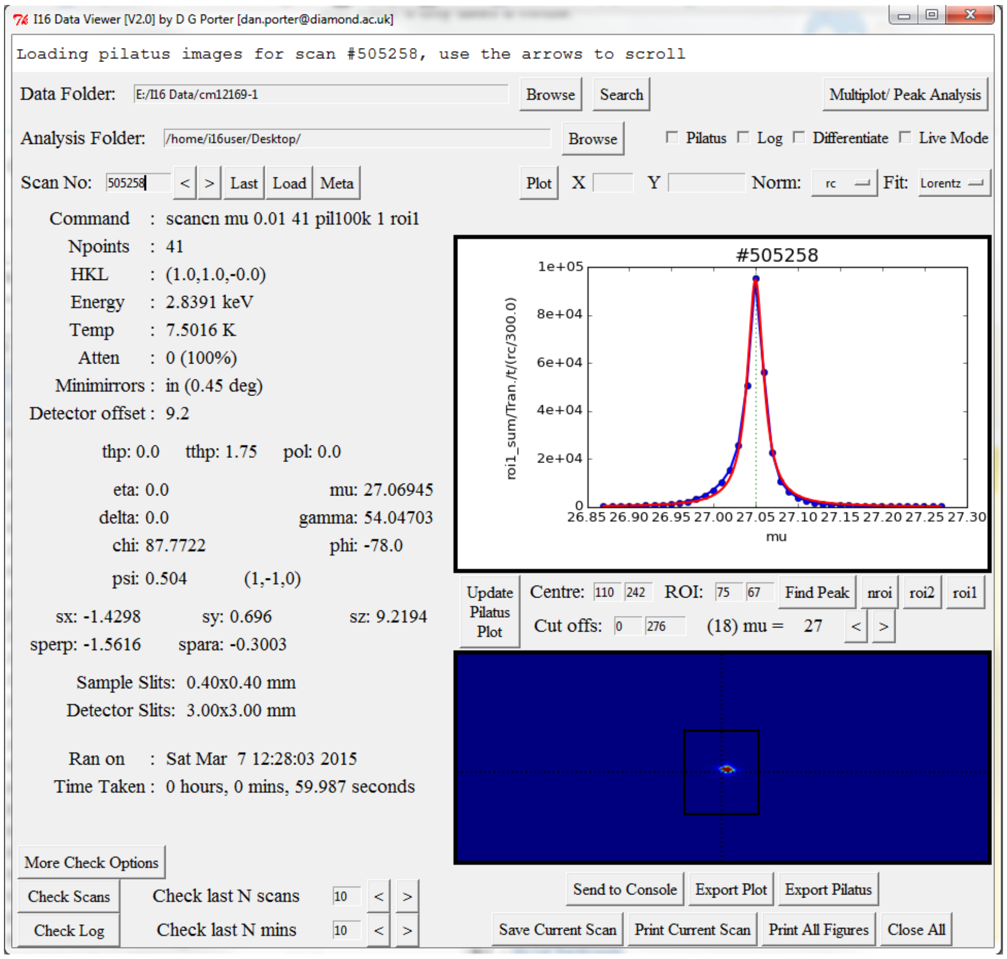
I16 Data Viewer: Manual

Daniel G Porter
dan.porter@diamond.ac.uk

September 27, 2016

Version 2.0



I16 Data Viewer [V2.0] by D G Porter [dan.porter@diamond.ac.uk]

```
Loading pilatus images for scan #505258, use the arrows to scroll
```

Data Folder: E:/I16 Data/cm12169-1

Analysis Folder: /home/i16user/Desktop/

Scan No: 505258

Norm: rc   Fit: Lorentz

Command : scancn mu 0.01 41 pil100k 1 roi1
Npoints : 41
HKL : (1.0,1.0,-0.0)
Energy : 2.8391 keV
Temp : 7.5016 K
Atten : 0 (100%)
Minimirrors : in (0.45 deg)
Detector offset : 9.2

thp: 0.0   tthp: 1.75   pol: 0.0

eta: 0.0   mu: 27.06945
delta: 0.0   gamma: 54.04703
chi: 87.7722   phi: -78.0

psi: 0.504   (1,-1,0)

sx: -1.4298   sy: 0.696   sz: 9.2194
sperp: -1.5616   spara: -0.3003

Sample Slits: 0.40x0.40 mm
Detector Slits: 3.00x3.00 mm

Ran on : Sat Mar 7 12:28:03 2015
Time Taken : 0 hours, 0 mins, 59.987 seconds

Centre: 110 242   ROI: 75 67
Cut offs: 0 276   (18) mu = 27

Check last N scans 10
Check last N mins 10

Contents

1 Introduction

I16 Data Viewer is an program written in python for viewing and simple analysis of data files generated on beamline I16 at Diamond Light Source Ltd. The app has a simple GUI interface allowing you to quickly view details of a particular scan, plot the scan data with automatic selection of the axes and perform basic fitting. There are several GUI windows including the main data viewer, the multi-plot / peak analysis window and the fitting window. Behind these GUIs is a module of python functions that load the data, automatically determine the scan variables, perform normalisation and fitting, amongst other things. It is possible to use these functions without the GUI, allowing for script writing and interactive console use.

The program is split into two files: `Py16progs.py` and `Py16GUI.py`, which contain the data analysis functions and GUI creation respectively.

The software has been designed and optimised for use on the beamline, allowing users to quickly plot previous scans and check experimental details, however it will also work on your own computer, providing some additional software has been installed:

- **Anaconda** - a python distribution favoured at Diamond, though any other would do providing it has the scientific libraries numpy, scipy, matplotlib and tkinter. Note that I16 Data Viewer has been developed and tested in python version 2.7. Care has been taken to make it compatible with python versions >3.0 but it has not yet been tested.

This manual will describe how to run and use the I16 Data Viewer software.

Feedback

If you have any comments, queries, feature requests or bug reports, please get in touch with me at dan.porter@diamond.ac.uk.

Warranty

Please note this software is free, for personal use and supplied without warranty.

2 Running I16 Data Viewer

There are several ways to run I16 Data viewer:

2.1 On the beamline (I16user account)

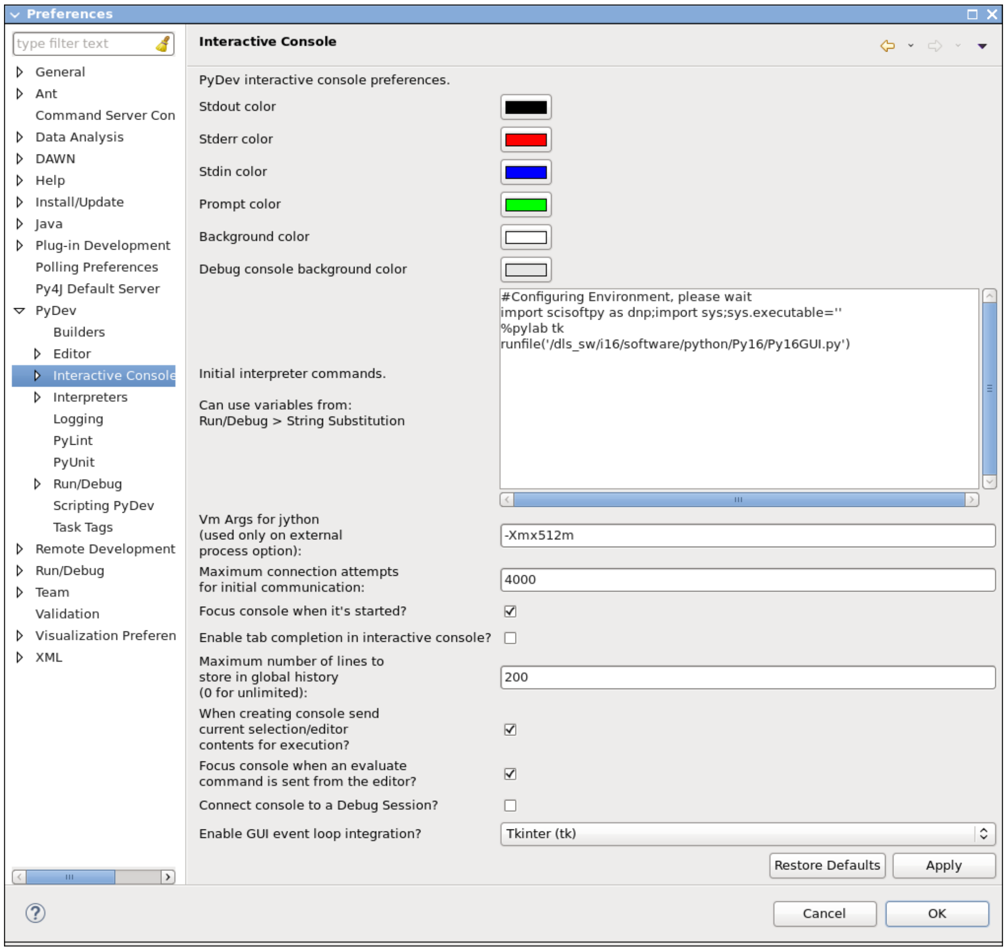
The software is already available on the beamline computers on the I16user account.

- Open Dawn - open terminal, type `module load dawn` then `dawn`
- Open the console - Click *Window >Show Veiw >Console*
- Create new *PyDev Console* using the button at the top right of the console tab.
- I16 Data Viewer should automatically start after a few seconds
- If you close the window, just type `I16_Data_Viewer()` in the console.

2.2 On the NX server (using your own account)

Connecting to the NX server is a useful way to access your data from your own computer, it gives you access to the data files and has Dawn pre-installed. You will have to append a few of the settings in Dawn however, so that the GUI runs correctly.

- Open Dawn - open terminal, type `module load dawn` then `dawn`
- Open preferences - Click *Window >Preferences*
- Check a python interpreter is set up - Click *PyDev >Interpreters >Python Interpreter*, there should be an interpreter listed in the top box, if not, click *Quick Auto-Config* - this should automatically find a python distribution and set it up.
- Go to the interactive console options - *PyDev >Interactive Console*
- If not already there, add the following commands to *Initial interpreter commands*:
  - `\%pylab tk`
  - `runfile('/dls_sw/i16/software/python/Py16/Py16GUI.py')`
- Check that *Tkinter (tk)* is selected in the dropdown menu for *Enable GUI event loop integration?*
- Click *OK* to save and exit the preferences window.
- Open the console - Click *Window >Show Veiw >Console*
- Create new *PyDev Console* using the button at the top right of the console tab.
- I16 Data Viewer should automatically start after a few seconds
- If you close the window, just type `I16_Data_Viewer()` in the console.



Interactive Console

PyDev interactive console preferences.

Stdout color	⬛
Stderr color	🟥
Stdin color	🟦
Prompt color	🟩
Background color	⬜
Debug console background color	⬜
Initial interpreter commands. Can use variables from: Run/Debug > String Substitution	see below
Vm Args for jython (used only on external process option):	-Xmx512m
Maximum connection attempts for initial communication:	4000
Focus console when it's started?	✓
Enable tab completion in interactive console?	☐
Maximum number of lines to store in global history (0 for unlimited):	200
When creating console send current selection/editor contents for execution?	✓
Focus console when an evaluate command is sent from the editor?	✓
Connect console to a Debug Session?	☐
Enable GUI event loop integration?	Tkinter (tk)

```
#Configuring Environment, please wait
import scisoftpy as dnp;import sys;sys.executable=''
%pylab tk
runfile('/dls_sw/i16/software/python/Py16/Py16GUI.py')
```

Figure 1: The Preferences window in Dawn, showing the correct set up for the Interactive Console

2.3 On your own computer

To run I16 Data Viewer on your own computer, you will first need to install a python distribution that includes numpy, scipy, matplotlib and tkinter. The easiest way to do this is to download and install Anaconda. Dawn (Eclipse) is also freely available and is an excellent python editor.

- Anaconda is freely available for Windows, Mac and Linux from Continuum Analytics: `https://www.continuum.io/downloads`. It is best to install the Python 2.7 version as later versions are untested and there are big changes to python in versions >3.0. On Windows, just download the installer and run it.

- Dawn is freely available for Windows, Mac and Linux: `http://www.dawnsci.org/downloads`. On Windows the program comes as a zipped file, just unzip the file into a program directory of your choice and run the executable. You will then need to set up Dawn:

  - Open Dawn

  - Open preferences - Click *Window >Preferences*
  - Check a python interpreter is set up - Click *PyDev >Interpreters >Python Interpreter*, there should be an interpreter listed in the top box, if not, click *Quick Auto-Config* - this should automatically find a python distribution and set it up.
  - Go to the interactive console options - *PyDev >Interactive Console*
  - If not already there, add the following commands to *Initial interpreter commands*: `\%pylab tk`
  - Check that *Tkinter (tk)* is selected in the dropdown menu for *Enable GUI event loop integration?*
  - Click *OK* to save and exit the preferences window.

Once both programs are installed, you can run I16 Data Viewer. You can copy the two files required directly from the beamline computers, they are stored here:

- `/dls_sw/i16/software/python/Py16/Py16progs.py`
- `/dls_sw/i16/software/python/Py16/Py16GUI.py`

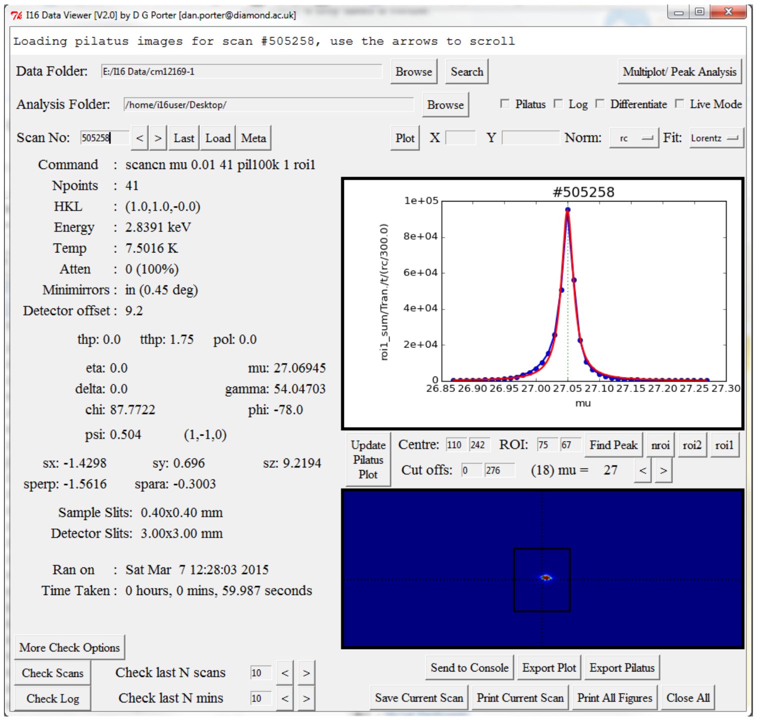
Alternatively you can download the latest version from GitHub at:

- `https://github.com/DanPorter/Py16`

Both files must be stored in the same directory. To run the program, either drag `Py16GUI.py` to the editor window of Dawn, then press `Ctrl+Alt+Enter` to open a console and run the program. Or you can run the script from a terminal/ console/ command prompt:

- Open a console (Linux ¿ Right click in folder, select Terminal. Windows ¿ SHIFT+Right click in folder, select Command Window. Mac ¿ Who knows!)
- type: `python Py16GUI.py`

3 Using I16 Data Viewer



```
Loading pilatus images for scan #505258, use the arrows to scroll
```

Data Folder: E:/I16 Data/cm12169-1

Analysis Folder: /home/i16user/Desktop/

Scan No: 505258   X   Y   Norm: rc   Fit: Lorentz

```
     Command    : scancn mu 0.01 41 pil100k 1 roi1
     Npoints    : 41
     HKL        : (1.0,1.0,-0.0)
     Energy     : 2.8391 keV
     Temp       : 7.5016 K
     Atten      : 0 (100%)
  Minimirrors   : in (0.45 deg)
Detector offset : 9.2

     thp: 0.0    tthp: 1.75    pol: 0.0

     eta: 0.0                  mu: 27.06945
   delta: 0.0               gamma: 54.04703
     chi: 87.7722             phi: -78.0

     psi: 0.504       (1,-1,0)

   sx: -1.4298      sy: 0.696        sz: 9.2194
sperp: -1.5616   spara: -0.3003

     Sample Slits:  0.40x0.40 mm
   Detector Slits:  3.00x3.00 mm

     Ran on     : Sat Mar  7 12:28:03 2015
   Time Taken   : 0 hours, 0 mins, 59.987 seconds
```

Centre: 110 242   ROI: 75 67

Cut offs: 0 276   (18) mu = 27

Check last N scans 10

Check last N mins 10

Figure 2: The I16 Data Viewer main window

Upon opening the I16 Data Viewer GUI, these steps will allow you to plot your data:

3.1 Plotting a scan

- Click the top *Browse* button to locate the directory of your data files.
- Click the next *Browse* button to locate the your analysis directory - this is where plots and scripts will be saved.
- Click the *Last* button to load and plot the latest scan.
- Alternatively, enter the required scan number in the *Scan No:* box and press Enter.
- Use the <and >arrow buttons by *Scan No:* to move between files.
- The button *Load* will load the given scan number without plotting.
- The button *Plot* will re-plot the given scan number using the plot options.

3.2 Plot options

- To change the variable plotted on the x-axis, enter the scan variable in *X* and click *Plot*.

- To change the variable plotted on the y-axis, enter the scan variable in *Y* and click *Plot*.
- You can normalise the plots in different ways using the dropdown menu *Norm:*
  - **rc** - Normalise by exposure time, attenuator transmission and ring current / 300, where 300 is the normal ring current.
  - **ic1** - Normalise by exposure time, attenuator transmission and monitor
  - **none** - No normalisation
- You can apply a peak profile to your plot by selecting your preferred profile in the *Fit:* and then click *Plot*. The height, width, centre and area are given in the console, Note that you may need to press Enter on the console to force it to update. You can also *Export Plot* and the values will be given on the plot.
- To save the plot, click the button *Save Current Scan* - a png image of the plot will be saved to your analysis folder.
- To print the plot directly, click the button *Print Current Scan* - the image of the plot will be sent directly to the printer. Note this feature is only available on the beamline computers.

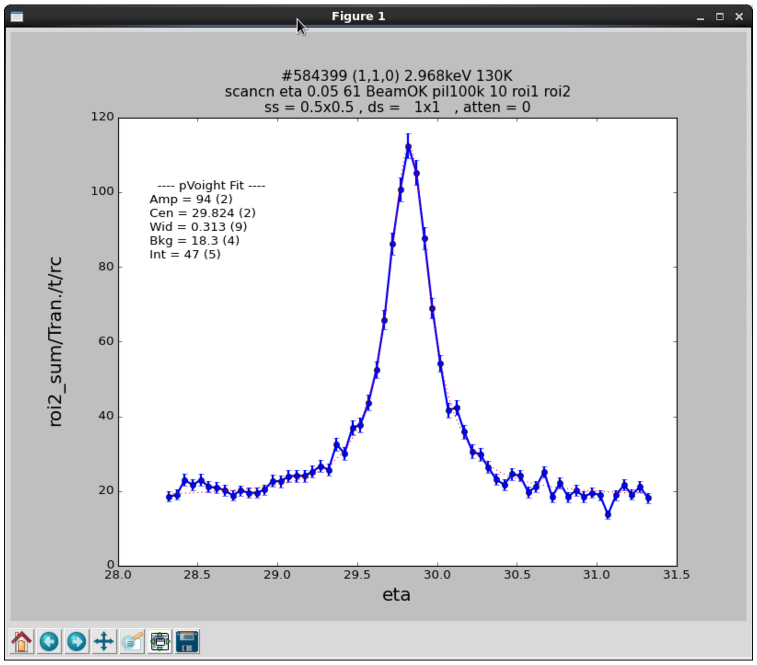


Figure 3: Plot of an individual scan, with a fitted peak profile, using either *Export Plot* in the GUI or `pp.plotscan(584399,fit='pVoight')` in the console.

3.3 Plotting Pilatus images

- To view the **Pilatus** images, click the *Update Pilatus Plot* button.
- The intensity max and min are chosen automatically, but you can change these by changing *Cut offs:* and pressing Enter.
- Using the <and >arrow buttons just above the pilatus plot, you can move through the different frames of the scan. The position of each frame is indicated in the main plot window by a dotted line.
- To see a larger version of the plot, press *Export Pilatus* to create a separate window.
- The cross-hair shown on the pilatus plot indicates the defined pilatus centre, this can be changed by altering the boxes next to *Centre:*
- The box drawn on the pilatus plot indicates the default region of interest, or ROI2, which is centred at the pilatus centre and has a size defined by the values in *ROI:*.
- By changing the *Centre* and *ROI* values, you can define your own region of interest, click *Update Pilatus Plot* to see the new region of interest on the plot, or click *nroi* (new region of interest) to send the region of interest value to *Y*, then press *Plot* to see the integrated area over the scan.
- You can automatically search for the brightest point in the scan using *Find Peak*, this will change the *Centre* to the maximum pixel position.

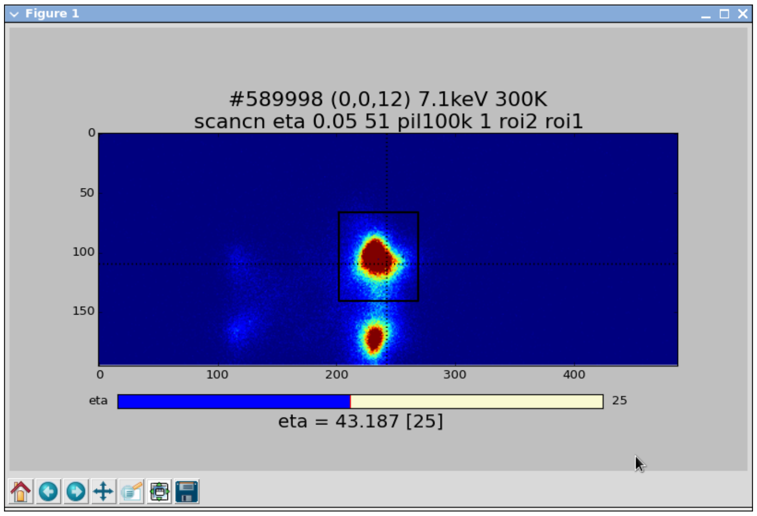


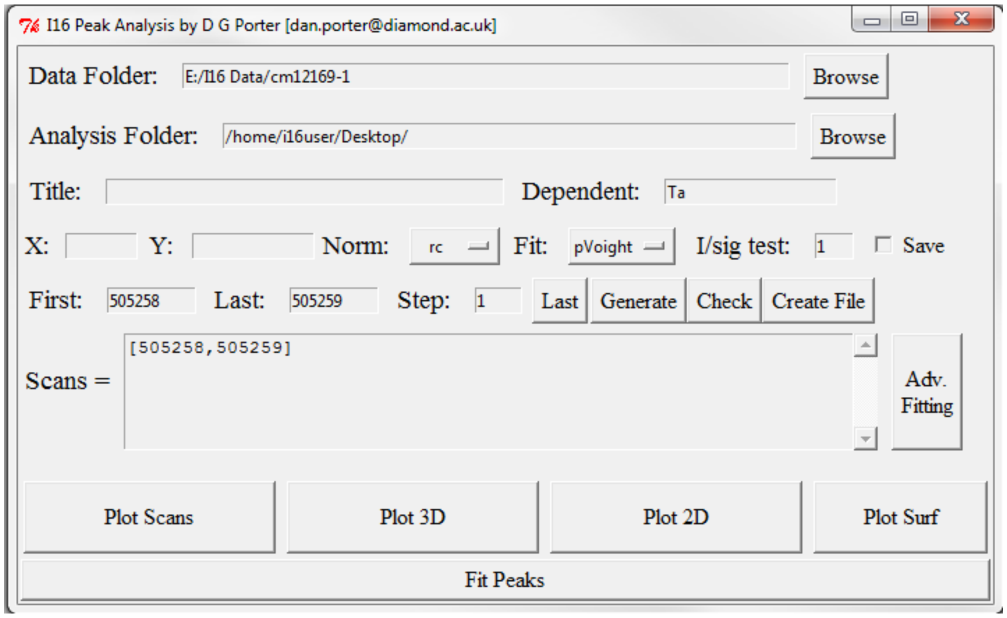
Figure 1
#589998 (0,0,12) 7.1keV 300K
scancn eta 0.05 51 pil100k 1 roi2 roi1
0
50
100
150
0
100
200
300
400
eta
25
eta = 43.187 [25]

Figure 4: Displaying the frames from a scan with the pilatus detector, using either *Export Pilatus* in the GUI or `pp.plotpil()! in the console`

3.4 Defining new regions of interest

4 Multi-scan plots and peak analysis

You can access the peak analysis GUI by clicking *Multiplot/ Peak Analysis* in the main GUI or typing `I16_Peak_Analysis()` in the console.


I16 Peak Analysis by D G Porter [dan.porter@diamond.ac.uk]

Data Folder: E:/I16 Data/cm12169-1

Analysis Folder: /home/i16user/Desktop/

Title:

Dependent: Ta

X:

Y:

Norm: rc

Fit: pVoight

I/sig test: 1

Save: ☐

First: 505258

Last: 505259

Step: 1

Scans = [505258,505259]

Figure 5: The I16 Peak Analysis GUI

As with the main GUI, start by entering the directory locations of the data files and your analysis folder. You can add a *Title* that will be added to the plots and the filenames of saved images. The value entered in the *Dependent* box will be used as the additional axis, therefore this should reflect the experimental variable that changes between scans, such as temperature (Ta), azimuthal angle (psi) or energy (energy).

The *Y axis*, *Norm* and *Fit* options perform the same functions here as in the main GUI. The *I/sig test* value defines the intensity / error ratio required for a scan to be defined as a peak, allowing you to stop fitting profiles to data where a peak disappears, such as the temperature dependence of a purely magnetic reflection. When the *save* tickbox is ticked, the generated plots will be saved to the *Analysis Folder*.

4.1 Defining the set of scan numbers

Scripted measurements on I16 usually happen within a **for** loop, with the same scan being performed over and over as some other variable is changed, for example a reflection may be re-scanned at multiple temperatures. To plot these scans, enter the numbers of the *First* and *Last* scans in the relevant boxes, and the *Step* between them. You can click the button *Last* to get the most recent scan number. Once these are entered, you must click the *Generate* button - this will display all the required scan numbers in the *Scans =* box, allowing you to remove erroneous scans manually. The scans that will be plotted are those in the *Scans =* box. Note that within the *Scans =* box you are entering **Python** code that will be evaluated, and therefore numbers must start and end lists with square brackets and separate numbers with commas: `[1,2,3]`.

Once the list is generated in the *Scans =* box, you can plot the data in various ways, or you can perform peak fitting and plot the height/ area/ centre/ width against your *Dependent* variable. Using the *Check* button you can see basic details of each scan in the console (you may need to press enter to update the console).

4.2 Types of plot

Plot Type	**Description**
Plot Scans	Plots each scan directly on top of one another in 2D, *Dependent* variable is used as a label.
Plot 3D	Plots the scan variable vs. *Dependent* variable vs. intensity on a 3D plot.
Plot 2D	Plots the scan variable vs. *Dependent* variable 2D surface with different colours representing varying intensity.
Plot Surf	Plots the scan variable vs. *Dependent* variable vs. intensity on a 3D plot with colour surface.

4.3 Peak Fitting

For sets of scans of simple peak shapes, you can use the automated peak fitting functions that will attempt to fit your preferred profile shape to each scan, then plot the resulting area/ width/ centre against the *Dependent* variable. Plots of each fit will also be generated for your inspection. To run the automatic peak fitting, simply *Generate* the list of scans and click the large *Fit Peaks* button, for large scan lists this can take a little time, so please be patient. If the *Save* tickbox is ticked, the resulting plots and files containing the results will be saved to the *Analysis Folder*.

The fitting routine is designed to be highly general and robust for both strong and weak data. Here is a brief explanation of the fitting routine:

- A test is performed to check if a peak is present in the scan, if the intensity/ error ratio is greater than the *I/sig test* value, a profile fit will be attempted, otherwise a linear background will be fitted.
- Estimates of the height, width, centre and background will be made from the scan.
- Using these estimates as starting values, the `scipy.optimize.curve_fit` function is used with the chosen profile shape to optimise the profile parameters.
- This is performed multiple times with slight variations on the input parameters, a valid solution with the lowest χ^2 is kept.

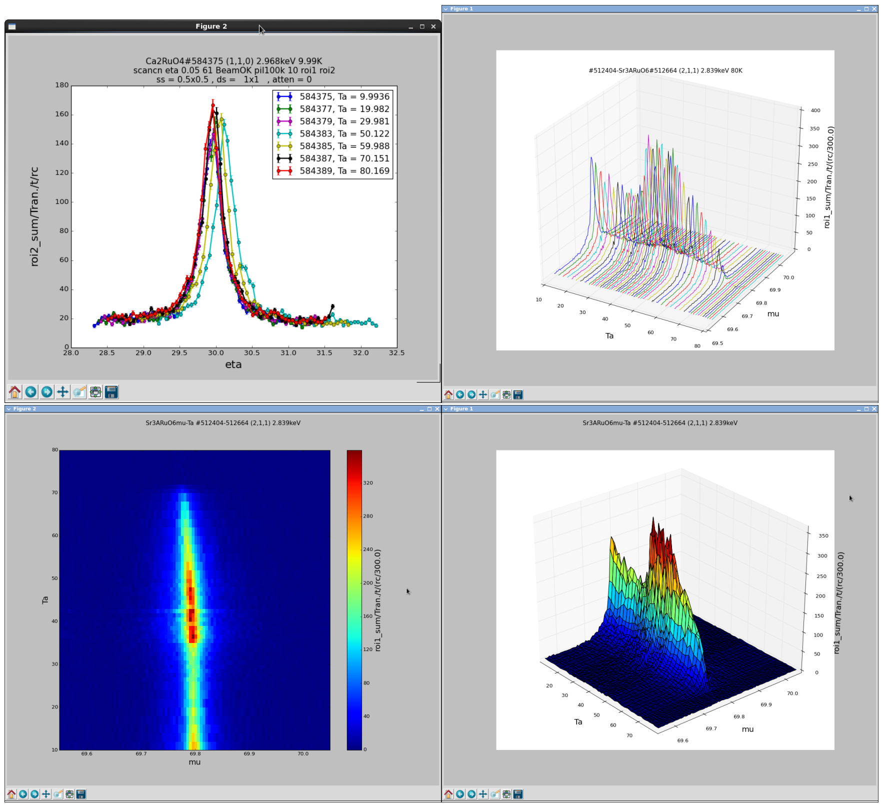


Figure 2
Ca2RuO4#584375 (1,1,0) 2.968keV 9.99K
scancn eta 0.05 61 BeamOK pil100k 10 roi1 roi2
ss = 0.5x0.5 , ds = 1x1 , atten = 0
584375, Ta = 9.9936
584377, Ta = 19.982
584379, Ta = 29.981
584383, Ta = 50.122
584385, Ta = 59.988
584387, Ta = 70.151
584389, Ta = 80.169
roi2_sum/Tran./t/rc
eta
Figure 1
#512404-Sr3ARuO6#512664 (2,1,1) 2.839keV 80K
roi1_sum/Tran./t/(rc/300.0)
Ta
mu
Sr3ARuO6mu-Ta #512404-512664 (2,1,1) 2.839keV
Ta
mu
roi1_sum/Tran./t/(rc/300.0)
Sr3ARuO6mu-Ta #512404-512664 (2,1,1) 2.839keV
Ta
mu
roi1_sum/Tran./t/(rc/300.0)

Figure 6: Types of plot in I16 Peak Analysis

- The area and other information about the profile is calculated.

4.4 Creating a script

It is very useful to have a script that will perform this fitting so that it can be done quickly again at a later time. The *Create File* button will automatically generate a script in your Analysis Folder that will import **Py16progs.py**, set the data directory and re-perform the fitting routine. To run the script, drag the file to the **Dawn** editor window, then press **Ctrl+Alt+Enter**.

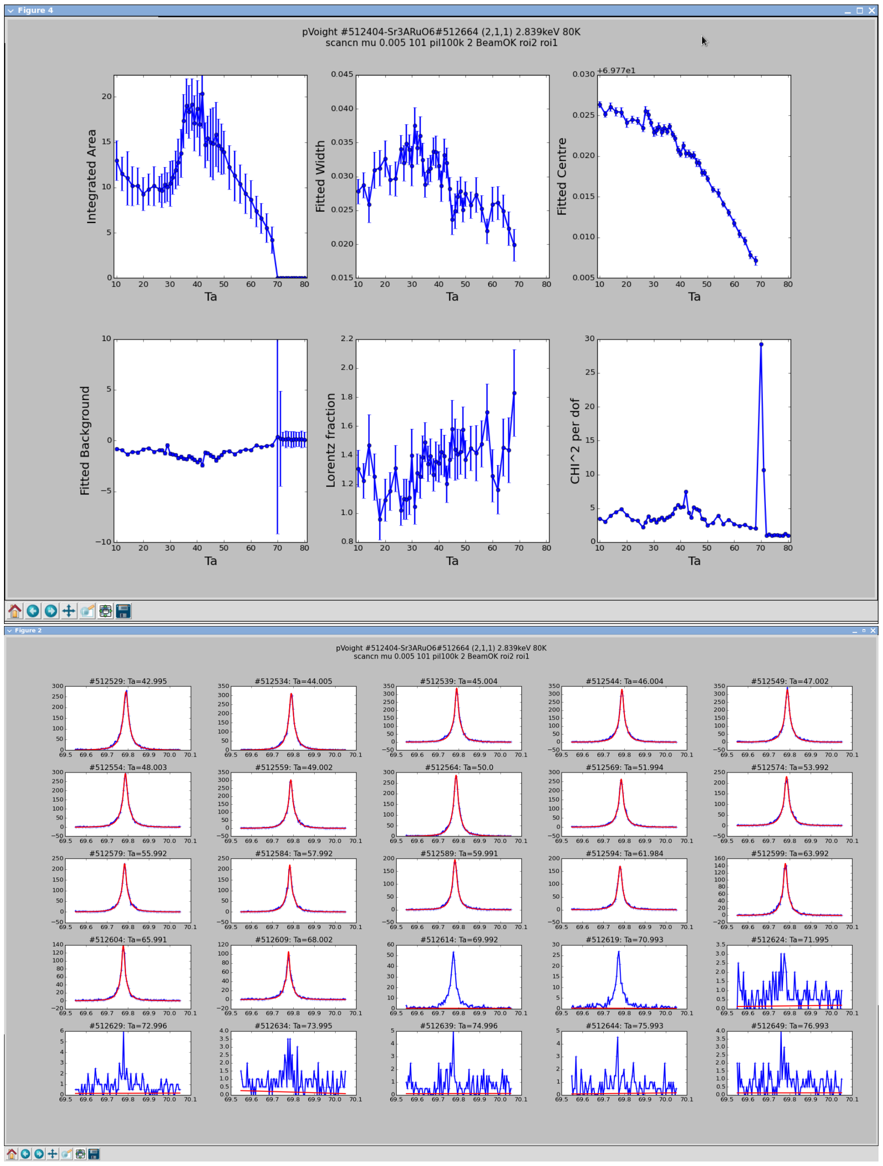


Figure 7: Plots generated by *Fit Peaks*.

5 Using the console

When functions in the **I16 Data Viewer** GUI are run, functions are called from the Python module **Py16progs.py** at the console. It is very simple bypass the GUI and perform the same functions using the console. Many of the functions have more features that can only be accessed by giving additional inputs via the console or a script.

- The Python module **Py16progs.py** is imported to the console namespace by default when **Py16GUI.py** is run as the variable `pp`. If it has not been imported, enter `import Py16progs as pp`
- The data directory, analysis directory and other experimental values are set as parameters in the module:
  - `pp.filedir = 'data\directory'` >sets the directory to read data from.
  - `pp.savedir = 'save\directory'` >sets the directory to save images and scripts.
  - `pp.normby = 'rc'` >Incident beam normalisation option: `'rc'`, `'ic1'` or `'none'`
- To check the experimental data directory is correct, use `pp.checkexp()` >This will give the date and number of scans in the directory.
- To list details of a set of scans, use `pp.checkscan(first,last)` where `first` and `last` are the first and last scan numbers of the set.
- The experiment **Log File** can be read using `pp.checklog()`, see `help(pp.checklog)` for more details.
- Individual scan data can be read using

  ```
  d = pp.readscan(123456)
  ```

  where the number is the desired scan number. The output variable `d` is a python dictionary as loaded by **SciSoftPy**, with arrays of scaned values given in the top level (e.g. `d.eta` and `d.roi2_sum`) and ancillary scan data stored in the metadata (e.g. `d.metadata.Ta` and `d.metadata.Energy`). Use `d.keys()` to list the available values.
- A more automated way to load data is to use

  ```
  x,y,dy,varx,vary,ttl,d = getdata(123456)
  ```

  This will automatically choose the scanned variable and intensity values, returning these as arrays `x` and `y` respectively. The error on `y` is given in the array `dy`. `varx`, `vary` and `ttl` are strings that describe the scan. `d` is the same as is given by `pp.readscan`. Intensity values are normalised by exposure time, transmission and the choosen normalisation option. You can directly plot or analyse the arrays `x`, `y` and `dy`, making for fast and efficient interpretation of data.

- Individual scans can be plotted using `pp.plotscan(123456)`, which will choose the x and y axis automatically.
- Profile fitting can be performed by adding the `fit` option: `pp.plotscan(123456,fit='gauss')` >The fitted peakshape and parameters will be displayed on the plot.
- Additional inputs can be added to any of these functions to specify a particular x or y axis using the `varx` and `vary` options: `pp.plotscan(123456,varx='eta',vary='sum')`.
- There are many additional functions available, many of these have advanced functionality and additional options. A list of functions is given in the appendix of this manual, and all functions are documented.
- To read the documentation for a particular function use python's help function:

  ```
  help(pp.plotscan)
  ```

6 Writing analysis scripts

For data analysis it is useful to write sequences of commands into a python script, which can be executed repeatedly with small variations to see the results, such as re-fitting data with different profiles or regions of interest, for example. Writing a python script is just the same as writing commands in the console, except that at the start of the script you must set the python path and import the required python modules. The following text should be included at the top of your script:

```
import sys
f='/dls_sw/i16/software/python/Py16/Py16progs.py'
sys.path.insert(0,f)
import Py16progs as dp
dp.filedir = '/dls/i16/data/2015/cm12169-2'
dp.savedir='/home/i16user/Desktop'
```

An example script is given in the same directory as **Py16progs.py**. You can also automatically generate scripts from the **I16 Peak Analysis** GUI using the button *Create File*. To run a script from the **Dawn editor**, select the file and press **Ctrl+Alt+Enter**.

A List of Py16progs.py functions

See `help(pp.function_name)` for more information.

- `d = readscan(num)` >Loads data from the scan number *num*. `d` contains arrays of scanned values and metadata, accessed using, for example `d.eta` or `d.metadata.Transmission`. Use `d.keys()` or `d.metadata.keys()` to see available values.
- `x,y,dy,varx,vary,ttl,d = getdata(num/d,'varx','vary',save=None)` >Automated data loading for a single scan. Picks x and y values based on scan command. Intensity values (y) are normalised for exposure time, attenuator transmission and incident beam intensity.
- `x,y,z,varx,vary,varz,ttl = joindata([nums],'varx','vary','varz')` >Automated data loading for a set of scans, loading into 2D arrays.
- `vol = getvol(num)` >Loads all frames of a pilatus scan as a 3D volumetric array with hot and broken pixels removed.
- `ROI_sum,ROI_maxval,ROI_bkg = pilroi(num,ROIcen,ROIsize,findpeak,peakregion)` >Define a new region of interest for pilatus images, returning the integrated area.
- `num = latest()` >Returns the latest scan number in the data folder.
- `checkexp()` >Displays information about the scans within the experiment folder.
- `checkscan(num1=None,num2=None)` >List details of scans between num1 and num2.
- `checklog(time=None,mins=2,cmd=False)` >Display text from the log file.
- `prend(start,end)` >Predict the end of a set of scans.
- `polflip(sigsig,sigpi,fit='Gauss',output=False)` >Calculate the flipping ratio between sigma-sigma and sigma-pi scans.
- `metaprint(d1,d2=None)` >Display all metadata for scan d1. If d2 is included the metadata will be compared.
- `savedata(num/d,'varx','vary')` >Save the scan values as a text file, which can be easily read using `x,y,dy=numpy.loadtxt('filename.dat')`
- `plotscan(num=None,vary='',fit=None,save=None)` >Plots the scan, choosing x and y automatically.
- `plotpil(num,cax=None,imnum=None)` >Plots images from a scan using the pilatus detector.
- `plotscans3D(runs,depvar='Ta',vary='',save=None)` >Plots a set of scans in 3D vs. the depdendent variable `depvar`.

- `fit,err = fit_scans(runs,depvar='Ta',vary='',save=None)` >Automated fitting routine for a set of scans
- `fit,err = load_fits(runs,depvar='Ta')` >Loads data generated by `fit_scans`
- `out,err = peakfit(x,y,dy,type='dVoight')` >General peak fitting function for 1D data. Outputs dictionaries of fitted parameters (out) and associated uncertainties (err).
- `ROIcen_ij,ROIcen_frame = pilpeak(vol,test = 1,disp=False)` >Finds the peak position in a volumetric array.
- `ispeak(Y,test = 1,disp=False)` >Determines whether 1D array Y contains a peak.
- `labels(title,xlabel,ylabel,zlabel)` >Adds formated labels to a figure
- `saveplot(name,dpi=None)` >Saves a figure as an image
- `vals = frange(start,stop=None,step=1)` >Operates the same was as frange on the beamline

B Example Script

```python
# Diamond I16 Analysis Script
# [Experiment Name]
# [Script Description]
#
# By [User]
# 18/08/2016

import sys,os
import matplotlib.pyplot as plt # Plotting
from mpl_toolkits.mplot3d import Axes3D # 3D plotting

# Load Py16progs
sys.path.insert(0,'/dls_sw/i16/software/python/Py16/Py16progs.py') # location of Py16progs
import Py16progs as dp

# Current Directory
cf=os.path.dirname(__file__)

# Directory to load data from
dp.filedir = '/dls/i16/data/2015/cm12169-2'

# Directory to save files to
dp.savedir='/home/i16user/Desktop'

# Experiment Parameters
dp.exp_ring_current = 300.0 # Average ring current for normalisation
dp.exp_monitor = 800.0 # Average monitor current for normalisation
dp.normby = 'rc' # ring current ('rc'), monitor ('ic1') or none ('none')
dp.pil_centre = [110, 242]
dp.peakregion=[7,153,186,332] # Search for peaks within this area of the detector [min_y,min_x,max_y,max_x
dp.exp_title = ''

# Scan numbers
scans = range(512404,512664,5) + [512665,512666]

# Multi-scan plots
help(dp.plotscan) # See the function documentation!
dp.plotscan(scans)
dp.plotscans3D(scans)
dp.plotscansSURF(scans)

# Automatic Peak Fitting & Integration
fit,err = dp.fit_scans(scans,vary='roi1_sum',depvar='Ta',peaktest=60,fit_type='pVoight',saveFIT=None,save=

# Load fitted data:
#fit,err = dp.load_fits(scans,depvar='Ta',fit_type='pVoight')

# Manual analysis of data
temp,value = [],[]
for scn in scans:
        x,y,dy,varx,vary,ttl,d = dp.getdata(scn,vary='roi1_sum') # read normalised data
        temp += [d.metadata.Ta]
        value += [sum(y)]

# Basic plotting with Matplotlib
plt.figure()
plt.plot(temp,value)
dp.labels('Title','temp','value')
```