Process Automation as the Backbone of Reproducible Science

Brian Lee Yung Rowe

June 4, 2020

Founder & CEO, Pez.AI
CEO, FundKo

Introduction

About Me

- Founder & CEO of Pez.AI, productivity chatbots that make communication and coordination more efficient
- CEO of FundKo, P2P lender in Philippines 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

Outline

- Why reproducible science?
- Process automation: code as methodology
- Process automation: computing environment
- Pearls of wisdom

Why Reproducible Science

What is truth?

Does hydroxychloroquine increase mortality rate of COVID-19?

Which is correct?

22 May (Lancet, NEJM):

(Source)

Which is correct?

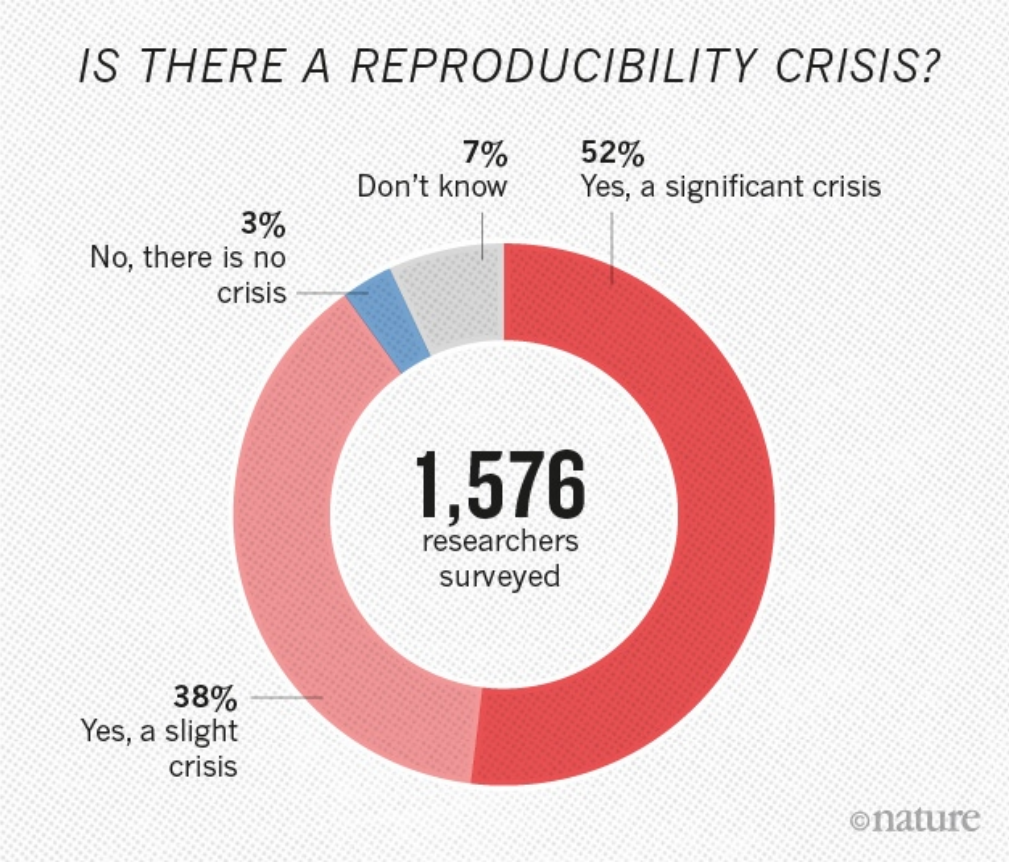
22 May (Lancet, NEJM):

(Source)

2-3 June (Lancet, NEJM):

(Source)

Reproducing results is hard



Source: Nature

Truth as a gradient

Truth only exists when claims can be confirmed

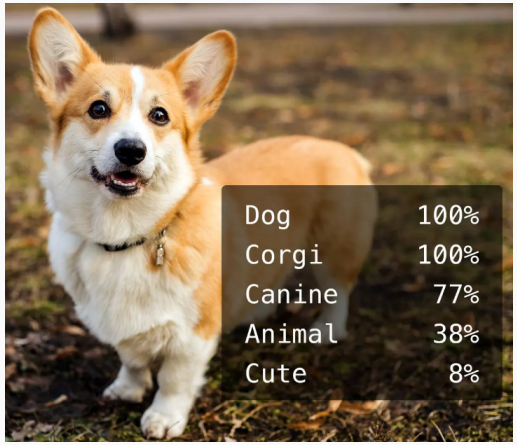
	Repeatable	Reproducible	Replicable	Generalizable
Hypothesis	same	same	same	same
Data	same	same	different	different
Code	same	same	different	same
User	same	different	different	either
Result	exact	exact	comparable	comparable
Certainty	weak	moderate	strong	moderate

TABLE 2.2: A simplified comparison of the hurdles to certainty. In all cases the same hypothesis is used. To achieve more certainty in the validity of a hypothesis, more variation is introduced in the experimental setup.

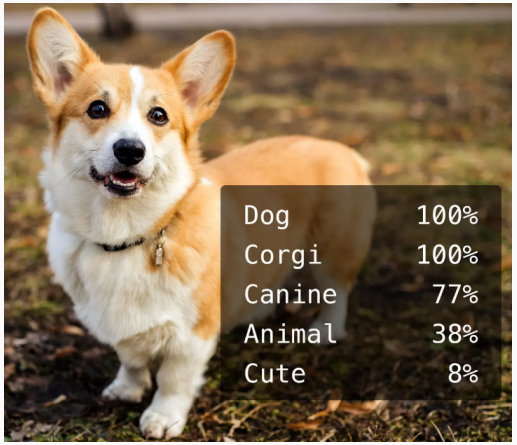
Source: *Introduction to Reproducible Science in R* – Brian Lee Yung Rowe (based on NSF definitions)

Example: Computer Vision/Image Classification

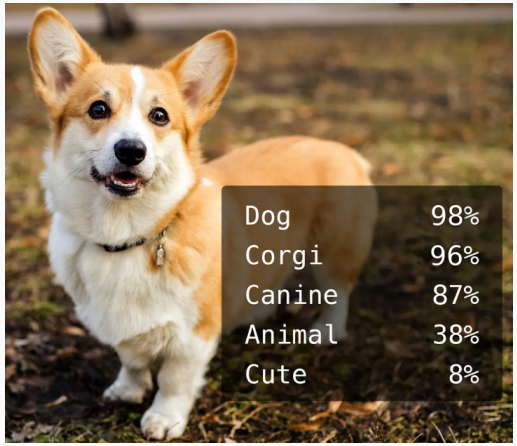
Repeatable



Reproducible



Replicable



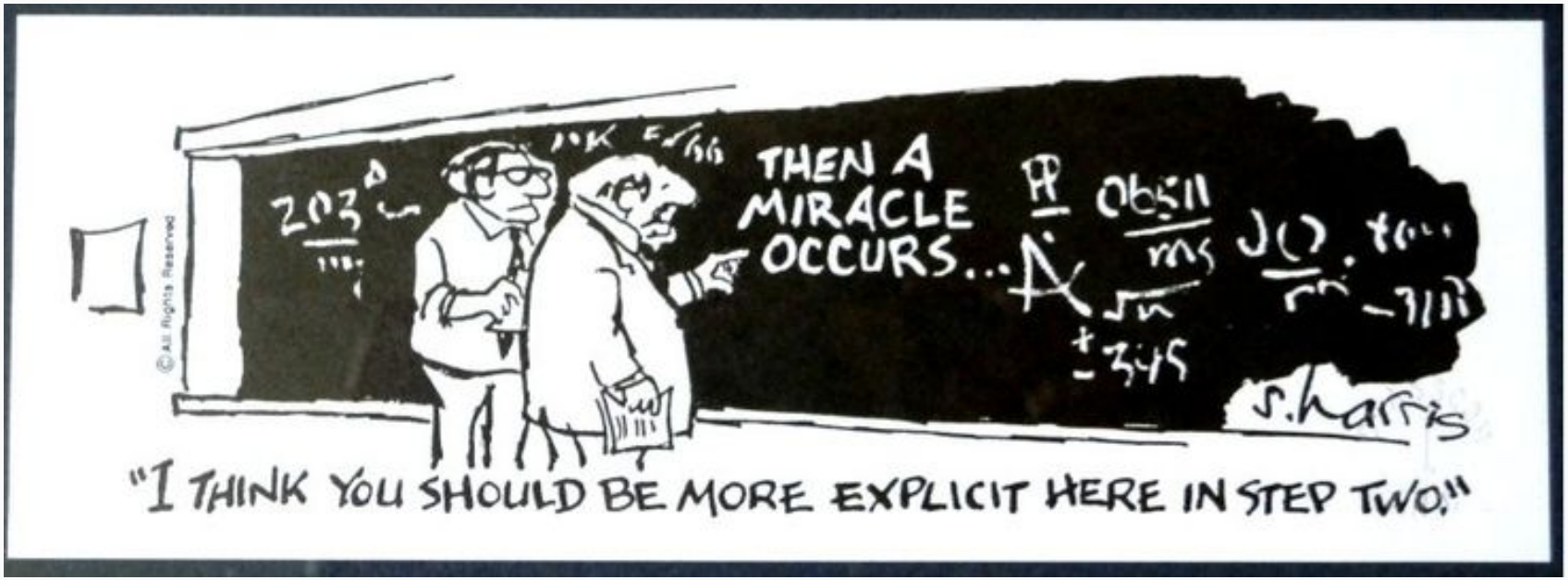
Generalizable (FAIL)

Factors Driving Reproducibility

Research is more reproducible when methodology and environment are transparent and accessible

$$\textit{reproducibility} \sim \textit{methodology} + \textit{environment} \tag{1}$$

Opaque Methodology



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

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

Process Automation: Methodology

Reproducibility as optimization problem

Objective: minimize time to reproduce results

$$\min \textit{reproduce}(\textit{env}, \textit{method}) \tag{2}$$

s.t.

env, method are transparent and accessible

The Data Science Paradox

The worse you are at programming, the more time you spend doing it.

$$t_{coding} \sim skill_{coding} \tag{3}$$

The Data Science Paradox

The worse you are at programming, the more time you spend doing it.

$$t_{coding} \sim skill_{coding} \tag{3}$$

Corollary

$$bugs \sim skill_{coding} + testing \tag{4}$$

The Data Science Paradox

The worse you are at programming, the more time you spend doing it.

$$t_{coding} \sim skill_{coding} \tag{3}$$

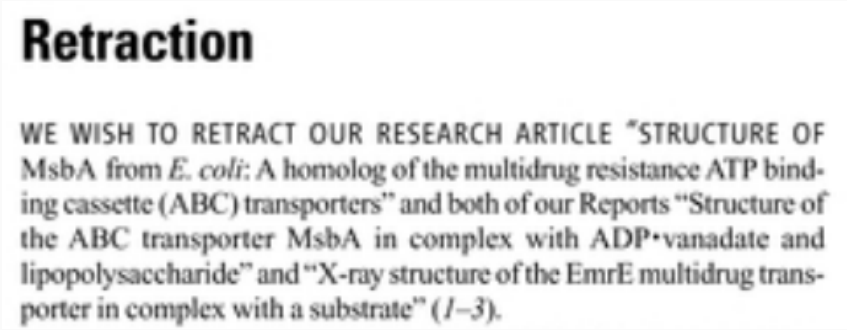
Corollary

$$bugs \sim skill_{coding} + testing \tag{4}$$

Conclusion: Rigorous software development practice is required for reproducibility

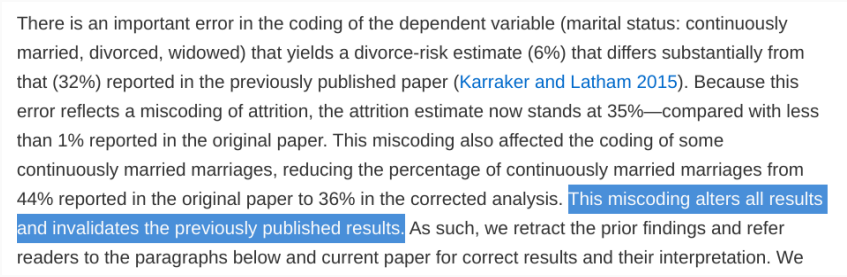
Code Is Your Methodology

Transparent and accessible code is easier to understand, faster to confirm, identify, troubleshoot bugs and avoids unnecessary retractions.

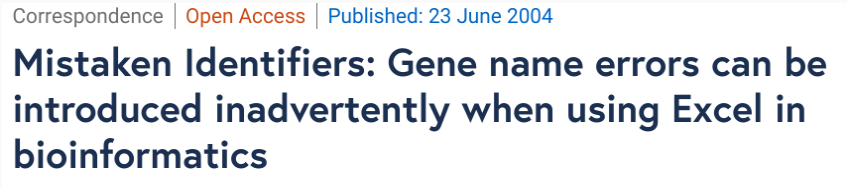


Retraction

WE WISH TO RETRACT OUR RESEARCH ARTICLE "STRUCTURE OF MsbA from *E. coli*: A homolog of the multidrug resistance ATP binding cassette (ABC) transporters" and both of our Reports "Structure of the ABC transporter MsbA in complex with ADP•vanadate and lipopolysaccharide" and "X-ray structure of the EmrE multidrug transporter in complex with a substrate" (*1*–*3*).



There is an important error in the coding of the dependent variable (marital status: continuously married, divorced, widowed) that yields a divorce-risk estimate (6%) that differs substantially from that (32%) reported in the previously published paper (Karraker and Latham 2015). Because this error reflects a miscoding of attrition, the attrition estimate now stands at 35%—compared with less than 1% reported in the original paper. This miscoding also affected the coding of some continuously married marriages, reducing the percentage of continuously married marriages from 44% reported in the original paper to 36% in the corrected analysis. This miscoding alters all results and invalidates the previously published results. As such, we retract the prior findings and refer readers to the paragraphs below and current paper for correct results and their interpretation. We



Correspondence | Open Access | Published: 23 June 2004

Mistaken Identifiers: Gene name errors can be introduced inadvertently when using Excel in bioinformatics

Automate Modeling Workflows

- Collecting/generating data
- Exploring data
- Parsing/transforming data
- Training a model
- Model validation
- Predictions
- Reporting

Automate Software Development Workflows

- project initiation
- building code
- testing code
- deploying code
- running code
- storing/versioning code

Project Initiation

Use standard directory structures and consistent naming conventions

Path	Description
~/workspace/caffeine	Project home
~/workspace/caffeine/bin	Executable scripts
~/workspace/caffeine/data	Package data
~/workspace/caffeine/inst	Other package files
~/workspace/caffeine/man	Package documentation
~/workspace/caffeine/notebooks	Jupyter notebook files
~/workspace/caffeine/private	Non-package data
~/workspace/caffeine/R	R source code
~/workspace/caffeine/tests	Test scripts

TABLE 3.1: Directories within an R project. Some directories, such as `bin`, `notebooks`, and `private` are not part of R package conventions.

```
1 #!/bin/bash
2
3 project=$1
4 mkdir -p $project/R $
      project/tests
5 cd $project
6 git init
```

Building Code

Create package, train model, build image

Inaccessible

```
$ R CMD build --compact-vignettes=gs+qpdf $MY_PACKAGE
```

Accessible

```
$ make build
```

Testing Code

Inaccessible

```
$ R CMD check --as-cran $MY_PACKAGE.tar.gz
$ pytest test
```

Accessible

```
$ make test
```

Train Model

Inaccessible

```
$ Rscript --vanilla -e \
    "library($MY_PACKAGE); \
     withCallingHandlers(train_model($INPUT, $OUTPUT), \
       warning=function(w) stop(w))"
```

Accessible

```
$ make train performance
```

Run Notebook

Inaccessible

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

Accessible

```
$ make notebook
```

Generate Report

Inaccessible

```
$ R

> library(rmarkdown)
> rmarkdown::render("reports/myreport.Rmd")
```

Accessible

```
$ make REPORT=reports/myreport.Rmd report
```

Process Automation: Environment

The Computing Environment

- Processor
- Memory
- Cache
- Disk
- Networking
- Operating system
- Software dependencies
- Package dependencies

Interlude: Computing History

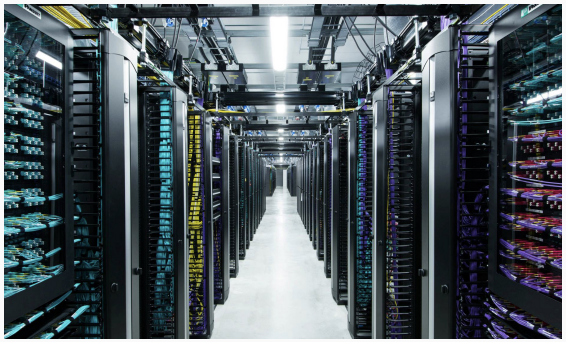
1970s

Dumb terminals - mainframes

late 1970s - 2000s

Personal computers

2000s - present



Smart terminals - mainframes

21st Century Virtual Solution

Environment creation and management is virtual and automatable

- Cloud IaaS solves hardware issues
- Containers solve software issues

Containers

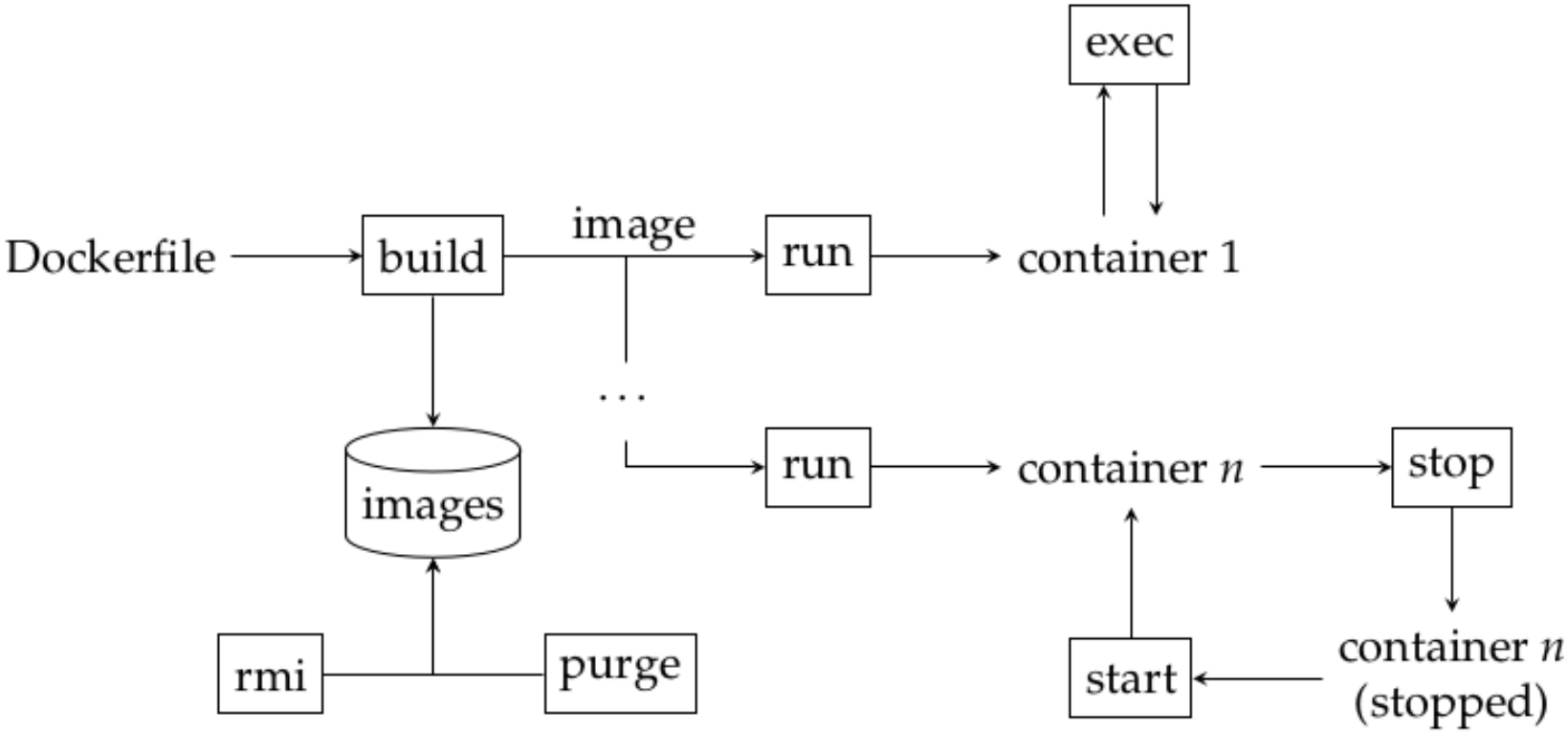


FIGURE 9.2: An extended view of Docker commands. Multiple containers can be created from the same image and commands can be `exec`uted on a running container. Built images are stored locally in a repository that must be cleaned periodically.

Example: Dockerfile

Environment creation becomes transparent via automation

```
1  FROM jupyter/minimal-notebook:dc9744740e12
2  MAINTAINER rowe@zatonovo.com
3
4  USER root
5  ENV DEBIAN_FRONTEND noninteractive
6
7  RUN \
8    apt-get update && \
9    apt-get install -qy software-properties-common && \
10   add-apt-repository -y ppa:opencpu/opencpu-2.0 && \
11   apt-get update && \
12   apt-get install -qy opencpu-server x11-apps
13
14 # Set opencpu password so that we can login
15 RUN \
16   echo "opencpu:opencpu" | chpasswd
```

Dependency managers

System packages (debian)

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

Python packages

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

R packages (crant)

```
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]
```

Orchestration

Create collection of connected services

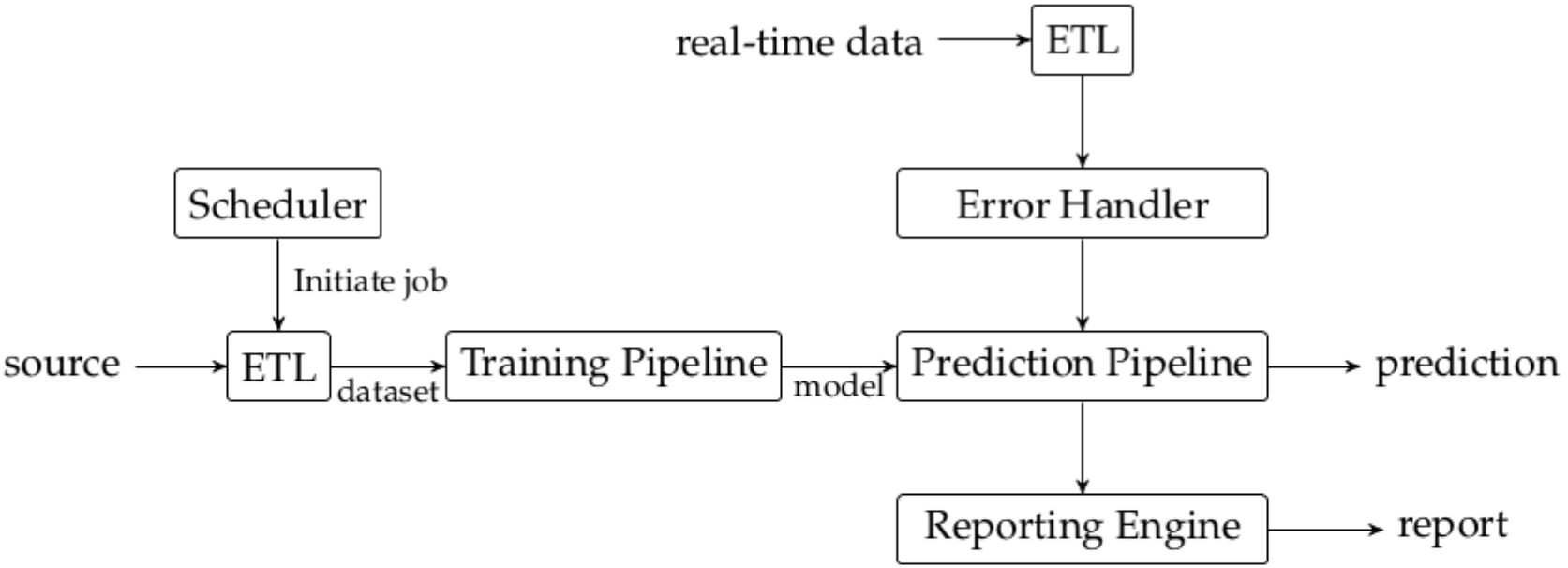


FIGURE 2.7: Automation required to operationalize a model. This workflow is meant to run without any human intervention. Two types of data can be extracted. In some applications, the model is updated incrementally with new or updated training set. Once the model is trained, a different extraction process 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.

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

Twitch: @cartesianfaith ← experiment for live data science/coding help

Twitter: @cartesianfaith

Personal website: https://cartesianfaith.com