Reproducible research workflows for psychologists

Other topics in reproducible research

Johannes Breuer & Frederik Aust

KU Leuven, 27.-28.04.2022

Other topics in reproducible research

As we said in the introduction, we cannot cover all tools and topics related to reproducible research in this workshop. However, we want to use this session to cover some additional tools as well as other topics in reproducible research:

- Collaborating with others who do not use `R Markdown` and/or `Git`
- R package dependency management
- Preventing code rot
- Publishing reproducible research environments and "one-click reproducibility"

Other approaches to collaboration

There also are `R` packages that you can use for collaborating on `R Markdown` documents with people who do not (want to) use `R Markdown` (and `Git`):

- `trackdown` uses *Google Drive* for this
- `redoc` "is a package to enable a two-way R Markdown-Microsoft Word workflow" (*note*: development currently suspended)

`trackdown`

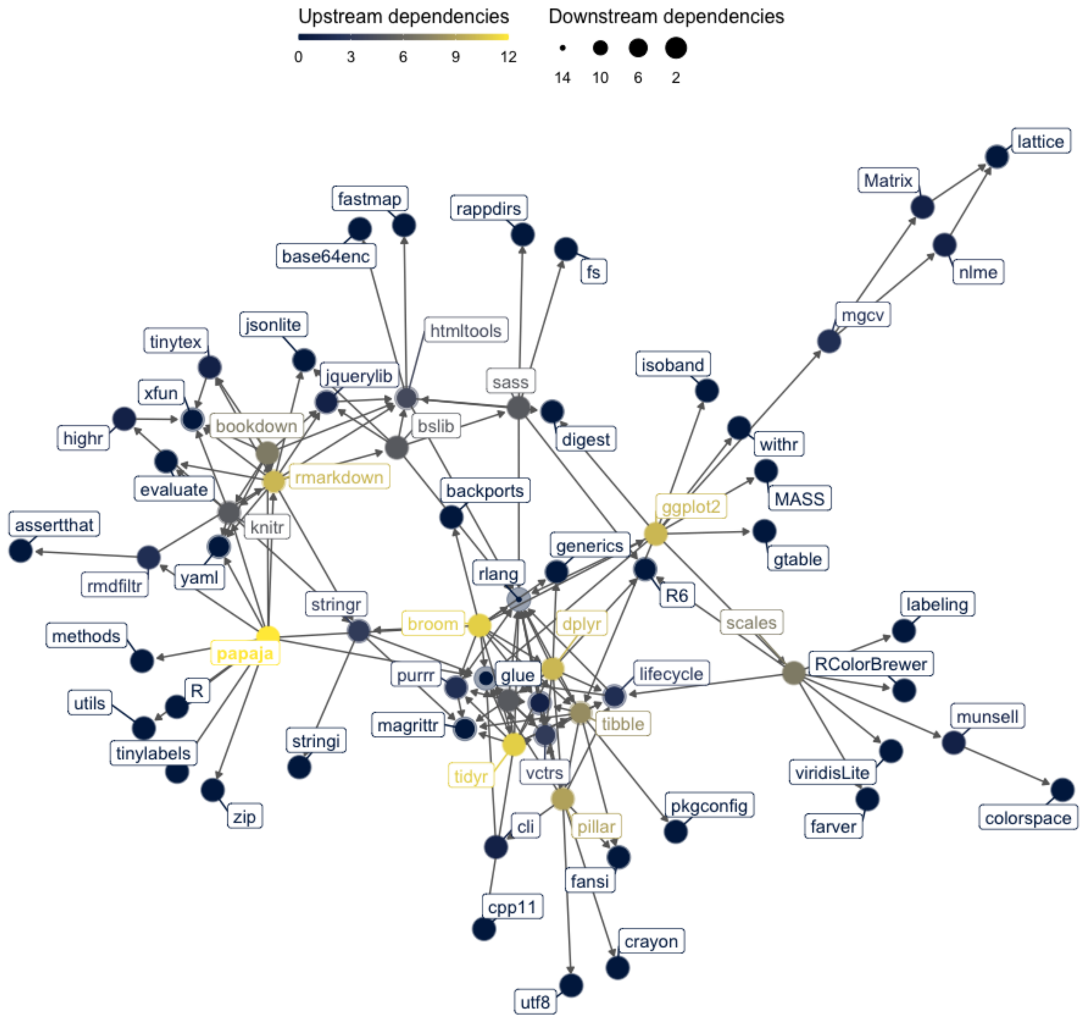
The basic workflow for `trackdown` is that you upload the content of an `.Rmd` file to *Google Drive* where you can collaboratively edit the text parts. You can then download the document again (e.g., to edit the code in the `R Markdown` document in *RStudio*), and also update the file on *Google Drive* after changing the `.Rmd` locally. The `trackdown` documentation provides further details.

Advanced use of `trackdown` with `Git`

To combine `R Markdown` with collaborative text editing via `trackdown` and version control (and to avoid potential issues caused by - possibly unintended - changes to the code parts on *Google Drive*), the author of the `trackdown` package, Claudio Zandonella Callegher, proposes a solution in an issue in the *GitHub* repository for the package.

Essentially, the idea here is to create a `trackdown` branch in the `Git` repository and merge it with the `main` branch which is (mainly) used to edit the code.

Dependency management



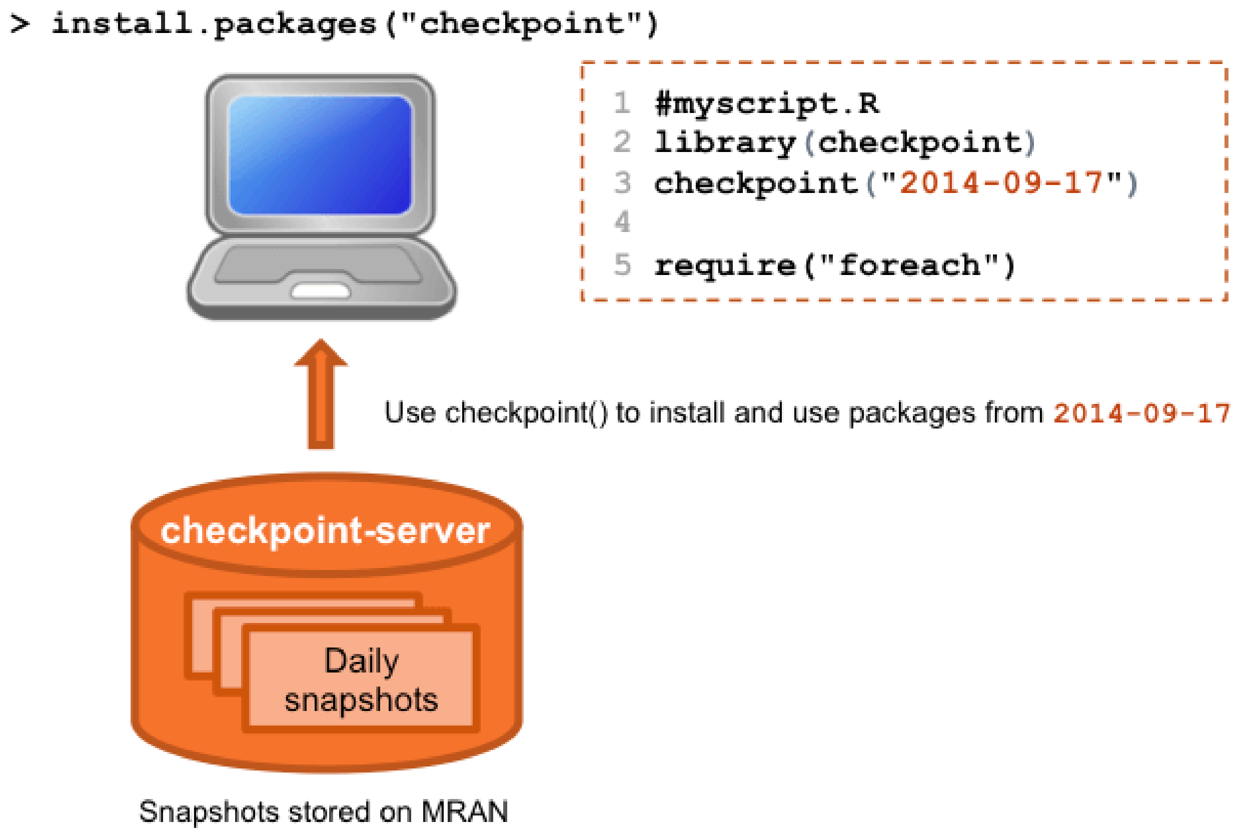
Dependency management

- Our projects may use/require different package versions
- Manually managing dependencies is a nightmare
  - Keeping track of the dependencies and their changes
  - Restore the R environment

Dependency management

1. `checkpoint` by Microsoft
    - Requires a project-based workflow
    - Package database will gradually grow
2. `groundhog`
    - Package database will gradually grow
3. `renv` by RStudio
    - Most flexible and powerful
    - Least straight forward to use
    - No "forensic" applications

Dependency management



```
> install.packages("checkpoint")
```

```
#myscript.R
library(checkpoint)
checkpoint("2014-09-17")

require("foreach")
```

Use checkpoint() to install and use packages from 2014-09-17
checkpoint-server
Daily snapshots
Snapshots stored on MRAN

Dependency management

Dependencies are detected automatically

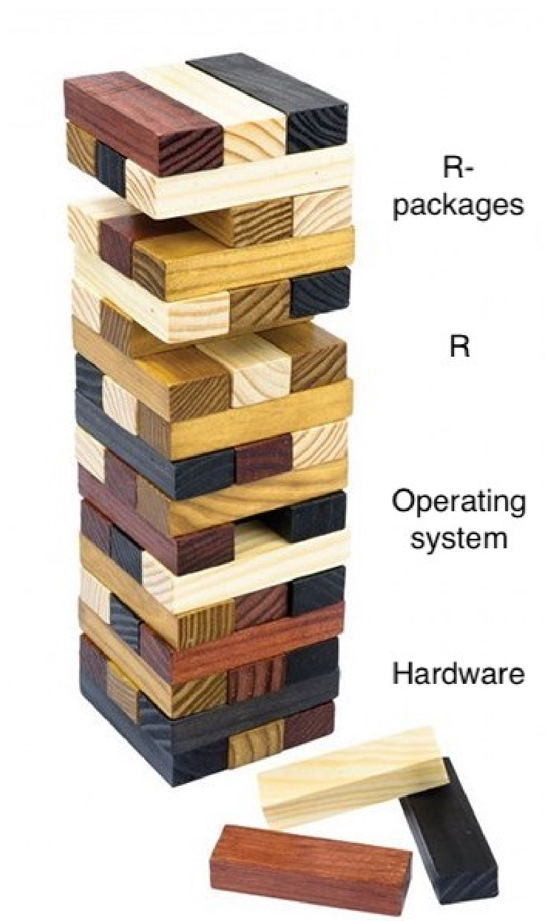
```r
library("checkpoint")
checkpoint("2022-04-27")

library("ggplot2")
```

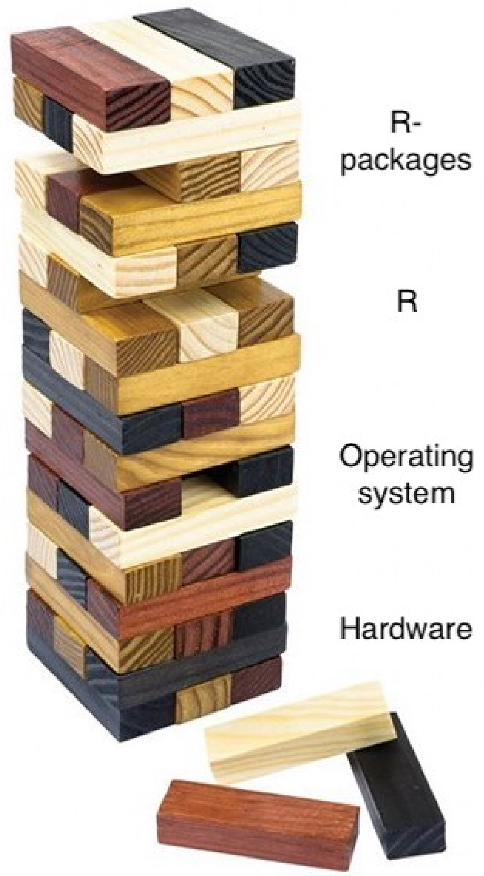
Uses a date-specific directory outside of usual library

```
~/.checkpoint/...
```

Code rot



Code rot



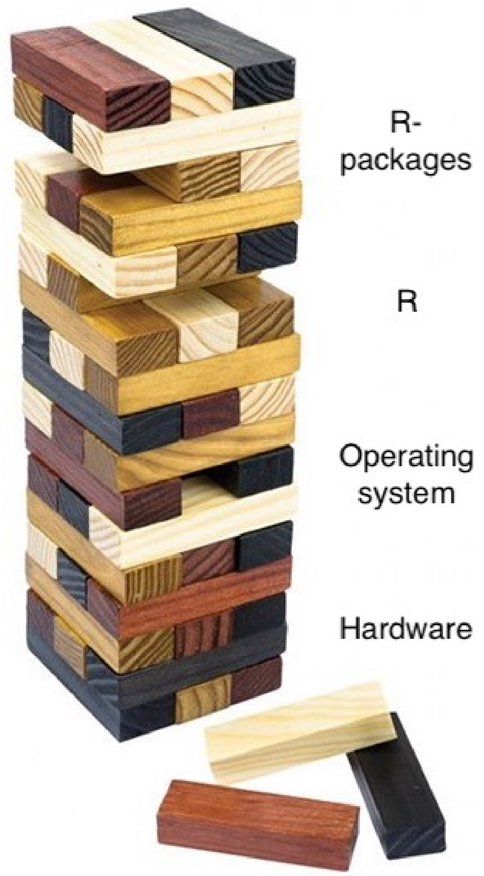
✓

Code rot

- We need to archive the computing environment
- But how?
  - Lock away computer used to run the analysis?
  - Trade-off between robustness and feasibility

```
checkpoint("2020-01-01", r_version = "3.6.2")
```

Code rot

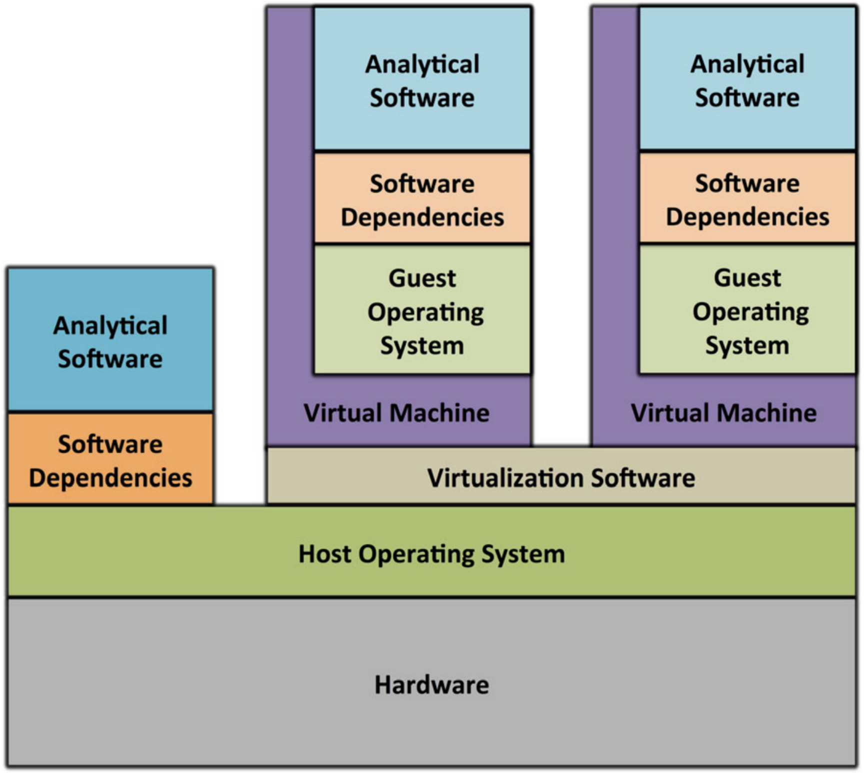


✓

(✓)

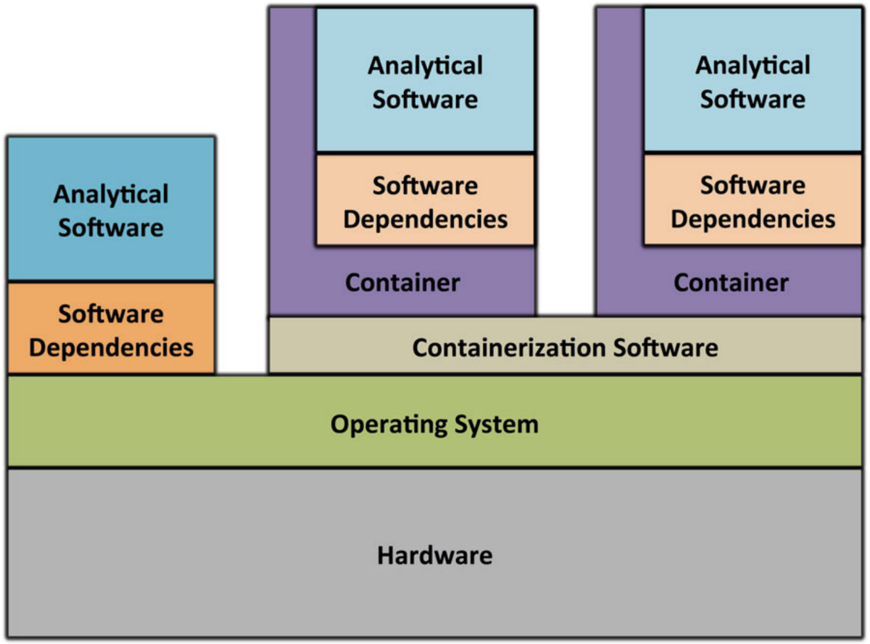
Code rot

Virtual machines



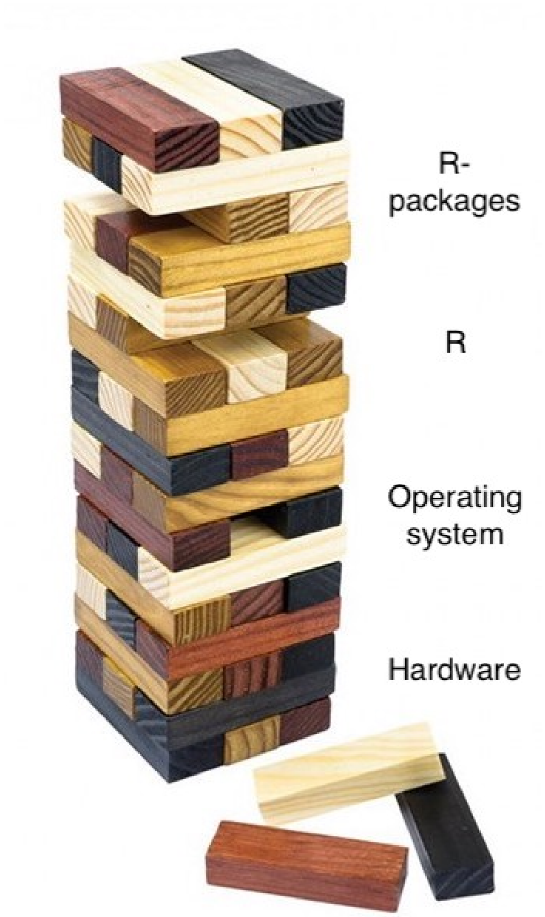
Code rot

Containers



(Piccolo & Frampton, 2016)

Code rot



✓

✓

✓

Docker

Docker: Kind of like Hermann

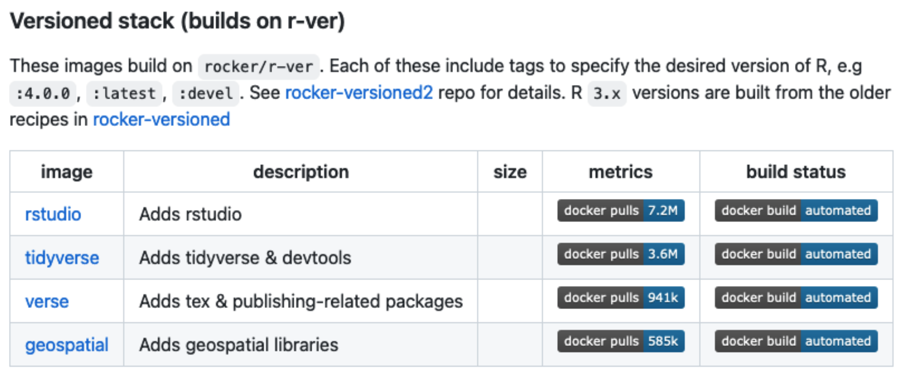
> Herman cake (often called Herman) is a 'friendship cake'. [...] the starter is passed from person to person (like a chain letter) and continues to grow as it contains yeast and lactic acid bacteria. One starter can, in theory, last indefinitely. (Wikipedia)

Docker: Kind of like Hermann

> Herman cake (often called Herman) is a 'friendship cake'. [...] the starter is passed from person to person (like a chain letter) and continues to grow as it contains yeast and lactic acid bacteria. One starter can, in theory, last indefinitely. (Wikipedia)

Docker

Built on a prepackaged base images from DockerHub ("rocker")


Versioned stack (builds on r-ver)

These images build on `rocker/r-ver`. Each of these include tags to specify the desired version of R, e.g `:4.0.0`, `:latest`, `:devel`. See rocker-versioned2 repo for details. R `3.x` versions are built from the older recipes in rocker-versioned

image	description	size	metrics	build status
rstudio	Adds rstudio		docker pulls 7.2M	docker build automated
tidyverse	Adds tidyverse & devtools		docker pulls 3.6M	docker build automated
verse	Adds tex & publishing-related packages		docker pulls 941k	docker build automated
geospatial	Adds geospatial libraries		docker pulls 585k	docker build automated

Docker

Simplified Docker example

```
# Select base image
FROM rocker/rstudio:4.1.2
```

- Latest Debian version
- R 4.1.2
- Latest RStudio version
    - Pandoc
- Git

Docker

Add system-level requirements

```
# System libraries
RUN apt-get update \
  && apt-get install -y --no-install-recommends \
    libgsl0-dev \
    libnlopt-dev \
    libxt6 \
    ssh
```

Docker

Install TeX Live 2021 and required LaTeX packages

```
# TeX Live
ENV CTAN_REPO=http://mirror.ctan.org/systems/texlive/tlnet
RUN /rocker_scripts/install_texlive.sh
ENV PATH=$PATH:/usr/local/texlive/bin/x86_64-linux

RUN tlmgr install \
    apa6 apa7 booktabs caption csquotes \
    ...
```

Docker

Install `papaja` and required R packages

```
# Setup R packages for papaja
RUN install2.r --error \
--skipinstalled \
   tinytex \
   remotes \
   markdown \
   mime

## Latest papaja development version
RUN Rscript -e "remotes::install_github('crsh/papaja')"
```

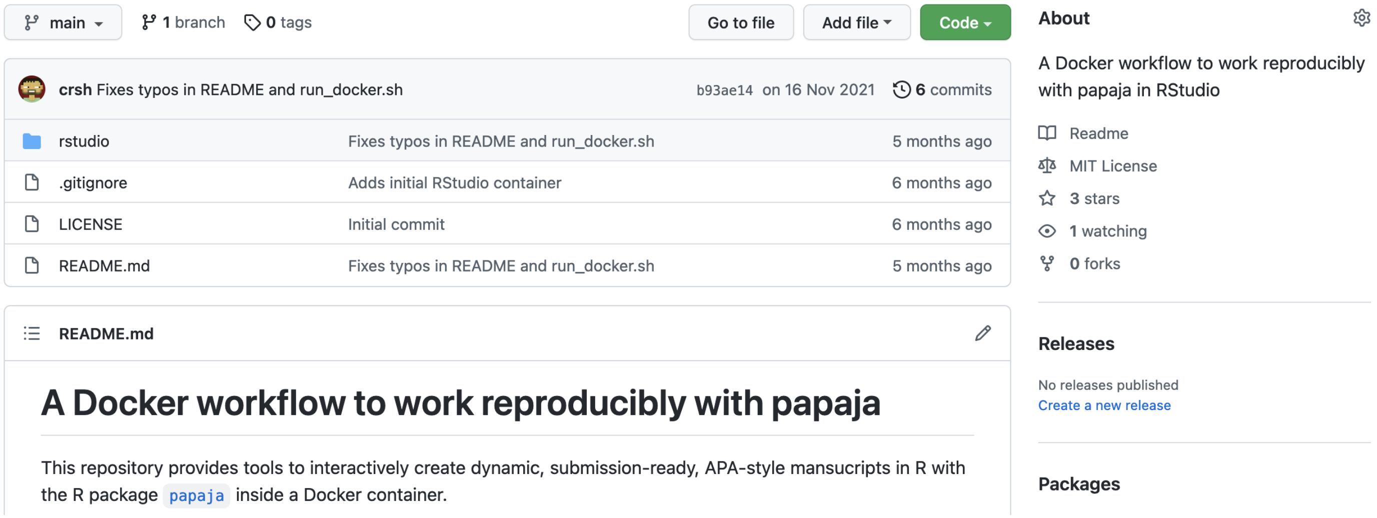
Docker

Our image bundles (among other things)

- Latest Debian version
- R 4.1.2
- Latest RStudio version
    - Pandoc
- Git
- TeX Live 2021
- Latest `papaja` version

Docker

A minimally obtrusive Docker workflow to work reproducibly with papaja



main 1 branch 0 tags Go to file Add file Code

crsh Fixes typos in README and run_docker.sh b93ae14 on 16 Nov 2021 6 commits

rstudio	Fixes typos in README and run_docker.sh	5 months ago
.gitignore	Adds initial RStudio container	6 months ago
LICENSE	Initial commit	6 months ago
README.md	Fixes typos in README and run_docker.sh	5 months ago

README.md

A Docker workflow to work reproducibly with papaja

This repository provides tools to interactively create dynamic, submission-ready, APA-style mansucripts in R with the R package `papaja` inside a Docker container.

About

A Docker workflow to work reproducibly with papaja in RStudio

Readme
MIT License
3 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

A quick demonstration!

Publishing reproducible research environments

If you want to publicly share your fully reproducible research environment and allow others to interact with it without having to install any software on their own machines, you can use services like *Code Ocean* or *RStudio Cloud*. A good free and open source alternative is *BinderHub*.

What is *BinderHub*?

From the *BinderHub GitHub* repository:

BinderHub allows you to `BUILD` and `REGISTER` a `Docker` image from a `Git` repository, then `CONNECT` with *JupyterHub*, allowing you to create a public IP address that allows users to interact with the code and environment within a live *JupyterHub* instance. You can select a specific branch name, commit, or tag to serve.

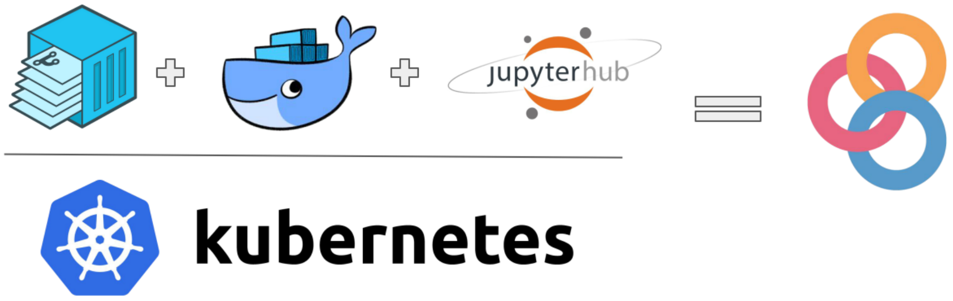
What is *BinderHub*?

From the *BinderHub GitHub* repository:

BinderHub ties together:

- *JupyterHub* to provide a scalable system for authenticating users and spawning single user `Jupyter Notebook` servers, and
- *Repo2Docker* which generates a `Docker` image using a `Git` repository hosted online.

What is *BinderHub*?



How to use *BinderHub*

Using a *BinderHub* deployment like *mybinder.org* or *GESIS Notebooks* you can turn an existing public `Git` repository into a publicly accessible executable environment.

In order for this to work, you just need to add a few *Binder*-specific files to your repo (i.e., "Binderize" it).

A platform that works in similar ways is *PsychNotebook* by the Leibniz Institute for Psychology (ZPID).

Binderizing your `Git` repository

The minimum requirements are the following:

1. Add a `binder` folder to your repo
2. In that folder, create two files: `runtime.txt` & `install.R`

Note: It would also be ok to add those files to the root folder of your project.

Binderizing your `Git` repository

In the `runtime.txt` file, you need to specify a version number and date, indicating which snapshot to use from the *R Studio Package Manager*. Example: `r-4.1-2022-04-22`).

Binderizing your `Git` repository

In the `install.R` file, you need to specify which `R` packages to install as you normally would in an `.R` file (e.g., `install.packages(c("gapminder", "tinytex"))`). *CRAN* packages are installed through the *R Studio Package Manager*.

Binderizing your `Git` repository

There are many more options for changing or extending the executable environment. Two good resources to learn more are the *Turing Way* guide *Zero-to-Binder* or the *Binder* example for `R`.

Deploying your executable environment

Once you have "Binderized" your repository, you can use *mybinder.org* to create the executable environment. You can set a few additional parameters in the process (such as the branch). *NB*: Creating the executable environment can take some time (esp. if you install many and/or large `R` packages).

Note: You can also use the `holepunch` package to Binderize your `R` project hosted on *GitHub*.

Deploying your executable environment

Once the executable environment has been created, anyone who has the link to it can use it. The easiest way to share it and enable "1-click reproducibility" for others is by adding a *Launch Binder* button to the `README` for your associated *GitHub* (or *GitLab*) repository. The `Markdown` code for this will be displayed on the *mybinder* site.

Things to note when using *Binder*

1. The *BinderHub* deployments are hosted on free-to-use servers (sometimes by academic institutions), so you might experience some "hiccups" in deploying and using the executable environments (e.g., if the service is heavily used at the time).
2. Related to that, the amount of RAM is limited for the executable environment.
3. One thing you need to consider is how to share your data when using *Binder*. The *Turing Way* guide *Zero-to-Binder* has some suggestions for that.
4. By default, the link to the executable environment will open an instance of *Jupyter Notebook*, but it is also possible to run *Jupyter Lab*, or *RStudio* (this can be done by adding parameters at the end of the URL, such as `$urlpath=rstudio`).

Test drive

We have prepared a Binderized repo for you that you can test, clone, fork or whatever else you would like to to: https://github.com/jobreu/binder-r-demo

Exercise

Try out and explore one or more of the tools we have just presented:

- `trackdown`
- one of the `R` packages for dependency management (such as `renv`, `checkpoint`, or `groundhog`)
- `Docker`
- `Binder`

Exercise

What did you do?

What were your experiences?

Do you think you are going to use the tool(s)? Why/why not? If yes, for what purposes?

Project setup and templates

In this workshop, we have shown you how to manually set up a reproducible research workflows. However, there are some tools that you can use to automate parts of this process. These can range from very simple to very elaborate solutions. We will show you two examples in the following.

Project setup and templates

`create-project.sh`: small shell script for initializing a basic project folder structure (which can be easily adapted and extended using any text editor)

To run the file, open a shell/command line interface (and navigate to) where the `create-project.sh` file is located. To execute it, you need to provide a valid path for the new project folder that should be created as an argument.

Project setup and templates

```
sh create-project.sh "./my-project"
```

Project setup and templates

Frederik's `R` package for initializing new projects:
https://github.com/crsh/template

Workflow tools

There also several `R` other packages for facilitating the creation and maintenance of reproducible research workflows, such as...

- `WORCS` - *Workflow for Open Reproducible Code in Science*
- `workflowr`
- `starter`
- `rrtools` - Tools for Writing Reproducible Research in R

start your lab also provides an `R` Project Template.

Choosing the right tools 🔨🔧

There are a few things to consider for choosing the right tools:

- Your **habits, knowledge, and preferences** (as well as those of your collaborators)
- Your **goals** and their relative importance: E.g., computational reproducibility, reusability, replicability
- Your **audience**: Future you, collaborators, the academic community, the general public

Shoulders of giants... but sometimes also clay feet

As you may have already experienced, not all tools always play together nicely. Keep in mind, that most tools that we have covered in this workshop are free and open source software (FOSS). Also, tool stacks can have break points and many tools themselves depend on other tools/tool stacks. Hence, things may not always work perfectly.

But don't despair! There usually are solutions (*Stack Overflow* and issues in associated *GitHub* repositories are good places to find them) and the advantage of FOSS is that there usually is an active development community that you can also get involved in.

Showing appreciation 👏

The creation and maintenance of FOSS takes a lot of time and this is rarely recognized as much as it should be. One thing we can do to change this is to at least give credit where credit is due and cite the tools and resources that we use.

Showing appreciation 👏

```r
citation("papaja")
```

```
##
##   Aust, F. & Barth, M. (2020). papaja: Prepare reproducible A
##   articles with R Markdown. R package version 0.1.0.9999. Ret
##   from https://github.com/crsh/papaja
##
## Ein BibTeX-Eintrag für LaTeX-Benutzer ist
##
##   @Manual{,
##     title = {{papaja}: {Prepare} reproducible {APA} journal a
##     author = {Frederik Aust and Marius Barth},
##     year = {2020},
##     note = {R package version 0.1.0.9999},
##     url = {https://github.com/crsh/papaja},
##   }
```

Showing appreciation 👏

When working with `R Markdown`, you can create a `packages.bib` file to cite the packages you have used, either manually, using `papaja::r_refs()`, or the `grateful` package.

Share the ❤️ for reproducible research tools

In addition to properly citing the tools and resources you use, you can make sure that they get the recognition they deserve by talking about them (e.g., on social media) and convincing your collaborators to use them as well.

Looking back

You created a *GitHub*/*GitLab* repository containing materials for a fully reproducible research pipeline! 🎉

If you created a public *GitHub* repository: Head over to http://starlogs.net/ and paste the URL of the repository to recap your heroic journey into the universe of reproducible research! 🌌

Looking forward

We hope that we could get you started or help you with with making your research (more) reproducible. Of course, as always, there is much more to explore and learn. The only way to really get familiar with the tools and workflows is if you use them for your own research.

Keep calm and stay reproducible! 😊

Thank you very much for participating in this workshop! 🙇‍♂️

We hope that you learned something and also had some fun (at least a little bit...)