Reproducible Research: Goals and Guidelines

Bootcamp 2018

Instructor:
Dawn Koffman, Statistical Programmer
Office of Population Research (OPR)
Princeton University

Reproducible Research

I. Definition and Goals

II. Guidelines

Reproducible Research - Definition

Your research is considered reproducible ...
if someone with access to your raw data, and hardware and software environment, including your code, can generate all of your tables and figures.

Motivating principle: traditional unit of scholarly communication –
published article – is only the tip of the iceberg of the research process.

"An article about computational results is advertising, not scholarship.
The actual scholarship is the full software environment, code and data, that produced the result." - Claerbout and Karrenbach, 1992

Reproducible Research - Goals

Why try to make research reproducible?

1. Demonstrate all decisions made ...
   allow readers to be fully-informed about your work as they judge its credibility.

2. Provide an efficient way to transmit knowledge to future researchers ...
   enable others to easily make use of your methods. Potential for greater impact.

Expose more of the research workflow to the audience ...
audience of your research includes yourself at some later date ...
"future you"

Guidelines for research reproducible …

Keep Code and Data Organized

Encapsulate all files (code, results, data (possibly)) under one directory tree

Separate raw data from cleaned/processed data

RawData/ subdirectory ... don't change names of raw data files

ProcessedData/ subdirectory, SummaryData/ subdirectory, etc.

Separate data from code

Python/ subdirectory, R/ subdirectory, SQL/ subdirectory, etc.

Use relative path names, never absolute pathnames

Choose file and directory names carefully

Try to be descriptive, but don't include spaces in names

Don't use "final" as part of a name ... things are rarely final!

Always Use Scripts

Point and Click, Copy-Paste are Not Reproducible

Get data in most-raw form possible, along with associated metadata

Download external data using scripts, and keep track of date/time of download

Change data format using scripts

Use scripts for all data cleaning

Use scripts for all data analysis

Use scripts to set seeds for random number generation

Use Open-Source Languages and Tools, and Non-Proprietary Data Formats (when feasible)

For example ...

Python or R vs Stata

csv data format vs Excel

Use a Version Control System

Changes to files are recorded, so that specific versions of files can be recalled and reviewed or re-run at a later time.

Automate Processing File Dependencies

Ideally, reproducing results is a one-step operation

- nice not just for others, but for you, and for "future you" (for example, if you get additional raw data)

This *could* be done with a "master (shell) script"

Often better to use a command called **make**

- originally written for generating an executable file by compiling source code files, but can also be used to coordinate any command-line driven process, including running the various scripts that underlie a data analysis project and the construction of figures and tables for a paper

- only the processing that needs to be re-run is re-run, based on time-stamps

Automate Processing File Dependencies

make automates a process and documents the file dependencies in a project:
"this file is generated from these files using this code"

make command reads its instructions from a file called makefile (by default)
makefile contains a series of dependency rules ...

```
targetfile1: dependency1  dependency2 ...  dependencyN
    code to make the target from the dependencies
```

make command compares modification time of target file with modification times of dependency files. Whenever a dependency file has a more recent modification time than its target file, code is executed to recreate the target.

Example:

5 source files:

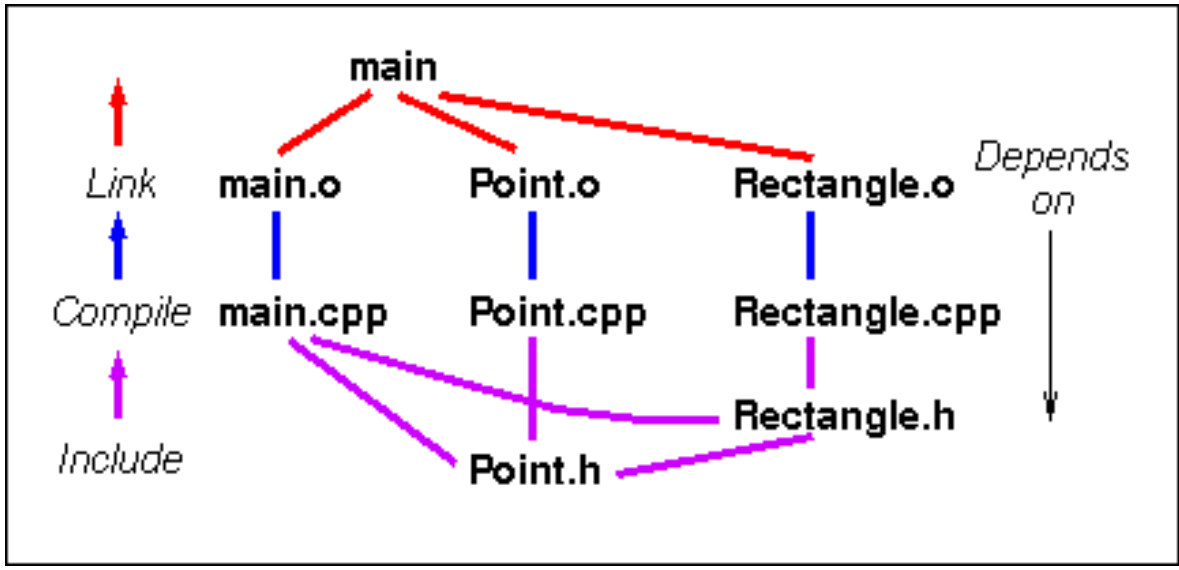
main.cpp Point.h Point.cpp Rectangle.h Rectangle.cpp

Compile source files and create object files:

gcc –c main.cpp → main.o

gcc –c Point.cpp → Point.o

gcc –c Rectangle.cpp → Rectangle.o



Link object files into an executable:

gcc –o main main.o Point.o Rectangle.o

Dependency diagram shows what needs to be recreated if a certain file changes

Suppose main.cpp changes, *what needs to be regenerated*?

What needs to be regenerated if Point.h changes?

Automate Processing File Dependencies

Tedious to keep track of exactly what needs to be done when a file changes.

Use make command to automate processing file dependencies.

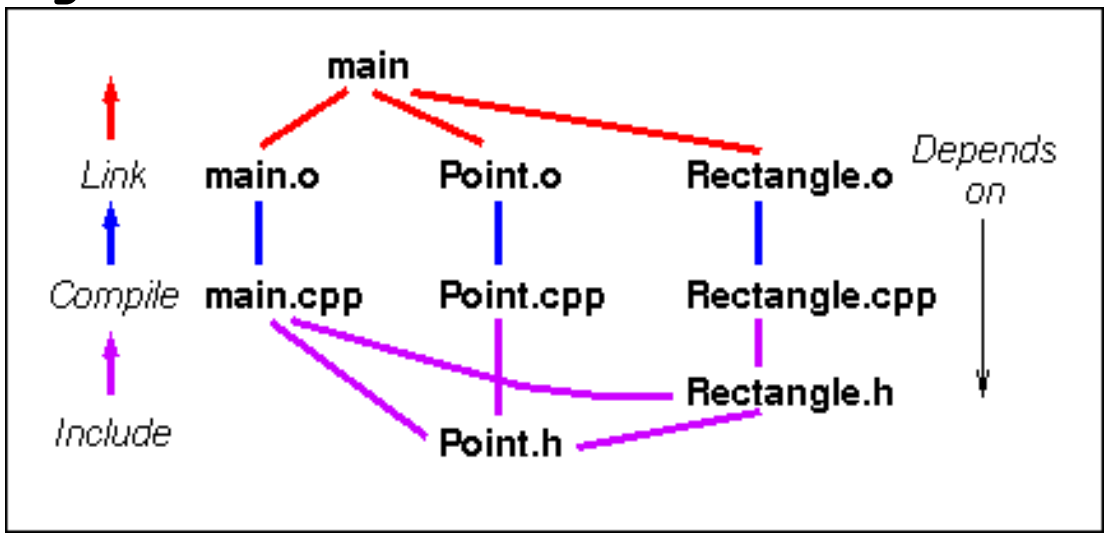
makefile

```
main: main.o  Point.o  Rectangle.o
      gcc –o main main.o Point.o Rectangle.o

main.o: main.cpp Point.h Rectangle.h
      gcc –c main.cpp

Point.o: Point.cpp Point.h
      gcc –c Point.cpp

Rectangle.o: Rectangle.cpp Rectangle.h Point.h
      gcc –c Rectangle.cpp
```



makefiles are machine-readable documentation that make a workflow easily reproducible

Make Code, Documentation and Raw Data Publicly Available (when possible)

Funding agencies may require sharing data
Journals may require sharing code and data
Legal/ethical/contractual considerations regarding sharing data
May need to construct “public-use” version of data or develop “restricted-use” data agreement
Need approval from Institutional Review Board (IRB) to share any data about human subjects
Provide codebook, metadata and documentation for data, along with code

Select a repository for sharing code and data
- general purpose, institutional, discipline-specific, departmental, project website

Choose a software license for your code
- provides copyright (“automatic” copyright does not allow others to use your code)
- provides protection from being held liable if your software doesn’t run as expected
- examples: MIT software license, GPL – General Public License ... read terms carefully

Sad story.