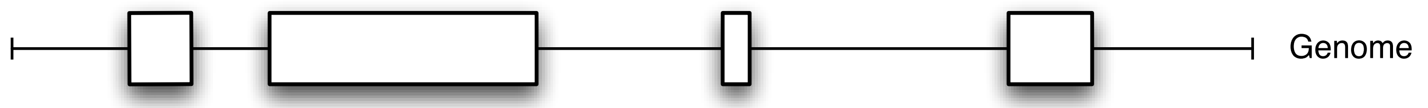
FAIRtracks and Omnipy

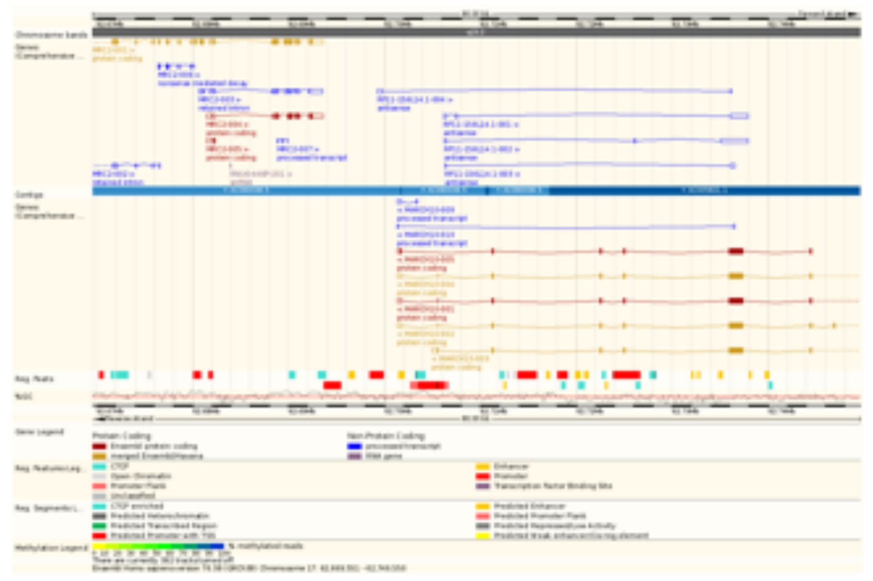
- FAIRtracks interoperability story

March 7, 2023
Sveinung Gundersen

www.elixir-europe.org

Genomic tracks





Any genomic data file mapped to a reference genome coordinate system!
- Not only for visual analysis!

- Gene regions, repeating elements, conserved regions
- Chromatin accessibility (*e.g.*, DNase I Hypersensitivity)
- Binding of Transcription Factors to DNA
- Histone modifications along DNA
- Gene expression, Gene fusion, Transcription start sites (TSS)
- Cis-regulatory elements (promoters, enhancers...)
- DNA methylation
- 3D genome structure
- GWAS SNPs for disease, SNVs and CNVs in cancer

Important goals for FAIRtracks

Findable

- Global identifiers for track files, as well as track collections, studies, samples, and experiments
- Search and import of individual track files across repositories, also repositories not supported by consortia data portals
- Search using formal (non-free-text) queries

Accessible

- Easy (automated) retrieval of track data
- Persistence of track data and versioned metadata

Interoperable

- Lack of standard metadata model with practically useful attributes
- Annotation using community-accepted vocabularies/ontologies
- Cross-references to records in relevant (meta)data repositories

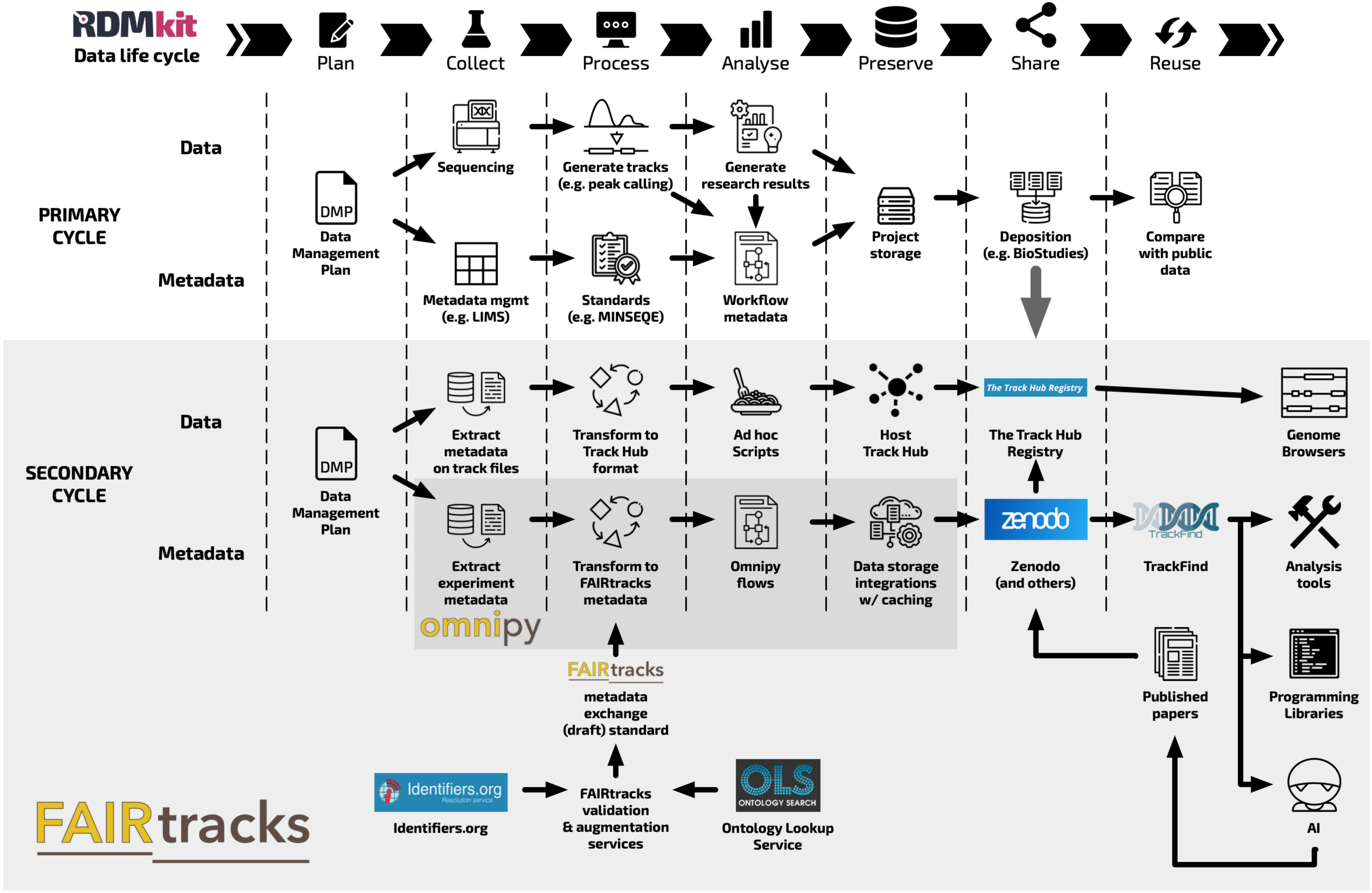
Reusable

- Support for detailed context-specific metadata content together with standardised summary attributes
- Simple process for data providers to submit data and metadata that at the same time accommodates the required stringency for (automated) downstream usability
- Easily available data usage licenses
- Detailed provenance of experimental and *in silico* analysis steps

Gundersen S, Boddu S, Capella-Gutierrez S *et al.* Recommendations for the FAIRification of genomic track metadata [version 1; peer review: 1 approved]. *F1000Research* 2021, **10**(ELIXIR):268 (https://doi.org/10.12688/f1000research.28449.1)

FAIRtracks
(tool assembly)

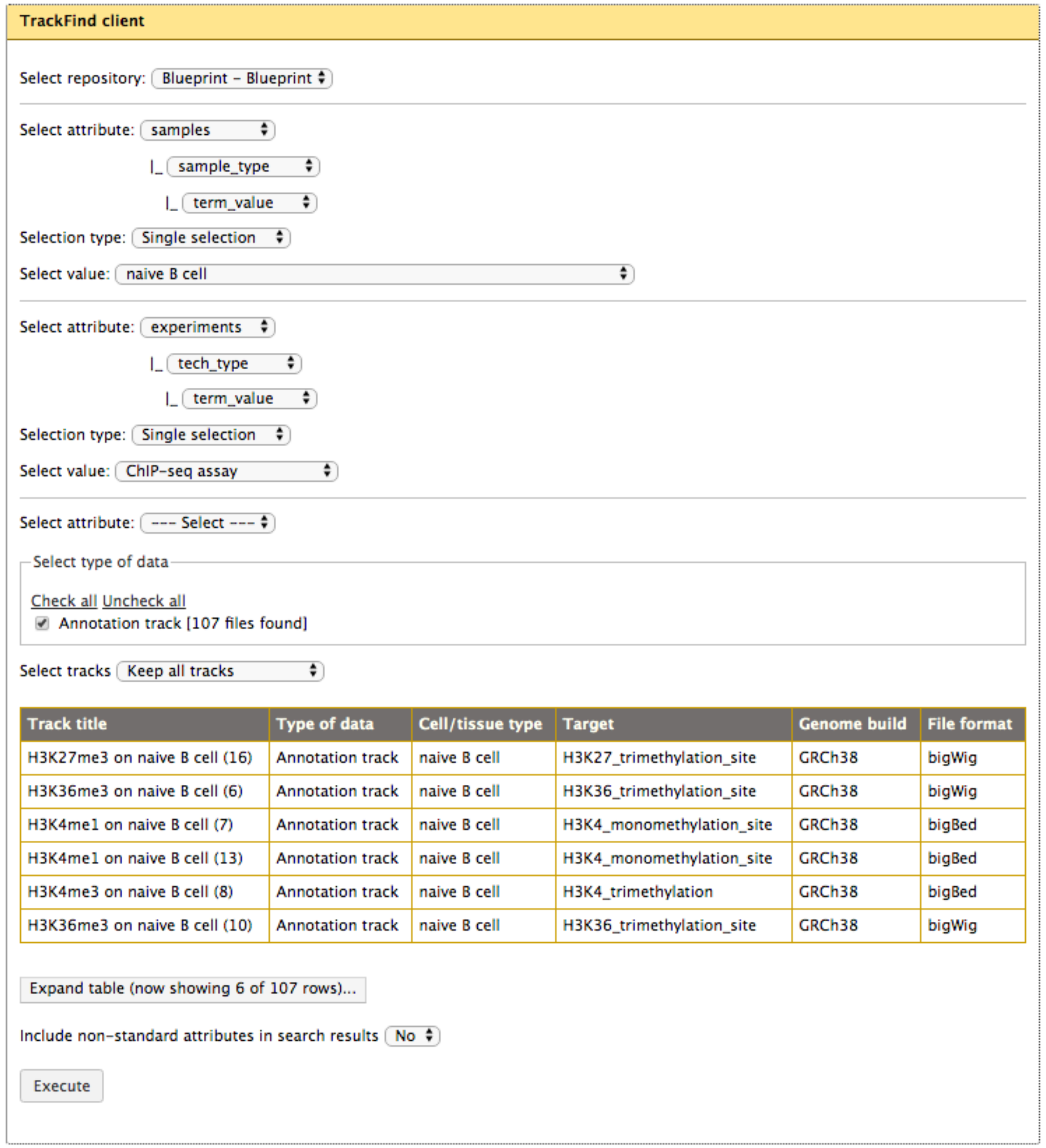
- Represents a secondary iteration of the data life cycle (as defined in RDMkit)
- FAIRtracks (more-or-less) begins when the primary data life cycle ends



Icons: RDMkit, Google Material Symbols, Flaticon.com

Tool integration

- TrackFind client implemented in GSuite HyperBrowser:
  - https://hyperbrowser.uio.no/trackfind_test (search for tool "trackfind")
- JSON and GSuite (http://gtrack.no) formats as metadata / search result exchange format
- Search results can be transferred to the HyperBrowser server, preprocessed, and used in statistical analyses
- Integration into vanilla Galaxy under development



TrackFind client

Select repository: Blueprint – Blueprint

Select attribute: samples
|_ sample_type
|_ term_value

Selection type: Single selection

Select value: naive B cell

Select attribute: experiments
|_ tech_type
|_ term_value

Selection type: Single selection

Select value: ChIP-seq assay

Select attribute: --- Select ---

Select type of data

Check all Uncheck all

☑ Annotation track [107 files found]

Select tracks: Keep all tracks

Track title	Type of data	Cell/tissue type	Target	Genome build	File format
H3K27me3 on naive B cell (16)	Annotation track	naive B cell	H3K27_trimethylation_site	GRCh38	bigWig
H3K36me3 on naive B cell (6)	Annotation track	naive B cell	H3K36_trimethylation_site	GRCh38	bigWig
H3K4me1 on naive B cell (7)	Annotation track	naive B cell	H3K4_monomethylation_site	GRCh38	bigBed
H3K4me1 on naive B cell (13)	Annotation track	naive B cell	H3K4_monomethylation_site	GRCh38	bigBed
H3K4me3 on naive B cell (8)	Annotation track	naive B cell	H3K4_trimethylation	GRCh38	bigBed
H3K36me3 on naive B cell (10)	Annotation track	naive B cell	H3K36_trimethylation_site	GRCh38	bigWig

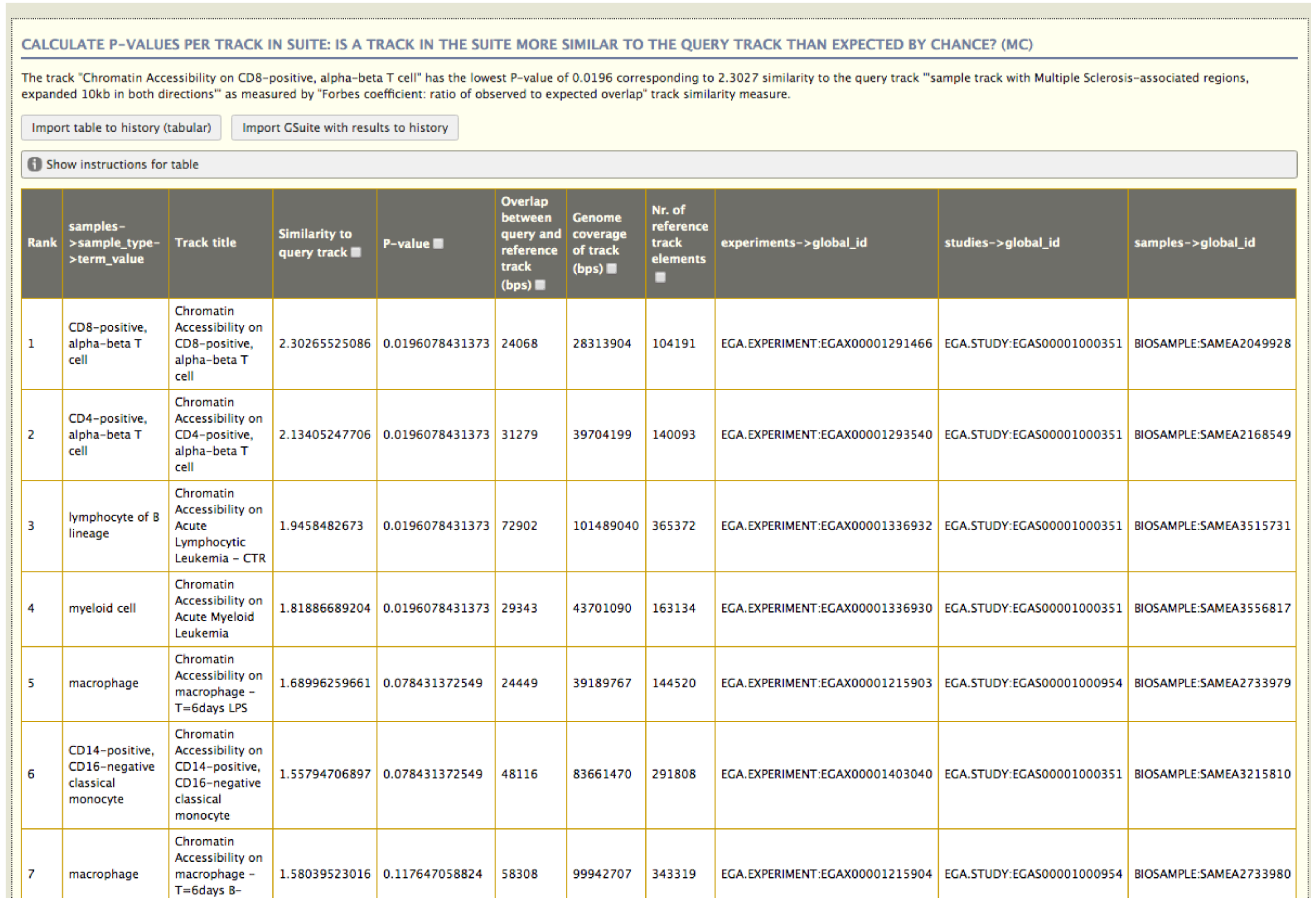
Expand table (now showing 6 of 107 rows)...

Include non-standard attributes in search results: No

Execute

Tool integration

- Analysis example
  - Sites of open chromatin for various cell types from BLUEPRINT (DNaseI HS)
  - Set of variants associated with Multiple Sclerosis (MS)
  - Which cell types are most relevant to MS? (i.e., where do the variants overlap the most with open chromatin?)



CALCULATE P-VALUES PER TRACK IN SUITE: IS A TRACK IN THE SUITE MORE SIMILAR TO THE QUERY TRACK THAN EXPECTED BY CHANCE? (MC)

The track "Chromatin Accessibility on CD8-positive, alpha-beta T cell" has the lowest P-value of 0.0196 corresponding to 2.3027 similarity to the query track '"sample track with Multiple Sclerosis-associated regions, expanded 10kb in both directions"' as measured by "Forbes coefficient: ratio of observed to expected overlap" track similarity measure.

Import table to history (tabular)
Import GSuite with results to history
Show instructions for table

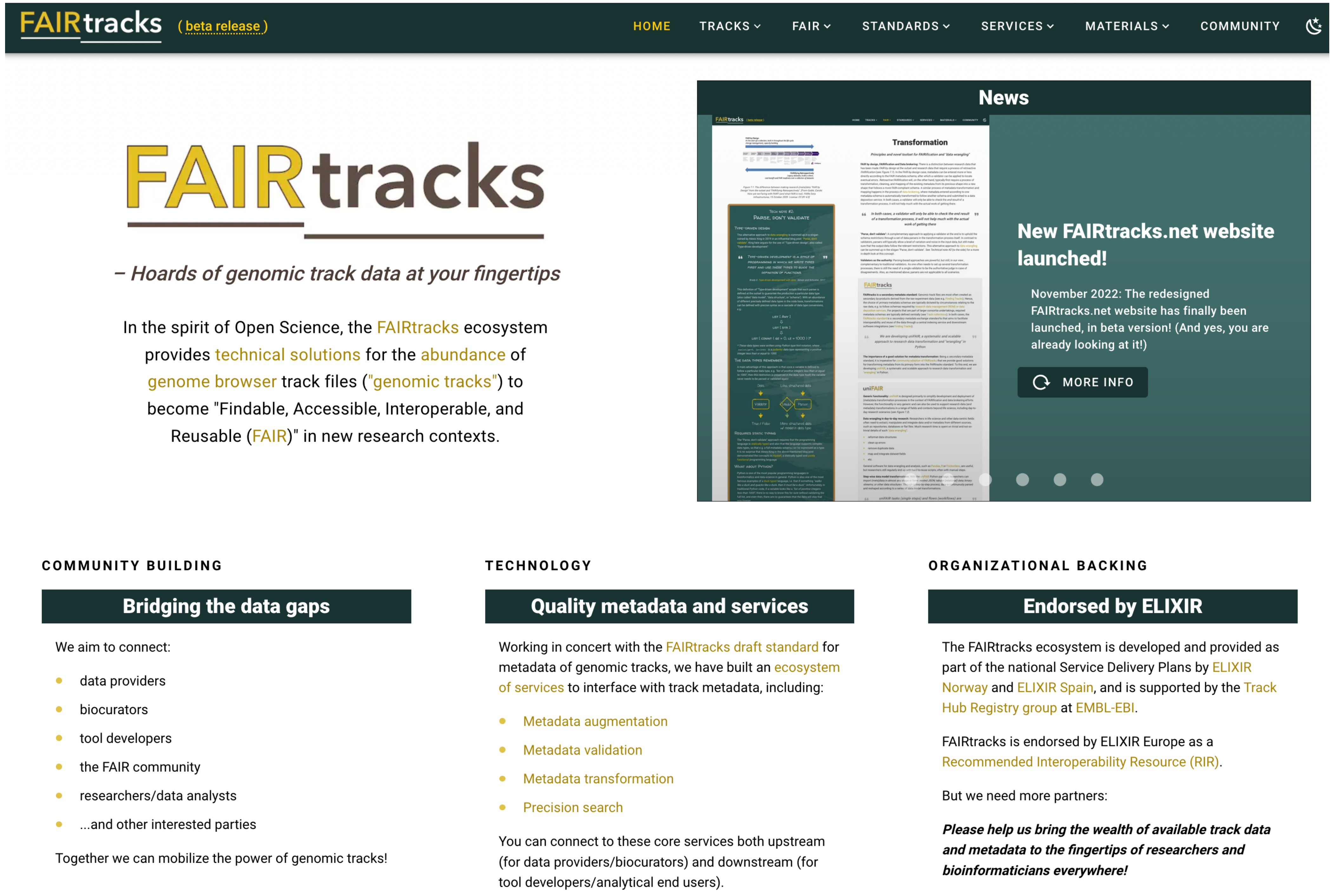
Rank	samples->sample_type->term_value	Track title	Similarity to query track	P-value	Overlap between query and reference track (bps)	Genome coverage of track (bps)	Nr. of reference track elements	experiments->global_id	studies->global_id	samples->global_id
1	CD8-positive, alpha-beta T cell	Chromatin Accessibility on CD8-positive, alpha-beta T cell	2.30265525086	0.0196078431373	24068	28313904	104191	EGA.EXPERIMENT:EGAX00001291466	EGA.STUDY:EGAS00001000351	BIOSAMPLE:SAMEA2049928
2	CD4-positive, alpha-beta T cell	Chromatin Accessibility on CD4-positive, alpha-beta T cell	2.13405247706	0.0196078431373	31279	39704199	140093	EGA.EXPERIMENT:EGAX00001293540	EGA.STUDY:EGAS00001000351	BIOSAMPLE:SAMEA2168549
3	lymphocyte of B lineage	Chromatin Accessibility on Acute Lymphocytic Leukemia - CTR	1.9458482673	0.0196078431373	72902	101489040	365372	EGA.EXPERIMENT:EGAX00001336932	EGA.STUDY:EGAS00001000351	BIOSAMPLE:SAMEA3515731
4	myeloid cell	Chromatin Accessibility on Acute Myeloid Leukemia	1.81886689204	0.0196078431373	29343	43701090	163134	EGA.EXPERIMENT:EGAX00001336930	EGA.STUDY:EGAS00001000351	BIOSAMPLE:SAMEA3556817
5	macrophage	Chromatin Accessibility on macrophage - T=6days LPS	1.68996259661	0.078431372549	24449	39189767	144520	EGA.EXPERIMENT:EGAX00001215903	EGA.STUDY:EGAS00001000954	BIOSAMPLE:SAMEA2733979
6	CD14-positive, CD16-negative classical monocyte	Chromatin Accessibility on CD14-positive, CD16-negative classical monocyte	1.55794706897	0.078431372549	48116	83661470	291808	EGA.EXPERIMENT:EGAX00001403040	EGA.STUDY:EGAS00001000351	BIOSAMPLE:SAMEA3215810
7	macrophage	Chromatin Accessibility on macrophage - T=6days B-	1.58039523016	0.117647058824	58308	99942707	343319	EGA.EXPERIMENT:EGAX00001215904	EGA.STUDY:EGAS00001000954	BIOSAMPLE:SAMEA2733980

Identifiers for genomic tracks?

- Should there be globally unique, persistent identifiers for genomic track files (e.g. BigBED/BigWIG, VCF, etc)
- No such thing exists, but we highly recommend that they should be created and indexed
- Also, identifiers for *track collections* would be very useful
  - Would easily FAIRify "Mix-and-match" track file collections analysed in particular research papers, if the track files are already FAIRified

Updates

- ELIXIR Recommended Interoperability Resource 2021
- Manuscript published Apr 2021
- Invited blog post in F1000Research published Dec 2021
- New web site published 2022: https://fairtracks.net



FAIRtracks

– Hoards of genomic track data at your fingertips

In the spirit of Open Science, the FAIRtracks ecosystem provides technical solutions for the abundance of genome browser track files ("genomic tracks") to become "Findable, Accessible, Interoperable, and Reusable (FAIR)" in new research contexts.

News

New FAIRtracks.net website launched!

November 2022: The redesigned FAIRtracks.net website has finally been launched, in beta version! (And yes, you are already looking at it!)

COMMUNITY BUILDING

Bridging the data gaps

We aim to connect:

- data providers
- biocurators
- tool developers
- the FAIR community
- researchers/data analysts
- ...and other interested parties

Together we can mobilize the power of genomic tracks!

TECHNOLOGY

Quality metadata and services

Working in concert with the FAIRtracks draft standard for metadata of genomic tracks, we have built an ecosystem of services to interface with track metadata, including:

- Metadata augmentation
- Metadata validation
- Metadata transformation
- Precision search

You can connect to these core services both upstream (for data providers/biocurators) and downstream (for tool developers/analytical end users).

ORGANIZATIONAL BACKING

Endorsed by ELIXIR

The FAIRtracks ecosystem is developed and provided as part of the national Service Delivery Plans by ELIXIR Norway and ELIXIR Spain, and is supported by the Track Hub Registry group at EMBL-EBI.

FAIRtracks is endorsed by ELIXIR Europe as a Recommended Interoperability Resource (RIR).

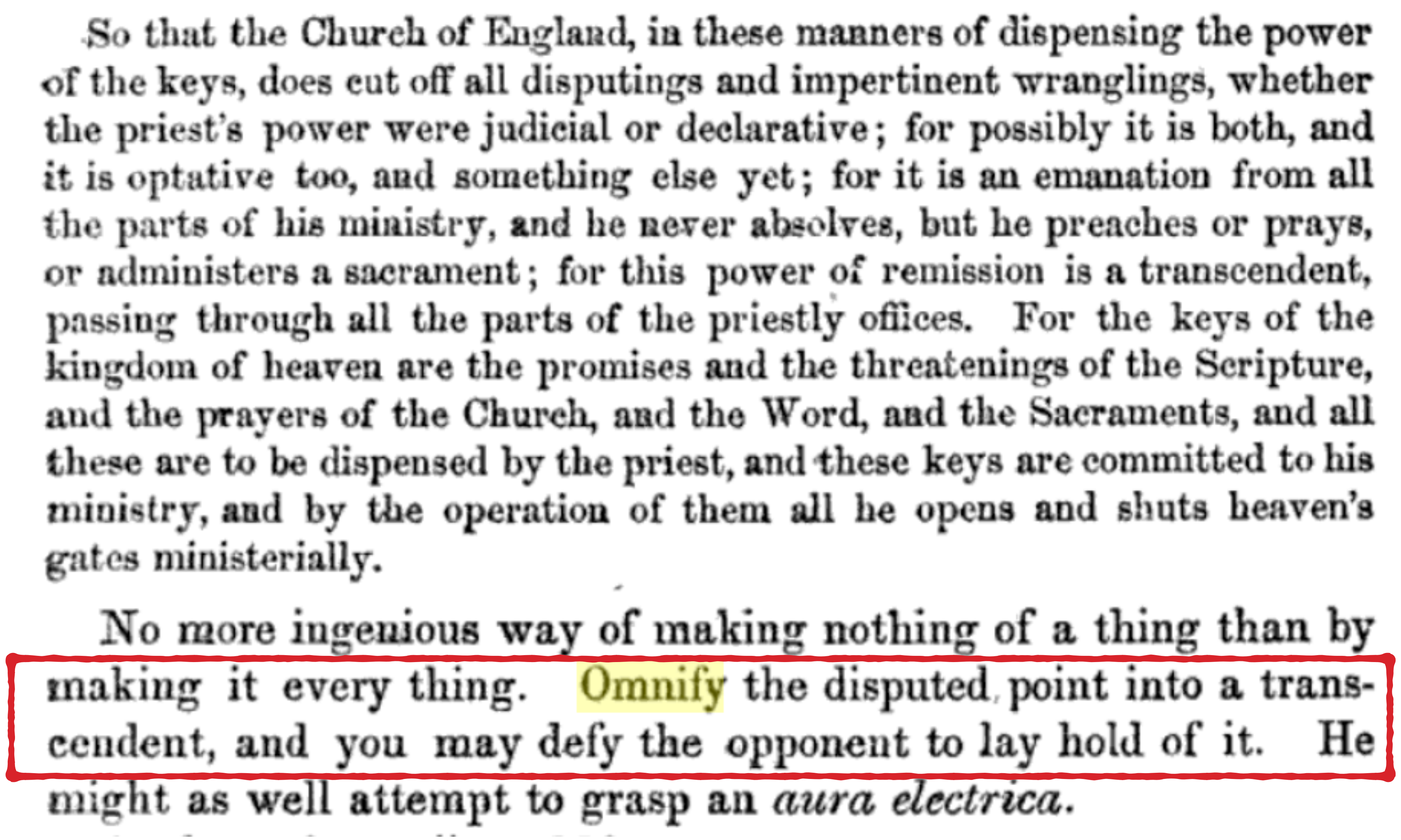
But we need more partners:

Please help us bring the wealth of available track data and metadata to the fingertips of researchers and bioinformaticians everywhere!

Projects / collaborations

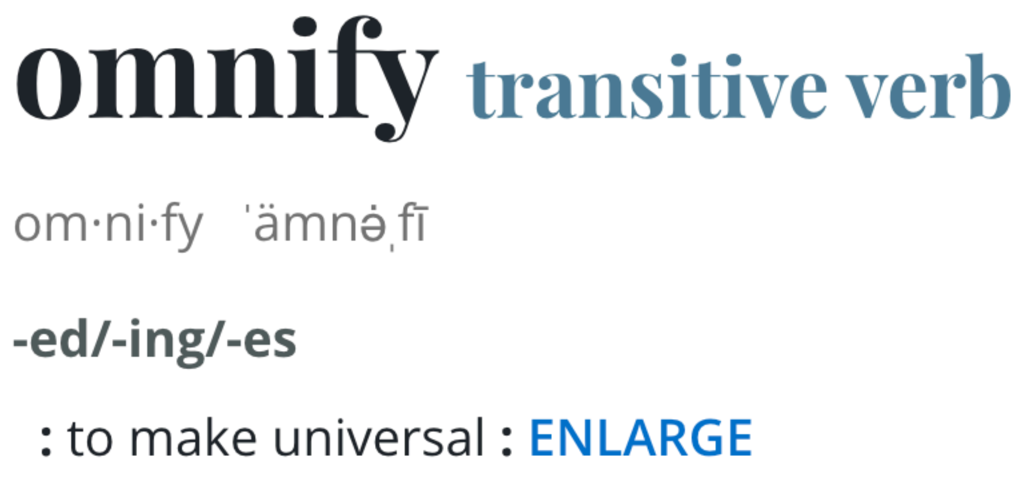
- Galaxy implementation study (2021-2023)
  - Task 1.4. Enhanced metadata functionality in Galaxy to support querying and importing of FAIRtracks-annotated genomic track data
    - Implement TrackFind client as a Galaxy tool
    - Idea: associate FAIRtracks metadata with data as a Galaxy dataset/collection, keeping this relation through downstream analysis steps
    - Investigate RO-Crate integration
- RDA TIGER (HORIZON-INFRA-2022-EOSC-01-04)
  - RDA/EOSC project started Jan 2023
  - Demonstrator Working Group for RDA TIGER (pre-approved, but still need to apply)
    - Goal is to integrate metadata from (at least) Functional Annotation of ANimal Genomes (FAANG) into FAIRtracks, and also expand tool interoperability
    - Some funding available
    - More importantly: door-opener into both RDA and EOSC

Literary digression: omnify


So that the Church of England, in these manners of dispensing the power of the keys, does cut off all disputings and impertinent wranglings, whether the priest's power were judicial or declarative; for possibly it is both, and it is optative too, and something else yet; for it is an emanation from all the parts of his ministry, and he never absolves, but he preaches or prays, or administers a sacrament; for this power of remission is a transcendent, passing through all the parts of the priestly offices. For the keys of the kingdom of heaven are the promises and the threatenings of the Scripture, and the prayers of the Church, and the Word, and the Sacraments, and all these are to be dispensed by the priest, and these keys are committed to his ministry, and by the operation of them all he opens and shuts heaven's gates ministerially.

No more ingenious way of making nothing of a thing than by making it every thing. Omnify the disputed point into a transcendent, and you may defy the opponent to lay hold of it. He might as well attempt to grasp an *aura electrica*.

- One of the most obscure words in the English language:

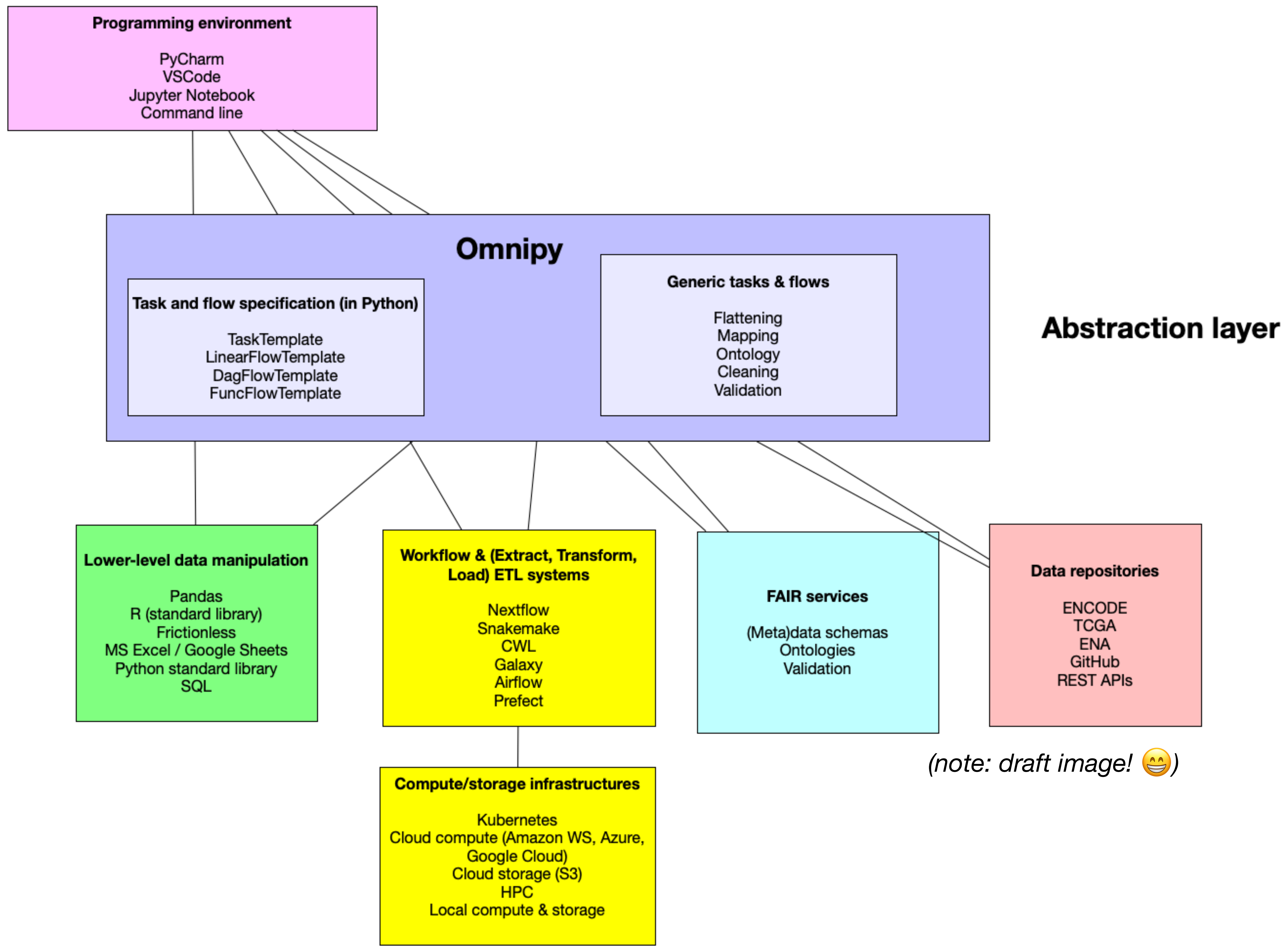

omnify transitive verb

om·ni·fy ˈämnə̇ˌfī

-ed/-ing/-es

: to make universal : ENLARGE

"Notes on Jeremy Taylor",
The Literary Remains Of Samuel Taylor Coleridge,
by Samuel Taylor Coleridge
Publication date: 1836-1839



(note: draft image! 😁)

Data portals

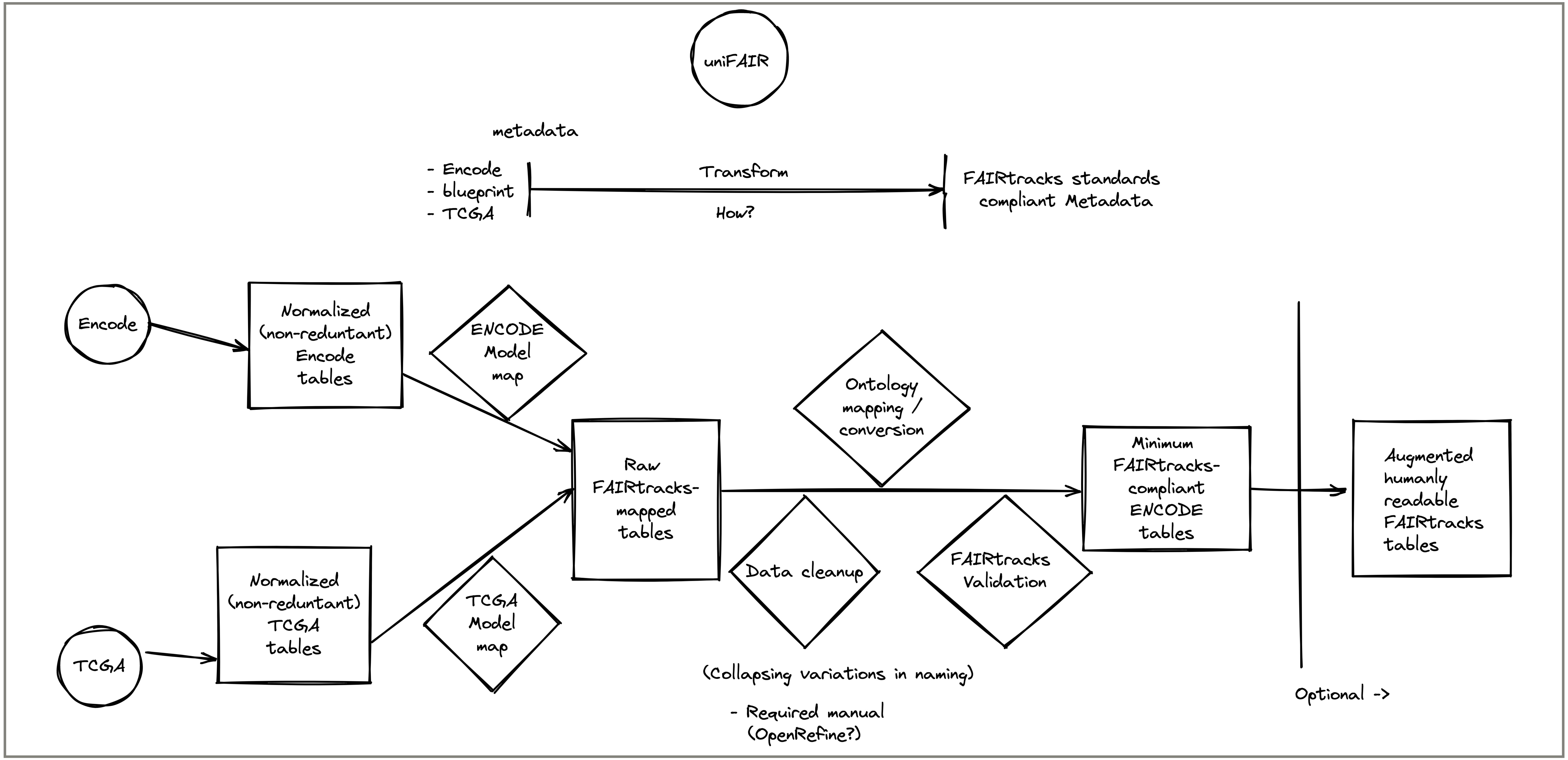
Custom metadata schemas:

ENCODE
FAANG
TCGA
ICSC
IHEC
…

Rerunnable metadata transformation workflows?

Metadata exchange standard

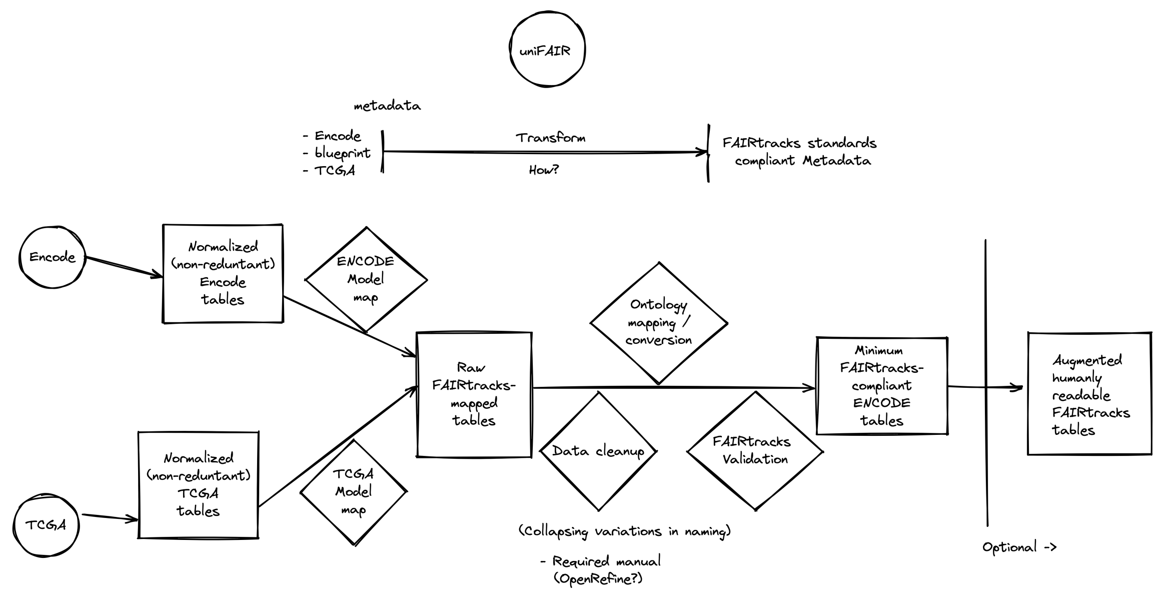
FAIRtracks


uniFAIR
metadata
- Encode
- blueprint
- TCGA
Transform
How?
FAIRtracks standards compliant Metadata
Encode
Normalized (non-reduntant) Encode tables
ENCODE Model map
TCGA
Normalized (non-reduntant) TCGA tables
TCGA Model map
Raw FAIRtracks-mapped tables
Ontology mapping / conversion
Data cleanup
FAIRtracks Validation
Minimum FAIRtracks-compliant ENCODE tables
Augmented humanly readable FAIRtracks tables
(Collapsing variations in naming)
- Required manual (OpenRefine?)
Optional ->

Data portals

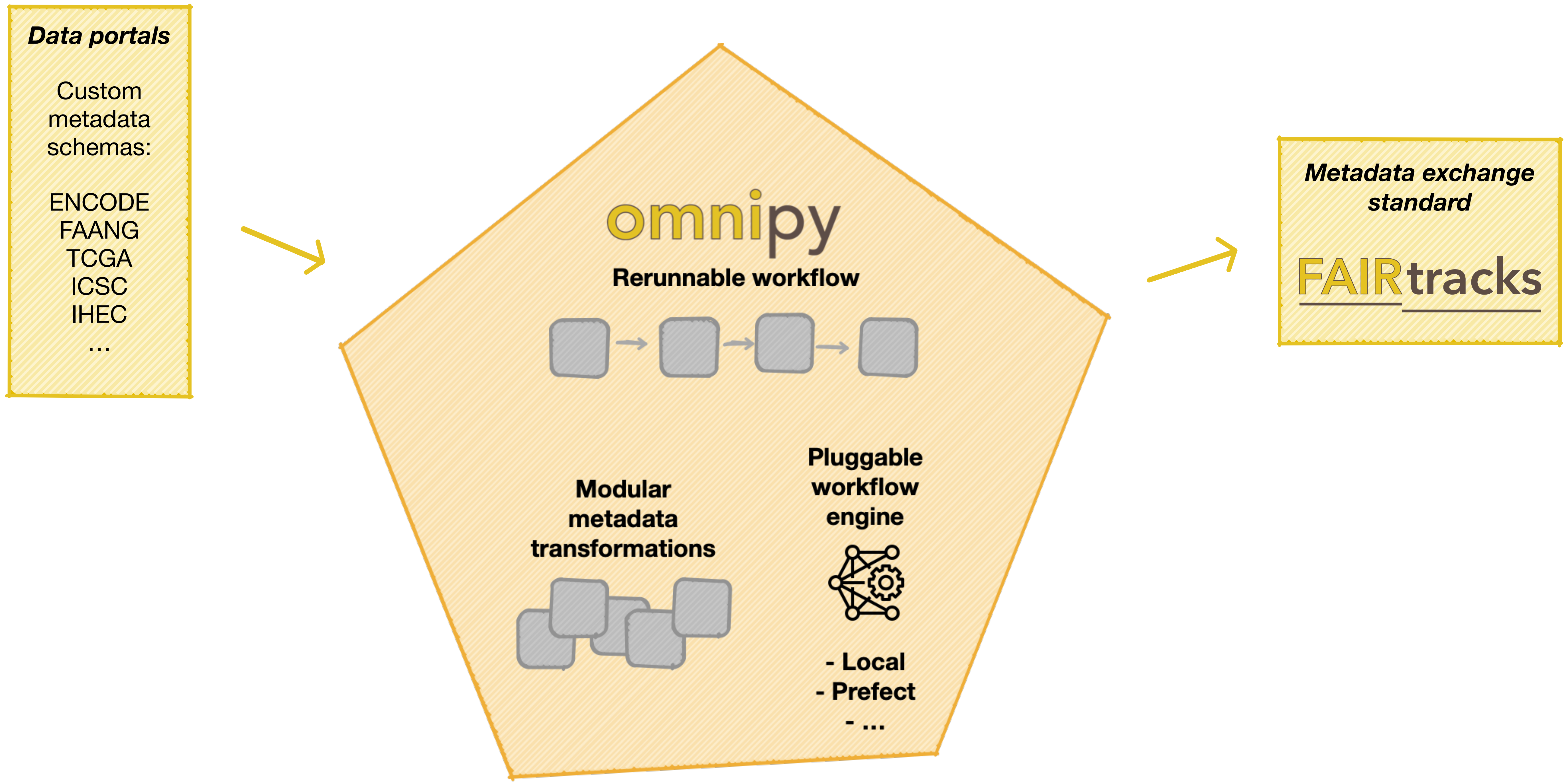
Custom metadata schemas:

ENCODE
FAANG
TCGA
ICSC
IHEC
…


uniFAIR
metadata
- Encode
- blueprint
- TCGA
Transform
How?
FAIRtracks standards compliant Metadata
Encode
Normalized (non-reduntant) Encode tables
ENCODE Model map
TCGA
Normalized (non-reduntant) TCGA tables
TCGA Model map
Raw FAIRtracks-mapped tables
Ontology mapping / conversion
Data cleanup
(Collapsing variations in naming)
- Required manual (OpenRefine?)
FAIRtracks Validation
Minimum FAIRtracks-compliant ENCODE tables
Optional ->
Augmented humanly readable FAIRtracks tables

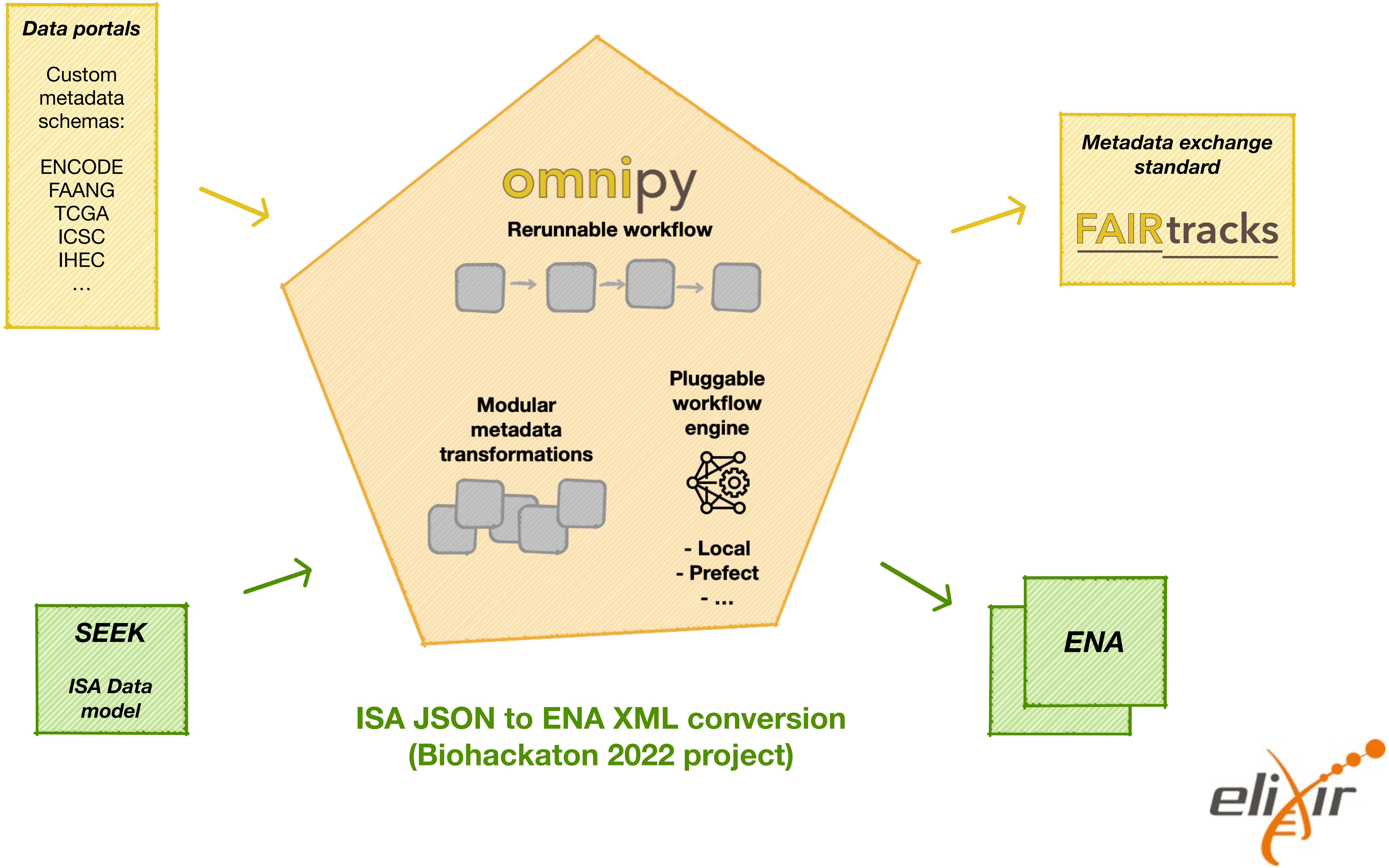
Metadata exchange standard

FAIRtracks

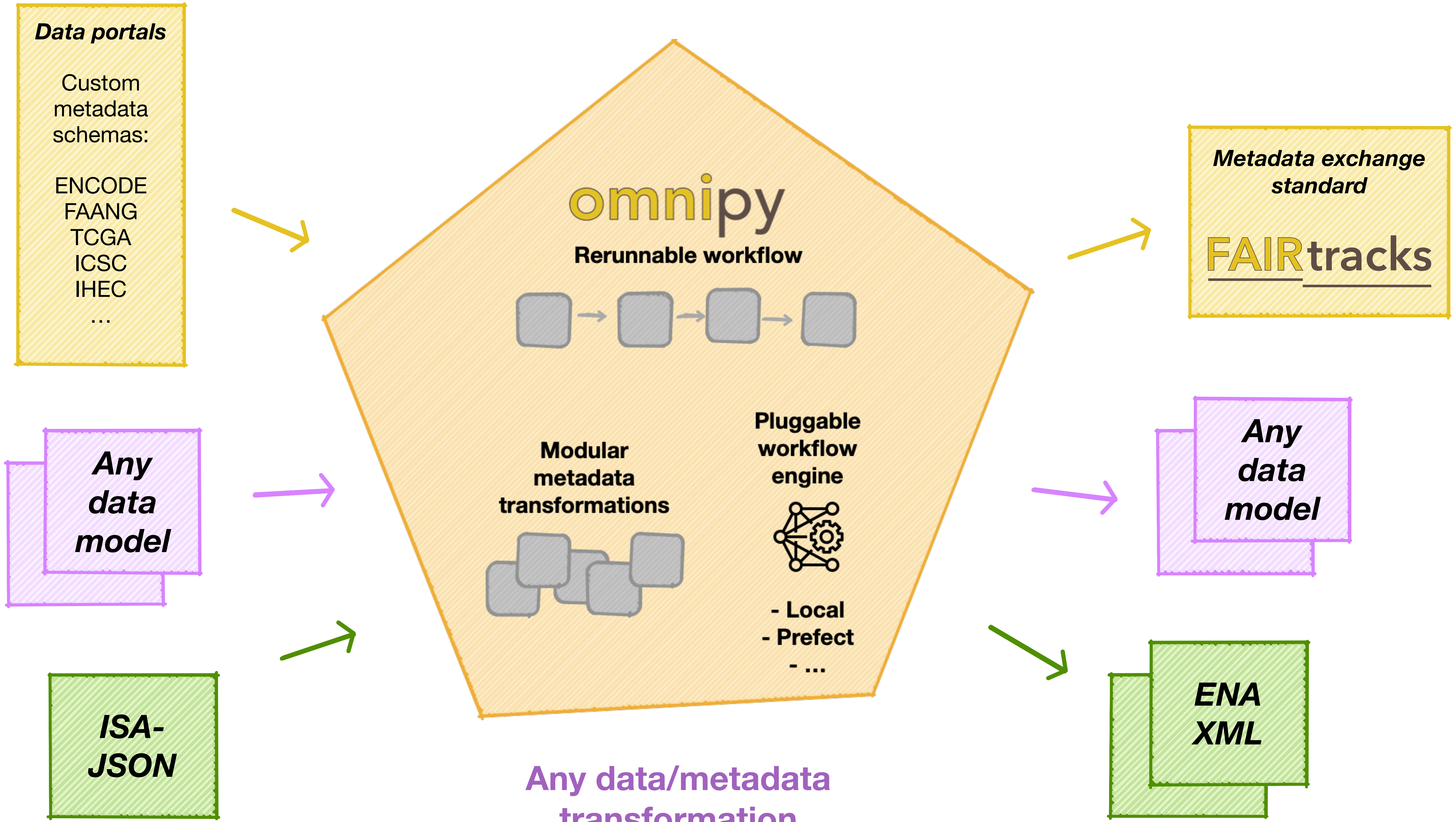


FAIRtracks

metadata ETL workflows



ISA JSON to ENA XML conversion (Biohackaton 2022 project)



Any data/metadata transformation workflow

Reusable metadata ETL modules

Extract

- Data repository API clients
- GA4GH Data Connect
- File importers (e.g. Excel)
- Relational database dumps
- Data repository API clients

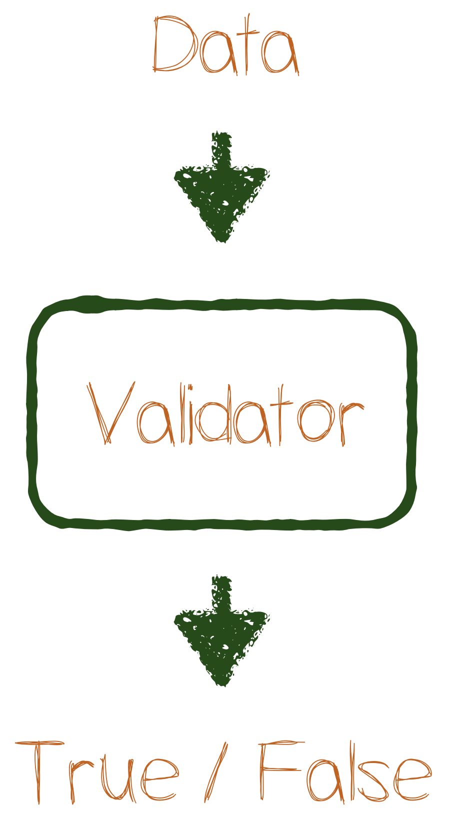
Transform

- JSON cleanup
- Database normalization (1NF, 2NF, 3NF)
- Data model mapping
- Table transformations
- Batch curation across records
- Ontology term mapping
- Ontology term conversion
- Validation towards destination schema

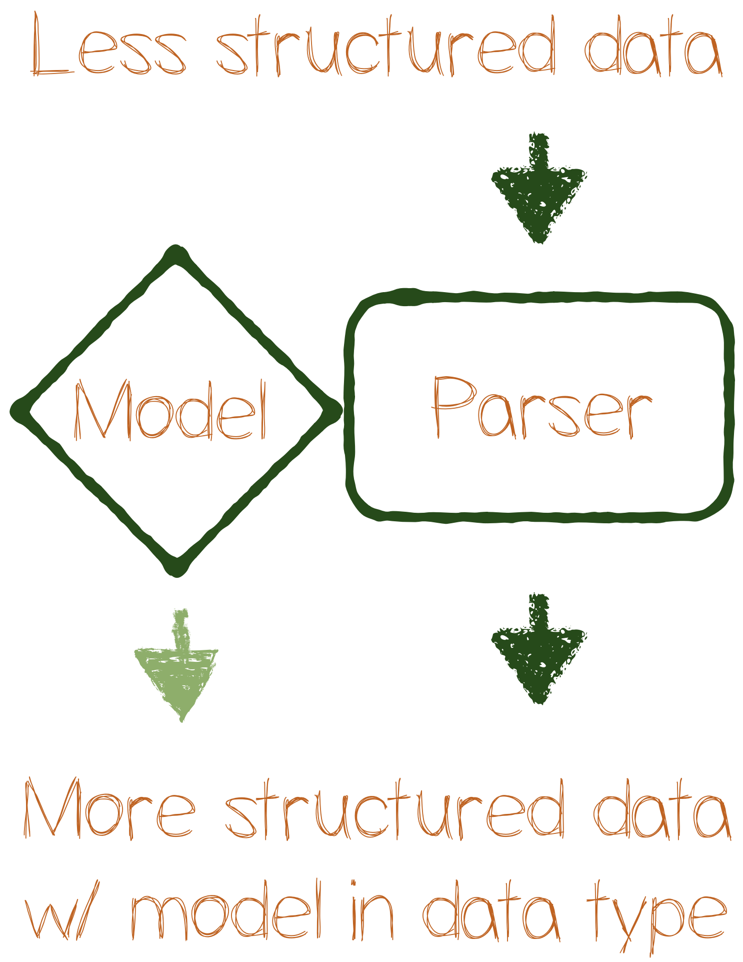
Load

- Data repository submission APIs
- File exporters (not Excel!)
- Relational database dumps
- GA4GH Data Connect

Parse, don't validate!



- Alexis King: "Parse, don’t validate" (2019 blog post)
    - Blog: https://lexi-lambda.github.io/blog/2019/11/05/parse-don-t-validate/
- Step-wise parsers munge the data to conform to schema/data model restrictions
- Allow for some variation in data input, with automatic type conversion
- Gradually improve structure of data

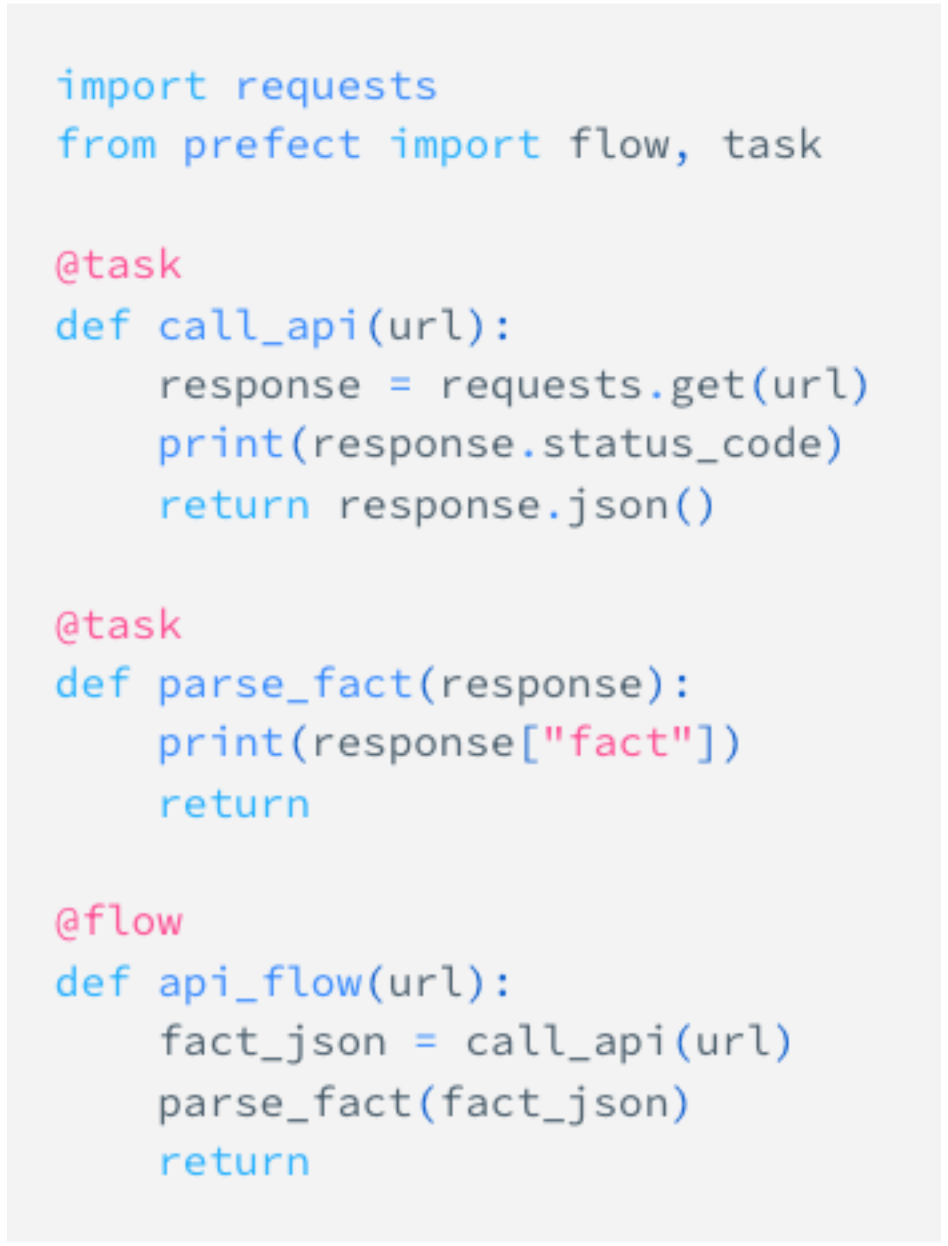


PREFECT v 2.0 - Orion

- Python-based dynamic orchestration engine
- Fully Open Source: *Apache 2.0 license*
- Dynamically registered, DAG-free workflows:
  - Supports *if/for/while* statements
  - Dynamic branching logic depending on runtime conditions
- Code as workflows:
  - No need to specify task and workflow parameters outside of normal code
  - Debug locally, run remotely using the same source code
  - Use tools you might already know, including:
    - Pydantic, FastAPI, RRule, Sqlite, asyncio, Dask, ...
- Rules engine:
  - State-based orchestration engine, with API separation
  - GUI Dashboard to administrer workflow runs
- A bunch of pre-built integrations:
  - data sources/destinations *(GitHub, PostgreSQL, Jupyter notebooks, etc.)*
  - deployment options *(Docker, Kubernetes, cloud storage, etc.)*

Simple Prefect workflow with two tasks

(from https://orion-docs.prefect.io/tutorials/first-steps/#run-a-basic-flow-with-tasks)



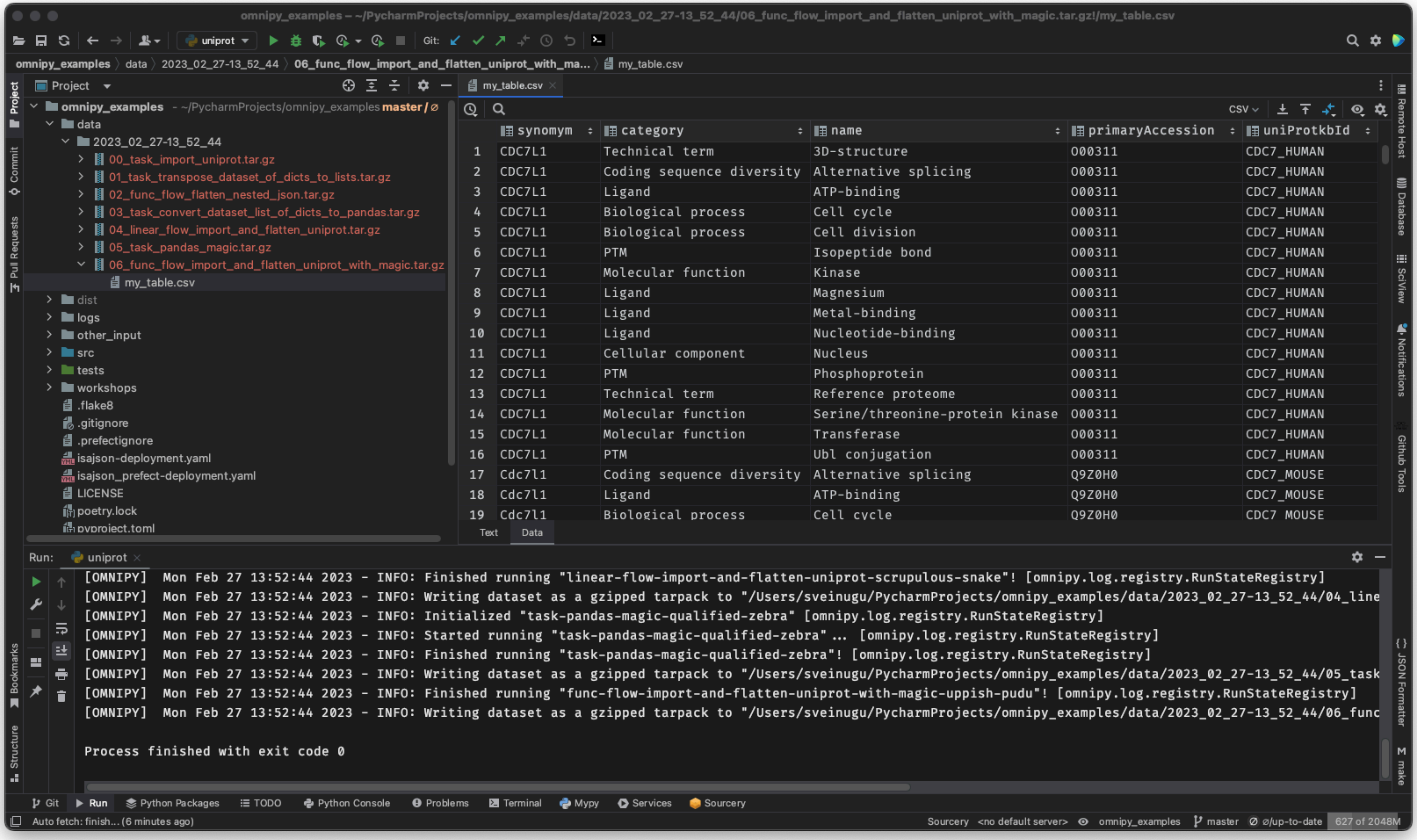
```python
import requests
from prefect import flow, task

@task
def call_api(url):
    response = requests.get(url)
    print(response.status_code)
    return response.json()

@task
def parse_fact(response):
    print(response["fact"])
    return

@flow
def api_flow(url):
    fact_json = call_api(url)
    parse_fact(fact_json)
    return
```

Develop, inspect and deploy directly from IDE



omnipy_examples – ~/PycharmProjects/omnipy_examples/data/2023_02_27-13_52_44/06_func_flow_import_and_flatten_uniprot_with_magic.tar.gz!/my_table.csv

omnipy_examples 〉 data 〉 2023_02_27-13_52_44 〉 06_func_flow_import_and_flatten_uniprot_with_ma... 〉 my_table.csv

```
omnipy_examples - ~/PycharmProjects/omnipy_examples master / ⌀
  data
    2023_02_27-13_52_44
      00_task_import_uniprot.tar.gz
      01_task_transpose_dataset_of_dicts_to_lists.tar.gz
      02_func_flow_flatten_nested_json.tar.gz
      03_task_convert_dataset_list_of_dicts_to_pandas.tar.gz
      04_linear_flow_import_and_flatten_uniprot.tar.gz
      05_task_pandas_magic.tar.gz
      06_func_flow_import_and_flatten_uniprot_with_magic.tar.gz
        my_table.csv
  dist
  logs
  other_input
  src
  tests
  workshops
  .flake8
  .gitignore
  .prefectignore
  isajson-deployment.yaml
  isajson_prefect-deployment.yaml
  LICENSE
  poetry.lock
  pyproject.toml
```

	synomym	category	name	primaryAccession	uniProtkbId
1	CDC7L1	Technical term	3D-structure	O00311	CDC7_HUMAN
2	CDC7L1	Coding sequence diversity	Alternative splicing	O00311	CDC7_HUMAN
3	CDC7L1	Ligand	ATP-binding	O00311	CDC7_HUMAN
4	CDC7L1	Biological process	Cell cycle	O00311	CDC7_HUMAN
5	CDC7L1	Biological process	Cell division	O00311	CDC7_HUMAN
6	CDC7L1	PTM	Isopeptide bond	O00311	CDC7_HUMAN
7	CDC7L1	Molecular function	Kinase	O00311	CDC7_HUMAN
8	CDC7L1	Ligand	Magnesium	O00311	CDC7_HUMAN
9	CDC7L1	Ligand	Metal-binding	O00311	CDC7_HUMAN
10	CDC7L1	Ligand	Nucleotide-binding	O00311	CDC7_HUMAN
11	CDC7L1	Cellular component	Nucleus	O00311	CDC7_HUMAN
12	CDC7L1	PTM	Phosphoprotein	O00311	CDC7_HUMAN
13	CDC7L1	Technical term	Reference proteome	O00311	CDC7_HUMAN
14	CDC7L1	Molecular function	Serine/threonine-protein kinase	O00311	CDC7_HUMAN
15	CDC7L1	Molecular function	Transferase	O00311	CDC7_HUMAN
16	CDC7L1	PTM	Ubl conjugation	O00311	CDC7_HUMAN
17	Cdc7l1	Coding sequence diversity	Alternative splicing	Q9Z0H0	CDC7_MOUSE
18	Cdc7l1	Ligand	ATP-binding	Q9Z0H0	CDC7_MOUSE
19	Cdc7l1	Biological process	Cell cycle	Q9Z0H0	CDC7_MOUSE

Run: uniprot

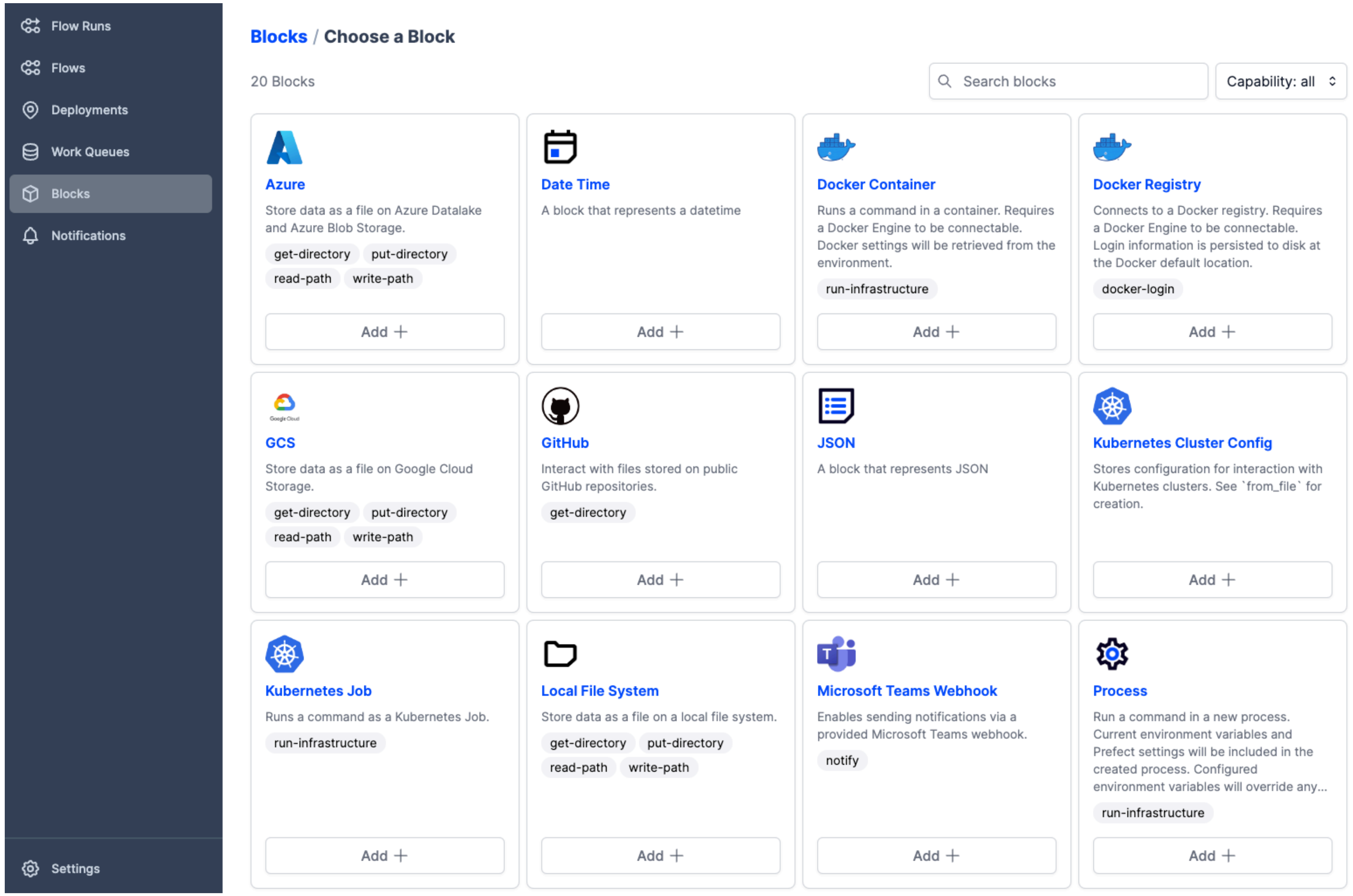
```
[OMNIPY]  Mon Feb 27 13:52:44 2023 - INFO: Finished running "linear-flow-import-and-flatten-uniprot-scrupulous-snake"! [omnipy.log.registry.RunStateRegistry]
[OMNIPY]  Mon Feb 27 13:52:44 2023 - INFO: Writing dataset as a gzipped tarpack to "/Users/sveinugu/PycharmProjects/omnipy_examples/data/2023_02_27-13_52_44/04_line
[OMNIPY]  Mon Feb 27 13:52:44 2023 - INFO: Initialized "task-pandas-magic-qualified-zebra" [omnipy.log.registry.RunStateRegistry]
[OMNIPY]  Mon Feb 27 13:52:44 2023 - INFO: Started running "task-pandas-magic-qualified-zebra" … [omnipy.log.registry.RunStateRegistry]
[OMNIPY]  Mon Feb 27 13:52:44 2023 - INFO: Finished running "task-pandas-magic-qualified-zebra"! [omnipy.log.registry.RunStateRegistry]
[OMNIPY]  Mon Feb 27 13:52:44 2023 - INFO: Writing dataset as a gzipped tarpack to "/Users/sveinugu/PycharmProjects/omnipy_examples/data/2023_02_27-13_52_44/05_task
[OMNIPY]  Mon Feb 27 13:52:44 2023 - INFO: Finished running "func-flow-import-and-flatten-uniprot-with-magic-uppish-pudu"! [omnipy.log.registry.RunStateRegistry]
[OMNIPY]  Mon Feb 27 13:52:44 2023 - INFO: Writing dataset as a gzipped tarpack to "/Users/sveinugu/PycharmProjects/omnipy_examples/data/2023_02_27-13_52_44/06_func

Process finished with exit code 0
```

Auto fetch: finish... (6 minutes ago)

Sourcery <no default server> omnipy_examples master ⌀/up-to-date 627 of 2048M

Prefect orchestration GUI for local and remote deployment



Blocks / **Choose a Block**

20 Blocks

Search blocks

Capability: all

Azure
Store data as a file on Azure Datalake and Azure Blob Storage.
get-directory put-directory read-path write-path
Add +

Date Time
A block that represents a datetime
Add +

Docker Container
Runs a command in a container. Requires a Docker Engine to be connectable. Docker settings will be retrieved from the environment.
run-infrastructure
Add +

Docker Registry
Connects to a Docker registry. Requires a Docker Engine to be connectable. Login information is persisted to disk at the Docker default location.
docker-login
Add +

GCS
Store data as a file on Google Cloud Storage.
get-directory put-directory read-path write-path
Add +

GitHub
Interact with files stored on public GitHub repositories.
get-directory
Add +

JSON
A block that represents JSON
Add +

Kubernetes Cluster Config
Stores configuration for interaction with Kubernetes clusters. See `from_file` for creation.
Add +

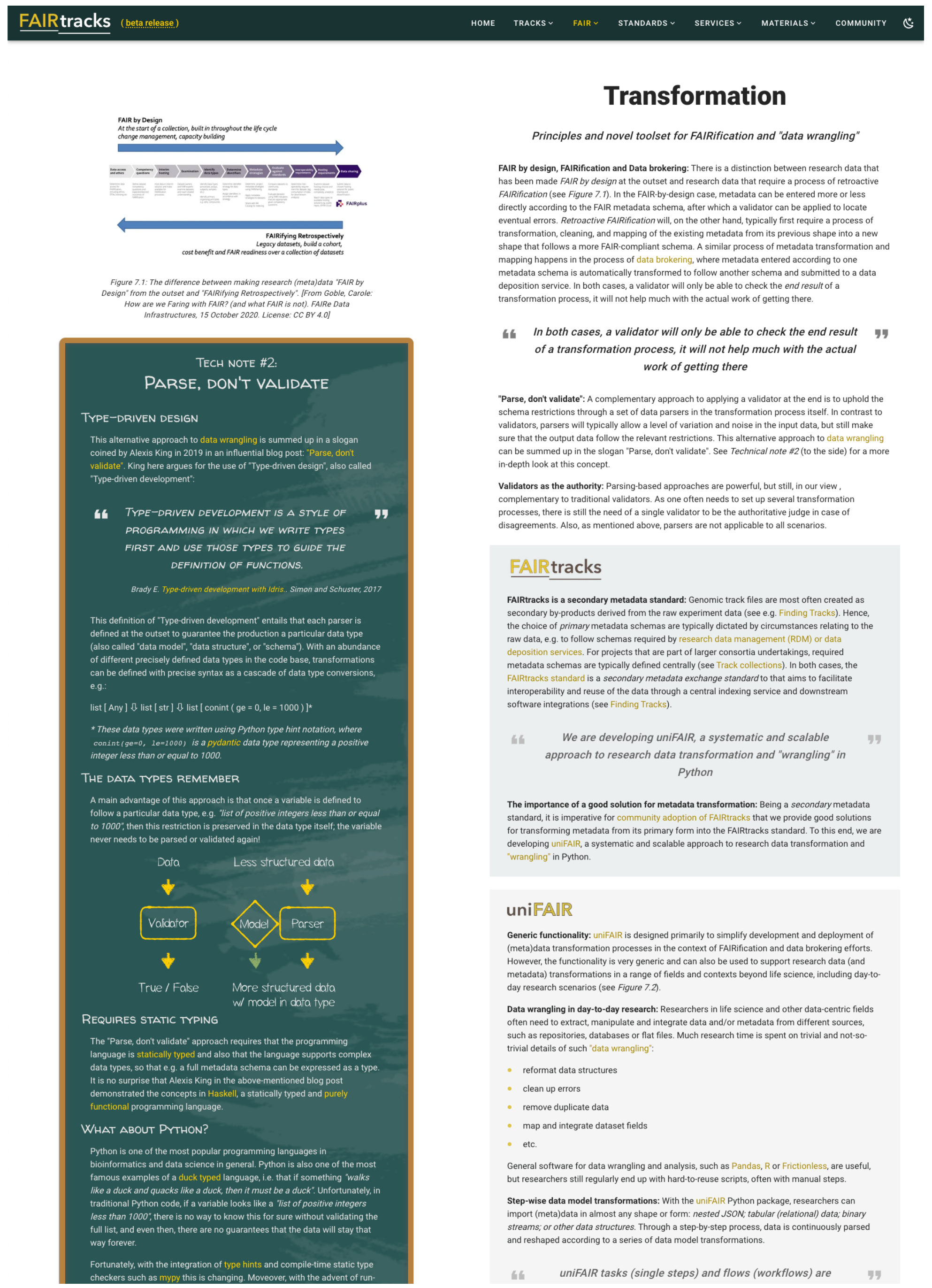
Kubernetes Job
Runs a command as a Kubernetes Job.
run-infrastructure
Add +

Local File System
Store data as a file on a local file system.
get-directory put-directory read-path write-path
Add +

Microsoft Teams Webhook
Enables sending notifications via a provided Microsoft Teams webhook.
notify
Add +

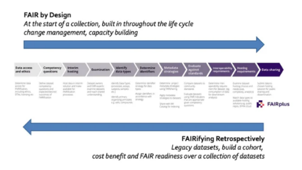
Process
Run a command in a new process. Current environment variables and Prefect settings will be included in the created process. Configured environment variables will override any...
run-infrastructure
Add +

More information on Omnipy



FAIRtracks (beta release)

HOME TRACKS FAIR STANDARDS SERVICES MATERIALS COMMUNITY



FAIR by Design
At the start of a collection, built in throughout the life cycle
change management, capacity building
FAIRplus
FAIRifying Retrospectively
Legacy datasets, build a cohort,
cost benefit and FAIR readiness over a collection of datasets

Figure 7.1: The difference between making research (meta)data "FAIR by Design" from the outset and "FAIRifying Retrospectively". [From Goble, Carole: How are we Faring with FAIR? (and what FAIR is not). FAIRe Data Infrastructures, 15 October 2020. License: CC BY 4.0]

Tech note #2:

Parse, don't validate

Type-driven design

This alternative approach to data wrangling is summed up in a slogan coined by Alexis King in 2019 in an influential blog post: "Parse, don't validate". King here argues for the use of "Type-driven design", also called "Type-driven development":

> *Type-driven development is a style of programming in which we write types first and use those types to guide the definition of functions.*
>
> *Brady E. Type-driven development with Idris.. Simon and Schuster, 2017*

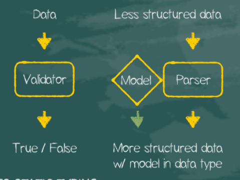
This definition of "Type-driven development" entails that each parser is defined at the outset to guarantee the production a particular data type (also called "data model", "data structure", or "schema"). With an abundance of different precisely defined data types in the code base, transformations can be defined with precise syntax as a cascade of data type conversions, e.g.:

list [Any] ⇩ list [str] ⇩ list [conint (ge = 0, le = 1000)]*

** These data types were written using Python type hint notation, where* `conint(ge=0, le=1000)` *is a pydantic data type representing a positive integer less than or equal to 1000.*

The data types remember

A main advantage of this approach is that once a variable is defined to follow a particular data type, e.g. *"list of positive integers less than or equal to 1000"*, then this restriction is preserved in the data type itself; the variable never needs to be parsed or validated again!



Requires static typing

The "Parse, don't validate" approach requires that the programming language is statically typed and also that the language supports complex data types, so that e.g. a full metadata schema can be expressed as a type. It is no surprise that Alexis King in the above-mentioned blog post demonstrated the concepts in Haskell, a statically typed and purely functional programming language.

What about Python?

Python is one of the most popular programming languages in bioinformatics and data science in general. Python is also one of the most famous examples of a duck typed language, i.e. that if something *"walks like a duck and quacks like a duck, then it must be a duck"*. Unfortunately, in traditional Python code, if a variable looks like a *"list of positive integers less than 1000"*, there is no way to know this for sure without validating the full list, and even then, there are no guarantees that the data will stay that way forever.

Fortunately, with the integration of type hints and compile-time static type checkers such as mypy this is changing. Moveover, with the advent of run-

Transformation

Principles and novel toolset for FAIRification and "data wrangling"

FAIR by design, FAIRification and Data brokering: There is a distinction between research data that has been made *FAIR by design* at the outset and research data that require a process of retroactive *FAIRification* (see *Figure 7.1*). In the FAIR-by-design case, metadata can be entered more or less directly according to the FAIR metadata schema, after which a validator can be applied to locate eventual errors. *Retroactive FAIRification* will, on the other hand, typically first require a process of transformation, cleaning, and mapping of the existing metadata from its previous shape into a new shape that follows a more FAIR-compliant schema. A similar process of metadata transformation and mapping happens in the process of data brokering, where metadata entered according to one metadata schema is automatically transformed to follow another schema and submitted to a data deposition service. In both cases, a validator will only be able to check the *end result* of a transformation process, it will not help much with the actual work of getting there.

> *In both cases, a validator will only be able to check the end result of a transformation process, it will not help much with the actual work of getting there*

"Parse, don't validate": A complementary approach to applying a validator at the end is to uphold the schema restrictions through a set of data parsers in the transformation process itself. In contrast to validators, parsers will typically allow a level of variation and noise in the input data, but still make sure that the output data follow the relevant restrictions. This alternative approach to data wrangling can be summed up in the slogan "Parse, don't validate". See *Technical note #2* (to the side) for a more in-depth look at this concept.

Validators as the authority: Parsing-based approaches are powerful, but still, in our view , complementary to traditional validators. As one often needs to set up several transformation processes, there is still the need of a single validator to be the authoritative judge in case of disagreements. Also, as mentioned above, parsers are not applicable to all scenarios.

FAIRtracks

FAIRtracks is a secondary metadata standard: Genomic track files are most often created as secondary by-products derived from the raw experiment data (see e.g. Finding Tracks). Hence, the choice of *primary* metadata schemas are typically dictated by circumstances relating to the raw data, e.g. to follow schemas required by research data management (RDM) or data deposition services. For projects that are part of larger consortia undertakings, required metadata schemas are typically defined centrally (see Track collections). In both cases, the FAIRtracks standard is a *secondary metadata exchange standard* to that aims to facilitate interoperability and reuse of the data through a central indexing service and downstream software integrations (see Finding Tracks).

> *We are developing uniFAIR, a systematic and scalable approach to research data transformation and "wrangling" in Python*

The importance of a good solution for metadata transformation: Being a *secondary* metadata standard, it is imperative for community adoption of FAIRtracks that we provide good solutions for transforming metadata from its primary form into the FAIRtracks standard. To this end, we are developing uniFAIR, a systematic and scalable approach to research data transformation and "wrangling" in Python.

uniFAIR

Generic functionality: uniFAIR is designed primarily to simplify development and deployment of (meta)data transformation processes in the context of FAIRification and data brokering efforts. However, the functionality is very generic and can also be used to support research data (and metadata) transformations in a range of fields and contexts beyond life science, including day-to-day research scenarios (see *Figure 7.2*).

Data wrangling in day-to-day research: Researchers in life science and other data-centric fields often need to extract, manipulate and integrate data and/or metadata from different sources, such as repositories, databases or flat files. Much research time is spent on trivial and not-so-trivial details of such "data wrangling":

- reformat data structures
- clean up errors
- remove duplicate data
- map and integrate dataset fields
- etc.

General software for data wrangling and analysis, such as Pandas, R or Frictionless, are useful, but researchers still regularly end up with hard-to-reuse scripts, often with manual steps.

Step-wise data model transformations: With the uniFAIR Python package, researchers can import (meta)data in almost any shape or form: *nested JSON; tabular (relational) data; binary streams; or other data structures*. Through a step-by-step process, data is continuously parsed and reshaped according to a series of data model transformations.

> *uniFAIR tasks (single steps) and flows (workflows) are*

- Check out the newly launched redesign of the FAIRtracks website:
  - https://fairtracks.net
- Specifically:
  - https://fairtracks.net/fair/#fair-07-transformation
- GitHub repo:
  - https://github.com/fairtracks/omnipy

Collaboration

We have limited development resources and are aiming for FAIRtracks to become a community endeavor!

We are planning a BioHackathon project on expanding to Omnipy and are looking for collaborators, e.g. to integrate other resources or standards into Omnipy or to use Omnipy to set up metadata transformation.

We are interested in any types of contributions & collaborations!

Please contact us at: fairtracks@elixir.no

Acknowledgements

FAIRtracks

S. Gundersen and the FAIRtracks team:

+

EMBL-EBI, Hinxton, UK

ELIXIR Norway,
Centre for Bioinformatics, UiO

ELIXIR Spain,
Barcelona Supercomputing Centre

ELIXIR Norway,
NTNU & UiB

+

+

Jeanne Cheneby,
Pável Vázquez

Acknowledgements

Spanish National Bioinformatics Institute

With funding from The Research Council of Norway

NTNU Norwegian University of Science and Technology

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EMBL-EBI

UiO University of Oslo