

Journée Reproductibilité en Sciences

Eric Boix 4 Avril 2024



Conference: reproducibility guidelines

Website: <https://osf.io/phmce/>
Version: December 2020
DOI: 10.17605/OSF.IO/CB7Z8



REPRODUCIBLE PAPER GUIDELINES

Full and short papers submitted to the AGILE conference **have** to include a **Data and Software Availability** section which documents data, software, and computational infrastructure to support reproduction, or mentions reasons for not publishing them.

The above requirement is the only one to comply with the AGILE Reproducible Paper Guidelines. The remainder of the document provides concrete recommendations for all involved stakeholders to increase transparency, reproducibility, and openness of computational GIScience research. The following table of contents shows the recommended parts for different readers. Familiarity with all sections is, of course, beneficial.

Author
Reproducibility Reviewer
Scientific Reviewer



Reproducibility Checklist

Helps to ensure authors and reviewers do not miss anything important.

2



Author Guidelines

Show how to write the Data and Software Availability Section and give practical recommendations to make data and computational workflows reproducible.

4

Writing the Data and Software Availability Section
Including Data in Research Papers
Including Computational Workflows in Research Papers

Conference: reproducibility guidelines

INCLUDING COMPUTATIONAL WORKFLOWS IN RESEARCH PAPERS

	Minimum requirements	Recommended practices
What? Computational environment	<ul style="list-style-type: none"> Describe the used environment and computational infrastructure, e.g., hardware specs, operating system List software versions Cite used software¹⁴ 	<ul style="list-style-type: none"> Provide the actual environment, e.g., a Dockerfile + container¹⁵ or a Virtual Machine (e.g., using OSGeo-Live) Provide a pinned freeze of your dependencies (structured configuration files with dependency information) Add a colophon or “reproducibility receipt”¹⁶ to your notebooks Installation and execution instructions for different operating systems
Computation steps	<ul style="list-style-type: none"> Document the detailed steps in a text file and/or flowchart (every action/click) Document expected execution times given computing power unless negligible Ask a colleague to try out the instructions 	<ul style="list-style-type: none"> Scripts/models and a README file that explains their use All figures are fully scripted and a peer has read your README’s instructions (incl. interactive visualisations and interactive adjustments) Multi-panel plots are composited with scripts¹⁷ Software package with structured metadata¹⁸, tests/CI¹⁹, and a pipeline framework²⁰ or workflow language²¹ Live documents for analyses, e.g., Binder²² Live demo of APIs/online applications (e.g., anonymous cloud resources, such as Google Cloud Run or AWS) Subset or a synthetic dataset for quick evaluation

Conference: reproducibility guidelines

Versioned code repository, such as GitHub or GitLab [...]

Computational environment

- Provide the actual environment, e.g., a Dockerfile
- Provide a pinned freeze of your dependencies
- Installation and execution instructions for different operating systems

Computational steps

- Document the detailed steps in a text file and/or flowchart (every action/click)
- Scripts/models and a README file that explains their use

Continuous Integration tools should nail it

VCityTeam / UD-Viz

Code Issues 40 Pull requests 2 Discussions Actions Projects 2 Wiki Security 4

UD-Viz Public Edit Pins Watch 9

master 8 Branches 63 Tags Go to file Add file Code

Mathieu Livebardon and mathieuLivebardon fix(showroom):css about an... e076c2c · 2 months ago 3,956 Commits

.github/workflows	Created Documentation Github Action	3 months ago
bin	Release 4.1.0 - Legonizer	2 months ago
docs	Release 4.1.0 - Legonizer	2 months ago
examples	fix(showroom):css about and help	2 months ago
img	update(readme):Mosaic online demos ; fix links ; add Gier...	2 months ago
packages	fix(showroom):css about and help	2 months ago
test	reviewing changes	5 months ago
.eslintrc.js	fix CI	5 months ago
.gitignore	refact(ud-viz) split into multiples packages	7 months ago
.prettierrc	update(prettierrc):add trailing comma rules	7 months ago
.travis.yml	update dov	7 months ago

CI worker logs



Search all repositories

VCityTeam / UD-Viz build passing log scan failing

Current Branches Build History Pull Requests Log Scans > Build #1319 Job #1319.3

More options

VCityTeam/VCity # 925

Duration: 2 min 50 sec
Finished: 2 days ago

VCityTeam/UD-Viz # 1338

Duration: 12 min 6 sec
Finished: 6 days ago

VCityTeam/UD-Demo-vcity-py3 # 39

Duration: 2 min 14 sec
Finished: 16 days ago

VCityTeam/ExpeData-Workflows # 123

Duration: 44 sec
Finished: about a month ago

VCityTeam/UD-Viz-Template # 196

Duration: 2 min 26 sec
Finished: 2 months ago

VCityTeam/py3dtilers # 799

Duration: 10 min 59 sec
Finished: 2 months ago

VCityTeam/UD-Graph # 200

Duration: 2 min 5 sec
Finished: 2 months ago

master fix(showroom):css about and help

#1319.3 passed

Restart job

Ran for 5 min 53 sec
about a month ago

- Commit e076c2c
- Compare 8ff76c0...e076c2c
- Branch master
- Mathieu Livebardon

- Functional test </> Node.js: 18
- AMD64
- Git

Job log

View config

```
1 Worker information
6
7 Build system information
164
165 OK
166
167 $ git clone --depth=50 --branch=master https://github.com/VCityTeam/UD-Viz.git VCityTeam/UD-Viz
177
178
179 $ npm install 18
185
186 npm ERR! `spin` is not a valid npm option
187
188 npm ERR! A complete log of this run can be found in: /home/travis/.npm_logs/2024-02-21T13_52_40_911Z-debug-0.log
189 Setting up build cache
195
```

Remove log Raw log

- worker_info 0.86s
- system_info 0.01s
- docker_mtu_and_registry_mirrors 0.47s
- resolvconf 3.78s
- git_checkout 7.43s
- npm_install 3.62s
- cache_1

Executed script: your job to write

```
# .travis.yml file
language: cpp
dist: trusty # Ubuntu 14.04 Trusty Tahr

os:
  - linux
  - osx

compiler:
  - gcc
  - clang

# For C++ projects, the env, compiler and os (provided as arrays)
# multiply to construct the build matrix.
matrix:
  fast_finish: true
  exclude:
    - os: osx
      compiler: gcc

addons:
  apt:
    packages:
      - libboost-date-time-dev
      - libboost-filesystem-dev

install:
  - if [ "$TRAVIS_OS_NAME" == "linux" -a "$QT" == "4" ]; then
    | sudo apt-get -y install libqt4-dev libqt4-opengl-dev; fi
  - git clone MY_DEPENDENCY_LIB
  - path MY_DEPENDENCY_LIB

script:
  - ... < Build your binaries, run them, test them > ...
```

CI has some limitations

- Runs are independent from one another
 - Not conceived for expressing numerical experiments
 - How to combine many run results into my graphic ?
- Exfiltration of resulting artifacts is not easy
 - Getting ones hand on results (tables, graphics) is a job
- Description of the context is really limited
 - What if need an upstream streaming feed ?
 - What is I need a patched database ?
- Containers (Docker) domination: **the numerical experiment building block can (will?) be docker-based**

Today's numerical experiments use many sub-sofwarees

```
# Shell script calling docker
echo -e "\n*** CLEANING UP PREVIOUS RUN GARBAGE ***\n"
docker-compose run --rm py3dtilers rm -rf /datademo/lods_3dtilers

echo -e "\n*** STEP1 ***\n"
docker run [...] pc2vol -i /data/galleries.pts --gridstep 0.025 -o /data/gal.vol
docker run [...] dgtal volSurfaceRegularization -i /data/gal.vol -o /data/gal.obj

echo -e "\n*** STEP2 ***\n"
docker run [...] mepp2 triangulate_faces /data/gal.obj /data/gal_triangulated.obj
docker run [...] py3dtilers obj-tiler -i /data/gal_triangulated.obj -o /data/lods_3dtilers
```

- py3tilers, pc2vol, dgtal, mepp2 are independent softwares, with their own source repository, build chain, dependencies...
- A publication reviewer has better chances at verifying results

Experiment with a Direct Acyclic Graph and docker components

The screenshot displays a workflow management interface. At the top, the breadcrumb navigation shows 'Workflows / default / branchflow-r8qcn'. The right side of the header contains 'WORKFLOW DETAILS' and several utility icons. Below the header, a toolbar includes buttons for 'RETRY', 'RESUBMIT', 'DELETE', 'LOGS', 'SHARE', and 'WORKFLOW LINK'. A search bar is located below the toolbar.

The main area shows a Direct Acyclic Graph (DAG) with the following nodes and status:

- branchflow-r8qcn**: Failed (red circle with 'X')
- start**: Succeeded (green circle with checkmark)
- a**: Succeeded (green circle with checkmark)
- b**: Failed (red circle with 'X')
- join**: Pending (grey circle with play button)

The DAG is connected to a detailed view of the container-level workflow, showing a complex tree structure of tasks and their dependencies.

On the right, the 'SUMMARY' tab is active, displaying the following details:

NAME	branchflow-r8qcn
TYPE	DAG
PHASE	Failed
START TIME	6m ago
END TIME	4m ago
DURATION	1m
PROGRESS	2/3
MEMOIZATION	N/A
RESOURCES	1m*(1 cpu),47s*(10Gi ephemeral-storage),33m*(100Mi memory)
DURATION	

A 'MANIFEST' button is located below the summary table. A 'GET HELP' button is in the bottom right corner.

Workflow expression still hurts

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
spec:
  entrypoint: main
  volumes:
  - name: workdir
    persistentVolumeClaim:
      claimName: vcity-pvc
      readOnly: false
  arguments:
    parameters:
      # Numerical experiment related
      - name: boroughs
      - name: pattern
      # Derived parameters
      - name: database_dump_filename
        value: "{{workflow.parameters.experiment_output_dir}}/result-{{workflow.parameters.database_name}}.sql"

  templates:
  - name: main
    steps:

    ### Looping to start databases as DAEMON/services
    - - name: 3dcitydb-start-db-loop
        template: 3dcitydb-daemon-vintaged
        arguments:
          parameters:
            - name: vintage
              value: "{{item}}"
            - name: database_name
              value: "{{workflow.parameters.database_name}}-{{item}}"
            - name: password
              value: "{{workflow.parameters.database_password}}"
            - name: user
              value: "{{workflow.parameters.database_user}}"
            - name: port
              value: "{{workflow.parameters.database_port}}"
          withParam: "{{workflow.parameters.vintages}}"
```

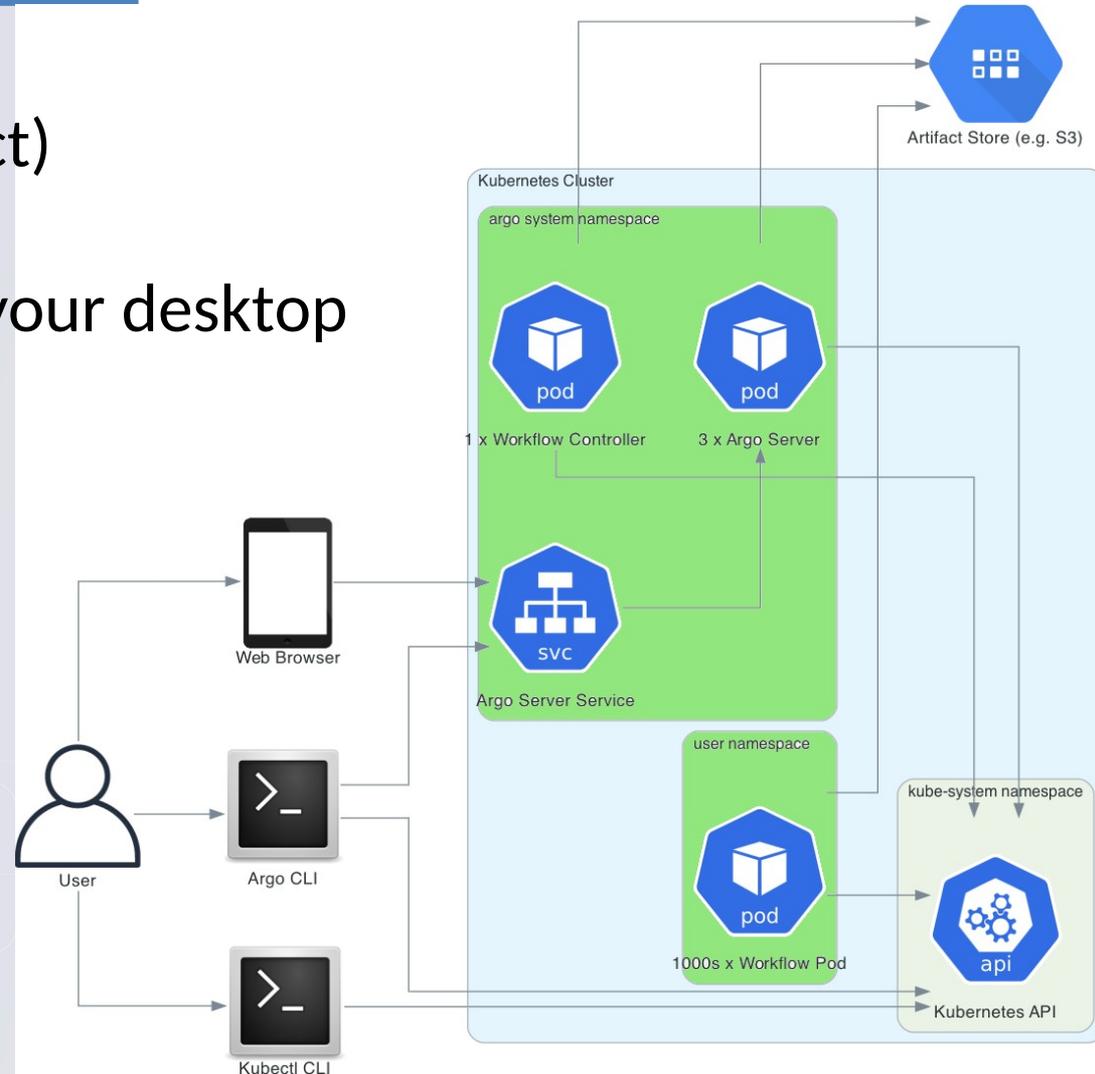
Workflow expression getting better

```
from hera.workflows import DAG, Task, models, Parameter, Workflow
with Workflow(generate_name="import-gml-", entrypoint="main") as w:
    with DAG(name="main"):
        for vintage in inputs.parameters.vintages:
            start_db_t = Task(
                name="start-db-daemon-" + str(vintage),
                template=threedcitydb_containers[vintage],
            )
            import_vintage_boroughs_t = Task(
                name="import-" + str(vintage) + "-boroughs",
                template_ref=models.TemplateRef(
                    name="workflow-import-" + str(vintage),
                    template=db_import_boroughs_template_names[vintage],
                ),
                arguments={"dbhostaddr": start_db_t.ip},
            )
            start_db_t >> import_vintage_boroughs_t
w.create()
```

Reproducing computations requires platforms (just like CI) ... which hurts

Hard to deal with (abstract)

- the Kubernetes layer
- This includes installing your desktop
- Code generation layer



Automatized numerical reproducibility is coming.

Au moins la table ronde permet d'en parler :-)