Collecting and Analyzing Provenance on Interactive Notebooks: when IPython meets noWorkflow

Instituto de computação uff

João Felipe Nicolaci Pimentel (UFF),

Juliana Freire (NYU), Leonardo Murta (UFF),

Vanessa Braganholo (UFF)

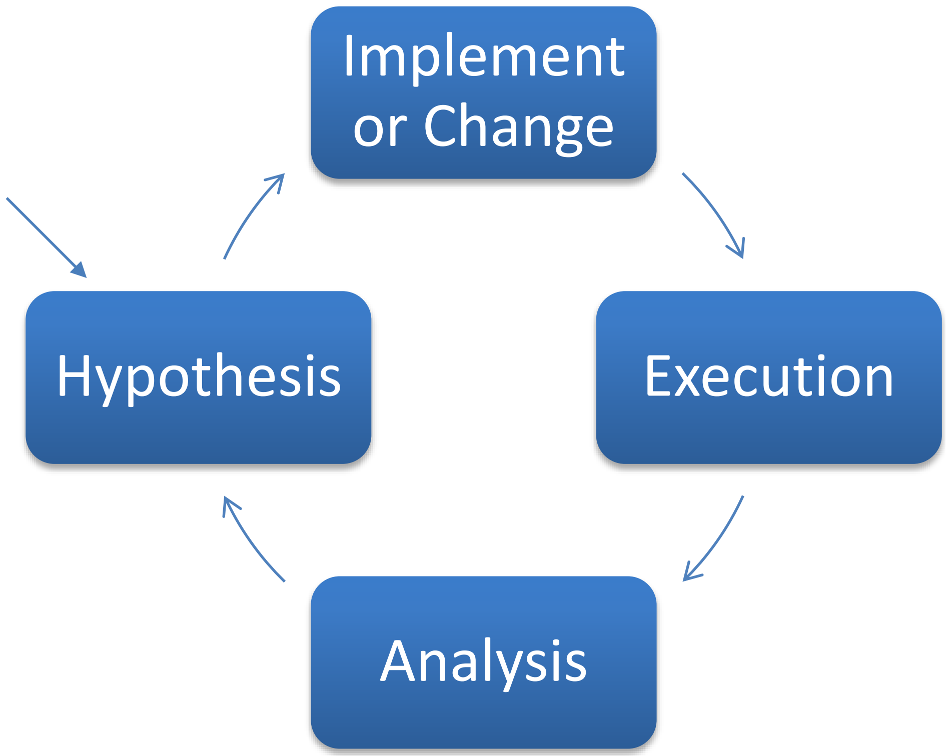
NYU POLYTECHNIC SCHOOL OF ENGINEERING

Outline

- Motivation
  - Exploratory research
  - Example
  - Interactive Notebooks
  - Example
  - Provenance Limitations
- Approach
  - Provenance Collection
  - Provenance Analysis
- Conclusion

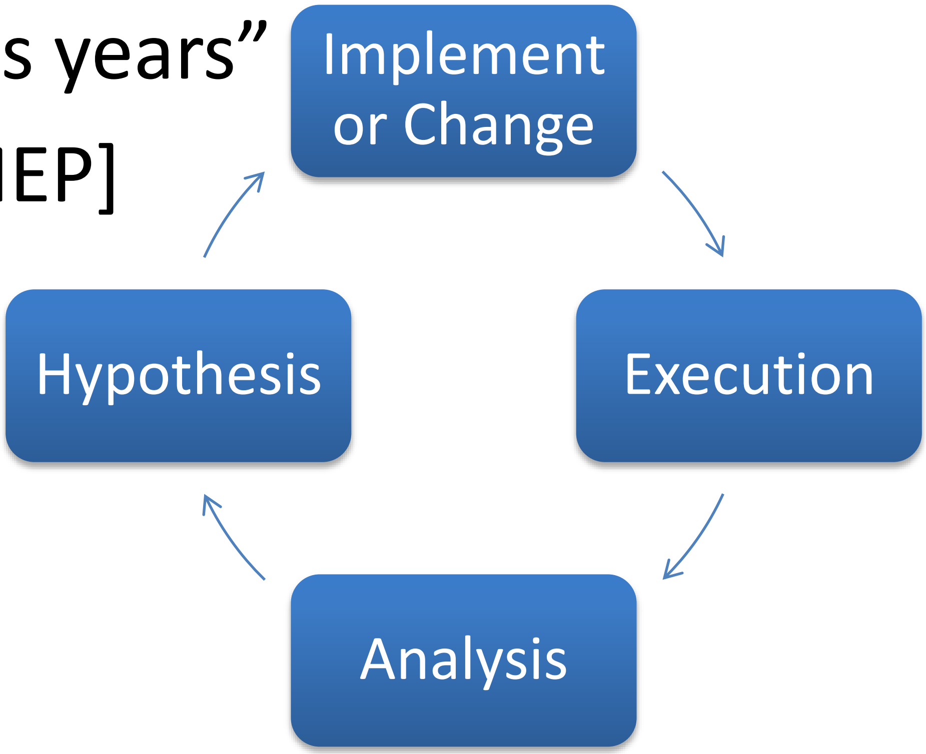
Motivation

Exploratory research



Example of exploratory research

- Analyze precipitation data from Rio de Janeiro
- Hypothesis: “The precipitation for each month remains constant across years”
- Data: 2013, 2014 [BDMEP]

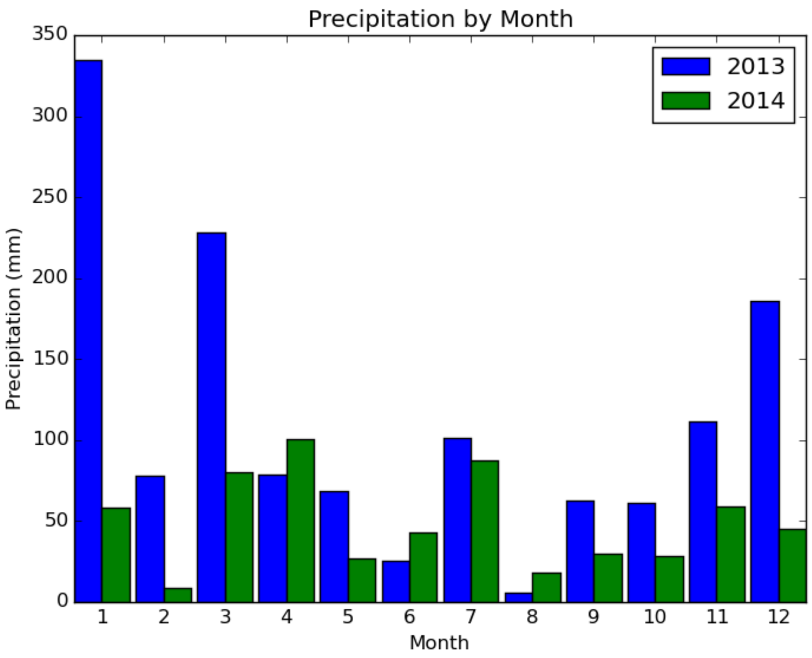


```python
import numpy as np
import matplotlib.pyplot as plt
from precipitation import read, prepare

def bar_graph(years):
    global PREC, MONTHS
    prepare(PREC, MONTHS, years, plt)
    plt.savefig("out.png")

MONTHS = np.arange(12) + 1
d13, d14 = read('p13.dat'), read('p14.dat')
PREC = prec13, prec14 = [], []
for i in MONTHS:
    prec13.append(sum(d13[i]))
    prec14.append(sum(d14[i]))
bar_graph(['2013', '2014'])
```

1st Iteration



- $ python experiment.py
- $ display out.png
- Analysis: “Drought in 2014”
- New Hypothesis:

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

- Data: 2012, 2013, 2014 [BDMEP]

2nd Iteration

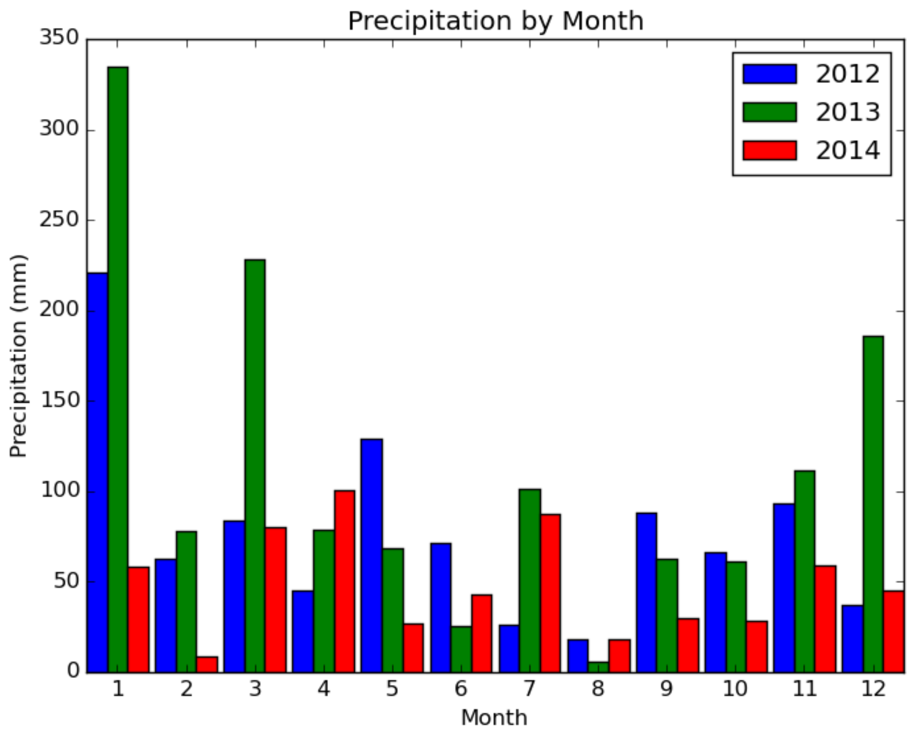
```python
MONTHS = np.arange(12) + 1
d12 = read('p12.dat')
d13, d14 = read('p13.dat'), read('p14.dat')
PREC = prec12, prec13, prec14 = [], [], []

for i in MONTHS:
    prec12.append(sum(d12[i]))
    prec13.append(sum(d13[i]))
    prec14.append(sum(d14[i]))

bar_graph(['2012', '2013', '2014'])
```

2nd Iteration

- $ python experiment.py
- $ display out.png
- Analysis: “2012 was similar to 2013”
- Cycle continues



Interactive Notebooks

- Documents
  - Text, code, plots, rich media
  - Share the documents with results
- Most famous:
  - IPython Notebook, knitr
- IPython Notebook has more than 500,000 active users
- Good for exploratory research

Example - 1st Iteration

Initialize Experiment

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from precipitation import read, prepare
        %matplotlib inline
        "Initialized"

Out[1]: 'Initialized'
```

Bar graph plot function

- `PREC` contains the precipitation data
- `MONTHS` defines the interval
- `years` is a list of legends

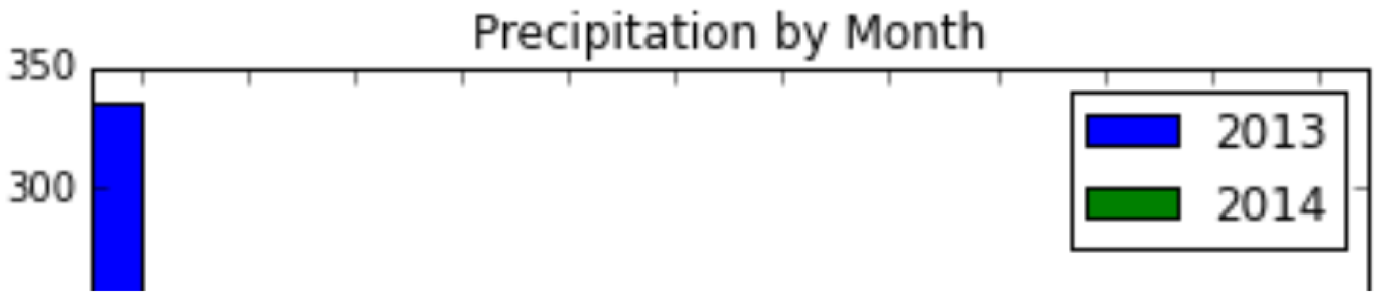
In [2]:
```
def bar_graph(years):
    global PREC, MONTHS
    prepare(PREC, MONTHS, years, plt)
    plt.savefig("out.png")
```

Experiment

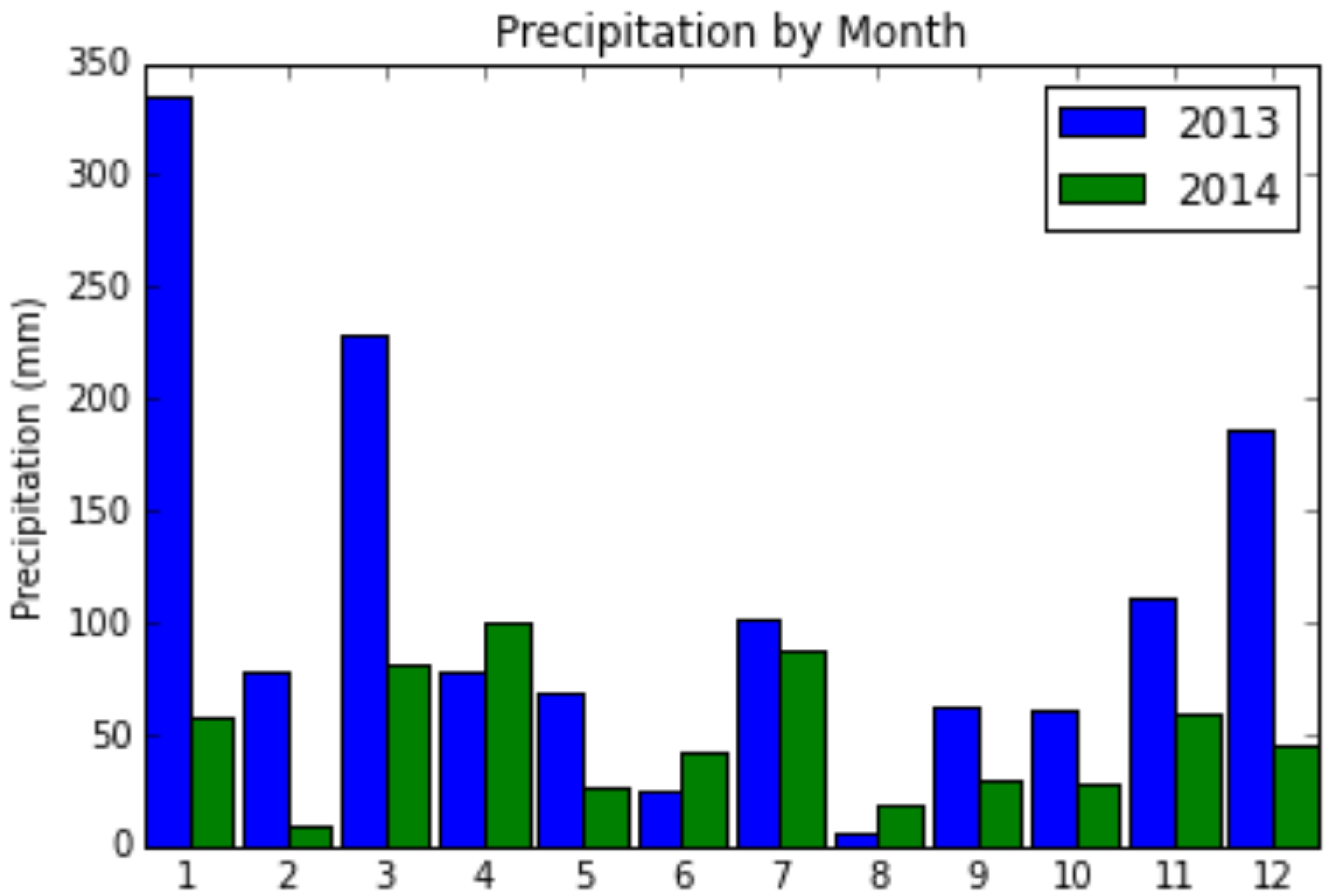
```
In [3]: MONTHS = np.arange(12) + 1
        d13, d14 = read('p13.dat'), read('p14.dat')
        PREC = prec13, prec14 = [], []

        for i in MONTHS:
            prec13.append(sum(d13[i]))
            prec14.append(sum(d14[i]))
```

```
In [4]: bar_graph(['2013', '2014'])
```



In [4]:
```
bar_graph(['2013', '2014'])
```

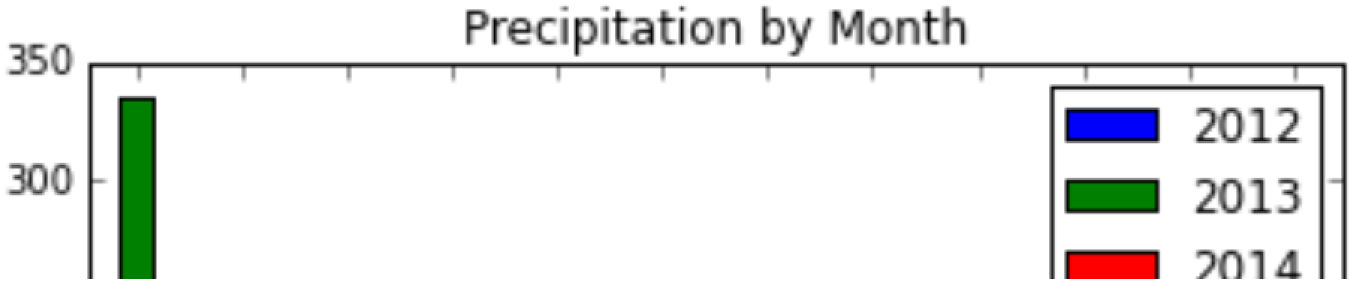


Experiment

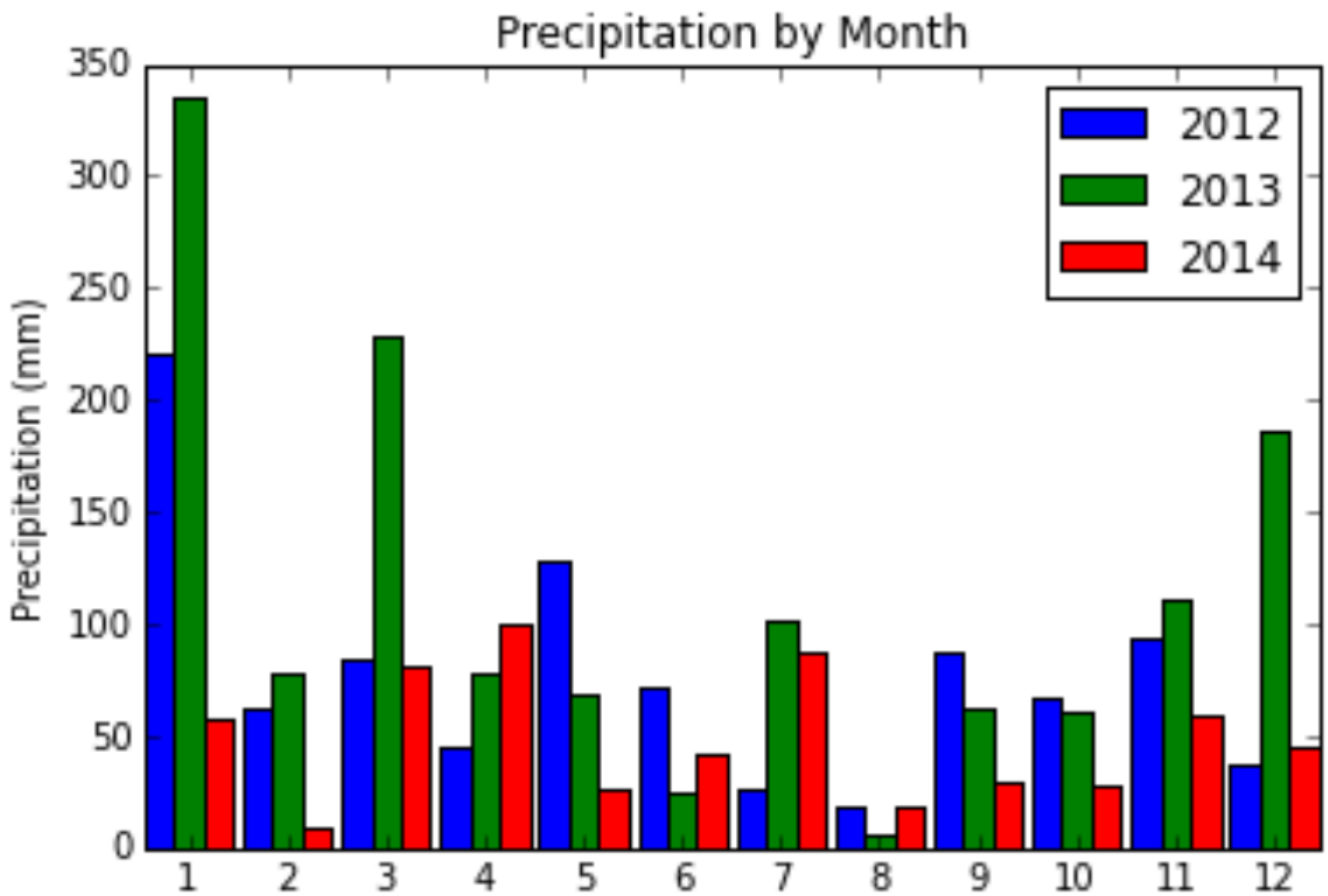
In [5]:
```
MONTHS = np.arange(12) + 1
d12 = read('p12.dat')
prec12 = []
PREC = prec12, prec13, prec14

for i in MONTHS:
    prec12.append(sum(d12[i]))
```

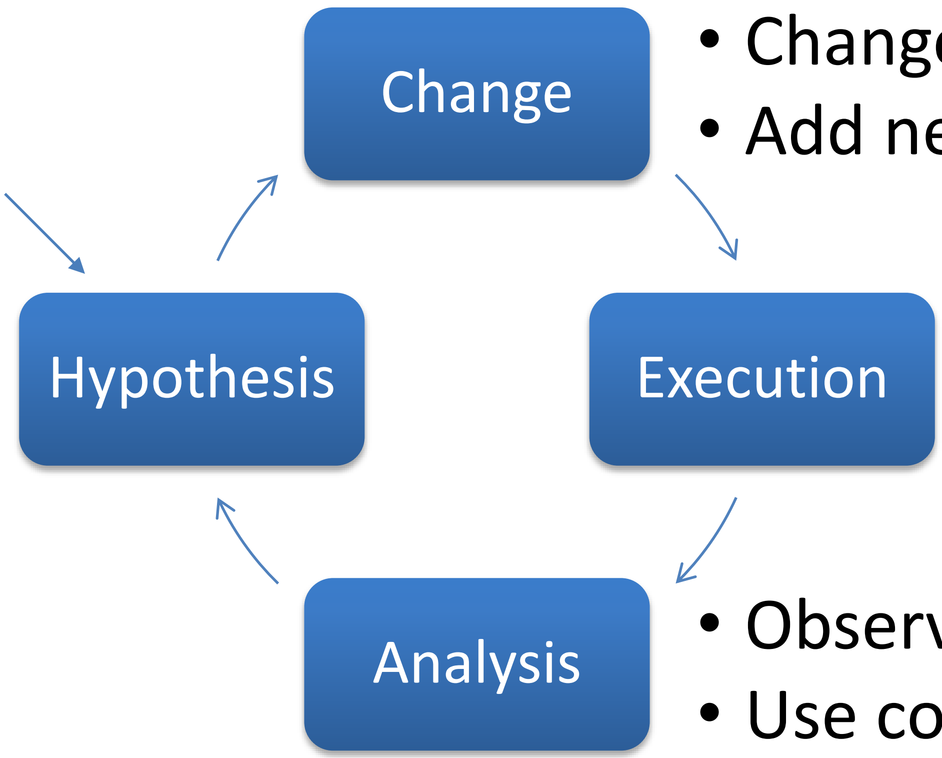
In [6]:
```
bar_graph(['2012', '2013', '2014'])
```



```
In [6]: bar_graph(['2012', '2013', '2014'])
```



Exploratory research



- Change existing cells
- Add new one

- Execute cells

- Observe the results
- Use code for analysis

Interactive Notebooks

Provenance Limitations

1. Which version of matplotlib, numpy and precipitation module is it using? Will it work on another environment?

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from precipitation import read, prepare
        %matplotlib inline
        "Initialized"

Out[1]: 'Initialized'
```

Provenance Limitations

2. The cell [3] were replaced by [5]. Where did the “prec13” and “prec14” come from? What changed from [3] to [5]?

In [5]:
```
MONTHS = np.arange(12) + 1
d12 = read('p12.dat')
prec12 = []
PREC = prec12, prec13, prec14

for i in MONTHS:
    prec12.append(sum(d12[i]))
```

Provenance Limitations

3. What happens inside the cell? What does read('p12.dat') return? How long did it take to execute each function?

In [5]:
```
MONTHS = np.arange(12) + 1
d12 = read('p12.dat')
prec12 = []
PREC = prec12, prec13, prec14

for i in MONTHS:
    prec12.append(sum(d12[i]))
```

Provenance Limitations

4. What is the content of 'p12.dat'?

```
In [5]: MONTHS = np.arange(12) + 1
        d12 = read('p12.dat')
        prec12 = []
        PREC = prec12, prec13, prec14

        for i in MONTHS:
            prec12.append(sum(d12[i]))
```

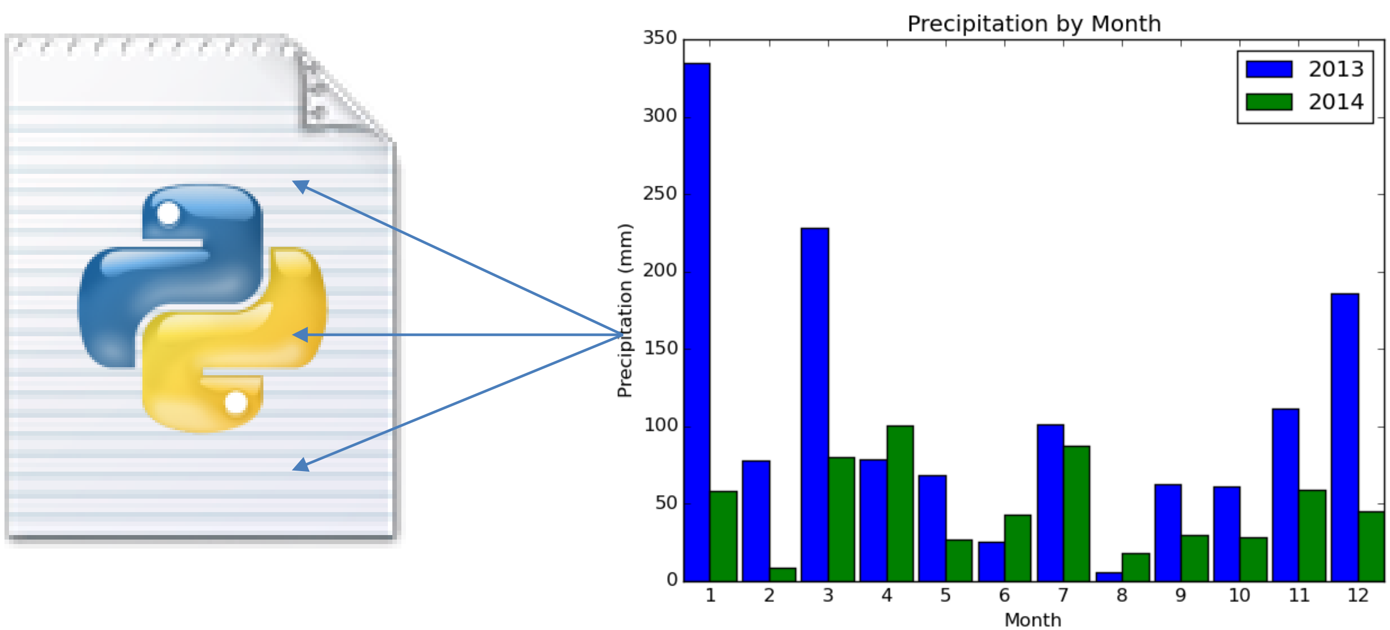
Provenance of Python Scripts

- API: (Bochner; Gude; Schreiber, 2008)
  - Require API calls
- StarFlow (Angelino; Yamins; Seltzer, 2010)
  - Require annotations
- Sumatra (Davison, 2012)
  - Require version control system
- **noWorkflow** (Murta et al., 2014)
  - Transparent collection

- None supports notebooks

noWorkflow

- **Transparently** captures provenance of Python scripts – no changes required!
- Allows users to analyze provenance data



```python
import numpy as np
import matplotlib.pyplot as plt
from precipitation import read, prepare

def bar_graph(years):
    global PREC, MONTHS
    prepare(PREC, MONTHS, years, plt)
    plt.savefig("out.png")

MONTHS = np.arange(12) + 1
d13, d14 = read('p13.dat'), read('p14.dat')
PREC = prec13, prec14 = [], []
for i in MONTHS:
    prec13.append(sum(d13[i]))
    prec14.append(sum(d14[i]))
bar_graph(['2013', '2014'])
```

1.9.2

1.4.3

1.0.1

PATH = /home/joao/...

PYTHON_VERSION = 2.7.6

```python
import numpy as np
import matplotlib.pyplot as plt
from precipitation import read, prepare

def bar_graph(years):
    global PREC, MONTHS
    prepare(PREC, MONTHS, years, plt)
    plt.savefig("out.png")

MONTHS = np.arange(12) + 1
d13, d14 = read('p13.dat'), read('p14.dat')
PREC = prec13, prec14 = [], []
for i in MONTHS:
    prec13.append(sum(d13[i]))
    prec14.append(sum(d14[i]))
bar_graph(['2013', '2014'])
```

PREC = [[...], [...]]

MONTHS = array(1,...,12)

p13.dat content b/a

p14.dat content b/a

read('p14.dat') -> {1: [7.1, 0.8, 0.0, ...], 2: ...}

Objective

IP[y]:

Interactive

noWorkflow

Provenance

noW[y]:

Interactive + Provenance

Approach

Provenance Collection – 1 / 2

```
In [1]: %load_ext noworkflow
```

```
In [2]: trial = %now_run experiment.py
        trial.id
Out[2]: 2
```

Provenance Collection – 2 / 2

In [3]:

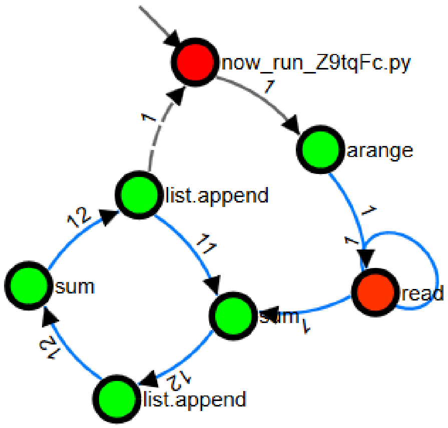
```
%%now_run --interactive
MONTHS = np.arange(12) + 1
d13, d14 = read('p13.dat'), read('p14.dat')
PREC = prec13, prec14 = [], []

for i in MONTHS:
    prec13.append(sum(d13[i]))
    prec14.append(sum(d14[i]))
```

Out[3]:

A

Trial 18. Ctrl-click to toggle nodes



Provenance Collection – 2 / 2

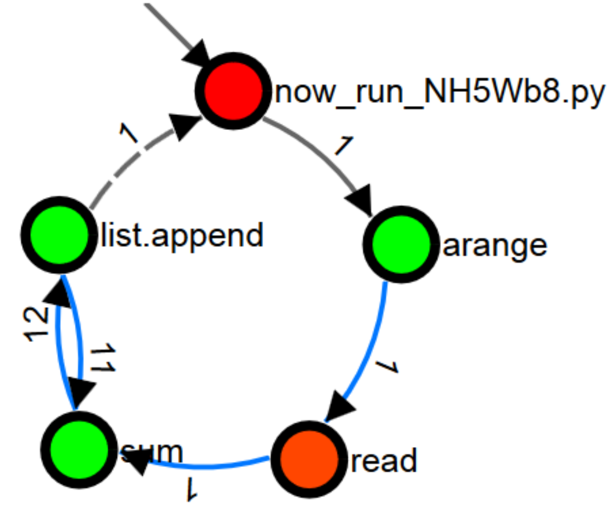
In [5]:

```
%%now_run --interactive
MONTHS = np.arange(12) + 1
d12 = read('p12.dat')
prec12 = []
PREC = prec12, prec13, prec14

for i in MONTHS:
    prec12.append(sum(d12[i]))
```

Out[5]:

Trial 19. Ctrl-click to toggle nodes



Provenance Analysis

- Call trial methods and properties
- Load Trial
- Trial visualization
- Perform SQL queries
- Perform Prolog queries
- Read file content before and after writing
- Advanced Analysis

Call Trial Method

1. Which version of precipitation module is it using?

```
In [2]: %%now_run --interactive
        import numpy as np
        import matplotlib.pyplot as plt
        from precipitation import read, prepare
```

```
...
```

```
In [3]: _.modules(find='precipitation')

Out[3]: OrderedDict([('id', 1085), ('name', u'precipit
        ation'), ('version', u'1.0.1'), ('path', u'/ho
        me/joao/projects/tapp_presentation/precipitati
        on.py'), ('code_hash', u'2490ecc8370cf879b46e3
        e7dc91b47790e57359a')])
```

Load Trial

2. What changed from [3] to [5]?

In [2]:
```
nip = %now_ip
w = 'w'
trial18 = nip.Trial(18)
trial19 = nip.Trial(19)
open('.18', w).write(str(trial18.script_content))
open('.19', w).write(str(trial19.script_content))
!diff .18 .19 | colordiff
```

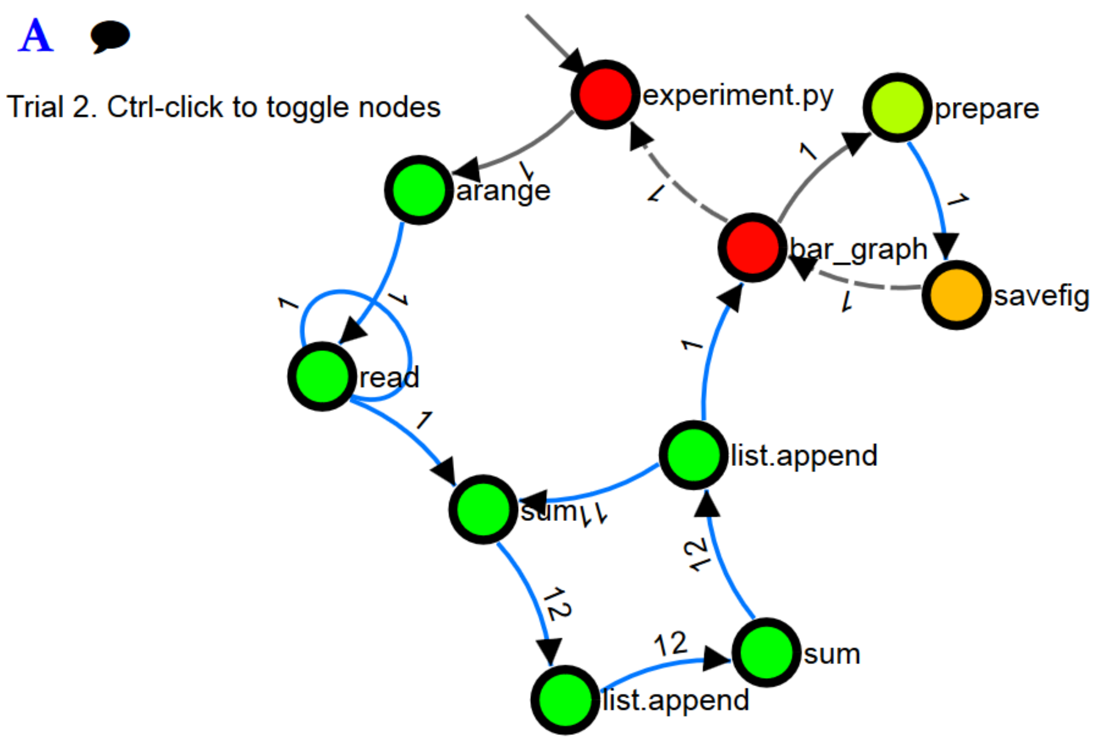
```
2,3c2,4
< d13, d14 = read('p13.dat'), read('p14.dat')
< PREC = prec13, prec14 = [], []
---
> d12 = read('p12.dat')
> prec12 = []
> PREC = prec12, prec13, prec14
6,7c7
<       prec13.append(sum(d13[i]))
```

Provenance Visualization

3. What happened inside the cell?

```
In [3]: trial
```

Out[3]:



SQL Queries

In [5]:

```
%%now_sql
SELECT name, content_hash_before
FROM file_access
WHERE trial_id = 2
AND name IN ("p13.dat", "p14.dat")
```

Out[5]:

name	content_hash_before
p13.dat	9418519556e2bca25481158e60a82d62c20ba54e
p14.dat	65f35fc7e0e6862c1344aa18016ff4dcbadb0db9

Prolog Queries

```
In [7]: %%now_prolog {trial.id}
        indirect_activation({trial.id}, bar_graph, X),
        duration({trial.id}, X, Y)

Out[7]: [{u'X': u'prepare', u'Y': 0.3820490837097168},
         {u'X': u'savefig', u'Y': 0.6897439956665039}]
```

Read file content

4. What is the content of 'p12.dat'?

```
In [7]: print(
nip.persistence.get('9138b1e2c0f6b80ab7ac902835e7bc88ea585c7a')
)
```

```
83743;01/01/2012;1200;7.8;
83743;02/01/2012;1200;44.2;
83743;03/01/2012;1200;30.6;
83743;04/01/2012;1200;0;
83743;05/01/2012;1200;0;
83743;06/01/2012;1200;0;
83743;07/01/2012;1200;46.2;
83743;08/01/2012;1200;0;
83743;09/01/2012;1200;2.2;
83743;10/01/2012;1200;0.1;
83743;11/01/2012;1200;0.8;
```

Advanced Analysis

- Combine:
  - Python code
  - SQL queries
  - Prolog queries
  - File content
  - External tools

Limitations

- Capture one cell at a time.
  - It is necessary to repeat “%%now_run” for every cell
- No Out [x]
  - The Out [x] is replaced by the trial object
- No IPython superset
  - It is not possible to invoke other special commands

Related Work

- Ducktape (Wibisono et al., 2014)
  - Use notebook only for interactive provenance visualization
- Lancet (Stevens, Elver, Bednar, 2013)
  - Requires definition of special launchers to capture provenance
  - Steep learning curve

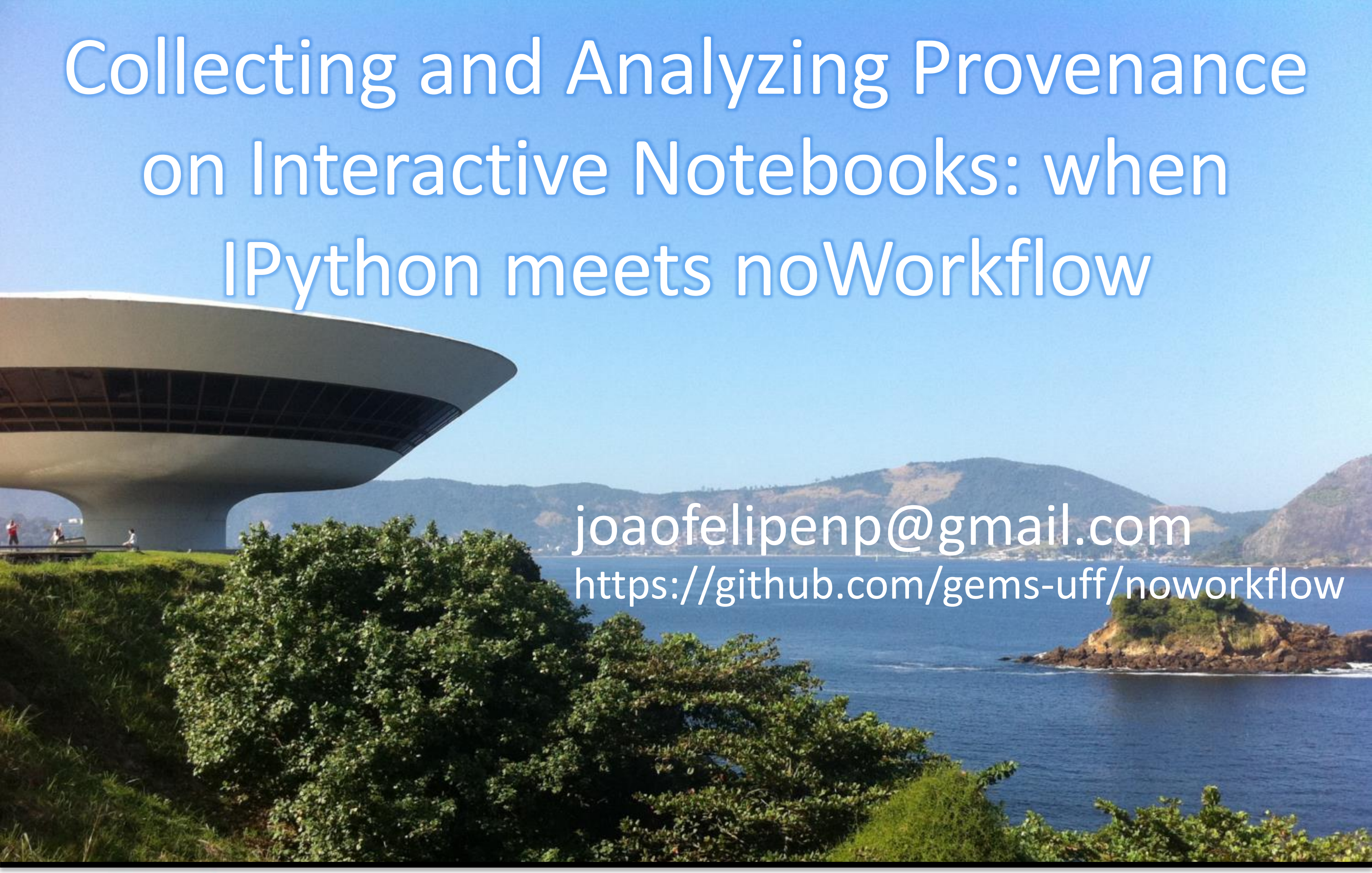
Conclusion

- Mechanism to collect and analyze provenance from IPython Notebooks
  - Invoke noWorkflow through special functions
  - Analytic tools: SQL queries, Prolog queries, object properties, graphs
- noWorkflow tracks history, environment, intermediate results and files
- Reproducible notebook!

Future Work

- New visualization methods for Provenance
  - Dependency graph
  - Diff visualization
- Integration with Pandas
  - Improve data analysis
- Collect provenance from other languages supported by Jupyter

Collecting and Analyzing Provenance on Interactive Notebooks: when IPython meets noWorkflow



joaofelipenp@gmail.com
https://github.com/gems-uff/noworkflow

Instituto de computação

João Felipe Nicolaci Pimentel (UFF),
Juliana Freire (NYU), Leonardo Murta (UFF),
Vanessa Braganholo (UFF)

NYU
POLYTECHNIC SCHOOL OF ENGINEERING

SQL Schema

```
In [2]: %now_sql_schema

Out[2]: create table trial (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                start TIMESTAMP,
                finish TIMESTAMP,
                script TEXT,
                code_hash TEXT,
                arguments TEXT,
                inherited_id INTEGER, -- Id of the
        prospective tuple that we are inheriting
        module information (due to --bypass-modules)
                parent_id INTEGER, -- Id of the
        parent trial that is used to create the
```

Prolog Schema

```
In [3]: %now_prolog_schema

Out[3]: %
        % FACT: activation(trial_id, id, name, start,
        finish, caller_activation_id).
        %



        %
        % FACT: access(trial_id, id, name, mode, conte
        nt_hash_before, content_hash_after, timestamp,
         activation_id).
        %



        %
```

Complex Analysis

```
In [4]: def extract(trial_id, filename, month):
            t = nip.Trial(trial_id)

            sql = first(nip.persistence.query("""
                SELECT name, content_hash_before
                FROM file_access
                WHERE trial_id = {}
                AND name = "{}"
            """.format(trial_id, filename)))
            fhash = sql['content_hash_before']

            content = nip.persistence.get(fhash)
            with open('.temp.dat', 'w') as f:
                f.write(content)

            result = !./precipitation.py .temp.dat $month
            return sum(map(float,result[0].split(';')))
```

Complex Analysis

```
In [5]: extract(18, 'p13.dat', 2)

Out[5]: 78.1
```