EUROPEAN
SPALLATION
SOURCE

Scipp

Scientific C++ and Python libraries for labeled multi-dimensional data

Simon Heybrock

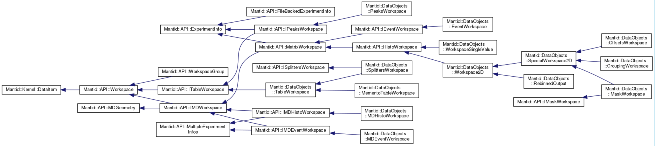
simon.heybrock@esss.se

Contributors:

Owen Arnold Igor Gudich Daniel Nixon Neil Vaytet

2019-09-17

Mantid Workspace hierarchy: restrictive, large, and complicated



Wish list

- Easy to learn and *remember*, obvious interfaces.
  - Usable by anyone who knows some Python.
  - But also not getting in the way or overly restrictive for experts.
- Flexible and accessible data structures.
  - Data and metadata content available at a glance.
  - Quick to inspect and visualize data.
  - Can store anything.
- Fast prototyping, give scientists tools to experiment with their data.

Requirements

Must-have features

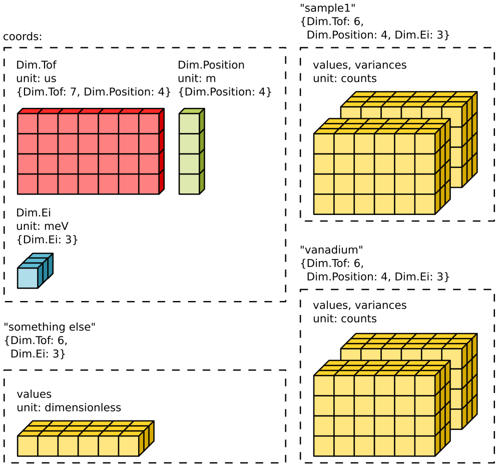
Carried over and evolved from Mantid:

1. Physical units *everywhere*.
2. Propagation of uncertainties.
3. Support for histograms, in particular bin-edges.
4. Event-data, a special case of sparse data (raw data later converted to histograms[1]).
5. `C++` backend for interfacing with existing `C++` codebases, performance opportunities.

[1] random-length list of events at each coordinate point, each event described by typically 1-4 small fields, e.g., `double`

Self-describing data using scipp

scipp's[2] Dataset, inspired by xarray (xarray.pydata.org):



[2] Etymology: Scientific C++ library → Sci++ → scipp

Anatomy of operations with scipp.Dataset

```
delta = d['a']                                  d['b']

             (1)                                  (1)
```

Natural and implicit:

1. Select dataset entries by name.

Anatomy of operations with scipp.Dataset

```
delta = d['a'][Dim.Pixel, 7:42]          d['b']

              (1)          (2)                        (1)
```

Natural and implicit:

1. Select dataset entries by name.
2. Slice based on named dimension.

Anatomy of operations with scipp.Dataset

```
delta = d['a'][Dim.Pixel, 7:42]   mean(d['b'], Dim.Energy)

            (1)          (2)          (4,5)  (1)     (3)
```

Natural and implicit:

1. Select dataset entries by name.
2. Slice based on named dimension.
3. Named dimensions for other ops, e.g., mean over given dim.

Anatomy of operations with scipp.Dataset

```
delta = d['a'][Dim.Pixel, 7:42]    mean(d['b'], Dim.Energy)

               (1)          (2)              (4,5)  (1)      (3)
```

Natural and implicit:

1. Select dataset entries by name.
2. Slice based on named dimension.
3. Named dimensions for other ops, e.g., mean over given dim.
4. Propagation of uncertainties.

Anatomy of operations with scipp.Dataset

```
delta = d['a'][Dim.Pixel, 7:42]    mean(d['b'], Dim.Energy)

                (1)           (2)               (4,5)  (1)        (3)
```

Natural and implicit:

1. Select dataset entries by name.
2. Slice based on named dimension.
3. Named dimensions for other ops, e.g., mean over given dim.
4. Propagation of uncertainties.
5. Handling of physical units.

Anatomy of operations with scipp.Dataset

```
delta = d['a'][Dim.Pixel, 7:42] - mean(d['b'], Dim.Energy)

              (1)         (2)      (4-8)(4,5)  (1)      (3)
```

Natural and implicit:

1. Select dataset entries by name.
2. Slice based on named dimension.
3. Named dimensions for other ops, e.g., mean over given dim.
4. Propagation of uncertainties.
5. Handling of physical units.
6. Matching of coordinates.

Anatomy of operations with scipp.Dataset

```
delta = d['a'][Dim.Pixel, 7:42] - mean(d['b'], Dim.Energy)

          (1)        (2)      (4-8)(4,5)  (1)     (3)
```

Natural and implicit:

1. Select dataset entries by name.
2. Slice based on named dimension.
3. Named dimensions for other ops, e.g., mean over given dim.
4. Propagation of uncertainties.
5. Handling of physical units.
6. Matching of coordinates.
7. Broadcasting into missing dimensions.

Anatomy of operations with scipp.Dataset

```
delta = d['a'][Dim.Pixel, 7:42] - mean(d['b'], Dim.Energy)

              (1)           (2)         (4-8)(4,5)  (1)        (3)
```

Natural and implicit:

1. Select dataset entries by name.
2. Slice based on named dimension.
3. Named dimensions for other ops, e.g., mean over given dim.
4. Propagation of uncertainties.
5. Handling of physical units.
6. Matching of coordinates.
7. Broadcasting into missing dimensions.
8. Transposing matching dimensions.

Anatomy of operations with scipp.Dataset

```
delta = d['a'][Dim.Pixel, 7:42] - mean(d['b'], Dim.Energy)

            (1)        (2)      (4-8)(4,5)  (1)     (3)
```

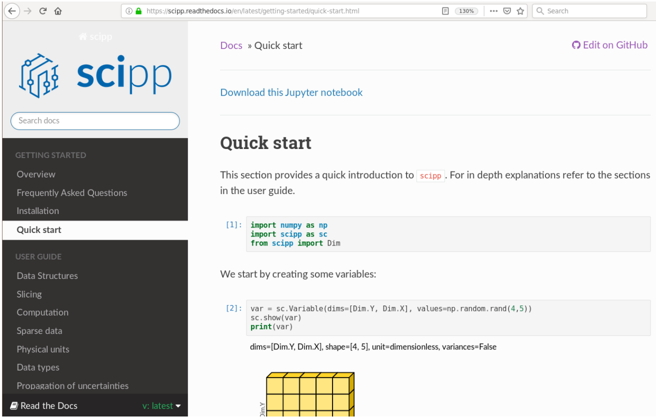
Natural and implicit:

1. Select dataset entries by name.
2. Slice based on named dimension.
3. Named dimensions for other ops, e.g., mean over given dim.
4. Propagation of uncertainties.
5. Handling of physical units.
6. Matching of coordinates.
7. Broadcasting into missing dimensions.
8. Transposing matching dimensions.

⇒ Free up mental capacity for the important things (science).

Example: Quick start (Jupyter notebook)

https://scipp.readthedocs.io/en/latest/getting-started/quick-start.html


https://scipp.readthedocs.io/en/latest/getting-started/quick-start.html

scipp

Search docs

GETTING STARTED
Overview
Frequently Asked Questions
Installation
Quick start

USER GUIDE
Data Structures
Slicing
Computation
Sparse data
Physical units
Data types
Propagation of uncertainties

Read the Docs v: latest

Docs » Quick start

Edit on GitHub

Download this Jupyter notebook

Quick start

This section provides a quick introduction to `scipp`. For in depth explanations refer to the sections in the user guide.

[1]:
```
import numpy as np
import scipp as sc
from scipp import Dim
```

We start by creating some variables:

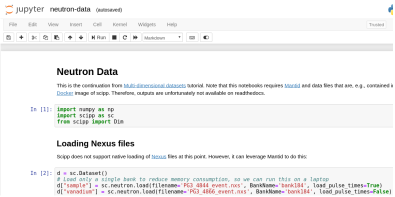
[2]:
```
var = sc.Variable(dims=[Dim.Y, Dim.X], values=np.random.rand(4,5))
sc.show(var)
print(var)
```

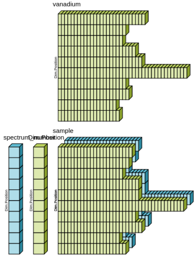
dims=[Dim.Y, Dim.X], shape=[4, 5], unit=dimensionless, variances=False

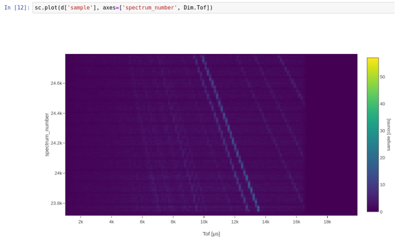
Dim.Y

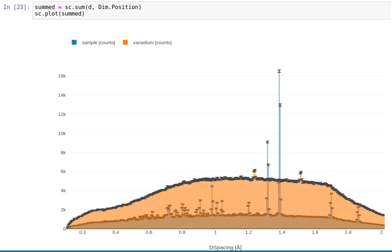
Example: Sparse data and neutrons (Jupyter notebook)

https://github.com/scipp/scipp-neutron-jupyter-demo
https://mybinder.org/v2/gh/scipp/scipp-neutron-jupyter-demo/master









About scipp

`scipp-units`	Physical units library, based on `boost-units`.
`scipp-core`	Core library with basic and generic operations.
`scipp-neutron`	Neutron-scattering specifics based on `scipp-core`, e.g., unit conversions.

. . . and more in the future?

Fact sheet

- C++17, Python bindings with pybind11
- 10 kLOC (`C++`) and 4 kLOC (`Python`)
  - not counting tests and documentation
- Jupyter notebooks with various visualization options
- Install using conda or docker
- Single threaded and not optimized at this point

Status and plan

- 2018: design and prototyping.
- First demo to wider audience in April 2019.
- Major refactor and cleanup of public interface and internals.
  - Conceptual changes → closer to `xarray` API.
- 0.1 release last month. This release is *intended for experimental use* and has *many limitations*.

Next

Continue work on core libraries, focus more on sparse data and performance.

- Using a *real workflow* to *drive and steer* development...
- ...thus avoid going off on a tangent.
- Determine what (if any) kind of parallelization beyond multi-threading required for our application. `dask`?

Conclusion

Summary

- scipp aimed at providing better and less error-prone processing of scientific data.
- Still a long way to go, but chiming in early could ensure that this does not become a specialized solution.
  ⇒ Comments and questions welcome!

Documentation:

- https://scipp.readthedocs.io
  - Download documentation pages as Jupyter notebooks.

Project home:

- https://github.com/scipp/scipp
  - Run scipp using Binder → notebook without installation.

Backup slide 1: Why not contribute to `xarray`?

Why not contribute to `xarray`?

- Too many additional requirements.
- Therefore not realistically achievable within the time frame(?).
- Some of the requirements are unlikely to be obtainable within `xarray`.

⇒ Focus on *interoperability* instead of reuse

- Good `NumPy` compatibility.
- May wrap `scipp.Dataset` ⇒ can use subset of `xarray.Dataset` functionality.
  - Successful proof of concept using `xarray` plotting.
  - Possible if a dataset does not use features beyond `xarray`'s.
- Potential for collaboration on subsets of functionality.