

2022

CX-ASAP

A LIBRARY OF MODULES TO ASSIST WITH THE HANDLING OF LARGE
CRYSTALLOGRAPHIC DATA SETS

AMY J. THOMPSON, KATE M. SMITH, DANIEL J. ERICSSON, JACK K. CLEGG, JASON R.
PRICE

Table of Contents

Introduction.....	3
Quick Start Guide.....	4
Requirements	6
Operating system requirements.....	6
Software requirements	6
Linux - Instructions for setting up a Linux Virtual Machine	7
Linux - Instructions for installing Python.....	10
Linux - Instructions for putting software in your path.....	12
Linux - Common Errors	12
MacOS - Instructions for installing Python	13
MacOS - Instructions for putting software in your path	14
MacOS - Common Errors	14
Windows - Installation Instructions	15
Windows - Common Errors	16
Additional notes for XDS Installation	17
Installing XPREP on a Mac	20
GitHub Configuration	22
Installation.....	23
CX-ASAP Installation.....	23
Testing Your Installation	24
Reporting Errors.....	24
Using CX-ASAP	25
Command-line Interface	25
Configuration Using ‘conf.yaml’	26
Error Logs	27
Structure and Execution	29
Pipelines.....	30
Cxasap Pipeline.....	30
What it does	30
Input Files/Directories.....	30
Output Files/Directories.....	30
Yaml Parameters	32
Cxasap Pipeline (setup).....	33
What it does	33
Input Files/Directories.....	33

Output Files/Directories	33
Yaml Parameters	35
Refine Pipeline	36
What it does	36
Input Files/Directories.....	36
Output Files/Directories.....	36
Yaml Parameters	37
Cif Pipeline	38
What it does	38
Input Files/Directories.....	38
Output Files/Directories.....	38
Yaml Parameters	38
Analysis Pipeline.....	38
What it does	39
Input Files/Directories.....	39
Output Files/Directories.....	39
Yaml Parameters	40

Introduction

CX-ASAP is a library of *modules* which was designed to increase the efficiency of handling large crystallographic data sets. The modular design of this library means that *pipelines* can piece these modules together in different ways to automate the processing of large data sets.

What counts as a large data set?

In this case, a ‘large data set’ refers to a series of measurements performed on the same crystal. Examples of dynamic experiments include variable temperature, variable pressure and variable position. Such experiments allow for valuable insights into crystal systems including (but certainly not limited to!) mapping phase transitions. The fact that these data sets are performed on the same crystal means that reference structures can be used to replace the monotonous task of solving and refining the same thing over and over. Where phase transitions occur, this can be handled by having multiple reference structures – specifically, one reference per structural phase.

What does not count as a large data set?

CX-ASAP will NOT help you with stand-alone structures with large unit cells and large collections of unrelated structures. While tools exist to automatically solve and refine structures, these are not fool-proof, and CX-ASAP does not attempt this. Rather, it takes a reference structure, and uses it to finalise large series of the same model.

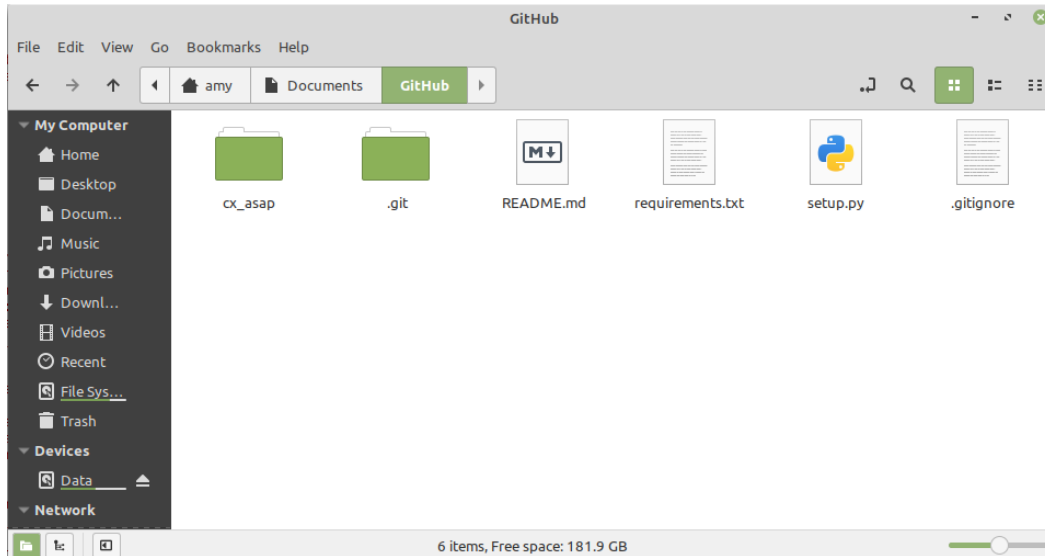
What can I expect out of this code?

Used correctly, CX-ASAP can provide finalised CIF Files, auto-generated checkCIF reports, and tabulated data of cif parameters (such as unit cell axis and bond lengths). Graphical analysis can quickly show how the unit cell axes are changing, if any of the specified structural features change, and provide feedback on the refinement quality. It is, however, important to keep in mind that you get out what you put in. If your reference model is not appropriate, then your finalised files will contain many errors. If you do not provide full instrument parameters for your collection, then these will be missing in the final CIF files.

By carefully following this manual, you can expect the highest (possible) quality for your data sets.

Quick Start Guide

Once you have extracted all the files, go into the parent folder and open README.md (shown below):



The README.md file will guide you through installation and troubleshooting. Then you can execute the code by typing the below into your terminal:

```
cxasap
```

You will be presented with a command-line interface that will guide you through usage of the code.

The rest of this document is aimed to assist in setting up your computer and providing a more in-depth guide if you require additional information.

In the command-line interface, you will be given a rough workflow, as well as several commands you can choose from. The general syntax is:

```
cxasap --option command --argument
```

The only 'option' available is the --help option.

To test that all software dependencies are satisfied, you should run the test command. You can do this by typing:

```
cxasap test
```

Help is always available as an argument as well. For instance, if you are unsure how to run the command 'module-refinement', you would type:

```
cxasap module-refinement --help
```

Note that 'options' are optional, while 'arguments' are required.

Within each command (except for 'errors' and 'test'), you should first use the arguments in the below order:

- dependencies (will tell you the required software for this code to run)
- files (will tell you the required file structure/input files)
- configure (will write your yaml file – *NOTE THAT YOU MUST FILL THIS IN MANUALLY*)
- run (will execute the code)

Requirements

Operating system requirements

Linux, Mac OS, or Windows.

For installation guides for the various operating systems, navigate within this document to the appropriate subheadings.

Software requirements

- Python (at least version 3)
- SHELXL (<http://shelx.uni-goettingen.de/download.php>)
- ShredCIF (<http://shelx.uni-goettingen.de/download.php>)
- Platon (<https://www.platonsoft.nl/platon/>)
- XDS (for future versions of CX-ASAP!)
- XPREP (for future versions of CX-ASAP!)

Note that all software needs to be on your path (also known as setting the environment variables for those used to Windows). If you are unfamiliar with setting your path, see the '*Instructions for putting software in your path*' section.

If I have a Windows computer, should I run CX-ASAP on Windows?

In this case, you have two options. You can either install a virtual machine running Linux or run CX-ASAP on windows. The first release of CX-ASAP, from structure refinement onwards, will function on Windows when set up correctly. Future releases of CX-ASAP, however, may not be compatible with Windows, as they utilise XDS, which is not currently compatible with Windows. Therefore, a virtual machine running Linux is recommended. The downside is that a virtual machine has limited computer resources and thus may run a bit slower; however, if your main computer has high enough specs, you should be able to dedicate enough resources for it to run at a reasonable pace.

Linux - Instructions for setting up a Linux Virtual Machine

Download

If you do not have linux running on a computer already, you will need to install two things: a program to handle virtual machines, and linux itself. A virtual machine is a way to run an additional computer within your computer. The below instructions are for Oracle VM VirtualBox, which is free software that can be downloaded using the below link:

<https://www.virtualbox.org/wiki/Downloads>

If your computer is running windows, you should download the latest version of 'Windows hosts' (currently at version 6.1.16).

Next, you will need to download Linux. The file you download is just like an installation disc, and you can delete it once linux is installed (otherwise you might run out of space on your computer!). This guide will use Linux Mint, although any version of linux should work. Linux Mint is available to download here: <https://www.linuxmint.com/edition.php?id=281> . You can use any of the mirroring services to download Linux Mint, although the two World Mirrors have found to be significantly faster than the Australian specific one.

Set up Linux VM within Oracle VM VirtualBox

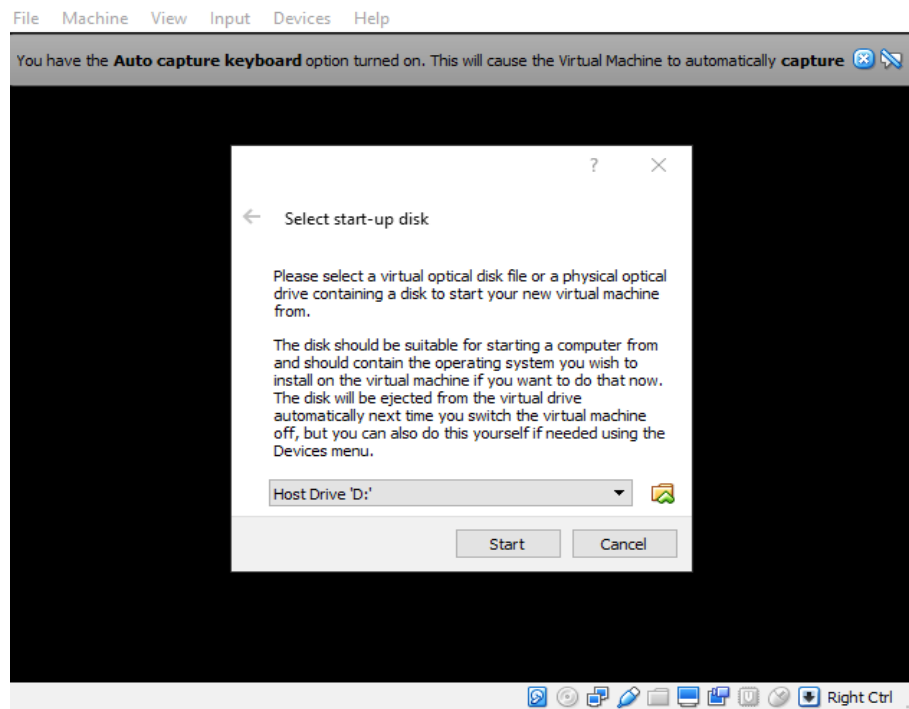
In addition to the steps outlined in the example provided above, some simple points are outlined below to assist you through the process. First, open Oracle VM and click 'new'. As you step through the setup, the below points will guide the options you pick.

- Name for your virtual machine - this can be anything (ie 'Linux for XDS')
- Type should be Linux
- Version should be Ubuntu 64-bit
- Memory - make sure you stay in the green bar (you will want to provide as much RAM as your hardware can, while also leaving enough for your main computer to run effectively - the green bar will guide this)
- Hard disk - leave it as 'create a virtual hard disk now'
- Hard disk file type - VDI - dynamically allocated is fine (if you know about computers and have different requirements feel free to change this)
- Location - the place your virtual machine will be saved on your computer, you will require a significant portion of space, especially because crystallography files can be large - recommended 100 GB if possible (should likely be 50 GB as a minimum)

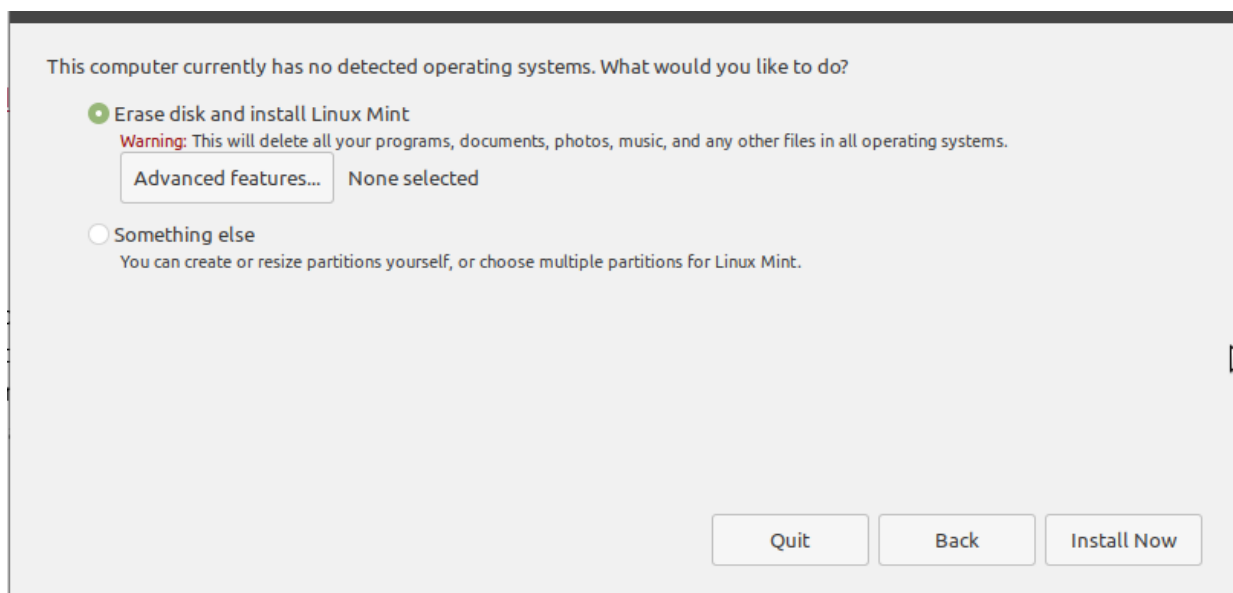
To run your virtual machine -

1. Select the VM from the home screen of Oracle VM VirtualBox (under whatever name you chose) and double click on it to start.

2. When the 'Select Start Up Disk' screen pops up (see below), change the option to the location you have downloaded Linux Mint to (from Step 1 - likely in your Downloads Folder) - this file should be a '.iso' file



The VM will then start up, and you will be prompted to install Linux Mint. Follow the prompts to install. When you see the below message, do not worry! Selecting this option **will not** erase your hard drive - the virtual machine does not know about the existence of your main computer.



Once you have installed Linux Mint within your VM, you will be prompted to enter a name and password for your virtual machine. At this point, your VM will begin downloading and will require a restart. This will only restart your VM, not your host computer.

Once restarted, the VM may prompt you to remove the disk - click enter - you don't have to delete anything right now; however, to save disk space, it is recommended that you do delete the .iso file you originally downloaded. Now that Linux Mint is installed, you no longer need the .iso file.

Restart the VM and prepare for more updates! To do this, open the terminal. The terminal is the fastest and most useful way to control Linux without need for a mouse. It is path specific, and you can open it in different locations in your directory tree; however, for this next step, it does not matter where you open it from. Use the below update/upgrade commands. It will ask for confirmation, at which point type 'Y'.

```
sudo apt-get update  
sudo apt-get upgrade
```

sudo = gives admin permissions (will require you to enter your password)
apt-get = command line tool for package management in linux
update = fetches list of available packages and current versions
upgrade = installs the newest versions of the packages

Note: If you get a 'check your video driver' notification about your system lacking video hardware acceleration, you will need to shut down your virtual machine and return to Oracle VM VirtualBox. Select your Linux VM and go to settings/display and check 'Enable 3D Acceleration'. You can then restart your linux VM.

Setting up a Share Folder

IMPORTANT NOTE - Do **NOT** turn off your virtual machine during **Step 3** except where explicitly directed! We have had cases where this has completely corrupted and crashed the VM.

Setting up a share folder will allow you to move files between your host computer and your virtual machine. First, there are some additional packages which you will need to download. Open the start menu of your linux machine and search for 'software manager'. Using the software manager, download the below three packages:

- virtualbox-guest-X11
- virtualbox-guest-utils
- virtualbox-guest-dkms

A link here is provided which will assist you in setting up a share folder for your Linux VM (<https://www.linuxbabe.com/linux-mint/install-virtualbox-guest-additions-in-linux-mint>). Follow these instructions, noting that you should have already updated all your packages. You also may not be able to see the shared folder during the instructions.

Linux- Instructions for installing Python

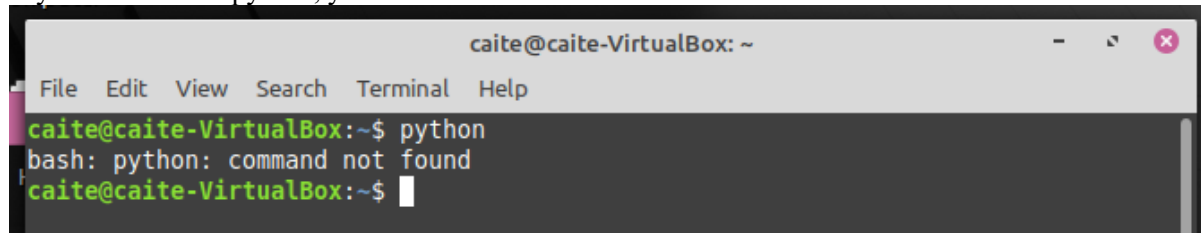
First, you will need to check and see if python is already installed on your Linux machine. You can do this by opening the terminal and typing the below command:

`python`

If your computer is already running python, something like the below image should come up.

```
Screams in Chemistry:~$ python
Python 3.6.9 (default, Apr 18 2020, 01:56:04)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

If you do not have python, you should see a different screen.

A terminal window titled 'caite@caite-VirtualBox: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command 'python' being entered, followed by the error message 'bash: python: command not found'.

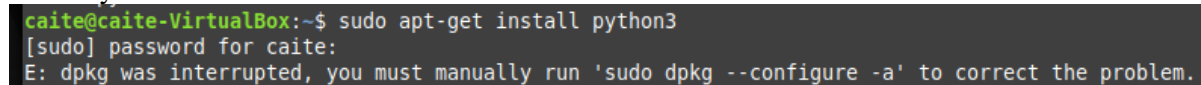
```
caite@caite-VirtualBox: ~
File Edit View Search Terminal Help
caite@caite-VirtualBox:~$ python
bash: python: command not found
caite@caite-VirtualBox:~$ █
```

Note that you will need to be running at least Python 3.6! If you have python already installed, check the version number – if you are running python 2, then you need to upgrade.

To install python, run the below code in your terminal:

```
sudo apt-get install python3
```

You may see an error like the one below:

A terminal window showing the command 'sudo apt-get install python3' being executed. It prompts for a password, and then displays an error message: 'E: dpkg was interrupted, you must manually run 'sudo dpkg --configure -a' to correct the problem.'

```
caite@caite-VirtualBox:~$ sudo apt-get install python3
[sudo] password for caite:
E: dpkg was interrupted, you must manually run 'sudo dpkg --configure -a' to correct the problem.
```

If this occurs, type into the terminal the below code:

```
sudo dpkg --configure -a
```

During configuration, accept any prompts the terminal provides. Once it is done, rerun the installation of python3.

To check if it has installed correctly, type '`python3`' into your terminal.

Next, to make sure that you are always running python3, you should set an alias in your bashrc file. Open your terminal and type the following command:

Linux - Instructions for putting software in your path

To put your programs into your path, it is recommended that you make a folder to dump all of your executables in. You may like to put it in `/usr/local/bin`. Note that to create folders and move files in this directory will need admin privileges. You obtain this by right clicking in the file explorer and selecting 'open as root'. Then, to set your folder as a path, type the following command into the terminal (located anywhere in the file system):

```
xed ~/.bashrc  
  
xed = a text editor  
  
~/.bashrc = path to your bashrc file
```

Note that `xed` may not be available (depending on what version of linux you are running). If this command doesn't work, we recommend you use 'kate'. You can install kate by typing:

```
sudo apt-get install kate
```

Then, you can access your bashrc file by typing:

```
kate ~/.bashrc
```

Anywhere within this file, type the following commands on a new line:

```
export PATH=full_path_name_to_folder:$PATH
```

full_path_name_to_folder = the path to your folder (example: `/usr/local/bin/my_program_folder`)

Anytime you make changes to your bash profile, you will need to restart your terminal for the changes to take effect.

Linux - Common Errors

Always check the file permissions! To do this, open the file explorer as root, right click on the files, click properties, and permissions. Make sure the user is set to read/write permissions (not just root!) and also tick the box that sets the file as executable (note that this can also be achieved using the 'chmod' command within the terminal if you are familiar with command line operations).

MacOS - Instructions for installing Python

While MacOS comes with python2, CX-ASAP requires python 3 to run. Firstly, you should download Xcode. This can be found in the apple store. Once you have installed Xcode, open a terminal and type the below code:

```
xcode-select --install
```

This will install the commandline tools associated with Xcode.

Next, you should install homebrew. Homebrew is a package manager which will be particularly useful for MacOS. To do this, open your terminal and copy the below code:

```
/bin/bash -c "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

Accept any prompts given in the terminal. You will also need to add Homebrew into your path (see ‘*Instructions for putting software in your path – MacOS*’). Note that the path you will need to add is “/usr/local/opt/python/libexec/bin”

Next, you can install python 3 by typing the below command:

```
brew install python
```

Make sure that the correct version of python is being called by typing the below code into your terminal:

```
python --version
```

MacOS - Instructions for putting software in your path

To put your programs into your path, it is recommended that you make a folder to dump all of your executables in. You may like to put it in `/usr/local/bin`. Then, to set your folder as a path, you will need to edit your bash profile.

First, check if you already have one. Open your terminal and navigate to the home folder by typing the below command:

```
cd ~/
```

To open your bash profile (if it exists), type:

```
open .bash_profile
```

If the file does not exist, create it by running the below command:

```
touch .bash_profile
```

To add a folder into your path, add the below command to the `bash_profile`:

```
export PATH="full_path_name_to_folder:$PATH"
```

If you have OS X 10.12 (Sierra) or older, you may need slightly different syntax:

```
export PATH=full_path_name_to_folder:$PATH
```

full_path_name_to_folder = the path to your folder (example: `/usr/local/bin/my_program_folder`)

Anytime you make changes to your bash profile, you will need to restart your terminal for the changes to take effect.

MacOS - Common Errors

Check commandline tools for Xcode are turned on. To do this, start Xcode and navigate to **Preferences**. In the General panel, click **Downloads**. Choose the **Components** tab. Click **Install** next to 'command line tools'.

Windows - Installation Instructions

Step 1: Firstly, you will need to download the C++ development tools for windows. This is done using the Visual Studio installer.

Visual Studio can be downloaded from the below site (select *Community 2022* from the dropdown menu):

<https://visualstudio.microsoft.com/>

Note: Visual studio installer 2019 also works, and you may need administrative rights to do this installation.

If Visual Studio Installer doesn't launch automatically, open it and install "*Desktop development with C++*".

Step 2: Download and install Python.

Python can be downloaded from the below site. Note that this code works with all versions of Python3 up to 3.9.11.

<https://www.python.org/downloads/windows/>

Python3.9.11 for 64-bit Windows can be directly downloaded from the below link.

<https://www.python.org/ftp/python/3.9.11/python-3.9.11-amd64.exe>

We recommend installing this with some custom settings, including putting it somewhere sensible like C:\program files\ , installing for all users, and adding it to your computers path (included in the installer).

You can test this by opening a command prompt and typing:

```
python
```

Note that to exit python you type the below:

```
exit()
```

Step 3: Restart your computer.

Step 4: Install CX-ASAP using the instructions in the README.md (or in the Installation section in this document).

Step 5: Check if you have 'make' installed on your computer. To do this, open windows powershell and type the below:

```
make
```


If the below text shows up, you have make installed correctly

“make: *** No targets specified and no makefile found. Stop.

If you have an error like below, you need to install make.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS H:\> make
make : The term 'make' is not recognized as the name of a cmdlet, function, script file, or operable program. Check
the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ make
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (make:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException
```

Make can be downloaded from the below website. Note that you should download the ‘Setup program’ version of make for ease of installation.

<http://gnuwin32.sourceforge.net/packages/make.htm>

You will then need to find the path to the make executable and add it to your system environment variables. It is recommended that you do this in the system wide path so that it will be accessible for all users. To check that this is done correctly, restart windows powershell and type ‘make’ again.

Windows - Common Errors

If you opt for the virtual environment installation of CX-ASAP (recommended), you may have some issues activating the environment. If you get an error saying that your virtual environment activation script “cannot be loaded because running scripts is disabled on this system”, follow the instructions on the below website to fix this issue.

<https://www.partitionwizard.com/clone-disk/running-scripts-is-disabled-on-this-system.html>

Additional notes for XDS Installation

Installing XDS

The documentation for XDS contains a lot of important information about the file types used, the parameters you can change, and the input/output files of each step. A link to the documentation is below.

http://xds.mpimf-heidelberg.mpg.de/html_doc/XDS.html

XDS itself can be installed from the below link. Instructions for installing XDS are also found from this link; however they are also expanded upon below.

http://xds.mpimf-heidelberg.mpg.de/html_doc/downloading.html

Once downloaded and uncompressed, you should put the files in your path. If you do not have a path, move your uncompressed XDS folder to /usr/local/bin (you can access the /usr folder from 'file system' in the Linux file explorer). You may need to use admin privileges to do this (by right clicking in the window and selecting 'open as root').

Open your bashrc (linux) or bash_profile (mac) file. Anywhere within this file, type the following two commands on new lines:

```
export PATH=full_path_name_to_XDS:$PATH  
export KMP_STACKSIZE=8m
```

- **Full_path_name_to_XDS** = the location of the folder where your XDS files are (example: /usr/local/bin/XDS-INTEL64_Linux_x86_64)

NOTE: if the very long name of 'XDS-INTEL64_Linux_x86_64' also annoys you and you have changed it to something nicer like 'XDS' make sure you change the path name (ie /usr/local/bin/XDS). The path name also does NOT need to include the file names - it is the path to the folder containing the files.

Make sure you close and reopen your terminal at this point.

If all is set up correctly, you should be able to type 'xds' into the terminal (located anywhere in the file system), and the program will run. Similarly, type 'xds_par' to run the parallel version.

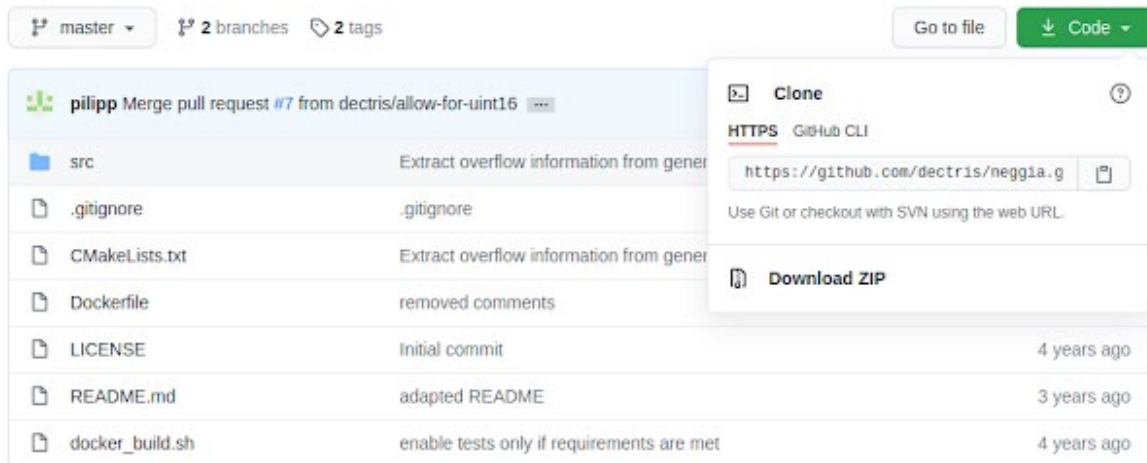
Preparing to Run XDS

There are some additional tools you will need before being able to process images from the Australian Synchrotron. The first of which is the 'dectris neggia library'. This is essentially a file that tells XDS how to read the .h5 files output from the current Eiger detectors.

Download the library from the below GitHub link:

<https://github.com/dectris/neggia>

First download the code by clicking ‘code’ and then ‘download zip’



Un-zip the folder and open your terminal. GitHub provides some instructions on how to install the required dependencies, but it is for different versions of linux, so you should follow the below instructions instead.

You will need to install the package ‘cmake’. To do this, type the below code into your terminal:

```
sudo apt-get install cmake
```

Again, you will need to type ‘Y’ to confirm the installation of packages.

Then, navigate to where your uncompressed folder is (it might be called something like ‘neggia-master’). Make sure your terminal is open in this folder - you can either navigate within your terminal, or simply open the terminal by right clicking on the screen inside the folder and selecting ‘open in terminal’. Make sure you are INSIDE the ‘neggia-master’ folder when you do this.

Next, type the below four lines into your terminal:

```
mkdir build
```

```
cd build
```

```
cmake ..
```

```
make
```

The only file you actually need from this folder is ‘dectris-neggia.so’, and it is found under ‘build/src/dectris/neggia/plugin’. You may wish to move this file somewhere more convenient, but this is optional.

XDS can also be run in parallel mode, meaning it can run over multiple processors to speed things up. You should look up how many processors you assigned to your virtual machine (found by selecting your VM in Oracle, clicking settings, and navigating to system/processor).

To execute xds in single processing mode, you will type into the terminal: 'xds'

To execute xds in parallel processing mode, you will type into the terminal: 'xds_par'

Installing XPREP on a Mac

Xprep does not natively run on a mac. For full pipelines involving data from the Australian Synchrotron, xprep is required. To have it functioning on your mac, you will need to have a copy of the linux version of xprep and follow the below instructions.

You will be using karton (<https://karton.github.io/>), and your computer will need to be on the web.

Step 1. Install Docker (<https://docs.docker.com/engine/install/>)

Step 2. Install Karton (<https://karton.github.io/install.html>). In a terminal window type: pip install karton.

Step 3. Use Karton to install a Ubuntu image called 'karton-ubuntu':

i) In terminal type: `karton image create karton-ubuntu ~/karton-ubuntu`
This will create the directory `karton-ubuntu` in your home directory and place file in it called `definition.py`

ii) Navigate to `~/karton-ubuntu` and open `definition.py`

a. Find `props.distro` and edit so it reads:

```
props.distro ='ubuntu:latest'
```

b. Edit the file to make directories you wish accessible to it at the very least make `~/src` accessible by including:

```
props.share_path_in_home('src')
```

We recommend using CX-ASAP in a directory called Frames so also add:

```
props.share_path_in_home('Frames')
```

You can add as many directories as you like in this way (or others following the instructions).

Also add:

```
props.packages.extend(['build-essential', 'gdb'])
```

save the file

iii) Now use Karton to build the image. In the terminal type:

```
karton build karton-ubuntu
```

iv) Copy the linux version of xprep somewhere that karton can access (we recommend `~/src`). Call it something like "linux-xprep"

v) Copy the following script into a text document:

```
#!/bin/sh
karton run ubuntu-image ~/src/xprep
```

and save it in `/usr/local/bin` as 'xprep'

If you cannot find `/usr/local/bin`, open Finder and press Command+Shift+G to open a dialogue box. Input '`/usr/local/bin`' to search for it.

Go back to your terminal and type:

```
cd /usr/local/bin
```

```
sudo chmod a+x xprep
```

Close the terminal

vi) Test by navigating to somewhere Karton has access to like ~/Frames and type xprep.

GitHub Configuration

If you only wish to use the code, then it is not a requirement to link up to GitHub. You can simply download the code and install. This has the downside of meaning you will have to go to GitHub to check for updates and redownload any new releases.

If you wish to be able to quickly update code, or if you wish to contribute to CX-ASAP, it is required that you use Git. Git is a version-control system for software development which is integrated into 'GitHub' where the source code for CX-ASAP is stored. To do this, first open your terminal.

For Linux Users:

To install Git, run the below command:

```
sudo apt-get install git-all
```

For MacOS Users:

Homebrew is the easiest way to install git on a Mac. To do this, run the below command:

```
brew install git
```

Next, regardless of your operating system, navigate in your terminal to the location in your file system where you want to save the code. Then, run the below command:

```
git clone https://github.com/cx-asap/CX-ASAP.git
```

Choose which branch you want to work from (main, or dev)

```
git switch dev OR git switch main
```

You should frequently check GitHub for updates. To do this, in the same folder, open a terminal and run the below command:

```
git pull origin main OR git pull origin dev
```

For Windows Users:

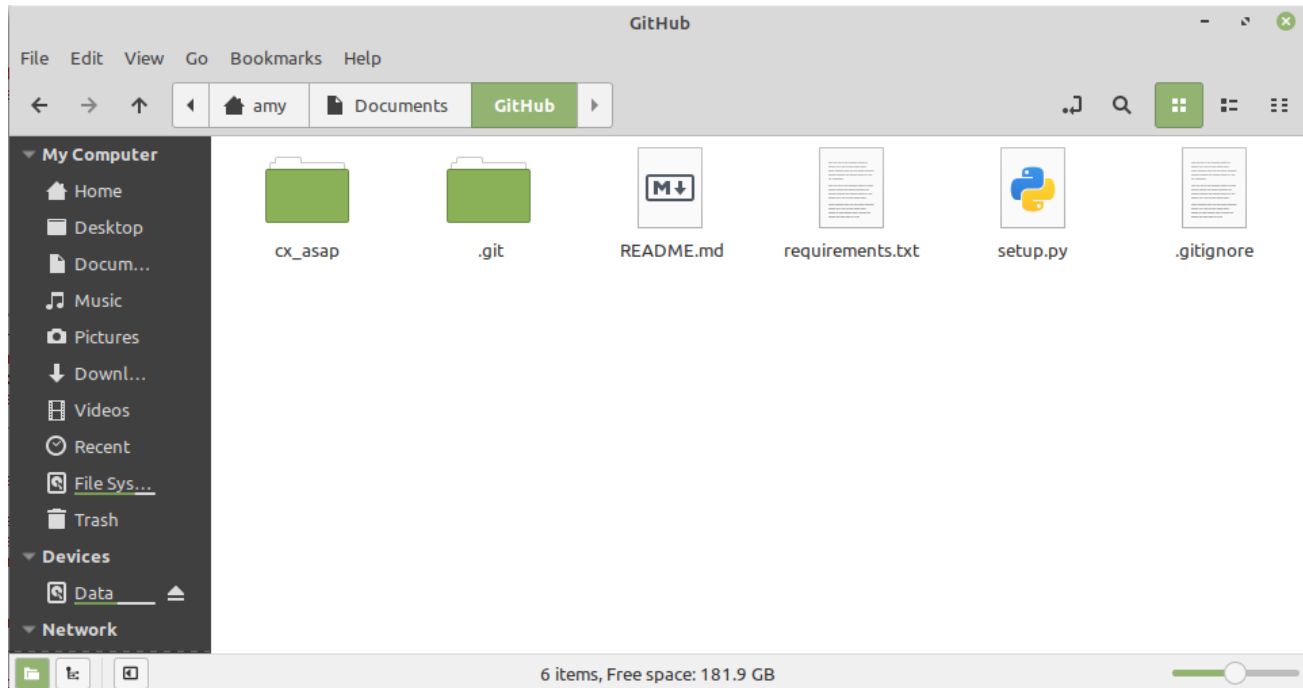
Go to the below website and download the Git for Windows tool.

<https://gitforwindows.org/>

Installation

CX-ASAP Installation

To install CX-ASAP, navigate to the location in your file system where you saved the code. You should be in this folder:



In this folder, you can check the ‘README.md’ file for initial instructions, including some installation troubleshooting. This file also tells you how to start the code.

Open a terminal in this location. In Linux, you can do this by right clicking on the window and selecting ‘Open in Terminal’, or you can navigate to it manually. Once here, you have several installation options depending on how you want your computer and environment set up.

If you wish to install CX-ASAP straight onto your computer (not recommended if you are familiar with computers and virtual environments), type the below command into your terminal:

```
make install-quick
```

If you are familiar with working with virtual environments, it is recommended that you use the below command instead. This will install CX-ASAP into a new virtual environment called *cxasap_venv* (located in the same path as the make file). It will require manual activation each time you wish to use the software.

```
make install-venv
```

If you want command-line completion (where you hit the TAB button and it autocompletes), run the additional command below (not required – also not available for windows).

```
make cxasap-complete
```


If you also want an alias set up for easy access to the yaml file, you can also run the below additional command (not required for use – also not available for windows). This means that if you are on Mac or Linux OS, you will be able to type ‘cxasap_yaml’ into the commandline and the yaml will open automatically for you.

```
make yaml-alias
```

Note that any changes you have made to your conf.yaml file will be erased upon re-installation. This is to make sure you are using the correct (and a simplified) conf.yaml file. Your error logs will also be reset upon re-installation. The purpose of these two files will be explained in the following section.

Testing Your Installation

The first time you run ‘cxasap’ you should chose the ‘test’ command. This will run through some analysis on the included test data set to make sure your programs and files are correctly set up. You can do this by typing:

```
cxasap test
```

Reporting Errors

If you encounter a problem with the code itself, please contact us and describe in detail the bug you are experiencng. This should be done through the GitHub page.

To further assist us in helping you, please provide any screenshots you feel may be relevant to describing your error, and attach a copy of your error logs. Unfortunately, this is not always enough to fix strange problems. If you feel comfortable doing so, and are not at risk of breaking intellectual property, sharing your raw data with us will likely assist us in solving the errors. However, ***never share data that would be a breach of IP, and you are never under any obligation to provide raw data.*** If you do feel uncomfortable sharing your data and we cannot fix the bug with screenshots alone, perhaps try and recreate the error on an already published data set that you are able to share with us.

If you have problems installing any of the requirements (ie python, platon, SHELXL, etc), please refer first to the many online help guides available. If you are still unable to solve these problems yourself, feel free to contact us and we will endeavour to assist you.

Using CX-ASAP

Command-line Interface

CX-ASAP uses a command-line interface. This will guide you through using the software and provide all the hints you need. First, you will need to type into your terminal:

```
cxasap
```

From here, you will be given a rough workflow, as well as several commands you can choose from. The general syntax is:

```
cxasap --option command --argument
```

The only ‘option’ available is the `--help` option. Help is always available as an argument as well. For instance, if you are unsure how to run the command ‘module-refinement’, you would type:

```
cxasap module-refinement --help
```

Note that ‘options’ are optional, while ‘arguments’ are required.

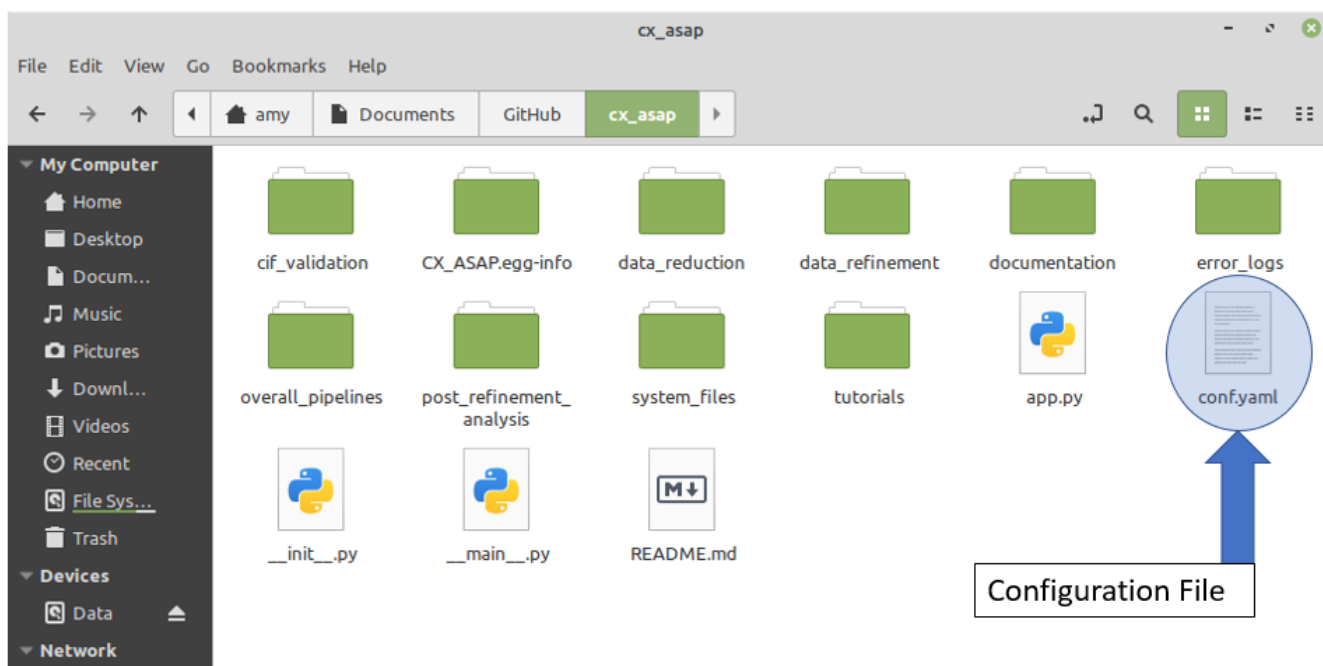
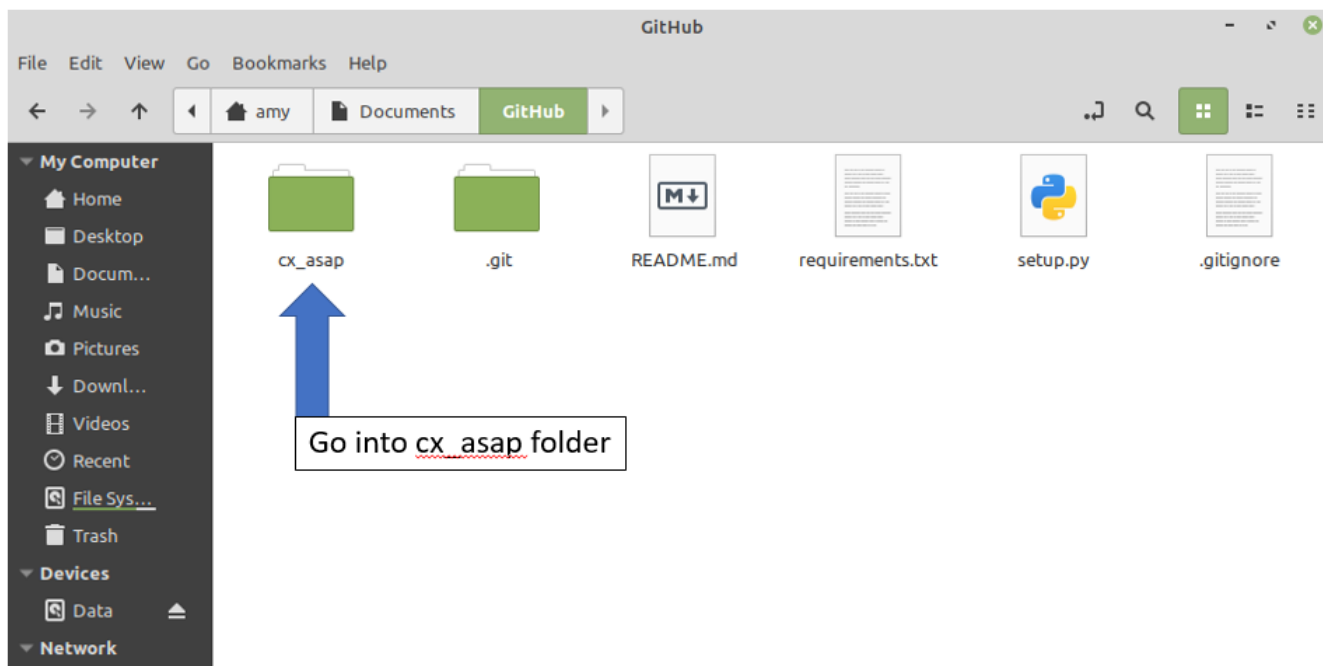
Within each command (except for ‘errors’ and ‘test’), you should first use the arguments in the below order:

- `--dependencies` (will tell you the required software for this code to run)
- `--files` (will tell you the required file structure/input files)
- `--configure` (will write your yaml file – *NOTE THAT YOU MUST FILL THIS IN MANUALLY*)
- `--run` (will execute the code)

Configuration Using 'conf.yaml'

After you have configured your experiment, you will find a file called 'conf.yaml' in the 'cx_asap' folder.

If you are on Linux or Mac OS and you used the "make yml-alias" option during installation, you can simply type "cxasap_yml" and the conf.yaml file will open automatically. If you are using windows, or you did not choose this option, you will need to navigate here manually and open the file with a text editor of your choice.



'conf.yaml' is a 'yaml' file, which is a recursive acronym for 'YAML Ain't Markup Language'. These files store values assigned to specific parameters. This is also how you will configure all of the code that you run. Once you

have selected a pipeline, you will need to fill out the required parameters. Yaml files can be edited with a text editor of your choosing.

Your yaml file will be rewritten every time you configure your experiment, so if you wish to save a copy elsewhere for later use, make sure you do that.

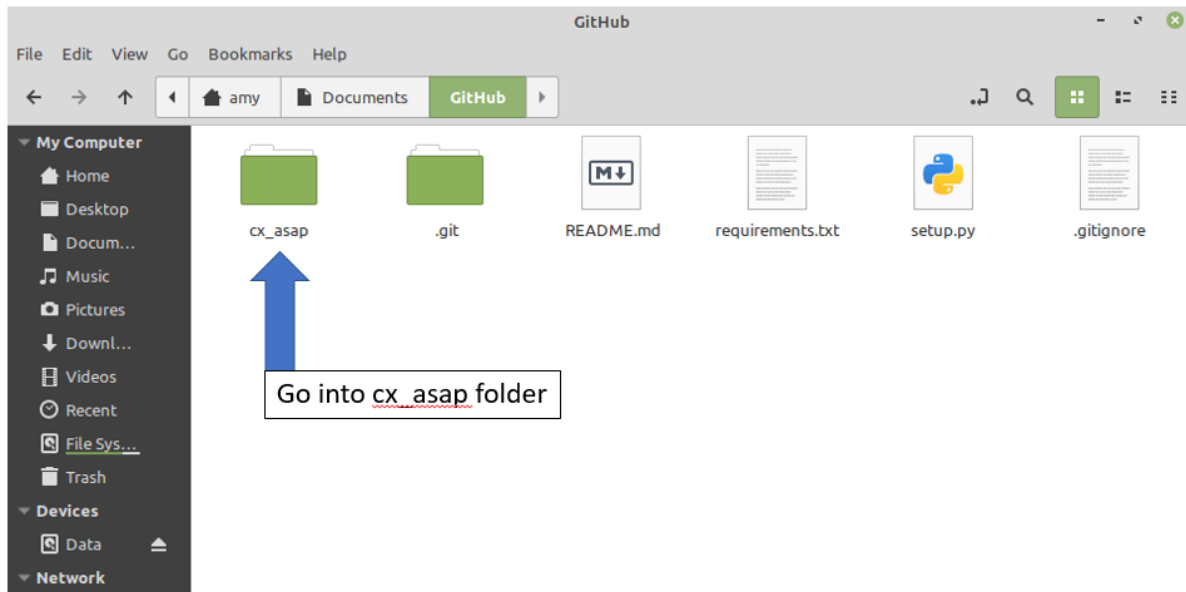
Note that re-installing CX-ASAP will remove any entries you have saved in your yaml file. This is to make sure that the most up-to-date configuration file will be used with the code.

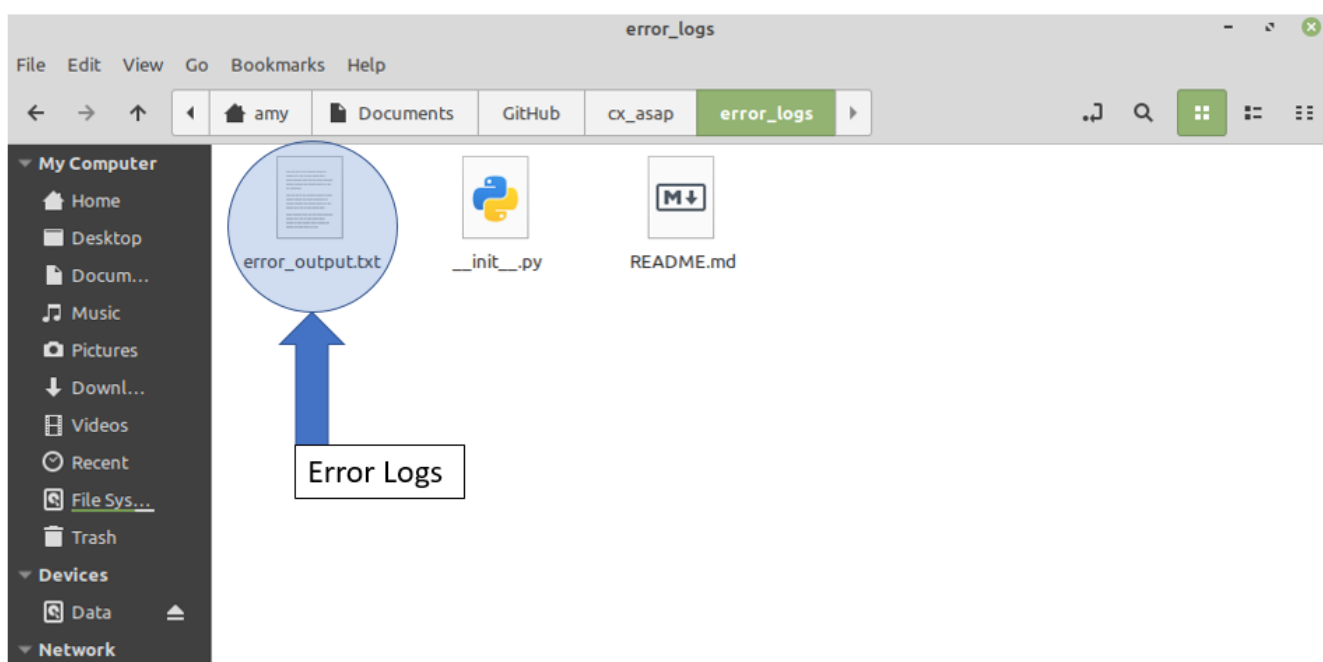
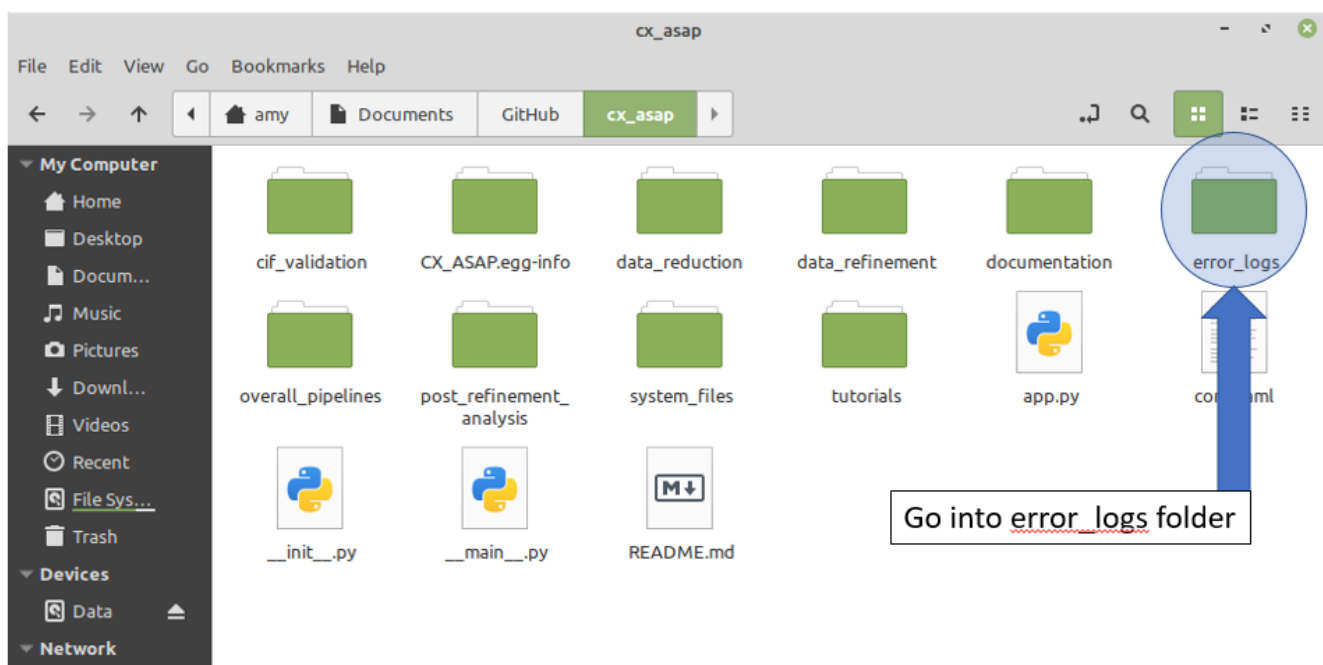
Tips for troubleshooting:

- File paths must be the full path!
- File paths should include the file *name (including the ending)* if it belongs to a file (not required if the path is to a folder)

Error Logs

CX-ASAP regularly logs information which may be useful in debugging errors. It may also prove useful for a user to read through if they get an unexpected output even if the code has successfully gone to completion. These logs can be found in the `cx_asap/error_logs` folder. This folder will contain the most recent 5 error logs, so may be useful if you need to go back. The most recent one relevant to your experiment will also be copied over to your data location where you ran the code. The error logs are contained in `'error_output.txt'`. Like the `conf.yaml` files, it can be opened using a simple text editor. Otherwise, this file can be accessed through the command-line interface.



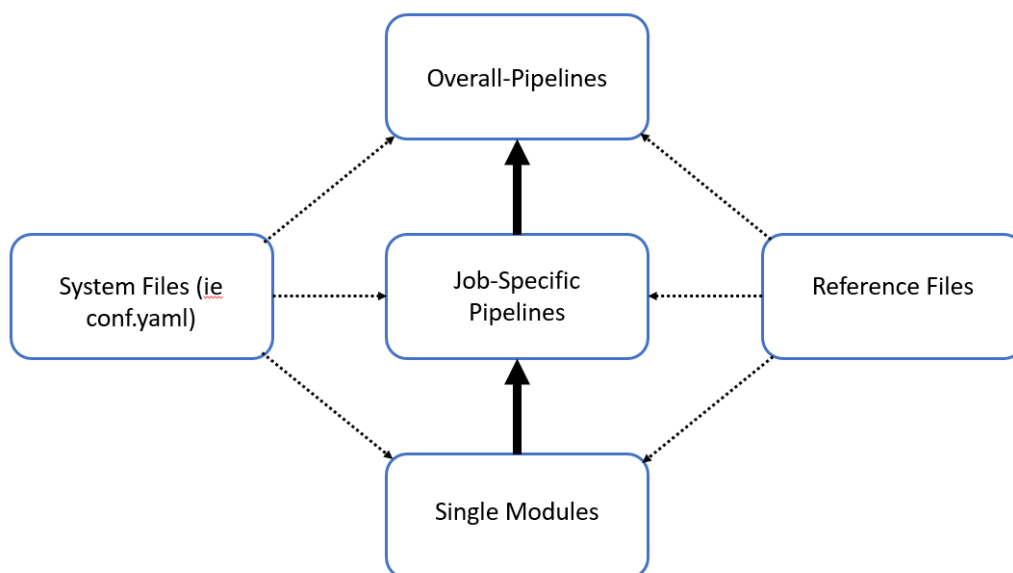


Note that re-installing CX-ASAP will clear and reset the error log. If you require a copy, make sure you save it in a different location prior to updating CX-ASAP.

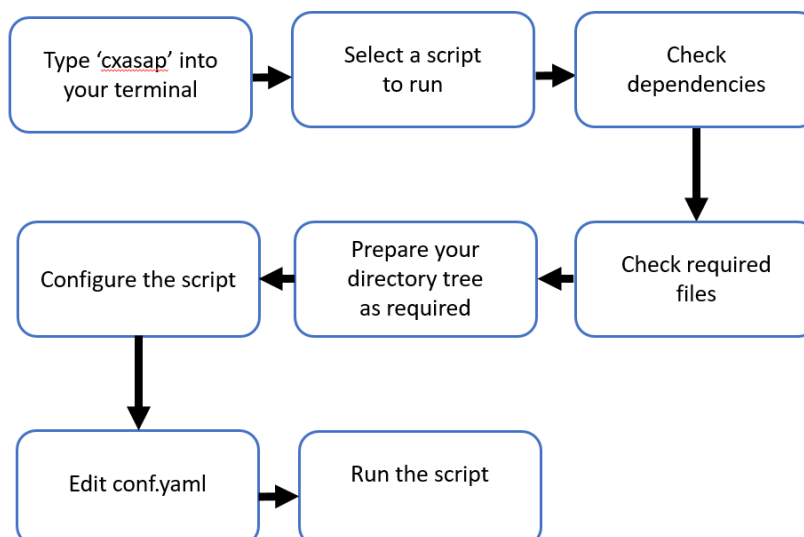
Structure and Execution

CX-ASAP is built from a library of modules which perform specific tasks, like refining a dataset. Job-specific pipelines were designed to execute these modules on a series of datasets. For example, a refinement pipeline would refine a series of structures. Overall pipelines bring the job-specific pipelines together to increase automation. For instance, it can refine your structures, compile the CIF files and analyse them. This means that certain steps can be repeated or done alone depending on the needs of your data. For instance, some tricky data may require manual refinement, but by using job-specific pipelines, you can still automate the compilation of CIF files and the extraction of key data.

CX-ASAP Code Hierarchy



The workflow for using CX-ASAP is shown in the below diagram. This document requires all required information. In the 'Pipelines' section there are detailed descriptions for what each pipeline does, what its input/output files are, how your directory tree should be set up, and what parameters require editing in the conf.yaml file.



Pipelines

Cxasap Pipeline

What it does

This is the general overall pipeline that will automatically refine, finalise and analyse a series of structures based off a common reference. You should use this pipeline if you have a set of reduced data files from a dynamic crystallography experiment, such as a variable temperature experiment. Note that this pipeline works from a reference CIF – if you do not have one of these, or you don't have your files in the correct directory structure (see *Input Files/Directories*), consider using 'Cxasap Pipeline (setup)'

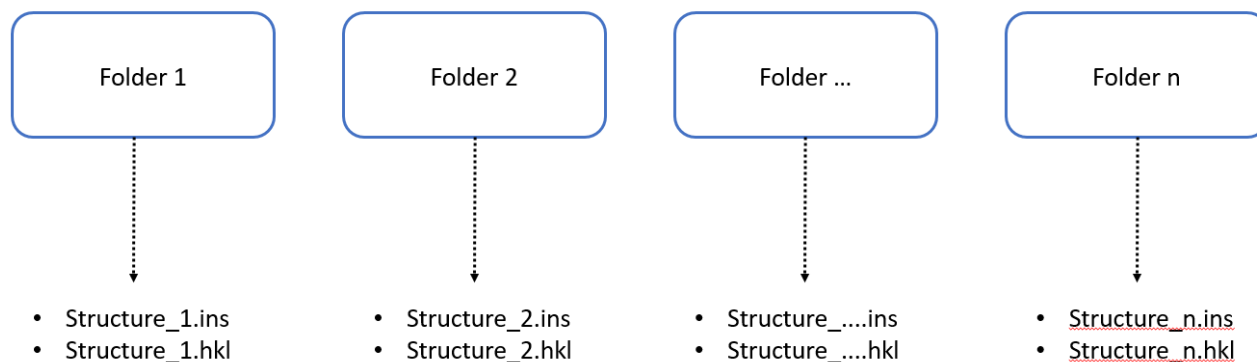
Input Files/Directories

You must have the following files to use this pipeline:

- .ins for each structure – must contain cell data
- .hkl for each data set – must contain all reflections
- Reference CIF – must contain instrument information and crystal information (ie size/colour/habit)

It does not matter where your reference CIF is located on your computer, but your .ins/.hkl files must be in separate folders. It also does not matter what your folders are called or what your .ins/.hkl files are called. The only restriction of course is that SHELXL requires your .ins and .hkl files to have the same name as each other. The file structure of your 'working directory' for n data sets should be as shown below:

Working Directory Structure



Output Files/Directories

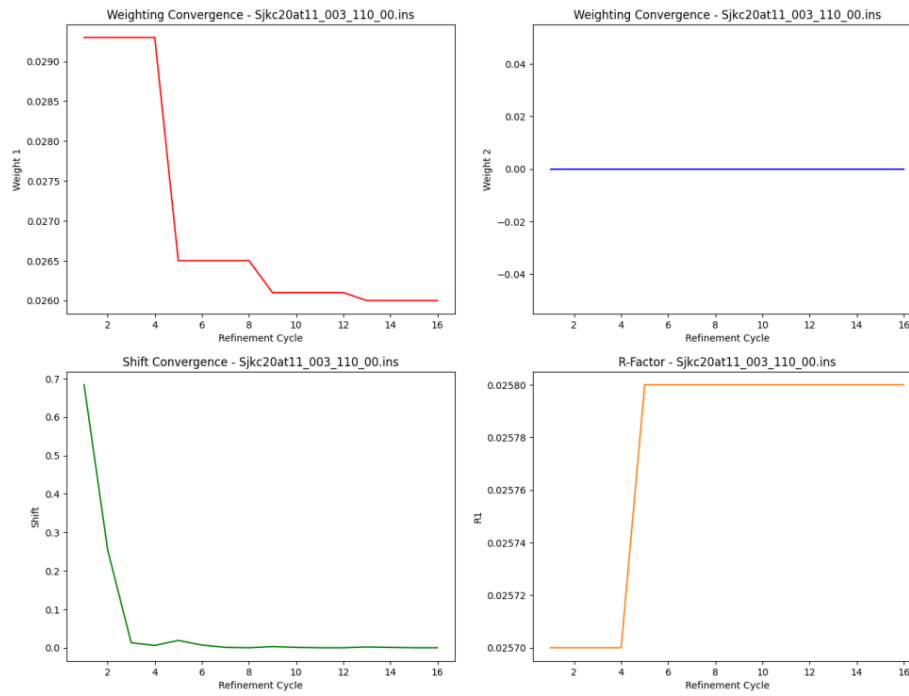
CX-ASAP will create two additional folders in your working directory: 'CIF_Analysis' and 'Refinement_Statistics'.

CIF_Analysis contains your finalised structures and various forms of analysis. Output files/directories in this folder are listed below:

- Combined.cif – this is your finalised CIF file, which combines each of the individual structures in your working directory
- Check_CIF.chk – this is your automated checkCIF report for all structures in the combined CIF file

- CIF_Parameters.csv – this file contains your tabulated CIF parameters including (but not limited to!) unit cell axis and data quality statistics – *your conf.yaml file will allow you to specify what data is extracted here*
- Cell_parameters.png – this graph shows how all of your cell parameters are changing
- Axis_deformation.png – this graph presents the a-, b-, c-axis and the cell volume in terms of the percent deformation from your reference structure
- Angle_deformation.png – this graph presents alpha, beta, gamma and the cell volume in terms of the percent deformation from your reference structure
- Quality_Statistics.png – this graph presents the quality statistics and how they change throughout the experiment
- Bond_Lengths.csv – this file contains all of the extracted bond length information from the finalised CIF files – *this file will only be present if specified in the conf.yaml file*
- Bond_Angles.csv – this file contains all of the extracted bond angle information from the finalised CIF files – *this file will only be present if specified in the conf.yaml file*
- Bond_Torsions.csv – this file contains all of the extracted torsion angle information from the finalised CIF files – *this file will only be present if specified in the conf.yaml file*
- Important_Bond_Lengths.csv – this file contains all of the extracted bond lengths only for the atoms of interest (ie a metal in a complex) – *this file will only be present if specified in the conf.yaml file, which is also where you specify the atom(s) of interest*
- Important_Bond_Angles.csv – this file contains all of the extracted bond angles only for the atoms of interest (ie a metal in a complex) – *this file will only be present if specified in the conf.yaml file, which is also where you specify the atom(s) of interest*
- Important_Bond_Torsions.csv – this file contains all of the extracted torsion angles only for the atoms of interest (ie a metal in a complex) – *this file will only be present if specified in the conf.yaml file, which is also where you specify the atom(s) of interest*
- Bonds.png – the important bond lengths have been graphed to look for distinct changes – *this file will only be present if specified in the conf.yaml file*
- Angles.png – the important bond angles have been graphed to look for distinct changes – *this file will only be present if specified in the conf.yaml file*
- Torsions.png – the important torsion angles have been graphed to look for distinct changes – *this file will only be present if specified in the conf.yaml file*
- Individual_Bond_Length_Data – each .csv file in this folder will contain the bond lengths for one set of atoms and how they change over the experiment – *this folder will only be present if specified in the conf.yaml file*
- Individual_Angle_Data – each .csv file in this folder will contain the bond angles for one set of atoms and how they change over the experiment – *this folder will only be present if specified in the conf.yaml file*
- Individual_Torsion_Data – each .csv file in this folder will contain the torsion angles for one set of atoms and how they change over the experiment – *this folder will only be present if specified in the conf.yaml file*

Refinement_Statistics has a graph for each dataset in your working directory, and shows how the statistics change during the refinement. Use these graphs to check that your refinements have satisfactorily converged. An example graph is given below:



Yaml Parameters

See information on the command line interface for descriptions of all yaml parameters.

Cxasap Pipeline (setup)

What it does

This is a variation of the general overall pipeline that will automatically refine, finalise and analyse a series of structures based off a common reference. You should use this extended version if you have a set of reduced data files from a dynamic crystallography experiment that are not necessarily in the correct file structure for 'cxasap pipeline'. You should also consider this option if you do not have a reference CIF file including required instrument parameters for publication. Instead, you will have the option of manually entering them into the conf.yaml file. Note that you can add additional CIF parameters to the conf.yaml file as required. You will also need a reference structure in the form of a .ins or .res file.

Input Files/Directories

You must have the following files to use this pipeline:

- .ins for each structure – must contain cell data
- .hkl for each data set – must contain all reflections
- Reference .ins or .res – must contain structure

It does not matter where your files are located on your computer, but note that when the code makes your directory structure, you will need to place your .ins/.hkl files in each folder before proceeding with the code. It also does not matter what your files are called. The only restriction of course is that SHELXL requires your .ins and .hkl files to have the same name as each other.

Output Files/Directories

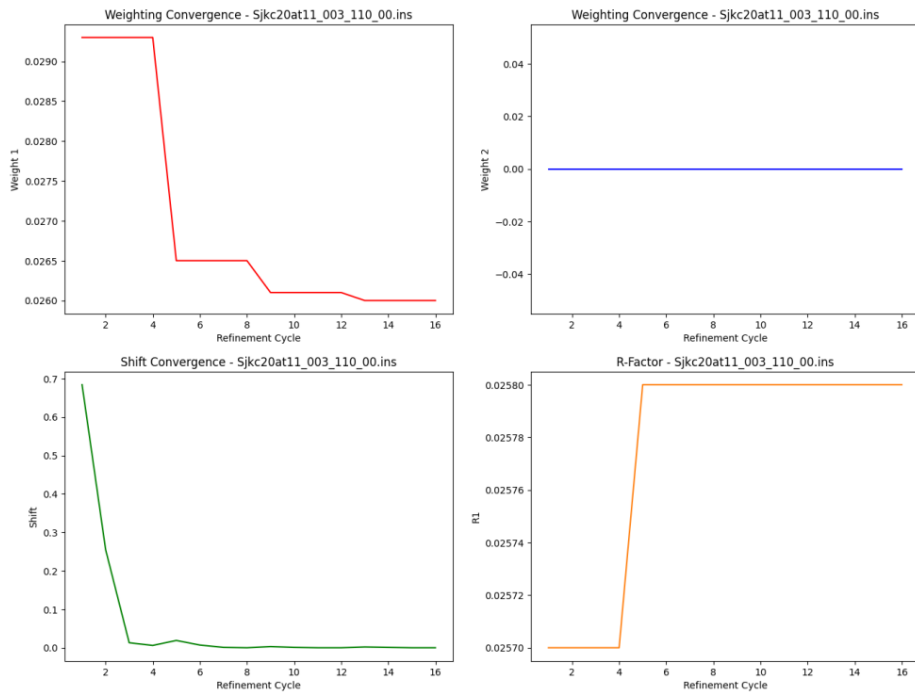
CX-ASAP will create two additional folders in your working directory: 'CIF_Analysis' and 'Refinement_Statistics'.

CIF_Analysis contains your finalised structures and various forms of analysis. Output files/directories in this folder are listed below:

- Combined.cif – this is your finalised CIF file, which combines each of the individual structures in your working directory
- Check_CIF.chk – this is your automated checkCIF report for all structures in the combined CIF file
- CIF_Parameters.csv – this file contains your tabulated CIF parameters including (but not limited to!) unit cell axis and data quality statistics – *your conf.yaml file will allow you to specify what data is extracted here*
- Cell_parameters.png – this graph shows how all of your cell parameters are changing
- Axis_deformation.png – this graph presents the a-, b-, c-axis and the cell volume in terms of the percent deformation from your reference structure
- Angle_deformation.png – this graph presents alpha, beta, gamma and the cell volume in terms of the percent deformation from your reference structure
- Quality_Statistics.png – this graph presents the quality statistics and how they change throughout the experiment
- Bond_Lengths.csv – this file contains all of the extracted bond length information from the finalised CIF files – *this file will only be present if specified in the conf.yaml file*
- Bond_Angles.csv – this file contains all of the extracted bond angle information from the finalised CIF files – *this file will only be present if specified in the conf.yaml file*

- Bond_Torsions.csv – this file contains all of the extracted torsion angle information from the finalised CIF files – *this file will only be present if specified in the conf.yaml file*
- Important_Bond_Lengths.csv – this file contains all of the extracted bond lengths only for the atoms of interest (ie a metal in a complex) – *this file will only be present if specified in the conf.yaml file, which is also where you specify the atom(s) of interest*
- Important_Bond_Angles.csv – this file contains all of the extracted bond angles only for the atoms of interest (ie a metal in a complex) – *this file will only be present if specified in the conf.yaml file, which is also where you specify the atom(s) of interest*
- Important_Bond_Torsions.csv – this file contains all of the extracted torsion angles only for the atoms of interest (ie a metal in a complex) – *this file will only be present if specified in the conf.yaml file, which is also where you specify the atom(s) of interest*
- Bonds.png – the important bond lengths have been graphed to look for distinct changes – *this file will only be present if specified in the conf.yaml file*
- Angles.png – the important bond angles have been graphed to look for distinct changes – *this file will only be present if specified in the conf.yaml file*
- Torsions.png – the important torsion angles have been graphed to look for distinct changes – *this file will only be present if specified in the conf.yaml file*
- Individual_Bond_Length_Data – each .csv file in this folder will contain the bond angles for one set of atoms and how they change over the experiment – *this folder will only be present if specified in the conf.yaml file*
- Individual_Angle_Data – each .csv file in this folder will contain the bond angles for one set of atoms and how they change over the experiment – *this folder will only be present if specified in the conf.yaml file*
- Individual_Torsion_Data – each .csv file in this folder will contain the torsion angles for one set of atoms and how they change over the experiment – *this folder will only be present if specified in the conf.yaml file*

Refinement_Statistics has a graph for each dataset in your working directory, and shows how the statistics change during the refinement. Use these graphs to check that your refinements have satisfactorily converged. An example graph is given below:



Yaml Parameters

See information on the command line interface for descriptions of all yaml parameters.

Refine Pipeline

What it does

This job-specific pipeline will refine a series of .ins/.hkl structures based on a common reference .ins or .res file. CIF files will not be compiled or edited, and analysis will not be undertaken. This pipeline may be useful in testing refinement models.

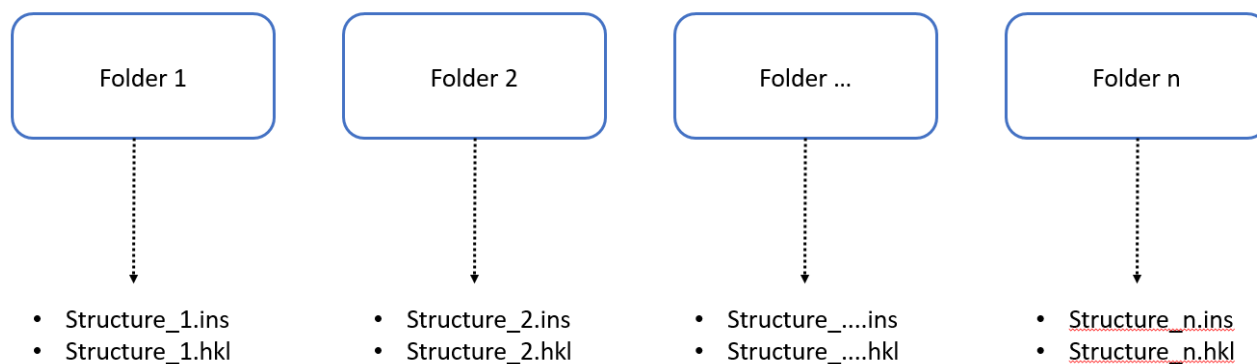
Input Files/Directories

You must have the following files to use this pipeline:

- .ins for each structure – must contain cell data
- .hkl for each data set – must contain all reflections
- Reference .ins or .res – must contain structure

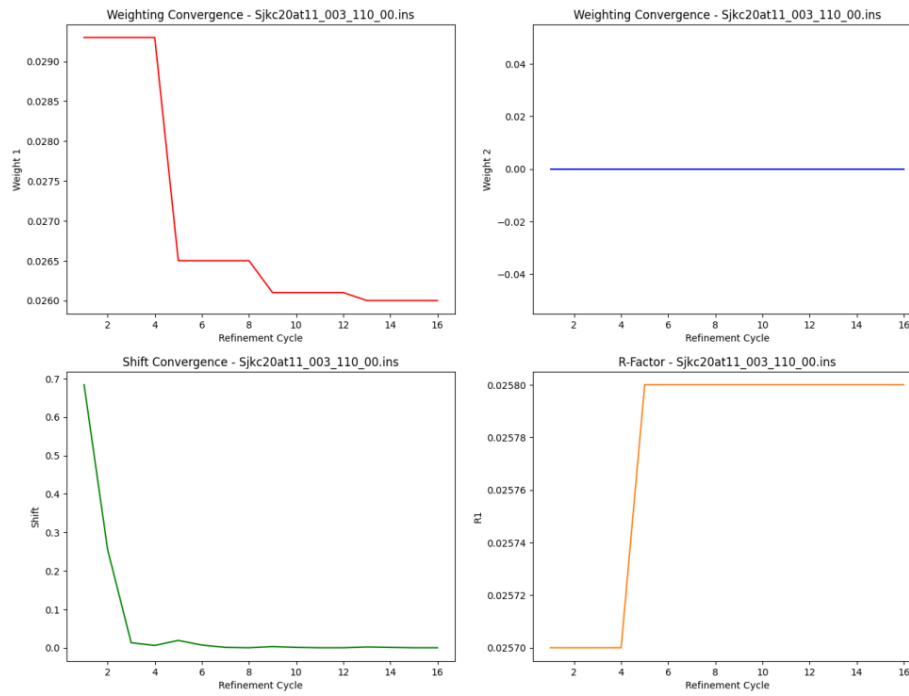
It does not matter where your reference file is located on your computer, but your .ins/.hkl files must be in separate folders. It also does not matter what your folders are called or what your .ins/.hkl files are called. The only restriction of course is that SHELXL requires your .ins and .hkl files to have the same name as each other. The file structure of your ‘working directory’ for n data sets should be as shown below:

Working Directory Structure



Output Files/Directories

CX-ASAP will create an additional folder in your working directory called ‘Refinement_Statistics’. Refinement_Statistics has a graph for each dataset in your working directory and shows how the statistics change during the refinement. Use these graphs to check that your refinements have satisfactorily converged. An example graph is given below:



Yaml Parameters

See information on the command line interface for descriptions of all yaml parameters.

Cif Pipeline

What it does

This job-specific pipeline will merge a series of .cif files with any instrument parameter CIF files and compile them into one combined CIF. Consider using this pipeline if you needed to do a large series of refinements by hand and wish to quickly combine and finalise your CIF files.

Input Files/Directories

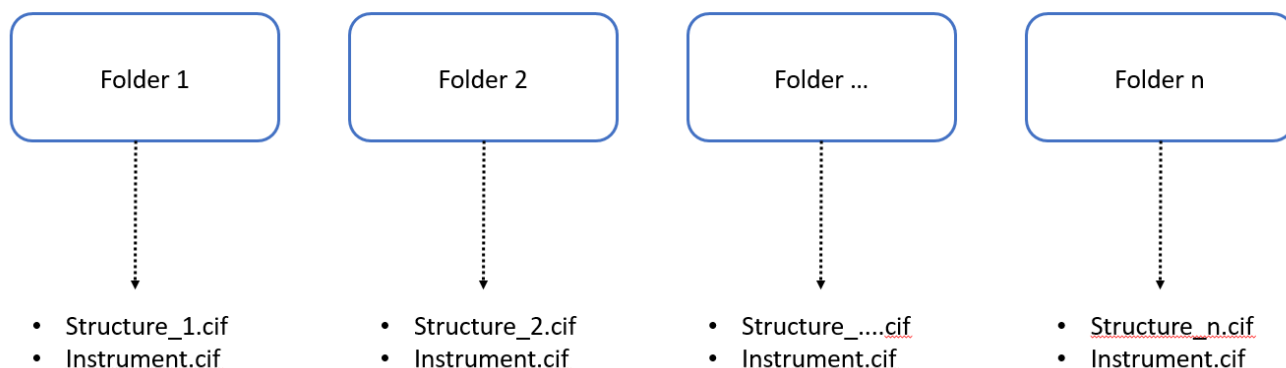
You must have the following files to use this pipeline:

- Structure CIF for each data set
- Instrument CIF for each data set

Your instrument CIFs will need to be distinguished from the structure CIFs. CX-ASAP gives you two options to do this:

- 1) Your instrument CIFs can all have different file names but a unique ending that is NOT '.cif'. An example of an acceptable instrument CIF is based off the Rigaku naming system – '*structure_1.cif_od*'.
- 2) All of your instrument CIFs over the different folders must have an IDENTICAL name. An example of an acceptable instrument CIF is based off the Australian Synchrotron's autoprocess naming system – '*autoprocess.cif*'.

Working Directory Structure



Output Files/Directories

Output files will appear in your working directory. Two files will be output by the program:

- Combined.cif – all of the CIF files merged into one file
- Check_CIF.chk – the automated checkCIF report corresponding to the combined CIF file

Yaml Parameters

See information on the command line interface for descriptions of all yaml parameters.

Analysis Pipeline

What it does

This job-specific pipeline will analyse a series of CIF files and extract desired parameters into .csv files (which can be opened in Excel). Some graphical analysis will be given as well. Examples of parameters you can tabulate include:

- Cell axis
- Refinement statistics
- Temperature
- Bond lengths
- Angles
- Torsions

You should consider using this pipeline if you needed to refine and prepare all your CIFs by hand, but would like quick extraction of parameters for data analysis.

Input Files/Directories

You must have the following files to use this pipeline:

- CIF files
- Reference .ins that has a unit cell to compare to for cell deformations

Your CIF files should be contained within a single folder, the name of which does not matter.

Output Files/Directories

CX-ASAP will output several analysis files based on your CIF file(s). These output files/directories in this folder are listed below:

- Combined.cif – this is your finalised CIF file, which combines each of the individual structures in your working directory
- Check_CIF.chk – this is your automated checkCIF report for all structures in the combined CIF file
- CIF_Parameters.csv – this file contains your tabulated CIF parameters including (but not limited to!) unit cell axis and data quality statistics – *your conf.yaml file will allow you to specify what data is extracted here*
- Cell_parameters.png – this graph shows how all of your cell parameters are changing
- Axis_deformation.png – this graph presents the a-, b-, c-axis and the cell volume in terms of the percent deformation from your reference structure
- Angle_deformation.png – this graph presents alpha, beta, gamma and the cell volume in terms of the percent deformation from your reference structure
- Quality_Statistics.png – this graph presents the quality statistics and how they change throughout the experiment
- Bond_Lengths.csv – this file contains all of the extracted bond length information from the finalised CIF files – *this file will only be present if specified in the conf.yaml file*
- Bond_Angles.csv – this file contains all of the extracted bond angle information from the finalised CIF files – *this file will only be present if specified in the conf.yaml file*
- Bond_Torsions.csv – this file contains all of the extracted torsion angle information from the finalised CIF files – *this file will only be present if specified in the conf.yaml file*
- Important_Bond_Lengths.csv – this file contains all of the extracted bond lengths only for the atoms of interest (ie a metal in a complex) – *this file will only be present if specified in the conf.yaml file, which is also where you specify the atom(s) of interest*

- Important_Bond_Angles.csv – this file contains all of the extracted bond angles only for the atoms of interest (ie a metal in a complex) – *this file will only be present if specified in the conf.yaml file, which is also where you specify the atom(s) of interest*
- Important_Bond_Torsions.csv – this file contains all of the extracted torsion angles only for the atoms of interest (ie a metal in a complex) – *this file will only be present if specified in the conf.yaml file, which is also where you specify the atom(s) of interest*
- Bonds.png – the important bond lengths have been graphed to look for distinct changes – *this file will only be present if specified in the conf.yaml file*
- Angles.png – the important bond angles have been graphed to look for distinct changes – *this file will only be present if specified in the conf.yaml file*
- Torsions.png – the important torsion angles have been graphed to look for distinct changes – *this file will only be present if specified in the conf.yaml file*
- Individual_Bond_Length_Data – each .csv file in this folder will contain the bond lengths for one set of atoms and how they change over the experiment – *this folder will only be present if specified in the conf.yaml file*
- Individual_Angle_Data – each .csv file in this folder will contain the bond angles for one set of atoms and how they change over the experiment – *this folder will only be present if specified in the conf.yaml file*
- Individual_Torsion_Data – each .csv file in this folder will contain the torsion angles for one set of atoms and how they change over the experiment – *this folder will only be present if specified in the conf.yaml file*

[Yaml Parameters](#)

See information on the command line interface for descriptions of all yaml parameters.