



| The European Synchrotron

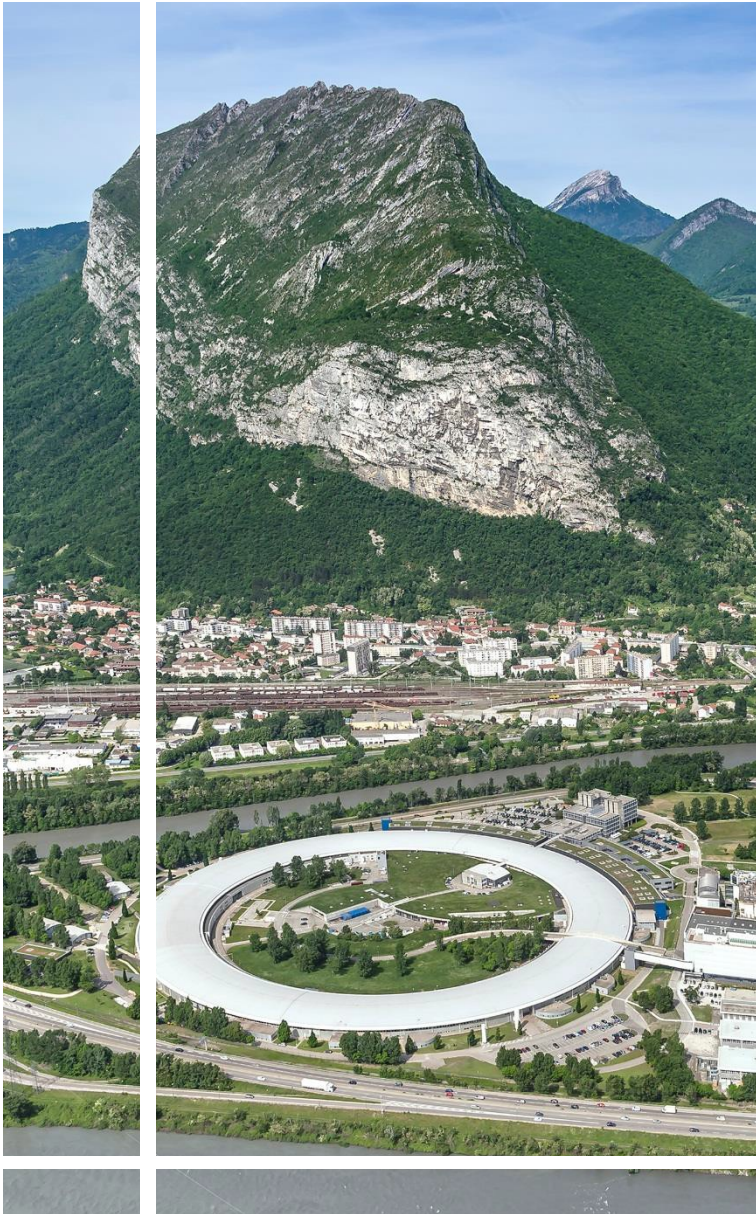


EWOKS (ESRF Workflow System) A meta workflow system

Wout De Nolf
ESRF (Data Automation Unit)



STREAMLINE has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 870313



Goal of this presentation

Demonstrate how the ESRF handles increasing data processing challenges with a “meta workflow” approach.



What are the data processing challenges?

1992-2018

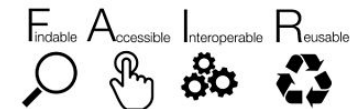


Produce Unique Data

2020-present



Produce Unique Results

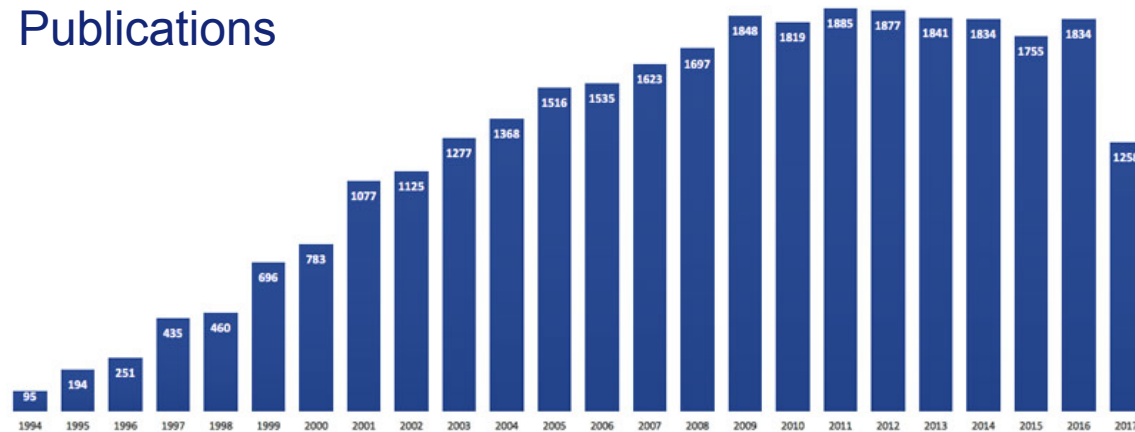




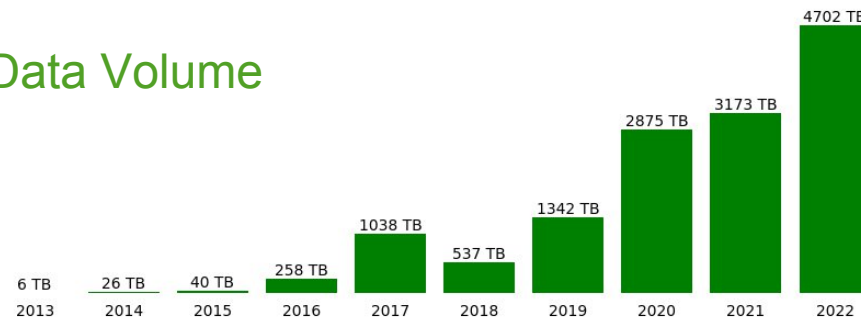
What are the data processing challenges?

⚠️ Plateau in scientific output while data volume increases ⚠️

Publications

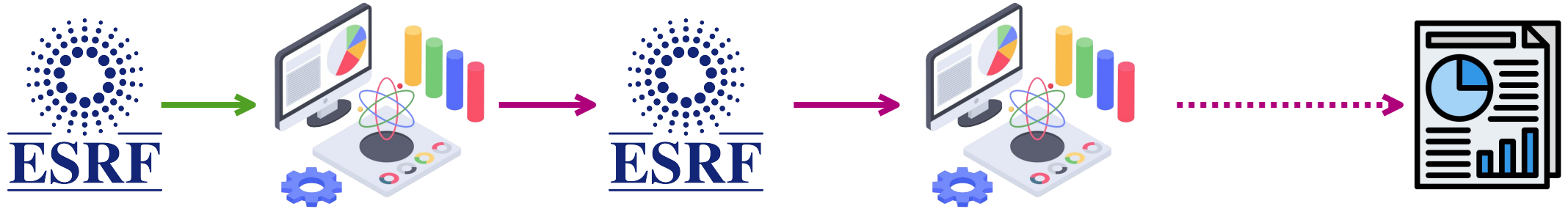


FAIR Data Volume



First Experiment

Publication



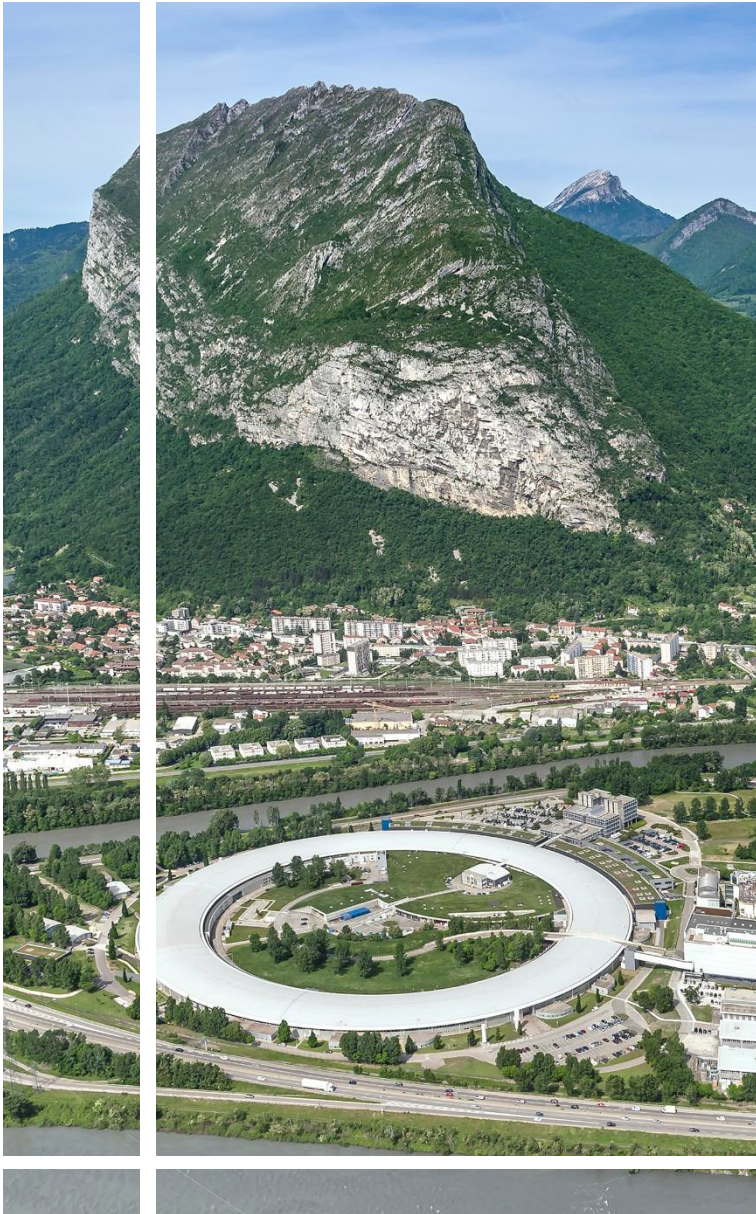
Timescale of months at best, often one year or more

Steep learning curve for new users from a wide range of scientific domains
Growing the synchrotron community has reached its limits



- ⌘ **Visualize the experiment** for humans, not the way X-ray detectors see it
- ⌘ Produce **domain specific results** when possible, X-ray results when necessary
- ➡ Integrate scientific software in the experiment (acquisition control system and data portal)

DATA PROCESSING CHALLENGES



- 🔑 **Visualize the experiment** for humans
- 🔑 **Produce domain specific results** when possible
- ➡ Integrate scientific software in the experiment

Attempts at the ESRF to do this are domain or beamline specific.

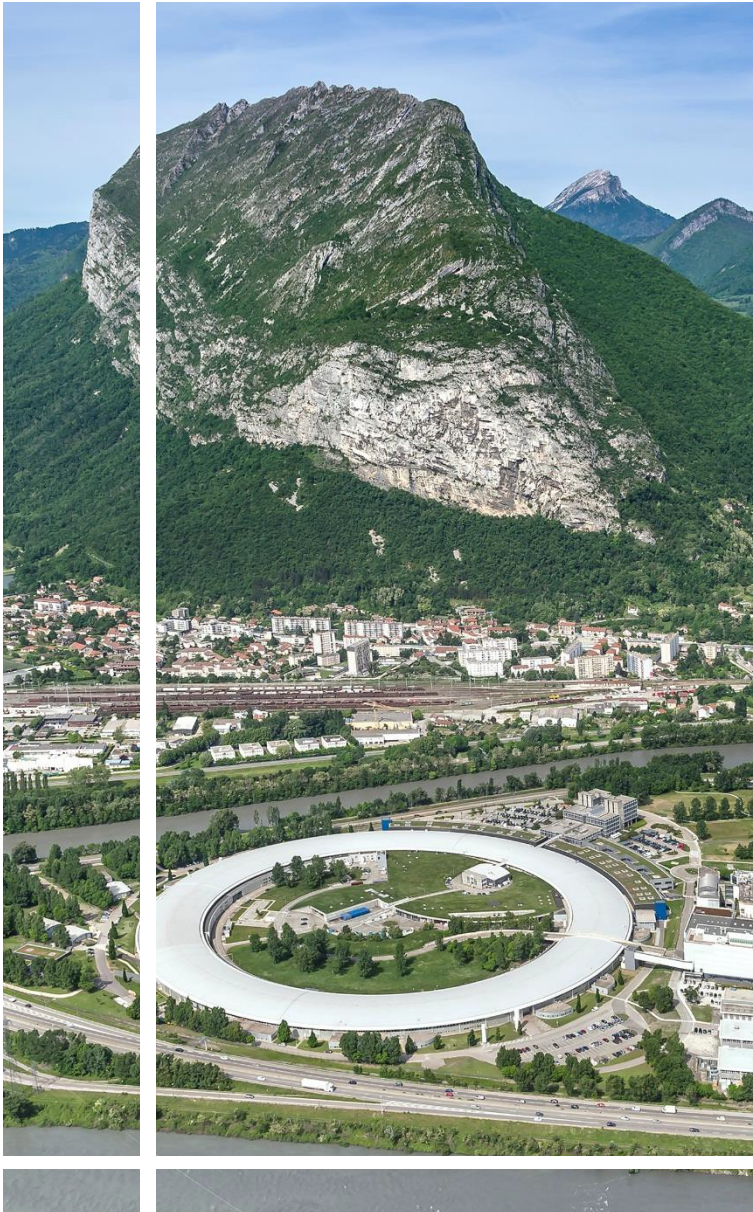
For example the *Structural Biology Services* at the ESRF (e.g. macromolecular crystallography) allow for completely automated experiments.

- custom designed workflow system that orchestrates the experiment and data analysis
- ISPyB based Laboratory Information Management System (LIMS) combining sample tracking and experiment reporting

Other attempts are all beamline specific with custom software.

- ➡ **reinvent the wheel for every beamline**
- ➡ **no shared tools or technologies**
- ➡ **very labor intensive**
- ➡ **relies on a single expert per project (knowledge leaves when they leave)**
- ➡ **no professional devops (users cannot run the software at home)**

DATA PROCESSING CHALLENGES



- 🔑 **Visualize the experiment** for humans
- 🔑 Produce **domain specific results** when possible
- ➡ Integrate scientific software in the experiment

We needed a domain and beamline agnostic solution with the following properties:

- **easy to install** (runs everywhere from laptops to clusters)
- **reproducible** data processing (*executable data provenance document* saved with the results)
- **modular** (share common data processing steps, e.g. SAXS - WAXS, XRF mapping - XRF tomography, needs to evolve with the state-of-the-art)
- **interactive** vs. non-interactive
- different types of **interfaces** for humans and machines (GUI with plots and buttons, job scheduling on a cluster, web service)
- scientific software (often in python) needs to be **integrated**, not re-written
- can be provided as **service** of the facility maintained by a group, not reliant on individual experts

DATA PROCESSING CHALLENGES



- 🔑 **Visualize the experiment** for humans
- 🔑 Produce **domain specific results** when possible
- ➡ Integrate scientific software in the experiment

A **workflow-based solution** was chosen for the ESRF because

- A workflow is an **executable data provenance document**.
- It encapsulated the decisions taken by an **expert** so the learning curve for non-experts is less steep: “how do I use this workflow” not “how do I process this diffraction data”.
- A workflow can be **reused** to process other data (the “R” in FAIR).
- Workflows can be developed by experts and **maintained and deployed** by the facility.

There are hundreds (yes, hundreds) of workflow systems



None have all features

- Graphical interface for desktop
- Interactive execution
- Parallel execution
- Distribution on a compute cluster
- Support for loops
- Python can be easily integrated
- Web service to manage and execute workflows
- Workflow execution without the need for infrastructure

Meta workflow system: isolate the workflow definition and implementation from the workflow system



EWOKS
ESRF Workflow System
<https://ewoks.esrf.fr>



DAGSTER



- Project started in July 2021. Stable since January 2023.
- 6 core developers in 2024
- Ewoks workflows are published under the FAIR principles so the ESRF is committed to long term support.
- Despite the name, nothing in the design makes it ESRF specific.
- MIT License
- Workflow systems currently supports:
 - Orange: desktop GUI
 - Dask: parallel + cluster
 - Pypushflow: parallel + loops

EWOKS tasks to be used in workflows are reusable and searchable

<https://ewoks.esrf.fr>

ESRF Workflow System Getting started Tasks catalog Related projects Press 🔍 Search Ctrl + K 🗂️

🏠 > Tasks catalog

Tasks catalog

This page lists the tasks provided by the *ewoksapps*. Each of these tasks can be used in an Ewoks workflow.

🔔 25 beamlines use Ewoks to process their data!

Discover 362 workflow tasks below or use the search box

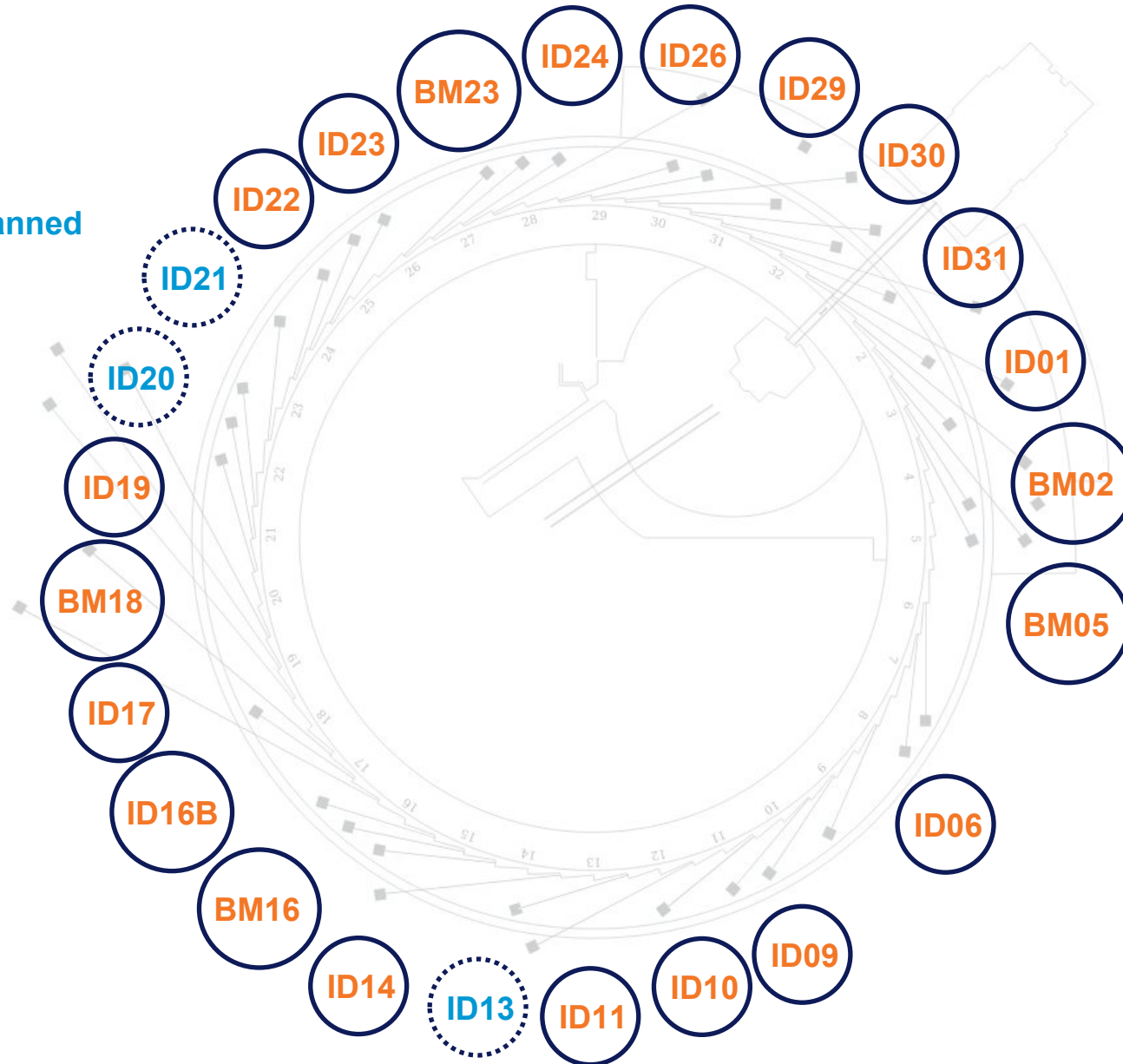
Tomography 🌿 BM05, BM18, ID11, ID16B, ID17, ID19 ⚙️ 38 tasks	SAXS/WAXS 🌿 BM02, ID09, ID11, ID16B, ID31 ⚙️ 20 tasks	Spectroscopy 🌿 BM23, ID24 ⚙️ 20 tasks
Fluorescence 🌿 ID16b, ID21 ⚙️ 11 tasks	Dark-field Microscopy 🌿 ID06, ID11 ⚙️ 16 tasks	Imaging 🌿 ID16b, ID21 ⚙️ 4 tasks



BM/ID In production



BM/ID In development/planned



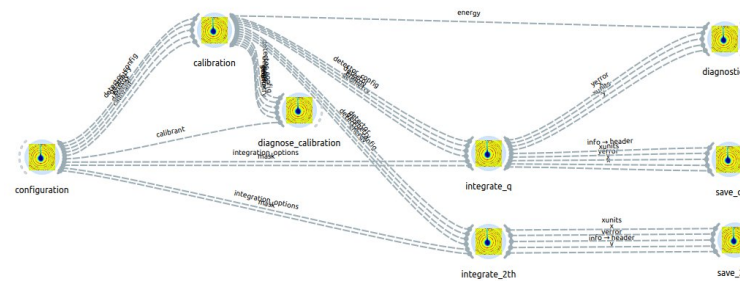
Typical setup for online data processing at the ESRF

Acquisition control system
(Bliss, Daiquiri, MxCube)



Execute EWOKS workflow on

- local machines (immediate feed)
- compute cluster



Visualization

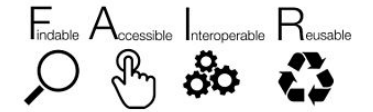


Persist result for further analysis

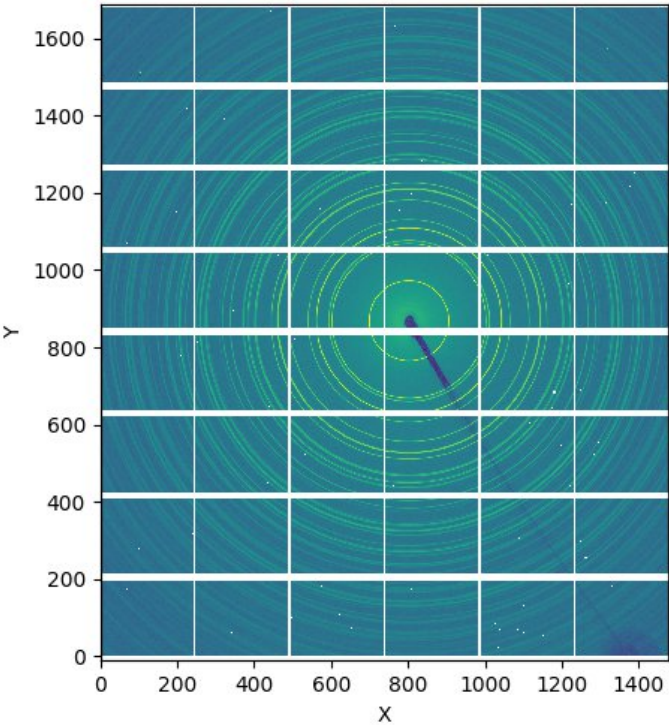


Data portal
(FAIR results including the workflows with DOI for publication)

Diffraction (SAXS/WAXS): convert synchrotron data to traditional power diffractograms



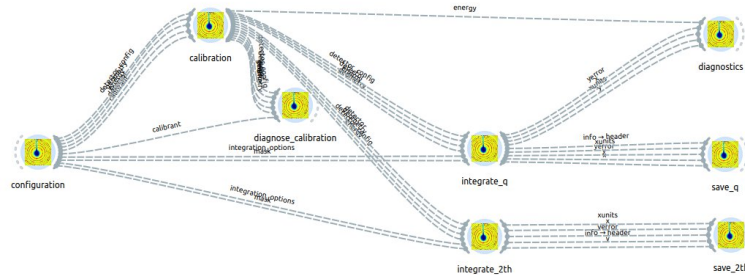
/1.1/measurement/p3[0, :, :]



- images + metadata (flux, energy, geometry)
- calibration (pyFAI)
- integration (pyFAI)
- save (HDF5)



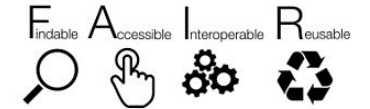
Dataset	0001	Distance	-340.00
Start	26/07/2023 13:28:48	Energy	75.00
End	26/07/2023 13:28:57	Vibration	40.0 %
Exp. Time	1 s		



Further analysis to be brought online in the future



Diffraction (macromolecular crystallography): Automatic structure solution from single-crystal diffraction



06/03/2024 11:48:26 MXPress1

Summary Beamline Parameters Acquisitions 4 Sample Autoprocessing 7 Workflow 5 Phasing 10

Path: /data/fd30b/inhouse/opid30b/20240306/RAW_DATA/Sample-8:2:16/run_03_MXPress1/run_03_05_datacollection

Protein:
Sample:
Prefix: opid30b_3
Run: 5
Images (Total): 900 (2492)
Transmission: 7.68326 %

Best result
from EDNA_proc

P 3 1 2 1	Compl.	Res.	Rmerge	I/Sigma
Inner	99.9%	43.2 - 5.1	4.2	41.4
Outer	100.0%	1.4 - 1.3	131.1	1.4
Overall	100.0%	43.2 - 1.3	6.4	14.6

Trigonal system (P3121)

a=b	c
54.5 Å	107.4 Å

MR phasing ?

MR Molecular_Replacement_from_cell P 3 1 2 1

map 1 level = 0.9534 e/Å³ (2.34 rmsd)

PBD: refined.pdb fullscreen unload

Automatic MR appears to have worked with the space group P3121

Dynamic aperture set to 30 μm 180.0 degree data collection with resolution 1.30 Å from characterisation.

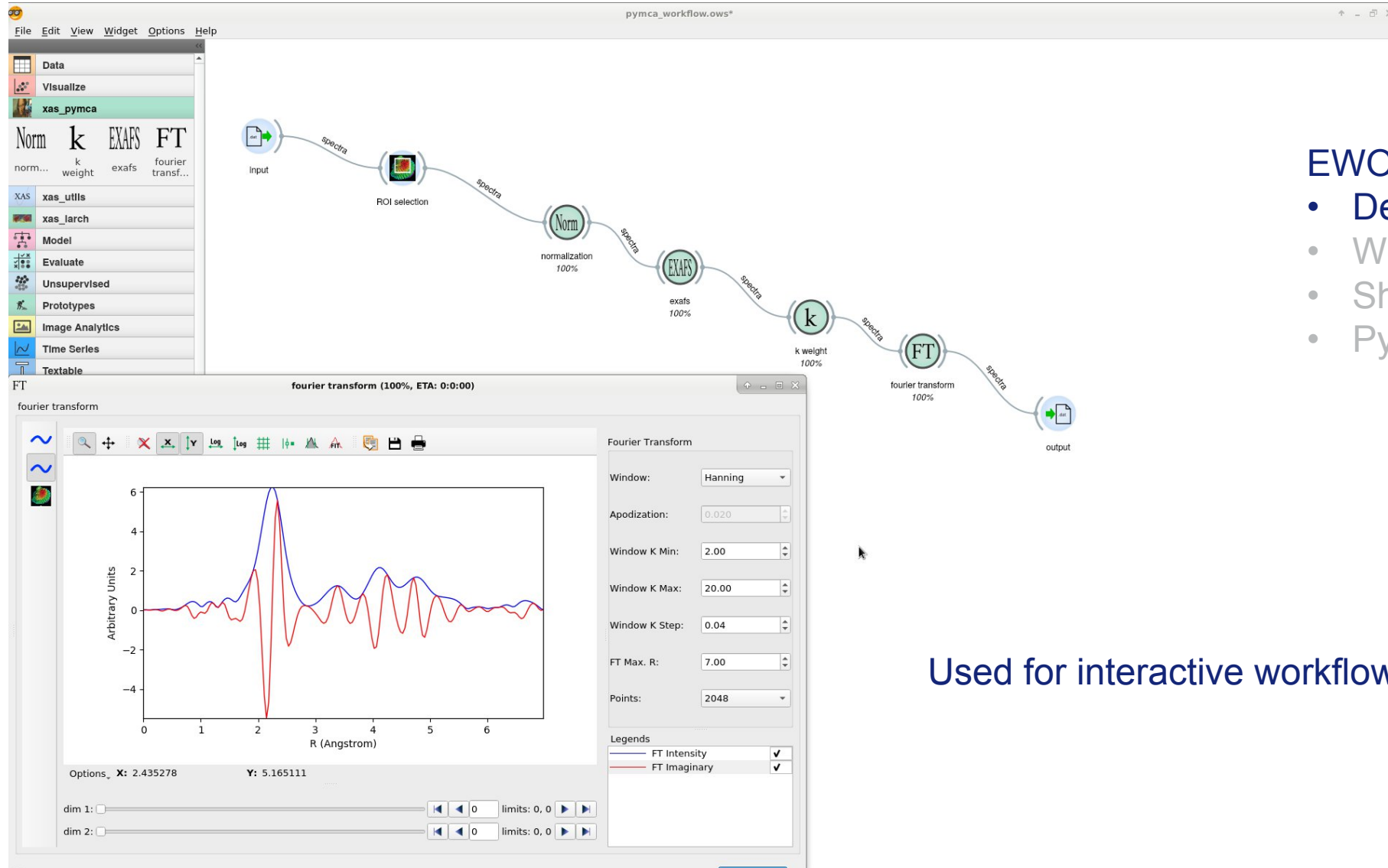
autoPROC autoPROC_stارانiso EDNA_proc grenades_CODGAS grenades_fastproc XDSAPP XIA2_DIALS

Example where the complete analysis was brought online (completely automated with diagnostics for validation)



To conclude an overview of all EWOKS interfaces and why they exist

- Desktop (interactive workflows)
- Web (workflows as a service)
- Shell (headless execution)
- Python (integration for developers)



EWOKS interfaces

- Desktop
- Web
- Shell
- Python

Used for interactive workflows

EwokWeb Edit Monitor Sum_then_integrate_with_saving

+ DISCOVER TASKS

- ewokscore
- ewoksrpd
- ewoksndreg
- ewoksfluo
- General

PyFaiConfig → Integrate1D

SumBlissScan Images → Integrate1D

Integrate1D → SaveNexusInt egrated

Integrate1D → SaveAsciiPatt ern1D

Data Mapping

Source	Target
x	radial
y	intensity
xunits	radial_units
yerror	intensity_error
info	info

Conditions

Output	Type	Value

EWOKS interfaces

- Desktop
- **Web**
- Shell
- Python

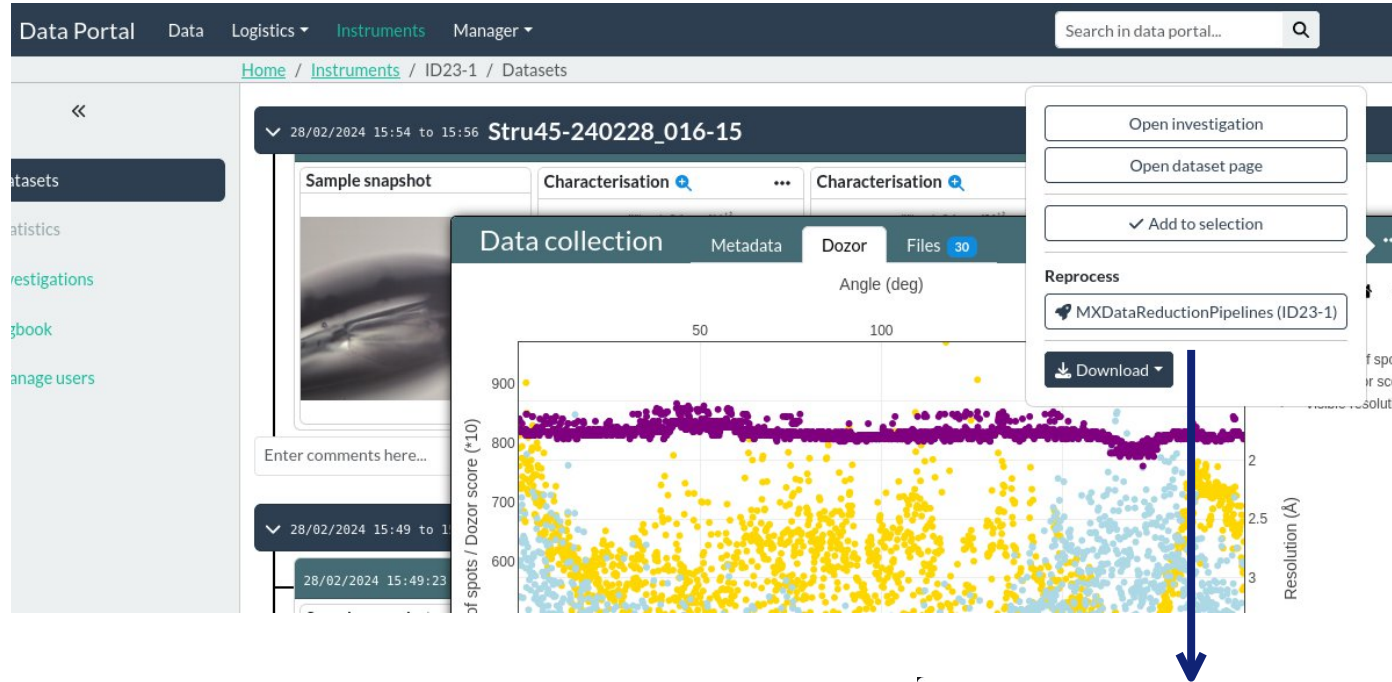
EwokWeb Edit Monitor

Executed workflows

- demo Job id: 17493721-305e-4c00-93fa-900d962e8040
29 seconds ago
Less than one second
Success
- demo Job id: e82a81bd-31ac-4431-b08c-1d3239d15cd1
30 seconds ago
Less than one second
Success
- demo Job id: 07bd15cd-e531-42fb-aebd-445da1fc5361
31 seconds ago
Less than one second
Success
- demo Job id: 58cf8483-4e8c-48e7-8221-1135af7f2301
30 seconds ago
Less than one second
Success
- demo Job id: e549647a-3502-4121-a0b4-e201a1bc1ad6
31 seconds ago
Less than one second
Success

Used to visualize workflows that don't have graphical components

Standalone + frontend (similar to Jupyter notebooks)



EWOKS interfaces

- Desktop
- **Web**
- Shell
- Python

Web service used by other web services (e.g. ESRF data portal)

Start reprocess

Demo POC

This is a simple example of Reprocessing by using Ewoks v2.0

Pipeline

- EDNA_proc
- autoPROC
- XIA2_DIALS
- grenades_fastproc

```
Terminal
(py38) denolf@lindenolf:~$ ewoks execute workflow.json -p a=100 --outputs all
#####
# Execute workflow 'workflow.json'
#####

RESULTS:
{'task0': {'sum': 3},
 'task1': {'result': 3},
 'task2': {'result': 100},
 'task3': {'result': 6},
 'task4': {'result': 104},
 'task5': {'result': 110},
 'task6': {'result': 116}}

FINISHED

(py38) denolf@lindenolf:~$
```

EWOKS interfaces

- Desktop
- Web
- **Shell**
- Python

Used for headless execution

```

from ewokscore import Task
from ewokscore import execute_graph

# Implement a workflow task
class SumTask(
    Task, input_names=["a"], optional_input_names=["b"], output_names=["result"]
):
    def run(self):
        result = self.inputs.a
        if self.inputs.b:
            result += self.inputs.b
        self.outputs.result = result

# Define a workflow with default inputs
nodes = [
    {
        "id": "task1",
        "task_type": "class",
        "task_identifier": "__main__.SumTask",
        "default_inputs": [{"name": "a", "value": 1}],
    },
    {
        "id": "task2",
        "task_type": "class",
        "task_identifier": "__main__.SumTask",
        "default_inputs": [{"name": "b", "value": 1}],
    },
    {
        "id": "task3",
        "task_type": "class",
        "task_identifier": "__main__.SumTask",
        "default_inputs": [{"name": "b", "value": 1}],
    },
]
links = [
    {
        "source": "task1",
        "target": "task2",
        "data_mapping": [{"source_output": "result", "target_input": "a"}],
    },
    {
        "source": "task2",
        "target": "task3",
        "data_mapping": [{"source_output": "result", "target_input": "a"}],
    },
]
workflow = {"graph": {"id": "testworkflow"}, "nodes": nodes, "links": links}

# Define task inputs
inputs = [{"id": "task1", "name": "a", "value": 10}]

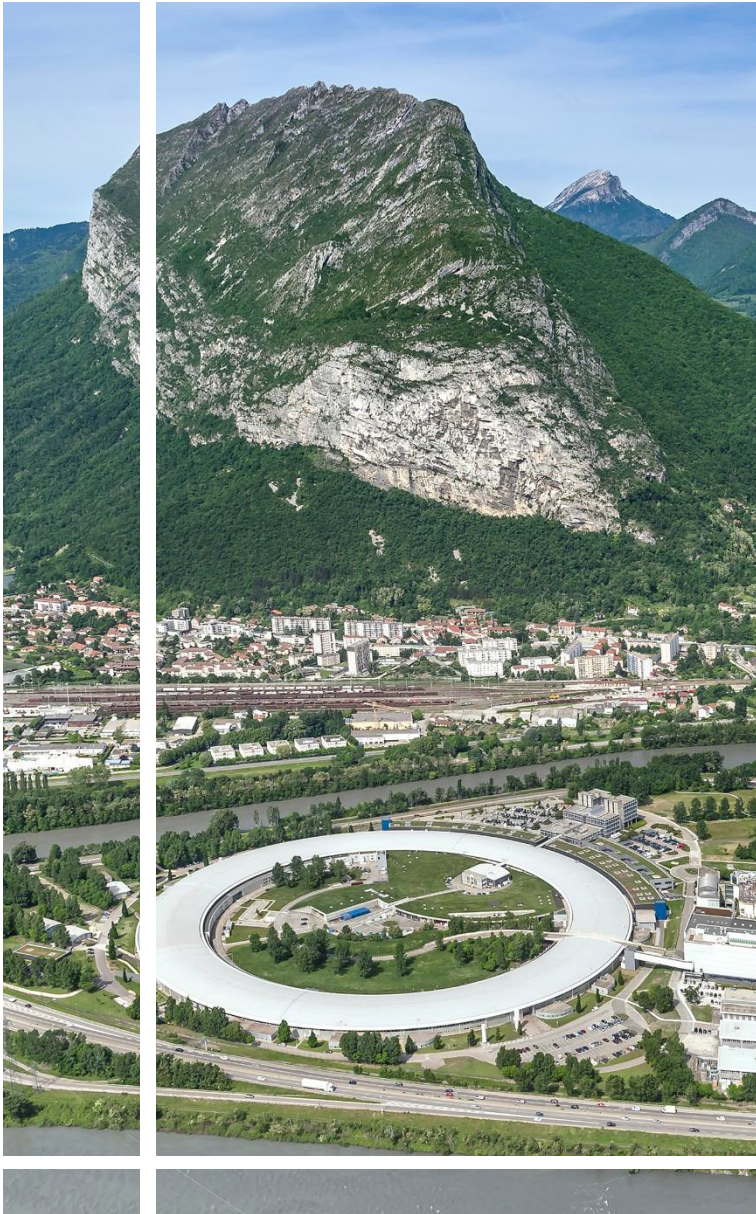
# Execute a workflow (use a proper Ewoks task scheduler in production)
varinfo = {"root_uri": "/tmp/myresults"} # optionally save all task outputs
result = execute_graph(workflow, varinfo=varinfo, inputs=inputs)
print(result)

```

EWOKS interfaces

- Desktop
- Web
- Shell
- Python

Developer usage like triggering workflows from the acquisition control system.



Increase the scientific output of the ESRF

- ↪ **Visualize the experiment** for humans
- ↪ Produce **domain specific results** when possible, X-ray specific otherwise
- ↪ Integrate scientific software in the experiment
- ↪ Workflow based solution (**executable data provenance**)
- ↪ **Meta approach**: decouple workflows and implementation from systems

Questions ?

<https://ewoks.esrf.fr>

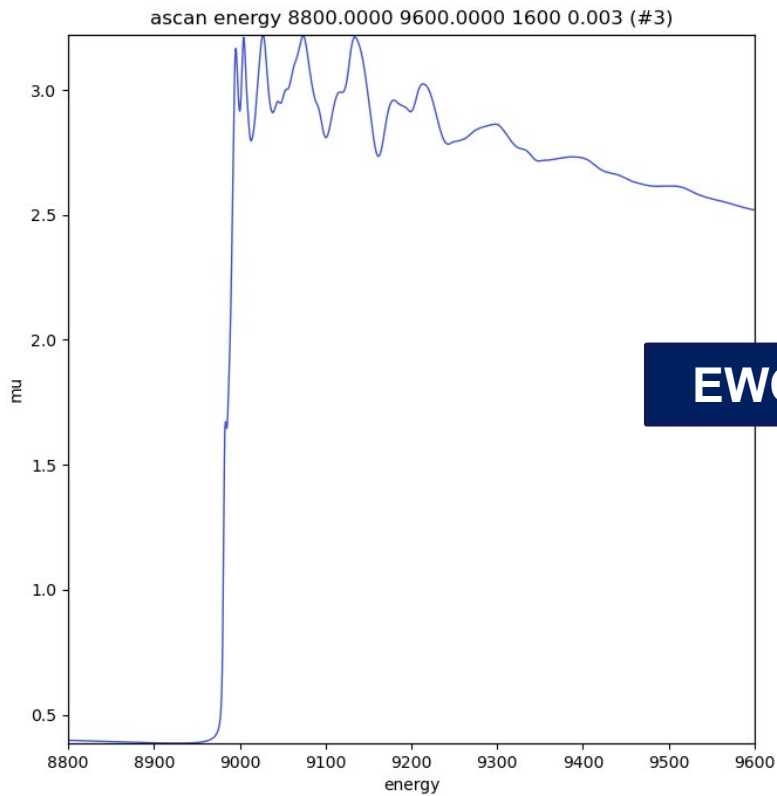


STREAMLINE has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 870313

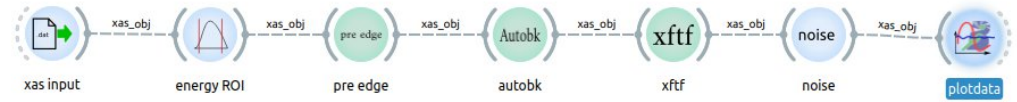
<https://streamline.esrf.fr/>

Spectroscopy: EXAFS visualization

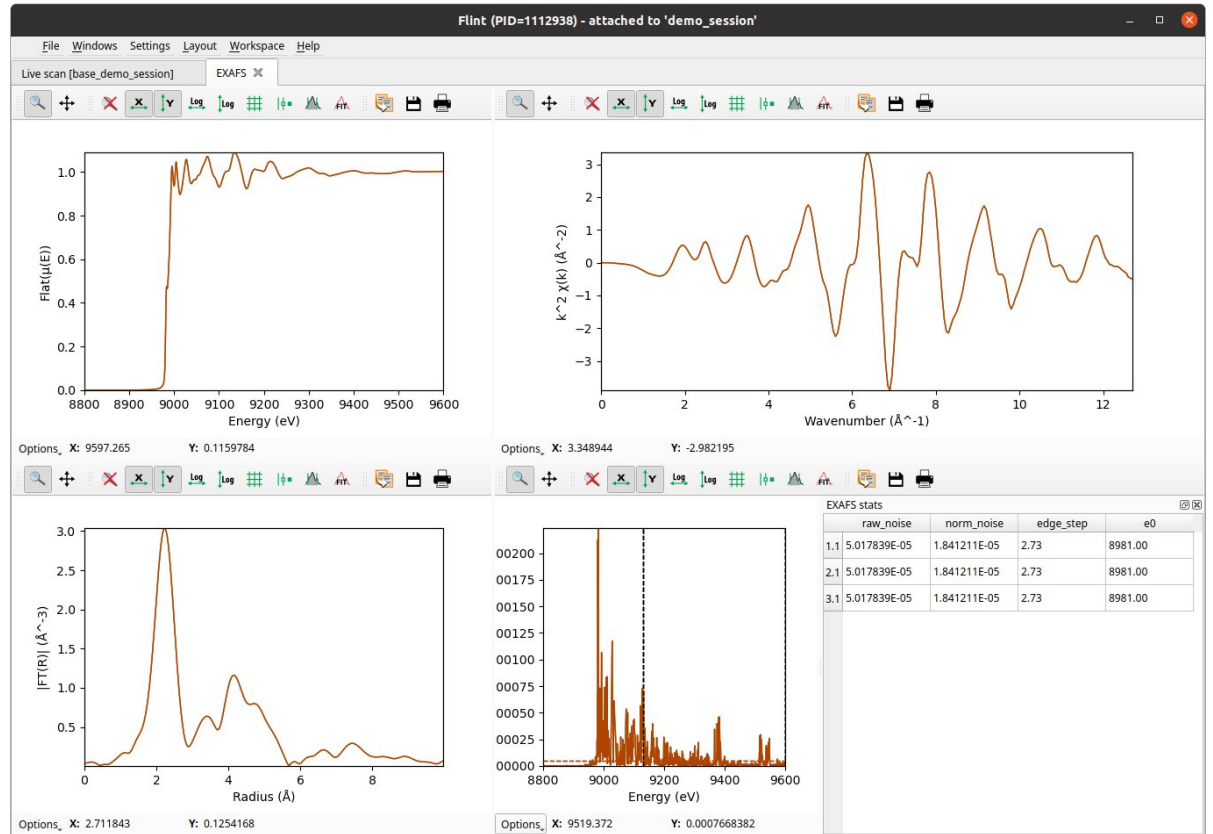
Raw Parameter Space



workflow based on xraylarch



Parameter space in which scientific decisions are made

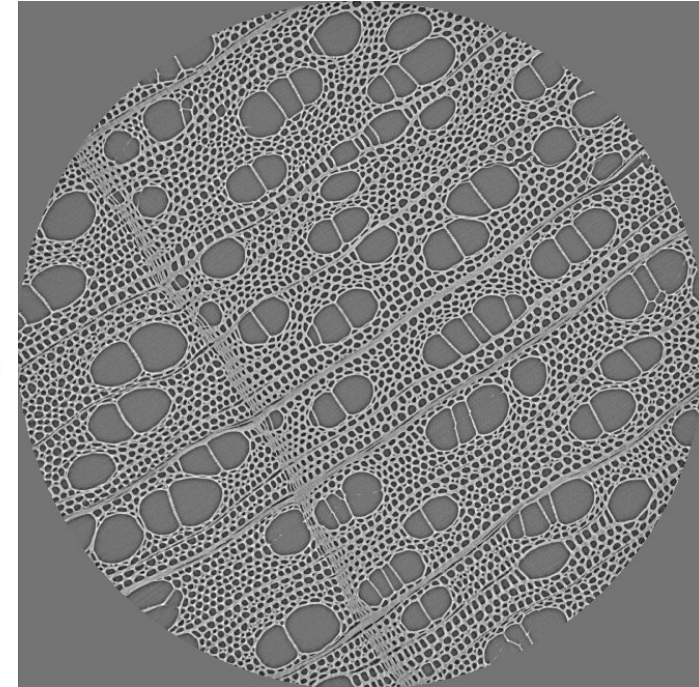
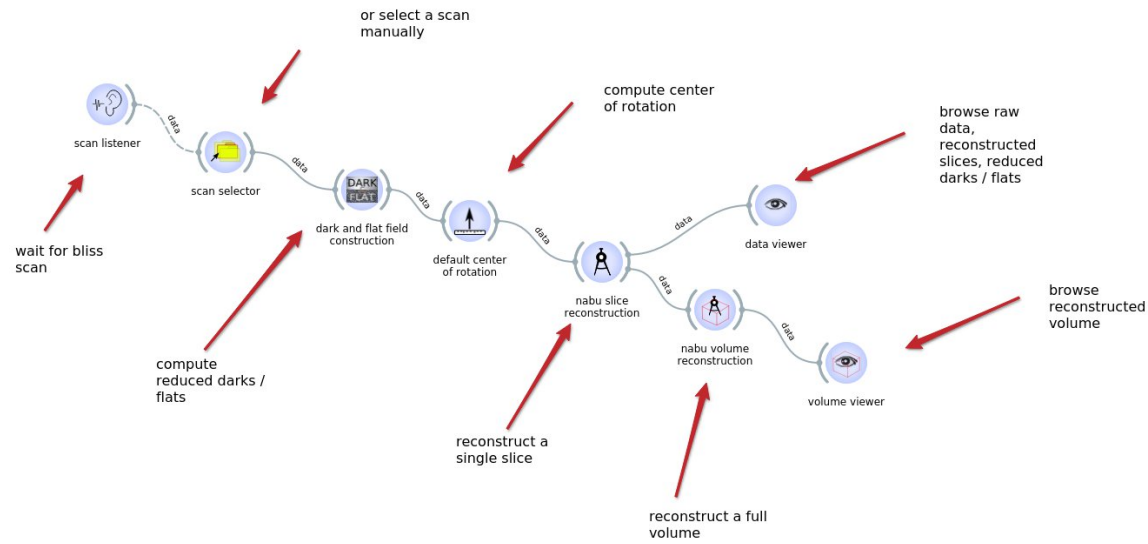


Tomography: 3D volume reconstruction of the X-ray density from projections



Inputs

Images of the sample in transmission + dark field images + flat field images + metadata



Outputs

Reconstructed volumes

