Ways of working

Corneel den Hartogh

2024-11-08

Table of contents

Ways of working

CEDA is a project within the pilot hub 'Studiedata & AI' (Educational data & AI) of the Npuls program. This repository is used to document the ways of working within CEDA. This document discusses what type of activities CEDA executes and how this is done.. In addition, there are certain procedural basics that are needed for collaboration, like getting access etc. These are discussed in the basics document.

Activities

CEDA delivers the following types of activities:

1. Data Analytics
    - Think of data preparation, modelling and dashboarding of educational data.
    - We build and actively share code for data analytics of uniform available data through public repositories.
2. Data Engineering
    - Think of code structure, containerization, connectors to storage, analytics developer environments.
    - We build and actively share code and technical configuration for open source tooling that supports the sharing and (re-)usage of the data analytics repositories.
3. Alignment & Coordination
    - Think of community contact, reference architecture and the role of analytics in flexibilisation.
    - We ensure that the results of the data analytics and engineering are aligned with:
        1. the practical needs of (data analists in) educational institutions
        2. the technical needs of the national data infrastructure
        3. the strategic vision for Dutch tertiary education

These activities all have their own specific guidelines and requirements. However, some overarching ways of work can be defined.

General

Sharing progress asynchronously

Since we all have other responsibilities (at our primary employer) and don't see each other daily, it is important to share our progress transparently and asynchronously.

For guidelines per issue see:

Issue guidelines

- Always use EPIC: Centre for Educational Data Analytics (CEDA)
- Always use Compontent: CEDA
- Always use a label to signify the 'product' the task contributes to
- A clear description per task of what is expected in max 5 sentences
- A task should take approximately a day of actual work

- Use the checklist or comments to document details and show progress

This results in an overview, with the quick filters 'only my issues' and 'recently updated' available to limit the scope of issues.

Repository guidelines

CEDA strives to have clear descriptive names for repositories.

All commit messages should be in British English and have the style:

<type>: <specification>

Types should be, if reasonably possible, chosen from the following list:

Table 1: This makes commits much more easy to navigate and overview.

type	usage
fix	For commits that fix bugs
add	For commits that add new functionality
temp	For commits that have temporary code
style	For commits that focus on style improvements
update	For commits that primarily update because data has been renewed
del	For commits that primarily remove scripts or code
move	For commits that primarily move scripts and code

Open Source

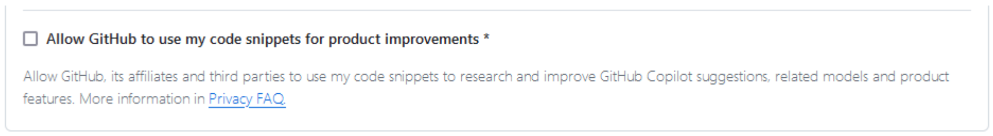
Open source is more than publishing code on github. Also, it is not only a technical or marketing choice but aligned with our place in the public sector. Practically this means that there are specific guidelines set up for public code in public sector. CEDA strives to conform to these guidelines.

See for instance, the omprehensive general standard from public code. Or the guidelines of Github themselves regarding repository quality. Regarding open source, we are in close contact with the SURF OSPO and others within Npuls. You can also see tips and tricks from the OSPO-NL Kennisbank.

Currently, there is no SURF policy regarding open source. Until there is, we have a couple of guidelines and considerations:

- For now, we will use the MIT licence by default. It is widely used and respected and easy to interpret.

- For now, we will publish on Github. It is a commercial US-based platform, but the most widely used for open source software and it has many features. Potentially, alternatives like https://code.europa.eu/ and https://opencode.de/ might be considered.
  - Within Github you don't have to allow Github to use your code for Github Copilot. Go in the top right corner to your avatar and click on 'Copilot'. Make sure there is no checkmark at:

    ☐ **Allow GitHub to use my code snippets for product improvements ***

    Allow GitHub, its affiliates and third parties to use my code snippets to research and improve GitHub Copilot suggestions, related models and product features. More information in Privacy FAQ.
  -
- A drawback of open source is that, even with the limitations of this license and not offically allowing Github to use the code for Copilot, code might still be scraped without permission for commercial purposes like monetizing LLMs or other business. This is an inherent consequence of open source.

Data analytics

Data analytics guidelines: Repositories

Data analytics repositories are either one of the following:

1. Data product (from raw data to clean, validated and enriched data)
2. Analysis product (based on data product but enriched with a model-based prediction)
3. Information product (based on either a data or analysis product, but a visualization or sort of summary)
   - Q: Is dashboard or final product a better name?

Data analytics guidelines: Language

Code and dashboards of CEDA should be open source and machine readable. We support the following:

- Python
- R
- PowerBI
- Azure

While PowerBI and Azure are itself not open source, we can still deliver the logic of the dashboards snf cloud configuration open. In addition, most educational institutions use the Microsoft Stack and others are planning to move to it. Regarding Azure we try to minimize the vendor-lock in risks by using within this environment as much open source tools (like Python and R) as possible.

Data analytics guidelines: Content of repository

Based upon: public code guidelines and adapted for data science.

1. A readme file which explains at least the goal of this repository and the context (how the acquire the start data and (a reference to) the outcome and impact of the outcome).
    - CEDA strives to make a uniform readme file.
2. Every repository has a clear structure in line with the language-specific best practices for data science.
    - For Python we will test the popular cookiecutter data science framework.
        - Depending on our tests we can work with a limited version of this.
    - For R we will use a package with additional files for interactive use.
        - The current R tooling is not one-on-one suited for our purposes.
        - To get a possible example: https://github.com/IndrajeetPatil/statsExpressions
3. Every repository has synthetic or dummy 'start' data.
4. There is a config file which has a setting for at least every institution-specific setting.
    - For instance, links to file locations, the name or BRIN of the institution, etc.
5. A correct working how-to Quarto file that explains and executes the underlying code step by step.
    - For Power BI repositories this is different.
        - Screenshots or a screencast might then be useful?
6. The main language in a repository is English. This goes for code, comments and documentation.
    - The repository name and column names can be (partly) Dutch due to specific terms.
        - Wiisselstroom for example, any translating would confuse most users.
    - In that case the data dictionary ensures that non-Dutch users can understand the terms.
    - As CEDA we strive to align the glossaries over all repositories.

7. Well styled code, also language specific.

    - Within in R the lintr package helps to conform to the tidyverse styleguide.
        - TODO: Add additional rules to lintr.
    - With Python the ruff package helps to conform to Python styleguides.
        - TODO: Implement Ruff in a specific way.

8. Files are machine-readable (.py, .R, .csv., .yaml, .json, md, qmd).
9. Data dictionaries at start and end.

    - All the start data has a clear dictionary with metadata and meaning (description). This also goes for the resulting product.
        - As CEDA we strive to have common dictionaries based on the type of repository (data, analysis or information) regardless of specific language.

10. Every repository is 'complete'. It has synthetic data, necessary metadata, config, documentation, etc, within. Of course, it can have references to external sources, but the all the information should be findable from the repository.

For later on:

- All the data files at start and end of the repository are automatically checked by validation rules.

    - As CEDA we strive to have uniform data validation rules and implementations
    - For instance, based on great expectations for Python and validate for R.

Data analytics guidelines: Project

Additionaly to the repository every project needs to

1. One or more workshops with data science professionals from other educational institutions to get it working for them

    - This is important to get feedback and to ensure that the product fullfills a need.

2. A blog or interview oriented at end-users to describe the whole project, some interesting real world insights about your own institution and something about the embedding within your own institution

Data analytics guidelines: Steps towards goals

Of course, data analytics is never done and never perfect. The list above at contents of repository does not need to be achieved 100% in order to have a successful project. In addition, the starting point of specific sub-projects will differ. Therefore, every lead of a subproject discusses with the CEDA project lead what the next few steps are. There is no need to plan the whole thing in detail, since we work agile.

Data engineering

Data engineering guidelines: Activities

The first set of repositories is about getting from raw data to useful information products. The data engineering part is about making it easy to set up and develop these repositories and making it easy to share and re-use this.

- Setting up containerization and automation
  - Think of Docker containers, config files, devcontainers and github actions.
  - This can also be added to the data analytics repositories
- Develop packages, repository structures, coding best practices, tests and automated tooling.
- Collaborate with the data analytics colleagues and projects regarding the application and development of these tools and practices.
- Experiment with open source tools in the Azure cloud. An important element is to find the limits of what we can get to work and document also the 'failures'. This is important, because CEDA could, if deemed useful, hire an external professional for advanced (cloud) engineering.