Achieving Practical Reproducibility with Transparency and Accessibility

Brian Lee Yung Rowe
July 29, 2020
Founder & CEO, Pez.AI
CEO, FundKo
Introduction
About Me

• Founder & CEO of Pez.AI, AI Experts (chatbots) helping humans in sales, customer success, talent, and academia (!)
• CEO of FundKo, P2P lender using behavioral economics to improve lending outcomes
• Author of Introduction to Reproducible Science in R, to be published by Chapman and Hall/CRC Press
• 6 years adjunct: NLP, machine learning, predictive analytics, mathematics
• 14 years quantitative finance/investment management
Introduction

Motivation

Science = Methodology + Environment

Reproducibility \sim Transparency + Accessibility

A Transparent and Accessible Methodology

A Transparent and Accessible Environment

Pearls of Wisdom
Motivation
What is truth?

Does hydroxychloroquine increase mortality rate of COVID-19?
Which is correct?

22 May (Lancet, NEJM):

(Source)

2-3 June (Lancet, NEJM):

Well yes, but actually no

(Source)
Non-reproducible science is indistinguishable from anecdote
Reproducing results is universally hard

**IS THERE A REPRODUCIBILITY CRISIS?**

- 52% Yes, a significant crisis
- 38% Yes, a slight crisis
- 7% Don’t know
- 3% No, there is no crisis

1,576 researchers surveyed

Source: Nature
Science = Methodology + Environment
Methodology

- Experiment design - population, data acquisition, data construction
- Data analysis - cleaning, transformation, feature construction, model
- Software development - design, building, testing, versioning, deploying
Buggy code leads to wrong conclusions, which leads to unnecessary retractions.
Environment

- Physical environment - location, climate, conditions, context
- Computing environment - CPU, memory, OS, tools, libraries
A clean, controlled environment is required for physical experiments, so why not computational experiments?
Reproducibility $\sim$ Transparency + Accessibility
transparent adjective

1 a : readily understood

1 b : characterized by visibility or accessibility of information especially concerning business practices the scientific method

(Source: Merriam-Webster)
Counter Example: Opaque Methodology

- vague/no documentation
- no data provenance
- undocumented assumptions
- spaghetti code
- manual steps

(Source: Science Integrity Digest)
accessible adjective

1. : capable of being reached

2. : easily used or accessed by people with disabilities code illiteracy or insufficient computing resources : adapted for use by people with disabilities code illiteracy or insufficient computing resources

(Source: Merriam-Webster)
Counter Example: Inaccessible Environment

We will give a couple of examples on the depth of contemporary neural networks (DNNs). The BERT is a new language representation model which stands for “Bidirectional Encoder Representations from Transformers” (Devlin et al, 2018). In its base form BERT has 110M parameters and its training on 16 TPU chips takes 4 days (96 hours). Another DNN from Radform et al (2019) has 1542M parameters, 48 layers and it needs 1 week (168 hours) to train on 32 TPUv3 chips. NVidia trained a 8.3 billion parameter version of a GPT-2 model known as GPT-2 8B.

(Source: Energy considerations for training deep neural networks)

**Cost:** $32/hour * 168 hours = $5376

- massive system requirements
- esoteric system requirements
- esoteric dependencies
- non-free data/tools

NEW! GPT-3 has 175B parameters and costs ≥ $4.6mm to train. (Source)
Rowe’s Law of Reproducibility

\[
\text{max } \text{automation} \rightarrow \text{max reproducibility}
\]
Scientific method is a workflow.

Workflows are algorithms.

Programs maximally automate algorithms.

\[ \therefore \text{ Programs maximally automate scientific method. } \]
Automation increases transparency and accessibility
A Transparent and Accessible Methodology
Automate Modeling Workflows

- Automate each workflow as independent pipeline
- Create a facade for each pipeline
- Show examples of how to run each pipeline
- Use `make` (or similar) to make workflows accessible
- Use logging to increase transparency of process
- Document key data structures
- Keep humans out of workflows!
Follow A Software Development Process

- Use standard packaging conventions
- Use git (or similar) source control and version numbers
- Write non-trivial tests
- Use continuous build systems to identify breaking changes
"Everything should be made as simple as possible, but no simpler"

- Use standard project conventions
- Be consistent
- Minimize dependencies
- Avoid "clever" code
- Avoid object-oriented programming (use graphs!)
Example: Build Package and Train Model

Inaccessible

1. `$ python setup.py develop`
2. `$ pytest test`
3. `$ R CMD build --compact-vignettes=gs+qpdf $MY_PACKAGE`
4. `$ R CMD check --as-cran $MY_PACKAGE.tar.gz`
5. `$ Rscript --vanilla -e \`
   "library($MY_PACKAGE); \`
   withCallingHandlers(train_model($INPUT, $OUTPUT), \`
   \`
   warning=function(w) stop(w))"`

Accessible

1. `$ make build test train performance`
A Transparent and Accessible Environment
Clean environments can be created automatically on demand

- Cloud IaaS provides virtual hardware
- Containers provides virtual OS
FIGURE 9.2: An extended view of Docker commands. Multiple containers can be created from the same image and commands can be executed on a running container. Built images are stored locally in a repository that must be cleaned periodically.
Container-oriented Development

Using containers during development ensures clean-room environment

- Run R/Python in container
- Run R server in container
- Run notebook in container
- Attach other processes to container (e.g. bash, R session)
Example: Dockerfile

Environment creation becomes transparent via automation

```
FROM jupyter/minimal-notebook:dc9744740e12
MAINTAINER rowe@zatonovo.com

USER root
ENV DEBIAN_FRONTEND noninteractive

RUN \
  apt-get update && \
  apt-get install -qy software-properties-common && \
  add-apt-repository -y ppa:opencpu/opencpu-2.0 && \
  apt-get update && \
  apt-get install -qy opencpu-server x11-apps

# Set opencpu password so that we can login
RUN \
  echo "opencpu:opencpu" | chpasswd
```
Dependency managers

System packages (debian)

1. RUN `apt-get install -qy package-1 package-2`

Python packages

1. RUN `pip3 install package-1 package-2`
2. RUN `pip3 install -r requirements.txt`

R packages (crant)

1. RUN `rpackage package-1 package-2`
Example: GCP

```
$ docker build -t [DOCKER_IMAGE] .
$ docker push [DOCKER_IMAGE] .
$ gcloud compute instances \
    create-with-container [INSTANCE_NAME] \
    --container-image [DOCKER_IMAGE]
```
Create collection of connected services

FIGURE 2.7: Automation required to operationalize a model. This workflow is meant to run without any human intervention. Two types of data can be extracted. In some applications, the model is updated incrementally with new or updated training set. Once the model is trained, a different extraction process is used to fetch data for operational use of the model. Operational data is either real-time (event-based) or fetched as microbatches based on a schedule or buffered process.
Example: Run Notebook Server

Inaccessible

$ docker run -it -p 8888:8888 $(MOUNT_HOSTDIR) -u jovyan -w /app/$(PACKAGE)/notebooks $(IMAGE)
jupyter notebook --allow-root

Accessible

$ make notebook
Pearls of Wisdom
Apply Empathy and Consider Others Using Your Code

Transparent:

- Self-contained workflows with no hidden (e.g., manual) steps
- Documentation that explains decisions/rationale for algorithms
- Consistent, simple code that is easy to read and debug
- Right tool for job
Apply Empathy and Consider Others Using Your Code

Transparent:

- Self-contained workflows with no hidden (e.g., manual) steps
- Documentation that explains decisions/rationale for algorithms
- Consistent, simple code that is easy to read and debug
- Right tool for job

Accessible:

- Easy to use end-user interfaces (e.g., make)
- Dataset easily acquired
- No human in the loop workflows
- Minimal dependencies
- Minimal cost
Assume Everything Will Be Done Again

Focus on repeatability:

- Use containers to exactly create your environment
- Use Linux to simplify scripting/automation
- Write executable scripts (e.g., bash) to document processes
- Minimize interactive development
- Create functions as much as possible
- Use error handling to avoid frustration
Thank You

Questions: rowe@zatonovo.com

Twitter: @cartesianfaith

Personal website: https://cartesianfaith.com