Data management software for characterization measurements

This Python script allows for data management for measurements performed with XRD, ellipsometry, SEM/EDX, XPS, UPS, REELS, and Raman spectroscopy. Features include:

- Loading multiple datasets into one data frame.
- Aligning measurements to a custom grid, even if measurement coordinates differ from said grid.
- Selecting specific datatypes at specific coordinates.
- Plotting data.
- Peak fitting for XRD and Raman data.
- Finding valence band onset for UPS data.
- Peak and onset fitting for REELS data.
- Data interpolation.
- Saving / loading data frames.

The script requires data provided through CSV or Excel files exported from associated characterization measurement software. Exact data export procedures are explained at the end of this readme.

XRD: SmartLab Studio II (CSV)
Ellipsometry: CompleteEASE (CSV)
SEM: AZtec LayerProbe and large area mapping (Excel)
XPS, UPS, and Raman Spectroscopy: Avantage (CSV and Excel)

Usage – Loading Data

The script is operated with functions through a Python interface. Create a new Python file in the same folder as the "functions" file, and load the functions as follows:

```
from functions import *
```

To begin assigning data, a measurement grid must be defined with the measurement_grid function. The options define number of columns, number of rows, length (x) and height (y) of grid **in mm**, and the coordinate of the lower-left grid corner.

```
grid = measurement_grid(ncolumns, nrows, gridlength, gridheight, startx, starty)
```

Data can now be loaded. This is done with read_[datatype], where [datatype] is either ellipsometry_thickness, ellipsometry_nk, XRD, XPS, UPS, raman or layerprobe. The function defines both **data and coordinates** of the measurements. The parameters are the file name (or file path if working in a different directory), and the defined grid. The coordinates are extracted

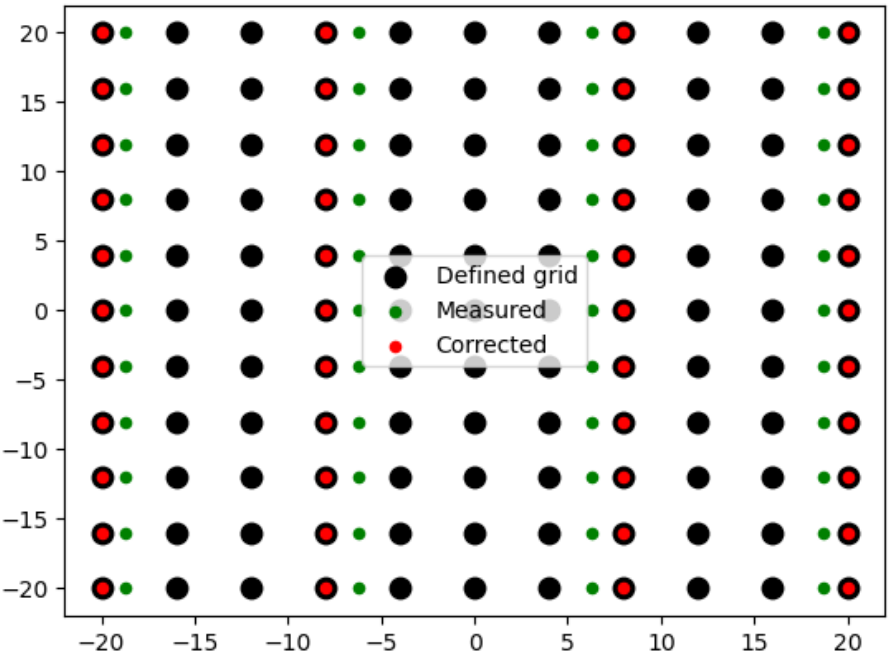
from the provided datafile and are then aligned to the closest grid coordinate point. Ensure that your sample has consistent rotation between different measurements, as there is no way for the program to correct the rotation of coordinates. If any grid coordinate does not have any associated data, it will simply be skipped.

As an example, here we load some ellipsometry thickness data:

```
data, coords = read_ellipsometry_thickness("ellipsometry_thickness.txt", grid)
```

You can visualize the grid, measurement coordinates, and corrected measurement coordinates with the plot_grid function. Shown below is a 11 by 11 grid, 40 by 40 mm dimensions, -20, -20 mm start point with some data that does not quite fit the grid. If you made a mistake when defining the grid, you can simply redefine it and read the data again with the new grid.

```
plot_grid(coords, grid)
```



If multiple separate samples should be made part of one large grid, the coordinates of a dataset can be offset with translate_data:

```
data, coords = translate_data(data, x, y)
```

Usage – Handling Data

The data from read_[datatype] is saved as a data frame. Multiple data frames can be combined with combine_data. If you would like to treatment on data, **you should do so before combining data frames.**

```
datalist = [XRD_data, ellipsometry_data, layerprobe_data, XPS_data]

dataframe = combine_data(datalist)
```

The data can be read with get_data. Options define which type of data to read from the frame, and the coordinates (x, y) of the data. Leave type as 'all' to select all types. Leave coordinates as 'all' to select all coordinates. These options are also default options if they are left blank.

```
get_data(dataframe, 'datatype', x, y)

# Print entire dataframe
get_data(dataframe)
# Print all data at a coordinate
get_data(dataframe, 'all', -20, -20)
# Print all heights
get_data(dataframe, 'Z (nm)')
# Print only height at a coordinate
get_data(dataframe, 'Z (nm)', -20, -20)
```

The data can be saved and loaded with save_data and load_data.

```
save_data(dataframe, "saved_data.txt")

dataframe = load_data("saved_data.txt")
```

Usage – Visualization

You can plot two types of data against each other with the plot_data function. You must specify a datatype for both the x and y axis, as well as any amount of plotting options you would like. A full selection of available options as well as an example are shown on the next page. The ordering of provided options is not important if you input "*option* = *value*". If an option is left blank, the default value is selected, as shown by an underline.

```
plot_data(data, 'datatype_x', 'datatype_y', options…)
```

A list of options is shown below. For the x, y, and datatype_select_value options, the selected values do not need to be exact; the closest value is automatically chosen.

```
x = 'all' / [coordinate_list] # Provide a list of x- and y-coordinates to limit-
points to plot. You must provide a list for each of equal length.
y = 'all' / [coordinate_list]
datatype_select = None / 'datatype' # For data with multiple values (e.g. ellipsom-
etry n is given for a certain energy), select the column name (e.g. "Energy (eV)")
here.
datatype_select_value = None / [value] # Provide the row value of the
"datatype_select" column (e.g. 4.2 to select 4.2 eV).
legend = True / False # Display legend
scatter_plot = True / False # Display scatter plot instead of line plot
plotscale = 'linear' / 'log' # Choose linear or log y-axis scale
title = 'auto' / 'Custom Title' / None # Choose title
```

Example options for a plot of refractive index n over a ratio of Zr/Ti is shown below. The produced plot is shown on the next page.

```
# We have some XPS and ellipsometry nk data
datalist = [ZrTiO_XPS_data, ZrTiO_ellipsometry_nk_data]
dataframe = combine_data(datalist)

# Example of selected options:
# datatype_x = 'Zr Total / Ti Total'
# datatype_y = 'n'
# datatype_select = 'Energy (eV)'
# datatype_select_value = 4.323
# scatter_plot = True
# title = 'n over Zr/Ti ratio'

# Plot data
plot_data(dataframe, datatype_x = 'Zr Total / Ti Total', datatype_y = 'n',
datatype_select = 'Energy (eV)', datatype_select_value = 4.323, scatter_plot =
True, title = 'n over Zr/Ti ratio)
```

If we wanted to only plot a few coordinates instead of the default “all”, e.g. points [-4,0], [0,0], and [4,0], we could have added:

```
# Coordinate examples
# x = [-4, 0, 4]
# y = [0, 0, 0]
```

You can also produce a heatmap of a dataset with the heatmap function. You must specify the type of data to plot, and optionally give your plot a title.

```
# Produce a heatmap of selected data
heatmap(data, 'datatype', options…)
```

A list of options is shown below. For the datatype_select_value, excluded_x, and excluded_y options, the selected values do not need to be exact; the closest value is automatically chosen.

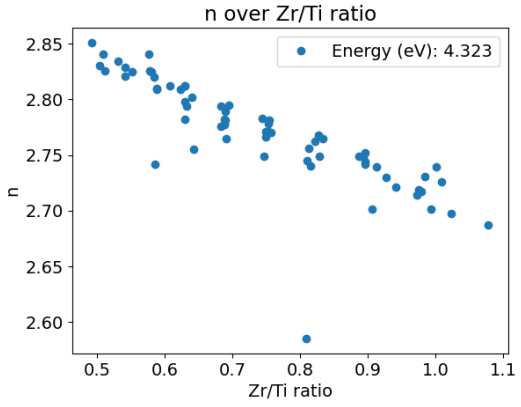
```
datatype_select = None / 'datatype' # For data with multiple values (e.g. ellipsometry n is given for a certain energy), select the column name (e.g. "Energy (eV)") here.
datatype_select_value = None / [value] # Provide the row value of the "datatype_select" column. (e.g. 4.2 to select 4.2 eV).
excluded_x = 'all' / [coordinate_list] # Provide a list of x- and y-coordinates to exclude from the dataset. You must provide a list for each of equal length.
excluded_y = 'all' / [coordinate_list]
min_limit = None / number # Exclude data below the limit
max_limit = None / number # Exclude data above the limit
title = "Custom Title" / None # Choose title
```

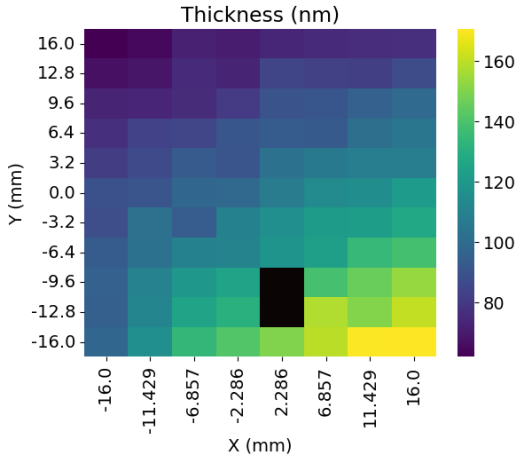
Example heatmap with two points excluded. Note **the coordinates do not have to be exact**.

```
# Example of selected options
# datatype = 'Layer 1 Thickness (nm)'
# title = 'Thickness (nm)'
# excluded_x = [2, 2]
# excluded_y = [-12, -10]

heatmap(data, datatype = 'Layer 1 Thickness (nm)', title = 'Thickness (nm)', excluded_x = [2, 2], excluded_y = [-12, -10])
```

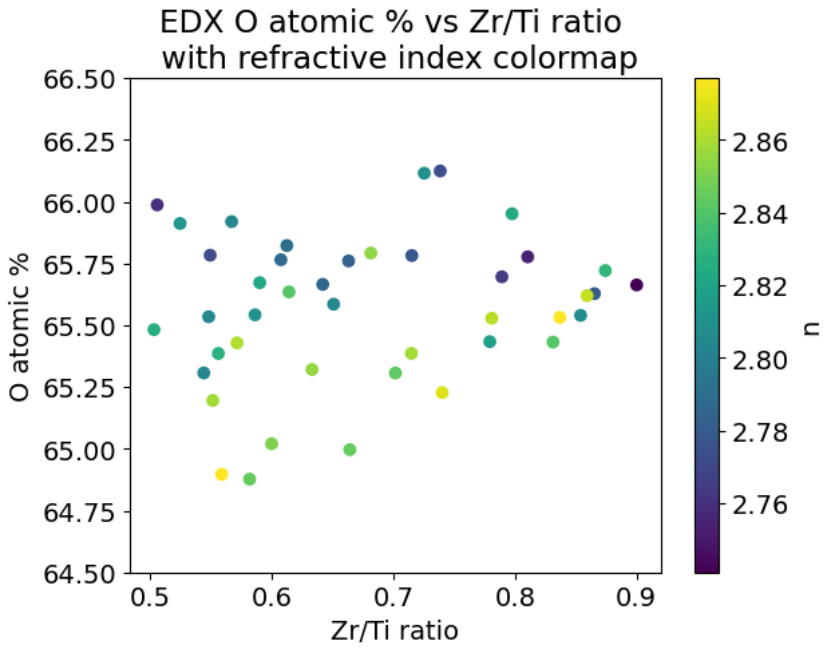
The two plots produced by the examples:





You can also plot a scatter plot with a colormap using the plot_scatter_colormap function. It functions in the same way the regular plot_data functions, except for also having to define a datatype_z as the basis for the colormap.

```
plot_scatter_colormap(data, datatype_x, datatype_y, datatype_z, x = "all", y =
"all", datatype_select = None, datatype_select_value = None, min_limit = None,
max_limit = None, plotscale = "linear", title = "auto"):
```



Usage – Additional Functions

Data interpolation

All datatypes can be interpolated using bicubic interpolation. This is useful if you want to reduce the number of measurements for slow characterization equipment. The data is interpolated over a grid defined with measurement_grid. If the provided grid spans a larger area than the original data, the region completely outside the original data is left blank, as no interpolation can be done there. Consider if your interpolated data is physically accurate and interpolatable when using this function.

```
interpolated_data = interpolate_grid(data, grid)
```

Math between columns of data

The function math_on_columns can find the results of an operation between two columns of your choice. The types provided must be **column names**. You can specify the type of operation (“+”, “-“, “*”, “/”) to add, subtract, multiply, or divide columns. The result is added as a new column at the end of each coordinate point. **This operation must be performed before combining datasets.**

An example below shows a use case for finding the ratio between elements P and S for a LayerProbe dataset.

```
# We have some “layerprobe_data”, and we want to find ratio between P and S.
type1 = "Layer 1 P Atomic %"
type2 = "Layer 1 S Atomic %"
layerprobe_data = math_on_columns(layerprobe_data, type1, type2, "/")
```

Convert ellipsometry wavelength column to eV

If ellipsometry n/k data is exported in wavelengths, it can be converted to an eV scale instead.

$$E = \frac{hc}{\lambda}, h = 4.135 \cdot 10^{-15}\,\frac{\mathrm{eV}}{\mathrm{Hz}}, c = 3 \cdot 10^{8}\,\frac{\mathrm{m}}{\mathrm{s}}$$

```
ellipsometry_nk_data = convert_to_eV(ellipsometry_nk_data)
```

XRD peak fitting

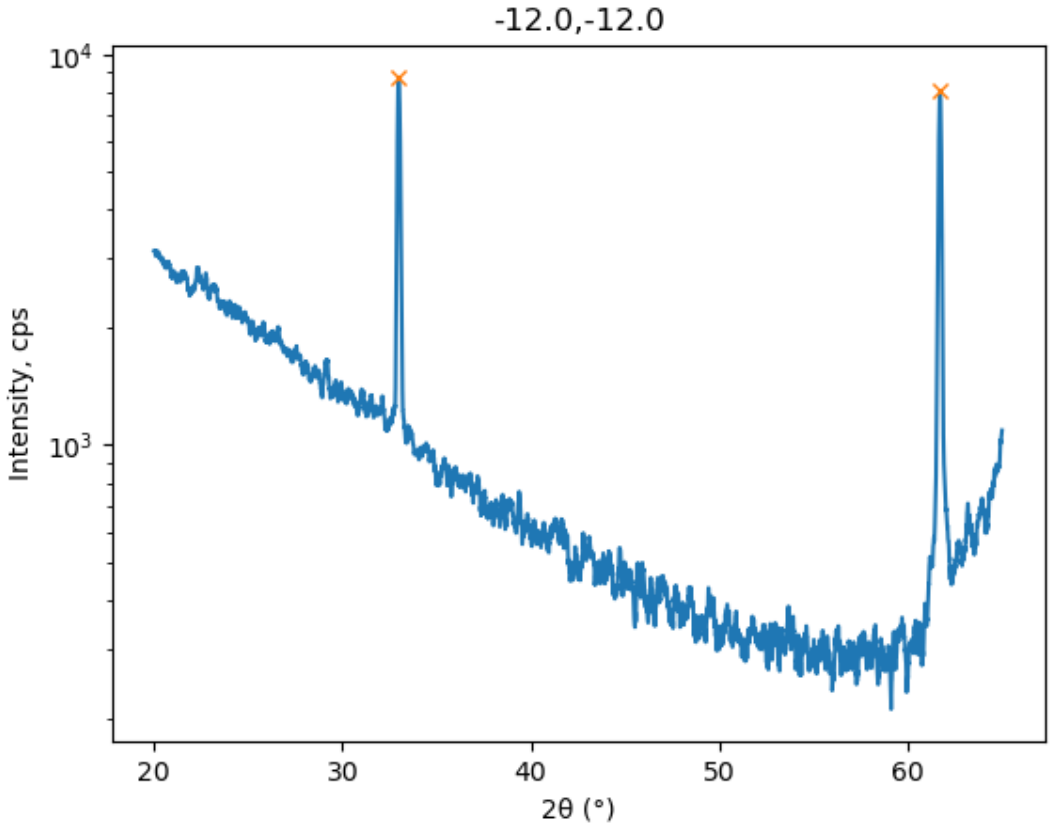
XRD analysis is done in two stages. First, the initial_peaks function detects peak center locations and amplitudes. Input is data frame from read_XRD. The data is filtered to avoid false peaks. A data range can be specified to only fit part of the measurement, this must be provided as the index of the data.

Filter strength is adjusted with filterstrength. Peak detection sensitivity can be adjusted based on how much the peak stands out from the surrounding signal with peakprominence and based on the width of the peaks with peakwidth. For these three options, you can provide either a list of [minimum, maximum] or a single value, interpreted as minimum.

Plots can be toggled on or off using withplots.

```
Peaks, XRD_data_fit = initial_peaks(data, dataRangeMin, dataRangeMax,
filterstrength, peakprominence, peakwidth, withplots = True, plotscale = 'log')
```

If plots are toggled on, they will be output for every point as code is running.



1 Example plot output, filterstrength = 30, peakwidth = 1, peakprominence = 350. Unfiltered data can be seen later under XRD fit.

The function outputs a data frame with peak locations and intensity and a data frame with the range limited unfiltered data.

Peaks

✓ 0.0s

Coordinate	12.0,12.0		-12.0,12.0		12.0,-12.0		-12.0,-12.0	
Data type	2θ (°)	Intensity, cps	2θ (°)	Intensity, cps	2θ (°)	Intensity, cps	2θ (°)	Intensity, cps
0	32.98	9366.667	33.00	10733.33	33.00	10466.670	32.98	9366.667
1	61.70	8966.667	61.69	10100.00	61.69	8033.333	61.69	9566.667

XRD_data_fit

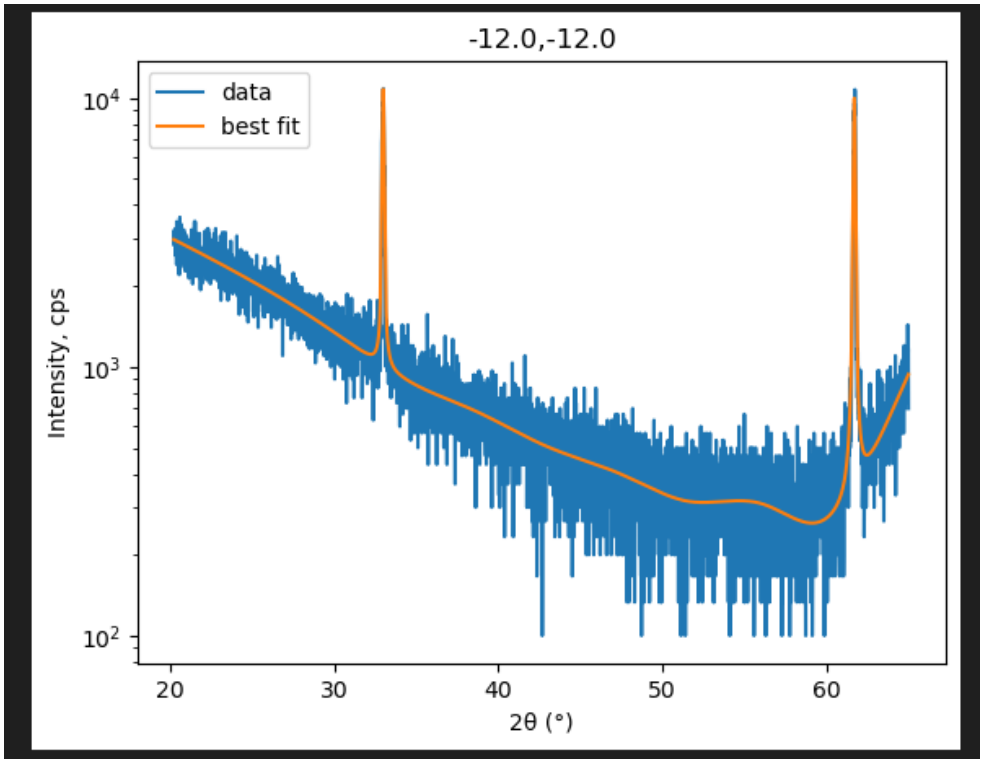
✓ 0.0s

Coordinate	12.0,12.0		-12.0,12.0		12.0,-12.0		-12.0,-12.0	
Data type	2θ (°)	Intensity, cps	2θ (°)	Intensity, cps	2θ (°)	Intensity, cps	2θ (°)	Intensity, cps
0	20.24	2760.358275	20.24	3067.741976	20.24	3108.673794	20.24	3021.863817
1	20.25	2773.743528	20.25	3067.667824	20.25	3102.481725	20.25	3021.811908
2	20.26	2787.128780	20.26	3067.593672	20.26	3096.289657	20.26	3021.759999
3	20.27	2800.514032	20.27	3067.519519	20.27	3090.097588	20.27	3021.708090
4	20.28	2813.899284	20.28	3067.445367	20.28	3083.905519	20.28	3021.656180
...	...	...	...	...	...	...	...	...
4471	64.95	1049.548881	64.95	1024.632239	64.95	937.305591	64.95	1027.081961
4472	64.96	1060.405389	64.96	1033.115737	64.96	941.599248	64.96	1036.010401
4473	64.97	1071.261897	64.97	1041.599236	64.97	945.892905	64.97	1044.938840
4474	64.98	1082.118405	64.98	1050.082734	64.98	950.186562	64.98	1053.867279
4475	64.99	1092.974912	64.99	1058.566232	64.99	954.480219	64.99	1062.795719

The second step is the xrd_fit function, it fits the data based on Peaks output from initial_peaks. Input the data from read_XRD and Peaks from xrd_initial_peaks. xrd_fit constructs a model with pseudo-Voigt peaks and a spline background, number of knots adjusted with knots. Plots can also be toggled here. The data range that is fitted can be limited using dataRangeMin and dataRangeMax. Background can be removed from the fit using the remove_background_fit modifier.

```
xrd_output = xrd_fit(data, Peaks, knots, dataRangeMin, dataRangeMax, withplots =
True, plotscale = 'log', remove_background_fit = False)
```

If withplots is toggled on, a plot will be output for every point as the function runs:



Peak positions:

	Peak 2θ	Peak intensity	FWHM	Lorentzian/Gaussian fraction
0	32.99	9772.467	0.15	0.5
1	61.69	9628.472	0.15	0.6

2 example with knots = 11

Final output is a data frame with measurements, fit, and peak locations, intensity, FWHM, and Lorentzian/Gaussian fraction for every point.

xrd_output

✓ 0.0s

Coordinate	12.0,12.0							-12.0,12.0			...	12.0,-12.0		
Data type	2θ	Measured intensity	Fit intensity	Peak 2θ	Peak intensity	FWHM	Lorentzian/Gaussian fraction	2θ	Measured intensity	Fit intensity	...	Peak intensity	FWHM	Lorentzian/Gaussian fraction
0	20.24	3033.3330	2936.463283	32.99	9535.683	0.15	0.61	20.24	2666.6670	3024.672110	...	9717.262	0.15	0.73
1	20.25	2833.3330	2934.383583	61.71	9690.465	0.14	0.53	20.25	3400.0000	3022.017604	...	9193.868	0.16	0.14
2	20.26	1933.3330	2932.305006	NaN	NaN	NaN	NaN	20.26	3466.6670	3019.366573	...	NaN	NaN	NaN
3	20.27	2833.3330	2930.227553	NaN	NaN	NaN	NaN	20.27	3400.0000	3016.719011	...	NaN	NaN	NaN
4	20.28	2766.6670	2928.151221	NaN	NaN	NaN	NaN	20.28	2733.3330	3014.074913	...	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4471	64.95	866.6667	915.246514	NaN	NaN	NaN	NaN	64.95	933.3333	925.123178	...	NaN	NaN	NaN
4472	64.96	1033.3330	917.920082	NaN	NaN	NaN	NaN	64.96	1200.0000	928.071561	...	NaN	NaN	NaN
4473	64.97	1000.0000	920.600967	NaN	NaN	NaN	NaN	64.97	1033.3330	931.028351	...	NaN	NaN	NaN
4474	64.98	1166.6670	923.289178	NaN	NaN	NaN	NaN	64.98	866.6667	933.993557	...	NaN	NaN	NaN
4475	64.99	1400.0000	925.984723	NaN	NaN	NaN	NaN	64.99	900.0000	936.967187	...	NaN	NaN	NaN

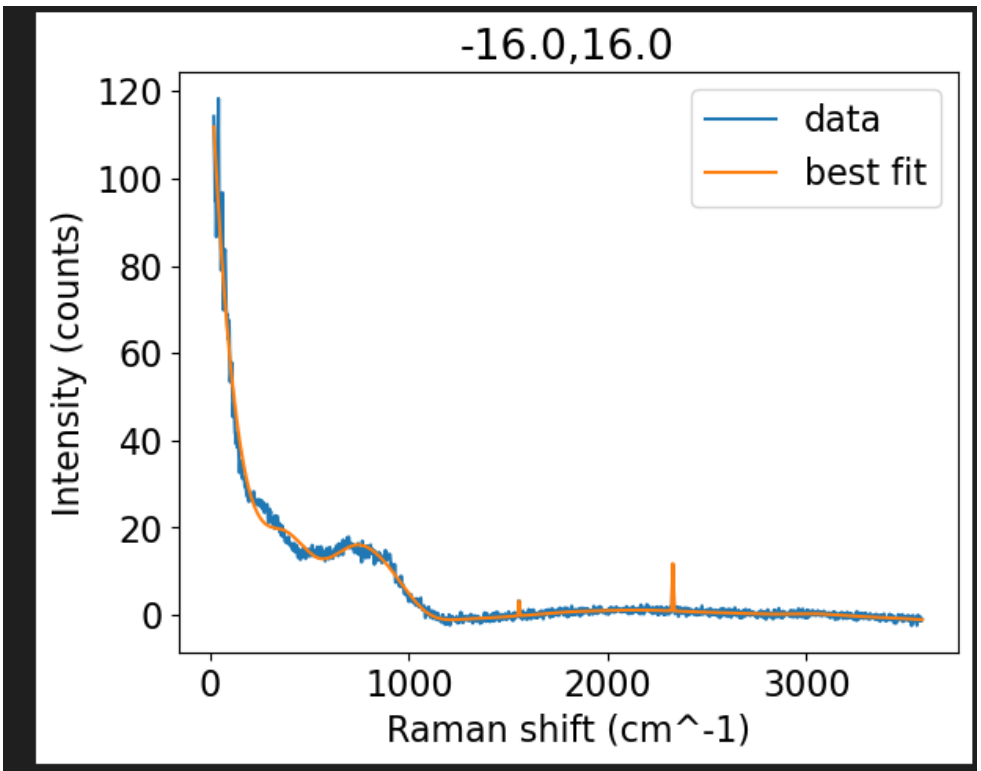
4476 rows × 28 columns

Raman fitting

Raman fitting is done in two stages. The first stage is to find initial peak location and amplitude data. This is done using initial_peaks, as described in the XRD fitting section. The data is fitted using the raman_fit function. This function takes the Peaks output from initial_peaks and the data frame output from read_raman. The background is fitted using a spline function, the number of points used for this is adjusted using knots. The peaks are fitted using a Gaussian function. The data range that is fitted can be limited using dataRangeMin and dataRangeMax. Background can be removed from the fit using the remove_background_fit modifier.

```
raman_output = raman_fit(data,Peaks,dataRangeMin, dataRangeMax, knots, withplots =
True, plotscale = 'log', remove_background_fit = False)
```

If plots are toggled on, one will be output for every point while the function is running. An example of such a plot is shown below:




```
Peak positions:
 Coordinate       -16.0,16.0
Data type  Peak Raman shift Peak intensity  FWHM
0                  1554.38          3.234  6.17
1                  2328.28         11.383  6.45
```

Final output is a data frame with the measurement data, fit, and peak locations, intensity and FWHM for every point.

```
raman_output
✓ 0.0s
```

Coordinate	16.0,-16.0						16.0,-8.0				...
Data type	Raman shift	Measured intensity	Fit intensity	Peak Raman shift	Peak intensity	FWHM	Raman shift	Measured intensity	Fit intensity	Peak Raman shift	...
0	18.8086	116.189000	112.073300	1554.46	2.581	6.86	18.8086	115.048000	111.628296	1554.32	...
1	19.7728	112.084000	111.244571	2328.27	11.863	6.22	19.7728	112.405000	110.803370	2328.15	...
2	20.7371	108.529000	110.420791	NaN	NaN	NaN	20.7371	109.487000	109.983352	NaN	...
3	21.7013	105.568000	109.602115	NaN	NaN	NaN	21.7013	105.632000	109.168398	NaN	...
4	22.6655	103.090000	108.788442	NaN	NaN	NaN	22.6655	102.178000	108.358407	NaN	...
...	...	...	...	...	...	...	...	...	...	...	...
3692	3578.7600	-0.942490	-1.258247	NaN	NaN	NaN	3578.7600	-0.921548	-1.126081	NaN	...
3693	3579.7200	-1.287580	-1.260605	NaN	NaN	NaN	3579.7200	-0.878424	-1.128123	NaN	...
3694	3580.6900	-0.990596	-1.262985	NaN	NaN	NaN	3580.6900	-0.952962	-1.130184	NaN	...
3695	3581.6500	-0.998895	-1.265336	NaN	NaN	NaN	3581.6500	-0.799231	-1.132220	NaN	...
3696	3582.6200	-1.370870	-1.267708	NaN	NaN	NaN	3582.6200	-0.404167	-1.134275	NaN	...

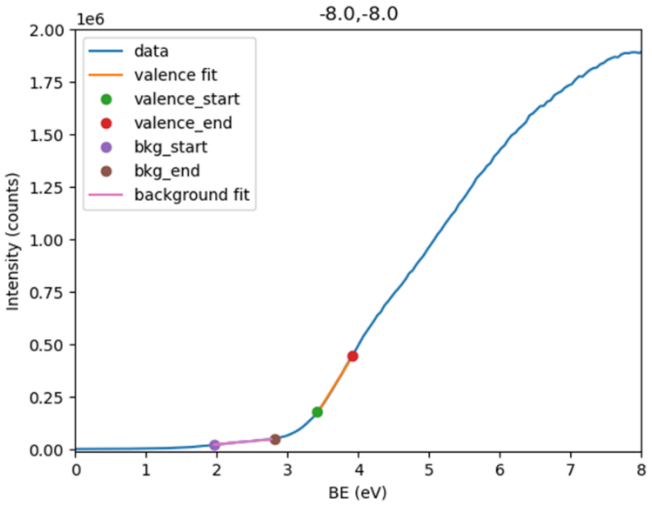
3697 rows × 150 columns

UPS fitting

UPS fitting is done to find the valence band onset. Input data frame from read_UPS. It automatically finds and performs a linear fit on the background as well as the valence band onset. Outputs the intercept between the fits as well as the background intercept with the x-axis, for every point in a data frame. Data analysis starting point can be chosen with startvalue. Plots toggled with withplots, if plots are toggled on it will output a plot for every point as the code is running. The y-axis scale can be chosen with plotscale.

```
UPS_output = UPS_fit(data, startvalue, withplots = True, plotscale = 'linear')
```

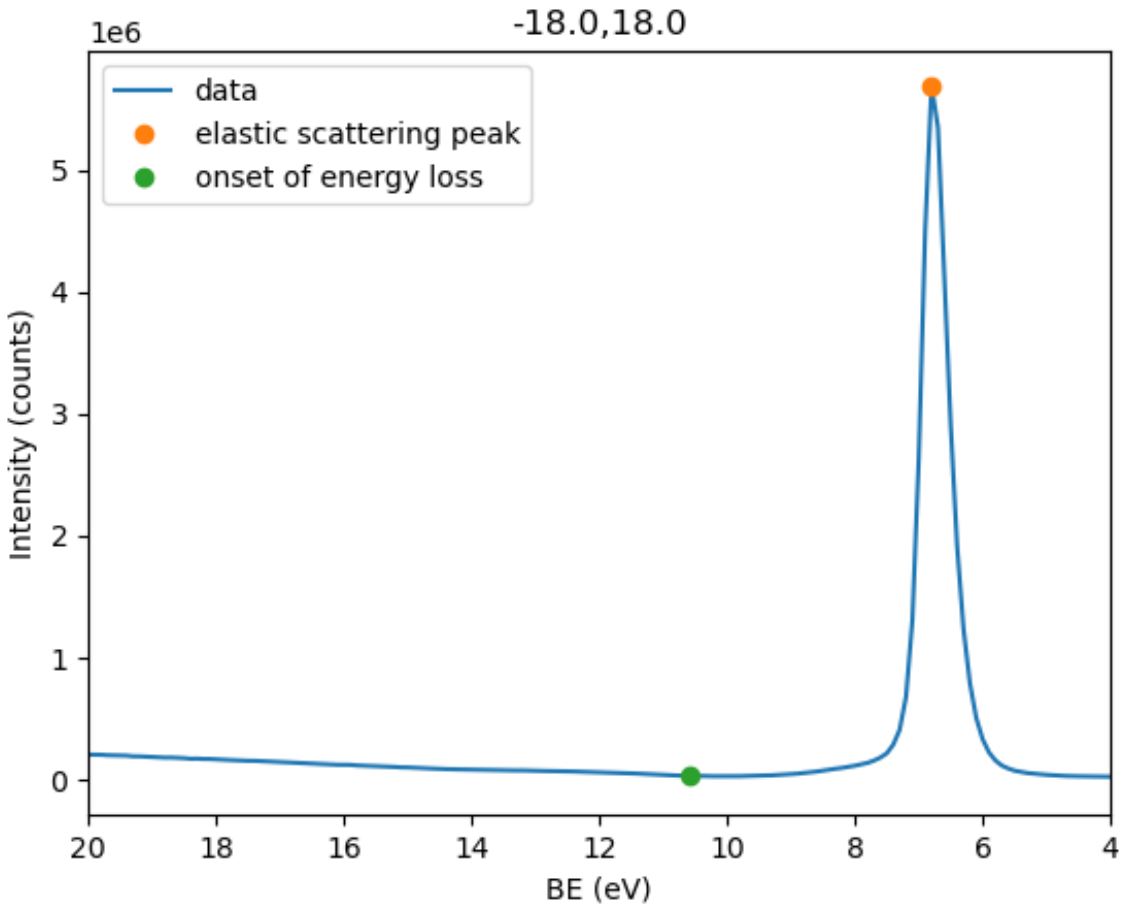
It will automatically identify the start and end point of the background and valence band for fitting, based on the slope.



REELS fitting

REELS fitting is done to find the band gap. Input data frame from read_REELS. It automatically finds the elastic scattering peak and onset of energy loss. Outputs the binding energy (BE) difference between peak and onset, for every point in a data frame. Plots toggled with withplots, scale chosen with plotscale.

```
REELS_output = REELS_fit(data, withplots = True,plotscale = 'linear')
```



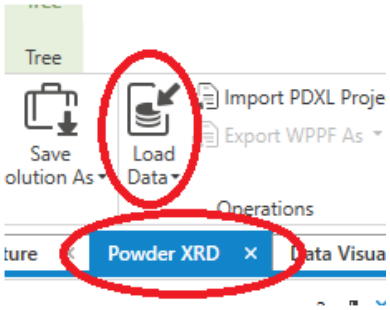
Data Export Guide from Associated Software

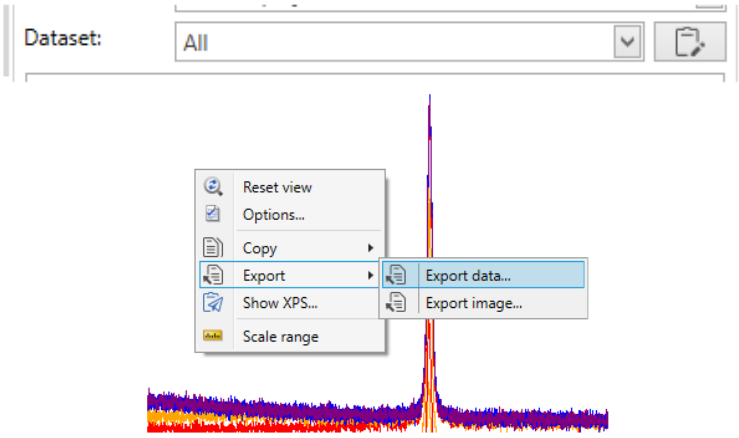
XRD data export

Select the “Powder XRD” tab and load all your data from the “Load Data” button.

Under “Dataset:” select “All”.

Right click the graph and select “Export” -> “Export data…”.

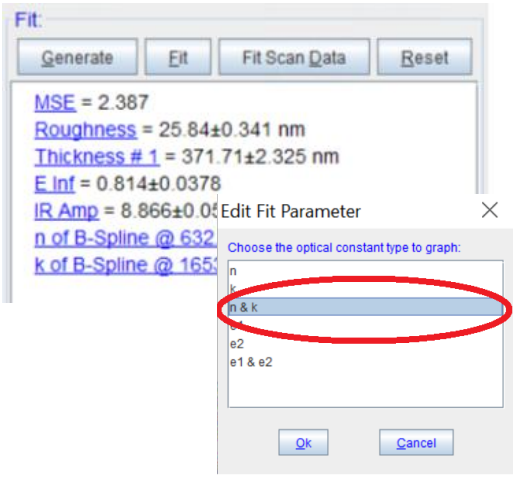


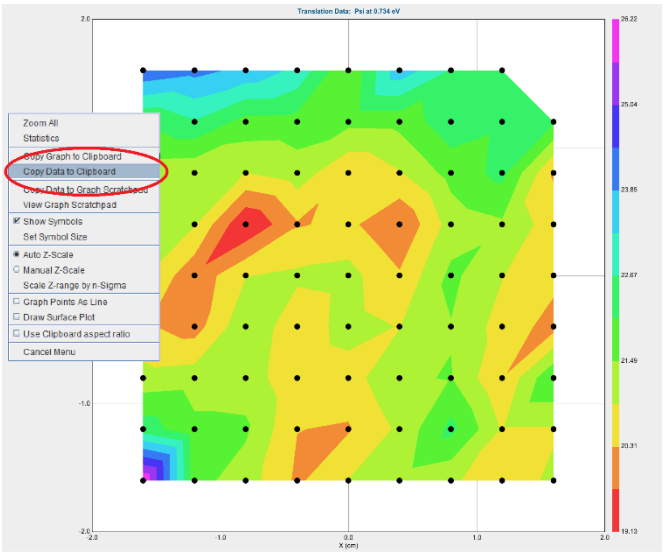


Ellipsometry data export

You must display a graph of the data you want to export. For thickness data, click on thickness in the Fit window. For n and k data, right click the Fit window, select “Graph OCs Vs. Position”, and then select “n & k”.

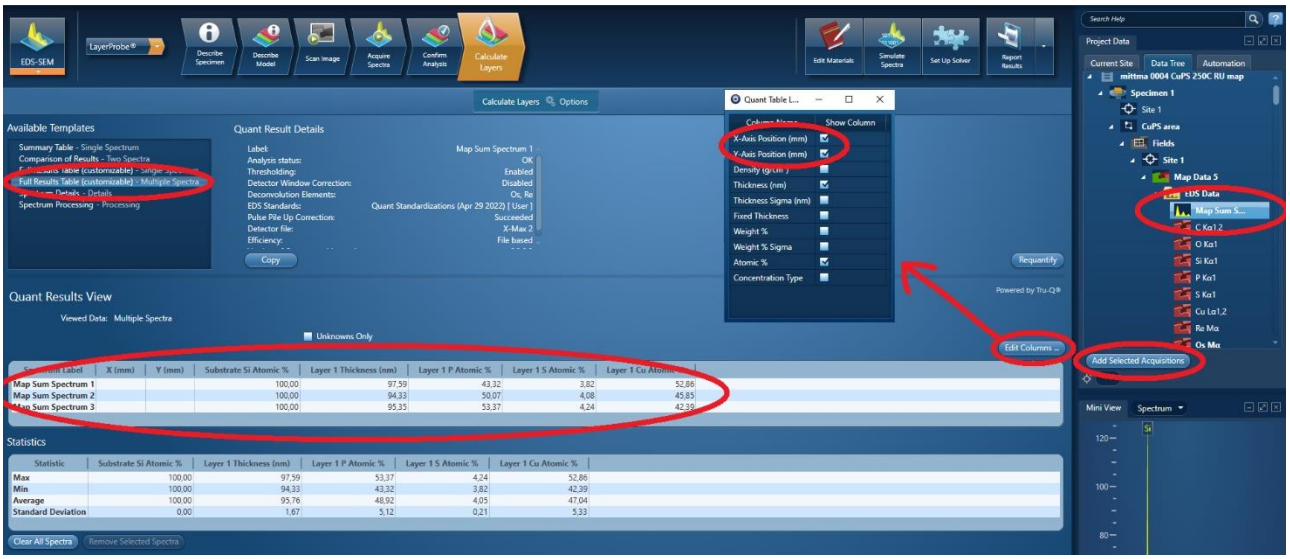
On the displayed graph, right click and select “Copy Data to Clipboard”. You might need to click a bit outside the graph to get the option to show up. Paste the data into an empty notepad file.





SEM LayerProbe data export

On the “Calculate Layers” tab, click “Full Results Table” (Multiple Spectra). On the right panel, now click through each of your measurements and add them via “Add Selected Acquisitions” one by one to the Quant Results View. It can be a good idea to manually rename the measurements to keep the entries sorted. Press the “Edit Columns…” and tick the X- and Y-axis options, alongside any other parameters you would like to export. Finally, click the table of Quant Results, CTRL+A to select all values, then CTRL+C and CTRL+V the table into an Excel sheet.



Even though LayerProbe should already know the coordinates of the measurements, there is unfortunately no way to export them automatically. You therefore as a last step need to manually add the coordinates of your measurements to the Excel sheet.

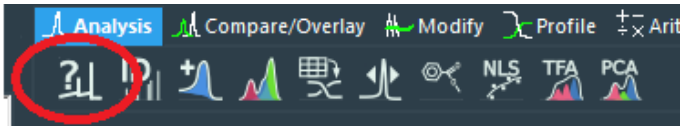
	A	B	C	D	E	F	G	H
1	Spectrum Label	X (mm)	Y (mm)	Substrate Si Atomic %	Layer 1 Thickness (nm)	Layer 1 P Atomic %	Layer 1 S Atomic %	Layer 1 Cu Atomic %
2	Map Sum Spectrum 1			100	97,59	43,32	3,82	52,86
3	Map Sum Spectrum 2			100	94,33	50,07	4,08	45,85
4	Map Sum Spectrum 3			100	95,35	53,37	4,24	42,39
5								

XPS, UPS, REELS, and Raman data export

Avantage unfortunately has poor support for exporting mapping data as CSV. Follow these steps for exporting to CSV for XPS, and an Excel file for UPS, REELS, and Raman.

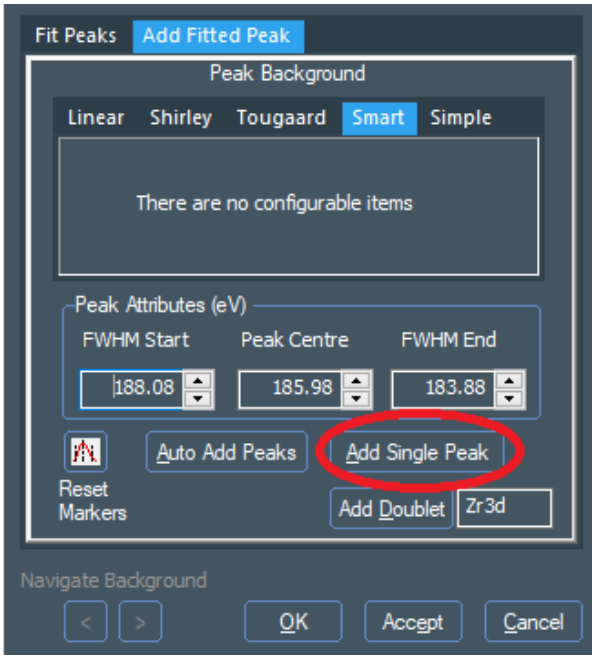
XPS data

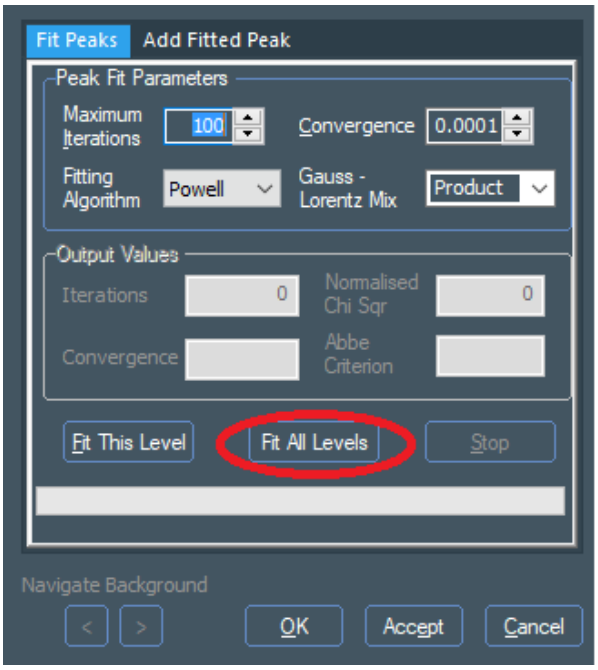
Select your data and do right-click -> Display Modes -> Energy Spectrum. You should have a set of graphs with a Survey measurement and a set with some Core Scans. For the Survey spectrum, select it and automatically ID the peaks:



For the Core Scans, select one and manually iteratively add peaks:

After you have added all peaks for one spectrum, press the Fit All Levels button. Repeat these steps for all your Core Scans.

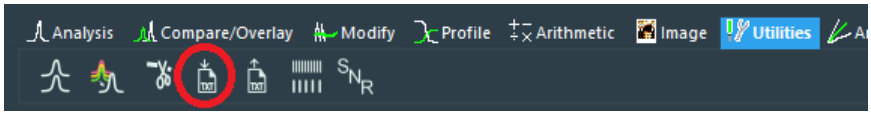


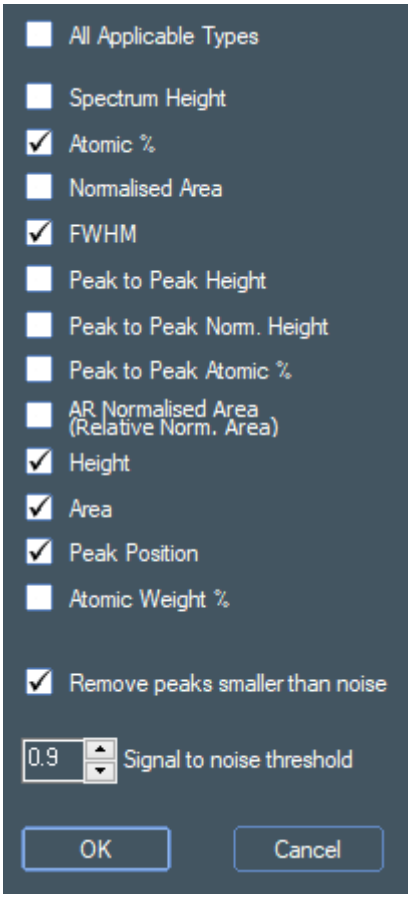


Select your Survey spectra and press the Create Profile button:

You must select these **exact** options. Repeat for the Core Scan spectra.

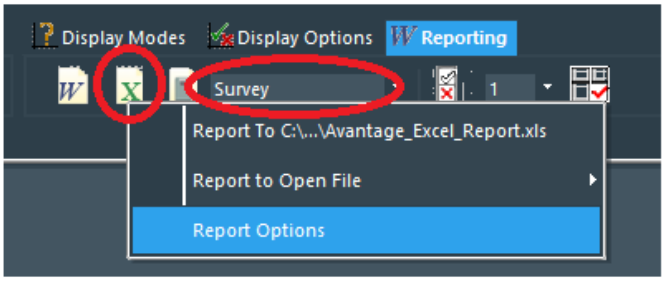
Finally select the newly created map, go to utilities, and export to text file.



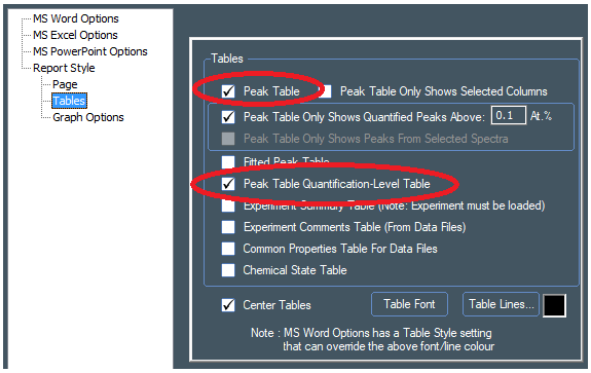


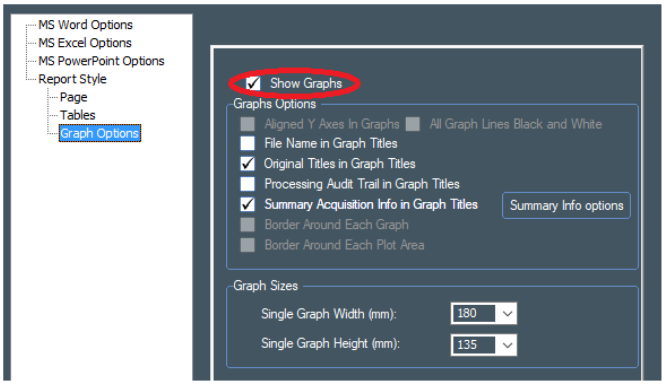
UPS, REELS, and Raman data

The export process is identical for UPS, REELS, and Raman. Go to “Reporting” and change export type to “Survey”. Then click the Excel button and open “Report Options”.

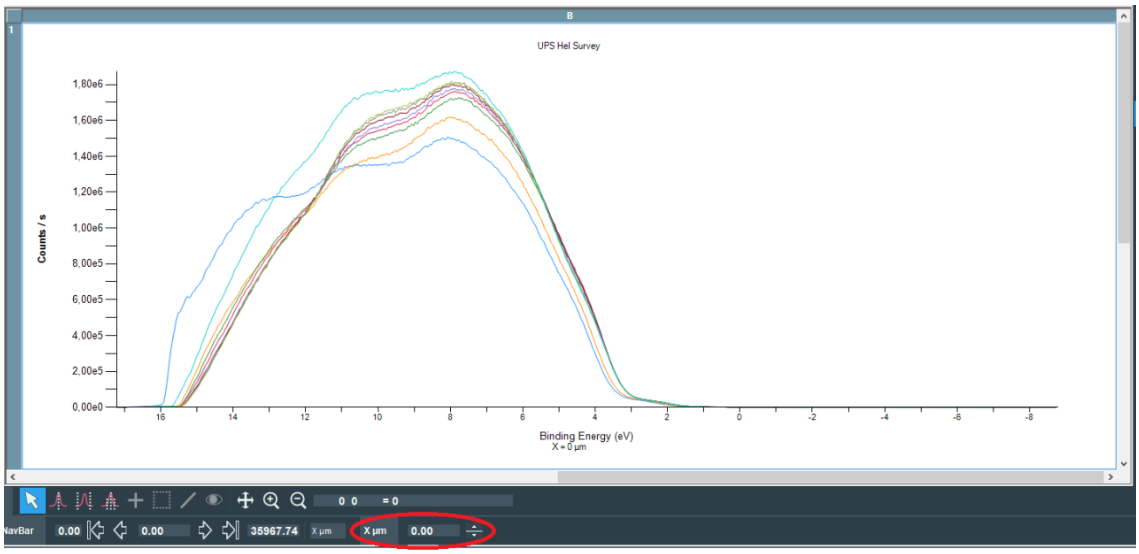


In the options, ensure the following options are ticked:





To speed up the export, right click the shown graph and set display mode to “2D Chart”. If the units make no sense, change to “Energy Spectrum”, then back to “2D Chart”. Your plot should look as follows:



Now create a blank Excel file in the same folder your Avantage file is opened from. Open the Excel file. Now select your graph, click the Excel button under the reporting tab, and “Report to Open File”. This will export a single row of measurements to the Excel file.

In order to export the entire map, you must manually change the selected coordinate row as highlighted above. Each time you change the coordinate, press the “Report to Open File” button again. **Do not touch the Excel file while you are exporting!** Additionally, Avantage takes a

couple seconds to export the data each time, and if you change the coordinates before it is done exporting there is a likelyhood the program will crash. Make sure the program is idle before changing coordinates.