

## Table of contents

1 Getting Started .....	1
2 Repository Structure .....	1
2.1 General interest files .....	1
2.2 Files for reproducibility .....	2
3 Reproducing the analysis .....	4
3.1.1 Overview .....	4
3.1.2 Option 1: Reproducing the figures in a web-browser using Binder .....	4
3.1.3 Option 2: Reproducing with locally installed R .....	4
3.1.4 Option 3: Docker or Apptainer container .....	5
3.1.4.1 To run in Docker container .....	5
3.2 To run in Apptainer container: .....	7
4 Pipelines: original project steps .....	8
4.1 1. Download and cache OSM data for Valencia .....	8
4.2 2. Run the speed test .....	8
4.3 3. Prepare the data for figures .....	9
4.4 4. Run the analysis and reproduce all figures .....	10
4.5 Generate pipeline visualisations .....	11
5 Building the computational environment from scratch .....	12
5.1 Building the Docker container .....	12
5.2 Building the Apptainer container .....	12

This [repository](#) contains the supplementary materials, data processing pipelines, computational environment preserved in Docker and Apptainer container images, and scripts used for ‘**spanishoddata: A package for accessing and working with large Spanish mobility datasets**’ article. The snapshot of this repository is at Zenodo: <https://doi.org/10.5281/zenodo.15207374>.

To open this repository online using interactive RStudio environment and run the scripts, just click the link >> <https://mybinder.org/v2/gh/e-kotov/spanishoddata-paper-supplement/HEAD?urlpath=rstudio>, then read the [relevant section below](#).

## 1 Getting Started

If you are only looking for the article plots and supplement data (such as articles analysis data, search queries and results), kindly see the [“Repository Structure”](#) -> [“General interest files”](#) section. If you would like to learn more about the data processing pipelines, please refer to the [“Repository Structure”](#) -> [“Files for reproducibility”](#) section and the following sections that describe each pipeline.

## 2 Repository Structure

### 2.1 General interest files

Top-level folder / file	Second-level item	Description
<b>plots/</b>	main-plots/	Figures for the main article generated by the “ <b>main</b> ” pipeline
	supplement-plots/	Figures for the supplement generated by the “ <b>main</b> ” pipeline
	pipeline-plots/	Figures for the all targets pipelines used in the project.
<b>supplement-data/</b>	articles-using-mitms-mobility-data.json	Raw collected data with attributes extracted from the analysed articles
	articles-using-mitms-mobility-data.csv	Flattened table derived from the <i>.json</i> file
	fig_package_workflow.gv	Graphviz DOT file for the workflow diagram in the supplement
	scopus-search-query.sql	SQL query used to search Scopus
	scopus-search-results.bib	Raw Scopus search results in BibTeX format
	speed_test_summary.csv	Raw speed-test data produced by the “ <b>speed_test</b> ” pipeline (used in supplement)
<b>qmd/</b>	—	Human-readable Quarto/Markdown files to reproduce the case studies

## 2.2 Files for reproducibility

Top-level folder / file	Second-level item	Description
<b>containers/</b>	Dockerfile	Dockerfile to reproduce the computational environment
	update-mermaid-js.R	R script that updates Mermaid.js library that i used for workflow plot generation
<b>R/</b>	Subfolders: cache_osm_data, data_prep, main, speed_test, pipeline_plots	R scripts and supporting functions used across data-processing pipelines in respective targets pipelines

Top-level folder / file	Second-level item	Description
<b>renv/</b>	—	Folder holding the renv activation script and settings for installing the packages required to reproduce all code in the repository. To learn more about renv see <a href="https://rstudio.github.io/renv/">https://rstudio.github.io/renv/</a> .
<b>renv.lock</b>	—	Lockfile pinning exact R-package versions for renv.
<b>.Rprofile</b>	—	R startup file that enables renv.
<b>_targets.yaml</b>	—	Global configuration for the targets pipelines used in the project. To learn more about targets see <a href="https://books.ropensci.org/targets/">https://books.ropensci.org/targets/</a> and <a href="https://docs.ropensci.org/targets/">https://docs.ropensci.org/targets/</a> .
<b>_targets_cache_osm_data.R</b>	—	targets script that caches OpenStreetMap data to make a snapshot. The cache is also stored at <a href="https://doi.org/10.5281/zenodo.15207222">https://doi.org/10.5281/zenodo.15207222</a>
<b>_targets_data_prep.R</b>	—	targets script for data-preparation steps. It downloads all data from original data sources and performs all of the preprocessing of data before passing it on to the “main” pipeline.
<b>_targets_speed_test.R</b>	—	targets script that benchmarks data processing speed between CSV, DuckDB and parquet formats and produces the <b>supplement-data/speed_test_summary.csv</b> file
<b>_targets_main.R</b>	—	targets script that creates all figures for the article.
<b>_targets_pipeline_plots.R</b>	—	targets script that builds the figures for this repository that

Top-level folder / file	Second-level item	Description
		provide an overview of each pipeline.
<b>Dockerfile</b>	—	Dockerfile for <a href="https://mybinder.org/">https://mybinder.org/</a> or <b>docker2repo</b> to run the computational environment for reproducing all targets pipelines.

Docker and Apptainer container images with preserved computational environment are deposited with a copy of this repository at <https://doi.org/10.5281/zenodo.15207374>.

## 3 Reproducing the analysis

### 3.1.1 Overview

This repository contains all files required to reproduce the analysis described in the article. Some analysis steps are inherently unreproducible, such as “cache\_osm\_data”, as they take a snapshot of latest data at the time of running the code and later the retrieved data may change. Some steps require very large data downloads and processing power (“speed\_test” and “data\_prep”). Therefore these steps were isolated in separate pipelines. The repository comes with snapshots from these pipelines that are sufficient to run the “main” pipeline that reproduces all figures for the main article and the supplement, however if you would like to reproduce all other pipelines, be prepared for large data downloads and long processing times.

The “main” step that generates figures can be reproduced in a cloud hosted environment using the pre-build container image.

### 3.1.2 Option 1: Reproducing the figures in a web-browser using Binder

- Just click the link >> <https://mybinder.org/v2/gh/e-kotov/spanishoddata-paper-supplement/HEAD?urlpath=rstudio>.
- Wait for RStudio to load inside the web browser tab.
- Run `targets::tar_destroy` in R console to delete the pipeline snapshots.
- Run `targets::tar_make()` in R console to regenerate all figures.
- You will find the updated figures in the `plots/` folder.

You will likely not be able to run any other pipeline in this environment, as it does not have enough compute power.

### 3.1.3 Option 2: Reproducing with locally installed R

Before proceeding with the steps make sure the computational environment is set up correctly.

- Install R 4.3.2 or later.
- Clone the repository or download it manually and unpack.

```
git clone https://github.com/e-kotov/spanishoddata-paper-supplement.git
```

- Run R or start RStudio/Positron/VSCode in the root directory of the project.
- Install packages with:

```
renv::restore(prompt = FALSE)
```

You can now proceed to the [pipelines section](#) below.

### 3.1.4 Option 3: Docker or Apptainer container

Docker and Apptainer Container images with complete computational environment to reproduce this project can be downloaded from <https://doi.org/10.5281/zenodo.15207374>. You also need to have [Docker](#) or Docker-compatible software installed to run the containers. For Apptainer container you can use [Apptainer](#) or [SingularityCE](#) — this is more common in High Performance Computing (HPC) academic clusters.

#### 3.1.4.1 To run in Docker container

##### 3.1.4.1.1 Using GitHub or Docker Hub container registries

- Clone the repository or download it manually and unpack.

```
git clone https://github.com/e-kotov/spanishoddata-paper-supplement.git
```

- To run the container image hosted in GitHub container registry run the following command while in the root directory of the current repository:

```
docker run --platform linux/amd64 --rm -p 8888:8888 -v $(pwd):/home/rstudio ghcr.io/e-kotov/spanishoddata-paper-supplement:4.4.2
```

If the container is not available anymore in GitHub container registry (e.g. because of change of hosting terms), you can try getting the same container image from Docker Hub:

```
docker run --platform linux/amd64 --rm -p 8888:8888 egorkotovdhub/spanishoddata-paper-supplement:4.4.2
```

If both container registries do not have the requested image, try downloading it using the [Zenodo hosted container](#) section below.

In terminal, look for the link 'http://127.0.0.1:8888/lab?token=SOMEALPHANUMERICSTRING' and open it in your browser. A JupyterLab will open in your browser. From there, click the RStudio button.

When in RStudio, run the following line to disable renv:

```
renv::deactivate()
```

The R session will restart and you will be able to run any pipeline using the packages that are already preinstalled in the container, no package installation from online source is necessary. To run the pipelines, kindly see the [relevant section below](#).

When you are done, press Ctrl+C in the terminal to stop the container.

#### 3.1.4.1.2 Using Zenodo hosted container

- Clone the repository or download it manually and unpack.

```
git clone https://github.com/e-kotov/spanishoddata-paper-supplement.git
```

Download the container image file:

```
curl https://zenodo.org/records/15207375/files/docker-container-image-r442-pkg.tar.gz?download=1 --output docker-container-image-r442-pkg.tar.gz
```

Or download manually from <https://doi.org/10.5281/zenodo.15207374>.

Unpack the file:

```
gunzip -c docker-container-image-r442-pkg.tar.gz > docker-container-image-r442-pkg.tar
```

Load the container image into Docker:

```
docker load -i docker-container-image-r442-pkg.tar
```

The image will be loaded with r442spod name, you can check that it was imported successfully with:

```
docker images
```

You can now run the container with:

```
docker run --platform linux/amd64 --rm -p 8888:8888 -v $(pwd):/home/rstudio r442spod
```

In terminal, look for the link 'http://127.0.0.1:8888/lab?token=SOMEALPHANUMERICSTRING' and open it in your browser. A JupyterLab will open in your browser. From there, click the RStudio button.

When in RStudio, run the following line to disable renv:

```
renv::deactivate()
```

The R session will restart and you will be able to run any pipeline using the packages that are already preinstalled in the container, no package installation from online source is necessary. To run the pipelines, kindly see the [relevant section below](#).

When you are done, press Ctrl+C in the terminal to stop the container.

### 3.2 To run in Apptainer container:

You can download the Apptainer container image from Zenodo <https://doi.org/10.5281/zenodo.15207374>.

The command to download would be:

```
curl https://zenodo.org/records/15207375/files/apptainer-container-image-r442-pkg.sif?download=1 --output docker-container-image-r442-pkg.sif
```

You can then run the container with:

```
apptainer exec --bind "$(pwd)":/home/rstudio apptainer-container-image-r442-pkg.sif bash
```

You may have to use extra commands and options to run Apptainer in High Performance Computing (HPC) academic clusters using SLURM job manager and bind the project folder, as well as forward the console output to a log file so that you can find the connection link. Kindly refer to the documentation provided by your HPC administrator as well as [Apptainer documentation](#). Some HPC admins may offer a web-interface to run Apptainer containers using Jupyter Hub.

In terminal, look for the link 'http://127.0.0.1:8888/lab?token=SOMEALPHANUMERICSTRING' and open it in your browser. A JupyterLab will open in your browser. From there, click the RStudio button.

When in RStudio, run the following line to disable renv:

```
renv::deactivate()
```

The R session will restart and you will be able to run any pipeline using the packages that are already preinstalled in the container, no package installation from online source is necessary. To run the pipelines, kindly see the [relevant section below](#).

When you are done, press Ctrl+C in the terminal to stop the container.

You can also use the same container to run on HPC as a developer container that you can connect to with SSH, as it has a built in SSH server, but you will need to configure it at start up time with a SLURM script and then tunnel/forward ports so that you can connect to it from Positron, VScode or Zed using their remote features.

## 4 Pipelines: original project steps

We provide all the code to reproduce the results in the paper, however we break it down into 4 steps. Step 1 caches some data on the date of the analysis, step 2 runs the speed test comparing CSV, DuckDB and parquet files, step 3 prepares the data for the plots, and step 4 generates the plots. Step 1 is not reproducible on a different date, as the data updates every day. Step 2 requires downloading a very large dataset and takes more then 24 hours to run. Step 3 is also rather resource intensive and downloads about 6 GB of data from the Internet.

All steps assume you have either local R or R running in the the provided containerised environment started in the project root directory.

### 4.1 1. Download and cache OSM data for Valencia

The line below will run the workflow to download the OSM data for Valencia and save it to the data/proc/osm directory.

```
Sys.setenv(TAR_PROJECT = "cache_osm_data"); targets::tar_make()
```

Because this data is changing every day, the downloaded file is then uploaded to <https://doi.org/10.5281/zenodo.15207222>, so that the data snapshot could always be found there.

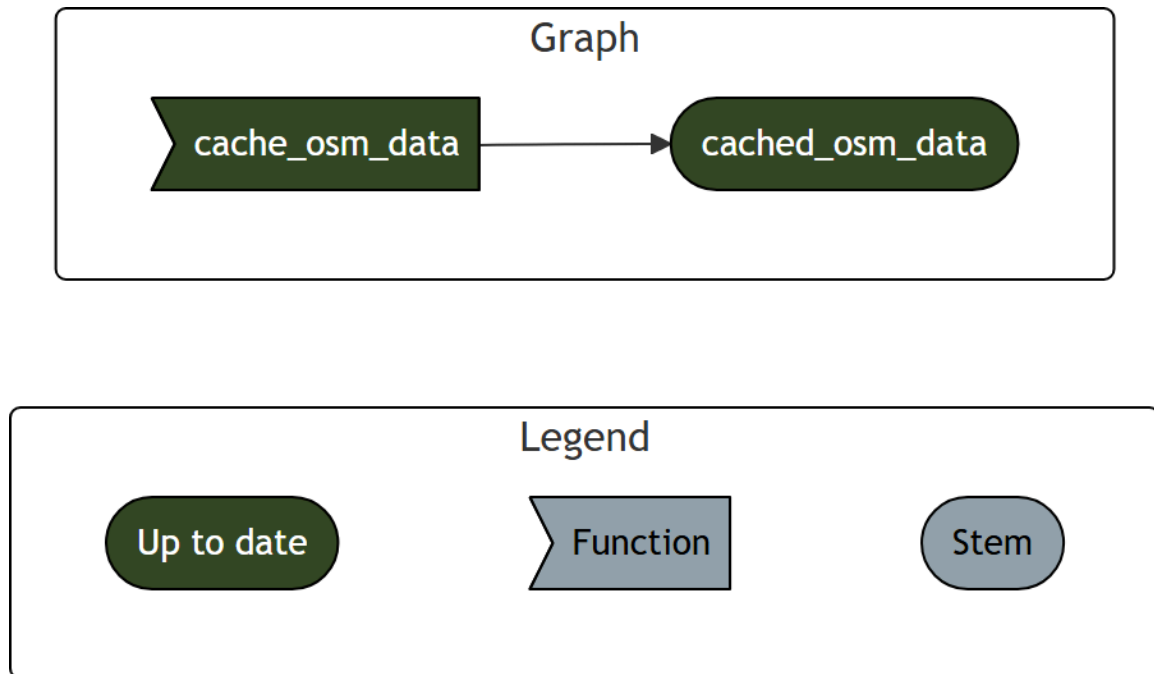


Figure 1: Pipeline visualisation of the cache\_osm\_data workflow

### 4.2 2. Run the speed test

The line below runs the workflow that downloads 18 GB of data using spanishoddata R package, converts it to duckdb and parquet (which will additionally take about 40 GB) and runs a series of tests with different number of threads and memory limits. This may take over a day and the



combinations of threads and memory may need to be adjusted in the `_targets_speed_test.R` file in the root of the repository depending on your available hardware. Currently the maximum hardware specs to test are set up at 128 GB of memory and 64 processor cores.

```
Sys.setenv(TAR_PROJECT = "speed_test"); targets::tar_make()
```

The results are saved into `supplement-data/speed_test_results.csv` and are reused in step 3 below.

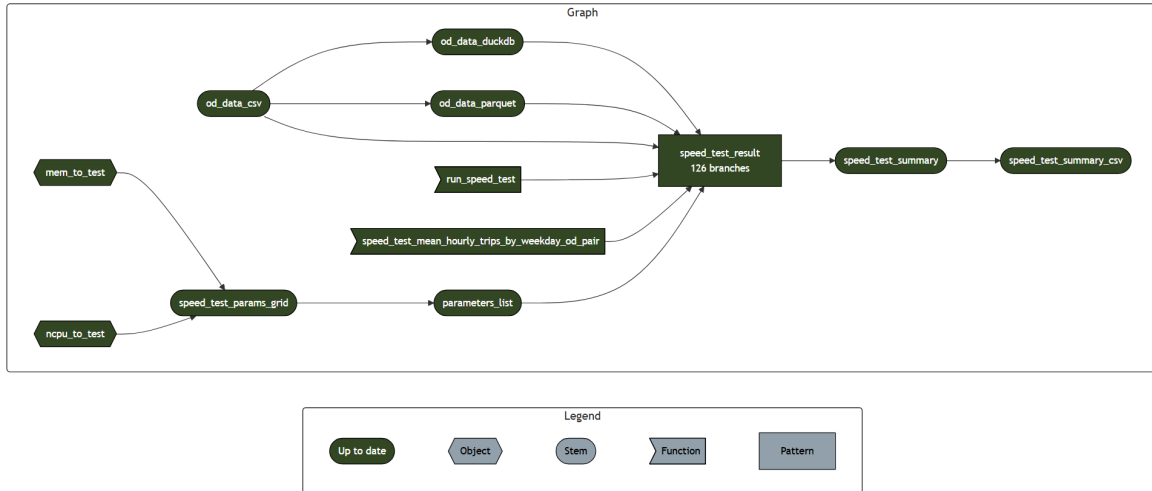


Figure 2: Pipeline visualisation of the `speed_test` workflow

### 4.3 3. Prepare the data for figures

This step prepares all data, except the speed test data, to produce the article and supplement figures.

```
Sys.setenv(TAR_PROJECT = "data_prep"); targets::tar_make()
```

Results are cached in the `_targets/data_prep` directory and are available in the repository, so that the final “main” pipeline can reuse them to generate the figures.

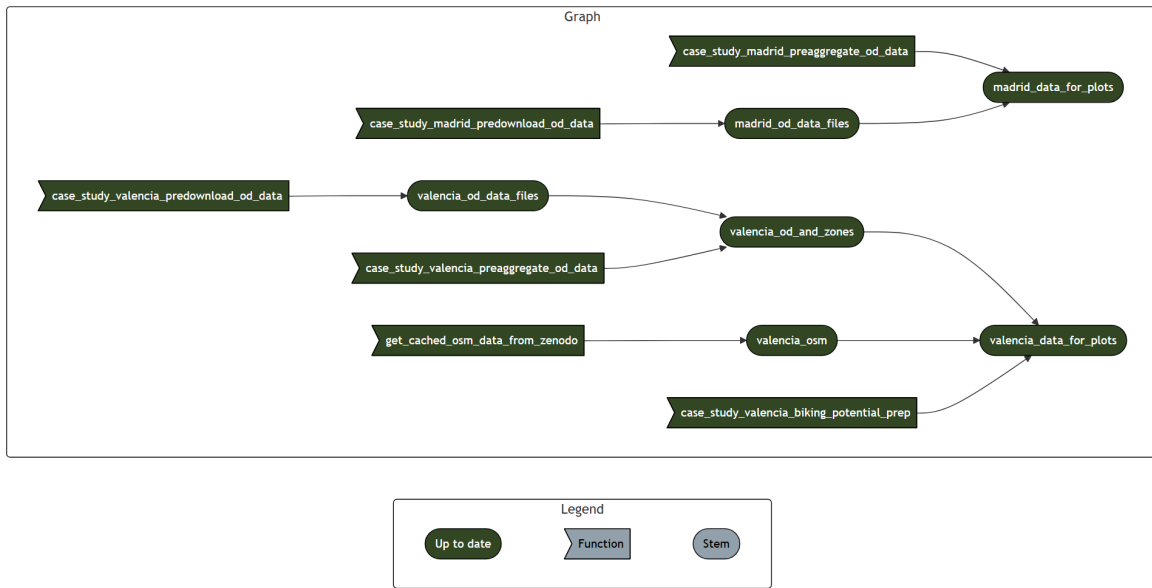


Figure 3: Pipeline visualisation of the data\_prep workflow

#### 4.4 4. Run the analysis and reproduce all figures

To run the analysis and reproduce all the figures run the following line:

```
Sys.setenv(TAR_PROJECT = "main"); targets::tar_make()
```

The figures are generated in the plots directory.

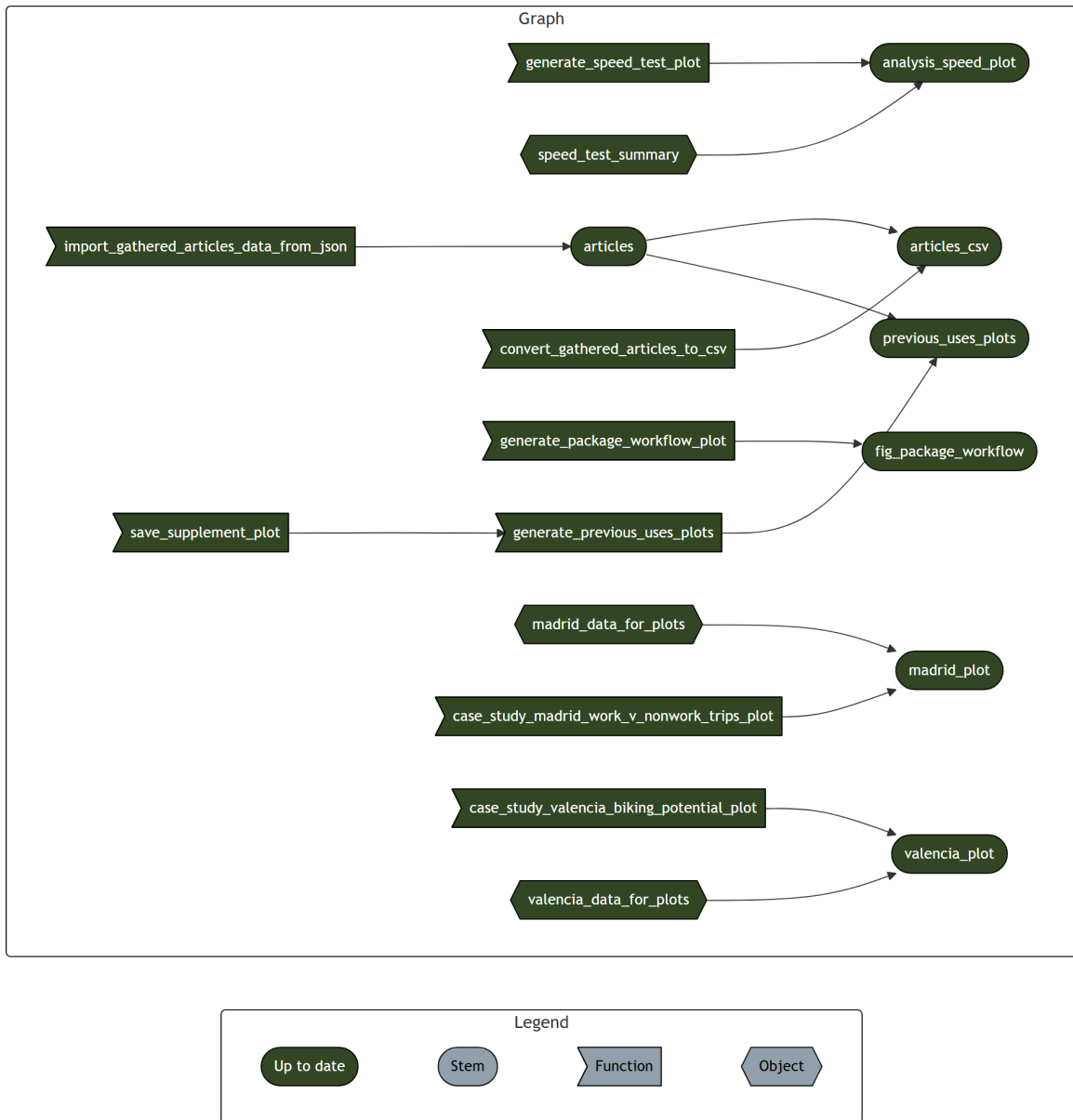


Figure 4: Pipeline visualisation of the main workflow

## 4.5 Generate pipeline visualisations

To generate pipeline visualisations run the following line:

```
Sys.setenv(TAR_PROJECT = "pipeline-plots"); targets::tar_make()
```

The visualisations are generated as code for mermaid diagrams, as html and png files in the media directory. These are the plots you can observe above for each pipeline.

## 5 Building the computational environment from scratch

Here we provide some details how to build the environment from scratch.

### 5.1 Building the Docker container

Make sure Docker is installed and running. Open a terminal application in the project root directory.

```
docker build --platform linux/amd64 -f containers/Dockerfile -t r442spod .
```

To test run the container image locally, run the following command:

```
docker run --platform linux/amd64 --rm -p 8888:8888 -v $(pwd):/home/rstudio r442spod
```

In terminal, look for the link 'http://127.0.0.1:8888/lab?token=SOMEALPHANUMERICSTRING' and open it in your browser. A JupyterLab will open in your browser. From there, click the RStudio button.

When you are done, press Ctrl+C in the terminal to stop the container.

To archive the container image to a file:

```
docker save r442spod -o ~/docker-container-image-r442-pkg.tar
```

### 5.2 Building the Apptainer container

The resulting tar file with the Docker container image can be converted to an Apptainer/Singularity container.

If you are on HPC, make sure to enable the apptainer module. This might be done with:

```
module load apptainer
```

Assuming the Docker image is saved as docker-container-image-r442-pkg.tar and uploaded to ~/ on HPC:

```
apptainer build ~/apptainer-container-image-r442-pkg.sif ~/docker-archive:docker-container-image-r442-pkg.tar
```

To run the Apptainer container in HPC you might have to set up a SLURM job. Kindly, follow the instructions provided by your HPC administrator.