NPLinker Community Meeting

2024-03-05

netherlands eScience center

netherlands eScience center

Agenda

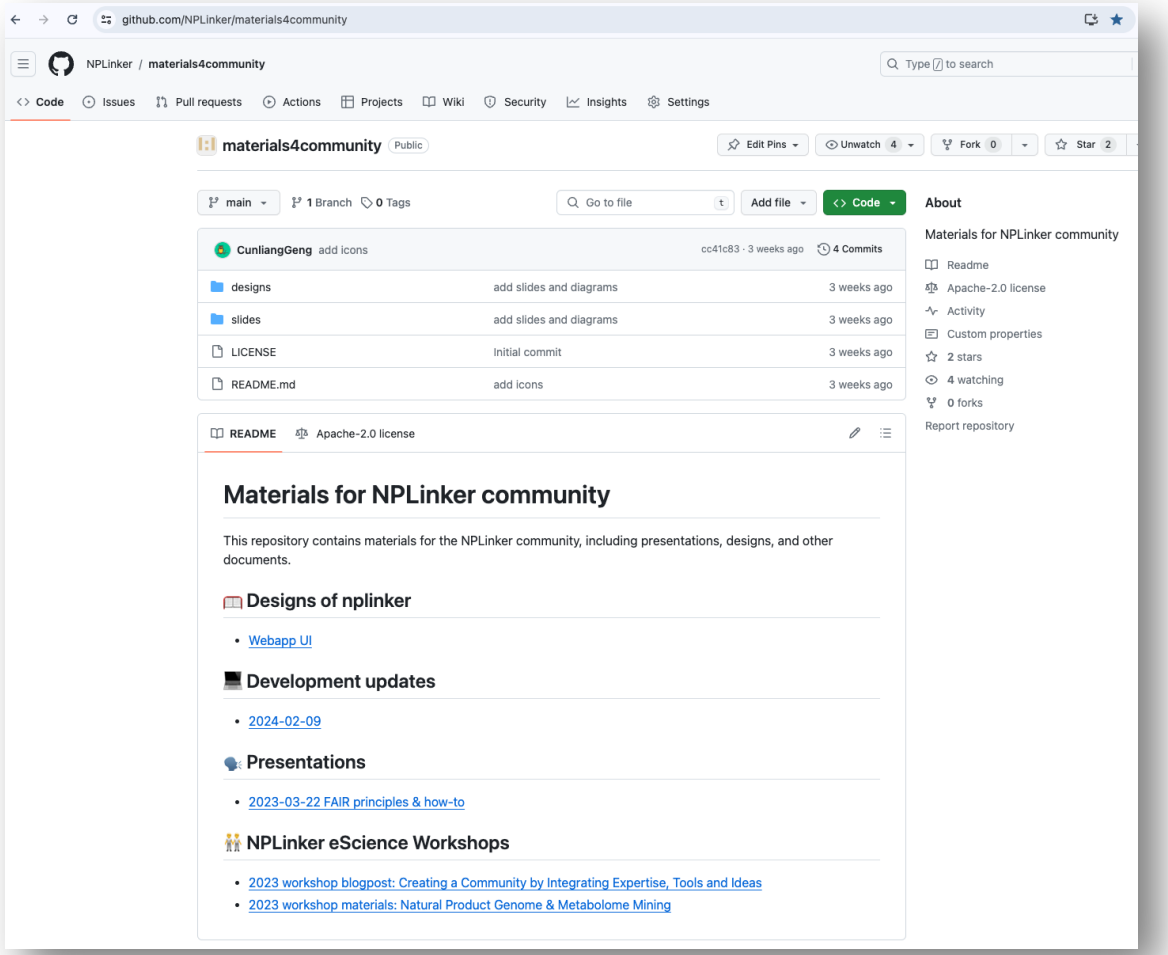
16:00	Short Introduction
16:05	Announcements and upcoming events
16:10	NPLinker development
16:35	Monthly plan
16:40	Q&A
17:00	.

Announcements and upcoming events

Arjan Draisma has joined the force of nplinker development and will first integrate BigScape v2 into NPLinker.

New repo for sharing the materials for NPLinker community
https://github.com/NPLinker/materials4community

The slides of community meetings are shared there.



netherlands eScience center

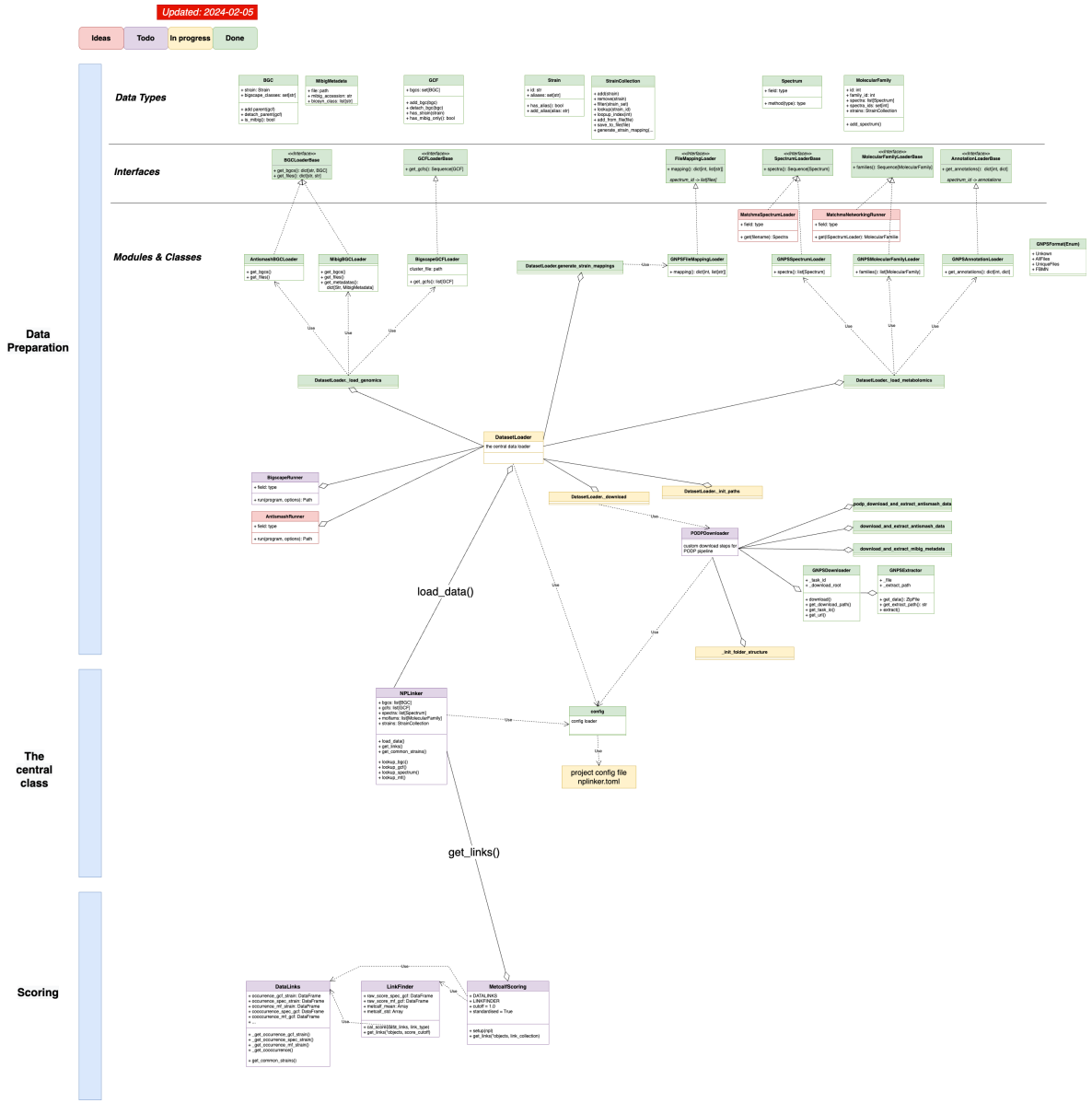
NPLinker Development

New features & changes

Contributors: Cunliang Geng, Giulia Crocioni

netherlands eScience center

NPLinker architecture



Data Preparation

- **Arranging data:**
  - Initialize project folder
  - Provide or download/extract data
  - Validate data
- **Loading data:** load data from files to python objects
  - ✅ in February's update

netherlands eScience center

New project directory structure

The project directory structure was redesigned to make it more structured, well-defined and flexible.

It's also important for enabling the local mode.

- **local mode**: users provide input data directly to nplinker
- **podp mode**: users provide a PODP id and let nplinker get input data from PODP platform

With this new project dir structure, users are required to

- Create a `root_dir` manually
- Create a config file `nplinker.toml`
- [Local mode] Provide
  - `strain_mappings.json`
  - input data (gnps, antismash, bigscape…)
- Run nplinker in the`root_dir`

[Issue 163]

```
root_dir
    │
    ├── nplinker.toml
    ├── strain_mappings.json
    ├── strains_selected.json
    │
    ├── downloads
    │       ├── paired_datarecord_4b29ddc3-26d0-40d7-80c5-44fb6631dbf9.4.json
    │       ├── GCF_000016425.1.zip
    │       ├── GCF_0000514975.1.zip
    │       ├── c22f44b14a3d450eb836d607cb9521bb.zip
    │       ├── genome_status.json
    │       └── mibig_json_3.1.tar.gz
    │
    ├── gnps
    │       ├── spectra.mgf
    │       ├── molecular_families.tsv
    │       ├── annotations.tsv
    │       └── file_mappings.tsv
    │
    ├── antismash
    │   ├── GCF_000016425.1
    │   │   ├── xxx.region001.gbk
    │   │   └── ...
    │   ├── GCF_000016425.1
    │   │   ├── xxxx.region001.gbk
    │   │   └── ...
    │   └── ...
    │
    ├── bigscape
    │   ├── mix_clustering_c0.30.tsv
    │   └── bigscape_running_output
    │       └── ...
    │
    ├── mibig
    │   ├── BGC0000001.json
    │   ├── BGC0000002.json
    │   └── ...
    │
    └── ...
```

netherlands eScience center

New template for config file

The settings for NPLinker were redesigned to make the config file as simple as possible.

Now users usually only need to set values for `root_dir`, `mode` and/or `podp_id`.

```
############################
# NPLinker configuration file
############################

root_dir = "<NPLinker root directory>"
mode = "podp"
podp_id = ""

[log]
level = "INFO"
file = "path/to/logfile"
to_stdout = true

[mibig]
to_use = true
version = "3.1"

[bigscape]
parameters = "--mibig --clans-off --mix --include_singletons --cutoffs 0.30"
cutoff = "0.30"

[scoring]
methods = ["metcalf"]
```

[PR 203, 206, 209]

netherlands eScience center

New config loader and validators

We added the config loader and validators by taking advantage of the library Dynaconf.

So users are able to

- specify the path to config file with the env variable `NPLINKER_CONFIG_FILE`
- Or directly put the config file `nplinker.toml` in the root dir

Validation of the config will be triggered by python's import, e.g. `import nplinker`, so validation is done even before running any code of nplinker.

[PR 209]

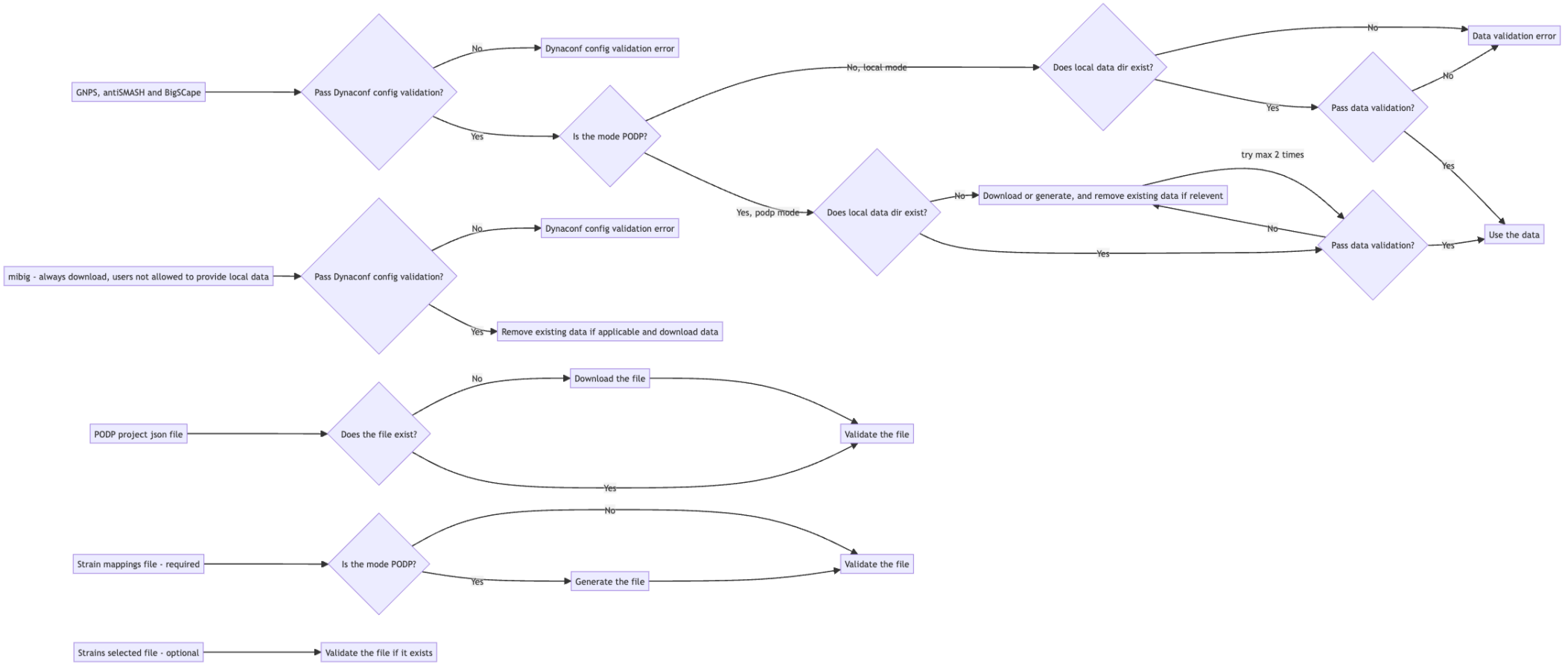
```python
# Locate the user's config file
user_config_file = os.environ.get("NPLINKER_CONFIG_FILE", "nplinker.toml")
if not os.path.exists(user_config_file):
    raise FileNotFoundError(f"Config file '{user_config_file}' not found")

# Locate the default config file
default_config_file = Path(__file__).resolve().parent / "nplinker_default.toml"

# Load config files
config = Dynaconf(settings_files=[user_config_file], preload=[default_config_file])
```

```python
validators = [
    # General settings
    ## `root_dir` value is transformed to a `pathlib.Path` object and must be a directory.
    Validator(
        "root_dir", required=True, cast=transform_to_full_path, condition=lambda v: v.is_dir()
    ),
    Validator("mode", required=True, cast=lambda v: v.lower(), is_in=["local", "podp"]),
    ## `podp_id` must be set if `mode` is "podp"; must not be set if `mode` is "local".
    Validator("podp_id", required=True, when=Validator("mode", eq="podp")),
    Validator("podp_id", required=False, when=Validator("mode", eq="local")),
    # Log
    ## `loglevel` must be a string and must be one of the supported levels. It is transformed to
    ## uppercase to avoid case sensitivity.
    Validator(
        "log.level",
        is_type_of=str,
        cast=lambda v: v.upper(),
        is_in=["NOTSET", "DEBUG", "INFO", "WARNING", "ERROR", "CRITICAL"],
    ),
    Validator("log.file", is_type_of=str, cast=Path),
    Validator("log.to_stdout", is_type_of=bool),
    #  Mibig
    Validator("mibig.to_use", required=True, is_type_of=bool),
    Validator(
        "mibig.version",
        required=True,
        is_type_of=str,
        when=Validator("mibig.to_use", eq=True),
    ),
    # BigScape
    Validator("bigscape.parameters", required=True, is_type_of=str),
    Validator("bigscape.cutoff", required=True, is_type_of=str),
    # Scoring
    ## `scoring.methods` must be a list of strings and must contain at least one of the
    ## supported scoring methods.
    Validator(
        "scoring.methods",
        required=True,
        cast=lambda v: [i.lower() for i in v],
        is_type_of=list,
        len_min=1,
        condition=lambda v: set(v).issubset({"metcalf", "rosetta"}),
    ),
]
```

netherlands eScience center

Pipeline of arranging data

The pipeline was designed to keep arranging data independently for each type of data while keeping them sharing common logics.



[Issue 117]

netherlands eScience center

DatasetArranger

The data arranging was coupled with data loading before.

By following the new pipeline of arranging data, a new class `DatasetArranger` and some validation functions were developed to be independently responsible for the steps of arranging data.

It supports both the local mode and podp mode.

```
140     def load_data(self):
141         """Loads the basic components of a dataset."""
142         arranger = DatasetArranger()
143         arranger.arrange()
144         self._loader.load()
```

It’s ready for you to try and test the refactored nplinker using your own data when this PR is reviewed and merged to dev branch.

[PR 215]

netherlands eScience center

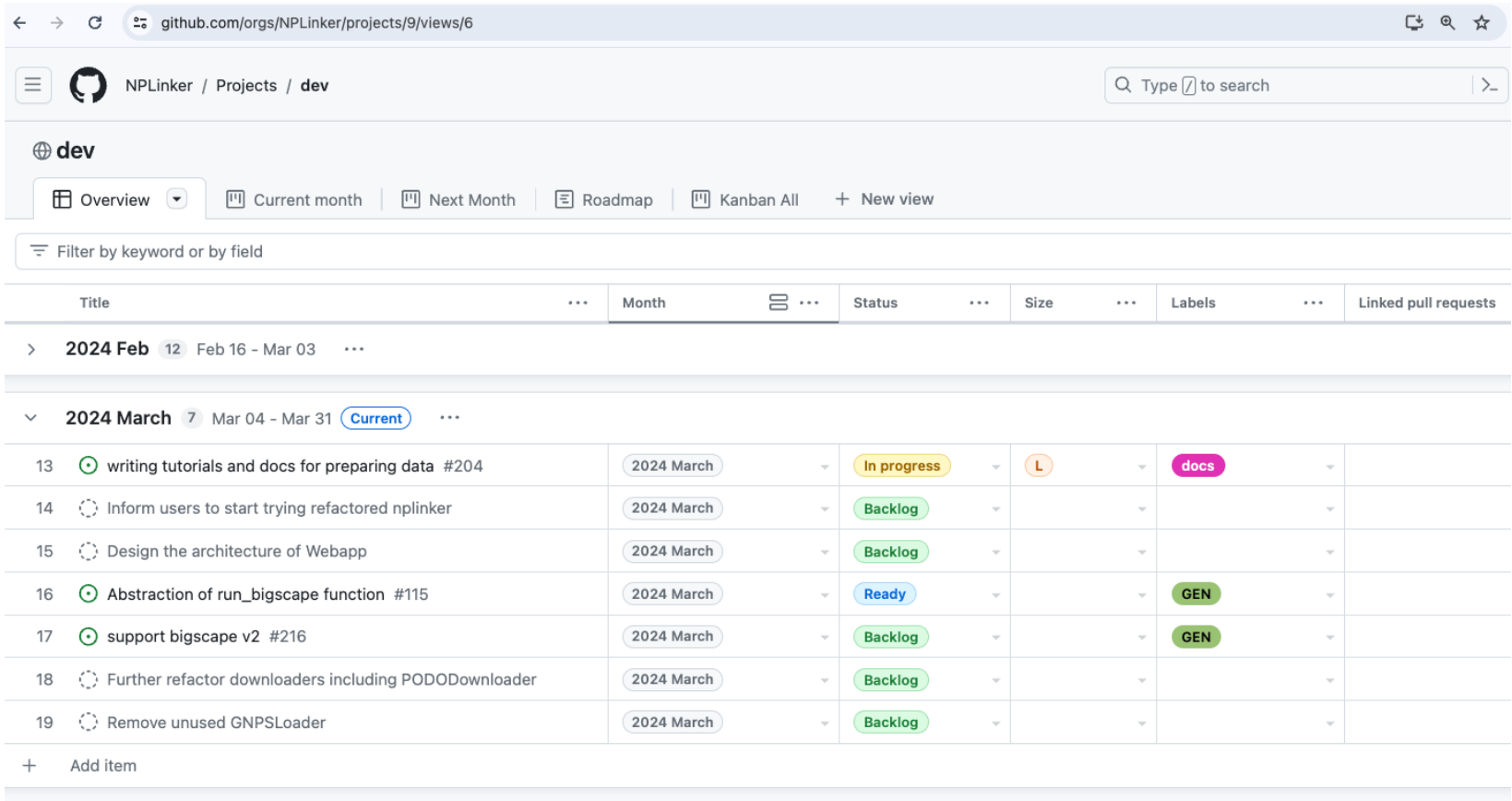
Monthly plan

What will happen before next meeting

netherlands eScience center

New Kanban board enhanced with planning feature

https://github.com/orgs/NPLinker/projects/9



netherlands eScience center

Q&A

Questions? Ideas? Feedback?