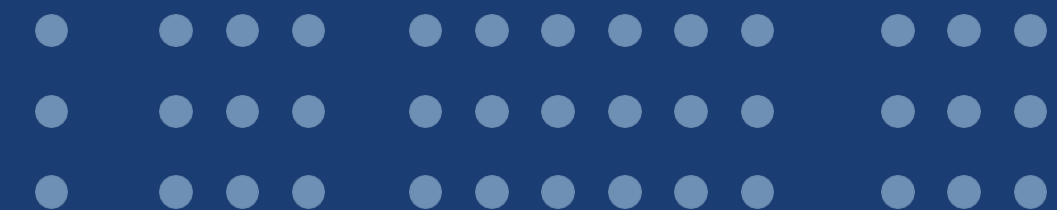
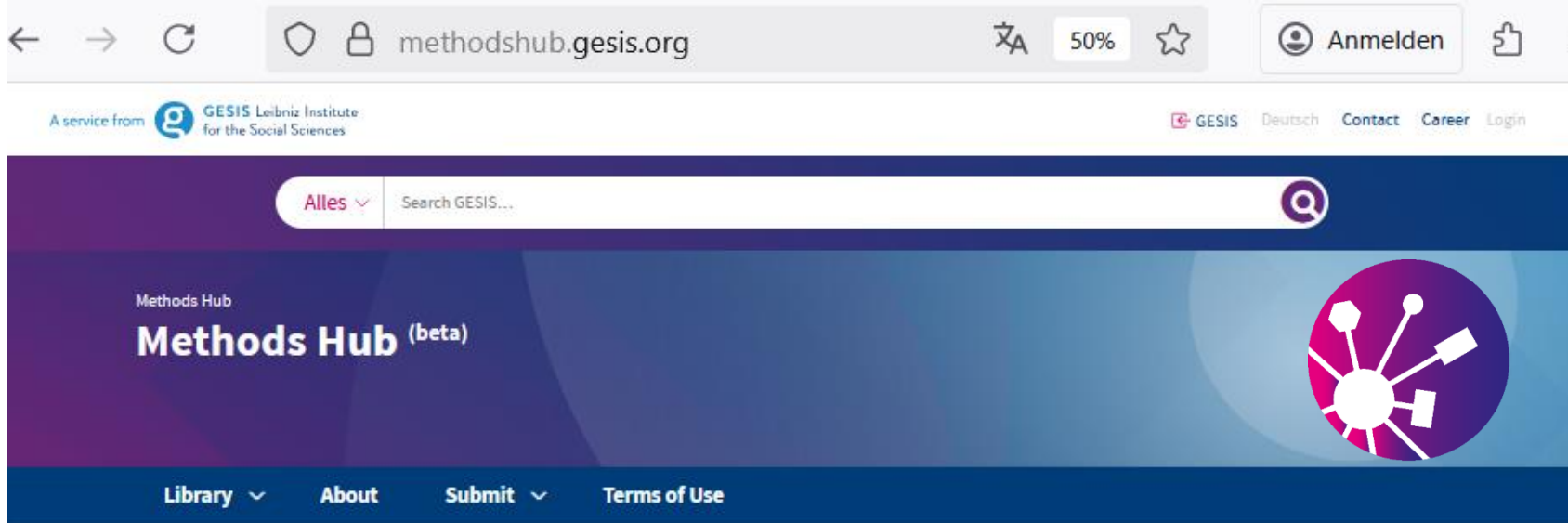


GESIS Methods Hub

Chung-hong Chan, Fakhri Momeni, Taimoor
Khan and Flix Münch





- Data Collection
- Preprocessing
- Data Analysis
- Data Visualization
- Computational Workflows

Methods Hub

Exploring computational methods for your social science research?

The Methods Hub is your starting point. As an open community portal, we bring together practical tools, tutorials, and interactive environments that help you explore and apply computational approaches—from data collection and preprocessing to analysis, results visualization and validation—to solve your social science research problems. Whether you're just getting started or looking to expand your toolkit, you'll find resources designed to support your research every step of the way.

The computational methods gathered from our community are presented from a social science perspective, highlighting their relevance and applicability to the field. Each methods' page connects to various additional resources such as the code and documentation for reproducibility, related datasets for replicability, and tutorials for learning more about using the method.

Curious what it looks like?

Check out the most recent contributions:

Methods Tutorials

- oolong
Create and administrate validation tests for automated content analysis tools
- rtoot
Interact with the mastodon API from R
- Semantic Search Over Social Media Posts
Find relevant social media posts in a collection via semantic search

Or browse our entire [library](#).

Want to try it yourself?

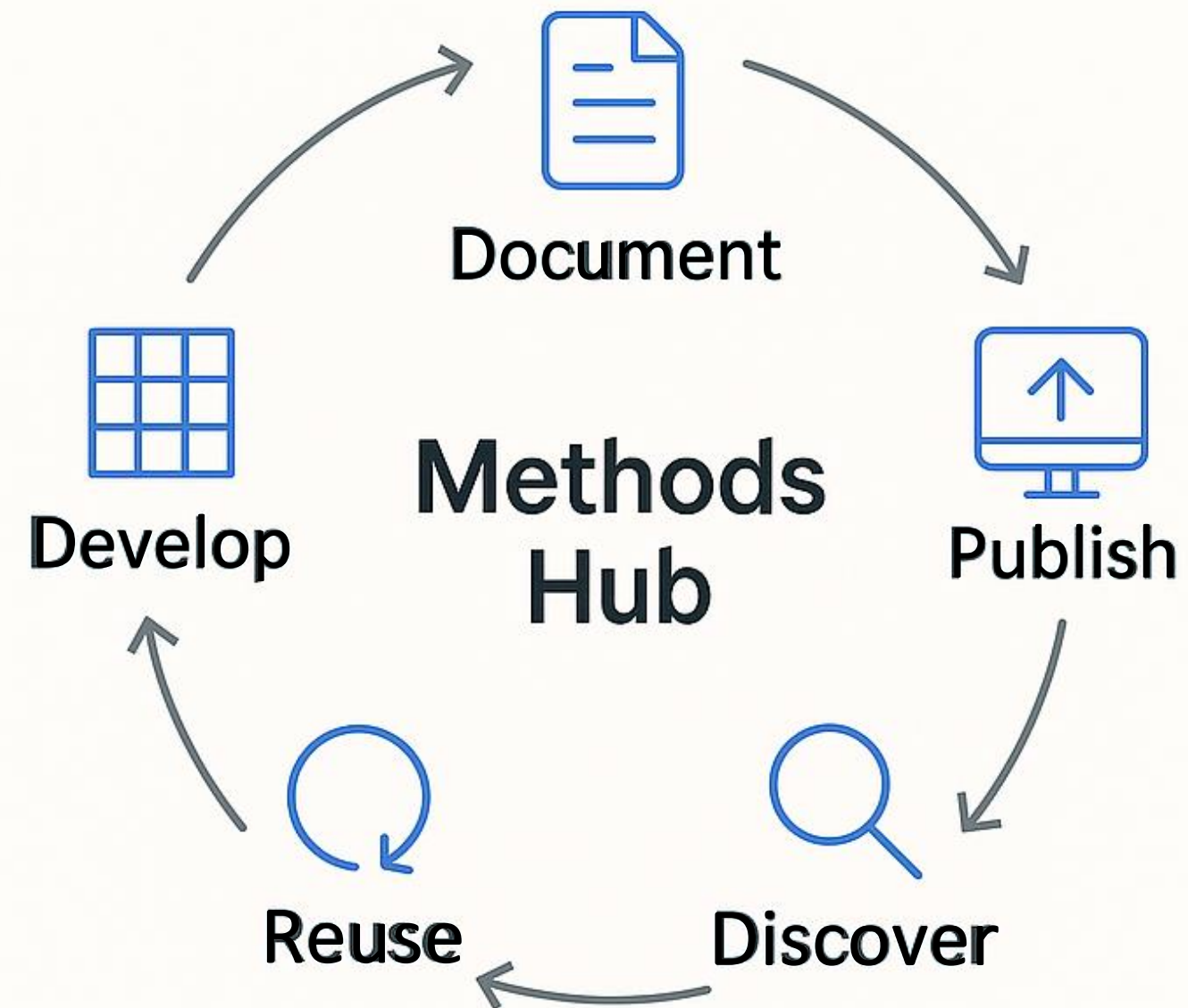
Many of our methods and tutorials support interactive environments, that allow you to explore and run them



Methods Hub

A GESIS initiative for discoverable reusable and reproducible research methods

What is the Methods Hub?

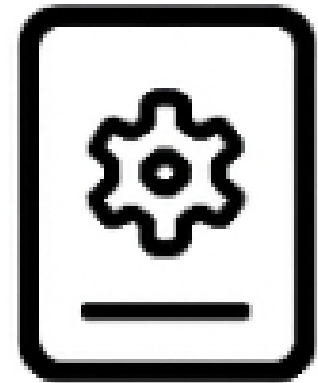


For Method Users



Search and Explore

what it does, who built it, which use cases it fits



Apply in Practice

Get clear guidance on implementation



Cite properly

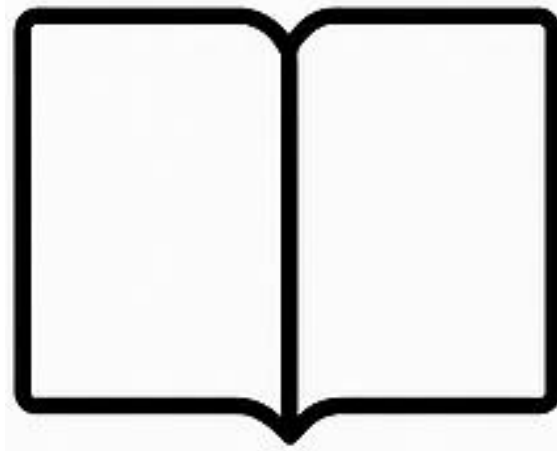
Guidance on citation



Learn from Tutorials

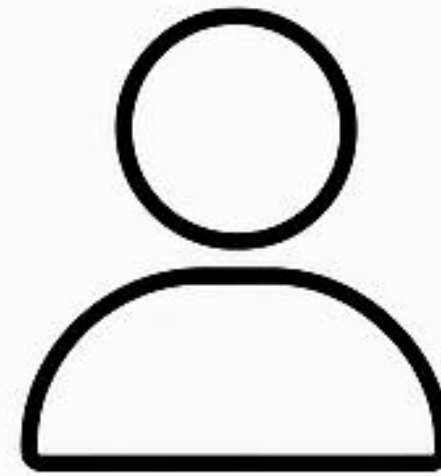
Step-by-step guides to reproduce or adapt the method

Reuse with Confidence



Link to the publications

See the research paper that use or describe the method



Contributor Attribution

Know who developed or contributed to it



Contextual Understanding

Understand how the methods fits in the research ecosystem

For Method Developers: Share, Cite, Reuse



Structured
publishing



Link datasets
& register DOI



Attach
tutorials



Attribution
& licensing



Less support, more
innovation



Trusted through metadata &
documentation



Easier to reuse and
cite

Make your methods discoverable, reproducible, and reusable by design.

Behind the Scenes: Curation & Support



**Review
Workflows**
Maintain quality



**Documentation
Standards**
Keep content
consistent



Onboarding
Support new
contributors



Interoperability
Link to research
infrastructure



**Searchable &
Discoverable**



**Connected to
Research Networks**



**High-Quality
& Reliable**

FAIR by Design



Findable

Each method
richly described
using standar-
dized metadata



Accessible

Methods
accessible
from **GESIS**
Search



Interoperable

Methods
interoperable
with other
research tools

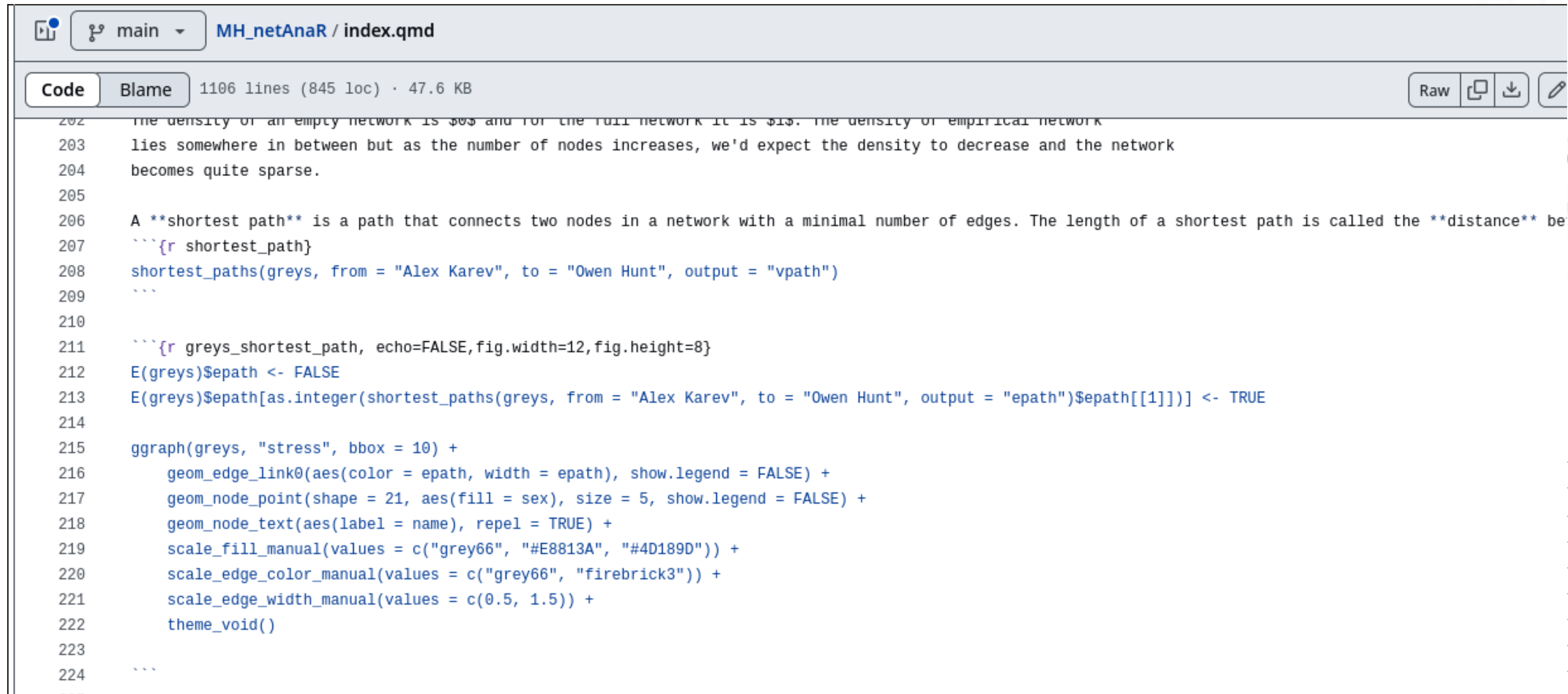


Reusable

Clear
documentation
and licensing
ensure method
reusability

methodshub.gesis.org

(debut: End of June)



```
202 the density of an empty network is 0$ and for the full network it is 1$. The density of empirical network
203 lies somewhere in between but as the number of nodes increases, we'd expect the density to decrease and the network
204 becomes quite sparse.
205
206 A shortest path is a path that connects two nodes in a network with a minimal number of edges. The length of a shortest path is called the distance be
207 ```{r shortest_path}
208 shortest_paths(greys, from = "Alex Karev", to = "Owen Hunt", output = "vpath")
209 ```
210
211 ```{r greys_shortest_path, echo=FALSE, fig.width=12, fig.height=8}
212 E(greys)$sepath <- FALSE
213 E(greys)$sepath[as.integer(shortest_paths(greys, from = "Alex Karev", to = "Owen Hunt", output = "epath")$sepath[[1]])] <- TRUE
214
215 ggraph(greys, "stress", bbox = 10) +
216   geom_edge_link0(aes(color = epath, width = epath), show.legend = FALSE) +
217   geom_node_point(shape = 21, aes(fill = sex), size = 5, show.legend = FALSE) +
218   geom_node_text(aes(label = name), repel = TRUE) +
219   scale_fill_manual(values = c("grey66", "#E8813A", "#4D189D")) +
220   scale_edge_color_manual(values = c("grey66", "firebrick3")) +
221   scale_edge_width_manual(values = c(0.5, 1.5)) +
222   theme_void()
223
224 ```
```

methodshub.gesis.org

(debut: End of June)

Files

main

Go to file

.gitignore

Netreader1.png

Netreader2.png

README.md

_quarto.yml

apt.txt

closeness.gif

crannet.RDS

featured.png

index.ipynb

index.qmd

install.R

postBuild

runtime.txt

signed_triads.png

triad_census.jpg

MH_netAnaR / install.R

 schochastics added emo to install.R

Code Blame 17 lines (17 loc) · 515 Bytes

```
1 install.packages("igraph")
2 install.packages("netrankr")
3 install.packages("remotes")
4 remotes::install_github("schochastics/networkdata")
5 install.packages("ggraph")
6 install.packages("graphlayouts")
7 install.packages("ggrepel")
8 install.packages("centiserve")
9 install.packages("knitr")
10 install.packages("ggforce")
11 install.packages("stringr")
12 install.packages("Matrix")
13 install.packages("signnet")
14 install.packages("emo")
15 install.packages("rmarkdown")
16 install.packages("concaveman")
17 remotes::install_github("hadley/emo")
```

MH_netAnaR / runtime.txt

 schochastics added utils

Code Blame 1 lines (1 loc) · 18 Bytes

```
1 r-4.4.1-2024-06-14
```

Section Navigation

Configuration Files

Example repositories

Home > Reference > Configuration Files

Configuration Files

`repo2docker` looks for configuration files in the repository being built to determine how to build it. In general, `repo2docker` uses the same configuration files as other software installation tools, rather than creating new custom configuration files.

A number of `repo2docker` configuration files can be combined to compose more complex setups.

The [binder examples](#) organization on GitHub contains a list of sample repositories for common configurations that `repo2docker` can build with various configuration files such as Python and R installation in a repository.

A list of supported configuration files (roughly in the order of build priority) can be found on this page (and to the right).

environment.yml - Install a conda environment

`environment.yml` is the standard configuration file used by [conda](#) that lets you install any kind of package, including Python, R, and C/C++ packages. `repo2docker` does not use your `environment.yml` to create and activate a new conda environment. Rather, it updates a base conda environment [defined here](#) with the packages listed in your `environment.yml`. This means that the environment will always have the same default name, not the name specified in your `environment.yml`.

Note

You can install files from pip in your `environment.yml` as well. For example, see the [binder-examples environment.yml](#) file.

On this page

- `environment.yml` - Install a conda environment
- `Pipfile` and/or `Pipfile.lock` - Install a Python environment
- `requirements.txt` - Install a Python environment
- `setup.py` - Install Python packages
- `Project.toml` - Install a Julia environment
- `REQUIRE` - Install a Julia environment (legacy)
- `install.R` - Install an R/RStudio environment
- `apt.txt` - Install packages with apt-get
- `DESCRIPTION` - Install an R package
- `postBuild` - Run code after installing the environment
- `start` - Run code before the user sessions starts
- `runtime.txt` - Specifying runtimes
- `default.nix` - the nix package manager
- `Dockerfile` - Advanced environments

https://mybinder.readthedocs.io/en/latest/using/config_files.html

methodshub.gesis.org

(debut: End of June)



Turn a Git repo into a collection of interactive notebooks

Have a repository full of Jupyter notebooks? With Binder, open those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere.

New to Binder? Get started with a [Zero-to-Binder tutorial](#) in Julia, Python, or R.

Build and launch a repository

GitHub repository name or URL

GitHub ▾ example: yuvipanda/requirements or https://github.com/yuvipanda/requirements

Git ref (branch, tag, or commit)

HEAD

File to open (in JupyterLab)

eg. index.ipynb

File ▾

launch

Fill in the fields to see a URL for sharing your Binder.

Badges for your README

[show](#)

Build Logs

[show](#) [view raw](#)

<https://mybinder.org/>