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’s² Dataset, inspired by xarray (xarray.pydata.org):

```
coords:
  Dim.Tof unit: us
  {Dim.Tof: 7, Dim.Position: 4}
  Dim.Position unit: m
  {Dim.Position: 4}

Dim.Ei unit: meV
  {Dim.Ei: 3}

"something else"
  {Dim.Tof: 6, Dim.Ei: 3}

values unit: dimensionless
```

```
"sample1"
  {Dim.Tof: 6, Dim.Position: 4, Dim.Ei: 3}

values, variances unit: counts

"vanadium"
  {Dim.Tof: 6, Dim.Position: 4, Dim.Ei: 3}

values, variances unit: counts
```

²Etymology: Scientific C++ library → Sci++ → scipp
Anatomy of operations with `scipp.Dataset`

```
1 delta = d['a']  
2            d['b']  
3 (1)        (1)  
```

**Natural and implicit:**

1. Select dataset entries by name.
Anatomy of operations with `scipp.Dataset`

1. `delta = d[ 'a' ][Dim.Pixel, 7:42]`  
2. `d[ 'b' ]`  
3. (1) (2) (1)

Natural and implicit:

1. Select dataset entries by name.
2. Slice based on **named** dimension.
Anatomy of operations with scipp.Dataset

1. \( \text{delta} = d[\text{'a'}][\text{Dim.PIXEL}, 7:42] \)
2. \( \text{mean}(d[\text{'b'}], \text{Dim.ENERGY}) \)

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`

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>delta = d['a'][Dim.Pixel, 7:42]</code></td>
</tr>
<tr>
<td>2</td>
<td><code>mean(d['b'], Dim.Energy)</code></td>
</tr>
<tr>
<td>3</td>
<td>(1) (2) (4,5) (1) (3)</td>
</tr>
</tbody>
</table>

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

\begin{align*}
delta &= d['a'][\text{Dim.Pixel}, 7:42] \quad \text{mean}(d['b'], \text{Dim.Energy}) \\
1 & \quad (1) \\
2 & \quad (2) \\
3 & \quad (4,5) (1) (3)
\end{align*}

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`

1. \[ \text{delta} = d[\text{Dim.Pixel}, 7:42] - \text{mean}(d[\text{Dim.Energy}]) \]

   (1) \hspace{1cm} (2) \hspace{1cm} (4-8)(4,5) \hspace{1cm} (1) \hspace{1cm} (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`

1. `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.
7. Broadcasting into missing dimensions.
Anatomy of operations with \texttt{scipp.Dataset}

1. \texttt{delta = d['a'][Dim.Pixel, 7:42] − mean(d['b'], Dim.Energy)}

Natural and implicit:

1. Select dataset entries by name.
2. Slice based on \texttt{named} dimension.
3. Named dimensions for other ops, e.g., mean over given dim.
4. Propagation of uncertainties.
5. Handling of physical units.
7. Broadcasting into missing dimensions.
8. Transposing matching dimensions.
Anatomy of operations with \texttt{scipp.Dataset}

\begin{verbatim}
1 delta = d['a'][Dim.Pixel, 7:42] - mean(d['b'], Dim.Energy)
2 (1) (2) (4-8)(4,5) (1) (3)
\end{verbatim}

Natural and implicit:

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

$\Rightarrow$ Free up mental capacity for the important things (science).
Example: Quick start (Jupyter notebook)


Quick start

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

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

We start by creating some variables:

```python
[2]: var = sc.Variable(dim=[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
```
## Example: Sparse data and neutrons (Jupyter notebook)

[https://github.com/scipp/scipp-neutron-jupyter-demo](https://github.com/scipp/scipp-neutron-jupyter-demo)

[https://mybinder.org/v2/gh/scipp/scipp-neutron-jupyter-demo/master](https://mybinder.org/v2/gh/scipp/scipp-neutron-jupyter-demo/master)

### Neutron Data

This is the continuation from the Multi-dimensional datasets tutorial. Note that this notebook requires `scipp` and data files that are, e.g., contained in an `archive` image of scipp. Therefore, outputs are unfortunately not available on mybinder.

```python
In [2]:
import numpy as np
import scipp as sc
from scipp import Dim

In [3]:
D = sc.Dataset()
D.load(filename="PG3_888_event_rec", bandpass=True, load_pulse_times=True)
D.rename("sample", "sc neutron load filename PG3 888 event rec", "BandName"="scs888", "load pulse times=Truex")

In [4]:
sc.plot(dim='sample', xname='spectrum number', Dim=Dim)
```

---

Simon Heybrock  
Scipp — Science with C++ and Python
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?
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.