Developing an Open Source and FAIR Ecosystem for Cheminformatics

Martin Šícho (Martin.Sicho@vscht.cz) – RDKit UGM 2025 – 2025-09-11

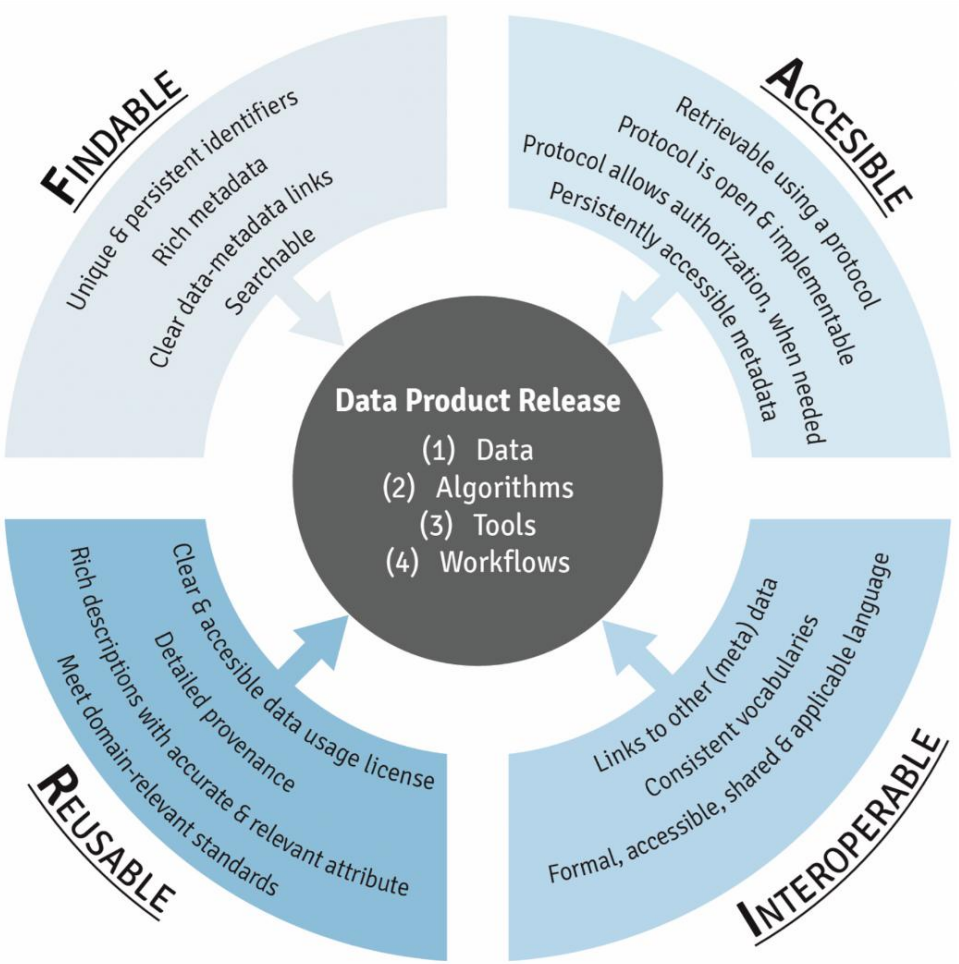
Universiteit Leiden

UNIVERSITY OF CHEMISTRY AND TECHNOLOGY PRAGUE

LACDR

Problems with Scientific Software and Data

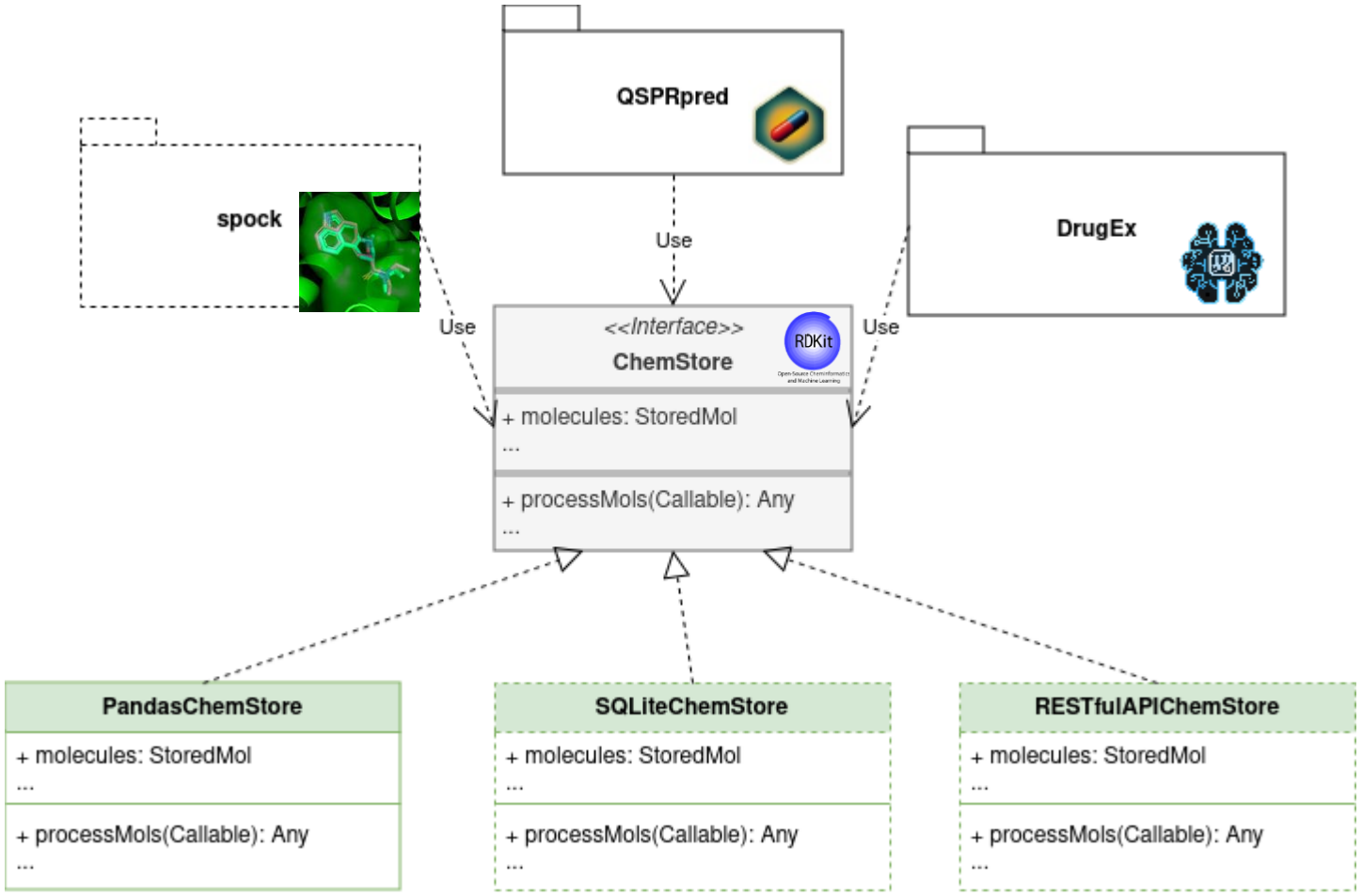
- Common pains:
  - **poorly documented** (i.e. no metadata)
  - **poorly structured** (i.e. lack of encapsulation, not installable, hard-coded parts)
  - **insufficiently tested** (i.e. small errors often lead to big problems for the users)
- Resulting problems:
  - steep learning curve for newcomers (i.e. students)
  - low adoption of novel innovative methods
  - reinventing the wheel
  - reproduction of scientific results
  - communication and knowledge sharing
  - reusability and interoperability of software developed by different people



Barker, M., Chue Hong, N.P., Katz, D.S. et al. Introducing the FAIR Principles for research software. Sci Data 9, 622 (2022). https://doi.org/10.1038/s41597-022-01710-x

Open Source Software Stack for Cheminformatics

https://github.com/CDDLeiden/



- **F**indable
  - unique and persistent identifiers
  - rich metadata
  - searchable
- **A**ccessible
  - retrievable using a protocol (i.e. JSON for metadata)
  - metadata is persistent and always readable
- **I**nteroperable
  - modular encapsulated code with documented APIs
- **R**eusable
  - installable packages
  - integrated pipelines
  - automatic and transparent testing

Universiteit Leiden

UNIVERSITY OF CHEMISTRY AND TECHNOLOGY PRAGUE

LACDR

Data Storages

```python
import os
import pandas as pd

from qsprpred.data.chem.identifiers import InchiIde
from qsprpred.data.chem.standardizers.papyrus impor
from qsprpred.data.storage.tabular.simple import Pa

## Create a library of compounds for the chemokine
library = PandasChemStore(
    name="CCRs_HUMAN_ALL",
    path="./data/",
    df=pd.read_table("data/CCRs_HUMAN_ALL.tsv").samp
    standardizer=PapyrusStandardizer(), # standardi
    identifier=InchiIdentifier(), # persistent uniq
    n_jobs=os.cpu_count(), # integrated multi-processing
    overwrite=True
)
## add custom metadata (arbitrary attributes can be created)
library.description = """Bioactivity data of molecules measured on at least one chemokine receptor.
Only wild type human data is considered.
"""
library
```

```python
# searchable -> i.e. find aromatic sulfonamides
library_subset = library.searchWithSMARTS(
    ["[ar]NS(=O)(=O)([ar])"],
    name="CCRs_ar_sulf"
)
len(library_subset)
```

```python
library.save()
```

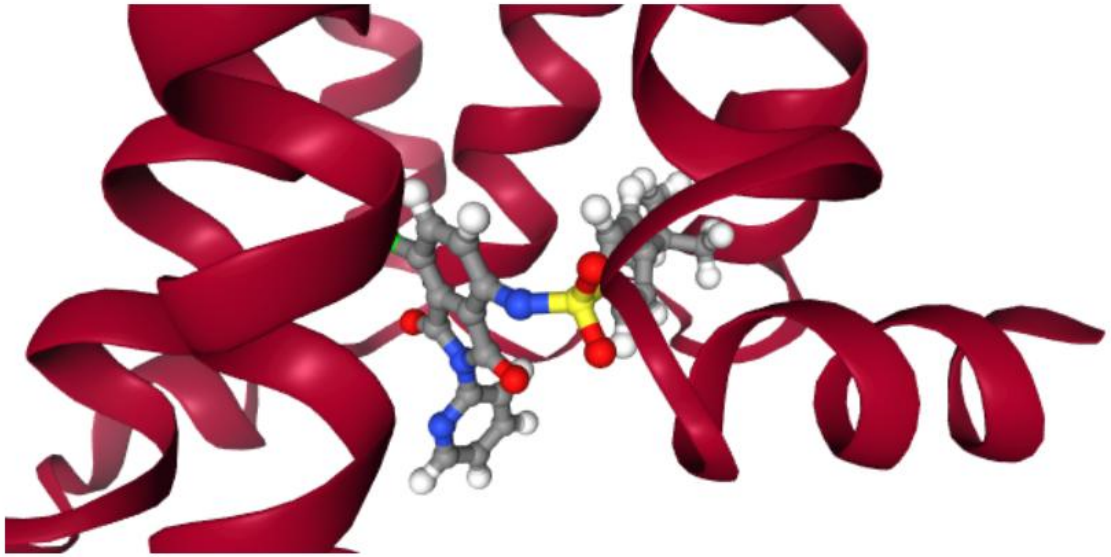
```
'/home/sichom/projects/spock/tutorial/data/CCRs_HUMAN_ALL/meta.json'
```

Docking with Spock

```python
from spock.storage.tabular import SpockProtein
from spock.docking.vina.cpu_local import VinaDockingCPULocal
import os

PROTEIN_NAME = "5T1A_clean_mutations_reversed_withHs"
N_CPUS = os.cpu_count()  # number of cpus to use for docking
EXHAUSTIVENESS = 8  # Vina exhaustiveness parameter
SEED = 42  # random seed for random operations
PROTEIN_FOLDER = './data/proteins'

docking = VinaDockingCPULocal(
    protein=SpockProtein(
        PROTEIN_NAME,
        props={
            "pdb": open(f'{PROTEIN_FOLDER}/{PROTEIN_NAME}.pdb', 'r').read(),
            "pdbqt": open(f'{PROTEIN_FOLDER}/{PROTEIN_NAME}.pdbqt', 'r').read(),
        }
    ),
    n_cpus=N_CPUS,
    box_spec={
        "center": [5.1, 28.0, 187.6],
        "box_size": [16.2, 17.8, 17.4]
    },
    embed_mols=True,  # set to False if conformers are already generated
    exhaustiveness=EXHAUSTIVENESS,
    seed=SEED,
)
```

```python
store.nJobs = os.cpu_count()
docking.dock_storage(
    storage=store,
    chunk_size=1
    save=
    overw
)
```

```python
import nglview

complex = store.get_complex_for_pose(pose_id=poses[0].id)
nglview.show_rdkit(complex)
```



slides with notes

Thank you!

demo notebook

Universiteit Leiden

UNIVERSITY OF CHEMISTRY AND TECHNOLOGY PRAGUE

LACDR