Achieving complete computational research reproducibility using containers (and why we need it)

Mark Ziemann PhD, GCHELT
mark.ziemann@burnet.edu.au

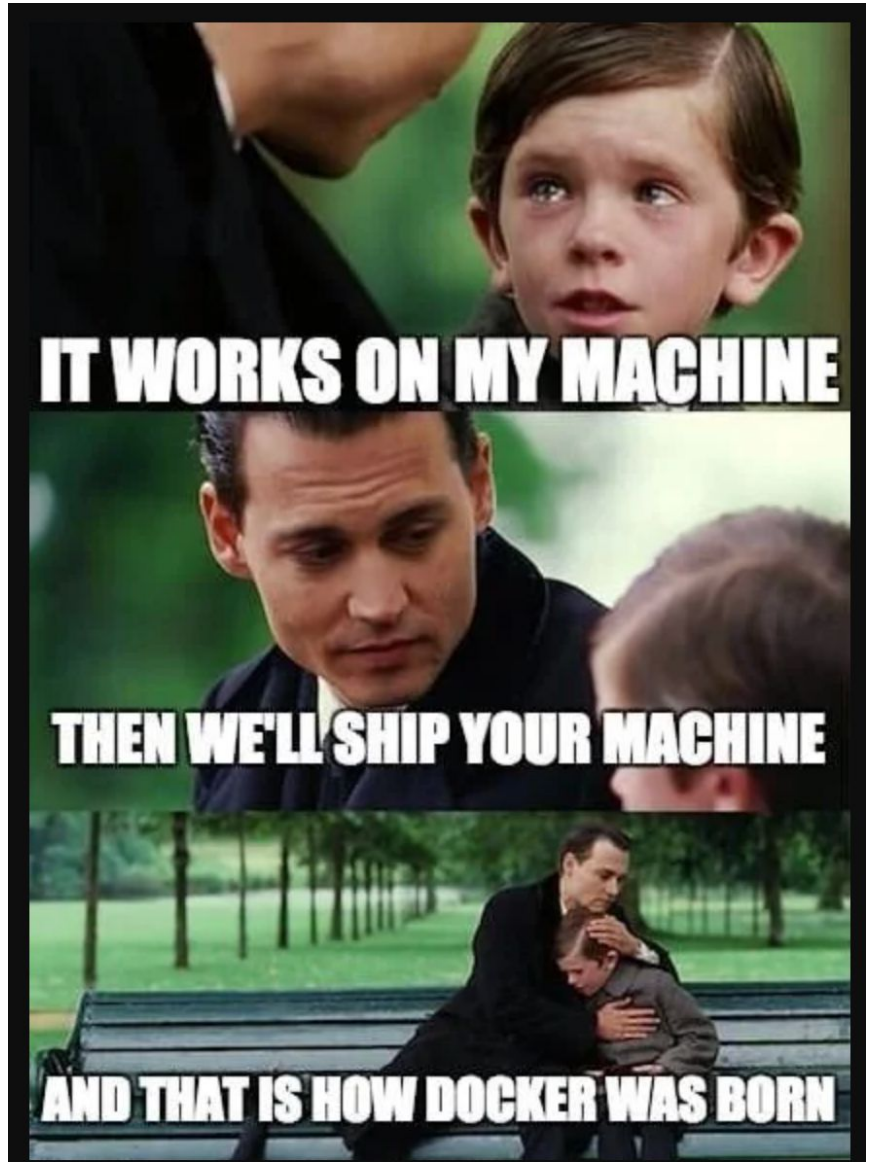
AIMOS 2025

Burnet
reach for the many

AT BURNET INSTITUTE, WE PROUDLY ACKNOWLEDGE THE BOON WURRUNG PEOPLE OF THE KULIN NATIONS AS THE TRADITIONAL CUSTODIANS OF THE LAND ON WHICH OUR OFFICE IS LOCATED AND RECOGNISE THEIR CONTINUING CONNECTION TO LAND, WATERS AND COMMUNITY. WE ACKNOWLEDGE ABORIGINAL AND TORRES STRAIT ISLANDER PEOPLES AS AUSTRALIA'S FIRST PEOPLES AND ACKNOWLEDGE THAT SOVEREIGNTY WAS NEVER CEDED. WE PAY OUR RESPECT TO ELDERS PAST AND PRESENT, AND EXTEND THAT RESPECT TO ALL FIRST NATIONS PEOPLE.

Overview

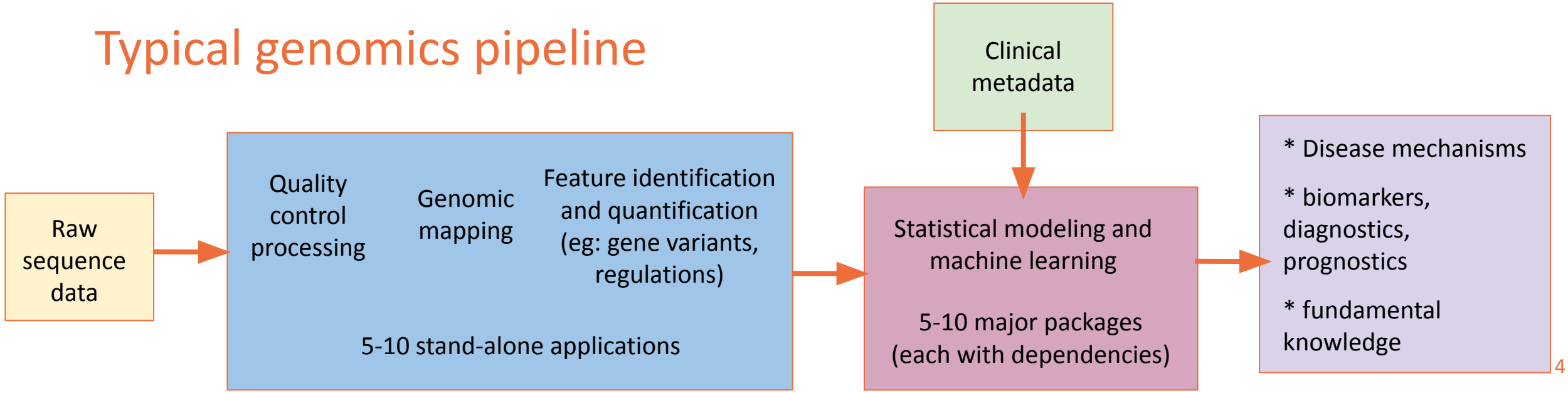
- ➔ Computational reproducibility in bioinformatics
- ➔ Genomics gone wrong
- ➔ Best practices for computational reproducibility
- ➔ What is a container anyway?
- ➔ How to get started using containers



What do we mean by complete computational reproducibility?

- Materials made available by the author allow for the reproduction of the results including charts and tables from the raw data
- Clearly documented and unambiguous steps → can be reproduced in a reasonable time
- Results match exactly
- Works on general purpose computers

Typical genomics pipeline



How reproducible is bioinformatics?

- A 2009 systematic evaluation showing only 2 of 18 articles could be reproduced (11%) [1]
- In 2020 an NIH pilot study tried to replicate 5 bioinformatics projects but couldn't reproduce *any* [2]
- In 2024, a systematic analysis of Jupyter notebooks in biomedical articles showed only 879/22578 notebooks (2.9%) gave similar results [3]
- Out of 20 studies using a bioinformatics web tool, only 20% were considered highly reproducible [4]

Overall, only ~10% of bioinformatics papers are reproducible, due to lack of data and code sharing, poor documentation and broken code.

No one is checking

1. Ioannidis et al, 2009 ; 2. Zaringhalam and Federer 2020; 3. Samuel and Mietchen 2024; 4. Bora and Ziemann 2023

A case study in genomics

Potti et al (2006) had a number of problems:

- Swapped “case” and “control” labels
- Some patients duplicated
- Some results ascribed to wrong drug
- Lack of documentation and code
- Likely analysed data with Excel, MatLab and other tools

The Annals of Applied Statistics
2009, Vol. 3, No. 4, 1309–1334
DOI: 10.1214/09-AOAS291

DERIVING CHEMOSENSITIVITY FROM CELL LINES: FORENSIC BIOINFORMATICS AND REPRODUCIBLE RESEARCH IN HIGH-THROUGHPUT BIOLOGY

BY KEITH A. BAGGERLY[1] AND KEVIN R. COOMBES[2]

nature medicine

Article | Published: 22 October 2006

Genomic signatures to guide the use of chemotherapeutics

Anil Potti, Holly K Dressman, Andrea Bild, Richard F Riedel, Gina Chan, Robyn Sayer, Janiel Cragun, Hope Cottrill, Michael J Kelley, Rebecca Petersen, David Harpole, Jeffrey Marks, Andrew Berchuck, Geoffrey S Ginsburg, Phillip Febbo, Johnathan Lancaster & Joseph R Nevins

Nature Medicine **12**, 1294–1300 (2006) | Cite this article

5. Potti et al, 2006; 6. Baggerly & Coombes 2010.

Case study outcome

- Retraction of at least 9 research papers
- Three clinical trial ran from 2007 to 2010 involving 117 patients [11]
- Potti was suspended and he later resigned after investigations found fraudulent claims in other internal documents including grant applications
- CancerGuide Diagnostics company collapsed
- Duke was served eight lawsuits from families of deceased trial participants seeking compensation
- Reputational loss

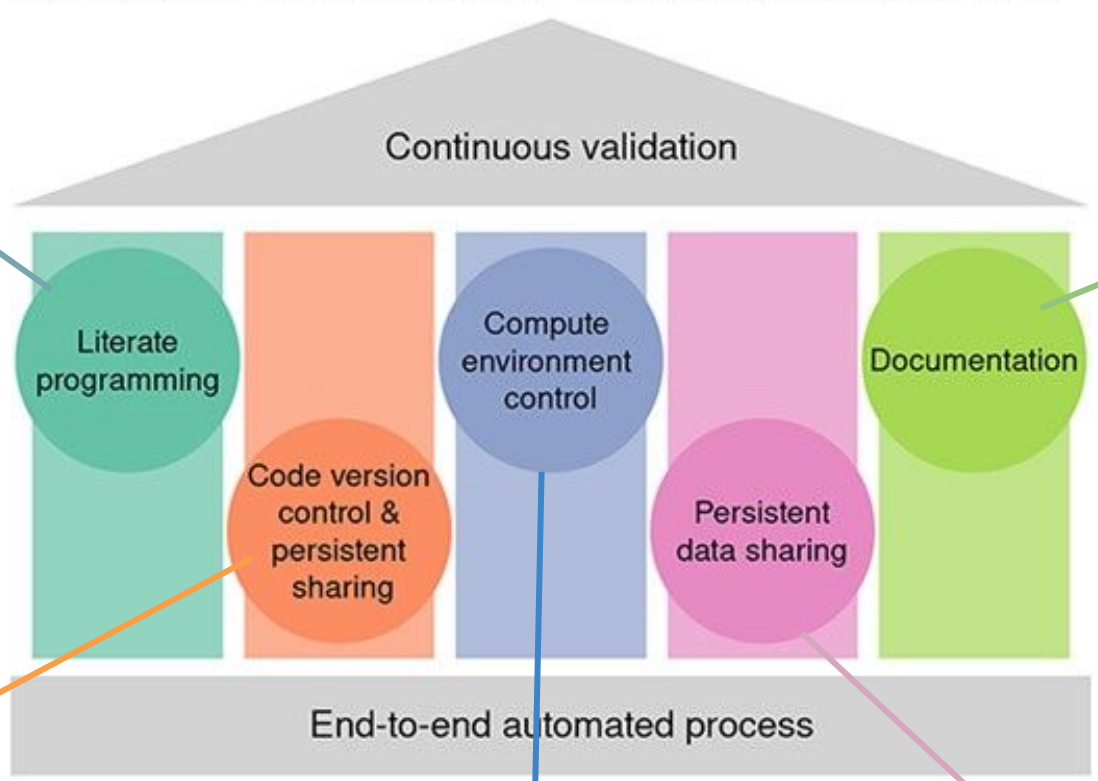
Root cause of the problem

- Simple errors in Excel caused sample mix-ups and dramatic downstream consequences [5]
- Lack of systematic, automated analysis process
- Lack of transparency, sanity checking and auditing
- Severe lack of documentation
- Pressure to publish

7. The Cancer Letter, 2015; 8. Kaiser 2015.

Five pillars framework for computational reproducibility

Use R Markdown Jupyter or another notebook!



Document all your resources so others know how to reproduce it!

Use GitHub! (or similar)

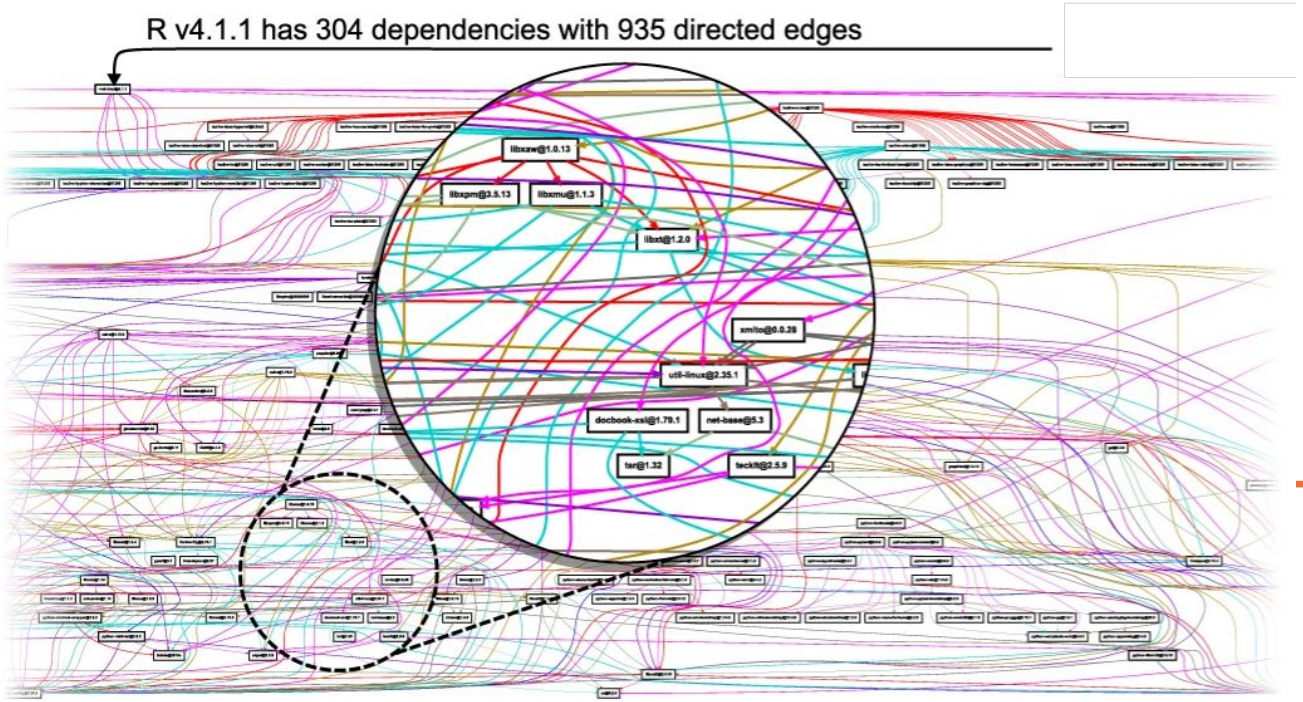
Containers make reproducibility so much easier as all dependencies are included

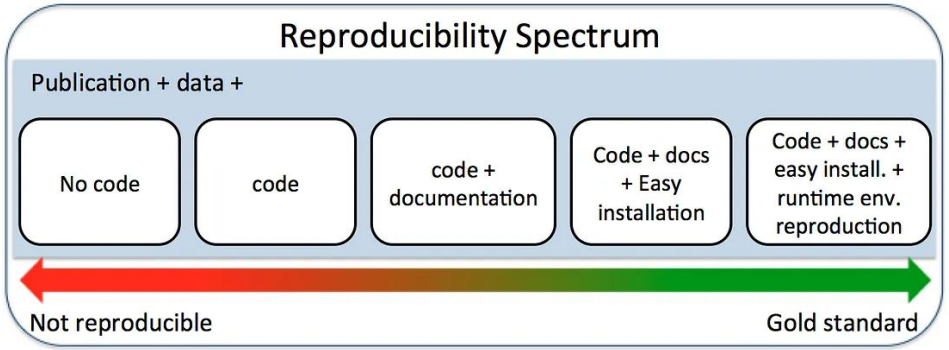
Make the data findable, in a proper repository

9. Ziemann et al 2023

Why we need compute environment control

- Code and data are insufficient to guarantee computational reproducibility. We also need the "environment" - the set of software dependencies.
- Eg: R v4.1.1 has 304 system dependencies. Differences in these can alter behaviour.
- Older R/python versions may not work on modern OSs.
- Older software packages might not be available.
- Bioinformatics and machine learning workflows involve highly complex software configurations.
- Difficult and time-consuming to reproduce the compute environment, and scripts will probably still fail to run properly.



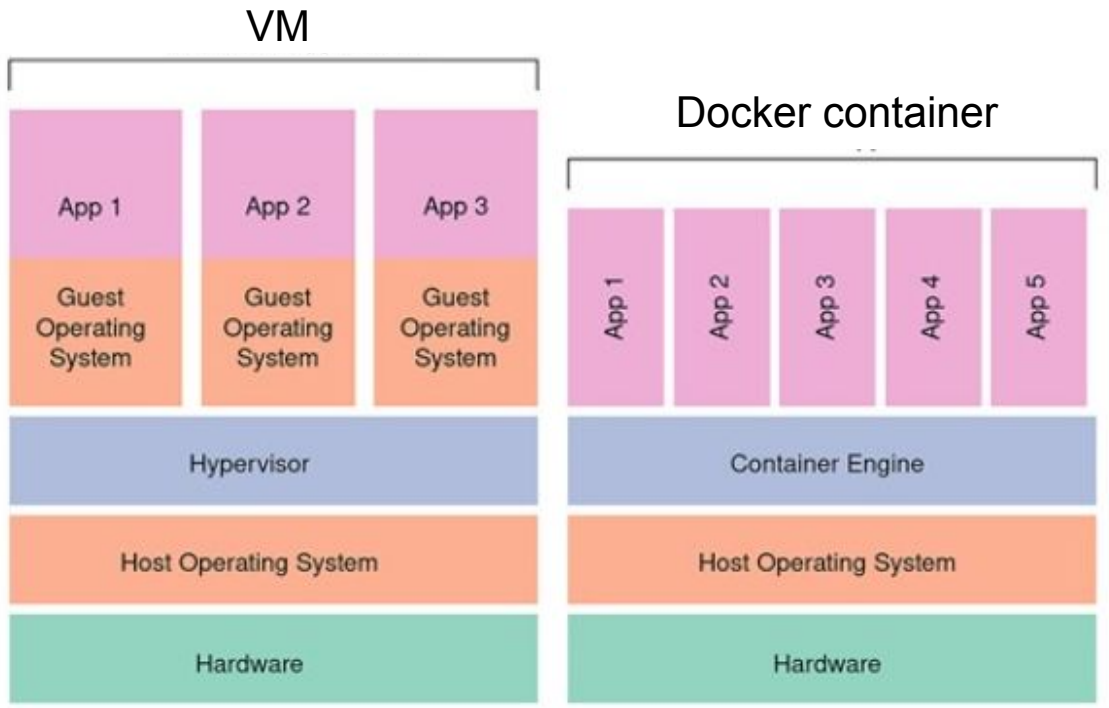


IDEA: Compute environment should be considered part of the data retention responsibilities (5+ years after publication).

10. Vallet et al, 2022; 11. Akalin 2018

Pillar 3: Compute environment control

- “Virtual machines” are guest operating systems, which allow us to run software separate from the host.
- Containers are more lightweight as they don’t replicate components from the host.
- Docker is the most used container app.
- Apptainer works on shared systems like HPCs.
- Researchers can make a Docker image containing all of the software required for each project.
- Images can be shared on Dockerhub or biocontainers, then archived to Zenodo.



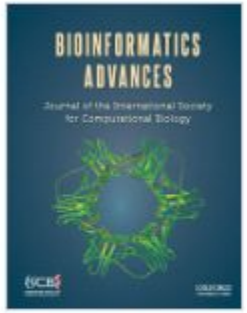
A complicated bioinformatics workflow can be reproduced in just a few minutes with Docker

```
sudo apt update && sudo apt install docker.io -y # install docker
sudo docker run -it jsmith/myproject bash # enter container
Rscript -e 'rmarkdown::render("analysis.Rmd")' # execute workflow
exit # exit container
docker cp $(docker ps -aql):/myproject/analysis.html . # copy report to host system
firefox example.html # inspect results
```

What about Conda?

Feature		
Type	Containerization (OS-level virtualization)	Package & environment manager
Scope	Encapsulates the entire OS environment: libraries, dependencies, executables, and even the kernel interface.	Manages Python/R packages and binaries, but relies on the host OS and system libraries.
Isolation level	Very high – full sandboxed environment.	Moderate – isolated package environments, but shares system libraries and drivers.
Reproducibility	Very high — container image can be rebuilt exactly, anywhere.	Moderate — environment.yml can lead to version drift or binary incompatibility across OSes.
Performance	Near-native speed, minimal overhead.	Near-native speed, but less isolation.
Portability	Excellent — “works anywhere with Docker or Singularity installed.”	Limited — Conda envs may fail to resolve or install identically across systems.

Practicing what we preach



Volume 4, Issue 1
2024

JOURNAL ARTICLE

Two subtle problems with overrepresentation analysis

Mark Ziemann ✉ , Barry Schroeter , Anusuiya Bora

Bioinformatics Advances, Volume 4, Issue 1, 2024, vbae159,
https://doi.org/10.1093/bioadv/vbae159
Published: 21 October 2024 **Article history** ▾

- Publicly available data
- Code on GitHub and Zenodo
- Docker image on Zenodo
- R/Shiny tool for interacting
- Validated data-to-manuscript workflow

```
sudo apt update && sudo apt install docker.io -y # install docker
sudo docker run -it mziemann/background bash # enter container
cd analysis
Rscript -e 'rmarkdown::render("main.Rmd")' # execute workflow
exit # exit container
docker cp $(docker ps -aql):/background docker_results  # copy report to host system
```

12. Ziemann et al, 2024

Getting started with containers

1. Install Docker (or Apptainer)
2. Make an inventory of the software used in a project
3. Write a Dockerfile that loads necessary software, add it to your GitHub repo
4. Build Docker image (go to #3 to fix any errors)
5. Write a script in such a way that it can access the raw data (automatically download from a public repository is best) and analyse it
6. Run the script inside the container (go to #3 or #5 to fix any errors)
7. Return the data to host computer, check it
8. Share image on Dockerhub and archive to Zenodo, institutional RDS

Tutorial on GitHub

https://github.com/markziemann/docker_for_r_tutorial

Anatomy of a Dockerfile

```
FROM bioconductor/bioconductor_docker:3.21-R-4.5.2

# Update apt-get
RUN apt-get update -y \
        && apt-get upgrade -y \
        && apt-get install -y nano git \
        ## Remove packages in '/var/cache/' and 'var/lib'
        ## to remove side-effects of apt-get update
        && apt-get clean \
        && rm -rf /var/lib/apt/lists/*

# Install CRAN packages
RUN Rscript -e 'install.packages(c("gplots","eulerr","kableExtra"))'

# Install bioconductor packages
RUN Rscript -e 'BiocManager::install(c("getDEE2","DESeq2"))'

# get a clone of the codes using HTTPS
RUN git clone https://github.com/markziemann/docker_for_r_tutorial.git

# Set the container working directory
ENV DIRPATH=/docker_for_r_tutorial
WORKDIR $DIRPATH
```

Choose the base image

Install system and python packages here

Get all R packages

Get project code

Building and running a Docker image

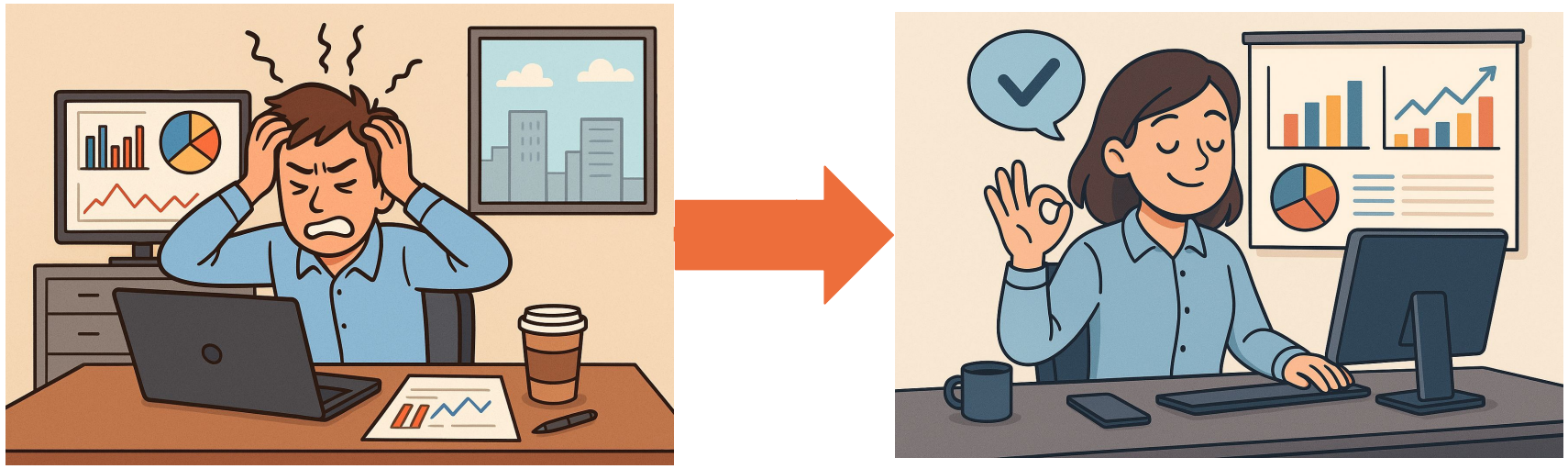
Replace "mziemann" with your username. Replace "docker_for_r_tutorial" with the name of your project

```
# Build an image based on a Dockerfile
docker build -t mziemann/docker_for_r_tutorial .

# get Rstudio server IDE through the web browser (http://localhost:8787)
docker run \
     -e PASSWORD=bioc \
     -p 8787:8787 \
     mziemann/docker_for_r_tutorial:latest

# or run in terminal mode - may be best option for servers
docker run -it \
    -e DISPLAY=localhost:10.0  \
    -v $HOME/.Xauthority:/root/.Xauthority:rw  \
    --network host  \
    mziemann/docker_for_r_tutorial bash

# inside the container, pull the latest project codes
git pull

# open R and start working or run a script
Rscript -e "rmarkdown::render('workflow.Rmd')"

# run this on the host machine to check output
docker cp $(docker ps -aql):/docker_for_r_tutorial docker_data
firefox docker_data/workflow.html
```

```
Other useful docker commands

# show the images available
docker images

# see which containers are running (have run)
docker ps

# delete a container
docker rm <container ID>

# delete an image
docker rmi <image ID>

# clean up closed containers and cached data
docker system prune -f
```

Sharing and archiving a Docker image

```
# Optional - share image on Docker Hub
docker login
docker push mziemann/docker_for_r_tutorial

# Save an image for archiving on institutional RDS or Zenodo (~5 mins)
docker save mziemann/docker_for_r_tutorial > docker_for_r_tutorial.tar

# Load an archived image
docker load -i docker_for_r_tutorial.tar
```



Limitations

- ➔ Build process works today, but might not work tomorrow
- ➔ Contents of an image can’t easily be verified
- ➔ While Docker is useful to containerise workflows, it can’t guarantee reproducible builds - this is where Guix and Nix are proposed to be a solution

Apptainer workflow

Ask your sysadmin to install it

Build directly from a saved docker image:

```
apptainer build myimage.sif docker-archive:myimage.tar
```

Or build from an apptainer definition file:

```
apptainer build myimage.sif definition.def
```

Then run a container

```
apptainer run --writable-tmpfs myimage.sif bash
```

mark.ziemann@burnet.edu.au

Conclusion

- Together with other best practices, containers enable complete computational reproducibility of complex workflows
- There is a significant learning curve and work involved, but it makes reproducibility much easier
- Containers should be standard practice in bioinformatics and machine learning
- ARDC is supporting Australian researchers with additional infrastructure

REFERENCES
1. Ioannidis et al, 2009, DOI:10.1038/ng.295
2. Zaringhalam & Federer 2020, DOI:10.5281/zenodo.3818329
3. Samuel & Mietchen 2024, DOI:10.1093/gigascience/giad113
4. Bora and Ziemann 2023, DOI: 10.31219/osf.io/r6kxg
5. Potti et al, 2006, DOI:10.1038/nm1491
6. Baggerly & Coombes 2010, DOI:10.1214/09-AOAS291
7. The Cancer Letter, https://cancerletter.com/the-cancer-letter/20150123_2/
8. Kaiser 2015 DOI:10.1126/science.aad7410.
9. Ziemann et al, 2023, DOI:10.1093/bib/bbad375
10. Valet et al, DOI:10.1038/s41597-022-01720-9
11. Akalin 2018. https://medium.com/data-science/scientific-data-analysis-pipelines-and-reproducibility-75ff9df5b4c5
12. Ziemann et al, 2024, DOI:10.1093/bioadv/vbae159