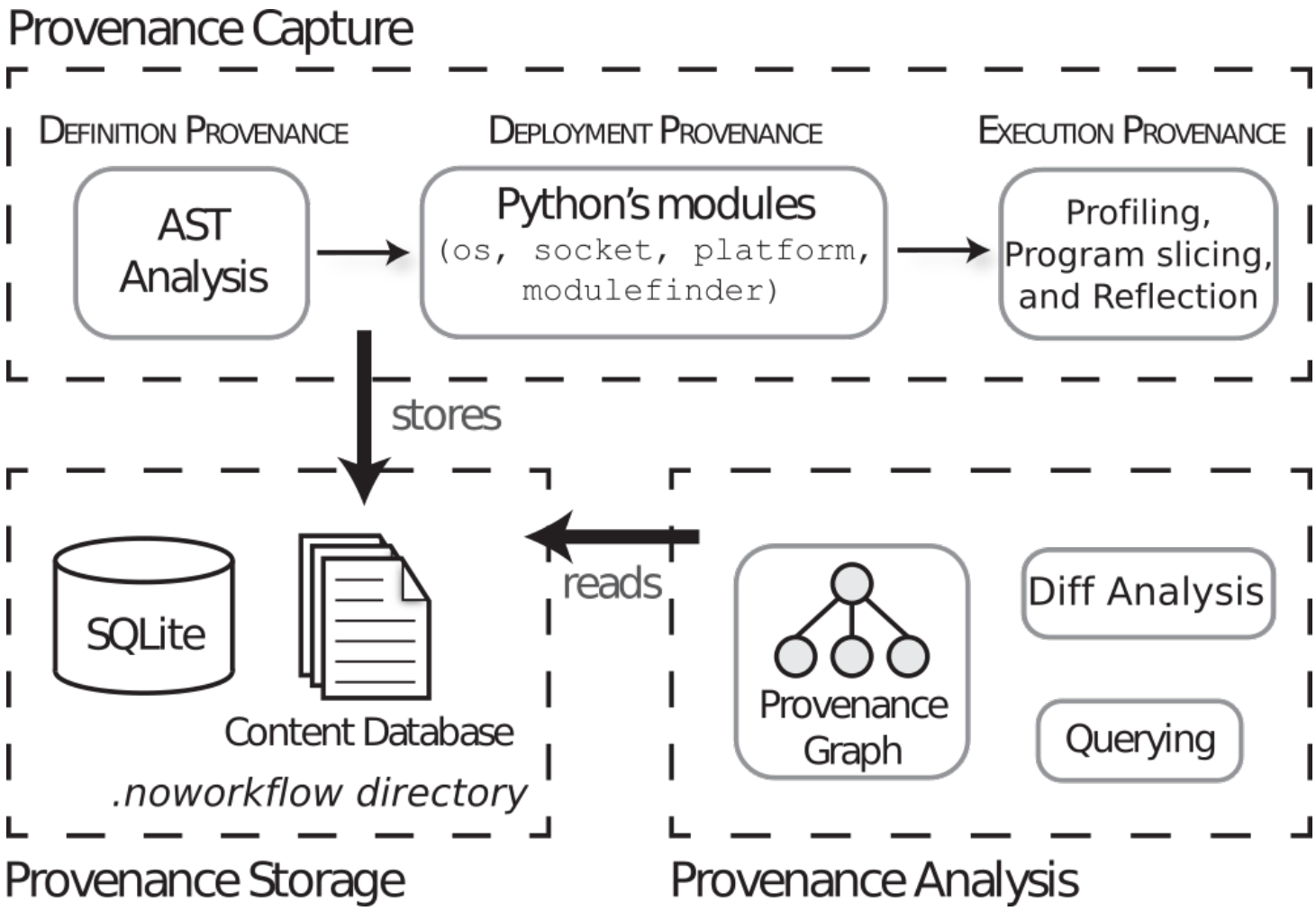
noWorkflow

What?

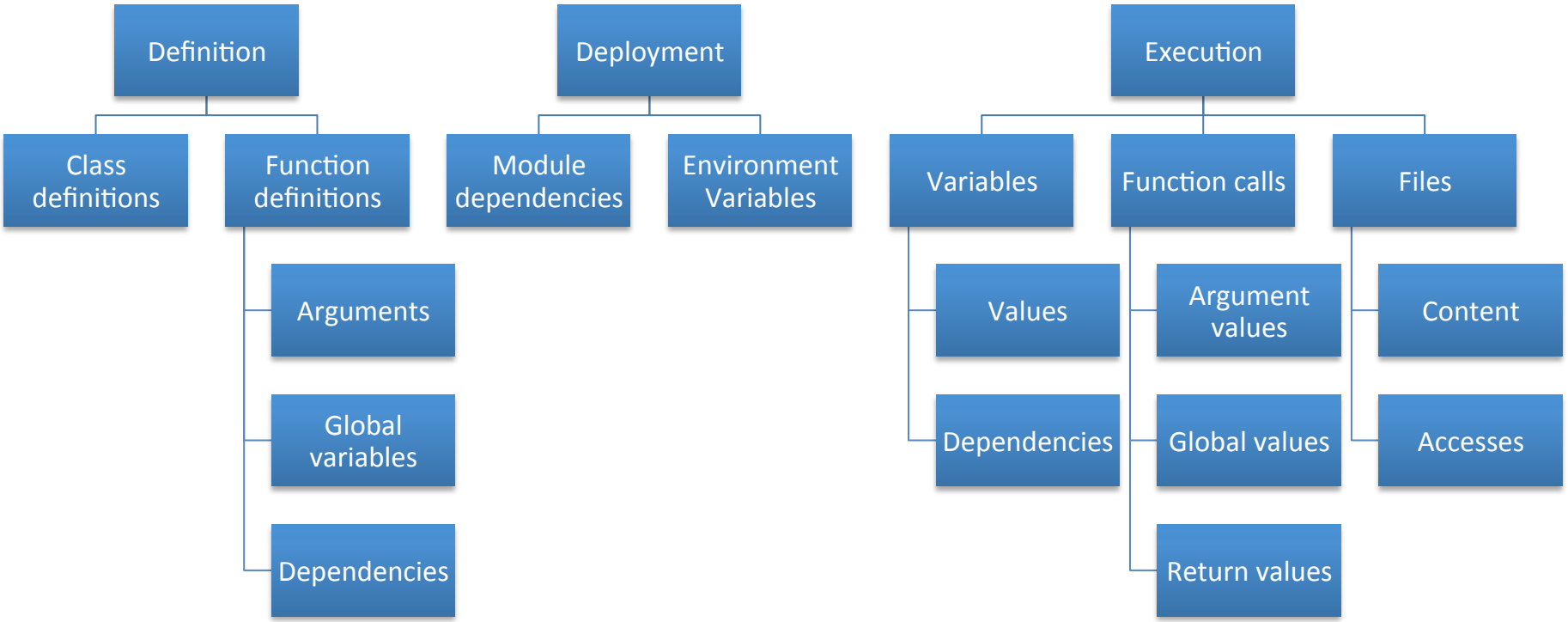
vs

How?

Architecture



Collected data



Definition provenance (function definitions)

```
import csv
import json
threshold = 15
def run_simulation(data_a, data_b):
    global threshold
    ...
def csv_read(f):
    fc = open(f, 'rU')
    ...
def extract_column(data, column):
    x = float(data[column])
    ...
def plot(x, y):
    ...
    plt.savefig("output.png")
...
```

AST analysis

sha1: baa730

```
def run_simulation(data_a, data_b):
    global threshold
    ...
```

Arguments: data_a, data_b
Global: threshold
…

sha1: a1149d

```
def csv_read(f):
    fc = open(f, 'rU')
    ...
```

Arguments: f
Calls: open
…

Definition provenance (function definitions)

Function definitions

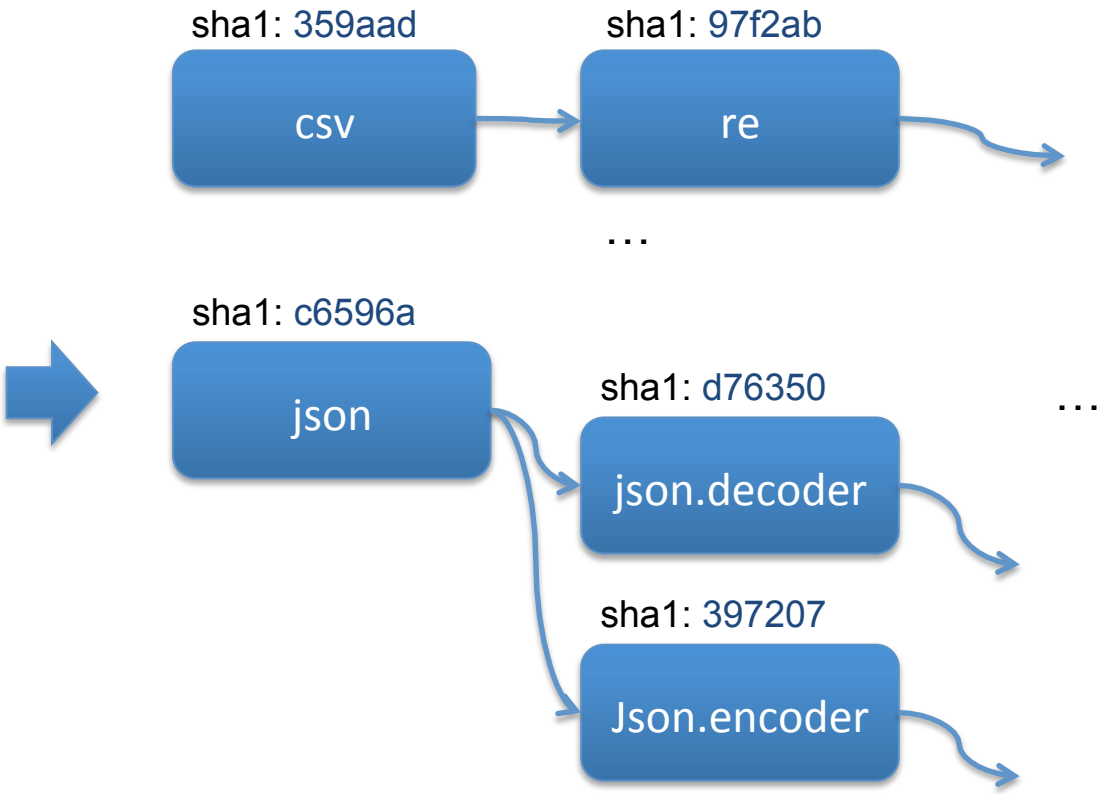
id	name	code_hash	trial_id
1	plot	ae65a2036cfcfeb70bc75646f...	1
2	run_simulation	aaf5d621671bda46f6f91d66...	1
3	extract_column	36fca5011c32e5bc8fe2309c...	1
4	csv_read	d914038c94dcbd4bfa656c1e...	1

Arguments, Global variables, and Function calls

id	name	type	function_def_id
1	x	ARGUMENT	1
2	y	ARGUMENT	1
3	data_a	ARGUMENT	2
4	data_b	ARGUMENT	2
5	threshold	GLOBAL	2
6	data	ARGUMENT	3
7	column	ARGUMENT	3
8	float	FUNCTION	3
9	f	ARGUMENT	4
10	open	FUNCTION	4

Deployment provenance (module dependencies)

```
import csv
import json
threshold = 15
def run_simulation(data_a, data_b):
    global threshold
    ...
def csv_read(f):
    fc = open(f, 'rU')
    ...
def extract_column(data, column):
    x = float(data[column])
    ...
def plot(x, y):
    ...
    plt.savefig("output.png")
...
```



Deployment provenance (module dependencies)

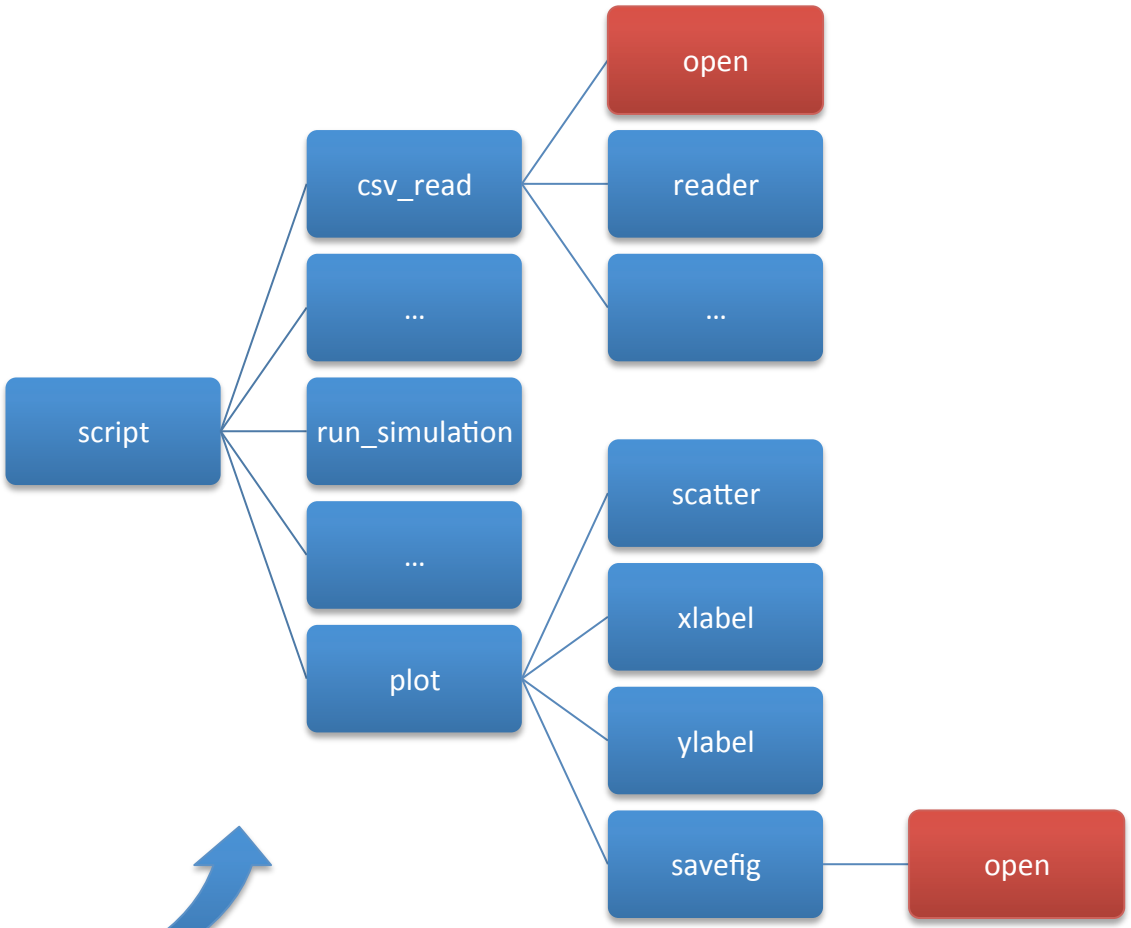
id	name	version	file	code_hash	trial_id
578	parser	0.5	/Applications/Canopy.ap...	1c3d29a3964c7ccce610...	1
266	logging	0.5.1.2	/Applications/Canopy.ap...	0f25732b370c8a9b5ef0...	1
51	PIL.JpegImagePlugin	0.6	/Users/leomurta/Library...	4c15e4929257f41438ee...	1
40	PIL.BmpImagePlugin	0.7	/Users/leomurta/Library...	81aa830d0648239fc115...	1
370	multiprocessing	0.70a1	/Applications/Canopy.ap...	29a75113696caffaaec85...	1
41	PIL.GifImagePlugin	0.9	/Users/leomurta/Library...	f32e15956af93ce5b32bf...	1
53	PIL.PngImagePlugin	0.9	/Users/leomurta/Library...	0bcca794a0059096acf6a...	1
94	_csv	1.0	/Applications/Canopy.ap...	ef4d2e093e01bdc577cba...	1
152	csv	1.0	/Applications/Canopy.ap...	359aad7eca382bb46c9a...	1
533	numpy.ma	1.0	/Users/leomurta/Library...	964546e27356efd234ce...	1
535	numpy.ma.extras	1.0	/Users/leomurta/Library...	92ed1d612e3c5fb8682e...	1
703	zlib	1.0	/Applications/Canopy.ap...	464f5b374771708a6823...	1
700	xmlrpclib	1.0.1	/Applications/Canopy.ap...	bc8883c8ae46b737b123...	1
583	platform	1.0.7	/Applications/Canopy.ap...	080d6c19c0535cea0c45...	1
95	_ctypes	1.1.0	/Applications/Canopy.ap...	e297fc0053a23eda4162...	1
153	ctypes	1.1.0	/Applications/Canopy.ap...	9631395474bf01bc81c9...	1
610	setuptools	1.1.6	/Applications/Canopy.ap...	367e53014dcff4a0e82c2...	1
39	PIL	1.1.7	/Users/leomurta/Library...	bdea292f3d7f3141d5fea...	1
44	PIL.Image	1.1.7	/Users/leomurta/Library...	c552a828c9f418951e26...	1
669	urllib	1.17	/Applications/Canopy.ap...	575f9e37b4b4c45cac90...	1
385	nose	1.2.1	/Users/leomurta/Library...	df681a5f034cf907160eb...	1

Deployment provenance (environment variables)

id	name	value	trial_id
73	OS_VERSION	Darwin Kernel Version 13.0.0:...	1
74	CS_XBS5_LP64_OFF64_LIBS		1
75	SC_AIO_PRIO_DELTA_MAX	-1	1
76	SC_THREAD_STACK_MIN	8192	1
77	TERM_PROGRAM_VERSION	326	1
78	SC_STREAM_MAX	256	1
79	OS_RELEASE	13.0.0	1
80	SC_MEMLOCK	-1	1
81	HOME	/Users/leomurta	1
82	TERM_PROGRAM	Apple_Terminal	1
83	LANG	pt_BR.UTF-8	1
84	SC_SHARED_MEMORY_OBJECTS	-1	1
85	Apple_PubSub_Socket_Render	/tmp/launch-Uo1Eq3/Render	1
86	SC_THREAD_THREADS_MAX	-1	1
87	SC_COLL_WEIGHTS_MAX	2	1
88	SC_THREAD_ATTR_STACKADDR	200112	1
89	_	/Users/leomurta/Library/Ent...	1
90	SC_THREAD_ATTR_STACKSIZE	200112	1
91	PYTHON_IMPLEMENTATION	CPython	1
92	SC_JOB_CONTROL	200112	1
93	SC_FSYNC	200112	1

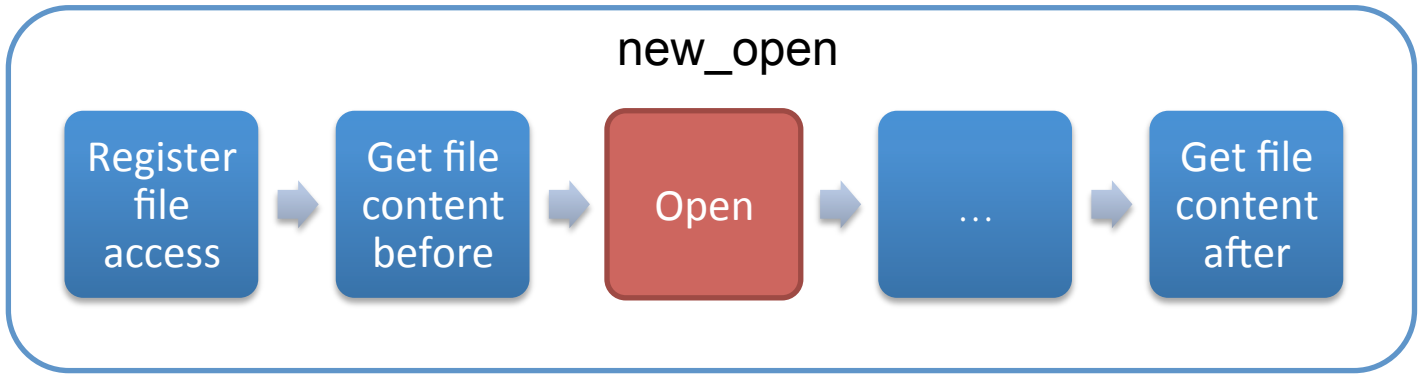
Execution provenance (function calls and file accesses)

```
import csv
import json
threshold = 15
def run_simulation(data_a, data_b):
    global threshold
    ...
def csv_read(f):
    fc = open(f, 'rU')
    ...
def extract_column(data, column):
    x = float(data[column])
    ...
def plot(x, y):
    ...
    plt.savefig("output.png")
...
```



Profiling

Execution provenance (function calls and file accesses)



Monkey patching: builtins.open = self.new_open(open)

Execution provenance (function calls and file accesses)

Instituto de Computação

ESI Laboratório de Engenharia de Sistemas e Informação

Function calls

id	name	line	start	finish	caller_id	trial_id
1	/Users/leo...	58	2013-11-01 ...	2013-11-01 ...		1
2	csv_read	43	2013-11-01 ...	2013-11-01 ...	1	1
3	open	17	2013-11-01 ...	2013-11-01 ...	2	1
4	reader	17	2013-11-01 ...	2013-11-01 ...	2	1
5	list.append	20	2013-11-01 ...	2013-11-01 ...	2	1

...

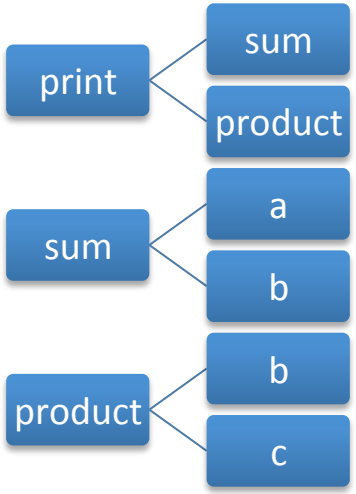
File accesses

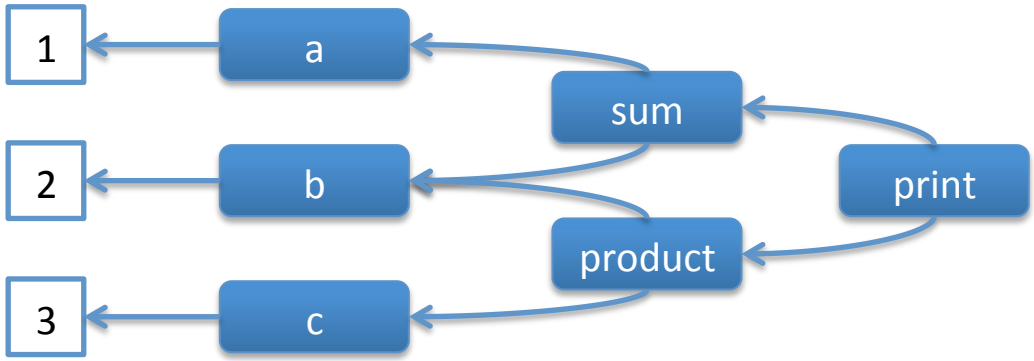
id	name	mode	buffering	content_hash_before	content_hash_after	timestamp	function_call_id	trial_id
1	data1.dat	rU	default	28f4192700d9e5d...	28f4192700d9e5...	2013-11-...	3	1
2	data2.dat	rU	default	802a73cb49af958...	802a73cb49af95...	2013-11-...	187	1
3	output.png	wb	default		a305a09040143f...	2013-11-...	1102	1

Execution provenance (data dependencies)

```
a = 1
b = 2
c = 3

sum = a + b
product = b + c

print(sum, product)
```

Program slicing





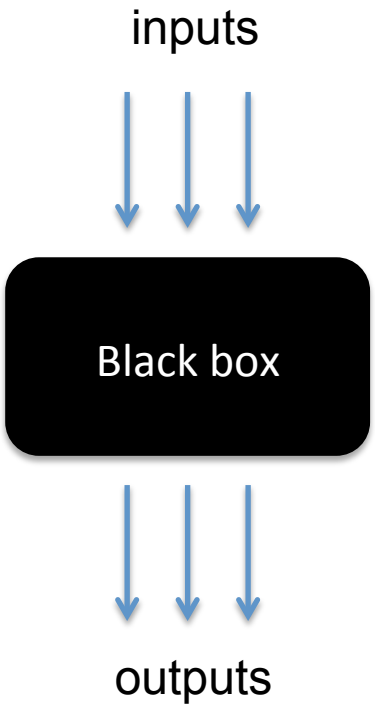
Challenges and Opportunities

Don't limit your challenges

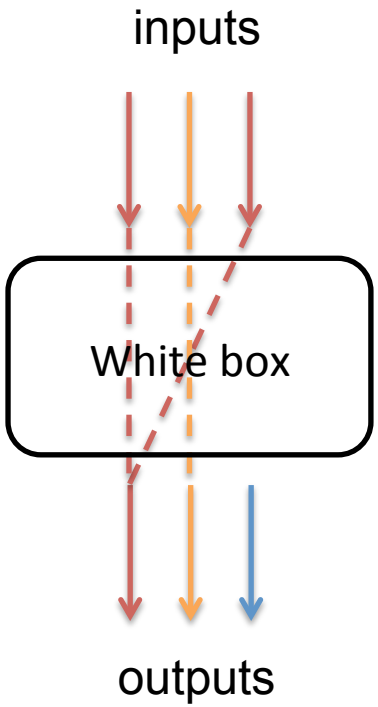
Challenge your limits!

Jerry Dunn

Efficacy vs. efficiency



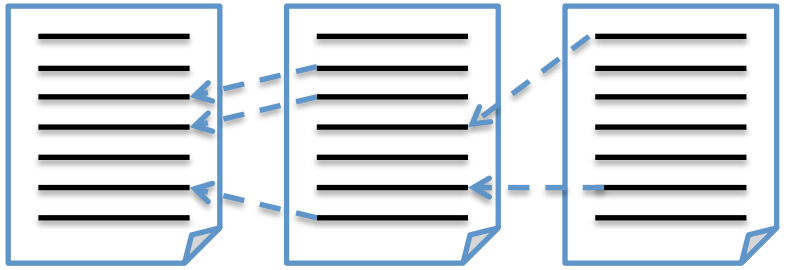
Fast
False-positives

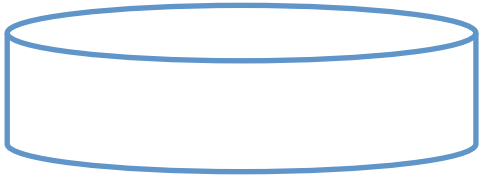


Slow
Precise

Additional provenance sources

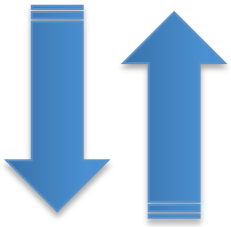
White-box files



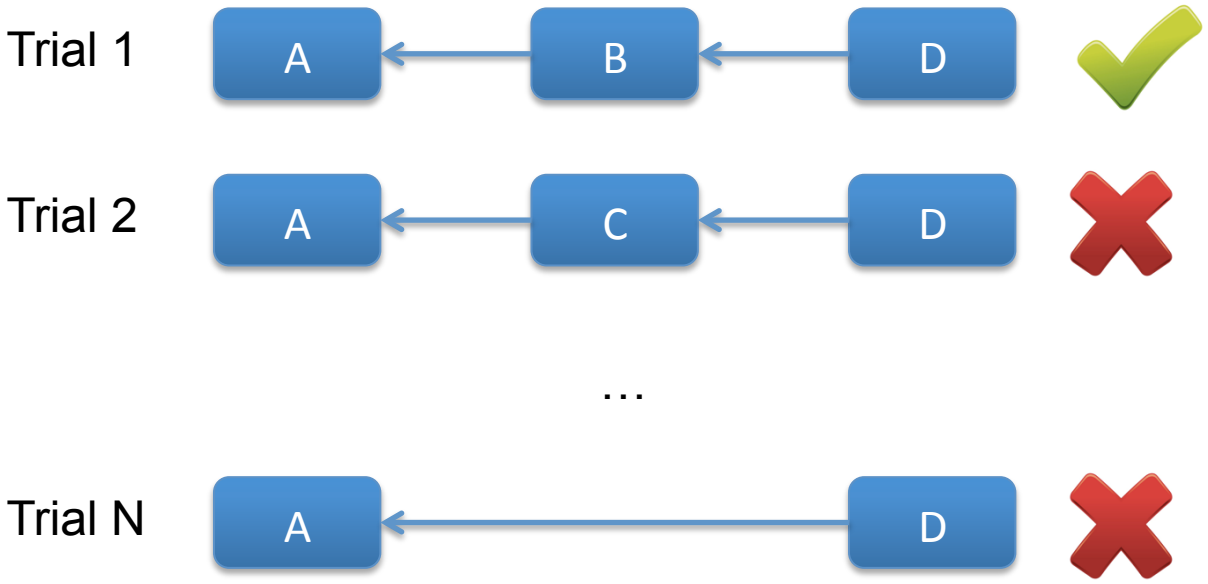


Databases

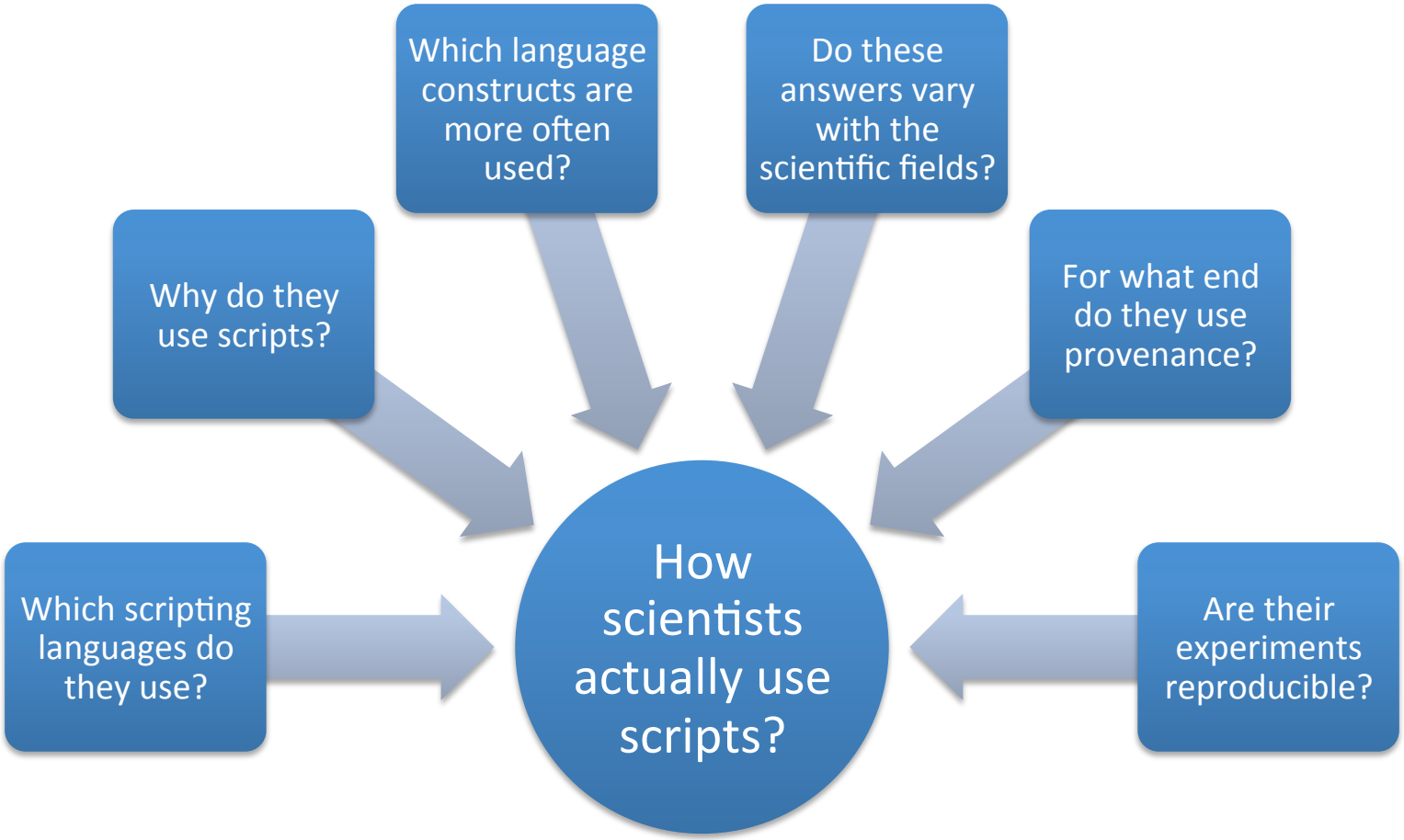
Network



Fine-grained multi-trial analysis

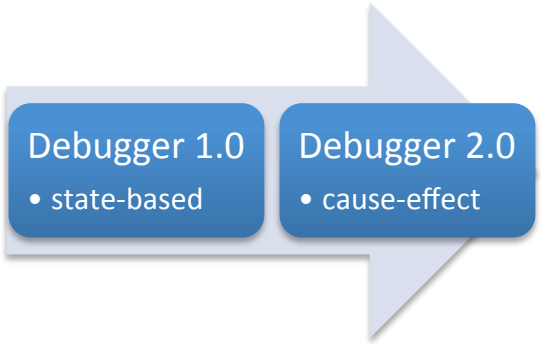


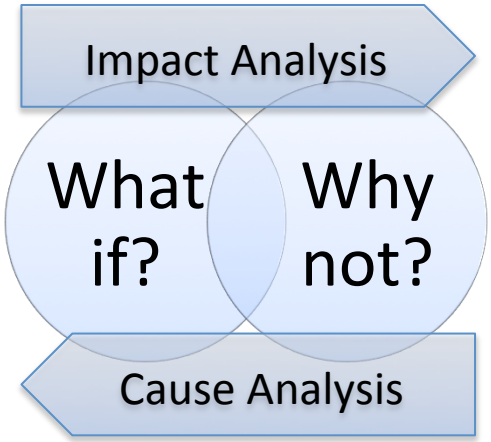
Archaeology



New applications

Debugging



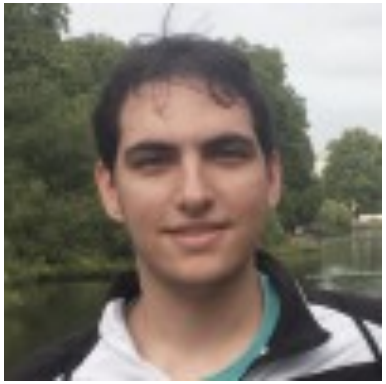


Education

Cyber security

Acknowledgments

noWorkflow main team



João Felipe Pimentel
IC/UFF

Vanessa Braganholo
IC/UFF

Leonardo Murta
IC/UFF

Juliana Freire
NYU

Other collaborators

- Bertram Ludäscher (U. Illinois at Urbana-Champaign)
- David Koop (UMass Dartmouth)
- Fernando Chirigati (NYU)
- Khalid Belhajjame (U. Paris Dauphine)
- Paolo Missier (Newcastle U.)
- Saumen Dey (UC Davis)
- Timothy McPhillips (U. Illinois at Urbana-Champaign)

Funding agencies

Provenance Gathering from Scripts: Challenges and Opportunities

Please, answer our survey at
https://survey.npimentel.net
and download noWorkflow at
http://gems-uff.github.io/noworkflow

Leonardo Gresta Paulino Murta
leomurta@ic.uff.br

Instituto de Computação

Laboratório de Engenharia de Sistemas e Informação