XChemExplorer

Manual

Current version: v1.4.2

Date

23/12/2019

Author

Tobias Krojer

tobias.krojer@sgc.ox.ac.uk

Table of Contents

Introduction

XChemExplorer (XCE) is a data management and workflow tool, which supports large-scale analysis of protein-ligand structures by X-ray crystallography. It is not an actual algorithm, but serves as a launch pad for batch submission and analysis of the essential steps in the structure determination of protein-ligand structures.

Reference

Krojer, T., Talon, R., Pearce, N., Collins, P., Douangamath, A., Brandao-Neto, J., Dias, A., Marsden, B., and von Delft, F. (2017). The XChemExplorer graphical workflow tool for routine or large-scale protein–ligand structure determination. *Acta Cryst D 73*, 267–278.

XChemExplorer makes extensive use of other people's software, therefore please cite their work accordingly:

- XIA2
- DIMPLE
- ACEDRG
- PanDDA
- COOT
- REFMAC
- PHENIX
- MolProbity
- GRADE

Getting started

Prerequisites

XCE works on any Mac OSX or Linux system, but it is essential that CCP4 (www.ccp4.ac.uk) version 7.0 or higher is installed and correctly configured. XCE uses the python version that comes with it and will therefore not work if it does not exist. Additionally, it may be useful to install PHENIX, since XCE uses several of its tools for validation purposes.

Installation

Download XChemExplorer from http://tkrojer.github.io/XChemExplorer

Put the gzipped tar archive to wherever you want XCE to be installed. In case you have no root privileges, put it somewhere into your home directory, e.g.:

/home/tkrojer/software

Then change to the respective directory and unpack the archive, e.g.:

cd /home/tkrojer/software

gunzip XChemExplorer-1.2.tar.gz

tar –xvf XChemExplorer-1.2.tar

This will create a new directory, i.e. the XChemExplorer directory. Change into this directory, e.g.:

cd XChemExplorer-1.2

The contents of the directory should look something like this when you type '**ls –l**':

```
-rwxr-xr-x 1 tkrojer users 238K Jun 18 13:58 XChemExplorer.py
-rwxr-xr-x 1 tkrojer users  269 Jun 18 13:58 XChemExplorer_local.sh
-rwxr-xr-x 1 tkrojer users  316 Jun 18 13:58 XChemExplorer_dmd.sh
drwxr-xr-x 3 tkrojer users 4.0K Jun 18 13:58 web
-rwxr-xr-x 1 tkrojer users  553 Jun 18 13:58 setupssh.sh
drwxr-xr-x 2 tkrojer users 4.0K Jun 18 13:58 setup-scripts
-rwxr-xr-x 1 tkrojer users 2.8K Jun 18 13:58 README.md
drwxr-xr-x 2 tkrojer users 4.0K Jun 18 13:58 lib
drwxr-xr-x 2 tkrojer users 4.0K Jun 18 13:58 image
drwxr-xr-x 2 tkrojer users 4.0K Jun 18 13:58 icons
drwxr-xr-x 2 tkrojer users 4.0K Jun 18 13:58 helpers
drwxr-xr-x 2 tkrojer users 4.0K Jun 18 13:58 gui_scripts
-rwxr-xr-x 1 tkrojer users  182 Jun 18 13:58 Dockerfile
-rwxr-xr-x 1 tkrojer users  465 Jun 18 13:58 compile_test.py
lrwxrwxrwx 1 tkrojer users   20 Jun 18 14:33 XChemExplorer -> XChemExplorer_dmd.sh
```

The only thing left to do is to edit the XChemExplorer_dmd.sh file with a text editor. Change the line

```
export XChemExplorer_DIR='/usr/local/scripts/tobias/XChemExplorer'
```

to where you XChemExplorer is installed. In our example this would be

```
export XChemExplorer_DIR='/home/tkrojer/software/XChemExplorer-1.2'
```

That's it!

Limitations: XCE does currently not have a dedicated update mechanism. It is possible to clone the current master branch of the XCE github repository (https://github.com/xchem/XChemExplorer) and then frequently pull from it!

Usage

You can now run XCE by typing

```
/home/tkrojer/software/XChemExplorer-1.2/XChemExplorer_dmd.sh
```

It may however be easier if you insert an alias into your .bashrc or .cshrc file:

```
alias XChemExplorer='/home/tkrojer/software/XChemExplorer-1.2/XChemExplorer.sh
```

Compatibility

XCE does not have a problem with compatibility between different versions. This is because the program is essentially file system centred. Please see the original publications for more information.

Notes

The terms *datasource* and *database* are used somewhat interchangeably in the XCE GUI and in this manual.

Acknowledgements

XCE was originally written by me, but the program is currently being co-developed and maintained together with Rachael Skyner. Elliot Nelson contributed significantly to the design and development of the PanDDA refinement module. Frequent feedback from our amazing colleagues at the SGC and Diamond Light Source and input from the XChem user community was of utmost importance for software development. Special thanks goes to Romain Talon, Patrick Collins, Alice Douangamath, Jose Brandao-Neto and Alexandre Dias. The Research Informatics team at the SGC was instrumental for implementing the generation of HTML summary pages. The team at RCSB Rutgers have massively helped to support the group deposition mechanism. Finally, this work would not have been possible without the generous support from various SGC funders: the SGC is a registered charity (No. 1097737) that receives funds from AbbVie, Bayer, Boehringer Ingelheim, the Canada Foundation for Innovation, the Canadian Institutes for Health Research, Genome Canada, GlaxoSmithKline, Janssen, Lilly Canada, the Novartis Research Foundation, the Ontario Ministry of Economic Development and Innovation, Pfizer, Takeda and the Wellcome Trust (092809/Z/10/Z).

Settings

The settings tab (Figure 1) contains information where XChemExplorer (XCE) will write files to and where it can find certain files. It also tells XCE the name and location of the SQLite database file.

If you use XCE at DLS as part of an XChem project and open it somewhere in /dls/labxchem/… then you will usually not have to change anything. XCE will populate the directories and information about the SQLite database file with defaults.

The situation is different when you leave the labxchem environment; now you need to specify the information. However, there are only 3 pieces of information required to get started:

Project directory

This is where all files that XCE creates will end up. The project directory contains a sub-directory for every crystal and the name of every sub-directory corresponds to the *CrystalName* field in the database. Please note that the names can be completely arbitrary, although it is highly recommended to choose meaningful names. Please, check the XCE publication for more information about the structure and filename conventions of the project directory.

Reference Structure Directory

You can provide reference PDB, MTZ and CIF files in this directory. These files will be used for map calculation with DIMPLE or for selection of the best auto-processing result. The filenames can be arbitrary, however, if the files belong together they need to have the same root.

Data Source

Here you need to specify the database file that you want to use. If you are starting from scratch, please check the next section, which explains how to create a new database file

CCP4_SCR director

This is the same as your CCP4_SCR directory. It is essentially a scratch directory, which XCE uses to save input scripts for processing, restraints generation and refinement to. Usually one can ignore this directory, however, it is a good place to start trouble-shooting in case a job did not behave as expected.

Group Deposition Directory

This is the directory where XCE will write the gzipped tar archive file for upload into the Protein Data Bank.

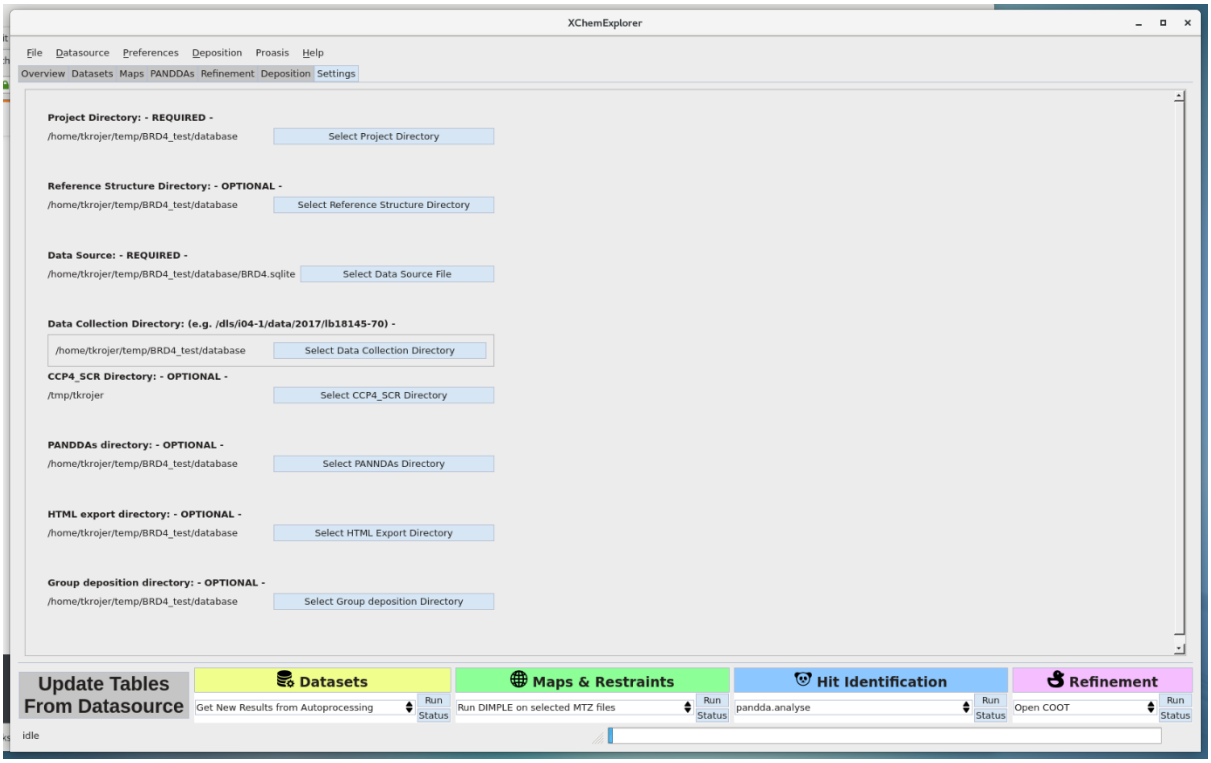


Figure 1. The settings tab in XChemExplorer.

Preferences

--- coming soon ---

Analysing DLS auto-processing results

XCE will read, copy and analyse the auto-processing results from data collected at the Diamond Light Source. Initially, it will check if the respective files exist, then it will copy the final MTZ and LOG files to the autoprocessing folder in the respective sample directory and finally it will try to select the 'best' result for further analysis, i.e. initial refinement. Please check out the XCE paper in case you want to know what 'best' means. It does currently read the results from xia2, dials, autoProc and autoProc-staraniso. Please note that the results from the fast_dp pipeline are ignored from v1.2 onwards! Additionally, It is important that the data are collected like this:

<protein_name>/<sample_name>

e.g.

lysozyme/lysozyme-x0001

Beamline directory

First, you need to select the *actual* data collection visit directory in the settings tab. For example, if your crystals were collected during visit /dls/i04-1/data/2019/mx19301-24 then press the 'Select Data Collection Directory' button and navigate to this directory. This is most likely obsolete by now, but please ignore the beamline directory in your labxchem folder (if it still exists)! In the past one could link multiple visits into the beamline directory, but this feature is obsolete!

If your data were collected in 'Agamemnon mode' (i.e. all crystals in a specific puck belong to a unique visit), then check the 'Read Agamemnon data structure' checkbox, then select one of the visits which contain your data. For example: if three of you pucks were collected and the data were saved in the following directories:

/dls/i03/data/2019/lb18145-139

/dls/i03/data/2019/lb18145-142

/dls/i03/data/2019/lb18145-137

You can then select any of them as your Data Collection directory. If the checkbox is checked then XCE sees that there are three lb18145 directories and it will first look for targets in them and later parse all three for data processing results.

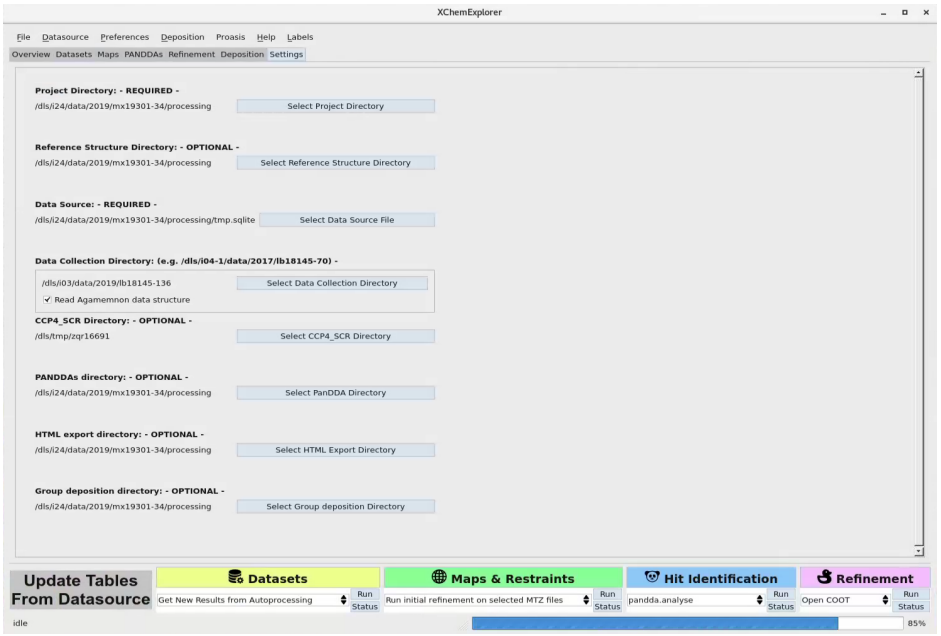


Figure 2. Selecting the Data Collection Directory in the Settings tab.

Getting data into XCE

Then, switch to the ***Datasets tab*** and select your target from the dropdown menu.

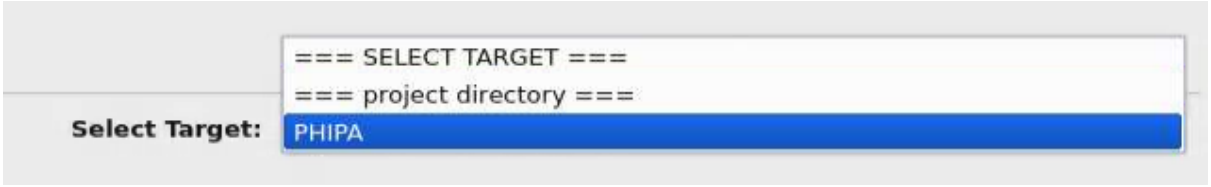


Figure 3. Target selection in the Datasets tab.

If your target does not show up in the dropdown menu, then it is usually because the *Data Collection Directory* was not set properly.

Select '*Get New Results from Autoprocessing*' from the yellow action box and press 'Run'.

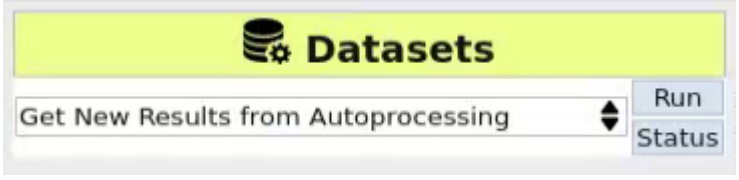


Figure 4. Parsing of the Data Collection Directory for new auto-processing results.

Manually selecting auto-processing results

--- missing ---

Changing selection criteria

--- missing ---

Rescoring

--- missing ---

Known issues

There are sometimes cases where you may have made changes to the project directory, but when you want to parse the data collection visit directory again, nothing happens. See I have wiped the contents of the project (initial_model) directory, but '*Get new results from autoprocessing*' does nothing when I run it again for details.

Database

XCE uses a simple SQlite database to capture information, results and outcomes that are generated during the project. If your samples were collected at the XChem facility at the Diamond Light Source, then you will usually just take the soakDB file that was created during crystal preparation. Otherwise, select '*Create New Datasource (SQLite)*' from the *Datasource* menu and create a new database (Figure 5).

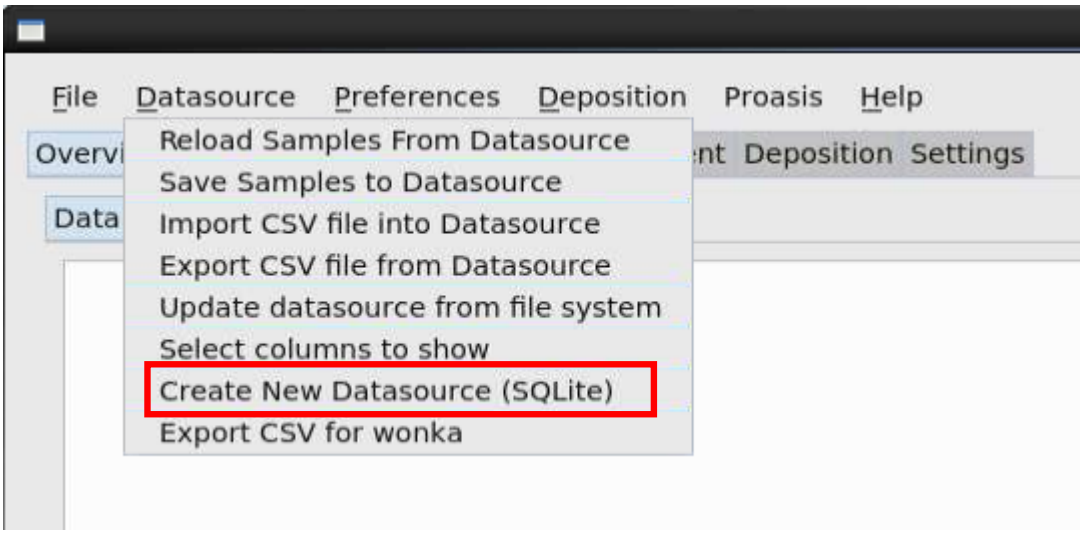


Figure 5. Datasource menu item 'Create New Datasource (SQLite)'.

Database update

There should be no need to make changes to the database when working on an XChem project. One notable exception is the entry of information about soaked/ co-crystallised compounds in case these compound are not part of the XChem fragment libraries. The initial ambition was to have a database update functionality available as part of XCE, but this approach will not be pursued for the time being. The *Datasource* menu contains some rudimentary functionalities for database changes, but it is recommended to follow the instructions outlined here in case the need for database manipulation arises. Also, keep in mind that the tables in XCE only display the contents of the database. Changing the value of the fields, although it is possible, has not effect on the database! A single click on '*Update Table From Datasource*' will bring back the actual content of the database.

Software

There are several programs for manipulation of SQLite files, the one we typically use is SQLite Browser (https://sqlitebrowser.org) (Figure 6). It is available for Windows, Mac and Linux.

Instructions

It is easily possible to make all the required changes directly within SQLite Browser, but this really only make sense if you need to change a few fields. For large-scale data entry, you should export the

respective table (usually only the *mainTable*) into CSV format and then make the required changes in Microsoft Excel. Other programs like OpenOffice should work as well, however, there have been anecdotal reports of import problems.

Before you start, make a copy of the current database in case something goes wrong!!!

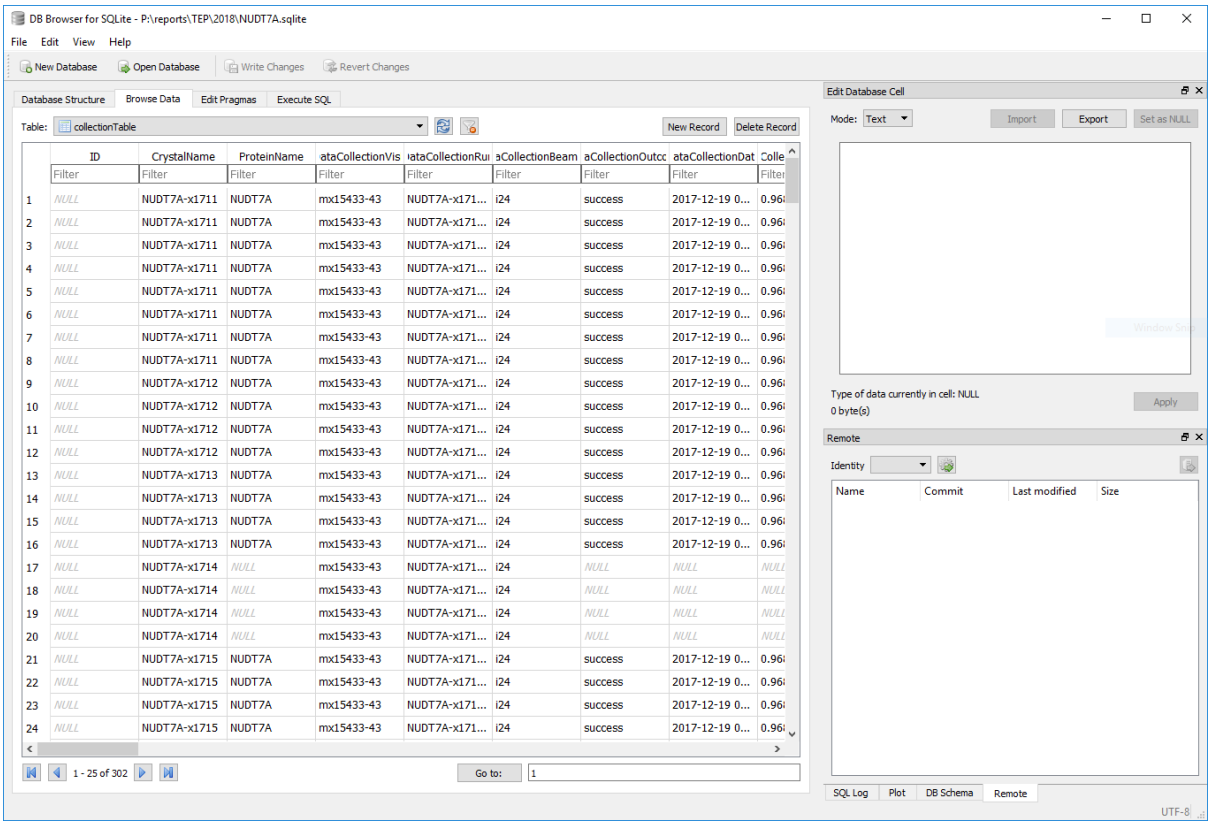


Figure 6. SQLite browser main window.

Next, open the sqlite file and export the export *mainTable* to a CSV file:

File -> Export -> Table(s) as CSV file…

Save as *mainTable.csv* if asked. Make sure that the box '*Column names in first line*' is checked (Figure 7).

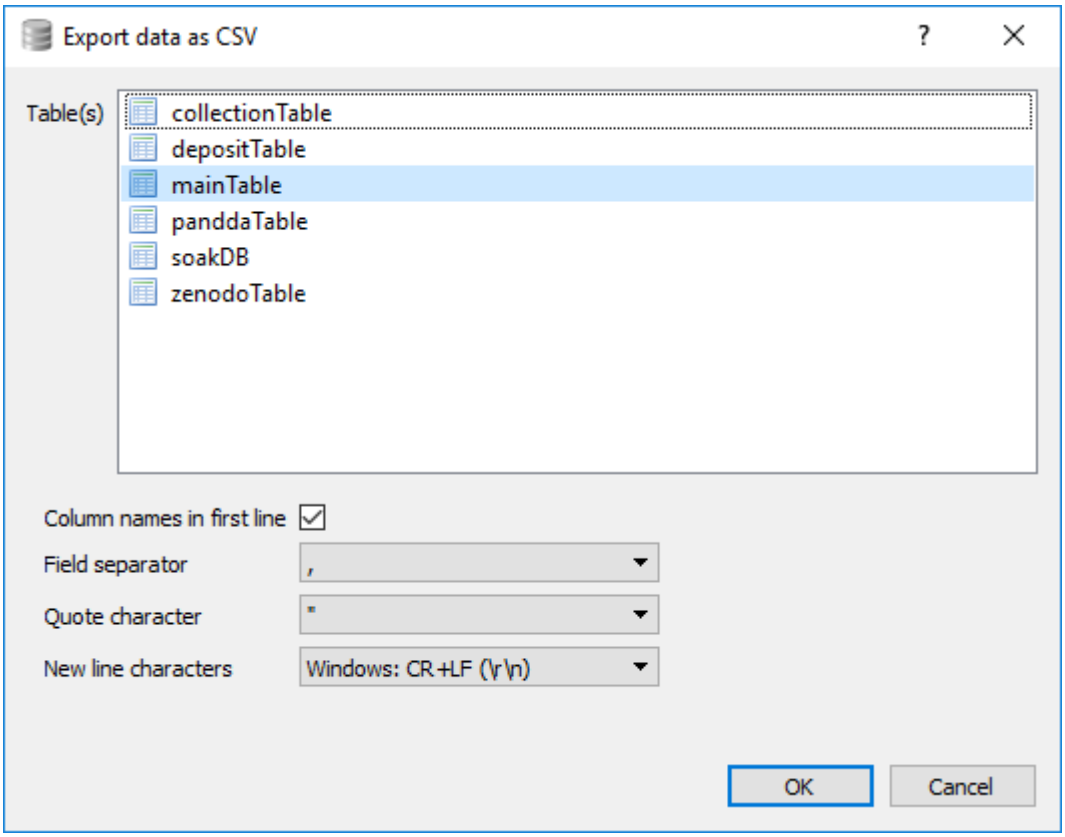


Figure 7. SQLite browser - export table to CSV file.

Next, delete the *mainTable* from database. In the '*Database Structure*' tab, highlight the table you want to delete, then, go to Edit -> Delete Table (Figure 8). Press 'Write Changes' and close the file.

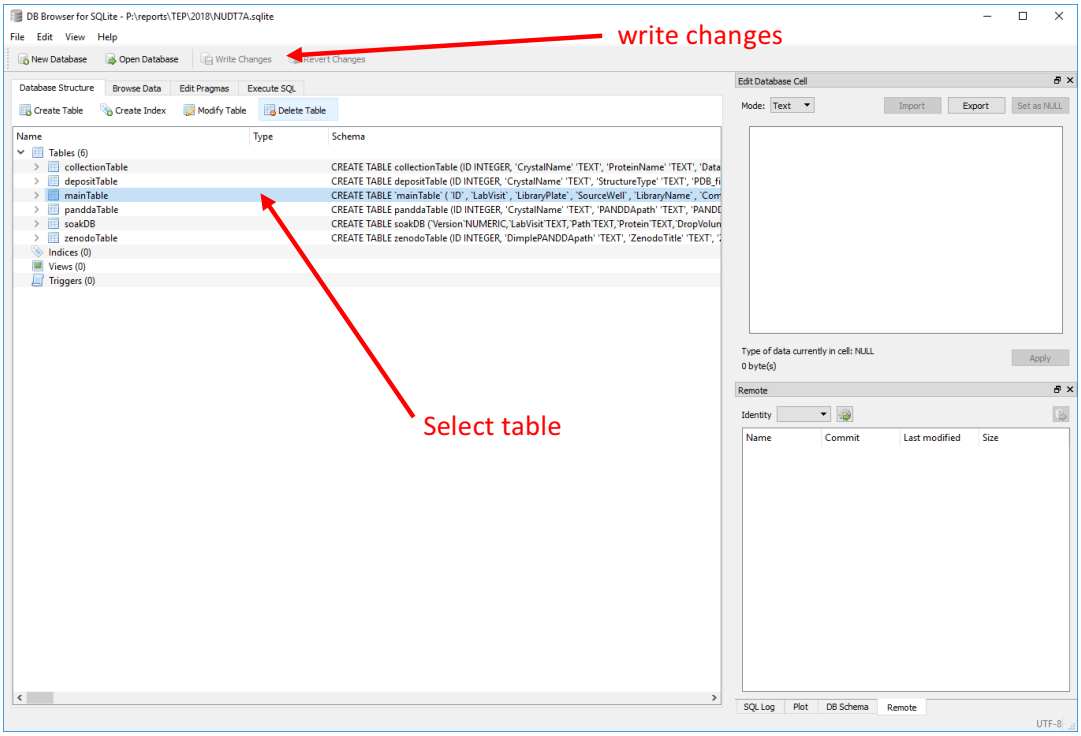


Figure 8. SQLite browser - select table for deletion.

Now open the CSV file in Microsoft EXCEL and make changes as necessary. The fields that most often will need changing are *CompoundSMILES* and *CompoundCODE*.

Open the SQLite file again and select File -> Import -> Table from CSV file…

Make sure the table name is *mainTable* and that '*Column names in first line*' is checked (Figure 9).

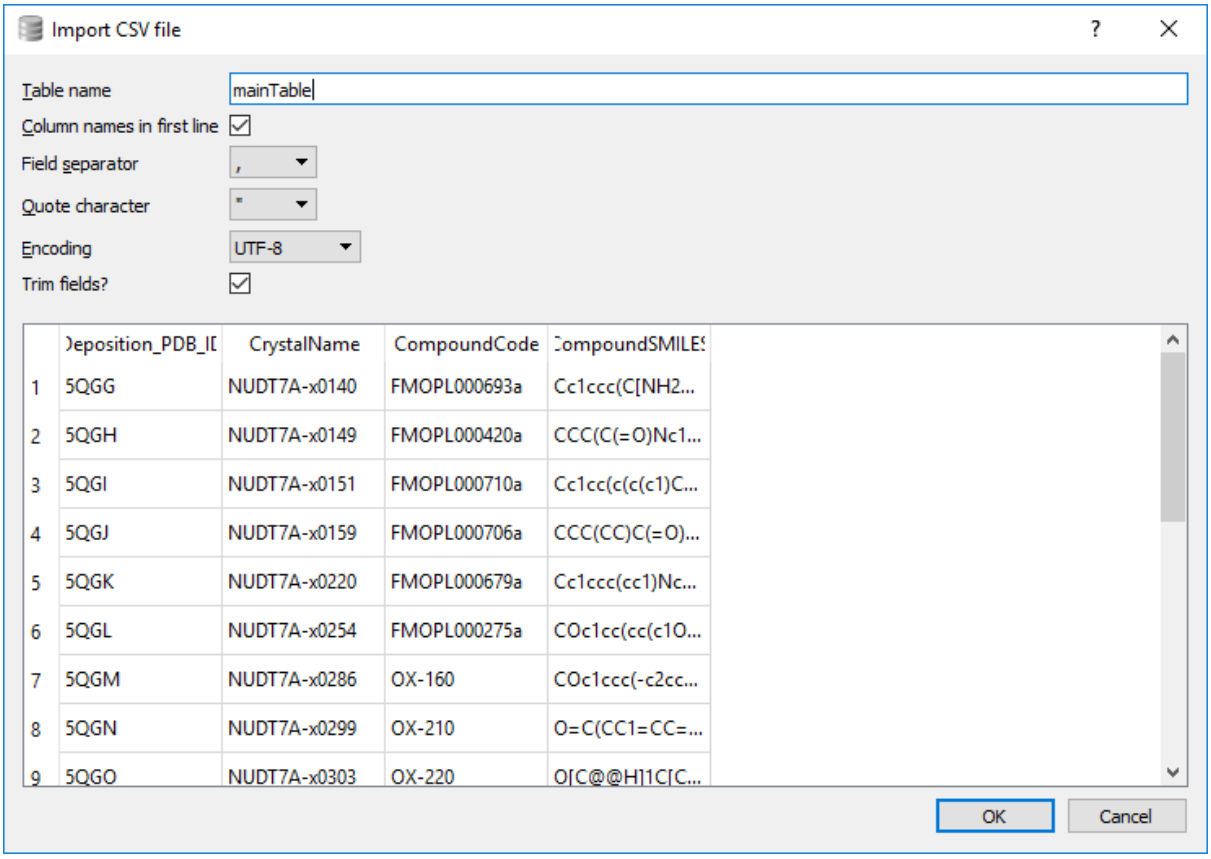


Figure 9. SQLite browser - import CSV file as table into SQLite database.

Dataset reprocessing

Running xia2 or dials

XCE can be used to reprocess datasets with either XIA2 or DIALS. This option is available in the Datasets/ Reprocess tab (Figure 10).

First, select the datasets directory, then, press '*Search Datasets*'. Note that if you want to process multiple-datasets then select the respective top-level directory under which the individual datasets are stored.

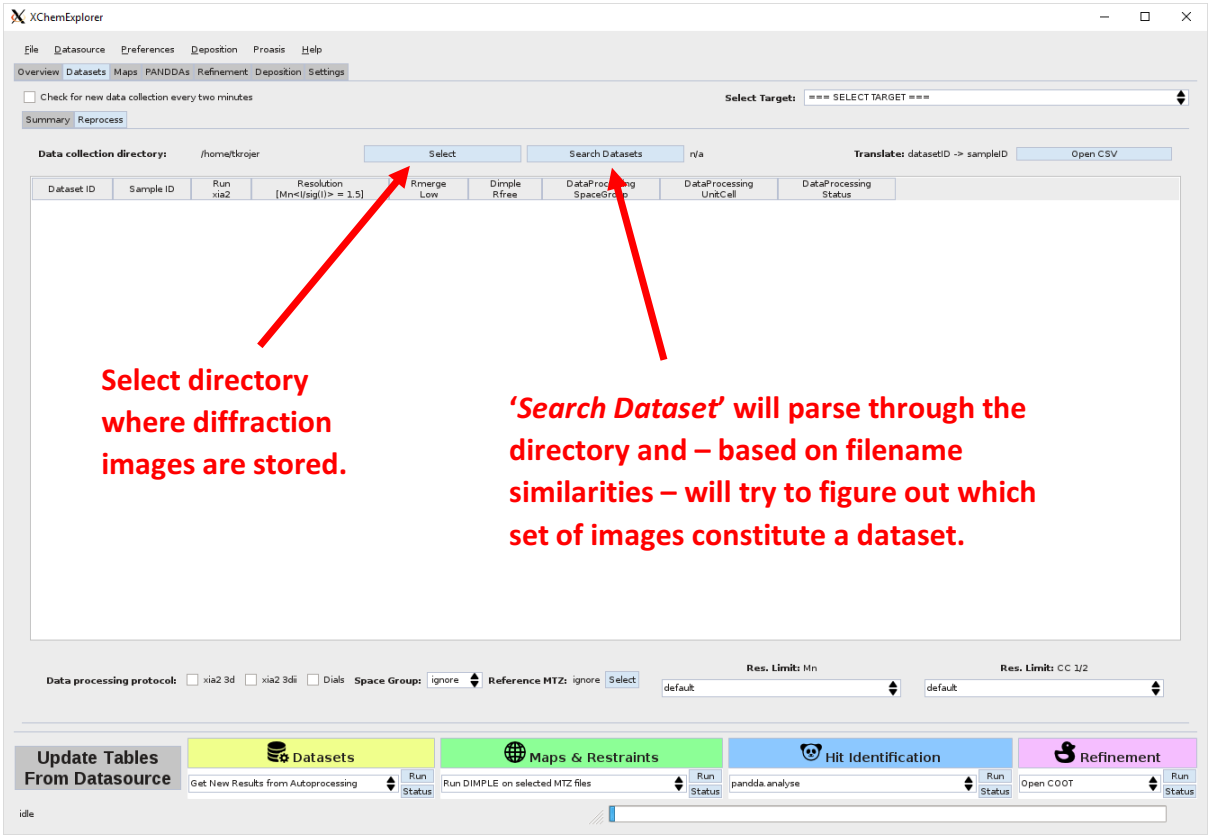


Figure 10. (Re-) processing of diffraction data in XCE.

Depending on the number of datasets, parsing of the respective directory may take a while. However, the progress bar will indicate how far along the program is and finally the table should get populated as in the exemplary screenshot below. There are a few things one should keep in mind:

- the *datasetID* is the name of each folder in the '*Data collection directory*'; e.g. all subfolders in /xdata/BRD4A

- only folders with more than 20 diffraction images in them will be listed

- XCE assumes that *datasetID=sampleID*; and if there is already an entry in the datasource for a particular *sampleID* then it will display things like Rmerge etc.

- you can change the sampleID simply by entering another value in the cell (Figure 11)

- if you want to use different sample IDs for many datasets, then you can use the '*Translate dataset ID -> sample ID*' option.

- you can either select all samples you want to run by clicking on the checkbox or you select a range of rows with the mouse, right-click, click on 'mark selected for reprocessing'

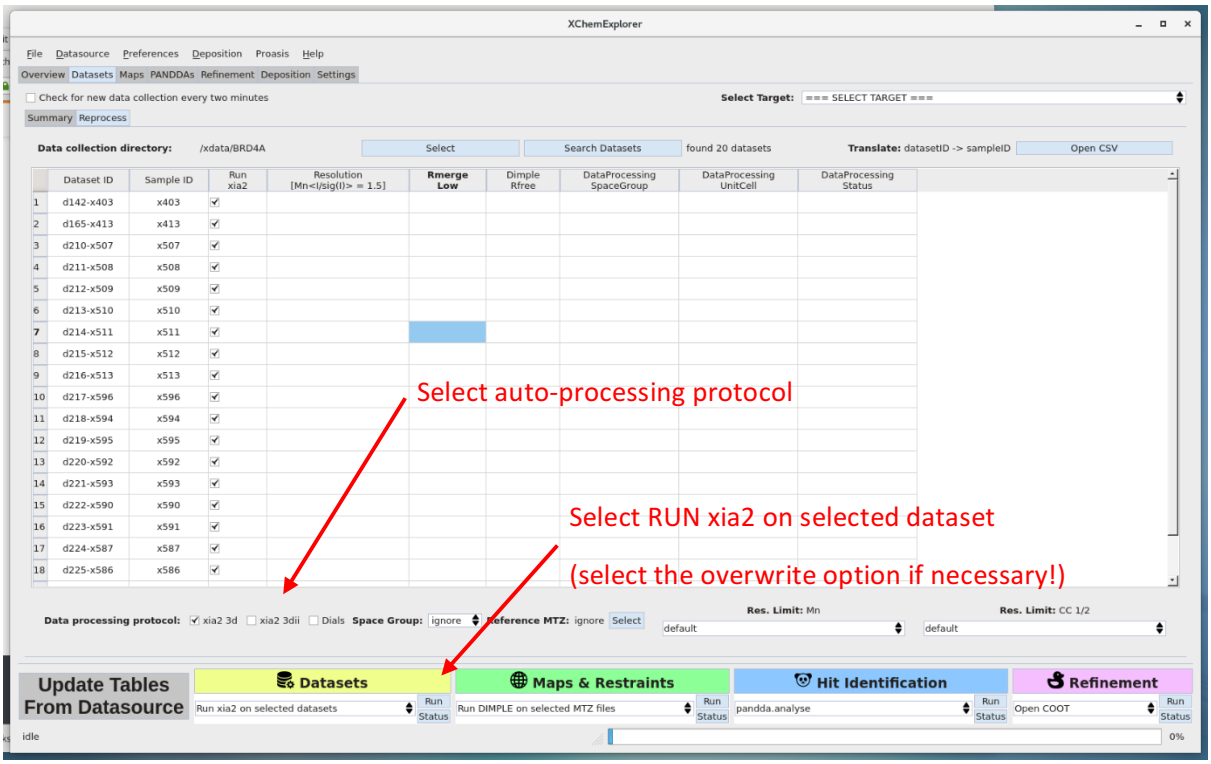


Figure 11. Dataset reprocessing tab after XCE found and assigned the diffraction images. Note that the values in the Sample ID column were manually entered. It would have been the same as in the Dataset ID column when XCE populated the table the first time.

Next, select the data processing protocol(s) you want to use (red arrow above), then select '*Run xia2 on selected datasets*' from the yellow *Datasets* actionbox and press '*RUN*'. Note that all data processing results will end up in the project directory.

If you want to provided reference files then put the respective files into the reference directory.

Please note that if your local machine is not automatically able to submit jobs to a computer cluster via qsub, it will process the datasets sequentially on the local machine. Needless to say that this may keep the machine busy for a while in case you want to process tens or even hundreds of datasets. In this case, it is advisable to use only one data processing protocol.

Please also note that XCE does currently not indicate when the jobs are finished (v1.2). You need to check the workload on your machine to find out.

Getting the results into the database

Now you need to get the data processing results/ outcomes back into the database. Stay in the main *Datasets* tab, but switch to the *Summary* sub-tab (Figure 12). Go to the select targets dropdown and select '*=== project directory ===*'. Select '*Get New results from Autoprocessing*' from the yellow action box and press *RUN*.

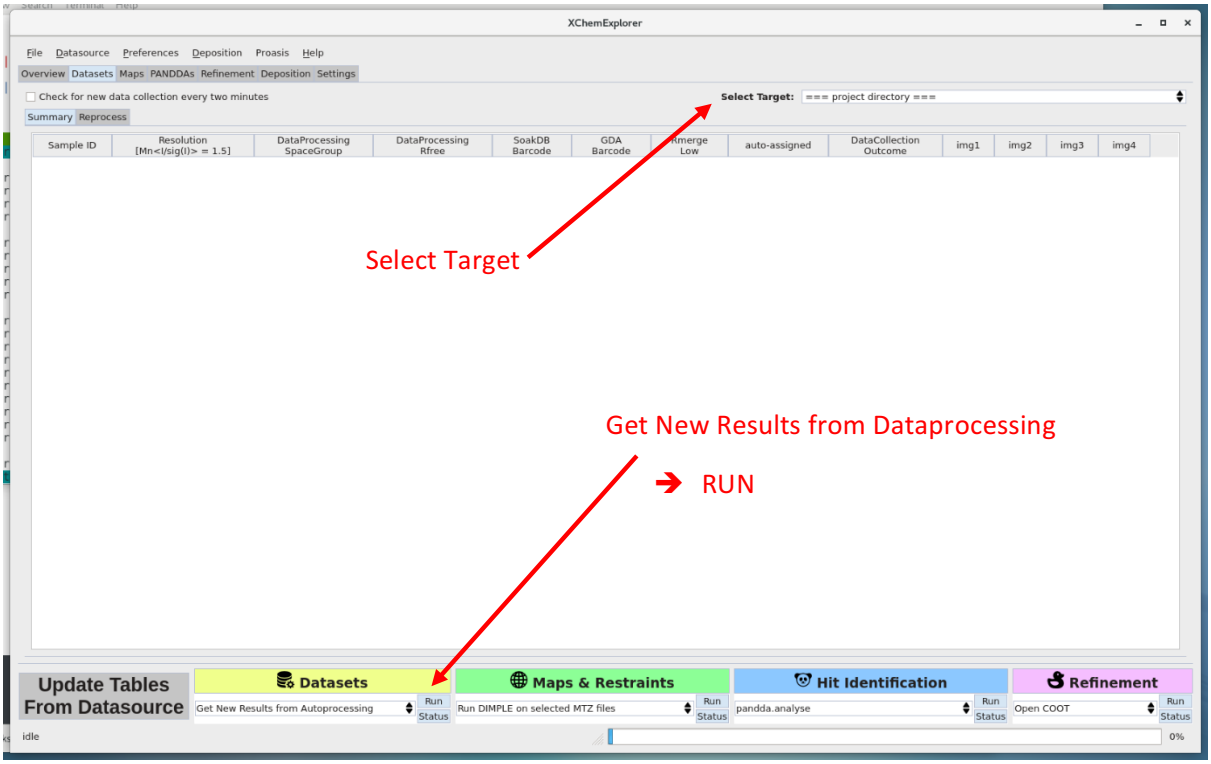


Figure 12. (Re-) processing of diffraction data in XCE. Reading the results.

If everything worked well, each sample directory with a successful data processing outcome should look something like this:

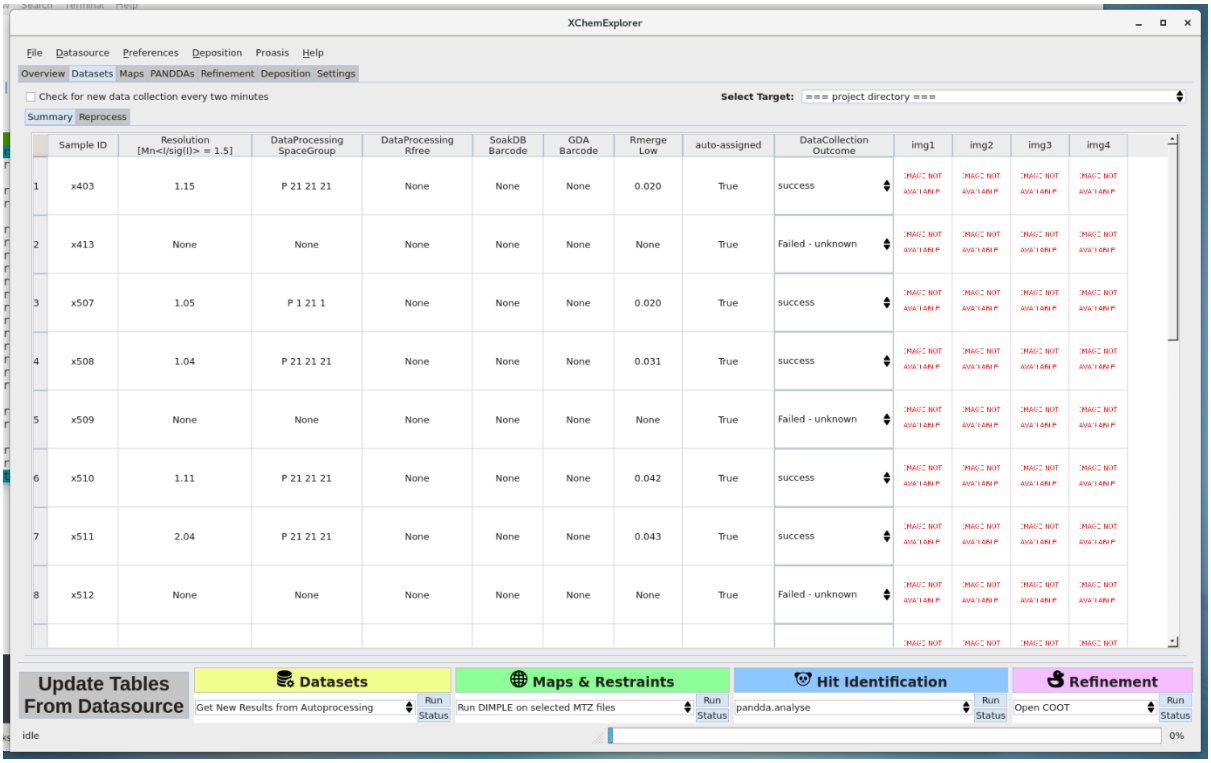


Figure 13. Results from reproccsing with xia2.

You can check the respective sample directories; as you can see below, there should be symbolic links to the *MTZ* and *AIMLESS* logfile from xia2; and the files have the same root as the respective sample directory.

```
drwxr-xr-x 3 tkrojer users 4.0K Jun 18 13:37 diffraction_images

drwxr-xr-x 3 tkrojer users 4.0K Jun 18 14:42 processed

drwxr-xr-x 3 tkrojer users 4.0K Jun 18 15:50 jpg

drwxr-xr-x 3 tkrojer users 4.0K Jun 18 15:50 autoprocessing

lrwxrwxrwx 1 tkrojer users   58 Jun 18 15:50 x596.mtz -> autoprocessing/database-run_13d/AUTOMATIC_DEFAULT_free.mtz

lrwxrwxrwx 1 tkrojer users   61 Jun 18 15:50 x596.log -> autoprocessing/database-run_13d/AUTOMATIC_DEFAULT_aimless.log
```

Initial Map Calculation

XCE uses DIMPLE to perform an initial round of refinement and to calculate the resulting 2fofc and fofc maps.

Please note that the tables in XCE do no update automatically! It does so during startup, but they will not refresh automatically. Also, there is currently a bug which means that the status indicators in the MAPS table are not always up to date (v1.2).

If you have successfully processed your datasets or read in the results from DLS auto-processing, press on the grey '*update tables from datasource*' button on the lower left hand corner (Figure 14). You can press this button as often as you want and it will update all tables with the exception being the Datasets Summary table with the latest information from the database.

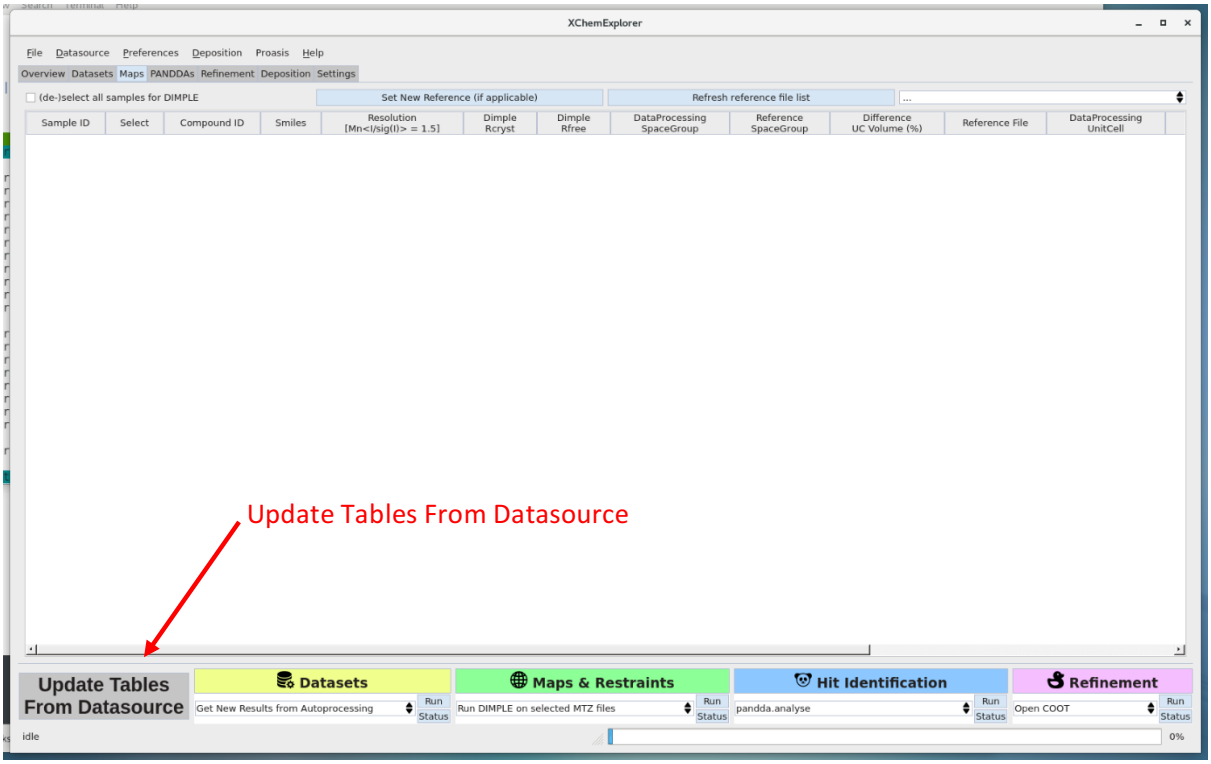


Figure 14. Update tables from datasource. All tables in XCE are currently not updated automatically and need active pulling from the database.

In our current example, the table will look like this:

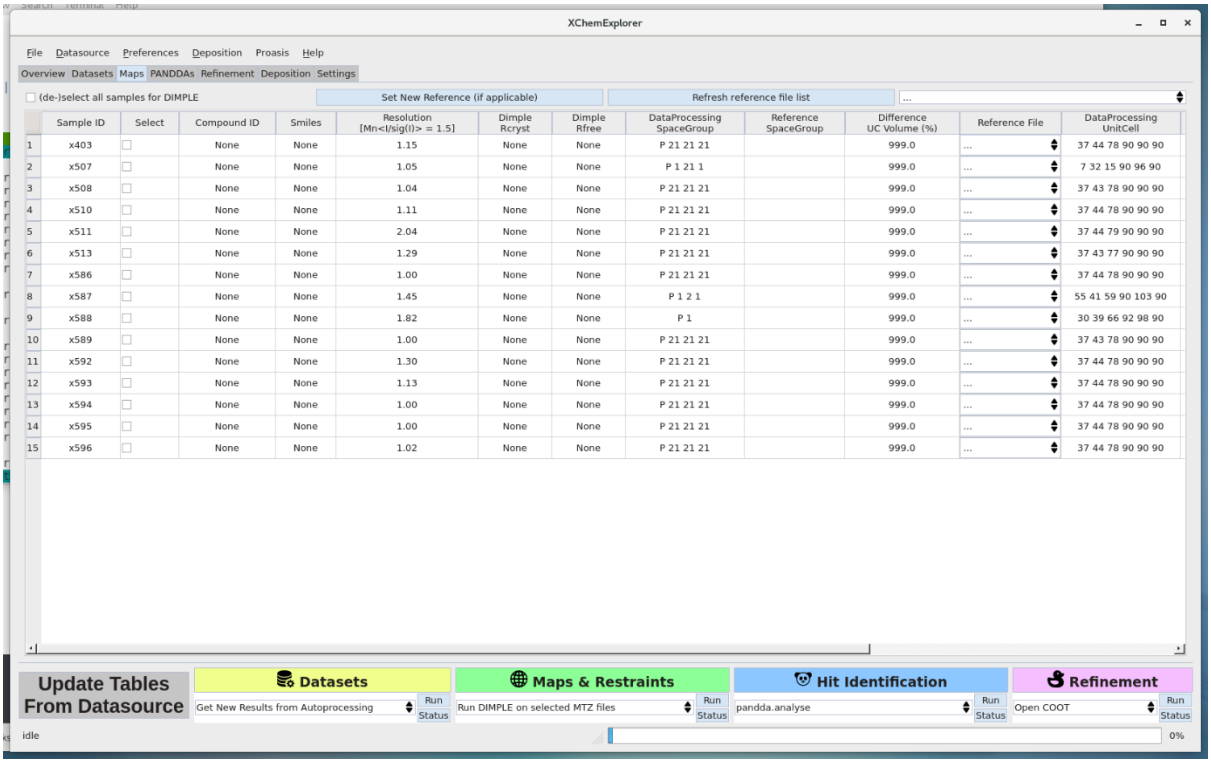
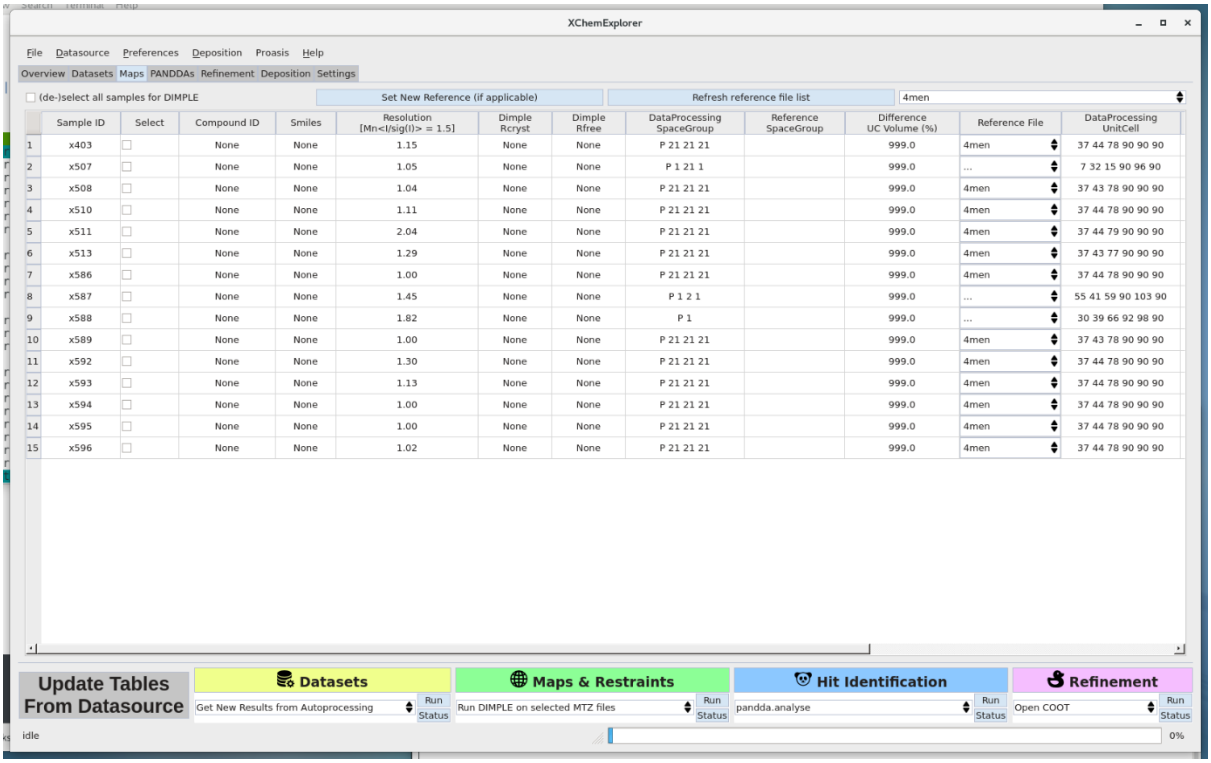


Figure 15. Populated Maps table.

We can now see information about the high resolution limit and space group for each crystal in the table, but the information about which reference PDB file to use is still empty. This is because in the example I have not yet provided a suitable reference PDB file in the reference directory. After this is done, one needs to press '*Refresh reference file list*', then select the reference file that you want to use, and press '*Set New Reference (if applicable)*'.

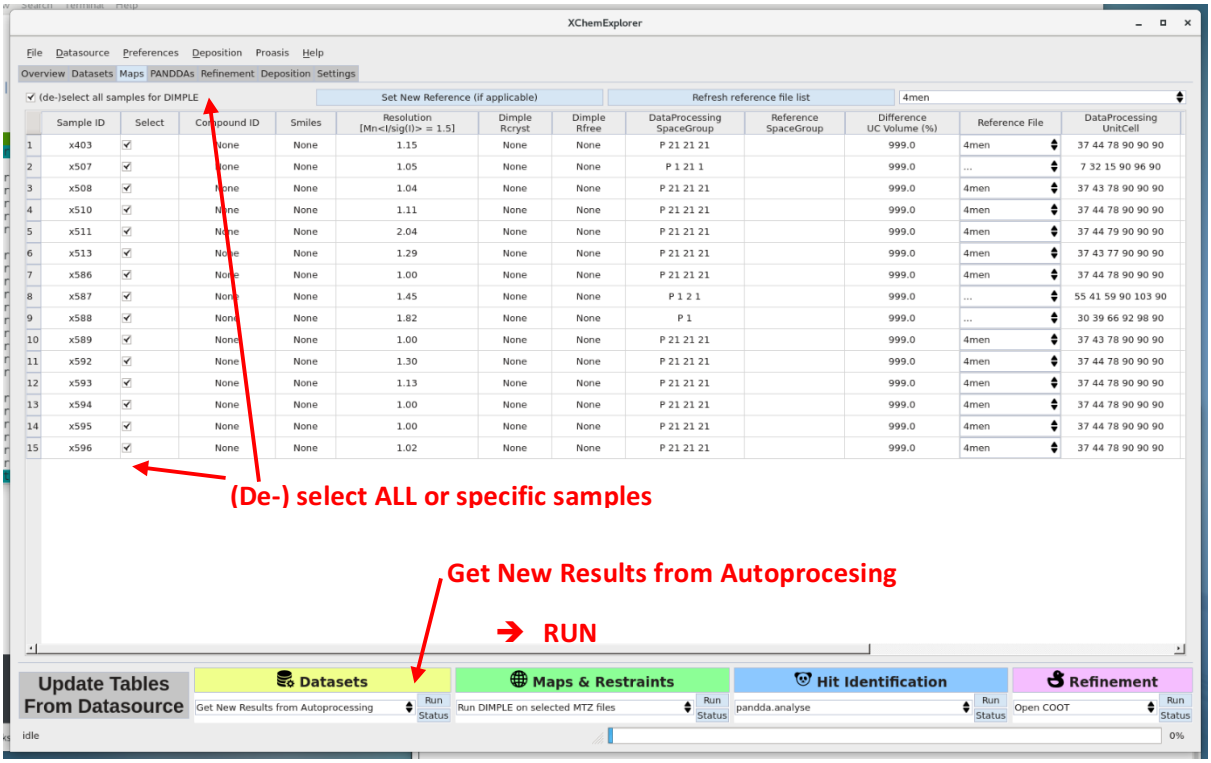


	Sample ID	Select	Compound ID	Smiles	Resolution [Mn<I/sig(I)> = 1.5]	Dimple Rcryst	Dimple Rfree	DataProcessing SpaceGroup	Reference SpaceGroup	Difference UC Volume (%)	Reference File	DataProcessing UnitCell
1	x403		None	None	1.15	None	None	P 21 21 21		999.0	4men	37 44 78 90 90 90
2	x507		None	None	1.05	None	None	P 1 21 1		999.0	...	7 32 15 90 96 90
3	x508		None	None	1.04	None	None	P 21 21 21		999.0	4men	37 43 78 90 90 90
4	x510		None	None	1.11	None	None	P 21 21 21		999.0	4men	37 44 78 90 90 90
5	x511		None	None	2.04	None	None	P 21 21 21		999.0	4men	37 44 79 90 90 90
6	x513		None	None	1.29	None	None	P 21 21 21		999.0	4men	37 43 77 90 90 90
7	x586		None	None	1.00	None	None	P 21 21 21		999.0	4men	37 44 78 90 90 90
8	x587		None	None	1.45	None	None	P 1 2 1		999.0	...	55 41 59 90 103 90
9	x588		None	None	1.82	None	None	P 1		999.0	...	30 39 66 92 98 90
10	x589		None	None	1.00	None	None	P 21 21 21		999.0	4men	37 43 78 90 90 90
11	x592		None	None	1.30	None	None	P 21 21 21		999.0	4men	37 44 78 90 90 90
12	x593		None	None	1.13	None	None	P 21 21 21		999.0	4men	37 44 78 90 90 90
13	x594		None	None	1.00	None	None	P 21 21 21		999.0	4men	37 44 78 90 90 90
14	x595		None	None	1.00	None	None	P 21 21 21		999.0	4men	37 44 78 90 90 90
15	x596		None	None	1.02	None	None	P 21 21 21		999.0	4men	37 44 78 90 90 90

Figure 16. Maps table after reference files for DIMPLE have been assigned.

You can check the XCE publication for more information about how XCE selects which reference file to use. Briefly, XCE will go through all PDB files that you have provided in the reference directory and read the CRYST card in their header. From the CRYST card, it will calculate the unit cell volume and determine the point group. Then it will compare this to the unit cell volume and point group of your crystal. If the unit cell volume differs by less than 12% (this can be adjusted in the *Preferences* menu) and the two have the same point group, then XCE will consider this to be a suitable input file for Dimple. This selection mechanism works well as long as you have either different point groups or the unit cell volume differs significantly between different space groups. It will most likely struggle in cases where you have different crystal forms that are not detectable by the rather coarse selection mechanism! However, I have often found that if XCE does not find a suitable reference file it is recommended to manually check what is going on. It may be that the presence of a certain ligand triggered some change that is best investigated outside the XCE workflow. If it really is a different crystal form, solve the structure as you usually would, then add the resulting structure to your reference file directory.

Now select which crystals you want to process (or simply choose *select all samples for DIMPLE*) and then choose '*Run DIMPLE on selected MTZ files*' from the green action box and press '*RUN*'.



If everything worked as expected, then then respective sample directory should now look like this:

```
drwxr-xr-x 3 tkrojer users 4.0K Jun 18 13:37 diffraction_images

drwxr-xr-x 3 tkrojer users 4.0K Jun 18 15:15 processed

drwxr-xr-x 3 tkrojer users 4.0K Jun 18 15:50 jpg

drwxr-xr-x 3 tkrojer users 4.0K Jun 18 15:50 autoprocessing

lrwxrwxrwx 1 tkrojer users   58 Jun 18 15:50 x403.mtz -> autoprocessing/database-run_13d/AUTOMATIC_DEFAULT_free.mtz

lrwxrwxrwx 1 tkrojer users   61 Jun 18 15:50 x403.log -> autoprocessing/database-run_13d/AUTOMATIC_DEFAULT_aimless.log

drwxr-xr-x 3 tkrojer users 4.0K Jun 18 16:18 dimple

lrwxrwxrwx 1 tkrojer users   53 Jun 18 16:21 dimple.pdb -> dimple/dimple_rerun_on_selected_file/dimple/final.pdb

lrwxrwxrwx 1 tkrojer users   53 Jun 18 16:21 dimple.mtz -> dimple/dimple_rerun_on_selected_file/dimple/final.mtz

lrwxrwxrwx 1 tkrojer users   53 Jun 18 16:21 2fofc.map -> dimple/dimple_rerun_on_selected_file/dimple/2fofc.map

lrwxrwxrwx 1 tkrojer users   52 Jun 18 16:21 fofc.map -> dimple/dimple_rerun_on_selected_file/dimple/fofc.map

-rw-r--r-- 1 tkrojer users 1.3M Jun 18 16:21 x403.free.mtz
```

You can now either run *PanDDA* or move straight to the Refinement tab and look at your electron density maps after initial refinement.

Twinning

If your data are almost perfectly twinned, chances are high that the autoprocessing pipelines will merge the data in the apparent higher symmetry laue group. Once this has happened, you need to reprocess the data in the respective lower symmetry laue group.

Automated ligand fitting

After initial map calculation and generation of ligand restraints, XCE can perform automated ligand fitting with phenix.ligandfit and rhofit in case these programs are available on your system. All you need to do is to select the samples in the MAPS tab and the respective option in the green action box and press RUN. Please note that these calculations can be very time-consuming and are best done on a computer cluster. Also, note that the fitting is currently only done in 2fofc maps and not in PanDDA Event maps! This means that it may be advisable to remove all water molecules from the site-of-interest before map calculation. However, this is something you should never do when you plan to run PanDDA! The different solutions can later be inspected through the XCE-COOT refinement interface (see page 30). In case different enantiomers are present, XCE will set up the calculations so that all of them are tried.

Figure 17. Automated ligand fitting into 2fofc maps with phenix.ligandfit and rhofit.

Ligand Restraints

Program selection for restrains generation

XCE supports generation of ligand restrains with *ACEDRG*, *phenix.elbow* and *grade*, as long the programs are installed on your system. *ACEDRG* is the default program. Use the *Preferences* menu (*Edit Preferences*) to switch to another program.

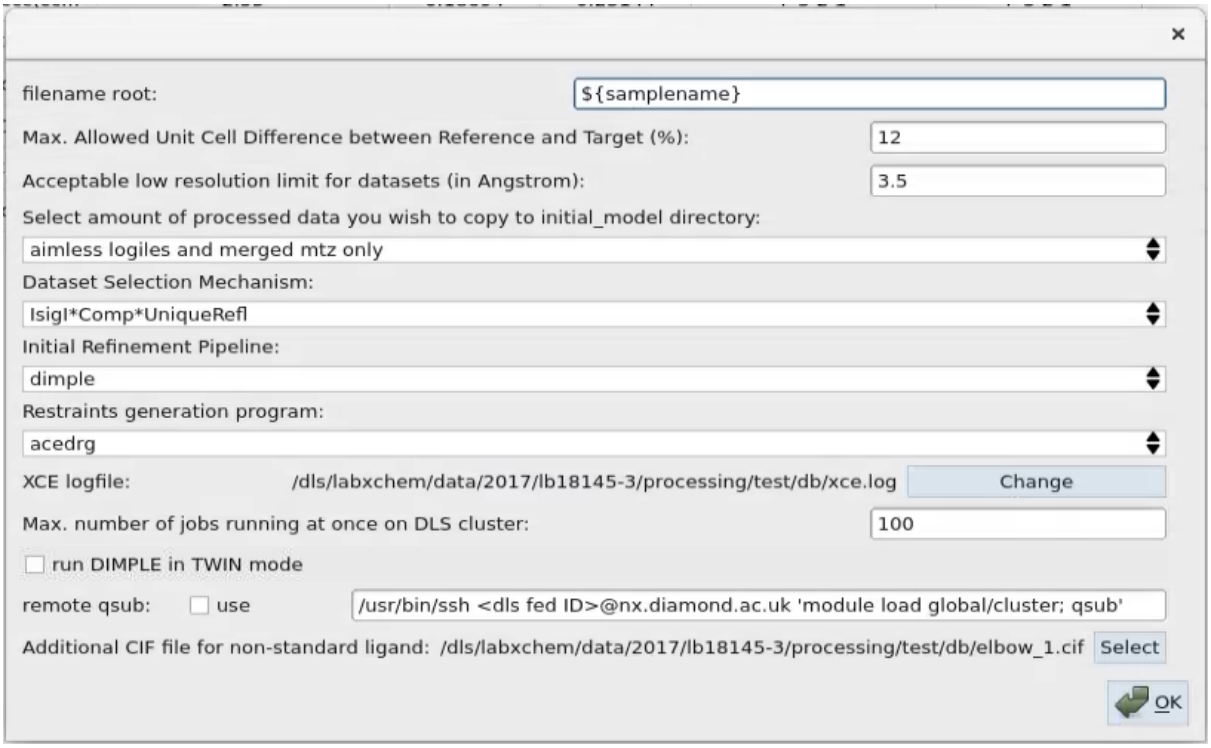


Figure 18. The preferences section of XCE. Use the combobox in the 'Restraints generation program' section to select your preferred program.

Creating ligand restrains

Before you can start making a new restraints file for your ligand, you have to make sure that the compound code and compound smiles information are provided in the database. This will happen automatically at the DLS XChem fragment screening facility, but you can update the database yourself if needed (see page 13 for details). Briefly, open the database with SQLite browser, then navigate to the *mainTable*. Look for the *CrystalName* column to find out which row you need to update. Then update the *CompoundCode* and *CompoundSMILES* fields, respectively. Finally, press '*Write Changes*', then close the file.

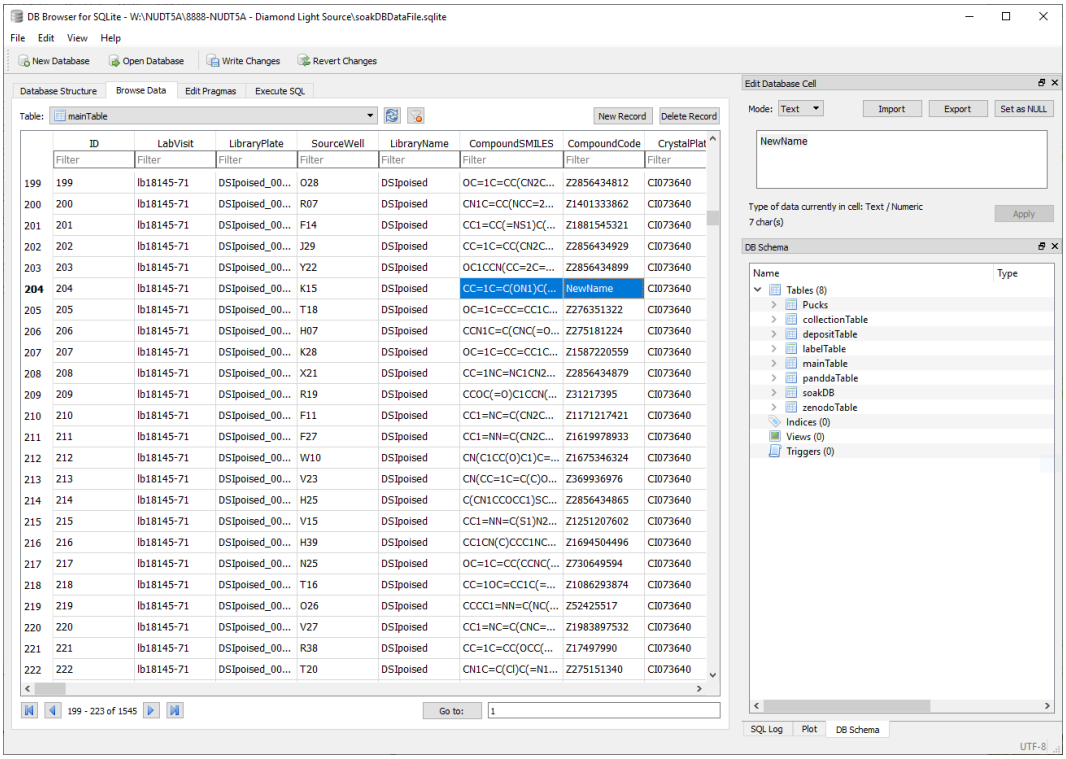


Figure 19. Updating compound information with SQLite brower.

Now open XCE and press '*Update Tables from Datasource*'. Switch to the *Maps* tab and the compound ID and associated SMILES string will show up. Select the compounds for which you want to generate restraints for (or select all using the *(de-)select all samples* checkbox at the upper left hand corner).

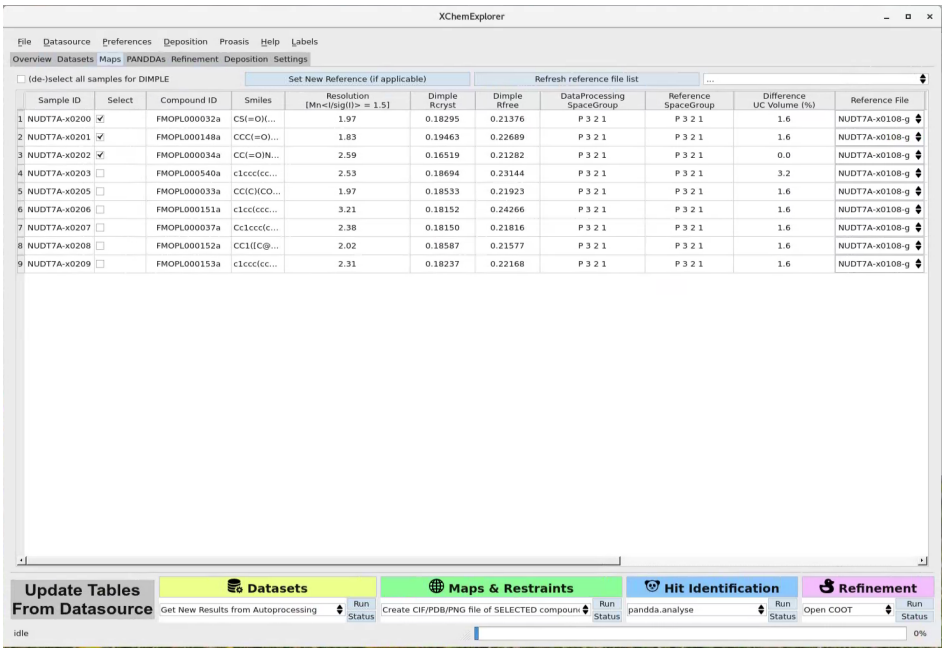


Figure 20. The Maps tab of XCE.

Finally, select '*Create CIF/PDB/PNG file of selected compounds*' from the green action box and press *Run*.

Figure 21. Green action box for creation of CIF files.

XCE will now create a folder named 'compound' in each of the selected sample directories and run the respective restraints generation program in there. It will also generate a 2D image of the compound. After this is completed, it will create symlinks of the final CIF, PDB and PNG files into the sample directory.

```
FMOPL000034a.cif -> compound/FMOPL000034a.cif
FMOPL000034a.pdb -> compound/FMOPL000034a.pdb
FMOPL000034a.pdb -> compound/FMOPL000034a.pdb
```

Merging ligand restrains with CIF file from non-standard ligand

There may be instances when your protein has already another small molecule ligand bound before you start probing it with fragments. This is no problem as long as the ligand is part of the CCP4 ligand dictionary and you are using exactly the same ligand that comes with it. However, this will become an issue once the ligand itself is new (or you have, for whatever reason, decided to generate new restraints for a known ligand). Refinement will fail in this case because REFMAC does of course have no clue what to do with this molecule. In this case, one needs to merge the two ligand dictionaries into one.

First, open the *Preferences* menu (*Edit preferences*) and at the very bottom of the page, select the CIF file of your non-standard ligand in the '*Additional CIF file for non-standard ligand*' section (see Figure 18). Next, select the samples which you want to merge in the *Maps* tab (exactly the same way as described before). Finally, choose '*Merge ligand CIF file with selected compounds*' and press *Run*.

Figure 22. Green action box for ligand merging.

XCE will now remove the symbolic link to the compound CIF file in the sample directory and prepare a merged version of the file in the sample directory with the same name. It does however not touch the original files in the *compound* subfolder!

There is only one important thing to consider before you start merging: **the ligand code of the additional ligand cannot be LIG or DRG!** Both codes are reserved for ligands generated by XCE.

Restore original CIF file

In case you need/ want to restore the original CIF file, first select the samples in the Maps tab which you want to restore (see above). Then choose '*Merge ligand CIF file with selected compounds*' from the green action box and press *Run*. Please note that this is not a requirement in case you want to merge another ligand. XCE will in this case first remove the old, merged CIF file, before doing the merging as described before.

Figure 23. Green action box for restoring the original ligand restraints.

Enantiomers

There are instances when the chirality of the ligand is not clear in advance, maybe because the compound was synthesised as a racemic mixture. After the ligands restraints are generated, XCE does now automatically check if there are chiral centres and if so, it uses *phenix.elbow* to enumerate all possible stereoisomers (Figure 24). Theses PDB/ CIF files are written to the compound subfolder as <compoundID>_S.cif, <compoundID>_R.cif, <compoundID>_SR.cif (in case of multiple chiral centres) etc.

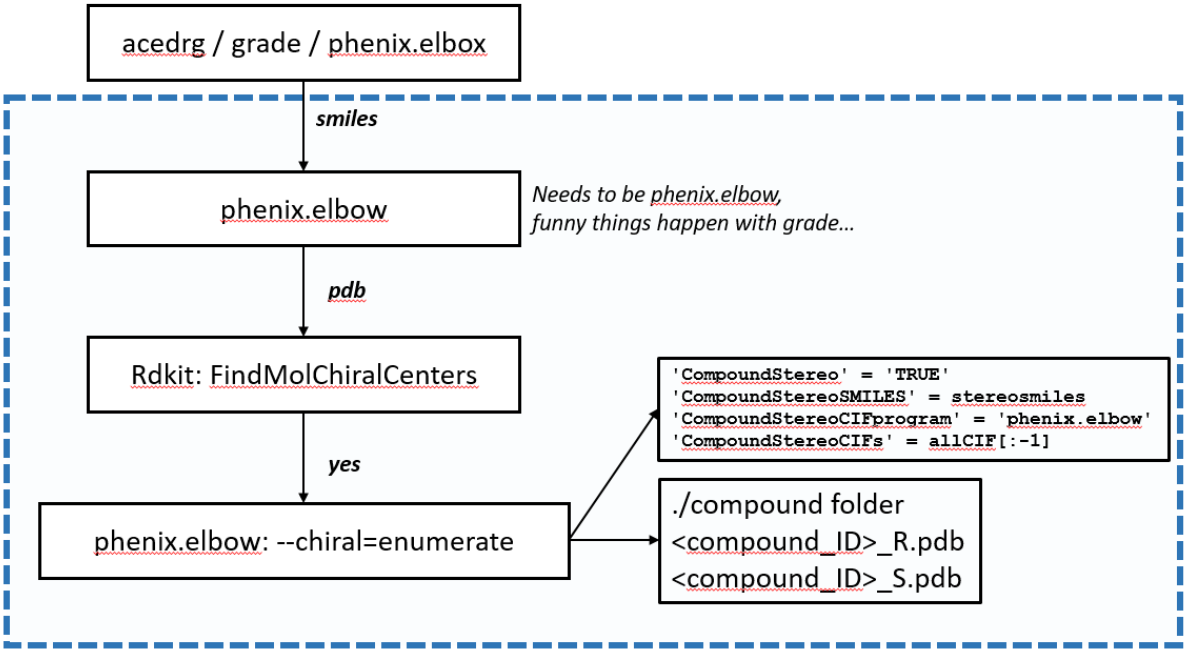


Figure 24. Enumeration of stereoisomers. Please note that the current procedure is quite complicated and therefore error prone. It will be changed once a newer version of the rdkit library is available in ccp4-python which enables enumeration of stereoisomers from smiles strings.

The different stereoisomers can be interactively selected through the XCE panel in coot (Figure 25). XCE checks for *compound* folder for a given sample and automatically recognizes all PDB/ CIF files whose file names that with <compound_ID>. It also allows selection of different solutions from automated ligand fitting (phenix = phenix.ligandfit, rhofit = global phasing program rhofit). Please note that this option is only available in 'non-PanDDA' mode!

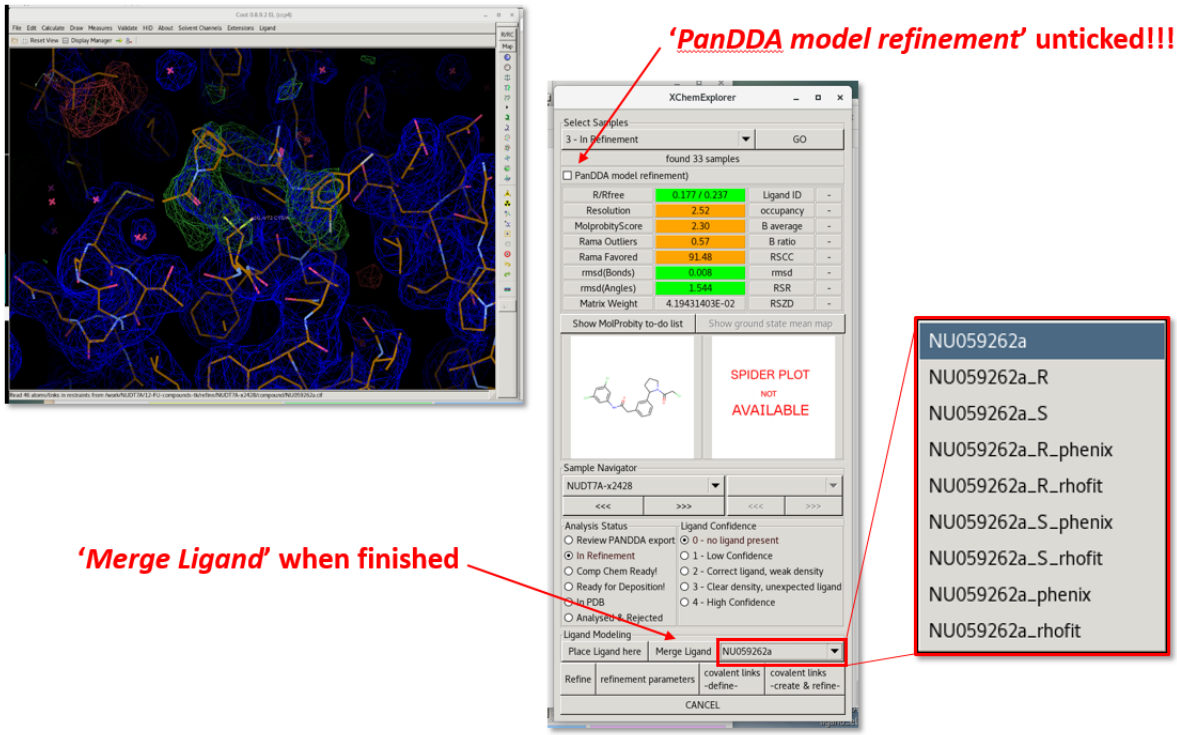


Figure 25. Interactive selection of different stereoisomers and solutions form automated ligand fitting.

PanDDA

XCE can be used to conveniently launch the PanDDA software. However, this is not necessary! One can also run *pandda.analyse* and *pandda.inspect* from the command line and only later, during the *pandda.export* stage use XCE to bring the results back into the XCE folder structure for further refinement. Also, keep in mind that PanDDA does not know about the XCE database! PanDDa only looks at the files in the specified data directory, which is the XCE project directory. The actual PanDDA analysis is performed in the PanDDA directory.

Suggested Workflow

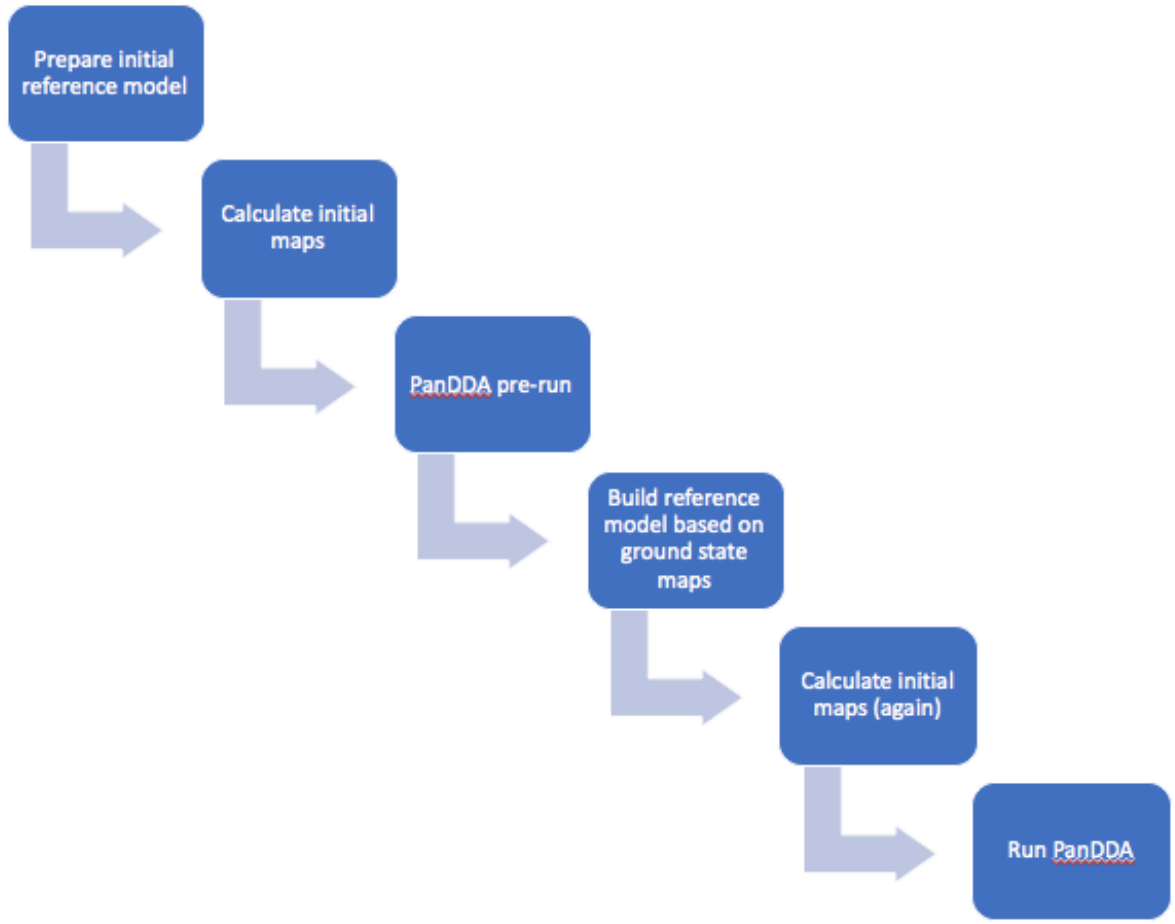


Figure 26. Step-by-step overview of the protocol required for creating a ground-state model for PanDDA analysis.

Ground-state model building

This protocol will help you to build a ground state model suitable for a X-ray fragment screening data analysis on XChem.

This method put the XChemExplorer data-management tool and on the PanDDa tool to good use. Basically, it consists in using either the XChemExplorer(XCE)-PanDDA data analysis outcome of a fragment pre-screen (100-150 fragments soaked) or a XCE-PanDDA pre-run outcome to build a

correct ground state model for the forthcoming PanDDA analysis of the full screen. It consists in six steps shown in the figure 1 above.

The ground state model represents the protein structure when no ligand is bound to it. However, It is not only a representation of the protein in its apo form. As it is calculated from data sets collected on hundreds of "ligand-free crystals" (with no ligand bound), it represents how the protein is in hundreds of ligand-free crystals. In addition, this ground state model will be used as reference model for building ligand. Updated models with ligand are "bound state" models which are a representation of the protein when a ligand is bound to it. What will be refine is the "ensemble model" which is a combination of the ground state model with the bound state model. This model explains the contribution to each structure factors in a more comprehensive way.

Therefore, modelling the right ground state model thus helps to capture and model rigorously every changed state due to a ligand binding. It also makes possible to rapidly have crystallographically correct protein-ligand models ready for computational chemistry analysis.

Initial reference model selection and quick preparation

Before you can enter the XCE workflow, you need to take a model from somewhere, *i.e.* a model straight from the PDB or from previous experiments and put it into the reference directory for calculation of initial maps. Make sure that the model is of reasonable quality and representative of the crystals used in the soaking campaign.

For example, it's always a good idea to take one of the best datasets from your solvent characterization experiment to build a good initial reference model.

A good initial reference model is a model with good stereochemistry, devoid of local errors, with good overall refinement statistics and build-in solvent molecules. However, don't overdo it at this stage, it's better to spend a bit more time later when you actually build the ground state model!

Calculating dimple maps and models in XCE

Once the initial reference model is ready, use this model to calculate electron density maps and models for all of the collected data sets using XCE ("maps" tab).

Here we are going to present you a workflow based on a PanDDA pre-run which we will not use all the available datasets but just a batch made of the first 100 data sets.

But do not wait to finish to collect all of the hundreds of data sets following a full fragment screen to build your ground state model! A good strategy is to use the data of the fragment pre-screen (100-150 fragments soaked) instead. You will have then have your ground state model ready for the full PanDDA analysis which takes longer to run!

PanDDA pre-run

After the maps are calculated, in XCE, change to the PanDDA tab and select '*pre-run for ground state model*' from the respective action box.

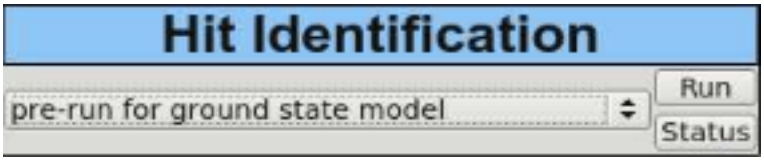


Figure 27. Action box for launching a PanDDA pre-run.

A dialog window will appear which prompts you to enter an appendix.

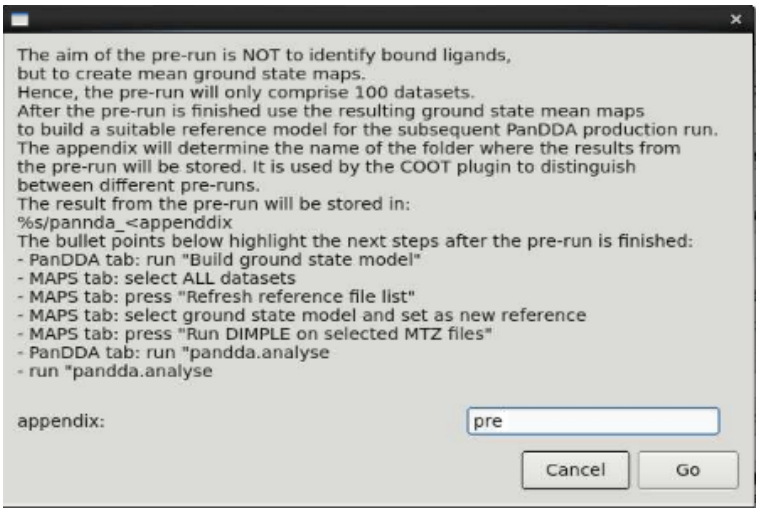


Figure 28. Dialog window with instructions of what to do after the pre-run is completed.

What will happen is: XCE will create a folder with the name *pandda_<appendix>* in the reference directory.

You can in principle launch several pre-runs, which may be useful if you are dealing with multiple crystal forms.

As the name implies, this is just a pre-run and the goal is not to identify ligands, but to get the ground state mean maps which will serve as a guide for building a proper ground state model for a later PanDDA production run. Because this is just a pre-run, this action will not use all the available datasets, but just the first 100 in order to save time.

Once PanDDA analyse is finished, XCE will automatically launch a script which will search for the most suitable reference mode amongst the 100 datasets used in the pre-run.

- It will parse the respective PanDDA logfile and look at all the datasets which belong highest resolution shell.
    - It will ignore all the datasets which contain interesting events since these have potentially ligands bound and are therefore not suitable as reference files.
        - From the remaining ones, it will take the one with the lowest Rfree and link the respective PDB & MTZ files as *<sampleID>-ground-state.pdb* and *<sampleID>-ground-state.free.mtz* into the reference folder.

It seems like a reasonable assumption to choose the ground state model according to these criteria, however, there is really no theoretical justification for doing so. If you think another model would be more suitable then simply link or copy the respective PDB & MTZ files as *<my_model>-ground-state.pdb* and *<my_model>-ground-state.free.mtz* file into the reference folder.

Make sure that the MTZ file contains columns labelled as F, SIGF and Rfree_flags.

Ground state model building

After PanDDA is finished, select '*Build ground state model*' from the respective action box.

XCE will not tell you when PanDDA is finished. The only way to find out is to check the respective PanDDA directory, in this case <reference_directory>/pandda_<appendix>. You know that PanDDA has finished once a file named pandda.done is present in the folder. If a file called pandda.errored is present you may need to check the logfiles to find out what went wrong.

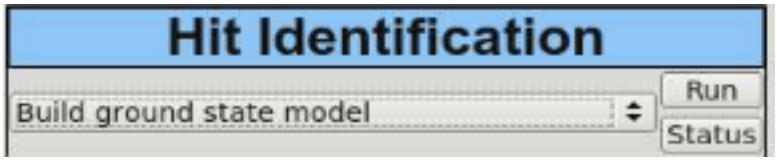


Figure 29. Action box for launching a COOT plugin which is specifically designed to build a ground state model.

This will launch a new COOT interface which allows you to modify and refine the initial reference model which was selected as described in first chapter.

Please note that no 2fofc and fofc maps will be displayed when you launch it the first time. These maps will only become available after the first cycle of refinement.

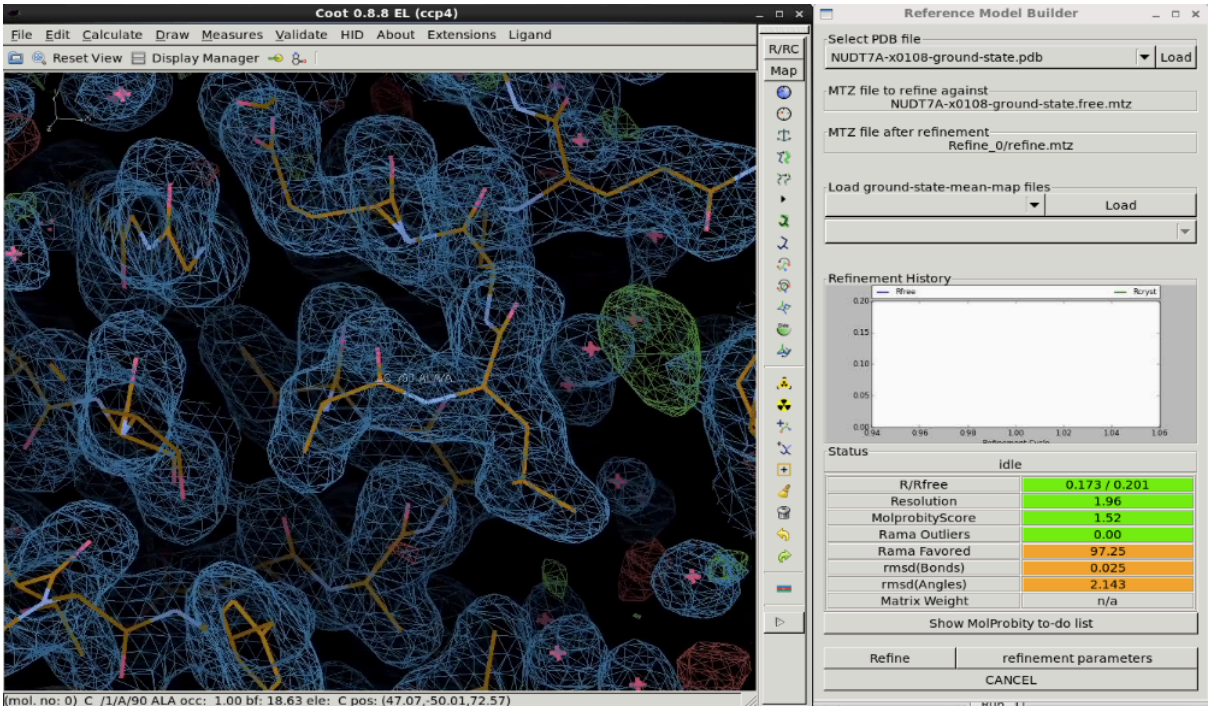


Figure 30. COOT interface for building and refinement of the ground state model.

You can now select the PanDDA pre-run from which you want to display the respective ground state mean maps. In the example given here, the folder name is panddatest.

This will take a while because it will load one ground state mean map for every resolution shell!

It will by default only display the one from the highest resolution shell, but you can use the combo box to select the respective maps from any resolution shell.

It is not yet clear if this is really a useful feature or if it just unnecessarily slows down the program; feedback is welcome!

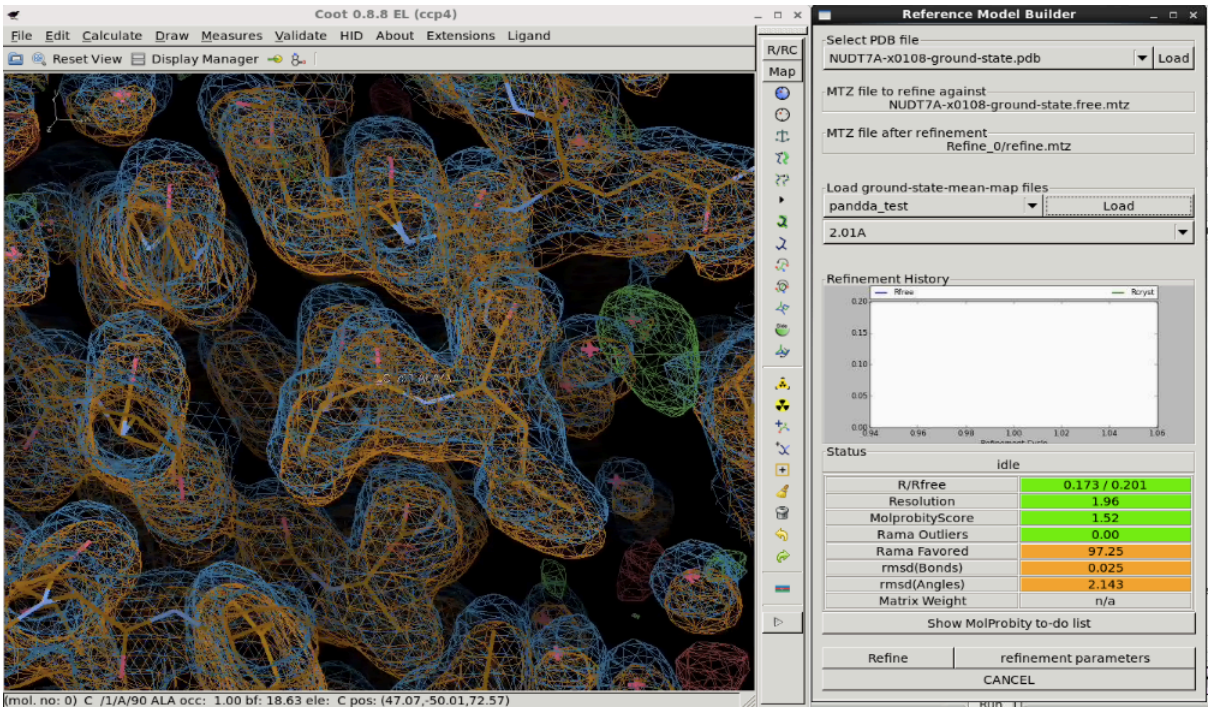


Figure 31. COOT interface with 2fofc maps (blue), fofc maps (green/blue) and ground state mean maps (brown).

You can now modify and refine the initial reference model so that it explains the ground state mean map.

Build or remove all side chains alternative conformation, all the water molecules, all the solvent molecules you have now evidence of. It is now worth spending a bit of time to build, as much as you can, a fairly accurate reference model which will then become your ground state model.

By doing so, you will save even more time in the forthcoming steps. First, you will save time during the PanDDA event inspection in pandda.inspect as less "blobs/event" due to missing features of the reference model will be found during the PanDDA analysis. Secondly, validating interesting models will also take you less time as all they all will be already well-refined after the first cycle of refinement.

Keep in mind that the mean maps are only shown so that you can understand what the mean ground state of all crystals looks like. Since this is not a crystallographic map, it does not contain any information that will be used during refinement!

Once you are happy with the first modifications of your model, adjust your refinement parameters and press '*Refine*'. The following bullet points describe what happens next:

- A new folder named *<sampleID>ground-state* will be created in the reference directory, if it does not already exist from previous refinements.
- Depending on the refinement cycle, a folder named *Refine_<cycle>* will be created in the *<sampleID>ground-state* folder and the modified PDB file from COOT will be saved as *in.pdb*.

- The modified model will be refined on the DLS cluster. A spinner will appear in the refinement panel and it will spin until the refinement is finished (see Figure 32). You cannot launch another refinement job while the current refinement is not finished.
- After the refinement is finished, the symbolic link named *<sampleID>-ground-state.pdb* will be removed from the reference directory and replaced by a link with the same name, but which points to the file that came out of refinement.
- In COOT, the old PDB file and the 2fofc and fofc maps will be deleted and the new PDB file and 2fofc & fofc maps which came out of refinement will be loaded. You can now iterate between model building and refinement until you have a model that fits your ground state maps and retains good stereochemistry, is devoid of local errors and has good overall quality statistics.

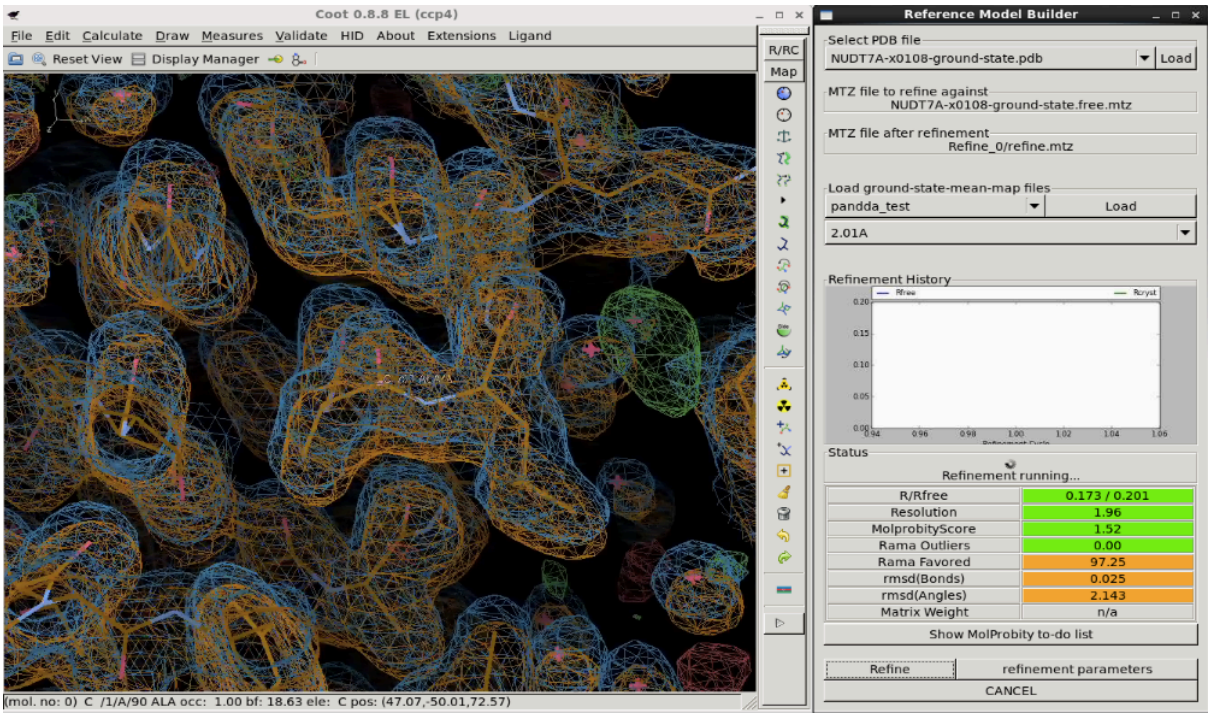


Figure 32. COOT interface during refinement. Note the little spinner in the middle of the panel which indicates that the refinement is still running.

Congratulation! You now have a correct ground state model/reference model for the PanDDA full analysis.

Map re-calculation

After you have finished building the ground state model, go back to the MAPS tab of your current instance of XCE.

We would recommend to first remove all the dimple files in order to avoid that any unwanted files remain! You can use the option in the "Maps and Restraints" action box.

The new ground state model is already in the reference folder, but XCE does at this point not know that it exists. Frist press '*Refresh reference file list*', then select the respective ground state model and finally press '*Set New Reference (if applicable)*'. Now you're good to go to calculate a new set of maps which can then be used for the PanDDA production run!

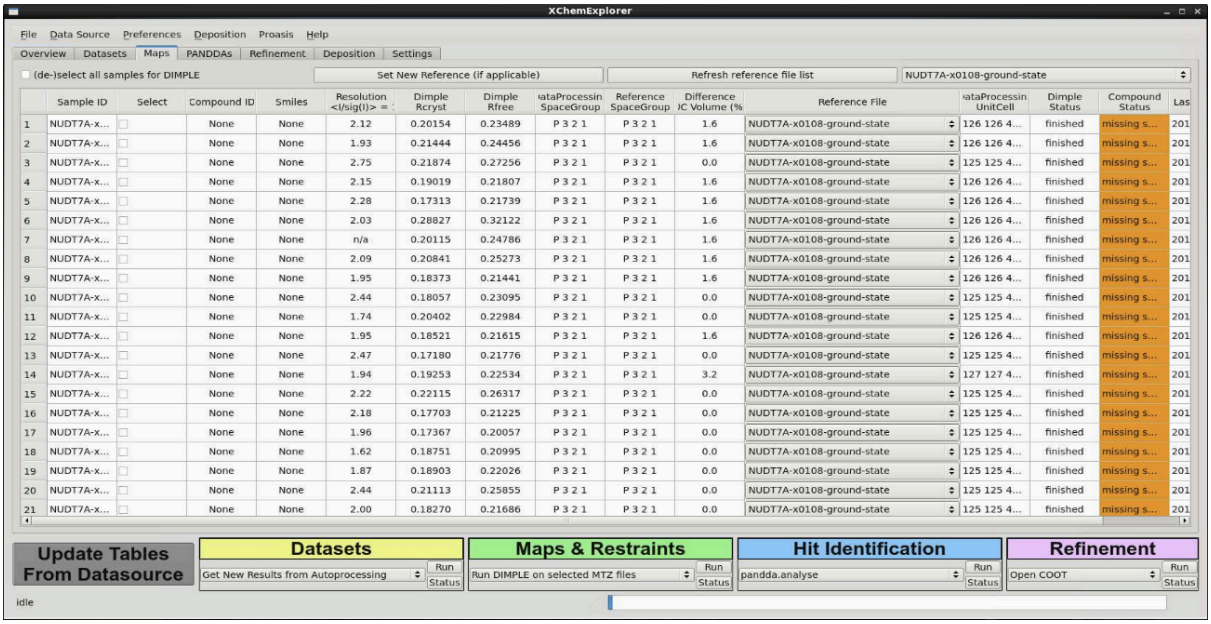


Figure 33. XCE Maps tab for calculating a new set of maps with a new ground state model.

pandda.analyse

--- coming soon ---

pandda.inspect

--- coming soon ---

PanDDA model export to the project directory

--- coming soon ---

Convert Event Maps to MTZ files

You need to convert your Event Maps into MTZ format if you want to include them into your deposition or if you want to create an HTML summary file. In order to do so, go to the PANDDA tab and select 'Event Map -> SF' from the 'Hit identification' action box (Figure 34).

Figure 34. Event map conversion.

When you press 'RUN', XCE will search all the folders in the project directory (!) and convert every event map that it finds into MTZ format. The conversion runs on the local machine, so this can take quite a while if you collected many datasets and have many event maps. Note that it currently can only convert the maps into a P1 MTZ file.

Refinement

Refinement stages

Figure 35 Refinement progress model of XCE

The figure summarises the refinement stage model that XCE uses to track the progress of each sample and which is also used to triage samples. You need at least do some initial refinement to be able to look at samples in the refinement interface.

Overview

A list of all models that are currently ‘in refinement’ can be viewed in the ‘Refinement’ tab.

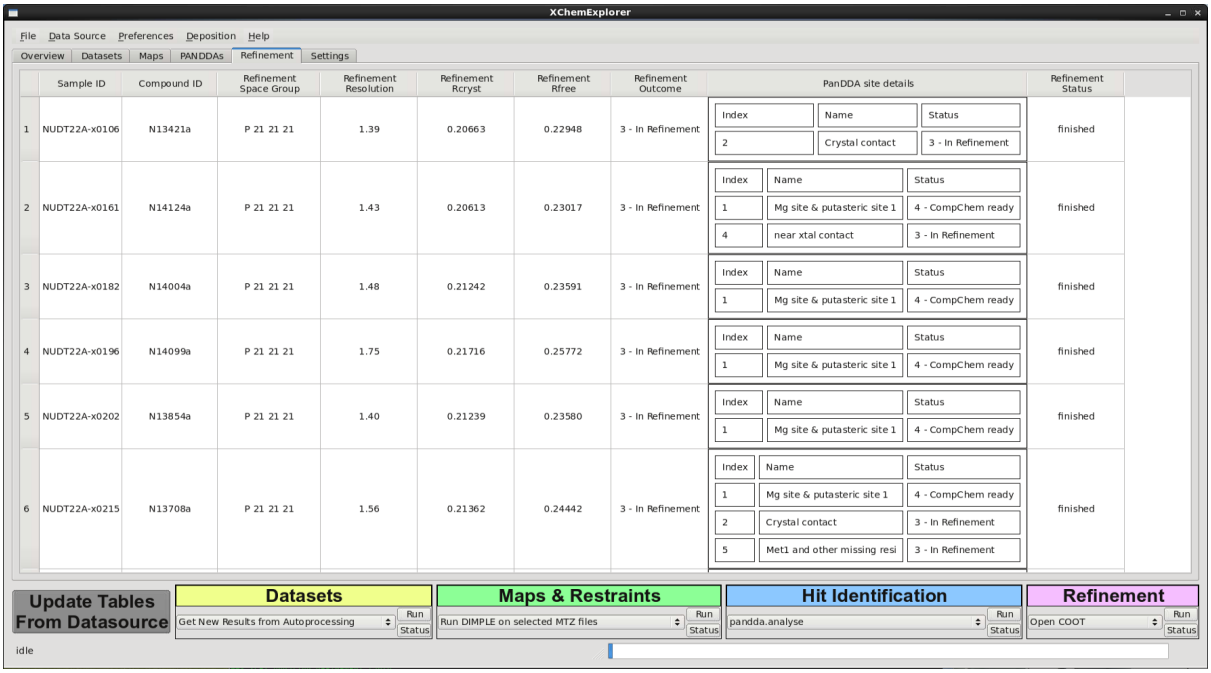


Figure 36 Refinement tab in XCE

If you want to inspect them in COOT and if necessary want to refine them further, choose ‘*Open COOT*’ from the magenta action box and press RUN. F

Figure 37 Refinement task box

This will launch COOT and the respective XCE interface (Figure 38).

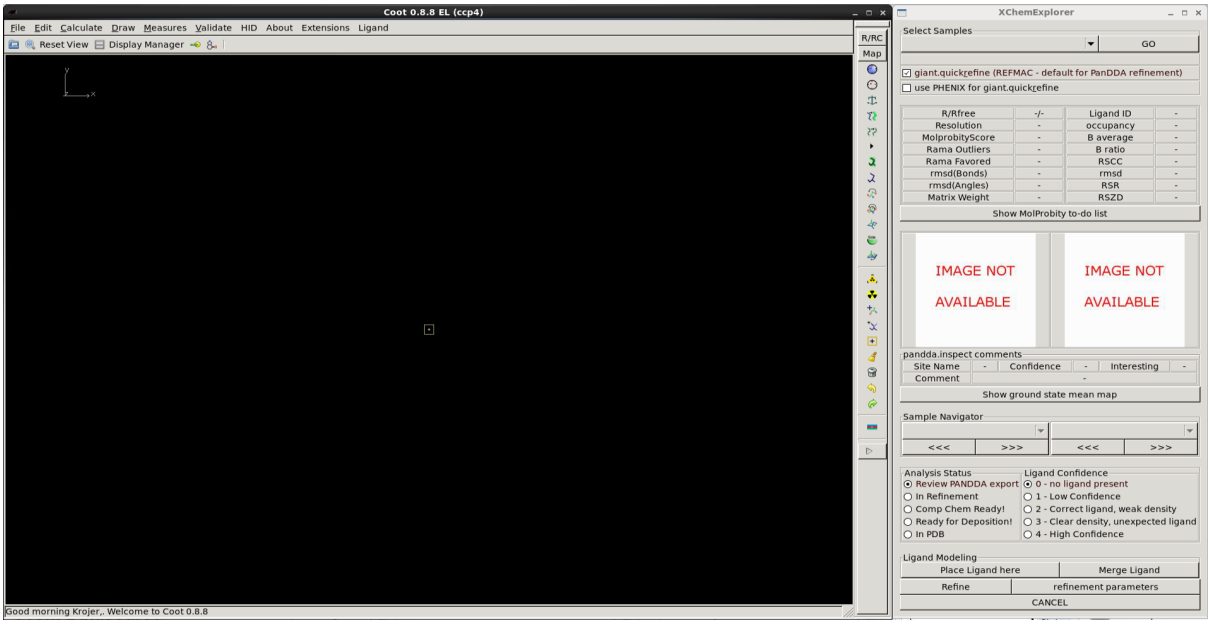


Figure 38 XCE Refinement interface for COOT

First you need to select the subset of samples you want to review/ refine from the drop-down menu at the top of the XCE interface (Figure 38). The drop-down lets you choose the Refinement Stage and once you press GO will load all samples that are in the respective Refinement stage. The image below shows you all the available categories

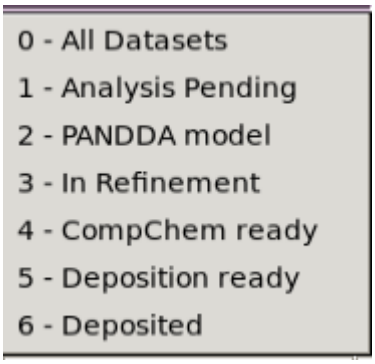


Figure 39 Sample selection criteria.

For example: if you want to load all models which are already in refinement, then select category 3 and press GO. In the example below, 79 structures are currently being refined (Figure 40).

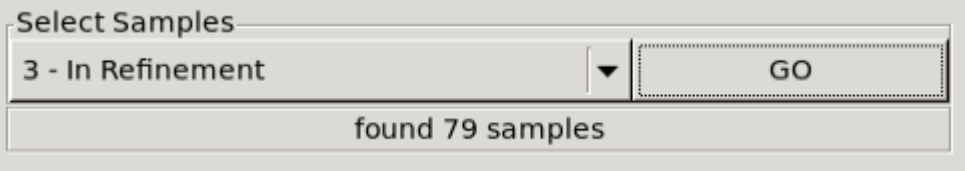
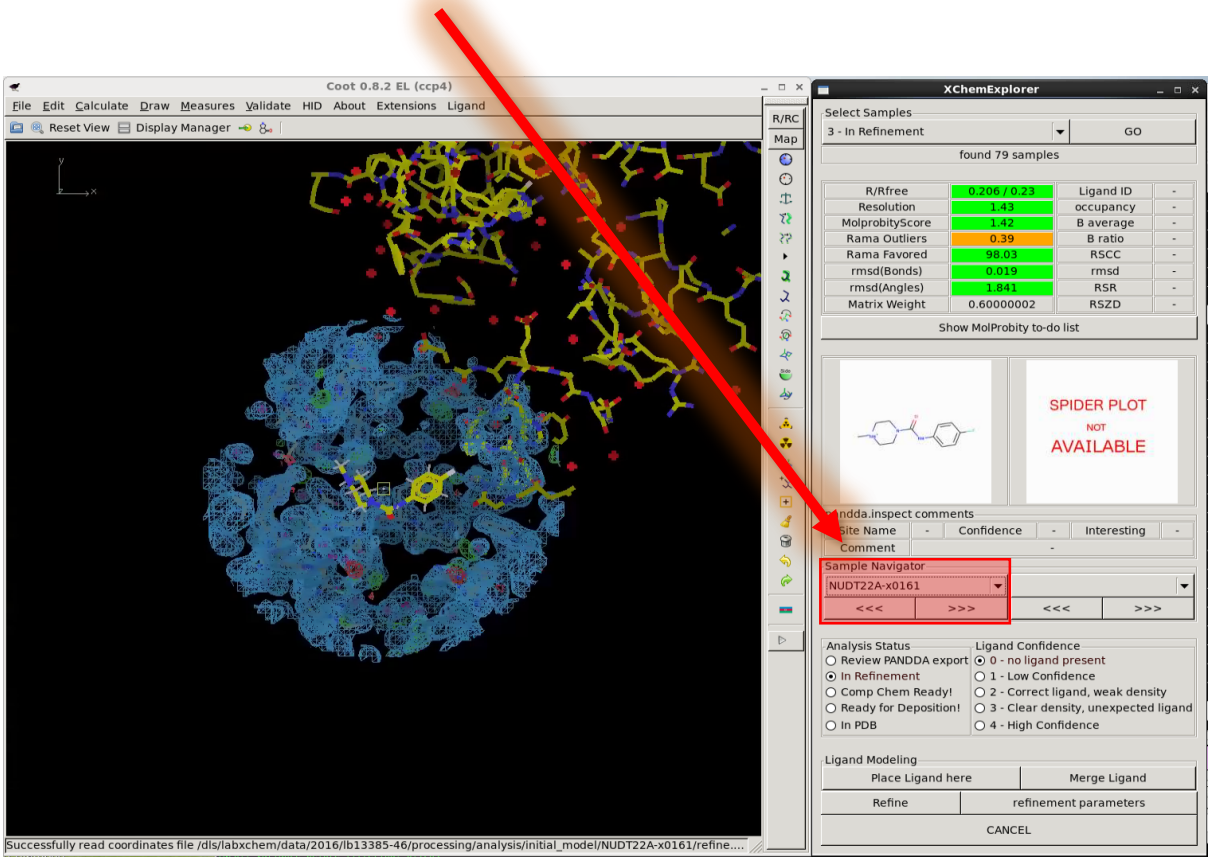


Figure 40 Exemplary reponse after sample selection.

In the left column of the 'Sample Navigator' Section, use the arrow buttons or the drop-down menu to select the structure of interest:



XCE will load the structure[1], 2fofc map (blue), fofc (green/red) map and a pdb file (+ dictionary) of the ligand if the latter was created and specified in the database. This ligand molecule may be slightly confusing because it may seem to just float in space. However, the molecule is completely

[1] Structure refers to the file called refine.pdb in the respective sample directory, or if no refinement has been carried out so far and category 0 or 1 are selected, then it will try to load dimple.pdb. Note: if you use XCE throughout the process, then refine.pdb as well as dimple.pdb are not actual files but symbolic links that point to the most recently refined file.

ignored as long as it is not merged into the main structure! It is only loaded to enable quick modelling if the ligand is not already part of the structure.

Interactive modelling of covalent bonds

If you have a situation where your ligand binds covalently to your protein, then you can model the new bond interactively in XCE. Please note that the feature currently only works when you are working in non-PanDDA mode, i.e. the '*PanDDA model refinement*' checkbox at the top of the panel needs to be un-checked! You only need to define the two atoms forming the new bond by first pressing "***covalent links – define –***" and clicking on the two atoms. Afterwards, you have to press "***covalent links – create & refine***" and XCE will make the link and initiate a round of refinement. XCE actually uses ACEDRG to create restraints for the new bond between the protein and the ligand and ACEDRG also adds them to the existing ligand restraints.

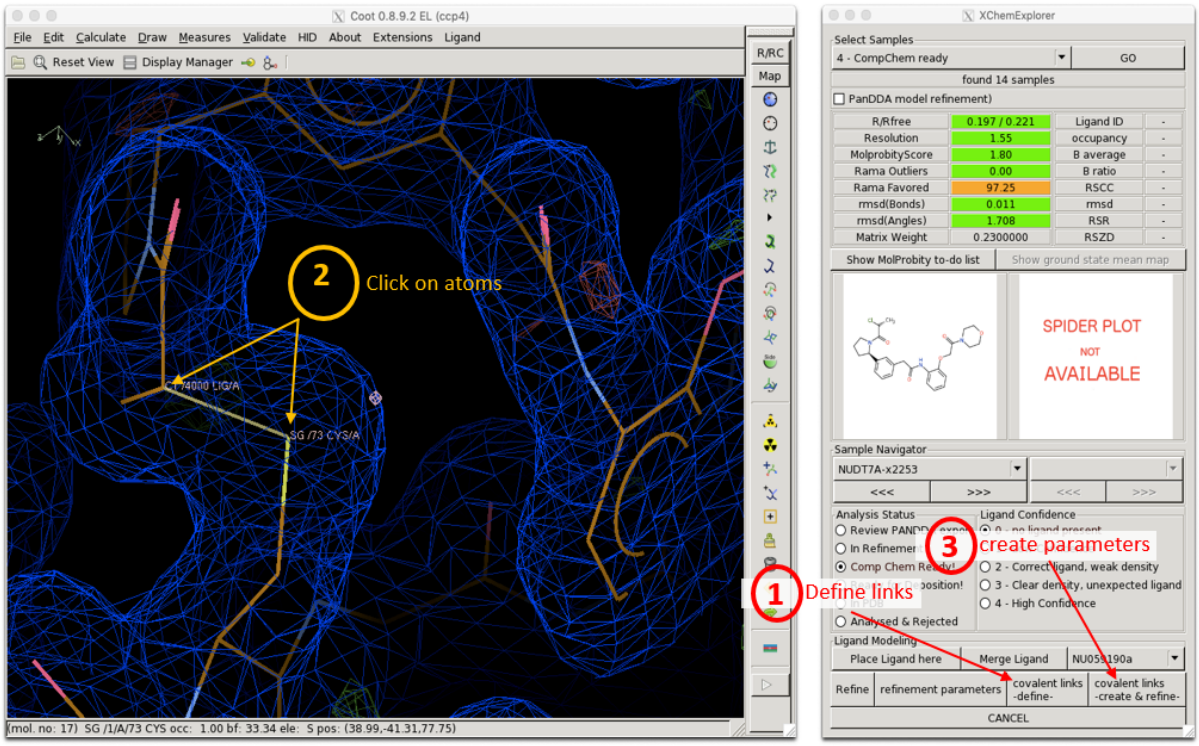


Figure 41. Interactive creation of covalent links.

Deposition

XCE can be used to generate files for deposition via the new group deposition tool at the PDB (https://deposit-group-1.rcsb.rutgers.edu/groupdeposit) (Figure 42).

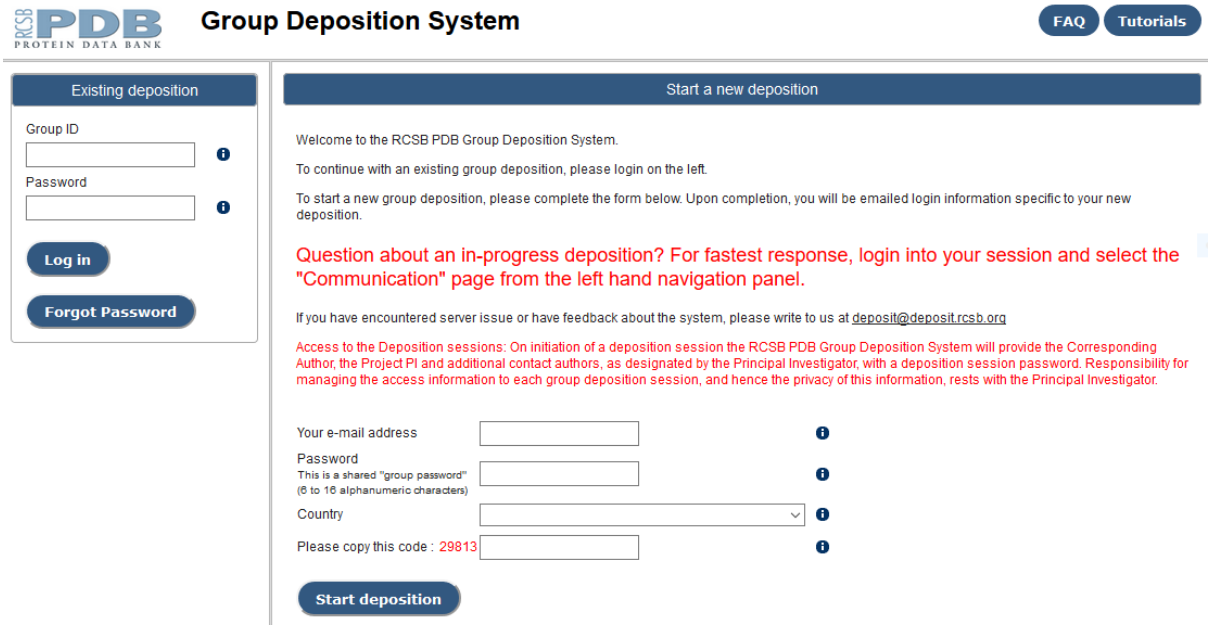


Figure 42. PDB Group Deposition website.

In order for a dataset to be considered for deposition, the *RefinementOutcome* flag has to be set to '*5 - Deposition ready*'. Once this is done, the respective sample will appear as a new line in the *depositTable* of the database.

Adding compulsory information

Every deposition needs information about title, authors, crystallization condition etc. This section describes how to enter this section. From the *Deposition* menu select '*Edit information*' (Figure 43).

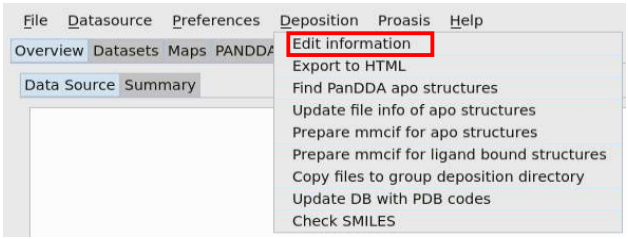


Figure 43. Deposition menu.

This will open a new window which provides a mask for input of all the essential pieces of information (Figure 44).

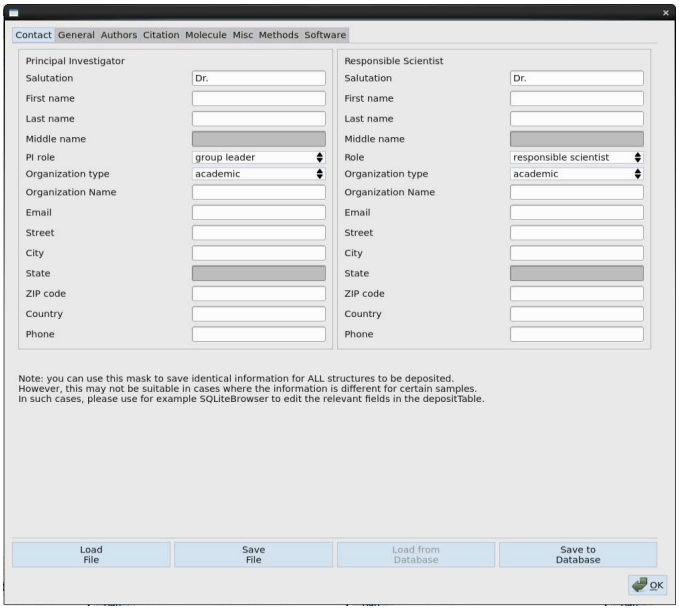


Figure 44. XCE entry widget for essential deposition information. Note that the compulsory fields have a white background whereas the fields where the user can provide additional information have a grey background.

It is possible to fill out all or part of the information and save the entered values into a file. Alternatively, it is possible to download an exemplary .deposit file from the XCE website (see Resources section in http://tkrojer.github.io/XChemExplorer). Download the file and select '*Load File*' and the fields of the widget should change:

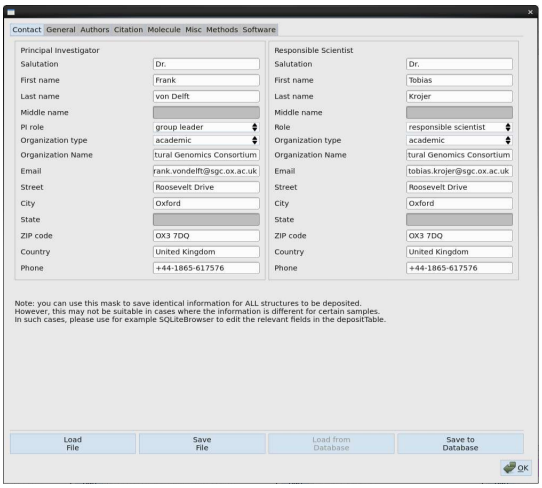


Figure 45. XCE entry widget for essential deposition information with data.

Most of the fields are self-explanatory, however, it is worth drawing the attention to the *General* tab where one can provide information about the title The given example contains two wildcards, *$ProteinName* and *$CompoundName*. The *$* sign indicates that XCE will fill in whatever is given as ProteinName and CompoundName in the database. While the former will most likely be the same for all the entries so that one can provide the specific protein name instead of a wildcard, it is useful to provide the compound name one wants to see in the title of the PDB deposition in the database. Once all the information is entered, use '*Save file*' to save a blueprint in case something needs changing, then press '*Save to Database*' in order to update all the entries that are currently in the deposition table. Note that if you add more samples later, you will again have to press again '*Save to Database*'.

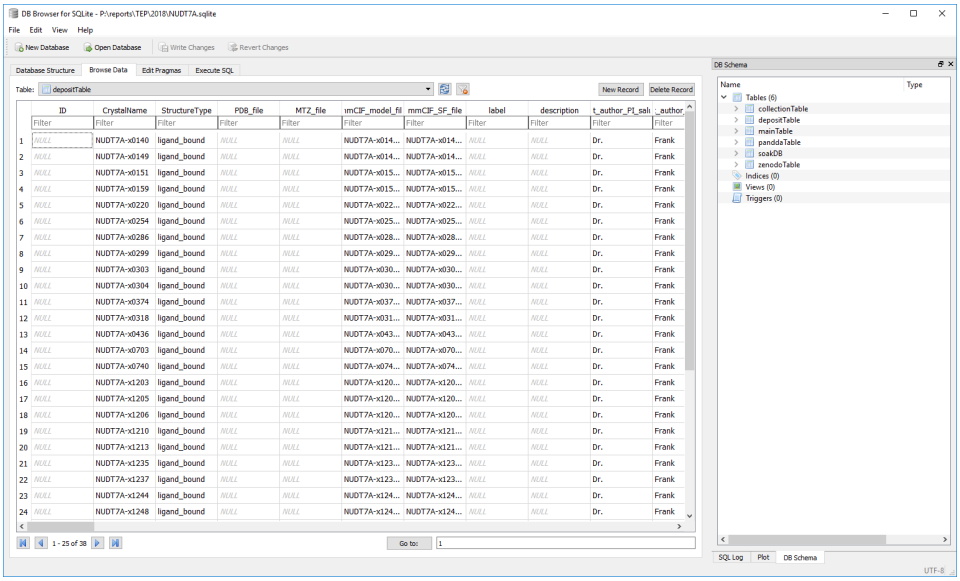


Figure 46. depositTable of the XCE database.

The interface obviously assumes that the information will be the same for every entry. But depending on the project, this may not be the case. Therefore, if you want to edit information for a specific entry, this is best done through the SQLite browser tool (Figure 1). Navigate to the *collectionTable* and then change the information as you see fit. Do not forget to press '*write changes*' once you have finished! But remember, if you add more samples later and you press '*Save to Database*', all the changed information will be overwritten. This is not ideal and will be addressed in one of the next software iterations.

Preparation of files for PDB upload

XCE will create a single gzipped tar file for direct upload into the Protein Data Bank. The *Group Deposition Directory* specifies the location where the file will be written to (see Settings, page 7). The next step is rather simple, from the *Deposition* menu choose '*Prepare mmcif for ligand bound structures*' and XCE will start the prepare separate mmcif files for the PDB file and MTZ files for each entry in the deposition table. Note that if this deposition is part of a PanDDA analysis, then the program will also add Fourier coefficients for the event maps to the structure factor mmcif file. This may take a while depending of the number of files that are earmarked for deposition.

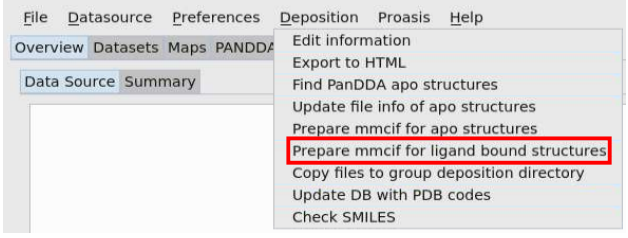


Figure 47. Prepare mmcif for ligand bound structures.

At the end, all mmcif files will be combined into a gzipped tar archive file, which can be directly uploaded into the PDB. At the moment, the only way to follow the progress of the process is by monitoring the output in the terminal window.

Troubeshooting

```
grep ERROR xce.log
```

Note: the XCE GUI does currently contain several obsolete items, which will be removed during the next software iteration.

Summary

PREREQUISITE:

All pandda event maps need to be converted to MTZ format:

Hit Identification: Event Map -> SF

PREPARATION:

1. Set crystals to '5-ready for deposition' in XCE refinement tab
2. Edit deposition information
3. Deposition menu -> Prepare mmcif for ligand bound structures
4. Deposition menu -> Copy files to group deposition directory (ligand bound)

Deposition of ground-state model

You can now deposit the ground-state PDB file as a single PDB entry. This PDB entry will contain the coordinates of the ground-state model, the ground-state mean map and all the structure factors of every (!!!) dataset that was used to do the PanDDA analysis. This means that everyone can now easily reproduce your event maps. You can find some more information here: https://openlabnotebooks.org/update-on-batch-deposition-of-xchem-structures

Prerequisite:

All apo MTZ files need to be converted to mmcif format:

Hit Identification: apo -> mmcif

Preparation:

1. Follow instructions in Deposition tab
2. Deposition menu -> Copy files to group deposition directory (ground state)

Tips & Tricks

DLS remote connection

NX is a great way of working remotely at Diamond. However, it is not always ideal if you have to do extensive rebuilding in *pandda.inspect* or during refinement. The method described here uses *Filesystem in Userspace* (FUSE) to recreate the Diamond file system at your home institution. All you need is a reasonably fast internet connection and a friendly IT team:

First, your IT team (unless you use your private computer) needs to install FUSE (https://github.com/libfuse/libfuse):

e.g. **`sudo yum install fuse-utils sshfs`** (for Centos distributions)

or **`sudo apt-get install sshfs`** (for Ubuntu)
See https://gist.github.com/cstroe/e83681e3510b43e3f618 for details (note that FUSE is also available for Mac). Avoid building from source unless you really have to.

Then, they need to create a /dls mount point:

e.g. **`sudo mkdir /dls`**

I am not sure if the next step is always necessary, but they had to change the ownership of the directory at SGC (Note xtalnmr is the group I belong to):

e.g. **`sudo chown tkrojer:xtalnmr /dls`**

Your IT team needs to set this up, but from now on, you can mount and unmount the Diamond file system whenever you want. You can mount the Diamond file system with the following command:

`sshfs -o reconnect <fed_id>@nx.diamond.ac.uk:/dls /dls`

e.g. **`sshfs -o reconnect zqr16691@nx.diamond.ac.uk:/dls /dls`**

And that's it. Now go into your labxchem visit and fire up XCE!

Note: if you cannot convince your IT team that this is a good idea, an alternative is to run linux through a virtual machine. Then you are ROOT and can do whatever you want!

Another thing to consider: initially, we described using sshfs without the '-o reconnect' option and while FUSE seemed to uphold the connection while someone was working, it seems to drop it if the user is away from keyboard for a while. The reconnect option seems to avoid this. However, it also seems to become less responsive if the system was idle for a while. Another option that I found on the internet is to enable ServerAliveInterval in the client's (your workstation) ssh_config file; i.e. open /etc/ssh/ssh_config and then append/modify values as follows:
ServerAliveInterval 15
ServerAliveCountMax 3

However, we have so far not tested this option!

Now go to your labxchem directory, then launch XCE.

e.g. **`cd /dls/labxchem/data/2017/lb18145-3/processing`**

But before you start, here are some considerations of how to run XCE:

Option 1 is to install XCE locally. You can find download and installation instructions on GitHub:

https://github.com/xchem/XChemExplorer

Several people have done this and it seems to work well. The only dependency is that you need to have ccp4 v7.0 or higher installed on your system.

Option 2 is to launch the DLS version of XCE on your machine. We haven't tested this extensively at this point, but it would be the preferred mode because you will then not have to worry about updating XCE etc. In principle, all you need to do is run the following command:

`/dls/science/groups/i04-1/software/XChem/xce`

This works, but there are some problems:

(i) It seems to take a very long time to launch XCE and pandda.inspect, i.e. 10 and 5 minutes, respectively, on our system. But once the programs are up and running they are as responsive as usual.

(ii) The xce-coot plugin did not work on our system due to a library mismatch. However, pandda.inspect, which uses its own ccp4 distribution at the moment does work without problems. It takes around 5 minutes for the gui to start, but once that's done it works nively.

Note: XCE and COOT now run on your local machine, but the files are pulled from Diamond and written back to the Diamond file system. Depending on the speed of your internet connection, it may take a bit to load big map files. However, you will see it's worth it!

And once you're done, unmount the file system:

`fusermount -u /dls`

Using XChemExplorer outside the labxchem environment

XCE was over the years optimised to support crystallographic fragment screening at the DLS XChem facility. It is however a generic tool and it will accept any data as long as they are organised in a certain hierarchical manner.

If you collect data at the Diamond Light Source, just make sure that you have set *${proteinname}/${samplename}* as your data collection directory and you are good to go. XCE is currently only able to parse the auto-processing results at DLS, but it should not be too difficult to adapt it to other synchrotron sites. Just get in touch in case you think this could be useful for your research!

The following section will describe what you need to do if your data are already processed and you want to import them into XCE.

Project Directory structure

XCE is essentially a file system based tool. It uses a SQLite database to record results and outcome and to track progress, but it can restore most of the information from the file system alone. Not everything of course, but enough to continue refining and depositing crystal structures. Therefore, as long as the data adhere to a few conventions and are present in the expected XCE directory structure, they can be imported. Please also check the

Database section for how to generate and edit a new database.

All files are stored in the same project directory and have to be organised in a hierarchical manner, i.e. files belonging to one crystal are stored in a subfolder and the name of this folder is equal to the sample identifier. The filenames have to adhere to a naming convention, but the requirements are minimal and it is easy to add data manually by adjusting folder and file names to the expected nomenclature. All that needs to be provided are an MTZ and an AIMLESS logfile in the respective sample directory. The files must have the same filename root as the folder name, e.g. if the folder name is TEST-x001, then the MTZ and logfile must be called TEST-x001.mtz and TEST-x001.log, respectively.

The following example will illustrate the data structure further:

The project directory structure of XChemExplorer is as follows:

<project_directory>/<sample_id>

e.g.

/Users/tobiaskrojer/SGC/PHIPA/fragment_screen/PHIPA-x001

/Users/tobiaskrojer/SGC/PHIPA/fragment_screen/PHIPA-x002

/Users/tobiaskrojer/SGC/PHIPA/fragment_screen/PHIPA-x003

etc.

Each sample folder needs to contain a MTZ file and the corresponding AIMLESS logfile, e.g.:

PHIPA-x001.mtz

PHIPA-x001.log

XCE uses a file called <sample_id>.free.mtz as input for refinement. This file is either automatically generated by the DIMPLE difference map pipeline or users can choose to append an existing Rfree set from a reference file by providing it in the reference folder. If you have already done some refinement and you have already an Rfree set in your MTZ file, then just make a symbolic link, e.g.

ln –s PHIPA-x001.mtz PHIPA-x001.free.mtz

MTZ column labels must have CCP4 default names, otherwise XCE may show unexpected behaviour, i.e. IMEAN, SIGIMEAN, F, SIGF, FreeR_flag. Additionally, the program can only parse AIMLESS logfiles at the moment.

After DIMPLE was run successfully, the resulting PDB and MTZ files will be linked as dimple.pdb and dimple.mtz into the respective sample directory.

Once the refinement stage is reached a subfolder for each refinement cycle will be created: Refine_<cycle number>. This subfolder contains the modified PDB file, executable shell script and output. The script contains the complete refinement and validation schedule. After successful refinement, the resulting PBD and MTZ files will be linked as refine.pdb and refine.mtz into the sample directory. Again, you can provide the files yourself!

It is possible to create this folder structure manually and then choose Data Source -> Update Data Source from filesystem from the XCE menu to import all the information into the database.

This is all you need to do. It does not matter if these are real files or symbolic links. Exemplary sample folders could look like this once you are finished:

XXX

Updating the database

The database will already contain most of the information after you ran 'Update Datasource from filesystem'. But if you want to edit or add more information like compound IDs or smiles strings (or anything else) then refer to Database update (page 13).

Running PanDDA

--- coming soon ---

Refinement

--- coming soon ---

Deposition

This chapter explains how to prepare data for deposition in case this is the only task you want to use XCE for. You need to prepare your data as described before, most importantly, the final PDB and MTZ files need to be present as *refine.pdb* and *refine.mtz* in the respective sample directories.

The only things left to do before you can continue with depositing your data as described in the

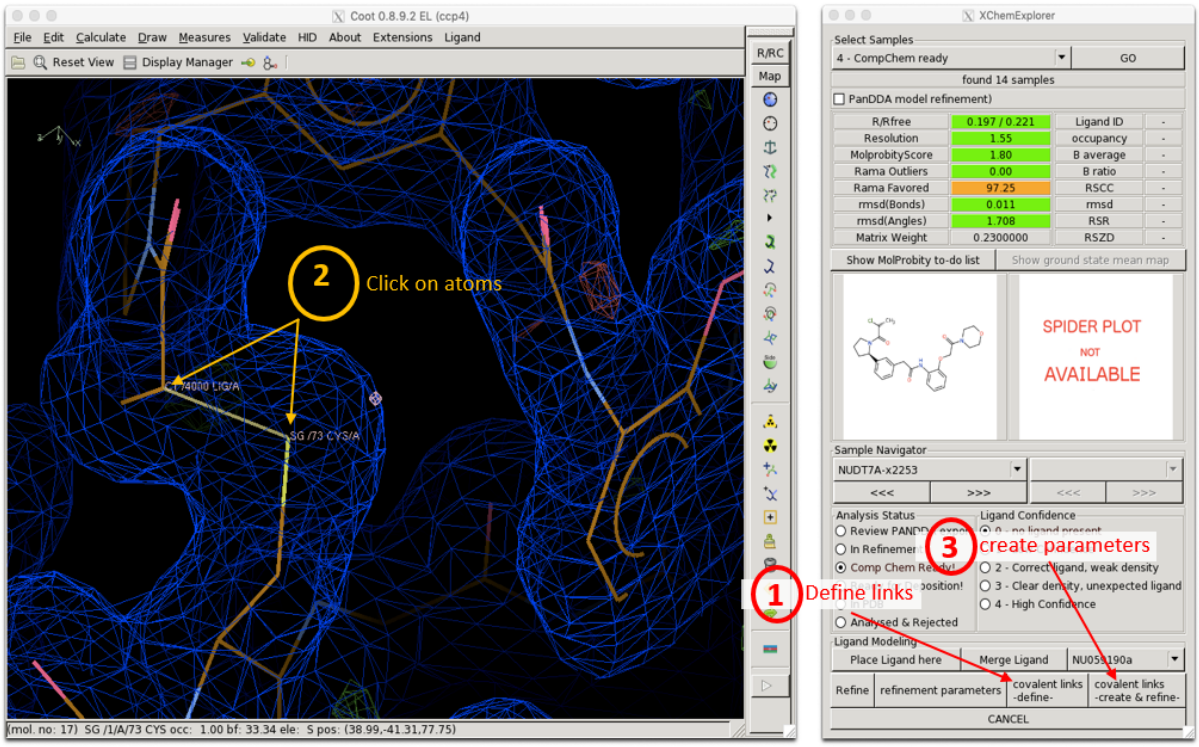


Figure 41. Interactive creation of covalent links.

Deposition (page 44), is to set the *RefinementOutcome* field in the *mainTable* of the database to '*5 - Deposition ready*'. This can be achieved by either manipulating the database with SQLite browser or by looking at your data in the XCE COOT interface and setting the *RefinementOutcome* field there.

Frequently asked questions

What happens when you step through the models?

XCE will load a file called refine.pdb (or dimple.pdb in case no refinement has been carried out so far) from the sample directory and if available a pdb file of the ligand and the respective restraints. Additionally, 2fofc as well as fofc maps are loaded (or they are calculated on the fly from refine.mtz/ dimple.mtz if the map files were for whatever reason no pre-calculated). Note that refine.pdb/mtz and dimple.pdb/mtz are therefore reserved file names. Hence, if you wanted to manipulate your model outside the XCE environment you can easily do so and XCE will read the manipulated model in as long as it's called refine.pdb and present in the respective sample directory. One thing to keep in mind though: XCE carries out the actual refinement in a subfolder called Refine_<cycle number> and only links the resulting pdb/mtz files as refine.pdb/mtz into the project directory. Every time a new refinement is launched, it will first delete the respective symbolic links. But it will also delete it if it is an actual file. So if you want to keep the original, better create a symbolic link called refine.mtz. When you go through the models, it will remove all currently loaded models and load the aforementioned files from the next sample directory.

What happens when I make changes to the model?

All changes that you make to the model called refine.pdb will be preserved if you launch refinement (see next point). They will however be lost if you go to the next dataset. XCE will currently not ask if you want to keep the changes, the pdb file will be deleted from the list of molecules in COOT and it will be lost forever! Also, be careful if you read in additional molecules with the same name, for example if you want to analyse something. COOT does not mind if molecules have the same name since every molecule that is read in get a unique internal identifier. XCE however recognises molecules by filename it may get confused in case of duplicates.

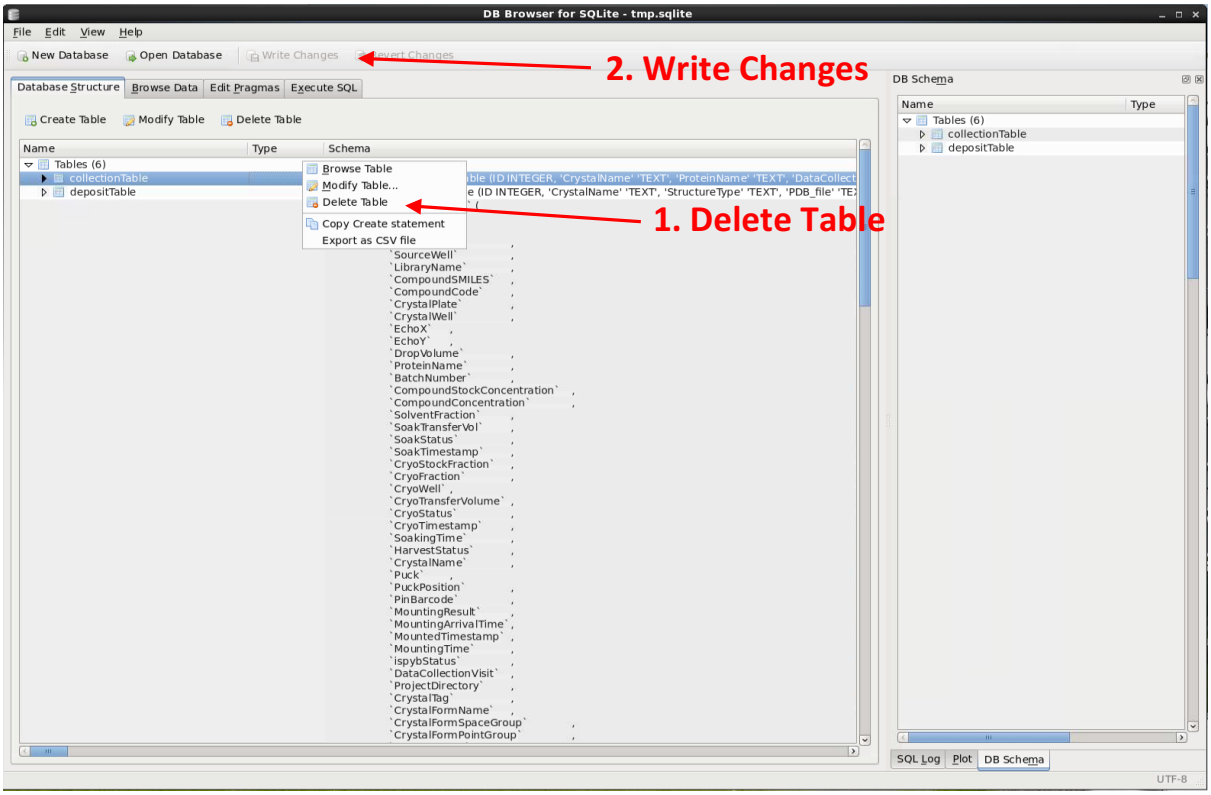
What happens when I refine the model?

When you press refine, XCE will take the model called refine.pdb and save it to a subfolder called Refine_<cycle number + 1> as in.pdb together with the shell script that will be used for refinement. If something goes wrong and the reason for failure is not clear it is usually a good starting point to look at the logfiles in the respective folder. Keep in mind that if you added for example water or other solvent molecules and did not merge them into refine.pdb, then they will not be included. XCE does not do any automatic merging!

I have wiped the contents of the project (initial_model) directory, but '*Get new results from autoprocessing*' does nothing when I run it again

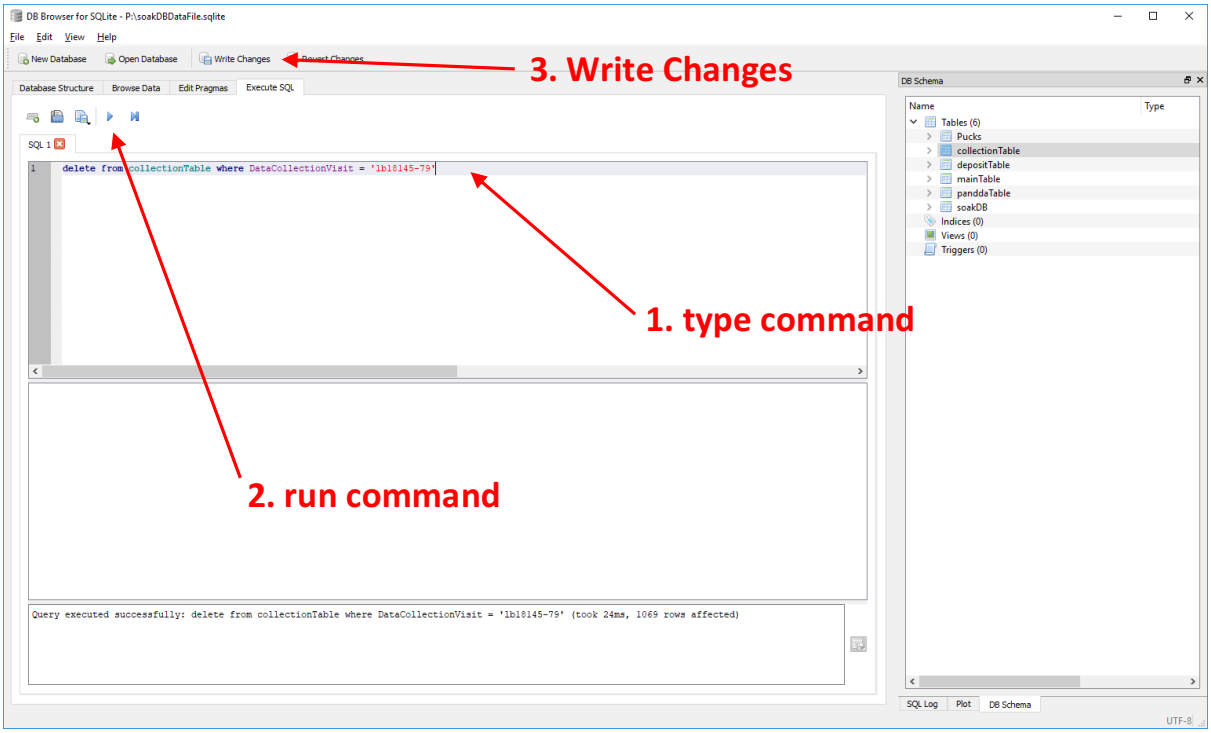
There may be instances when something gets messed up and a fresh start seems like the only option. Please note that if this is because XCE did something funny, let us know! You can delete the contents of your project directory at any time and as long as the data have not been backed up at Diamond, you will be able to recreate the folders. However, once you have run '*Get new results from autoprocessing*' and all the files are copied over, XCE will not do anything after you have wiped the project directory. This is because XCE will first query the *collectionTable* in the database and check which autoprocessing results have already been copied over. It does not check if the actual files in the project directory exists, at this point. The only thing that matters for XCE is if an entry for a particular autoprocessing pipeline exists in the *collectionTable*. The reason why it works like this is because running an sqlite query is really fast, whereas checking if files exist is rather slow. So in order to copy the files over again, you can do two things:

Remove the complete collectionTable (if your project directory is empty)



Open your database file with SQLitebrowser. In the 'Database Structure' tab, right-click with the mouse on *collectionTable*; select 'Delete Table', then click 'Write Changes'.

Remove all the autoprocessing results from a particular visit (if you only deleted folders/ crystals belonging to a certain data collection)



Open your database file with SQLitebrowser. Go to 'Execute SQL' tab. Then enter the command below; just change the visit (green, below) to whichever visit you want to delete:

```
delete from collectionTable where DataCollectionVisit = 'lb18145-79'
```

Run the command and 'Write Changes'.

I cannot see my model in COOT after pandda.export or refinement!!!

If you expect to see a refined model, but the nothing shows up when you select the respective crystal in the COOT interface, or if COOT automatically jumps to the next crystal, then the first thing to do is to go into the respective sample directory to find out what went wrong. If you work in PanDDA refinement mode (which is the default), COOT expects two PDB files:

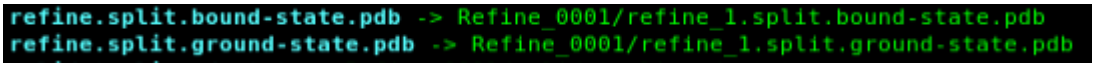
`refine.split.bound-state.pdb`

`refine.split.ground-state.pdb`

However, if the files are not present, COOT will immediately jump to the next crystal in the list until it finds a set of valid files. Be careful, the files are in fact symbolic links, which point to the files in the folder for the latest refinement cycle. XCE will create the links regardless if refinement was successful, i.e. in case refinement did not work the links will be broken. Type the following command in the sample directory to see if everything is OK:

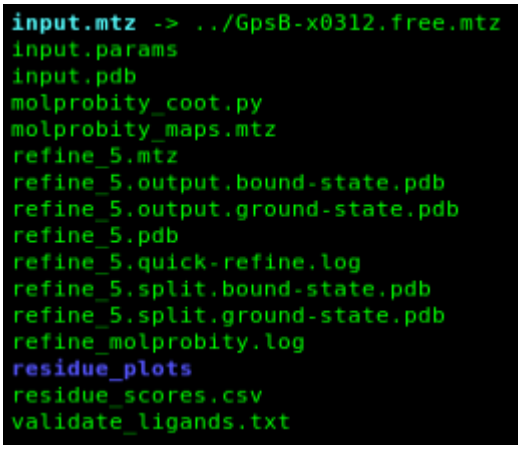
`ls -lh --color=tty`

It should look like this:



However, the links are broken if it looks like this:

Something went wrong during refinement cycle 6 (Refine_0006) in the example above. This folder is the first thing to check in order to find out what might have gone wrong. The folder should contain the following files and subfolders:



Start with checking the respective the quick-refine logfile in the respective folder (refine_6.quick-refine.log). However, this may not always be informative, because the actual error might have happened earlier.

Hence, before we continue, we need to explain what actually happens under the hood when you press 'REFINE':

First, XCE determines what the last refinement cycle was and then it creates a new folder in **`cootOut/Refine_<current_cycle>`** accordingly. It then merges the current ground-state and bound-state model and saves the resulting model as **`multi-state-model.pdb`** into this folder. Next, XCE runs **`giant.make_restraints multi-state-model.pdb`** in order to generate the required occupancy restraints. Afterwards, XCE creates a shell script (refmac.csh) which contains a list of instructions regarding refinement and validation and which is usually (at least if you process your data at Diamond) submitted to the cluster. Hence, the current refinement folder in cootOut is usually the place to start troubleshooting: check if all the files are there and look at the logfiles. Sometimes the logs can be quite clear, but they may also contain a long list of cryptic python errors. Although this may not look very useful, it at least gives us an idea where to start.

What happens next?

After the script is submitted to the cluster, **`giant.quick_refine`** will create another **`Refine_<current_cycle>`** folder (but with *four* digits) and this folder contains another set of logfiles, as well as the final (split) PDB and validation files. At the very end of each refinement cycle, the script will create the above mentioned symbolic links. Again, please keep in mind that the links are created regardless if refinement was successful or not, which is why they can be broken, but

might still give you the illusion that everything is OK. Hence, this folder is the next place to look for potential problems.

How to continue?

Now might be the time to post to xchembb or talk to your local contact. However, you can also try mending the problem yourself, or in case the problem is due to an XCE bug and a new version is available you need to manually reset the affected sample directories. There are different options available, but first, delete all the broken symbolic links and temporary files:

```
rm -f refine.pdb refine.mtz refine_molprobity.log refine.output.bound-state.pdb
refine.output.ground-state.pdb refine.split.bound-state.pdb refine.split.ground-
state.pdb validate_ligands.txt validation_summary.txt
```

One option is to revert back to the last successful refinement cycle and then try again. This is the way forward in case a new XCE version became available that addressed a specific bug. For example, let's assume that something went wrong in refinement cycle 6. After you have removed the links, go back to cycle 5 (or any other successful cycle) by recreating the links:

```
ln -s Refine_0005/refine_5.split.bound-state.pdb refine.split.bound-state.pdb

ln -s Refine_0005/refine_5.split.ground-state.pdb refine.split.ground-state.pdb

ln -s Refine_0005/refine_5.pdb refine.pdb

ln -s Refine_0005/refine_5.mtz refine.mtz
```

You will now see the models again in COOT and will be able to launch refinement.

Another possibility is to remove the links *and* the failed refinement folder (Refine_0006 in our example). Then change into **`cootOut/Refine_6.`** You can try modifying the refmac.csh file and run it locally or on the cluster and see what happens. Or, launch the **`giant.quick_refine`** command as present in the script from the command line. The latter option gives you immediate feedback, which can be very helpful for trouble-shooting.

"-bash: ccp4-python: command not found"

This error may come up after you installed XCE on a MAC. The issue might be that, while ccp4 is installed and up to date, it is not sourced in your bash_profile. In your home directory, you will find a hidden file called:

`.bash_profile`

Open it with any text editor and add the following line (there a tons of instructions on google about how to show hidden files btw)

```
source /Applications/ccp4-7.0/setup-scripts/ccp4.setup-sh
```

At least that's the installation location on my mac. Just check the /Applications folder in case it does not work. Then open a new terminal and try again.

Known Problems

Bash vs C-shell

XCE was mostly tested in the bash shell. It should work in any of the C-shells, but there could be problems when XCE executes scripts in a subshell.