

Exercise: git, documentation, unit testing, arg checks

Christoph Molnar, Florian Pfisterer

2020, 2nd week

Exercise

The [Kaya Identity](#) is an equation that expresses yearly CO2 emissions as a product of four factors:

- Population size (in millions)
- GDP per capita (in 1000\$/person)
- Energy Intensity (in Gigajoule/\$1000GDP). Energy Intensity is the energy needed to produce a certain amount of economic value.
- Carbon Intensity (in tonnes CO2/Gigajoule). Carbon Intensity is the CO2 emitted for produced energy. This number depends on the energy mix used (coal, solar, ...).

The Kaya Identity can be computed as: $\text{co2} = \text{pop} * \text{gdp} * \text{enInt} * \text{carbInt}$

You can see by the units that they automatically resolve to million tonnes of CO2.

For the following exercises, you are free to use either R or Python 3.

1. Installation and Initialization

- Before implementing this, create a [Github](#) account if you don't have one yet. Github is a web platform for hosting code and collaborating on code projects. Also [install the version control software git](#) locally on your machine. git is a programmer tool for tracking changes of source code.
- Initialize an empty repository on Github (hint: click "New Project") and clone the repository to your local machine as well (hint: *git clone*). You have to [add an SSH key](#) for the communication between your machine and Github.
- Initialize an empty [R package](#) or [Python package](#) locally in the repository. For convenience you can use the `create` function from the `devtools` R package.

2. Implementation

- Implement the Kaya Equation as a function. Don't think too complicated, the function is easy to implement on purpose. Try out your function: What do you get for Germany? Inputs:
 - pop (mio): 82.4
 - gdp (in \$1000/person): 44
 - enInt (in GJ/\$1000GDP): 5
 - carbInt (in tonnes CO2/GJ): 0.05
- Document your function with [roxygen](#) (R) or [Docstring](#) (Python)
- Commit all your changes and push the commits to Github (hint: *git add*, *git commit*, *git push*).

3. Testing

- What happens if you input nonsense to your function, like a negative population size? Introduce argument checks that throw an error if the input is not meaningful. Use `checkmate` package for R, and throw errors for Python.

- b. Write a few unit tests for your function. Unit tests check whether your code behaves correctly and makes it safer for you to make code changes later. One test should check if the function returns the correct results. Calculate the correct results by hand and base your unit test on this result. Also write a test that checks whether the function throws an exception when you input negative numbers. Use the `testthat` package for R and `unittest` in Python.

4. Github Issues

It would be great to also have the option to output the tonnes of Carbon (C) instead of CO2. Go to Github and open an issue for this. Explore how you can visualize the issue in a Board (hint: left navigation bar, Issues/Boards) and drag it between boards.

5. Branches and Pull Requests

- a. Create a new git branch on your local machine. You can call the new branch e.g. “feature_c” (hint: `git checkout -b feature_c`).
- b. Work in the new branch. Add a new argument `output_type` to your function, which can take on the options “CO2” and “C” (for Carbon), where “CO2” is the default. With option “CO2” the output remains the same. But when the option “C” is chosen, the results should be given in tonnes C instead of tonnes CO2. 3.67 tonnes of CO2 equals 1 tonne of C. What do you get for the inputs of 4) as a result with `output_type = “C”`? Add at least one test for this new functionality.
- c. Commit your changes in this new branch. Add the number of the issue (probably #1) to the commit message. Push the new branch with these changes to Github.
- d. Go to Github and open a Pull Request from this new branch to your master branch. What do you see? Explore the diff, the options for commenting on the pull request and so on. The pull request will be a way to also collaborate in the teams. Finally merge the branch into master on Github.
- e. Go back to your local machine and pull the master from Github into your local master branch. You could have achieved the same by merging locally your branch with your local master branch.

6. Watch [this video about Continuous Integration](#).

7. If you finish this exercise early, please help your teammates.

Link collection

- Manual for Git-Setup on moodle page.
- [Git book](#), especially chapter 2 & 3 on the basics and branching are highly recommended.
- [Git tutorial](#).
- [Style Guide](#) of the computational statistics working group.
- [Issues](#).
- [Branching and merging in Git](#).
- [How to work with .gitignore](#)