QTDISPLAYSPEC.PY DOCUMENTATION

MICHAEL TOPPING

1. Classes

spectrumClass.

init.

- Paremeters:
  - `loc` - directory that the spectrum is in.
  - `filename` - filename of the spectrum file.
  - `emRedshift` - guess for the redshift based on emission features.
  - `absRedshift` - guess for the redshift based on absorption features.
  - `aperture` - the aperture in the fits file that the object spectrum is in.
- Description: This procedure will open the data file and set member variables spec and wavelengths to the appropriate values. If there are more than one aperture in the file, the aperture that contains the object needs to be specified. When creating the wavelengths array, the wavelengths solution is pulled from the fits header. If no information on the wavelengths solution is available, a default is used that may be incorrect for the current file. We are only looking at Lyα as $z = 3$ so we cut down the range of the spectrum. Then we call a function to create a smoothed version of the spectrum.

getSmoothedSpectrum.

- Paremeters:
  - `smoothFac` - number of points that will be used to calculate median for smoothing, must be an odd number.

- Description: First the current spectrum is smoothed by taking the median of each `smoothFac` number of points. Then it is fit to a spline and interpolated to match the number of data points of the original spectrum.

removeSpec.

- Parameters:
  - `center` - center wavelength of the line that you want removed.
- Description: Used to take out sky lines that will confuse the code. Input the wavelength of the line and it will be replaced by the "continuum" of the spectrum.

findRms.

- Paremeters:
  - `rmsThresh` - the amount above the continuum that a datapoint must be for the code to declare it an emission feature.
- Description: This is what finds the lines. If the noise is not constant through the entire spectrum this will split it up and check different parts of the spectrum separately. First the rms value of the spectrum is calculated, then anything that is `rmsThresh` times that will be picked out as a possible line and added to the peaks member variable.

removeDupPeaks.

- Description: The `findRms` function picks out every data point that is above the threshold and adds it to the list of peaks. In some cases more than one data point in a single spectral line will be added to the list. If this happens, the central one is kept in the list of peaks, and the others are removed. Otherwise, the same emission line could be fit more than once.

fitline.

- Paremeters:
  - `peak` - the index of the peak that you want to fit to a gaussian.

  - `plotBool` - boolean, whether you want the procedure to create and save plots of the fits.

- Description: This is the function that fits the emission lines to a gaussian. Given an initial guess for the center of the emission line, the points ± 10 from the peak (roughly $15\mathring{A}$) are picked out and set to the `subSpectra` and `subWavelengths` variables. Then we make a check to see if the line is right on the edge of the spectrum, which would cause some problems. Then the line is fit to a gaussian function using scipy curve_fit function.
  If the `plotBool` variable is true, a plot of the full spectrum and a zoom in of the line with its fit is saved as a .png.

datasetClass.

init.

- Parameters
  - `folder` - the directory name for the list of spectrum
  - `listFile` - the name of the file that list the spectrum
- Optional Parameters
  - `usecols=()` - the columns that will be read in from the spectrum list file.
- Description: A dataset object requires a file that contains a list of spectra filenames. It will read in this file and keep track of all the filenames in the dataset. The optional keyword argument is used if there are multiple columns in the file, e.g. redshift data, or notes. This object also contains one `spectrum` object at a time, and is initialized to the first spectrum in the list.

setSpectrum.

- Parameters
  - `index` - the index of the current spectrum
- Description: Changes the member `spectrum` object to a different one. It will use the `index`th one in the list of spectra.

2. Application (QtDisplaySpec.py)

PlotCanvas. This class extends the `FigureCanvas` class. It doesn't really do much, just sets some parameters of the plot.

DynamicPlotCanvas. Extends `PlotCanvas`. This is where the spectrum is plotted. The input is a `spectrum` class object.

update_figure. This is called every frame. It clears the current axes, and plots the spectrum in the plotting window. If emission lines have been found, it will plot their locations as well.

ApplicationWindow. Extends `QMainWindow`. This is what controls the application.

init. This is where the list of `dataset` object is created. Once created, the current spectrum is set to the first spectrum in the first dataset. We also want the application to do something so we will create the button objects as member variables. Then we add the buttons and the plotting window to the layout of the application. Finally, we connect all the buttons to check if they are clicked. If they are clicked then a procedure is called.

fileQuit. Quits the application

closeEvent. Calls the `fileQuit` procedure which closes the application.

findLines. Calls the `spectrum` class `findRms` function to detect all of the lines, then removes duplicates.

findZ. This finds the redshift of the current spectrum. If `findLines` has not been called yet, so lines have not been found, then that procedure is called. Then it looks for the strongest line, fits it to a gaussian, and assumes that it is $Ly\alpha$, and can calculate the redshift from that.

findAllZ. This will run through all of the spectra in all of the datasets and create a list of all of their redshifts. It runs through the same procedure that `findZ` goes through to detect emission lines and calculate their redshift from that. If the emission lines are too difficult for the code to pick out of the spectrum and a guess redshift is available, the code then will take the line corresponding to $Ly\alpha$ at that redshift, and fit a gaussian to it, then returns the center of that line as the new redshift. If the code cannot find any lines, and no emission guess redshift is provided, but there is an absorption guess redshift, it will cross correlate the spectrum with a list of metal absorption lines to find the most likely redshift. Then a histogram is created from the list of redshifts.

cycleDataSet. This will cycle through all of the datasets that are in the list of datasets.

prevSpec. Moves to the previous spectrum in the list.

nextSpec. Moves to the next spectrum in the list.

loadData. Opens up a file browser to select more data to analyze. It is currently not finished.