1 Introduction

This describes a Python program wrapper for the CCP4 Program Scala (Evans, P "Data Reduction" in Proceedings of 1993 CCP4 Study Weekend.) This wrapper provides most of the commonly used functionality implemented in Scala.

2 Use Cases

2.1 Simple 1: Scale and Merge Data

The simplest implementation of the Scala wrapper is to simply take data from Mosflm and merge to a "standard" mtz file. This will need to perform the following operations:

- Apply a resolution limit.
- Apply a scaling model.
- Apply an error model.
- Add project, crystal, dataset information.

2.2 Simple 2: Just Merge

This implementation would take data previously scaled by an outside program (for instance XSCALE; SCALEPACK) and merge the reflections to provide scaling statistics and a merged reflection file. This will need to provide the following operations:

- Apply a resolution limit.
- Add project, crystal, dataset information.

2.3 More Complex 1: Merging Data, 2 Passes

Two datasets processed with Mosflm, appropriately REBATCHED and SORTED. The resulting MTZ file will need to be scaled and merged appropriately to give reasonable merging statistics from the two or more runs. This will require:

- Apply resolution limits.
- Apply a scaling model.
- Apply error models per run.
- Add project, crystal, dataset information.

The error models are best decided by scaling the data separately then recycling the parameters.

2.4 More Complex 2: Scaling MAD Data

More than one data sets collected from the same crystal but at different wavelengths. The data will need to be scaled separately but used together for building the absorption model etc. This is much more complex.

This will produce a number of output reflection files.

This will further need the reflections to be appropriately sorted and merged together in the file, with sensible project/crystal/dataset information in advance of the processing.

FIXME this needs to be specified.