

# A manual for Porchlight

Jakub Konkol  
George Tsilomelekis

December 9, 2022



A light to guide you.

# Contents

<b>1</b>	<b>Running Porchlight</b>	<b>3</b>
<b>2</b>	<b>The main program window</b>	<b>4</b>
2.1	The menu bar . . . . .	4
2.2	Left Panel . . . . .	4
2.3	The Right Panel . . . . .	5
<b>3</b>	<b>Importing Data</b>	<b>6</b>
<b>4</b>	<b>Exporting Data</b>	<b>6</b>
<b>5</b>	<b>File Format Support</b>	<b>6</b>
<b>6</b>	<b>Transformations</b>	<b>7</b>
6.1	Trim . . . . .	7
6.1.1	Trim . . . . .	7
6.1.2	Inverse Trim . . . . .	7
6.2	Baseline . . . . .	7
6.2.1	Polyfit . . . . .	7
6.2.2	AsLS . . . . .	8
6.3	Smooth . . . . .	8
6.3.1	Rolling . . . . .	8
6.3.2	Savitzky-Golay . . . . .	8
6.4	Normalization . . . . .	9
6.4.1	SNV . . . . .	9
6.4.2	Area . . . . .	9
6.4.3	Vector . . . . .	9
6.4.4	MinMax . . . . .	10
6.4.5	MSC . . . . .	10
6.4.6	Peak Normalization . . . . .	10
6.5	Centering . . . . .	10
6.5.1	LastPoint . . . . .	10
6.5.2	Mean . . . . .	10
6.6	Derivative . . . . .	11
6.6.1	Savitzky-Golay Derivative . . . . .	11
6.7	Dataset . . . . .	11
6.7.1	Reset . . . . .	11
6.7.2	Subtract . . . . .	11
<b>7</b>	<b>Known Issues</b>	<b>11</b>
<b>8</b>	<b>Some resources for data</b>	<b>12</b>

# 1 Running Porchlight

To install Porchlight, you can install from the PyPi package index using the command:

```
1 pip install Porchlight
```

Alternatively, you can install the most recent version of Porchlight directly from the GitHub repository using

```
1 pip install git+https://github.com/Jakub-Konkol/Porchlight.git
```

To run the Porchlight GUI, run the following command in Anaconda Prompt:

```
1 python -m porchlight
```

## 2 The main program window

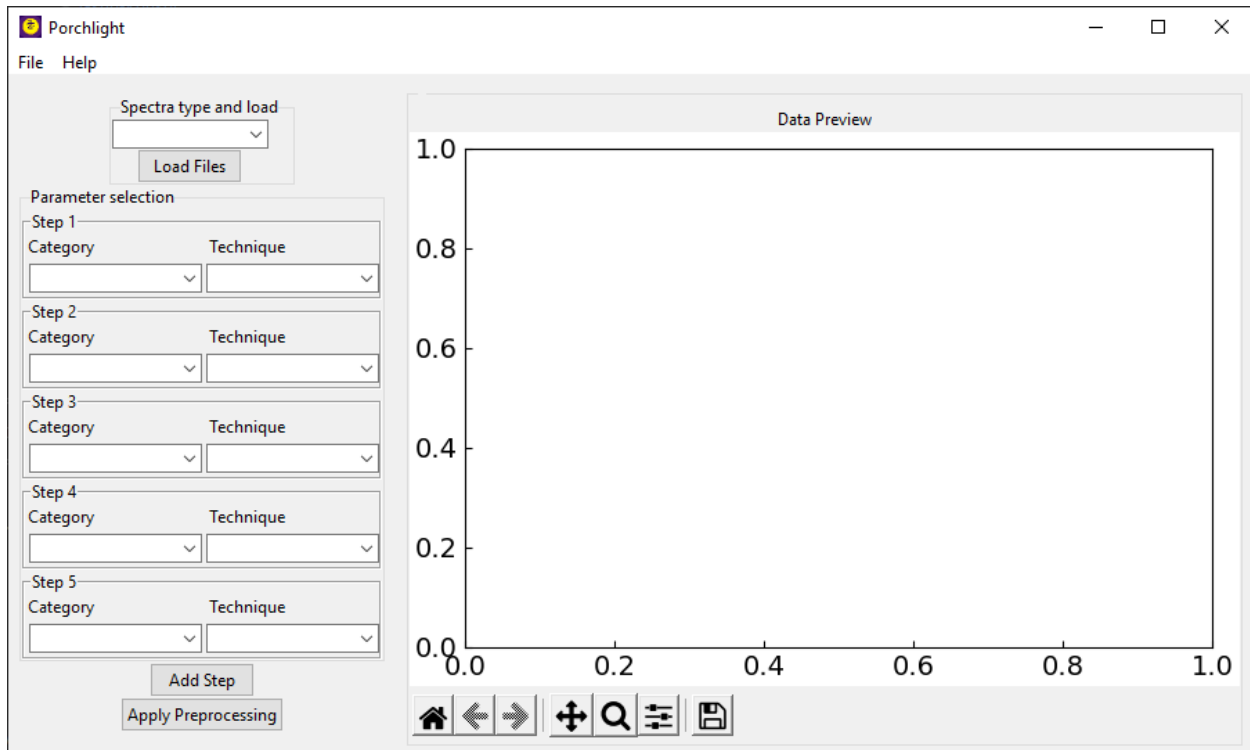
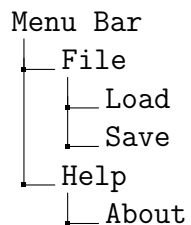


Figure 1: The main Porchlight window.

The main program window for Porchlight is where the user spends most of their time. It is currently broken up into three main parts: the menu bar, the left panel, and the right panel.

### 2.1 The menu bar

The menu bar contains the standard expected options.



### 2.2 Left Panel

Going from top to bottom:

**Type of spectrum** is a drop down that allows the user to choose the type of data they will load in. Current choices are:

- IR - Transmission
- IR - Absorbance
- Raman
- UV-Vis

The type of spectrum will change the plotting style of the chart on the right, as well as some common motifs seen in that type of spectroscopy, such as the reversed axis in IR spectroscopy.

**Load Data** lets the user choose the files they would like to load. Supported filetypes include:

- CSV
- Text
- Excel
- SPC

Loading Thermo-Galactic .spc files uses `spc_spectra` to read the data, so users can install that package using `pip install spc_spectra`.

Porchlight can load multiple files at a time, and those files can contain one spectra or multiple. The main thing to understand is that Porchlight makes the assumption that the longest axis in the table is the abscissa axis. For example, if a file contains a dataset that is  $3250 \times 24$ , Porchlight will read it as 23 spectra, with the first column containing the energy axis (such as wavenumbers for IR or Raman) and vice versa.

If the energy values don't match up perfectly, Porchlight has some logic that will try to identify and match energies. For example, say there is a section of File A where the energy points go 406.4311, 406.9132, and 407.3953; while File B has the same section as 406.4311, 406.9133, and 407.3953. Porchlight will identify a mismatch in 406.9132 and 406.9133 and combined those into a column at 406.91325.


**Parameter Selection** allows the user to choose a category of transformation to apply, then a specific technique within that category, and then some options. The specific methods and options are detailed in Section 6.

## 2.3 The Right Panel

This section contains the plot of the data. There the data can be interacted by using the widgets at the bottom of the plot.


**Home**  resets the plot to its original position.

**Forward and Back**  and  will redo and undo your changes, respectively.

**Pan/Zoom**  will let you pan and zoom the plot. Use left click to pan the plot. Hold “x” to lock the movement to only the x-axis, and “y” to lock the movement to only the y-axis. Use right click to zoom. You can manipulate the x-axis by pressing “x”, moving to the right to zoom in and to the left to zoom out. Manipulate the y-axis by holding “y” and then move the cursor up to zoom up or down to zoom out. Holding “CTRL” zooms while maintaining the aspect ratio.

**Zoom**  lets you draw a rectangle to zoom to on the plot.

**Subplot Configuration**  lets you manipulate the visible settings of the plot, useful is a label is extending past the visible region but you want to save it.

**Save**  lets you save plot figure to an image file. Supported formats include:

- png
- ps
- eps
- svg
- pdf

## 3 Importing Data

1. Select the type of spectra from the dropdown.
2. Click on the “Load files” button and choose your files. CTRL+left click to select multiple individual files at once, Shift+left click to choose a sequence of files.
3. Press the “Open” button to load the files into Porchlight.

## 4 Exporting Data

To save the data, use the Menu bar by going to File → Save. Porchlight will save to CSV and Excel files.

Please note that the default file format is CSV. Due to a Tkinter bug, if you want to save in a different format such as Excel .xlsx, you must type in the file extension when saving. Meaning, to save the file as a an Excel file name “spectra”, you must type “spectra.xlsx” as the file name.

## 5 File Format Support

Porchlight will natively support loading text files (.txt), comma seperated values files (.csv), and Microsoft Excel files (.xlsx). Thermo-Galactic spectral files (.spc) can be read if the user has the package `spc_spectra` installed.

In txt, csv, and xlsx files, Porchlight assumes that the longest dimension of the data is the spectral waveform. To this end, the first column or row will be taken as the spectral axis, and the subsequent data as the spectral intensity values. Files can contain single or multiple spectra as long as a wavelength axis supplied.

## 6 Transformations

This section describes the types of transformations and the options needed. A code snippet will show how to call this function from a script, and then the method is described.

### 6.1 Trim

Trimming the spectra is recommended unless you are certain the spectra loaded are all the same range. While many methods are one-way methods that are applied to each spectra individually, some methods such as MSC assume the spectra are all the same length.

When performing trimming, we are performing a variable selection. We remove sections of the spectra if we know they do not contain meaningful information, or if they contain regions of high noise that may hinder interpretation.

#### 6.1.1 Trim

```
1 spectra.trim(start , end)
```

Trimming cuts down the spectra to a specified range. “Start ”is the lower bound, “End ”is the upper bound. The spectra removes anything smaller than “Start” or greater than “End”.

#### 6.1.2 Inverse Trim

```
1 spectra.invtrim(start , end)
```

The “inverse trim” cuts out data between two points. “Start” is the lower bound, “End” is the upper bound. Useful for cutting out high noise regions in the middle of spectrum, such as the 2300-1900  $\text{cm}^{-1}$  region of FTIR data collected on a diamond ATR.

### 6.2 Baseline

Baseline correction functions attempt to correct for changes that occur in the baseline. Two examples include a drift in the spectrometer calibration or fluorescence in a Raman spectrum.

#### 6.2.1 Polyfit

```
1 spectra.polyfit(order , niter)
```

Polyfit takes one mandatory parameter, the order of the polynomial, and one optional parameter, the number of iterations to perform.

The polyfit algorithm works by fitting the data to a polynomial, and then replacing values in the fit by any spectral intensities that are less than the fit. Then, it performs the polynomial fitting again for a set number of iterations or until no changes occur. The idea here is that by always replacing the fitting data with lower values, the fit continues to pull down the baseline until it follows the baseline.[\[1\]](#)

### 6.2.2 AsLS

```
1 spectra.AsLS(penalty, asymmetry, niter)
```

AsLS is an asymmetric least squares method that uses multiple iterations to attempt to fit a line beneath the spectrum. It takes two parameters, a Penalty term and an Asymmetry term. The implementation is based on the one provided Eilers and Boelens.[\[2\]](#) They recommend the penalty term to be set between  $10^2 - 10^9$ , while the symmetry term between  $0.001 - 0.1$ . When working in Porchlight, you can use e notation. For example, `10e6` represents  $10^6$ .

The method can be difficult to use because the “optimal” parameters are fairly subjective. Here are some hints based on my experience. The penalty factor controls the the baseline’s pliability. If the penalty factor is small, then the baseline is soft and follows the curves of the spectrum more easily. If the penalty factor is large, then the baseline is stiff and will follow broader trends in the data. Be warned that if the baseline is too soft it will start fitting into broad peaks. Too large, and you may not be able to follow the baseline. The authors’ describe the asymmetry parameter as being related to noise. The noisier the data, the larger asymmetry should be. It’s challenging to describe it’s behavior, I recommend trying 0.001, 0.01, and 0.1, seeing which gives you a better spectrum, and work from there. If you have a spectrum with sharp peaks and little noise try a small asymmetry value first.

## 6.3 Smooth

Smoothing algorithms attempted to tackle noise within a spectrum, decreasing high frequency noises to reveal underlying signal peaks. Smoothing can be extremely helpful in interpreting noisy spectra, but it may make quantification challenging as peak shapes can change and peak heights will decrease.

### 6.3.1 Rolling

```
1 spectra.rolling(window)
```

Rolling is a rolling window smoothing, or “boxcar” averaging. The parameter “Window ” is an odd integer for the number of datapoints used in the averaging. The larger the value, the more smoothing you will have but the greater the deformation of the peaks.

### 6.3.2 Savitzky-Golay

```
1 spectra.SGSsmooth(window, poly)
```



This technique applies a Savitzky-Golay smoothing to the data.[\[3\]](#) The Savitzky-Golay smooth is a moving window that fits a polynomial to a subsection of data points, and then sets the center of that window to the fitted polynomial value. “Order” is the order of the polynomial and “Window” is the number of data points. Order must be an integer greater than 0 that is less than Window. Window must be a positive, odd integer.

Generally, the following trends are true for smoothing data: a higher order polynomial will result in less smoothing of the data while a larger window will result in more smoothing. As one can imagine, a higher order function will be able to fit the data better, which is why it appears to have less of an effect on smoothing. However, too much smoothing will reshape your data to the point it may no longer be representative of the underlying spectrum. Peaks will broaden and shrink. Therefore, smooth sparingly.

## 6.4 Normalization

### 6.4.1 SNV

```
1 spectra.snv()
```

This technique applies the Standard Normal Variate (SNV) transformation.[\[4\]](#) It takes no options.

$$sn_i = \frac{s_i - \bar{sn}}{\text{std}(sn)} \quad (1)$$

$sn_i$  is the transformed spectrum,  $s_i$  is a datapoint in that spectrum,  $\bar{sn}$  is the spectrum mean,  $\text{std}(sn)$  is the standard deviation of the spectrum.

### 6.4.2 Area

```
1 spectra.area()
```

This technique normalizes the spectrum by the integrated area under the curve. It takes no options.

$$sn_i = \frac{s_i}{\int s \, ds} \quad (2)$$

### 6.4.3 Vector

```
1 spectra.vector()
```

This technique applies a vector normalization.[\[5\]](#) It takes no options.

$$sn_i = \frac{s_i}{\sqrt{\sum s_i^2}} \quad (3)$$

$sn_i$  is the transformed spectrum,  $s_i$  is a datapoint in that spectrum. In literature, vector normalization is often applied after derivatives.

#### 6.4.4 MinMax

```
1 spectra.minmax(min_val, max_val)
```

This technique applies a min-max transformation that normalizes the data between 0 and 1.[5] The largest value in the spectrum becomes 1 and the lowest value in the spectrum becomes 0. You may provide a Maximum and Minimum value if your specific application requires different values.

$$sn_i = \frac{s_i - s_{min}}{s_{max} - s_{min}} \quad (4)$$

#### 6.4.5 MSC

```
1 spectra.msc(spectrum)
```

Applies a Multiplicative Scattering Correction (MSC).[6] The algorithm requires a reference spectrum that is considered the “ideal” spectrum of that dataset. One can be provided by setting option A to a positive integer of the array value in the series, or if set to 0 the reference spectrum is the mean of all spectra.

The algorithm then fits each spectrum to the reference spectrum to a line  $y = mx + b$ . Each datapoint is then manipulated by the equation

$$sn_i = \frac{s_i - b}{m} \quad (5)$$

If a reference spectrum is chosen, it is ignored by any further transformations.

#### 6.4.6 Peak Normalization

```
1 spectra.peaknorm(wavenumber)
```

Normalized the spectral intensity to be 1 at a given wavelength.[5] “Peak Position” is the position of the peak.

### 6.5 Centering

#### 6.5.1 LastPoint

```
1 spectra.lastpoint()
```

Subtracts each datapoint in the spectrum by the last point in the array.

#### 6.5.2 Mean

```
1 spectra.mean_center(option)
```

Mean centering can refer to subtracting by the mean of the spectrum or by the mean spectrum of all the spectrum.

**Self = 0** then Porchlight calculates the mean value of the spectrum and then subtracts each point in the spectrum by that value.

**Self = 1** then Porchlight calculates the mean spectrum of all the spectrum loaded into Porchlight then subtracts each spectra by the mean spectrum.

## 6.6 Derivative

### 6.6.1 Savitzky-Golay Derivative

```
1 spectra.SGDeriv(window, poly, order)
```

This performs a Savitzky-Golay derivative on the data.[\[3\]](#) This method is popular as it simultaneously smooths as it calculates the derivative spectrum. “Window” is the number of datapoints to fit, “Polynomial” is the order of polynomial to be fit, and “Deriv. order” is the derivative order. You must make sure that the order of the derivative is less than the polynomial you are fitting. As an example, if you are fitting a 3rd order polynomial, you can only calculate up to the third derivative of your spectra. Similar caveats to the window size and fitting polynomial apply here as the smoothing. For the derivative order, note that the higher the derivative the lower the signal to noise ratio. Usually, first or second derivatives are enough.

## 6.7 Dataset

This section will contain miscellaneous functions are not transformations as much as data handling.

### 6.7.1 Reset

```
1 spectra.reset()
```

This will reset the data to its original form.

### 6.7.2 Subtract

```
1 spectra.subtract(spectrum)
```

This will subtract the loaded spectra by the spectrum given in “Spectrum”. As an example, useful for removing the contribution of a solvent, or the initial spectrum of a time series.

## 7 Known Issues

The loading of files is slow, you may need to wait a few seconds for the data to display.

Certain operations such as baseline fitting are very computationally intensive, if performing baseline fitting be prepared for the program to lock up.

Tkinter on Windows has a bug where you need to explicitly type out the filetype of the saved file, otherwise it defaults to CSV. For example, to save as an Excel file named "data", you need to explicitly type "data.xlsx" into the filename when saving.

## 8 Some resources for data

The article from Acquarelli et. al. lists a variety of spectroscopic datasets that may prove useful, some of which are listed below.<sup>[7]</sup>

The Council for Near-Infrared Spectroscopy has NIR spectra for wheat alongside protein content, as part of the International Diffuse Reflectance Conference 2016 Chemometrics ShootOut. The data can be found on their website <https://www.cnirs.org> and then navigating to Education → Documents in the top bar, and then IDRC 2016 → Shootout 2016. I was also able to find the shootout data for 2014, 2016, and 2018 by performing internet searches with the terms "IDRC XXXX Shootout," substituting for the year.

Eigenvector Research has some datasets available on their website as well. <https://eigenvector.com/resources/data-sets/> contains NIR and IR spectra sets, but will need massaging from the MATLAB .mat format of the files. They should be able to be opened using `scipy.io`'s `loadmat()` function.

The University of Copenhagen's Department of Food Science has a variety of datasets available. <http://www.models.life.ku.dk/datasets> The spectral types include Raman, IR, NMR. Other data such as fluorescence EEMs is present, but not currently applicable in Porchlight. Most data is provided in .mat format, which would need converting to an ASCII file such as CSV.

The Quadram Institute also provides some datasets, primarily in IR. <https://csr.quadram.ac.uk/example-datasets-for-download/> The main benefit here is that all seem to use CSV as the data storage medium, which should not require any data file conversions.

## References

- [1] C. A. Lieber and A. Mahadevan-Jansen, "Automated method for subtraction of fluorescence from biological raman spectra," *Applied Spectroscopy*, vol. 57, pp. 1363–1367, nov 2003.
- [2] P. H. C. Eilers and H. F. M. Boelens, "Baseline correction with asymmetric least squares smoothing," techreport, Leiden University, Leiden, The Netherlands, Oct. 2005.
- [3] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures.," *Analytical Chemistry*, vol. 36, pp. 1627–1639, July 1964.
- [4] R. J. Barnes, M. S. Dhanoa, and S. J. Lister, "Standard normal variate transformation and de-trending of near-infrared diffuse reflectance spectra," *Appl. Spectrosc.*, vol. 43, pp. 772–777, May 1989.

- [5] R. Gautam, S. Vanga, F. Ariese, and S. Umapathy, “Review of multidimensional data processing approaches for raman and infrared spectroscopy,” *EPJ Techniques and Instrumentation*, vol. 2, June 2015.
- [6] P. Geladi, D. MacDougall, and H. Martens, “Linearization and scatter-correction for near-infrared reflectance spectra of meat,” *Applied Spectroscopy*, vol. 39, pp. 491–500, may 1985.
- [7] J. Acquarelli, T. van Laarhoven, J. Gerretzen, T. N. Tran, L. M. Buydens, and E. Marchiori, “Convolutional neural networks for vibrational spectroscopic data analysis,” *Analytica Chimica Acta*, vol. 954, pp. 22–31, feb 2017.