TaPP'13

ReproZip

Using Provenance to Support Computational Reproducibility

Fernando Chirigati
NYU-Poly

Dennis Shasha
NYU

Juliana Freire
NYU-Poly & NYU

Reproducibility

Good science requires reproducibility

> *"If I have seen further, it is by standing on the shoulders of giants."*
> *Isaac Newton*

Computational experiments require *reproducibility*

A program *P* running on computational environment *E* at time *T* is said to be *reproducible* if it yields the same answer on environment *E'* at time *T'* > *T*

Computational Reproducibility

Few computational experiments are reproducible

Why?

Author

How to encapsulate my experiment?
What should be included?
Too many dependencies…
Too many files to keep track…
Sigh.

We need **provenance**

- Description of the data
- Specification of the experiment
- Description of the environment

Computational Reproducibility

Manually tracking provenance is rarely feasible

Description of computational environment is *hard* to capture: It is *time consuming* and *error prone*

> *"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

The process should be *simple* and *automatic!*

What tools are available to support reproducible experiments?

State of the Art

Domain-specific tools [GenePattern, Madagascar, Sumatra,…]

- Do not capture provenance of experiments that straddle multiple tools

Scientific workflow systems [VisTrails, Kepler, Taverna, …]

- Fail to capture provenance of the computational environment
- Do not support portability
- Users must integrate the software they need into these systems
  - Time consuming, and scientists do not have time to spare…

Configuration management tools [Chef, Puppet, Fabric]

- Recipes to configure machines may interfere with the target computational environment

State of the Art

- Virtual machines
  - Portable across multiple operating systems
  - Snapshots are usually *very* large
  - Users must port the experiment to a virtual machine: Again, time consuming
- System-level provenance capture
  - [Burrito, ES3, PASS] Describe how data products were derived *in detail*, but do not create an executable description to attain reproducibility and portability
  - [CDE] supports reproducibility
    - Lower overhead than a virtual machine: copy just what you need
    - Hard to further *explore* the experiment
    - Adds run-time overhead when executing the packaged experiment

Our Approach: ReproZip

Automatically and systematically captures required **provenance** of *existing* experiments

Uses captured provenance to:

- Create self-contained *reproducible packages* for the experiment
  - Include all the binaries, data and dependencies
- Derive a *workflow specification* for the experiment

Readers/reviewers can then extract the packages and execute the workflow to *reproduce* and *explore* the experiment

How does it work?

Packing Experiments

Computational Environment ***E***

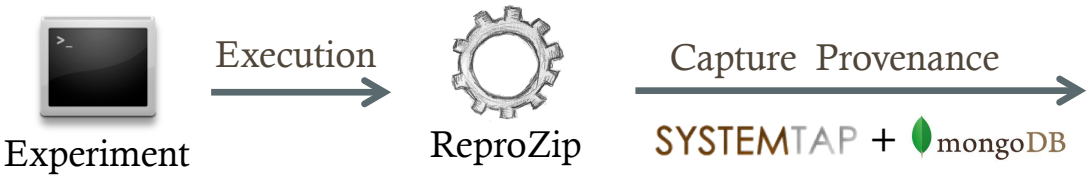
Experiment

Packing Experiments

Computational Environment ***E***

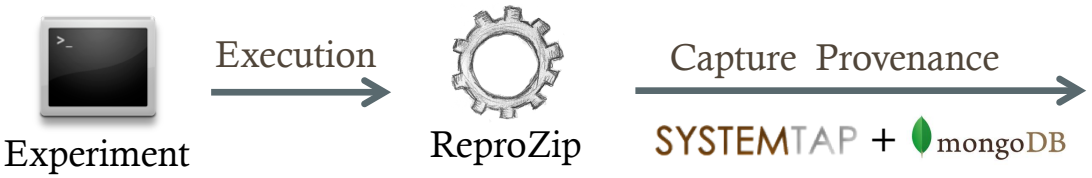
Packing Experiments

Computational Environment ***E***



Packing Experiments

Computational Environment ***E***



Packing Experiments

Computational Environment ***E***

Packing Experiments

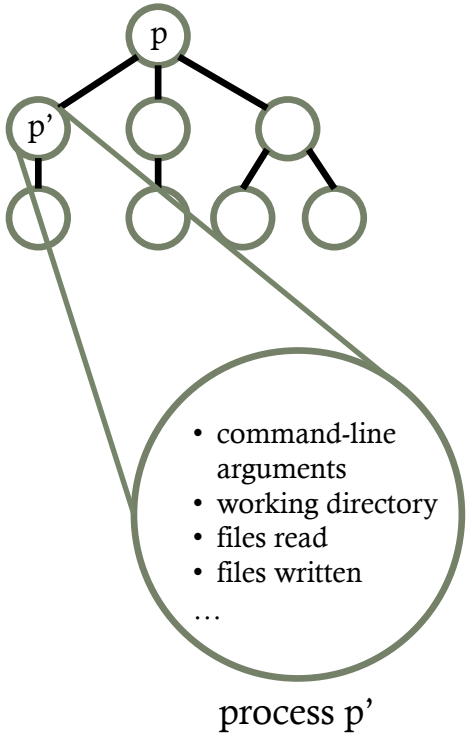
Computational Environment ***E***

Experiment

Execution

ReproZip

Capture of Provenance



Packing Experiments

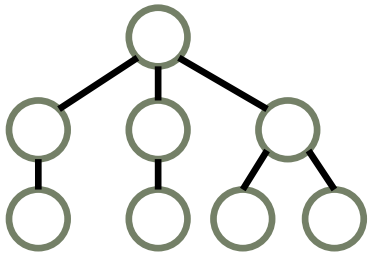
Computational Environment ***E***

Experiment

Execution

ReproZip

Capture of Provenance



Provenance Tree

Identification of Necessary Components

Description of data

Input and output files

Description of experiment

Executable programs and steps

Description of environment

Environment variables, dependencies, …

Packing Experiments

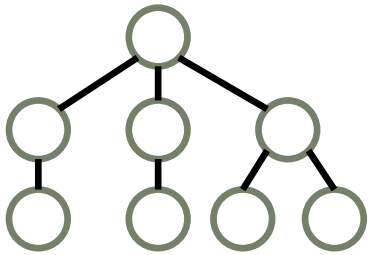
Computational Environment ***E***

Experiment

Execution

ReproZip

Capture of Provenance



Provenance Tree

Identification of Necessary Components

Description of data

Input and output files

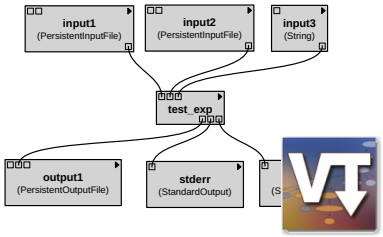
Description of experiment

Executable programs and steps

Description of environment

Environment variables, dependencies, …

Specification of Workflow



VisTrails Workflow

Packing Experiments

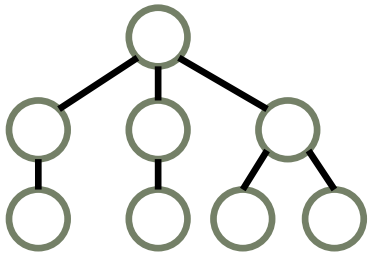
Computational Environment ***E***

Experiment

Execution

ReproZip

Capture of Provenance



Provenance Tree

Identification of Necessary Components

Description of data

Input and output files

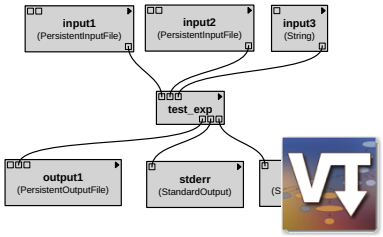
Description of experiment

Executable programs and steps

Description of environment

Environment variables, dependencies, …

Specification of Workflow



VisTrails Workflow

Reproducible Package

Verifying the Topological Correctness of Marching Cubes Algorithms

An example of making an experiment reproducible with ReproZip

Packing: Example

./mc33verification input/3741-scalar_field.iso output/output.txt

Original Command Line

Packing: Example

./mc33verification input/3741-scalar_field.iso output/output.txt

Original Command Line

python ~/reprozip/pack.py -e -c "./mc33verification input/3741-scalar_field.iso output/output.txt"

Packing with ReproZip

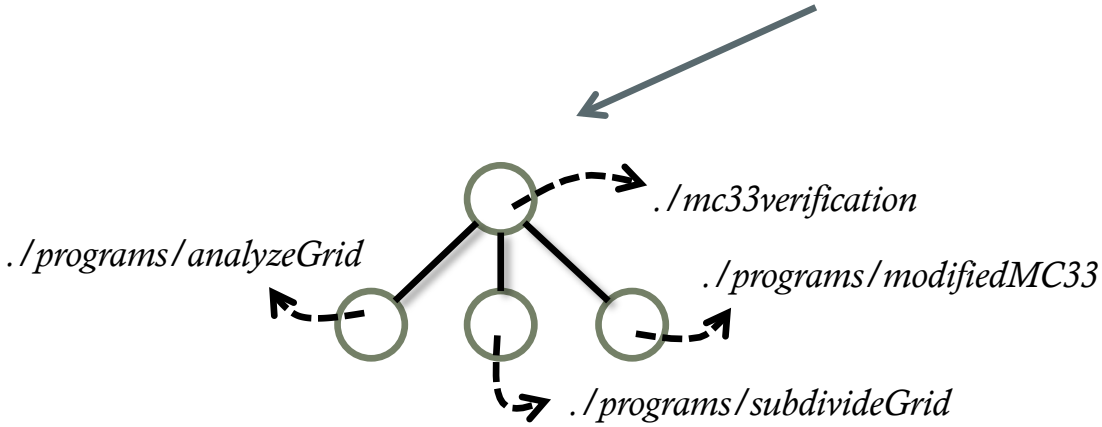
Packing: Capture Provenance

./mc33verification input/3741-scalar_field.iso output/output.txt

Original Command Line

python ~/reprozip/pack.py -e -c "./mc33verification input/3741-scalar_field.iso output/output.txt"

Packing with ReproZip



Provenance Tree

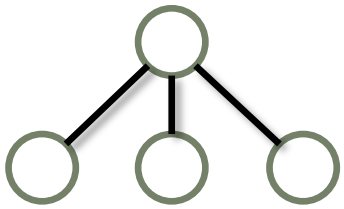
Packing: Configure Package

./mc33verification input/3741-scalar_field.iso output/output.txt

Original Command Line

python ~/reprozip/pack.py -e -c "./mc33verification input/3741-scalar_field.iso output/output.txt"

Packing with ReproZip



Provenance Tree

Configuration File

```
# Platform: Linux-3.2.0-32-generic-i686-with-Ubuntu-12.04-precise
# Processor: i686
# Number of CPUs: 1


* Original File *                                          * Size (KB) *   * Include? *

[main program]
/home/fernando/mc33/build/component_analysis                    7.55           Y

[other programs]
/home/fernando/mc33/build/others/vtkMarchingCubes              13.01           Y
/home/fernando/mc33/build/others/ModifiedMC33                 311.59           Y
/bin/sh                                                        97.93           Y
/bin/dash                                                      97.93           Y

[other input files]
/home/fernando/mc33/input/aneurism.iso                      65536.04           Y
/home/fernando/mc33/input/aneurism.vtk                      16384.19           Y

[dependencies]
/usr/lib/i386-linux-gnu/i686/cmov/libavutil.so.51.22.1        118.49           Y
/usr/lib/i386-linux-gnu/libkrb5support.so.0                    29.61           Y
/usr/lib/i386-linux-gnu/libkrb5support.so.0.1                  29.61           Y
/usr/lib/i386-linux-gnu/libroken.so.18.1.0                     82.46           Y
/usr/lib/i386-linux-gnu/libogg.so.0                            25.39           Y
```

Packing: Deriving Workflow Specification

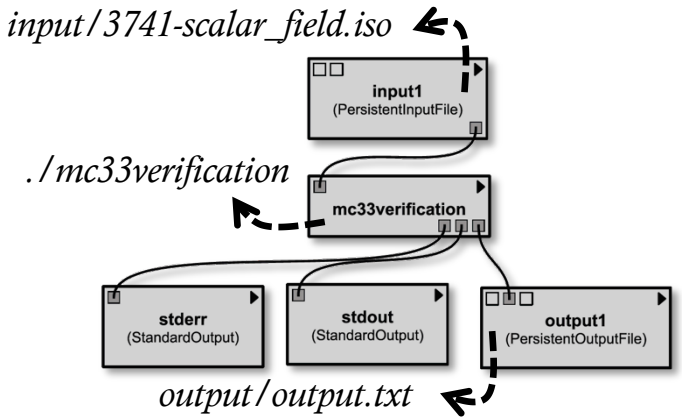
python ~/reprozip/pack.py -g --name=mc33verification

Creating the reproducible package and the workflow

Packing: Deriving Workflow Specification

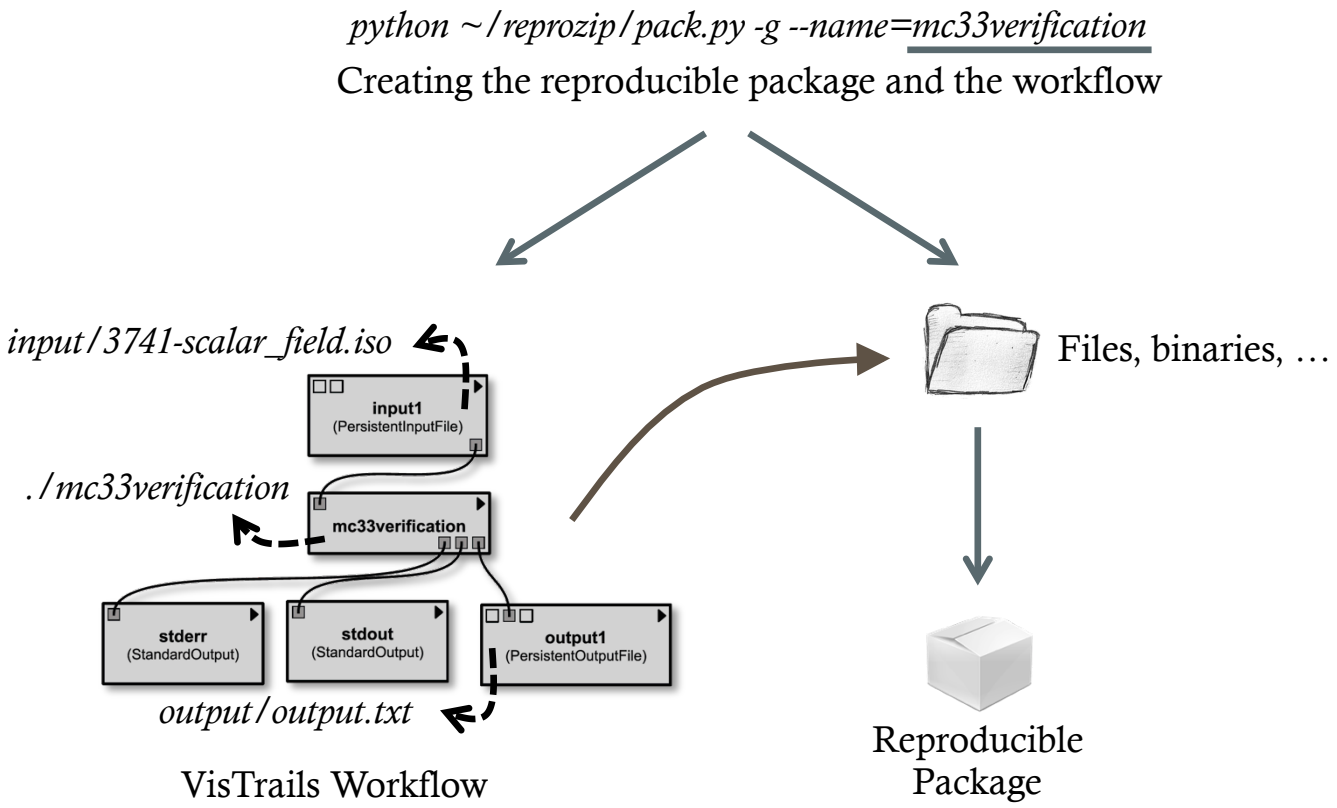
python ~/reprozip/pack.py -g --name=mc33verification

Creating the reproducible package and the workflow



VisTrails Workflow

Packing: Deriving Package



Unpacking Experiments

Computational Environment ***E'***

E' compatible with *E*

Reproducible Package

Unpacking Experiments

Computational Environment ***E'***

E' compatible with *E*

Reproducible Package

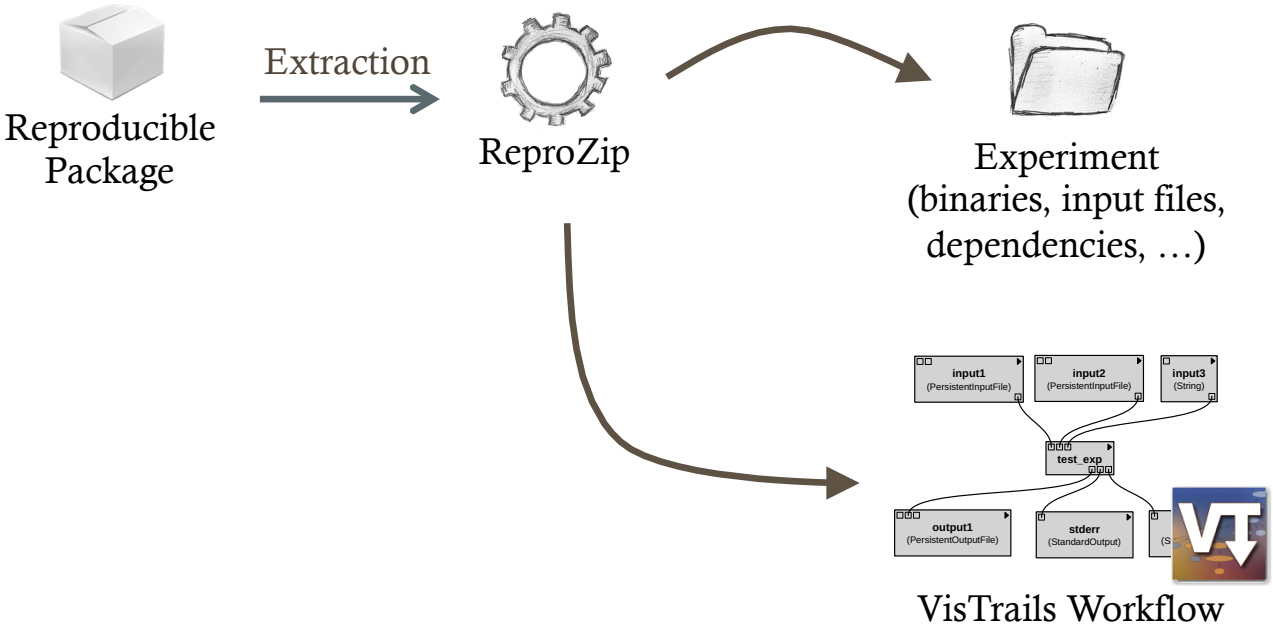
Extraction

ReproZip

Unpacking Experiments

Computational Environment ***E'***

E' compatible with *E*



Unpacking: Example

python ~/reprozip/unpack.py mc33verification

Unpacking experiment

Unpacking: Example

python ~/reprozip/unpack.py mc33verification

Unpacking experiment

./mc33verification

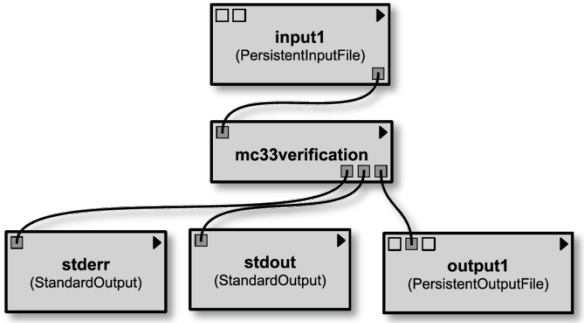
Verification and Exploration

Reproducibility of *deterministic* process

Two ways to reproducing the results:

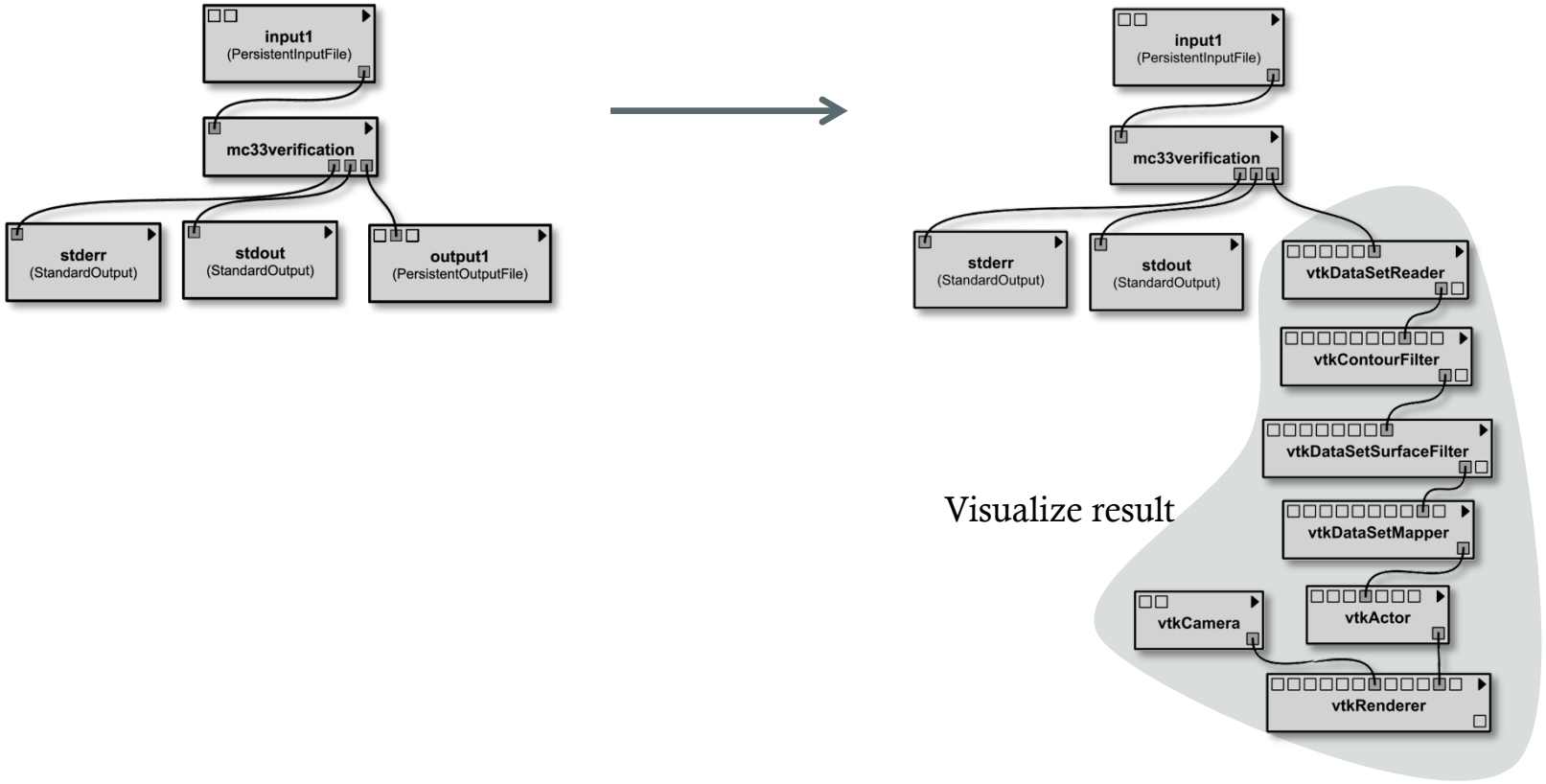
./mc33experiment/rep.exec

Command-line execution

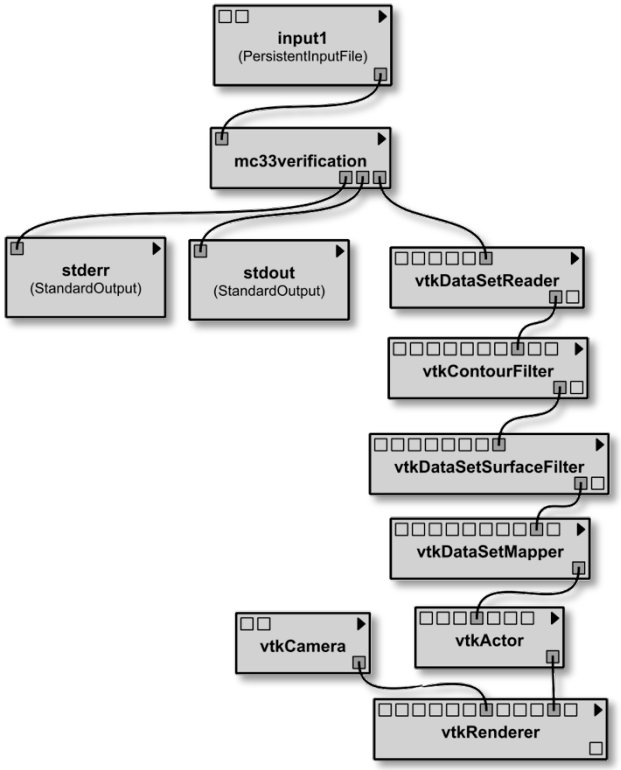


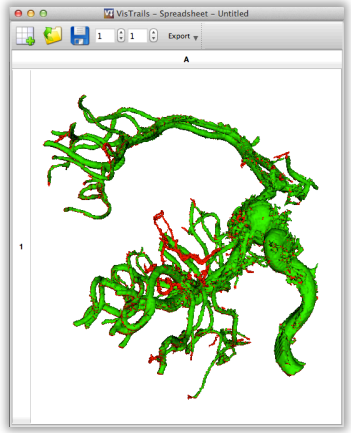
VisTrails Workflow

Verification and Exploration



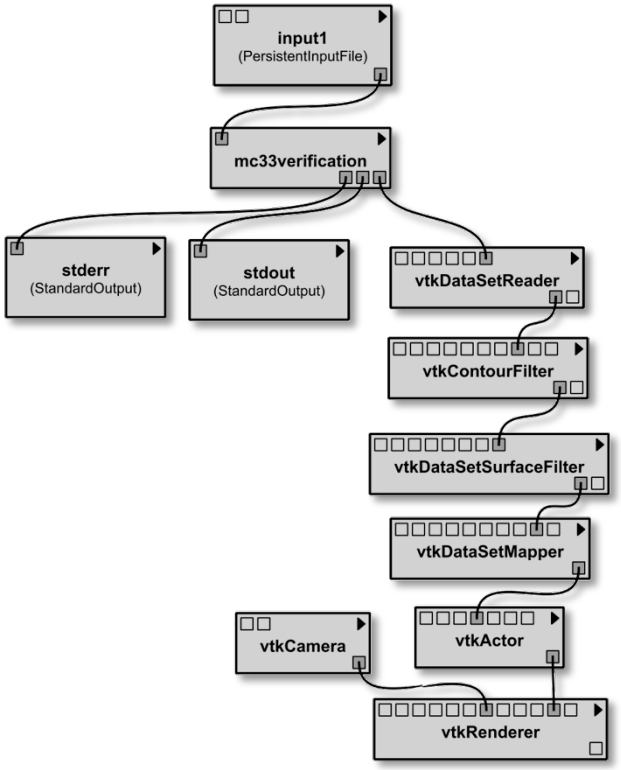
Verification and Exploration

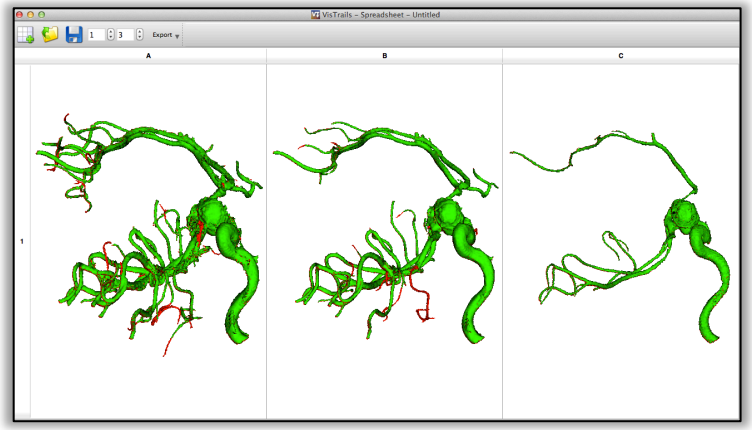




Visualization

Verification and Exploration





Parameter Exploration

Conclusion

ReproZip aims to simplify the creation of reproducible experiments

- It captures provenance and identifies the components needed to reproduce results
- Users can customize the package

Integrated with scientific workflows

- Scientists can reap the benefits without the cost
- Further explore the results and get review provenance for free

Limitations

- Works only on Linux
- Package may not run
  - If underlying software is incompatible with target environment---for this situation, we suggest the use of a VM
  - Executables that use hard-coded paths

Acknowledgments

- Claudio Silva and the VisTrails team
- This work is partially supported by the National Science Foundation awards CNS-1229185, IIS-1139832, IIS-1142013, CNS-1153503, IIS-0905385.

Thank you!