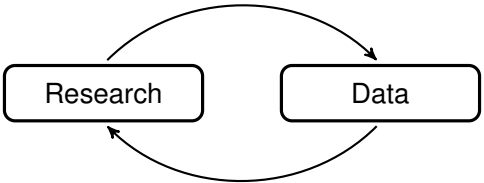
Reproducibility in computational mathematics
A hands-on session

Tobias Boege

Research Data in Discrete Mathematics,
MPI-MiS Leipzig,
14 March 2023

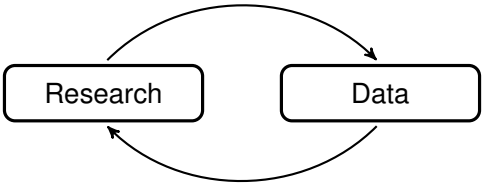
Reproducibility



In this talk:

- **Reproducibility**: how to ensure that *independent* repetitions of an experiment yield consistent results. (Ideally without contacting the original authors.)

Reproducibility



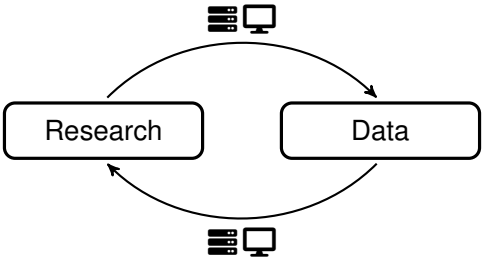
In this talk:

- **Reproducibility**: how to ensure that *independent* repetitions of an experiment yield consistent results. (Ideally without contacting the original authors.)

Not in this talk:

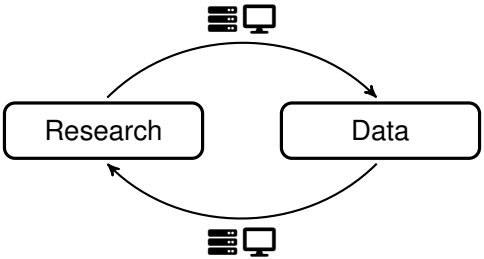
- **Correctness**: how to produce correct results.
- **Certification**: how to allow verification of claims without reproducing data.

The reproducibility dilemma



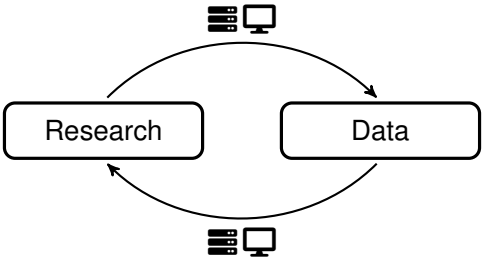
- Computers help us generate and use large amounts of data.
- Reproducibility suffers from rapidly changing computing environments.

The reproducibility dilemma



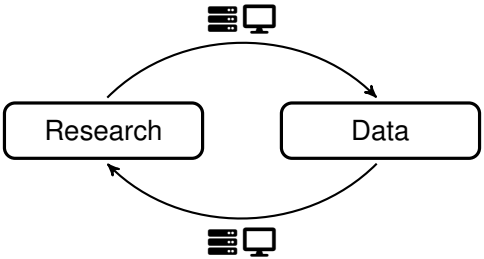
- Computers help us generate and use large amounts of data.
- Reproducibility suffers from rapidly changing computing environments.
  - Software does not compile anymore or changes behavior.

The reproducibility dilemma



- Computers help us generate and use large amounts of data.
- Reproducibility suffers from rapidly changing computing environments.
  - Software does not compile anymore or changes behavior.
  - Dependencies or special setup procedures are not documented.

The reproducibility dilemma



- Computers help us generate and use large amounts of data.
- Reproducibility suffers from rapidly changing computing environments.
  - Software does not compile anymore or changes behavior.
  - Dependencies or special setup procedures are not documented.
  - Proprietary or closed-source software cannot be obtained or repaired.

Story 1: Getting the job done

```
# hydra: /opt/local/scip/8.0.3/bin/scip
# my laptop: /Users/me/Documents/scipoptsuite-8.0.2/scip/bin/scip
const scipexe = "/Users/me/Documents/scipoptsuite-8.0.2/scip/bin/scip"
function scip(filename, outputname)
    run(`$scipexe -f $filename -l $outputname -q`);
end
```

Story 1: Getting the job done

```
# hydra: /opt/local/scip/8.0.3/bin/scip
# my laptop: /Users/me/Documents/scipoptsuite-8.0.2/scip/bin/scip
const scipexe = "/Users/me/Documents/scipoptsuite-8.0.2/scip/bin/scip"
function scip(filename, outputname)
    run(`$scipexe -f $filename -l $outputname -q`);
end
```

- Ad-hoc installation of the currently latest version of SCIP.
- Hard-coded path for two computing environments with different setup.

Story 1: Getting the job done

```
# hydra: /opt/local/scip/8.0.3/bin/scip
# my laptop: /Users/me/Documents/scipoptsuite-8.0.2/scip/bin/scip
const scipexe = "/Users/me/Documents/scipoptsuite-8.0.2/scip/bin/scip"
function scip(filename, outputname)
    run(`$scipexe -f $filename -l $outputname -q`);
end
```

- Ad-hoc installation of the currently latest version of SCIP.
- Hard-coded path for two computing environments with different setup.
- Solution: Make sure `scip` is always in `PATH`.

Story 1: Getting the job done

```
# hydra: /opt/local/scip/8.0.3/bin/scip
# my laptop: /Users/me/Documents/scipoptsuite-8.0.2/scip/bin/scip
const scipexe = "/Users/me/Documents/scipoptsuite-8.0.2/scip/bin/scip"
function scip(filename, outputname)
    run(`$scipexe -f $filename -l $outputname -q`);
end
```

- Ad-hoc installation of the currently latest version of SCIP.
- Hard-coded path for two computing environments with different setup.
- Solution: Make sure `scip` is always in `PATH`.
- Bonus question: Is SCIP output reproducible?

Story 2: Lack of interfaces

https://github.com/zvihr/algebraic-matroids

1. Generate a numerical Jacobian matrix, in one of the following ways:
   i. Compute a Jacobian symbolically, and then specialize to a generic point of the variety. A generic point can be obtained for a parametrization by choosing random values for the parameters, and for a variety defined by its ideal using Bertini Tracktype:1.
   ii. Compute the Jacobian numerically using Bertini: Run Tracktype:1, save a witness point from main_data in start, and run Tracktype:-3.
2. Use numerical linear algebra, in Sage, Maple, or MATLAB, to compute a list of bases and circuits from the Jacobian matrix.

Story 2: Lack of interfaces

https://github.com/zvihr/algebraic-matroids

1. Generate a numerical Jacobian matrix, in one of the following ways:
   i. Compute a Jacobian symbolically, and then specialize to a generic point of the variety. A generic point can be obtained for a parametrization by choosing random values for the parameters, and for a variety defined by its ideal using Bertini Tracktype:1.
   ii. Compute the Jacobian numerically using Bertini: Run Tracktype:1, save a witness point from main_data in start, and run Tracktype:-3.
2. Use numerical linear algebra, in Sage, Maple, or MATLAB, to compute a list of bases and circuits from the Jacobian matrix.

- Switching between software requires manual conversion of data.
- Not really software — more a description of software we wish existed.

Story 2: Lack of interfaces

https://github.com/zvihr/algebraic-matroids

1. Generate a numerical Jacobian matrix, in one of the following ways:
   i. Compute a Jacobian symbolically, and then specialize to a generic point of the variety. A generic point can be obtained for a parametrization by choosing random values for the parameters, and for a variety defined by its ideal using Bertini Tracktype:1.
   ii. Compute the Jacobian numerically using Bertini: Run Tracktype:1, save a witness point from main_data in start, and run Tracktype:-3.
2. Use numerical linear algebra, in Sage, Maple, or MATLAB, to compute a list of bases and circuits from the Jacobian matrix.

- Switching between software requires manual conversion of data.
- Not really software — more a description of software we wish existed.
- Solution (nowadays): Implement it in OSCAR.

Story 3: Dying languages

`http://web.archive.org/web/20070720064410/http://atrey.karlin.mff.cuni.cz/~simecek/skola/models/`

Utilities for manipulation of data files

All sources contain no comments and they are quite messy (sorry). Originally, they were written only for my own purposes. The first 3 source codes have been written for GNU Pascal, the last one should be compiled in Borland Pascal.

Story 3: Dying languages

http://web.archive.org/web/20070720064410/http://atrey.karlin.mff.cuni.cz/~simecek/skola/models/

Utilities for manipulation of data files

All sources contain no comments and they are quite messy (sorry). Originally, they were written only for my own purposes. The first 3 source codes have been written for GNU Pascal, the last one should be compiled in Borland Pascal.

- Data meanwhile deleted from institute website.

Story 3: Dying languages

http://web.archive.org/web/20070720064410/http://atrey.karlin.mff.cuni.cz/~simecek/skola/models/

Utilities for manipulation of data files

All sources contain no comments and they are quite messy (sorry). Originally, they were written only for my own purposes. The first 3 source codes have been written for GNU Pascal, the last one should be compiled in Borland Pascal.

- Data meanwhile deleted from institute website.
- GNU Pascal compiler for programs hard to obtain.
  - Even active programming language communities only care about *reported* problems which surface when code is *actively* used.

Story 3: Dying languages

http://web.archive.org/web/20070720064410/http://atrey.karlin.mff.cuni.cz/~simecek/skola/models/

Utilities for manipulation of data files

All sources contain no comments and they are quite messy (sorry). Originally, they were written only for my own purposes. The first 3 source codes have been written for GNU Pascal, the last one should be compiled in Borland Pascal.

- Data meanwhile deleted from institute website.
- GNU Pascal compiler for programs hard to obtain.
  - Even active programming language communities only care about *reported* problems which surface when code is *actively* used.
- Hardly documented compiler-specific, binary floating point data format.

Story 3: Dying languages

http://web.archive.org/web/20070720064410/http://atrey.karlin.mff.cuni.cz/~simecek/skola/models/

Utilities for manipulation of data files

All sources contain no comments and they are quite messy (sorry). Originally, they were written only for my own purposes. The first 3 source codes have been written for GNU Pascal, the last one should be compiled in Borland Pascal.

- Data meanwhile deleted from institute website.
- GNU Pascal compiler for programs hard to obtain.
  - Even active programming language communities only care about *reported* problems which surface when code is *actively* used.
- Hardly documented compiler-specific, binary floating point data format.
- Reimplementation: `https://github.com/taboege/simecek-tools`.

Story 4: Platform differences

https://github.com/CInet/CInet-Alien-MiniSAT-All

- `src/component_types/base_packed_component.h:215:29: warning: non-static data member initializers only available with -std=c++11 or -std=gnu++11`

Story 4: Platform differences

https://github.com/CInet/CInet-Alien-MiniSAT-All

- `src/component_types/base_packed_component.h:215:29: warning: non-static data member initializers only available with -std=c++11 or -std=gnu++11`
- `Linking nbc_minisat_all_static`
  `ld: unknown option: --static`

Story 4: Platform differences

https://github.com/CInet/CInet-Alien-MiniSAT-All

- `src/component_types/base_packed_component.h:215:29: warning: non-static data member initializers only available with -std=c++11 or -std=gnu++11`
- `Linking nbc_minisat_all_static`
  `ld: unknown option: --static`

- Convoluted hierarchy of build systems.
- Static linking is different between GNU's and MacOS's `ld`.

Story 4: Platform differences

https://github.com/CInet/CInet-Alien-MiniSAT-All

- `src/component_types/base_packed_component.h:215:29: warning: non-static data member initializers only available with -std=c++11 or -std=gnu++11`
- `Linking nbc_minisat_all_static`
  `ld: unknown option: --static`

- Convoluted hierarchy of build systems.
- Static linking is different between GNU's and MacOS's `ld`.
- Solution: Require `gcc` . . . ?

What can we do?

- Ship static binaries of external dependencies using reproducible builds:

  https://reproducible-builds.org

What can we do?

- Ship static binaries of external dependencies using reproducible builds:

  https://reproducible-builds.org

- When preparing data for publication, re-run all computations in a clean (Docker) container and document or even automate setup steps.

What can we do?

- Ship static binaries of external dependencies using reproducible builds:

  https://reproducible-builds.org

- When preparing data for publication, re-run all computations in a clean (Docker) container and document or even automate setup steps.
- Automate the entire computation and arrange for regular test runs, a sort of “MathRepo health check”. (Keyword: continuous integration.)

What can we do?

- Ship static binaries of external dependencies using reproducible builds:

  https://reproducible-builds.org

- When preparing data for publication, re-run all computations in a clean (Docker) container and document or even automate setup steps.

- Automate the entire computation and arrange for regular test runs, a sort of “MathRepo health check”. (Keyword: continuous integration.)

- Use e.g. Julia’s `Manifest.toml` to track exact dependency versions.

  https://pkgdocs.julialang.org/v1/creating-packages/

```
[[deps.AlgebraicSolving]]
deps = ["LinearAlgebra", "Markdown", "Nemo", "Test", "msolve_jll"]
git-tree-sha1 = "0697dc8e50db21519459c729b2b38b99a65cee12"
uuid = "66b61cbe-0446-4d5d-9090-1ff510639f9d"
version = "0.2.2"
```

Hands on!

Online worksheet: `https://github.com/taboege/rddm23`

- Likelihood degenerations (Example 4.1):
  What is the ML degree of the Pappus matroid?
- No eleventh conditional Ingleton inequality (Section 3.1):
  Find the 6 814 shortest masks of the Ingleton expression.
- Package the SAT solver `Kissat` (with proof support) for Julia:
  see Creating Packages and Artifacts.
- Reproduce the research data in `https://github.com/taboege/rddm23` on a freshly installed OS (container) of your choice. Automate this procedure.
- Count loopless matroids using the axiomatization in `arXiv:2303.06668`.