Resources

Noushin Nabavi & Monica Granados

2021-02-13

Contents

Chapter 1

Preface

A repository to house materials for a open and reproducible workflows in the public service workshop.

The goal of this workshop is to introduce participants to concepts of `reproducible research`. Reproducible research is the idea that data analyses, and more generally, scientific claims, are published with their data and software code so that others may verify the findings and build upon them.

This workshop focuses on the concepts and tools behind reporting modern data analyses in a reproducible manner. As part of this, we introduce tools that enable publishing data analyses in a single document that allows others to easily execute the same analysis to obtain the same results.

Additionally, as part of this workshop, we briefly introduce working with `R` and Rstudio to create a Rmarkdown document. R is a popular statistical computing language, commonly used in many scientific disciplines for statistical analysis, generating production-quality graphics, and automating data workflow tasks.

1.1 Learning goals

By the end of this workshop, EW2 participants will learn be able to:
- Define reproducible research and open workflows - Discuss current issues surrounding reproducibility - Discuss solutions and important components of reproducibility - Identify tools that are used for reproducible and open research

In the demonstration part of the workshop, we will:

- Write basic markdown documents
- Use knitr, rmarkdown and bookdown R packages to build various document types (e.g. PDF, HTML and DOCX)
- Create reproducible rmarkdown documents leveraging .Rproj and .RData
- Create presentations from Rmarkdown documents that include R code
- Work with git version control tools
- Create reproducible and "backed up" analysis via remote repositories (e.g github)

Chapter 2

Introduction to reproducible research

The terms `reproducible research` was coined by Jon Claerbout in the 1980s when he writes an essay on reproducible computational research, and describes the hurdles he faces when making a text book that incorporates text, data, and results in a stand-alone document. According to Claerbout's principle, 'scholarship does not only consist of theorems and proofs but also and perhaps even more important of data, computer code and a runtime environment which provides readers with the possibility to reproduce all tables and figures in an article'.

The need for reproducibility is increasing dramatically as data analyses become more complex and involve larger datasets, analysts, and more sophisticated computations.

2.1 what is reproducible research?

According to a U.S. National Science Foundation (NSF) subcommittee on replicability in science, "reproducibility refers to the ability of a researcher to duplicate the results of a prior study using the same materials as were used by the original investigator. This entails that a result obtained by an experiment or observational study should be achieved again with a high degree of agreement when the study is replicated with the same methodology by different researchers

2.2 Terminology distinctions

Reproducible research is sometimes known as reproducibility, reproducible statistical analysis, reproducible data analysis, reproducible reporting, and literate

programming.

2.3 Reproducible versus replicable

Replicability means obtaining consistent results across studies aimed at answering the same scientific question, each of which has obtained its own data.

2.4 Reproducible versus repeatable

Repeatability measures the variation in measurements taken by a single instrument or person under the same conditions, while reproducibility measures whether an entire study or experiment can be reproduced in its entirety.

This is a way for researchers to verify that their own results are true and are not just chance artifacts.

2.5 Reproducibility crisis

The replication crisis (or replicability crisis or reproducibility crisis) is, as of 2020, an ongoing methodological crisis in which it has been found that many scientific studies are difficult or impossible to replicate or reproduce. The replication crisis affects the social sciences and medicine most severely.

2.6 What needs to be reproduced?

Actual results themselves, which includes: - Tables - Visualizations/figures/graphs - Values reported in the text - The statistical evidence in support of the findings (e.g., p-values, confidence intervals, credible intervals).

2.7 Motivation

Some aspects to consider that may make your experiments, processes, and reports more reproducible: 1. Don't Read Between the Lines. … 2. Be Strict. … 3. Keep Things Transparent. … 4. Collaborate. … 5. Automate Your Processes.

2.8 Benefits of reproducubility

- Increased likelihood that the research will be correct
- Reproducibility makes it easier to check the research
- It is easier to reproduce the research independently
- Easier to extend the research
- Reusable code and instruction resulting in increased efficiencies

2.9 How to make research reproducible

1) The first reason to repeat experiments is simply to verify results. Different science disciplines have different criteria for determining what good results are.
2) The next reason to repeat experiments is to develop skills necessary to extend established methods and develop new experiments. For these, we need reproducible methods, tools, and platforms.

2.10 Requirements for reproducibility

1) The "raw" data is made available, where "raw" refers to the data prior to any manipulation by the researcher (e.g., prior to any data cleaning and transformation).
2) A complete set of instructions is provided explaining all steps used in the processing and analyzing the data.

2.11 Additional requirements

a) A set of files is provided containing the data and code, and it is possible to create the tables and any data-derived charts/graphics/visualizations by running the code.

b) Details about the system being used to run the analysis: operating system, patches, random number seeds, specific versions of all software/packages/libraries are listed.

c) The code is written in a way that can be readily understood.

d) Open/transparent. All the data and materials are available (as opposed to "available upon request") – e.g., posted on GitHub, or in an international data repository.

e) That is, either:

– Another party (e.g., a reviewer) has successfully reproduced the results and certified them as such. – Logs demonstrate that key results were successfully created from the inputs. – The key results are linked to the data and code, so the relationship can be directly inspected.

A final requirement, which is sometimes known as literate programming, is that:

f) The entire report is written using code. That is, a file or files are provided which, when run, import the data, produce all the results, insert the results into the text of the report, and format the report.

Chapter 3

Tools for reproducible projects

"An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures." — Jonathan Buckheit and David Donoho, paraphrasing Jon Claerbout

"In 2002, I felt like I would just remember everything forever," Karl Broman, a biostatistician at the University of Wisconsin, Madison. "It was only later that it became clear that you start to forget things within a month."

3.1 Tools

- R, Rstudio, python, open-refine, good tables
- notebooks (Rmarkdown, Jupyter, holepunch)

3.2 Components to control

- operating system (and libraries)
- R/Python versions
- Package versions

if all of these are controlled, the analysis will always be reproducible

Tools that help: - docker: complete control of operating system - package snapshots - packrat

Chapter 4

Reproducible research project structure

4.1 Setting up an Rproject / Notebook

4.2 Data component

- Tidy data principles
- tidyr and dplyr
- what are data structures?
- what are data frames?
- tidy principles
- missingness
- demo with data

4.3 Documentation component

Chapter 5

Introduction to markdown

- how does rmarkdown work? Advantages, disadvantages
- rmarkdown syntax (examples)

5.1 Rmarkdown and usage

- reproducible reporting with rmarkdown
- dichotomizing Rmarkdown document content to (1) computation component and (2) narrative component (**?**, Yale) collectively called as research compendia
- Research Compedia: a container for the different elements that make up the document and its computations (i.e. text, code, data, …) and as a means for distributing, managing, and updating the collection.

5.2 Computation components

- Objects to be presented: usually plots and tables
- Reproducible code to visualize data

5.3 Narrative components

- Provide backgrounds, goals
- Contexualize computational components
- Establish themes
- Convey the results
- Render in pdf, html, docx

5.4 Rproject set-up

- Allows for integration of the two components (i.e. creation of Rmarkdowns with compuational components)
- Integrates literate programming
- Provide interpretability and facilitate reproducibility

5.5 Demo

- demo one example

Chapter 6

GIT Commands

- GIT Terminology
- GIT Commands
- Resources

6.1 GIT Terminology

- `origin` : connection pointing to the remote repository
- `master` : name of your default branch. A branch in Git is simply a lightweight movable pointer to a commit
- `working directory` : local repository
- `.git directory` : Git stores all of its repository data in the .git directory. This is created when a local repository is initialised using the `init` command
- `.git.ignore` : Git uses this file to determine which files and directories to ignore, before you make a commit
- `hash`: the commit command creates a unique ID called a hash, which is an absolute path
- `HEAD` : pointer to the last commit of the branch you are currently on. If you are on the master branch, then HEAD and master will refer to the same commit. This is a relative path. To see the previous commit use `HEAD~1`

6.2 GIT Commands

Initialise

- `git init <local repository name>`initialises a new local repository

Remotes

- `git remote add <remote name> <url>` creates a new connection to a remote repository is the shortcut for the and is typically set to 'origin'
- `git remote show <remote name>` shows which branch is automatically pushed to when you run git push while on certain branches. It also shows you which remote branches on the server you don't yet have, which remote branches you have that have been removed from the server, and multiple local branches that are able to merge automatically with their remote-tracking branch when you run git pull
- `git remote rename <remote name> <new remote name>` rename a remote
- `git remote remove <remote name>` remove a remote
- `git remote -v` lists the remotes that are configured

Branches

- `git branch <new branch name>` adds a new branch, a structure with trees for saved states of files
- `git checkout <branch name> <filename>` checks out (i.e. switches to another version) an old version of a file
- `git branch` lists all of the branches in a repository, with a * next to the branch you are currently on
- `git checkout <branch-name>` switches to another branch-name
- `git checkout -b branch-name>` creates the branch and switches you to it
- `git merge source destination` merges two branches

Status

- `git status` shows which files have changed/new in your repository
- `git diff` shows the changes you made to the file
- `git diff --staged` shows the difference between the last committed change and what's in the staging area
- `git diff directory` shows the changes to the files in the directory
- `git diff -r HEAD` -r flag refers to compare to a particular revision
- `git log` view the log of the project's history
- `git show <hash>` view the details of a specific commit, with the first few characters of the commit's hash
- `git annotate <filename>` shows who made the last change to each line of a file and when

Clone

- `git clone <remote name>` to clone a repo and download a copy of a repo to a local folder This automatically creates the remote called origin

Add

- `git add` adds from your working directory to your staging area, ie specifies what will go in a snapshot
- `git add <filename>` stages a file
- `git add -A` stages all new, modified and deleted
- `git add <foldername>/*` adds folder and contents to your staging area

Remove

- `git clean -n` shows a list of files that are in the repository, but whose history Git is not currently tracking
- `git clean -f` will then delete those files

Undo

- `git reset` undo ALL changes that have been staged with git add
- `git reset HEAD <filename>` undo changes to a specific filename that have been staged on HEAD

Commit

- `git commit -m "<message>"` commits the file with the snapshot to the `local repository` locally. The commit records the changes to the file ie actually takes the snapshot and makes a permanent record of it

Fetch

- `git fetch` gets any new work since last clone or fetch. Fetch does not however merge remote work with our work

Pull

- `git pull` automatically fetches and then merges that remote branch into your current branch

Push

- `git push` adds the files to your remote git

Note on adding files to the remote:

- When it is your first push from a repo, you will first have to make the link between the local and remote repository via: `git push --set-upstream origin master`, or shorter `git push -u origin master`. As of then, `git push` will refer to the upstream branch you've set: i.e. origin / master

6.3 Resources

- Happy Git and GitHub for the useR by Jennifer Bryan adapted under Creative Commons Attribution-NonCommercial 4.0 International License.
- Pro Git book, written by Scott Chacon and Ben Straub adapted under the Creative Commons Attribution Non Commercial Share Alike 3.0 license

- Version Control with Git by Software Carpentry adapted under the Attribution 4.0 International (CC BY 4.0 license

Chapter 7

Bibliography

Chapter 8

Resources

- Reproducible research with R and RStudio: http://christophergandrud.github.io/RepResR-RStudio/
- Tools for reproducible research: https://kbroman.org/Tools4RR/
- Data privacy and security: https://dataprivacymanager.net/security-vs-privacy/
- BC-Gov framework for github https://github.com/bcgov/BC-Policy-Framework-For-GitHub
- Making slides with Xaringan package in RMarkdown: https://arm.rbind.io/slides/xaringan.html
- Data wrangling with R: https://cengel.github.io/R-data-wrangling/
- Data cleaning with R and tidyverse: https://towardsdatascience.com/data-cleaning-with-r-and-the-tidyverse-detecting-missing-values-ea23c519bc62
- Gallery of missing data visualization: https://cran.r-project.org/web/packages/naniar/vignettes/naniar-visualisation.html
- How does R handle missing values: https://stats.idre.ucla.edu/r/faq/how-does-r-handle-missing-values/
- What does research reproducibility mean? https://stm.sciencemag.org/content/8/341/341ps12
- Challenge to scientists: does your ten-year-old code still run? https://www.nature.com/articles/d41586-020-02462-7?utm_source=twitter&utm_medium=social&utm_content=organic&utm_campaign=NGMT_USG_JC01_GL_Nature#ref-CR1
- Reproducible Research and open science: https://ropensci.github.io/reproducibility-guide/sections/introduction/