Containers

Outline

- The problem: dependencies
- Possible solutions
- Containers
    - What they are
    - Why use them
    - Comparison of Docker, Podman, Singularity
- Brief tutorial
- Additional advantages of containers
- Discussion

The problem: dependencies

My First Attempt at Reproducibility (c. 2018)

1. Create an R package for the method
2. Create an R package for data and helper routines
3. Create scripts to run the analyses
4. Put everything on github

Then Time Passes...

- When revising the paper, we updated our code, re-ran the analysis, and... got very different results for a method we had compared against

Then Time Passes...

- When revising the paper, we updated our code, re-ran the analysis, and...
  got very different results for a method we had compared against

- After a great deal of debugging, we discovered a dependency of a dependency of a dependency had changed

Then Time Passes...

- When revising the paper, we updated our code, re-ran the analysis, and... got very different results for a method we had compared against

- After a great deal of debugging, we discovered a dependency of a dependency of a dependency had changed

- Moral: Saving your code is not enough. You need to save the entire computational environment

Example: `lme4`

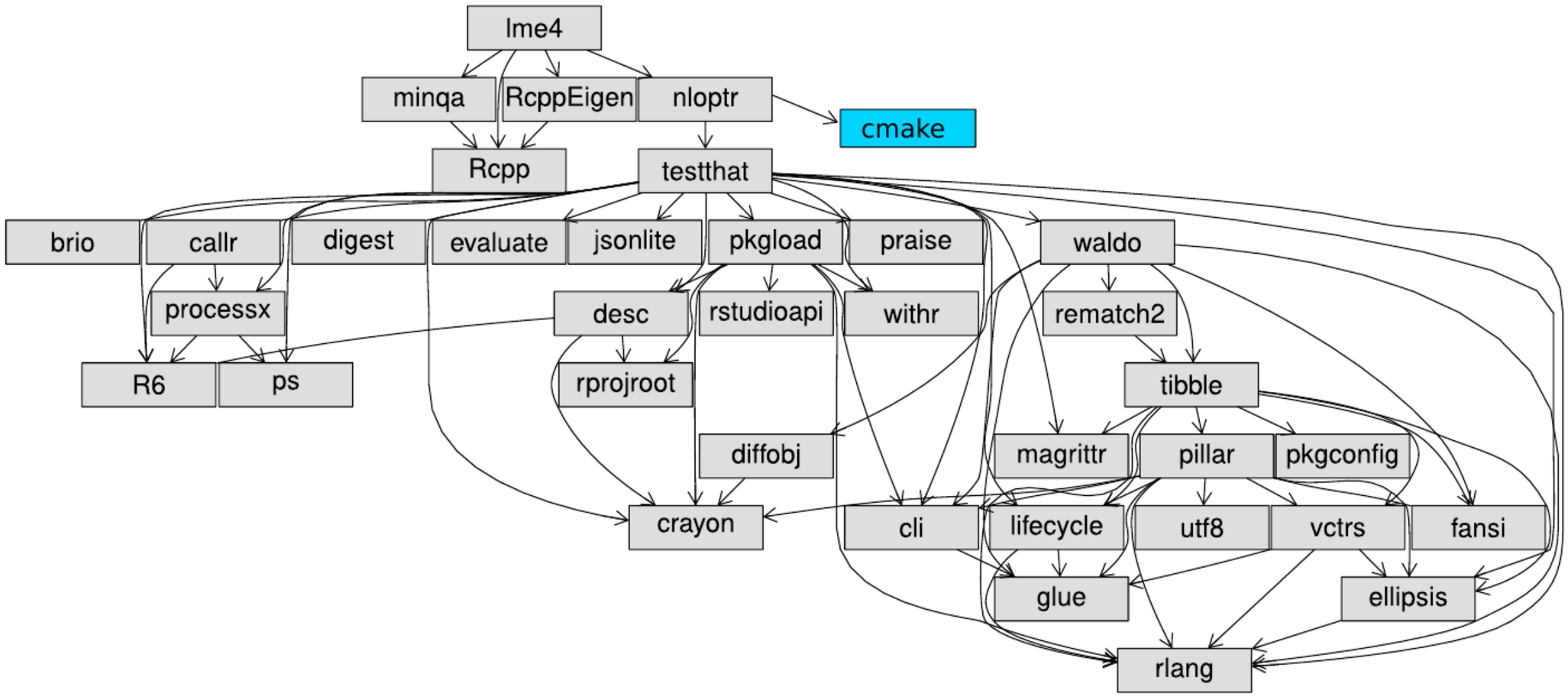


Figure 1: Dependency graph for the R package **lme4**. Grey boxes are R add-on packages. Arrows indicate dependency. The blue box indicates the system-level dependency of the package for Linux OS Ubuntu ver. 20.04.

Possible solutions

- Python virtual environments
- R: `renv`
- Containers (e.g., Docker)
- Others?

Another example (c. 2020)

- Writing a paper with a student that analyzed social media data (Tweets)
- The student created a full analysis pipeline and shared on github
- Fairly simple pipeline, so waited to containerize until final revisions complete

Another example (c. 2020)

- Writing a paper with a student that analyzed social media data (Tweets)
- The student created a full analysis pipeline and shared on github
- Fairly simple pipeline, so waited to containerize until final revisions complete

- Time passes... paper accepted, time to containerize
- One figure in the paper: Randomly selected example tweets
- They changed! (And one was now very offensive)

Another example (c. 2020)

- Writing a paper with a student that analyzed social media data (Tweets)
- The student created a full analysis pipeline and shared on github
- Fairly simple pipeline, so waited to containerize until final revisions complete

- Time passes... paper accepted, time to containerize
- One figure in the paper: Randomly selected example tweets
- They changed! (And one was now very offensive)

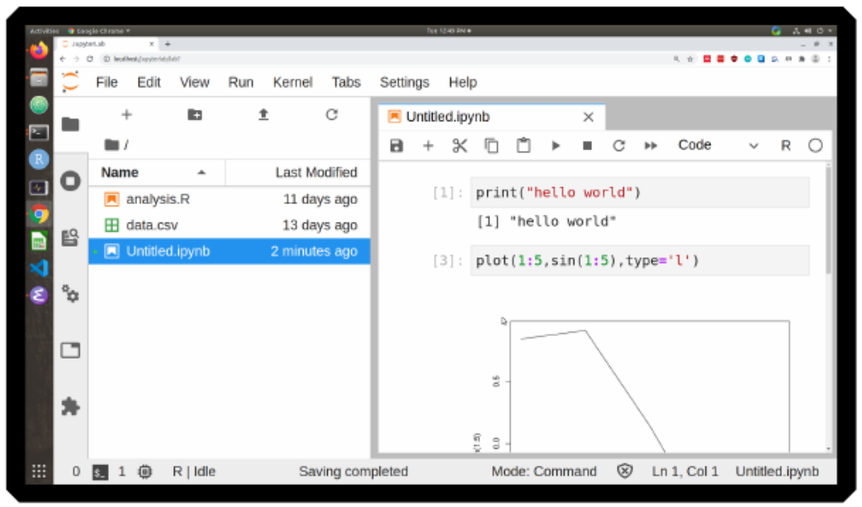
- The method of random number generation for the `sample` command had changed
- https://www.r-bloggers.com/2019/08/remember-the-change-in-the-sample-function-of-r-3-6-0/
- Not even a "dependency"

Containers

What are containers

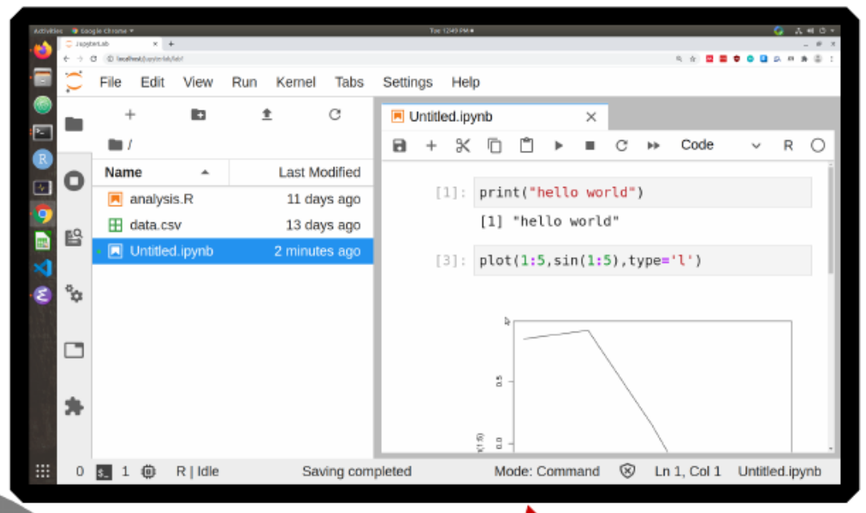
Original Analysis

Original Computing Environment



Third Party

container
(Copy of Computing Environment)



(1) containerization

image

cloud

(2) upload

image

(3) download

image

(4) run container

What are containers

- For a user, feels very similar to virtualization
- But technically, not quite -- the kernel is shared

What are containers

- For a user, feels very similar to virtualization
- But technically, not quite -- the kernel is shared

- Very fast -- almost native speed
- Still need similar hardware (AMD64 vs M1)

Why use them

- Save (nearly) the **entire computing environment**
    - System libraries and utilities
    - Python, R, etc.
    - Packages
- Fast
- Easy to share
    - Single file
    - Cross platform (Linux, Mac, Windows)

Comparison of Docker, Podman, Singularity

Table 1: Comparison of Docker, Singularity, and Podman for containerization of reproducible analyses.

	Docker	**Singularity**	**Podman**
O/S Support	Linux, Mac, Windows	Linux	Linux
Image Type Support	Docker	Docker, Singularity	Docker
Admin. Privileges	Required	Not Required	Not Required
Host/Container Isolation	Yes	No	Yes
Container Mutability	Read/Write	Read Only	Read/Write

Brief tutorial

Brief tutorial

- Many great tutorials online
- Our paper:
  https://jdssv.org/index.php/jdssv/article/view/53

Key ingredients

- Base image
- Dockerfile
- Your existing analysis

Base image

- Minimal Ubuntu
- R (Rocker)
- Jupyter
- Many, many more

Dockerfile

(A) A Simple Dockerfile

```
1 FROM jupyter/datascience-notebook
2 RUN R -e "install.packages('ggplot2', repos =
    'http://cran.us.r-project.org')"
3 COPY --chown=1000 data.csv data.csv
4 COPY --chown=1000 analysis.ipynb analysis.ipynb
5 CMD jupyter lab
```

base image with R and jupyter

R package to install

file on local computer

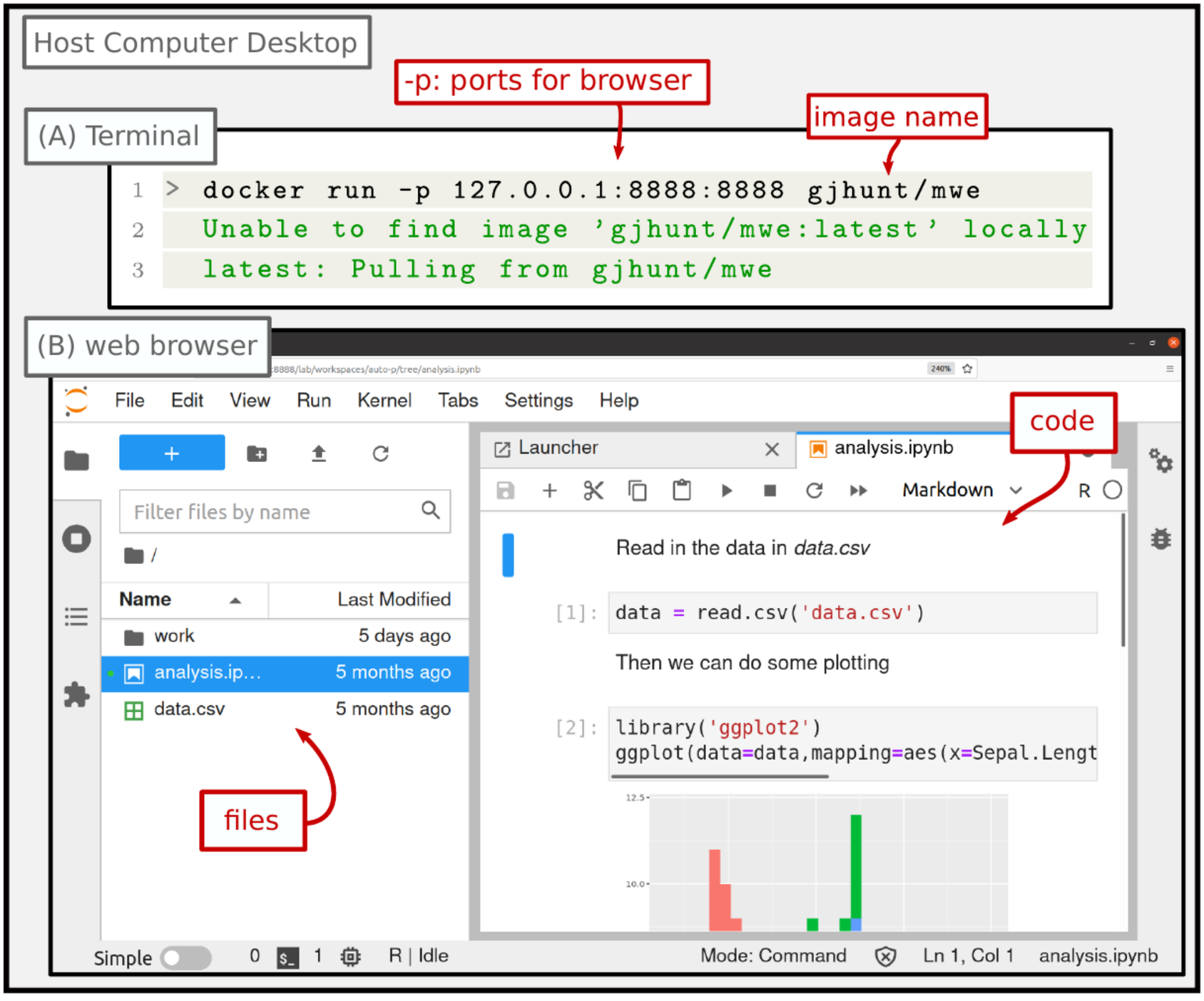
desired name/location in container

(B) Building

```
1 > docker build -t gjhunt/mwe .
2   ...
3  Successfully built bd5ddab32a75
4  Successfully tagged gjhunt/mwe:latest
```

desired name

Running


Host Computer Desktop
-p: ports for browser
image name
(A) Terminal
1 > docker run -p 127.0.0.1:8888:8888 gjhunt/mwe
2 Unable to find image 'gjhunt/mwe:latest' locally
3 latest: Pulling from gjhunt/mwe
(B) web browser
File Edit View Run Kernel Tabs Settings Help
code
Launcher
analysis.ipynb
Markdown
R
Filter files by name
Name
Last Modified
work 5 days ago
analysis.ip… 5 months ago
data.csv 5 months ago
files
Read in the data in data.csv
[1]: data = read.csv('data.csv')
Then we can do some plotting
[2]: library('ggplot2')
ggplot(data=data,mapping=aes(x=Sepal.Lengt
Simple
R | Idle
Mode: Command
Ln 1, Col 1
analysis.ipynb

Another example

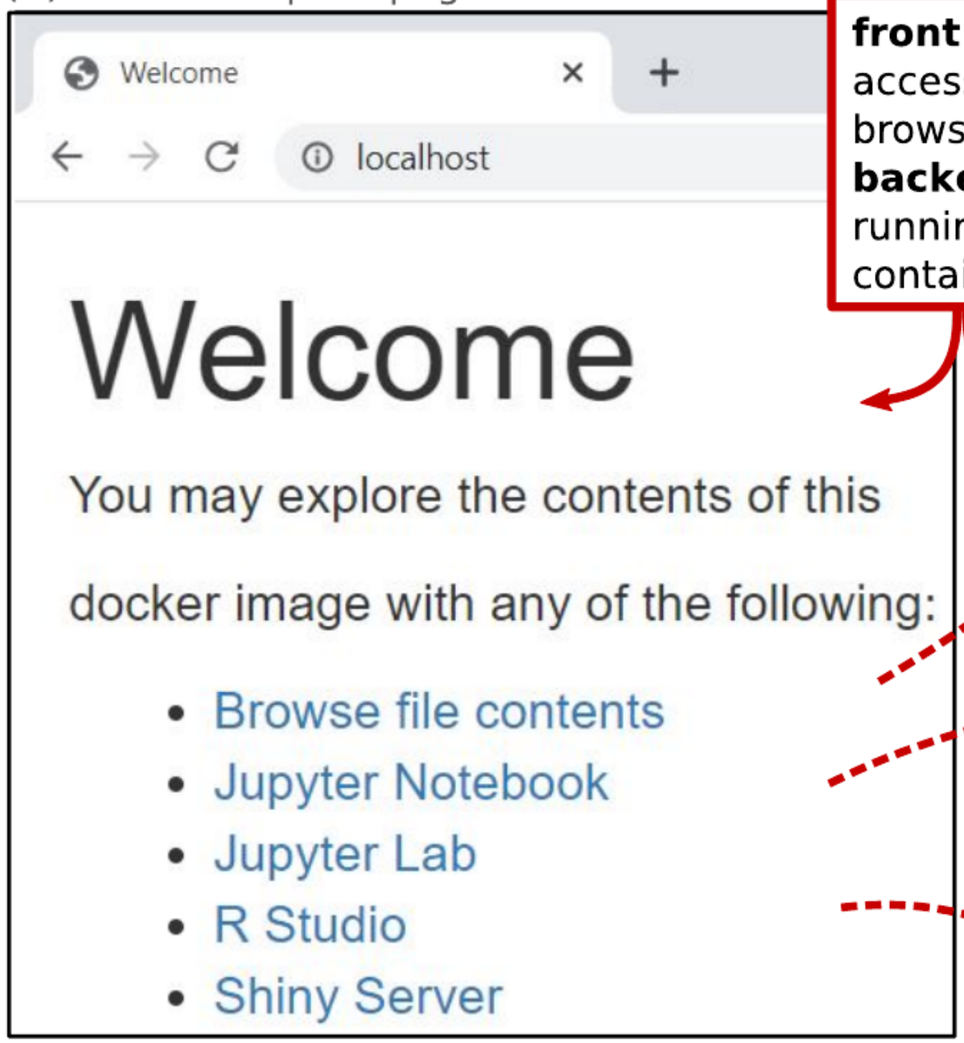
(A) Interactive Dockerfile

```
FROM johanngb/rep-int
WORKDIR /home/rep/
COPY data.csv data.csv
COPY analysis.ipynb analysis.ipynb
RUN jupytext --set-formats ipynb,rmarkdown,R
    analysis.ipynb
RUN jupyter nbconvert --to html analysis.ipynb
```

(B) Running an interactive Container

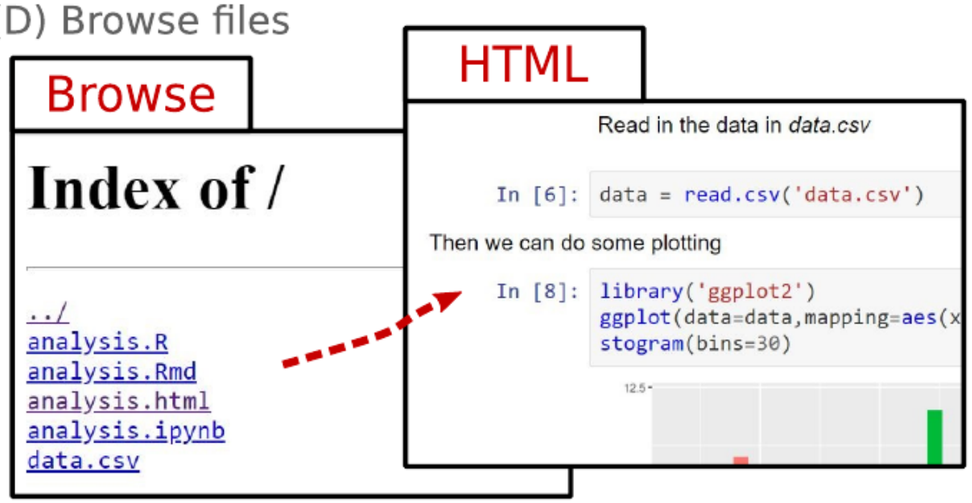
```
> docker run -it --name ex_container -p
    127.0.0.1:80:80 gjhunt/mwe:2
  To get started, please enter the following
    URL into your web browswer:
          http://localhost/
  ...
```

(C) Welcome Splashpage

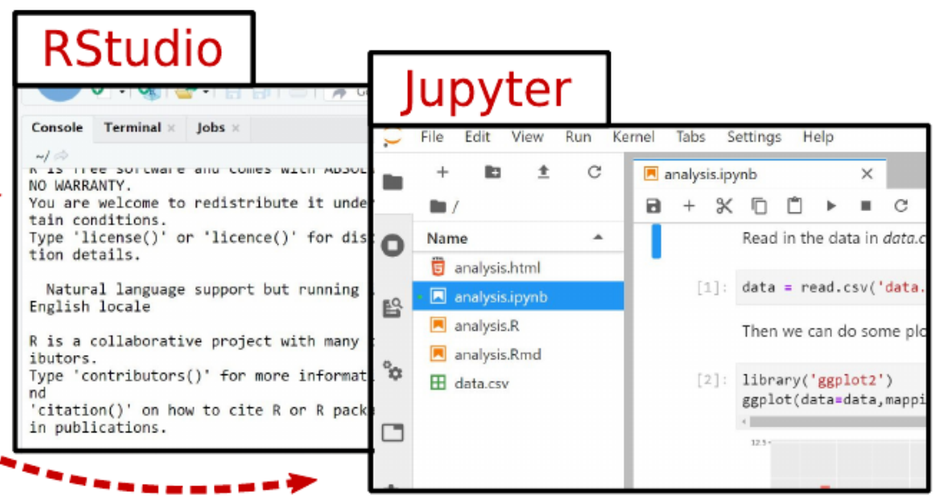


front end: access from browser
backend: running from container

(D) Browse files



(E) Interact with notebooks



Comments and Discussion

Additional advantages of containers

Our goals:

1. Exactly reproducible
2. User friendly
3. Transparent
4. Reusable
5. Archived
6. Version controlled

User friendly

- Code easy to access and inspect, ideally even without downloading
- Should require minimal effort for a user to install and run
- Should cause minimal disruption to a user's resources (e.g., not install unwanted software on their system)
- etc.

User friendly

- Code easy to access and inspect, ideally even without downloading
- Should require minimal effort for a user to install and run
- Should cause minimal disruption to a user's resources (e.g., not install unwanted software on their system)
- etc.

- Minimize the user's security concerns

Discussion