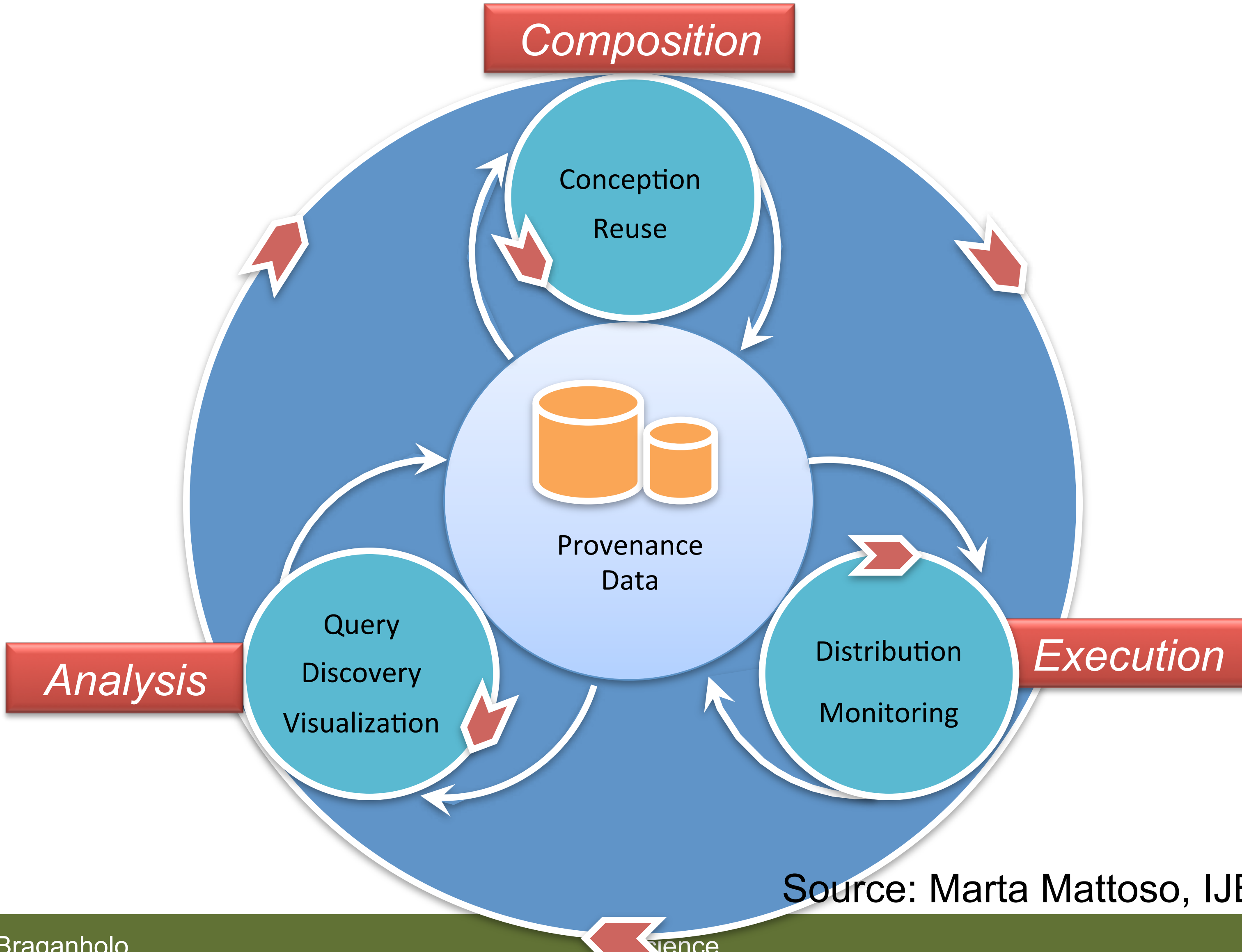
Scientific Experiments as Workflows and Scripts

Instituto de computação uff

Vanessa Braganholo

The experiment life cycle



Source: Marta Mattoso, IJBPIM, 2010

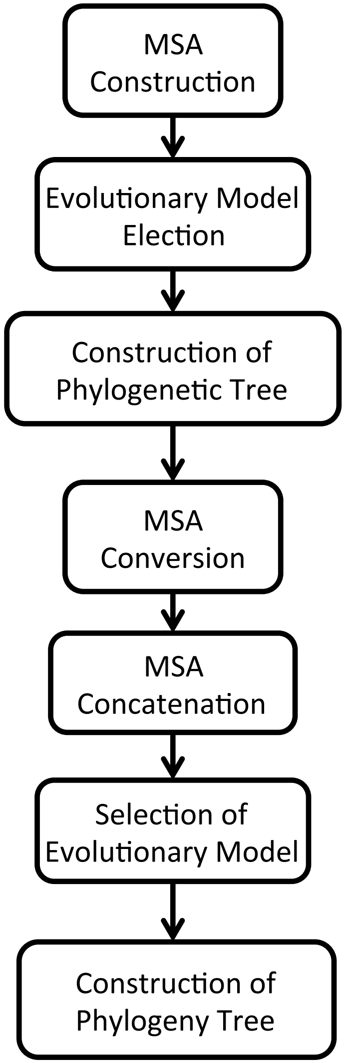
Agenda

- Abstract Representation of Scientific Experiments
- Workflows
- Scripts
- Black Boxes X White Boxes
- Workflow Management Systems
- Provenance Management Systems for Scripts

Composition: Conceiving Scientific Experiments

- Scientists usually design an experiment using a **high abstraction level representation** that is later mapped into a workflow or script

Phylogeny Analysis Experiment (Abstract Workflow)

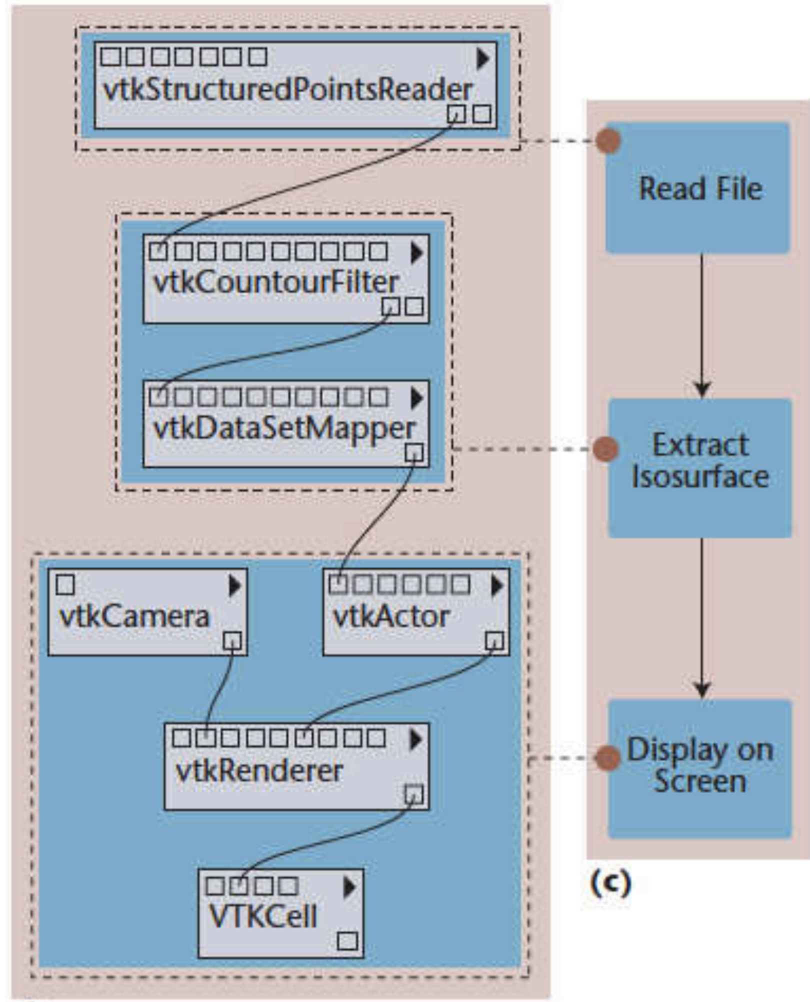


Source: MARINHO et al. Deriving scientific workflows from algebraic experiment lines: A practical approach. Future Generation Computer Systems, v. 68, p. 111-127, 2017.

Abstract x Concrete

- The **abstract** workflow is later mapped into a **concrete** workflow or script

```
import vtk

data = vtk.vtkStructuredPointsReader()
data.setFileName("../../../examples/data/head.120.vtk")

contour = vtk.vtkContourFilter()
contour.SetInput(0, data.GetOutput())
contour.SetValue(0, 67)

mapper = vtk.vtkPolyDataMapper()
mapper.SetInput(contour.GetOutput())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)

cam = vtk.vtkCamera()
cam.SetViewUp(0,0,-1)
cam.SetPosition(745,-453,369)
cam.SetFocalPoint(135,135,150)
cam.ComputeViewPlaneNormal()

ren = vtk.vtkRenderer()
ren.AddActor(actor)
ren.SetActiveCamera(cam)
ren.ResetCamera()

renwin = vtk.vtkRenderWindow()
renwin.AddRenderer(ren)

style = vtk.vtkInteractorStyleTrackballCamera()
iren = vtk.vtkRenderWindowIneractor()
iren.SetRenderWindow(renwin)
iren.SetInteractorStyle(style)
iren.Initialize()
iren.Start()
```



(b)

Source: Freire et al., 2008. Provenance for Computational Tasks: A Survey.

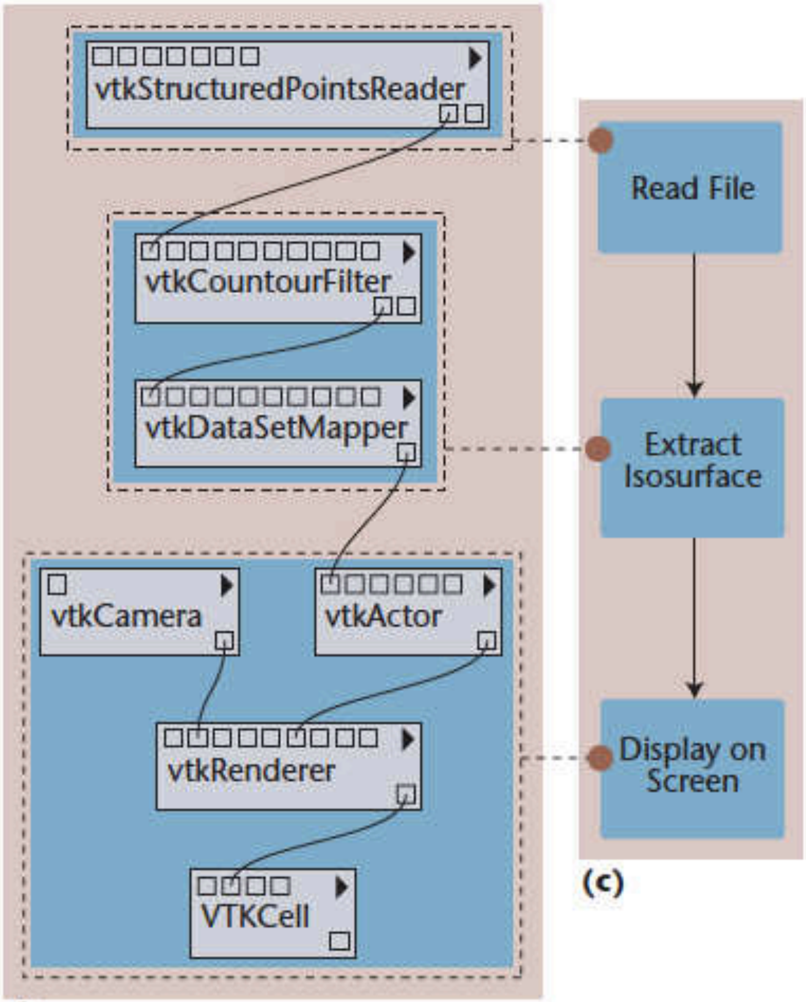
Scientific Workflow

- A scientific workflow is a **chain of activities** organized in the form of a **data flow**

Data Flow

- In a data flow, the **execution is guided by the data**
- As soon as all the input data of an activity is available, it starts executing

Example



- Activities **vtkStructuredPointsReader** and **vtkCamera** do not depend on other activities data, so they can start executing right away

Source: Freire et al., 2008. Provenance for Computational Tasks: A Survey.

Script

- Definition is controversial
- One of the most accepted definitions is that a script language is a "**programming language that does not require an explicit compilation step**"
- In other words, scripts are usually written in Languages that are interpreted instead of compiled
- Examples: Python, R, MatLab, etc.

Script

- Execution follows a **control flow** instead of a data flow
  - Commands explicitly define the execution order

```python
DRY_RUN = ...

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 = 2 ** 4000
final = process(n)
if DRY_RUN:
    final = 7
print(show(final))
```

Source: Pimentel et al., 2016. Fine-grained Provenance Collection over Scripts Through Program Slicing

Running an Experiment

- A workflow or script is just part of an experiment
- In order to prove or refute an hypothesis, it is usually necessary to run the workflow or script several times, varying inputs, parameters and programs
- Each of those runs is called a **trial** of the experiment

New experiment!

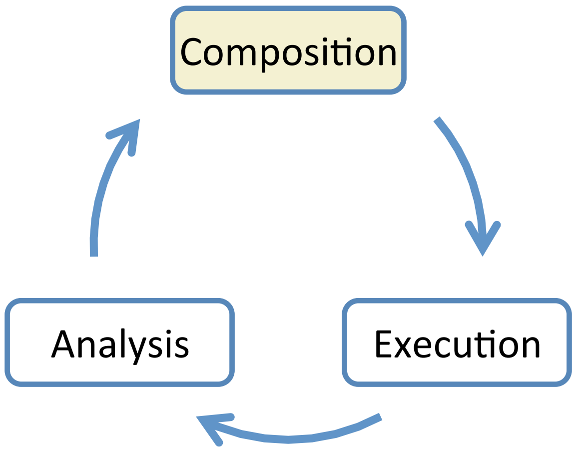
Could you check if the precipitation of Rio de Janeiro remains constant across years?

1st Iteration

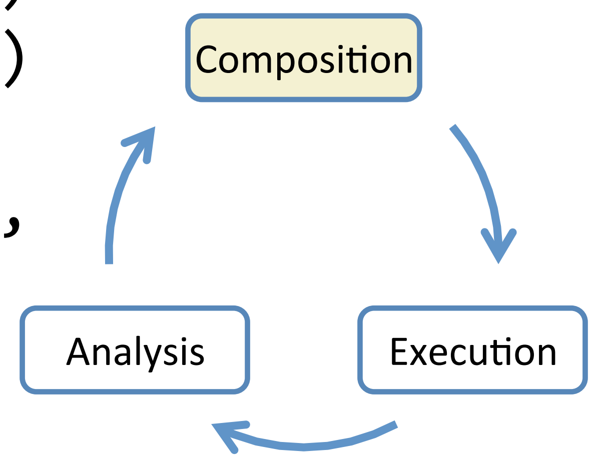
- H_1 : “The precipitation for each month remains constant across years”

Project Data: 2013, 2014 [BDMEP]

- **experiment.py**
- precipitation.py
- p13.dat
- p14.dat



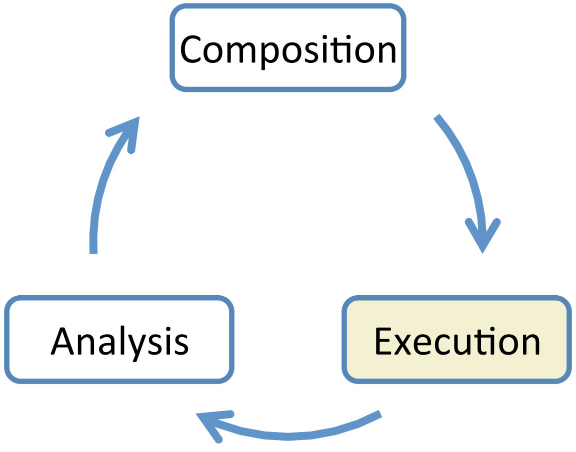
```python
import numpy as np
from precipitation import read, sum_by_month
from precipitation import create_bargraph

months = np.arange(12) + 1

d13, d14 = read("p13.dat"), read("p14.dat")

prec13 = sum_by_month(d13, months)
prec14 = sum_by_month(d14, months)

create_bargraph("out.png", months,
    ["2013", "2014"],
    prec13, prec14)
```



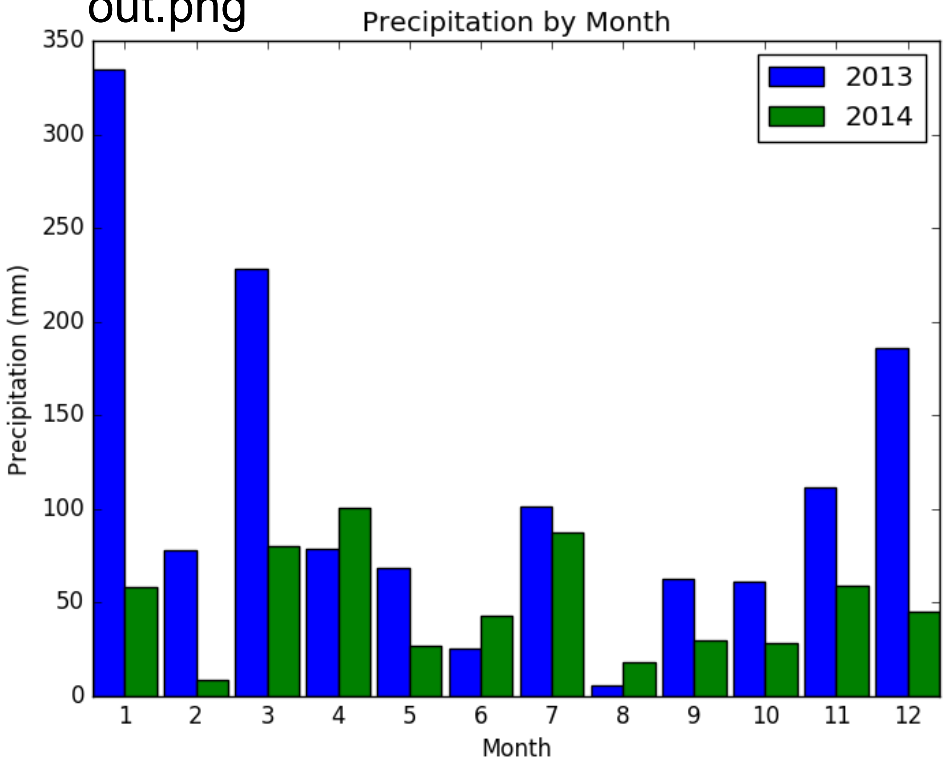
Trial

$ **now run -e Tracker** experiment.py

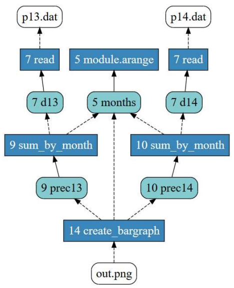
- Project
  - **experiment.py**
  - precipitation.py
  - p13.dat
  - p14.dat
  - out.png

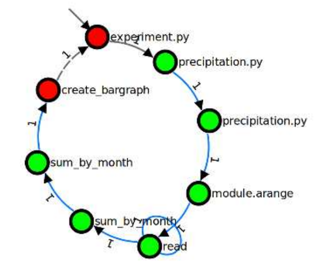


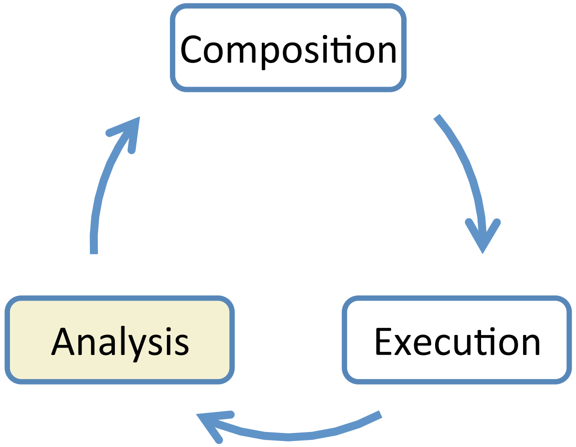
out.png



SELECT …







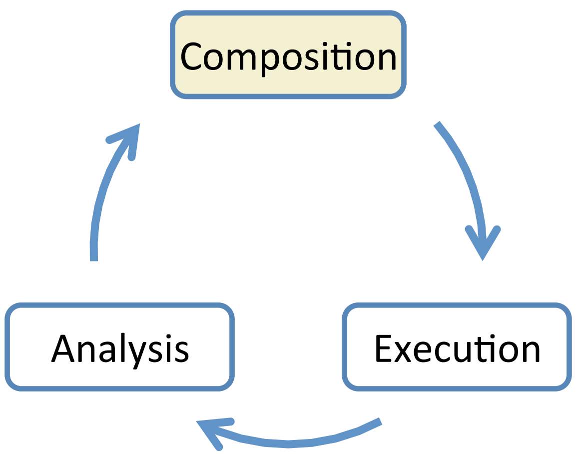
Conclusion:
"Drought in 2014"

2^{nd} Iteration

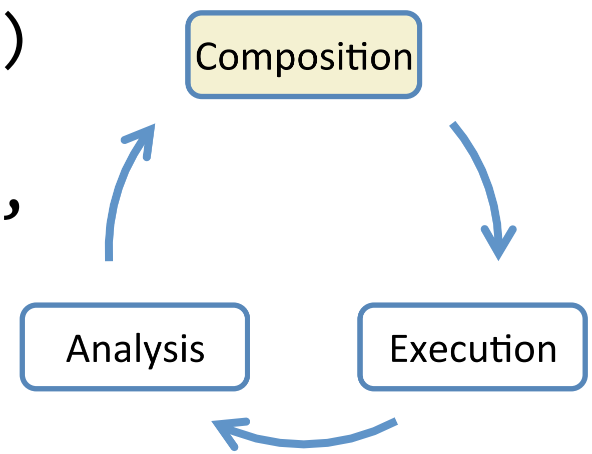
- $H{\downarrow}2$: “The precipitation for each month remains constant across years if there is no drought”

Project   Data: 2012, 2013, 2014 [BDMEP]

- **experiment.py**
- precipitation.py
- p12.dat
- p13.dat
- p14.dat

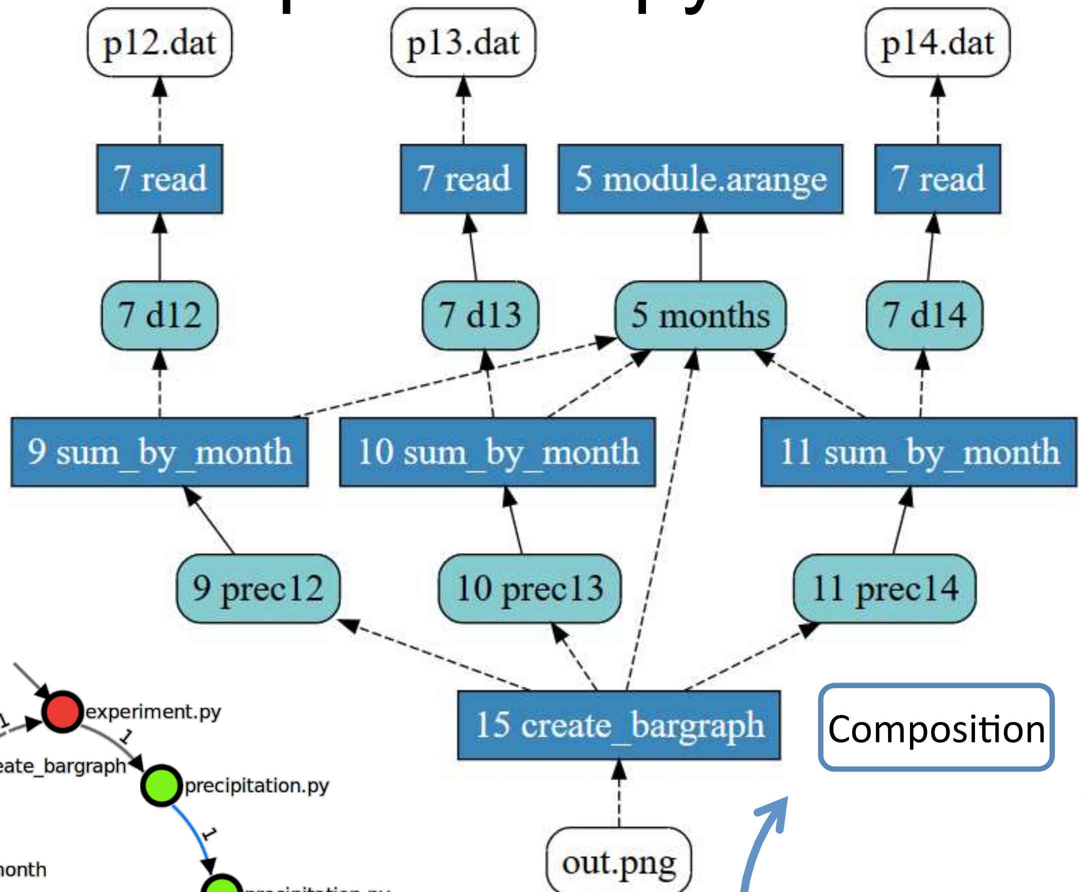


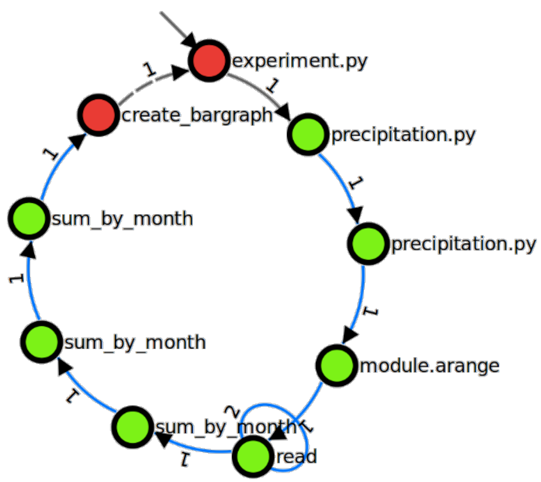
```python
import numpy as np
from precipitation import read, sum_by_month
from precipitation import create_bargraph

months = np.arange(12) + 1
d12 = read("p12.dat")
d13, d14 = read("p13.dat"), read("p14.dat")
prec12 = sum_by_month(d12, months)
prec13 = sum_by_month(d13, months)
prec14 = sum_by_month(d14, months)

create_bargraph("out.png", months,
    ["2012", "2013", "2014"],
    prec12, prec13, prec14)
```

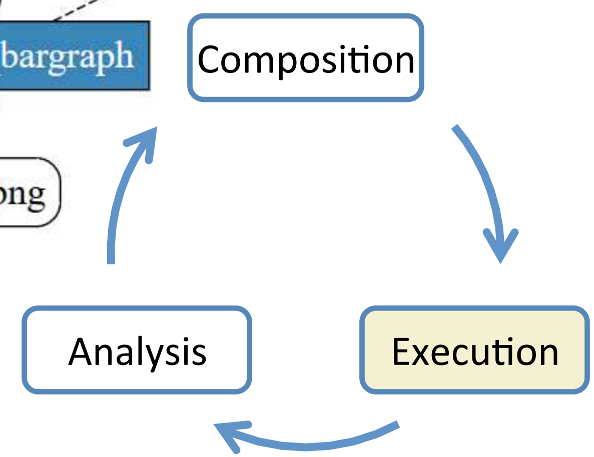


$ now run -e Tracker experiment.py

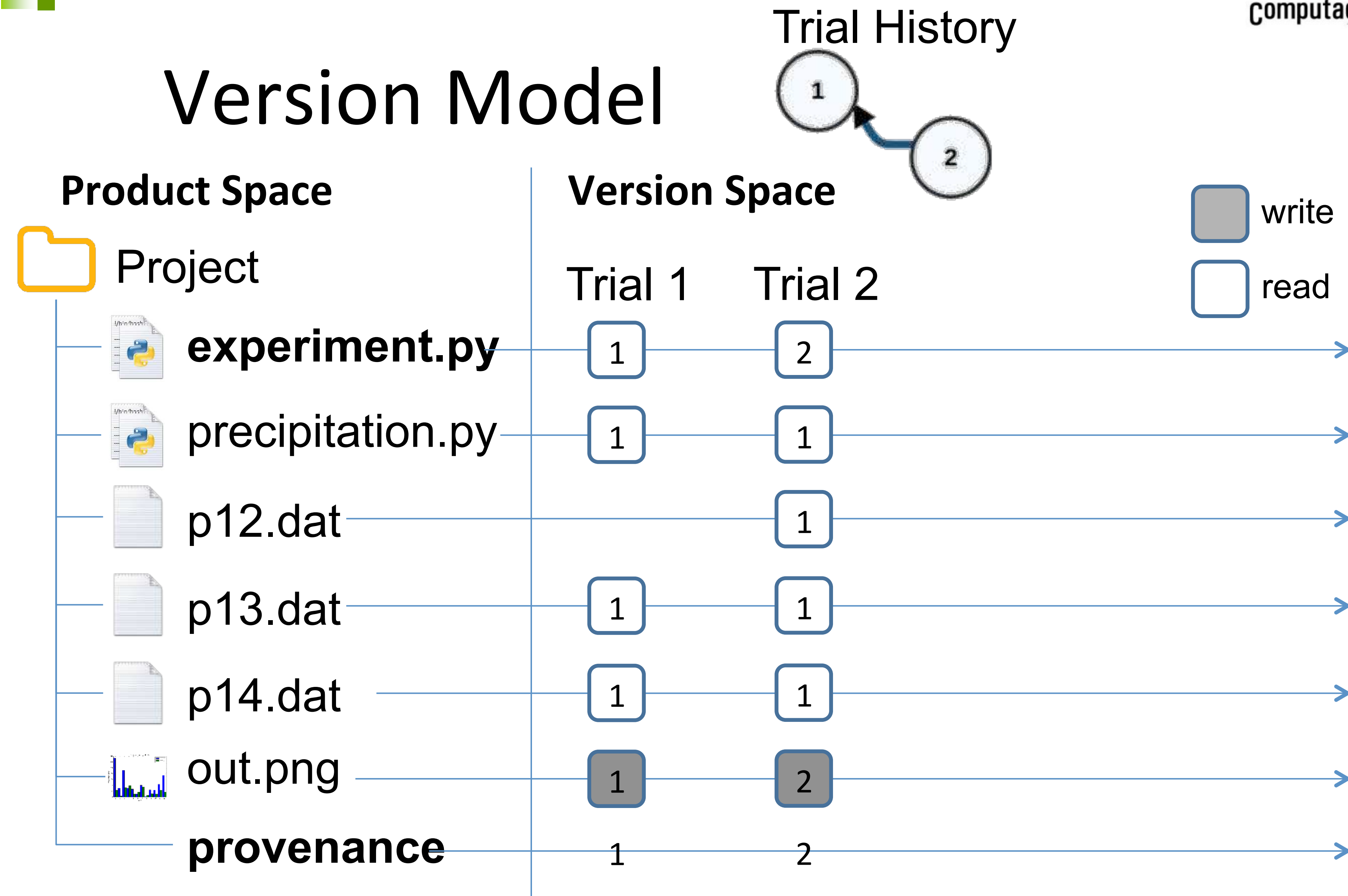
- Project
  - **experiment.py**
  - precipitation.py
  - p12.dat
  - p13.dat
  - p14.dat
  - out.png



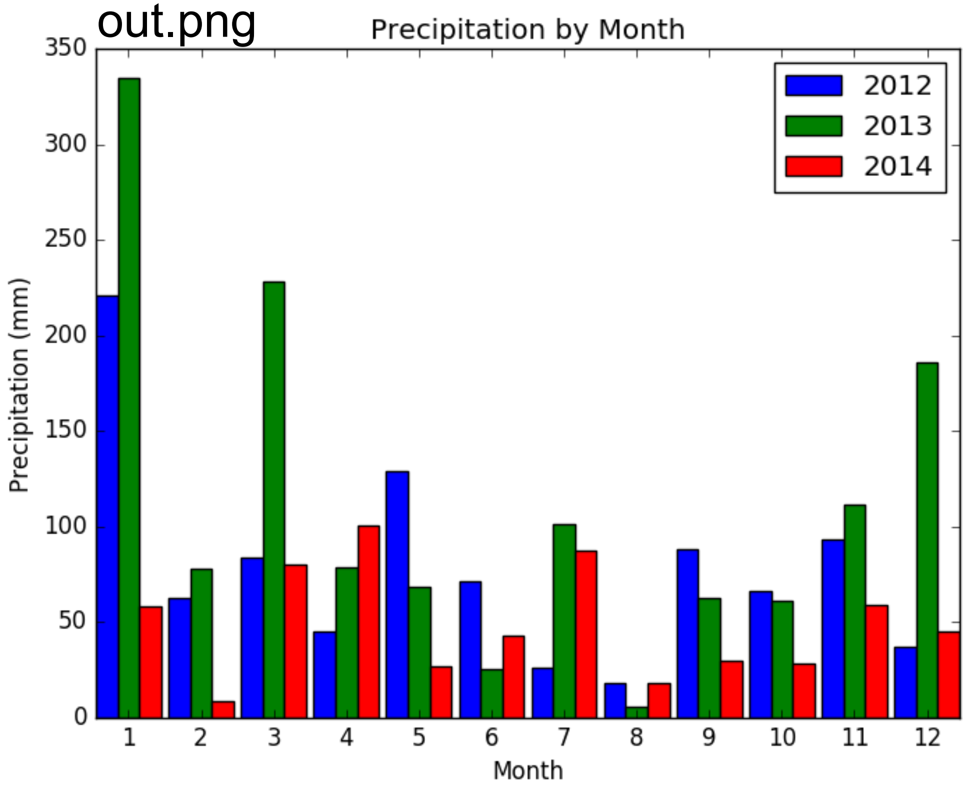




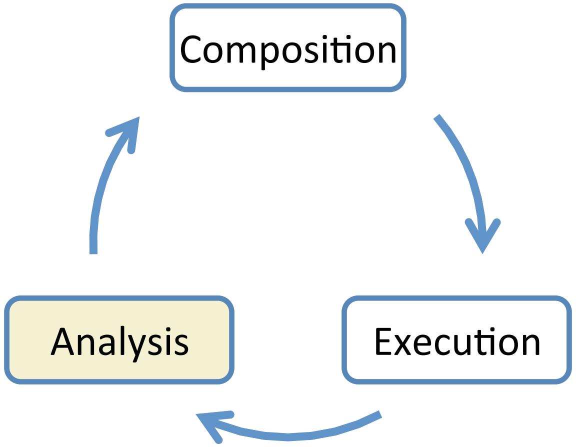
Version Model



out.png



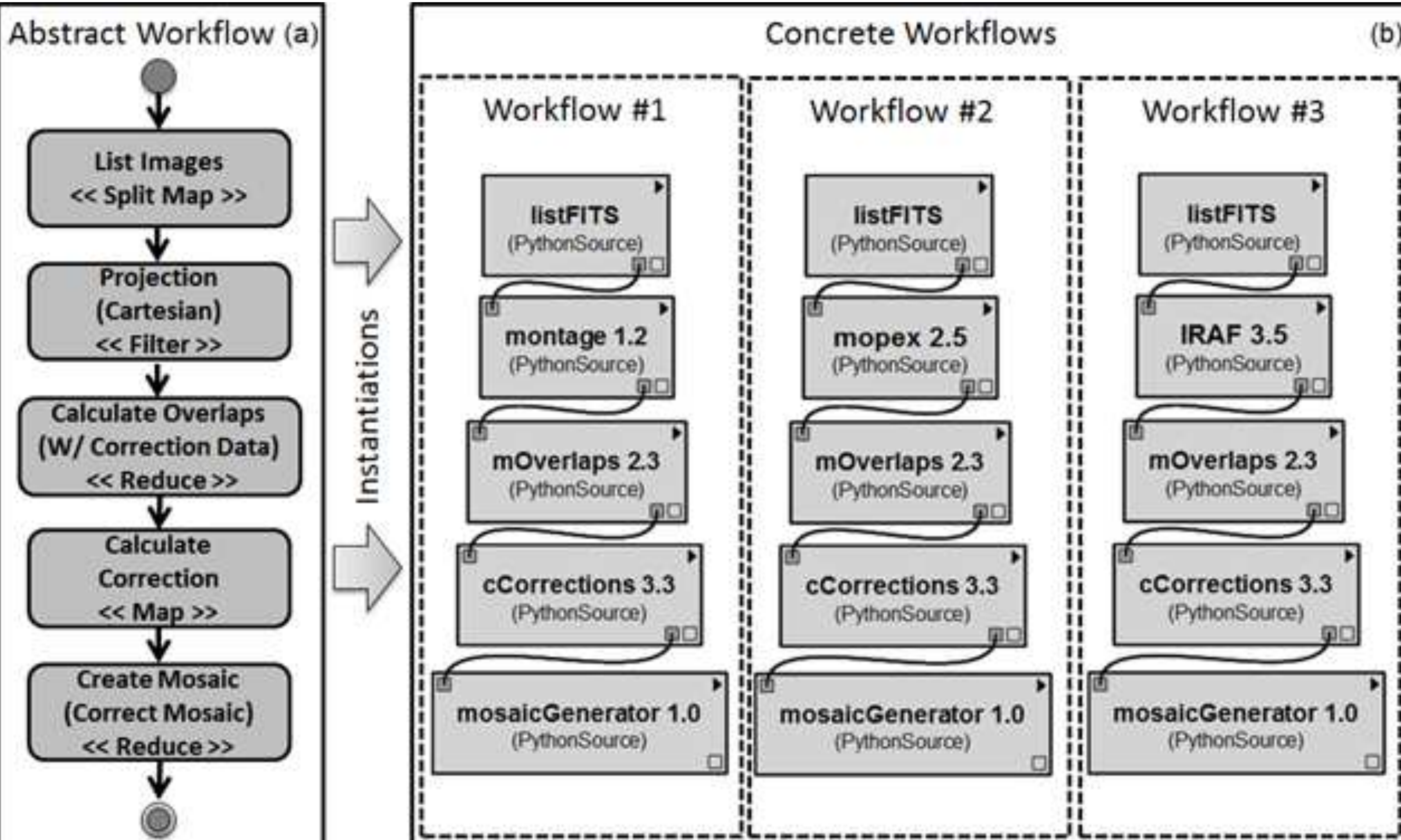
Conclusion:
“2012 was similar to 2013”



I don't think its enough to compare just these years. Could you add data from 2015?

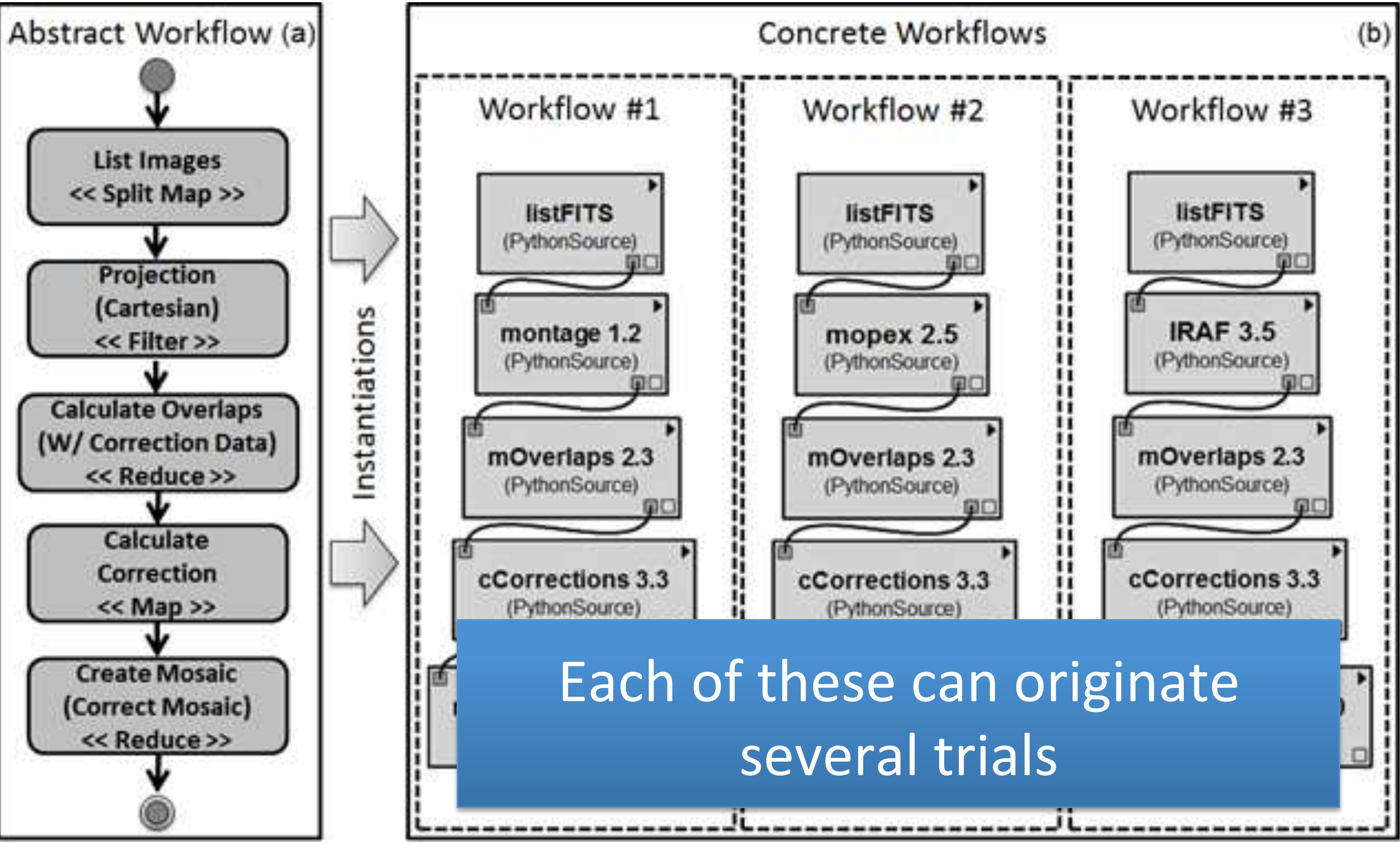
• • •

The same can be done for workflows



Source: MARINHO, A. Algebraic Experiment Line: an approach to represent scientific experiments based on workflows. PhD Thesis, UFRJ, 2015.

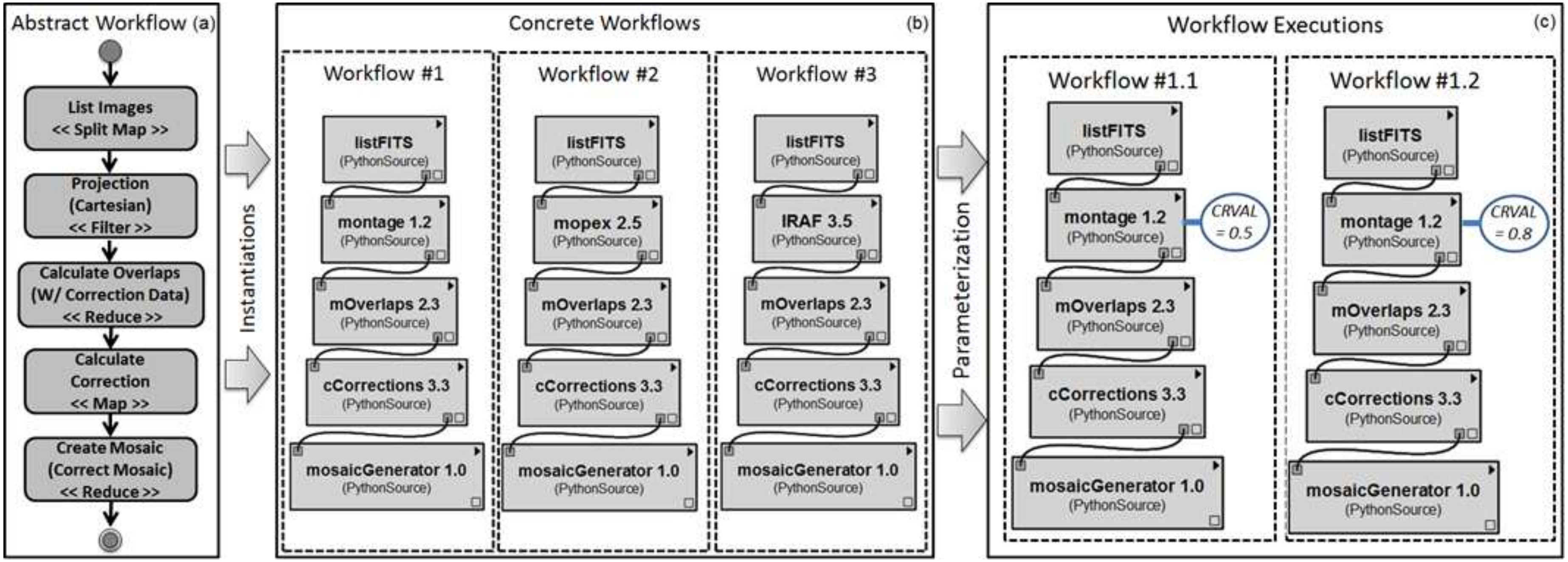
The same can be done for workflows



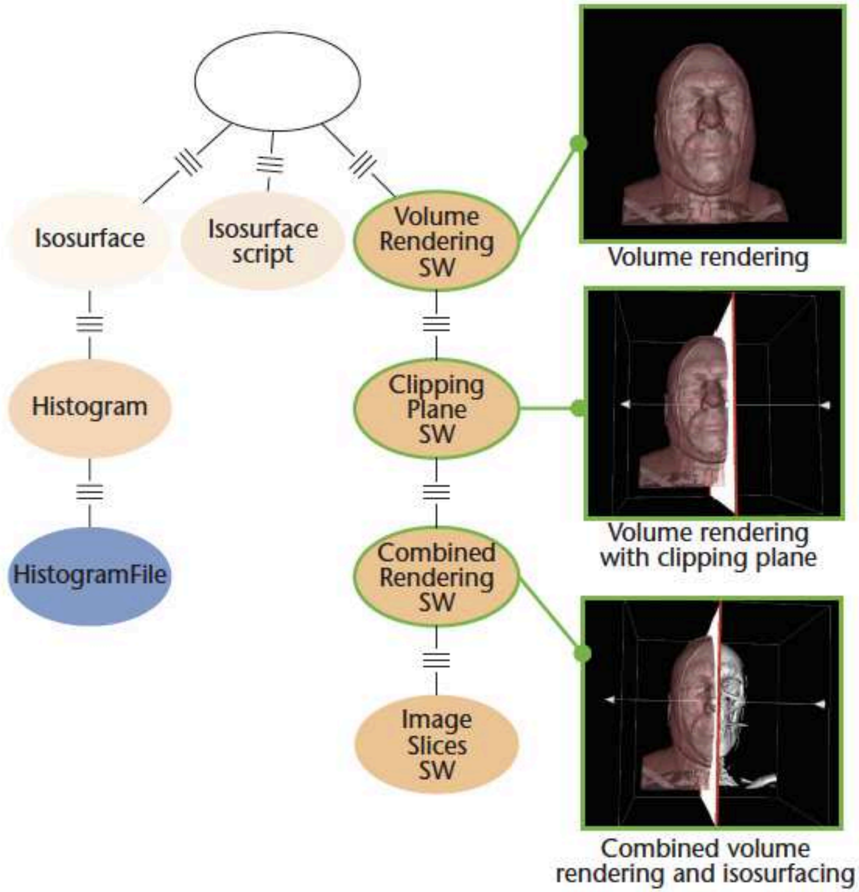
Each of these can originate several trials

Source: MARINHO, A. Algebraic Experiment Line: an approach to represent scientific experiments based on workflows. PhD Thesis, UFRJ, 2015.

Trials in Workflows



History Graph (VisTrails)



Source: Freire et al., 2008. Provenance for Computational Tasks: A Survey.

Several ways to go from abstract to concrete

- When using scripts, there are several ways to go from abstract to concrete workflows
  - Activities are implemented one after the other in the script (no functions)
  - Activities are mapped into functions (each activity becomes one or more function)

Black Box X White Box

- In Workflow systems, activities are black boxes
  - What goes in and out are known, but what happens inside is not known
- In scripts, activities can be black boxes or white boxes
  - An activity in a script can call an external program, and in this the activity is a black box
  - When the function is implemented in Python, it is a white box

Black Box X White Box

- Black boxes have implications in provenance analysis

```python
DRY_RUN = ...

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 = 2 ** 4000
final = process(n)
if DRY_RUN:
    final = 7
print(show(final))
```

Which values influence the result printed by this script? (variable **final**)

Source: Pimentel et al., 2016. Fine-grained Provenance Collection over Scripts Through Program Slicing

```python
DRY_RUN = ...

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 = 2 ** 4000
final = process(n)
if DRY_RUN:
    final = 7
print(show(final))
```

If **DRY-RUN** is 7, then **final** depends only on **DRY_RUN**

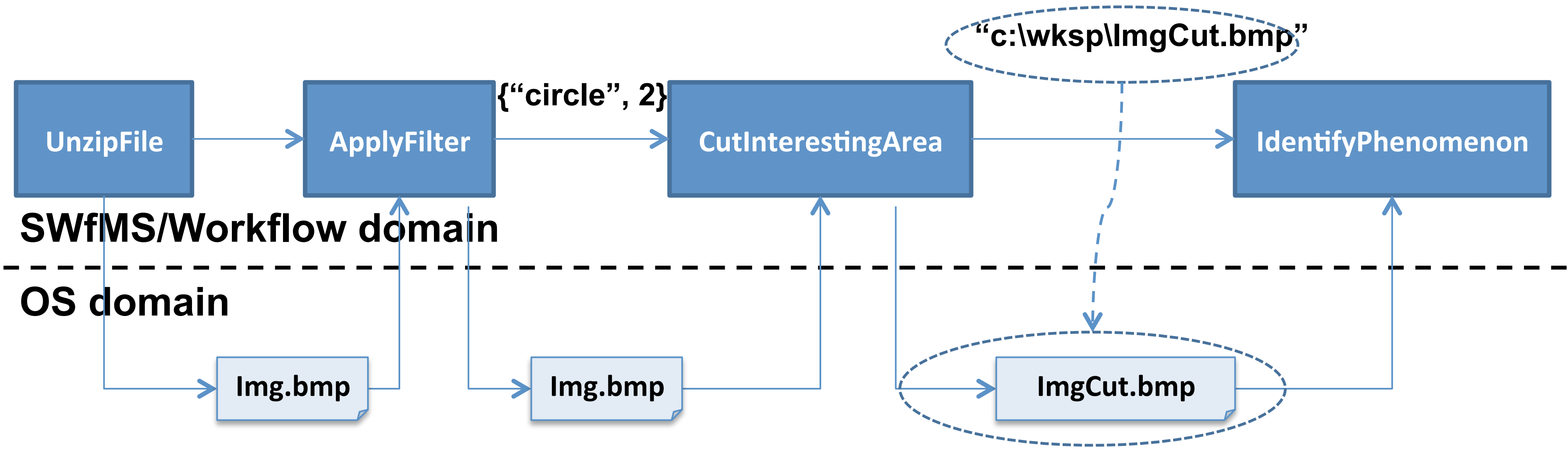
If not, then **final** also depends on **n**

Source: Pimentel et al., 2016. Fine-grained Provenance Collection over Scripts Through Program Slicing

Implications of Black Boxes

- If **process(number)** were a black box, anything could happen inside it
- It could, for example, read a file that could influence the value returned by the function, so dependencies would be missed
- This is a common case of **implicit provenance**, that is missed by several provenance capturing approaches

Implicit Provenance



Sources:
Neves et al., 2017. Managing Provenance of Implicit Data Flows in Scientific Experiments.
Marinho et al., 2011. Challenges in managing implicit and abstract provenance data: experiences with ProvManager.

Implicit Provenance

- OS-Based approaches are able to capture this kind of provenance
- Other approaches need special components to handle it (e.g. PROVMONITOR)

Overview of Existing Systems

- Workflow Management Systems
- Provenance Management Systems for Scritps

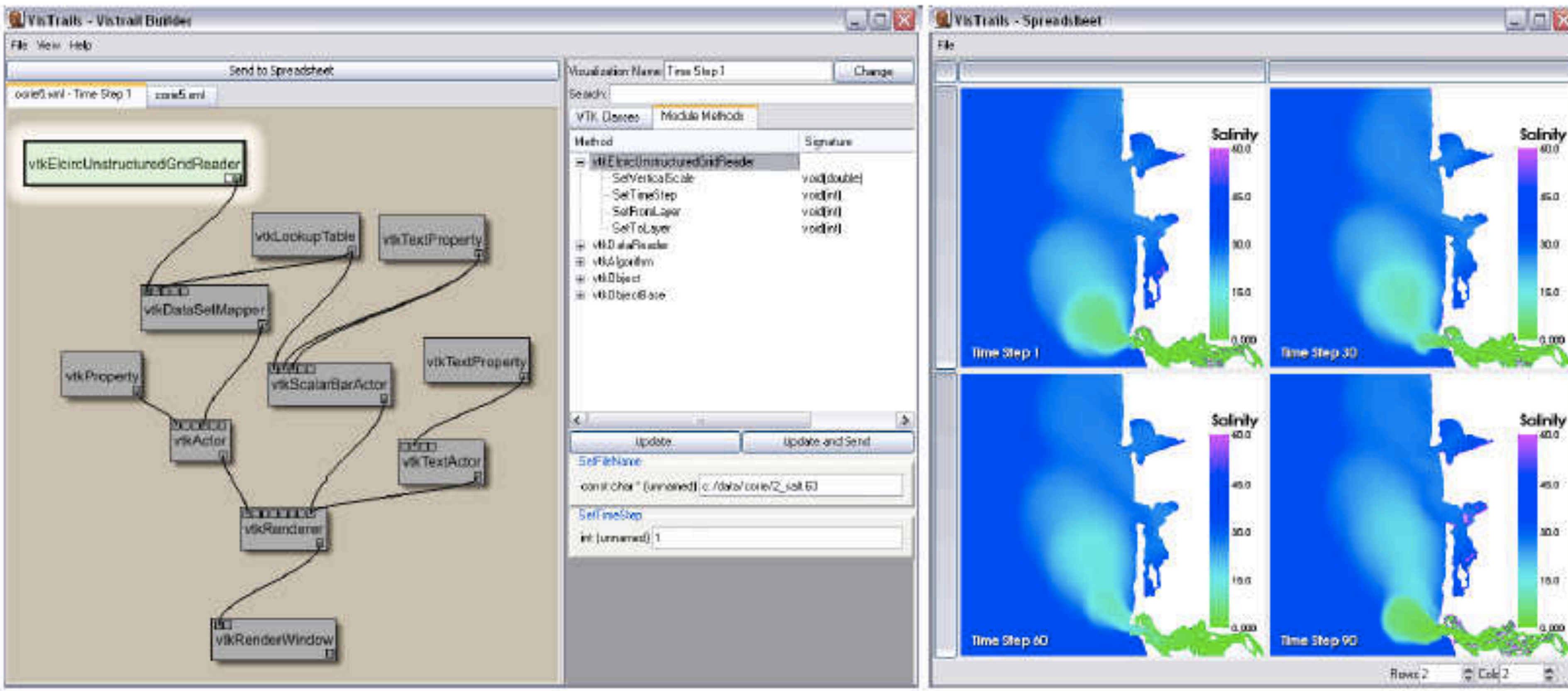
Workflow Management Systems

- VisTrails
- Taverna
- Kepler
- Swift
- SciCumulus
- Pegasus
- ...

VisTrails

- Visual drag and drop interface for workflow composition
- Captures history of changes in the workflow structure
- Allows comparing results side-by-side
- Focus on visualization

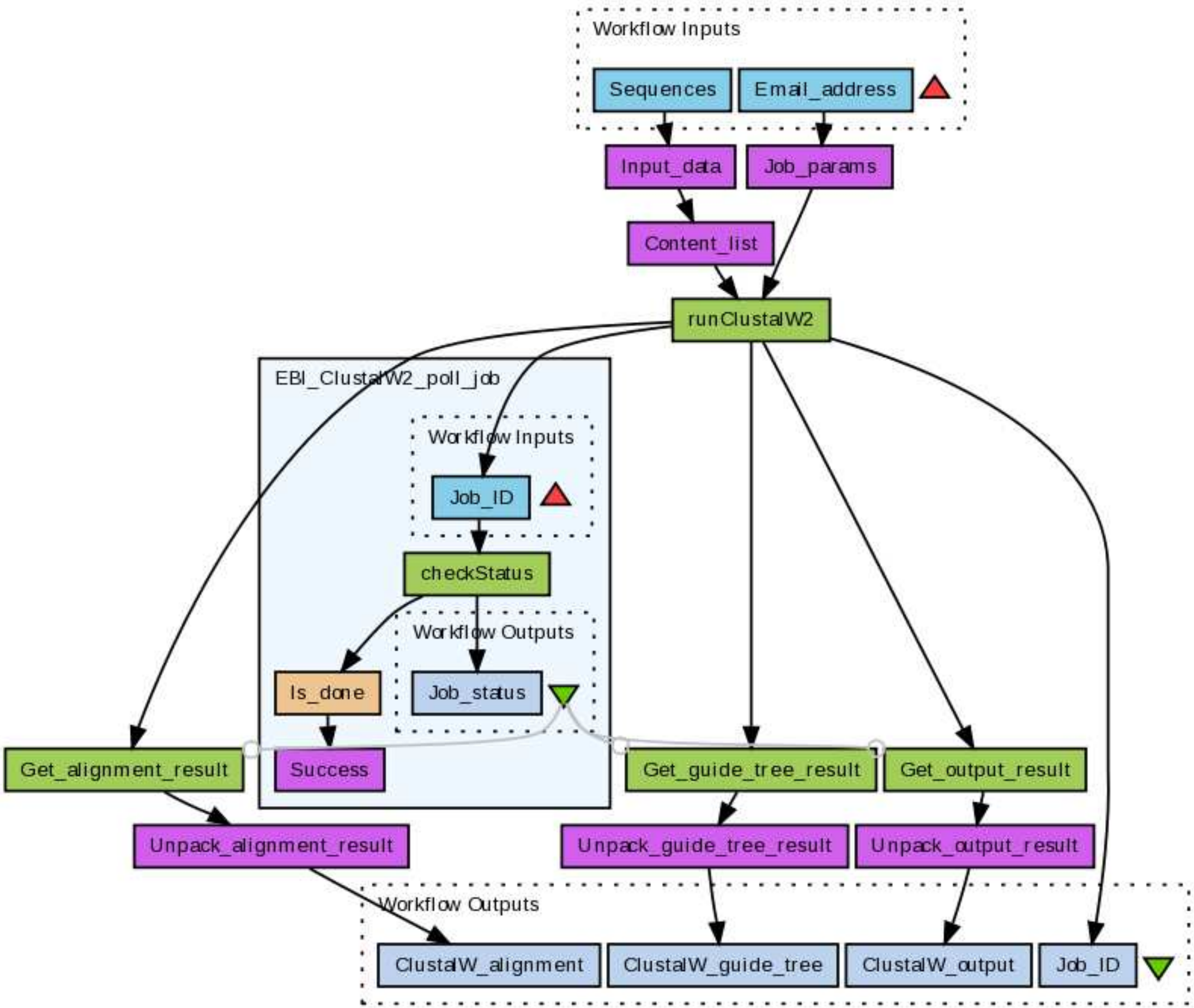
VisTrails


VisTrails - Vistrail Builder
File View Help
Send to Spreadsheet
Visualization Name: Time Step 1
Change
VTK Classes
Module Methods
Method
Signature
vtkElcircUnstructuredGridReader
vtkLookupTable
vtkTextProperty
vtkDataSetMapper
vtkProperty
vtkScalarBarActor
vtkTextProperty
vtkActor
vtkTextActor
vtkRenderer
vtkRenderWindow
Update
Update and Send
SetFileName
SetTimeStep
VisTrails - Spreadsheet
File
Salinity
Time Step 1
Time Step 30
Time Step 60
Time Step 90

Taverna

- Focus on Bioinformatics
- Several ready-to-use bioinformatics services
- Drag and Drop graphical interface for workflow composition

http://www.taverna.org.uk/

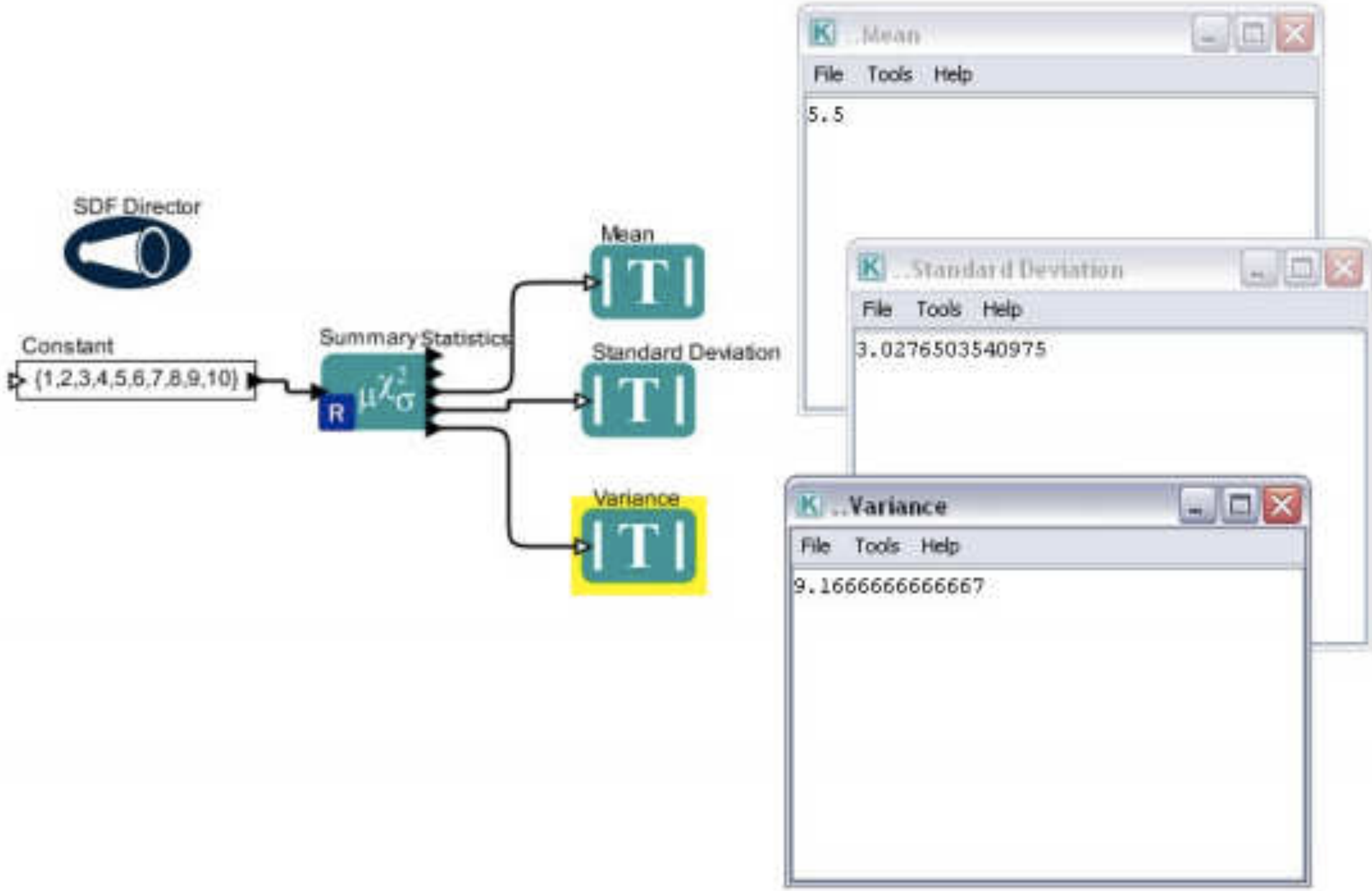


Workflow Inputs
Sequences
Email_address
Input_data
Job_params
Content_list
runClustalW2
EBI_ClustalW2_poll_job
Workflow Inputs
Job_ID
checkStatus
Workflow Outputs
Is_done
Job_status
Success
Get_alignment_result
Get_guide_tree_result
Get_output_result
Unpack_alignment_result
Unpack_guide_tree_result
Unpack_output_result
Workflow Outputs
ClustalW_alignment
ClustalW_guide_tree
ClustalW_output
Job_ID

Kepler

- Drag and Drop graphical interface for workflow composition
- Different actors that rules how the workflow executed – Kepler workflows are not DAG

https://kepler-project.org/


SDF Director
Constant
{1,2,3,4,5,6,7,8,9,10}
Summary Statistics
R
Mean
Standard Deviation
Variance
..Mean
File Tools Help
5.5
..Standard Deviation
File Tools Help
3.0276503540975
..Variance
File Tools Help
9.166666666667

Swift, SciCumulus and Pegasus

- Focus on High Performance
- Workflows are specified in XML (no graphical interface) in SciCumulus and Pegasus
- In Swift, workflows are specified as scripts in a specific language

http://swift-lang.org/main/index.php
https://scicumulusc2.wordpress.com/
https://pegasus.isi.edu/

Provenance Management Systems for Scripts

- noWorkflow
  - captures provenance for Python scripts
- RDataTracker
  - captures provenance for R scripts
- Sumatra
  - captures provenance for Python, R and MatLab scripts

Exercise

- Choose one of the systems presented in today's class and search the Web to find:
  - What is the format in which provenance is stored
  - If they export provenance in the PROV format

Provenance of these slides

- MARINHO, A. ; WERNER, C. M. L. ; MATTOSO, M. L. Q. ; BRAGANHOLO, V. ; MURTA, L. G. P. . Challenges in managing implicit and abstract provenance data: experiences with ProvManager. In: USENIX Workshop on the Theory and Practice of Provenance (TaPP), 2011, Heraklion, Creta, Grécia, p. 1-6.
- MATTOSO, M. L. Q. ; WERNER, C. M. L. ; TRAVASSOS, G. H. ; BRAGANHOLO, V. ; MURTA, L. G. P. ; OGASAWARA, E. ; OLIVEIRA, D. ; CRUZ, S. ; MARTINHO, W. . Towards Supporting the Life Cycle of Large Scale Scientific Experiments. International Journal of Business Process Integration and Management (Print), v. 5, p. 79-92, 2010.
- NEVES, V. C. ; OLIVEIRA, D. ; OCANA, K. A. ; BRAGANHOLO, V. ; MURTA, L. G. P. . Managing Provenance of Implicit Data Flows in Scientific Experiments. ACM Transactions on Internet Technology, 2017.
- PIMENTEL, J. F. N. ; FREIRE, J. ; BRAGANHOLO, V. ; MURTA, L. G. P. . Tracking and Analyzing the Evolution of Provenance from Scripts. In: International Provenance and Annotation Workshop (IPAW), 2016, Washington, D.C., v. 9672. p. 16-28.
- PIMENTEL, J. F. N. ; FREIRE, J. ; MURTA, L. G. P. ; BRAGANHOLO, V. . Fine-grained Provenance Collection over Scripts Through Program Slicing. In: International Provenance and Annotation Workshop (IPAW), 2016, Washington D.C., v. 9672. p. 199-203.