FIXME

Release v0.1.0

Remi Gau

May 08, 2024

CONTENT

CHAPTER

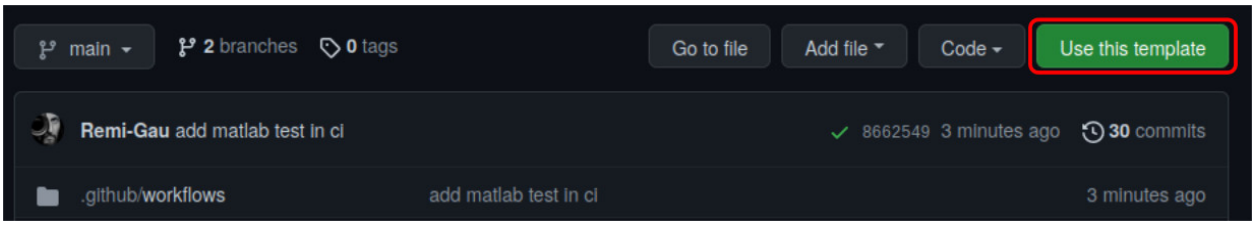
ONE

TEMPLATE REPOSITORY FOR MATLAB / OCTAVE PROJECTS

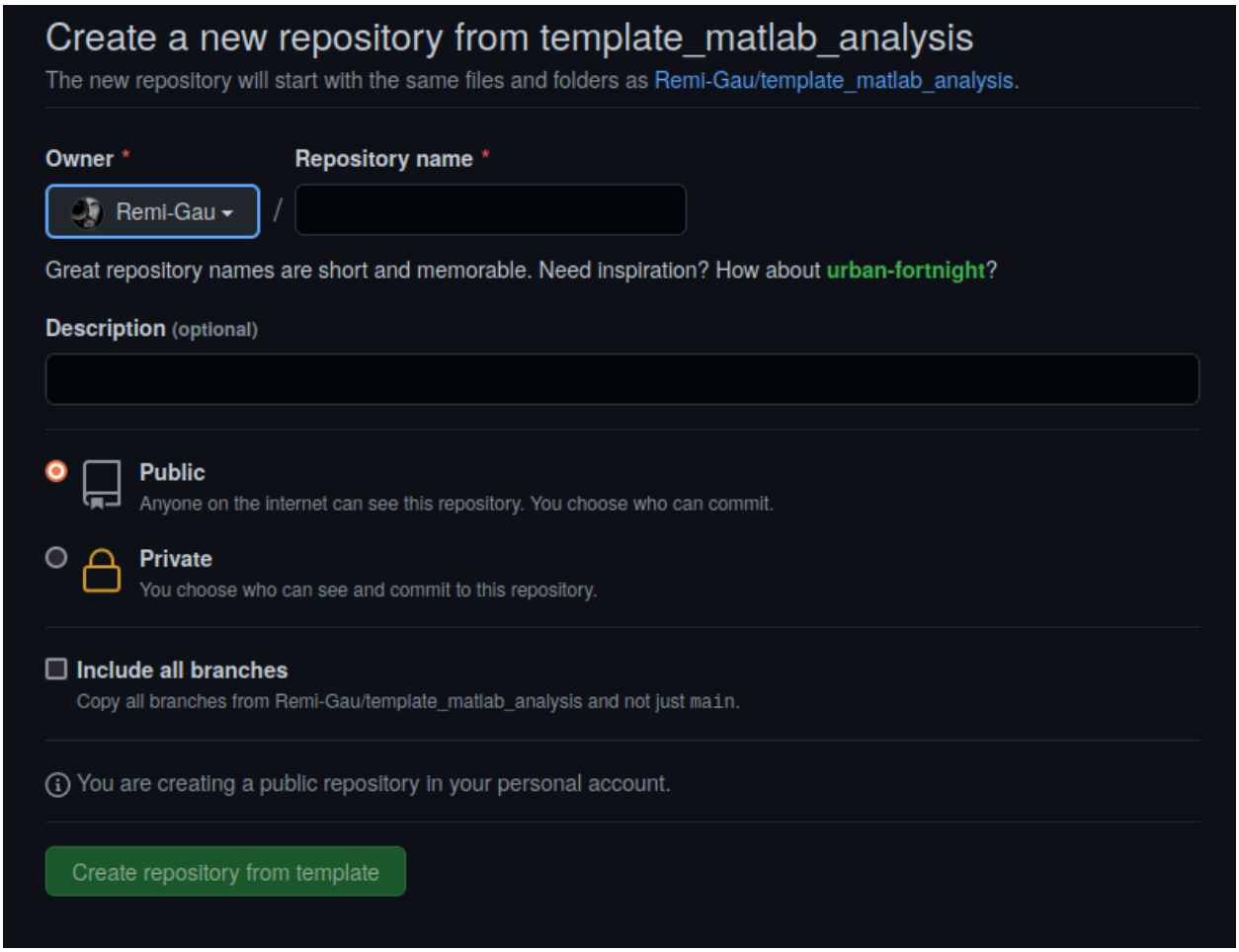
1.1 How to install and use this template

1.1.1 Install with Github

1. Click the green button `Use this template`.



2. Give a name to the repository you want to create. Something short that contains the name of your project: `analysis_my_study`.



3. Decide if you want this new repo to be public or private.
4. Click on `Create repository from template`

   You now have a copy of the template on your Github account. You can then download the code and the pre-set dependencies like this.
5. Click on green `Download` button and copy the `URL_to_your_repo` that is shown there.
6. Open a terminal and type this:

```
git clone URL_to_your_repo
```

1.1.2 Install with cookiecutter

Install Cookiecutter:

```
pip install -U cookiecutter
```

Generate project:

```
cookiecutter https://github.com/Remi-Gau/cookiecutter_matlab_analysis.git
```

1.2 Configuration

Check the CONTRIBUTING.md for more information on setting up this repo.

CHAPTER

TWO

FUNCTION DESCRIPTION

List of functions in the `src` folder.

---

src.**my_fibonacci**(*varargin*)

Returns vector of n iterations of the Fibonacci sequence.

USAGE:

```
results = my_fibonacci(nb_iterations)
```

Parameters

foo (positive integer) – Optional argument. Number of iterations to run. Default = 5;

Returns

- **results**
  (array) (1 x nb_iterations + 2)

Example:

```
results = my_fibonacci(5);
```

2.1 Utilities

src.utils.**is_octave**()

Returns true if the environment is Octave.

USAGE:

```
retval = is_octave()
```

src.utils.**root_dir**()

Returns fullpath the root of the repository.

USAGE:

```
retval = root_dir()
```

Returns

- **root_dir**
  (path)

src.utils.**get_version**()

Reads the version number of the pipeline from the txt file in the root of the repository.

USAGE:

```
version_number = get_version()
```

Returns

- **version_number**
  (string) Use semantic versioning format (like v0.1.0)

CHAPTER

THREE

DOCUMENTATION STYLES

You can choose different ways of documenting the help section of your code.

Those are adapted from their equivalent in python.

Those functions can be found here

—

src.**count_line_google_style_help**(*file*, *line*)

Counts the number of times a line occurs. Case-sensitive. White space padding are ignored.

USAGE:

```
num_instances = count_line_google_style_help(file, line)
```

Arguments:

file (cellstr): content of file to scan

line (char): the line to count

Returns:

num_instances (int): the number of times the line occurs.

—

src.**count_line_numpy_style_help**(*file*, *line*)

Counts the number of times a line occurs. Case-sensitive. White space padding are ignored.

USAGE:

```
num_instances = count_line_google_style_help(file, line)
```

Parameters

file: cellstr

content of file to scan

line: char

the line to count

Returns

num_instances: int

the number of times the line occurs.

—

src.**count_line_rst_style_help**(*file*, *line*)

Counts the number of times a line occurs. Case-sensitive. White space padding are ignored.

USAGE:

```
num_instances = count_line_google_style_help(file, line)
```

Parameters

- **file** – content of file to scan
- **line** – the line to count

Returns

- **num_instances**
  (int) the number of times the line occurs.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

MATLAB MODULE INDEX

S

INDEX