Requirements

1. The code can be run by placing it in your current directory, but it seems a more sensible approach is to put it in your python path. This will look something like one of the following lines in your .cshrc file:

```
#setenv PYTHONPATH {$PYTHONPATH}:/Users/alhenry/py
setenv PYTHONPATH /Users/alhenry/py
```

   My setup does not have PYTHONPATH prior to my own definition, so I'm using the latter. You can `echo $PYTHONPATH` to check yours.

2. put the refit_wisplines code in that python path. also put mpfit.py there, which I will distribute. My version has a small edit, replacing a deprecated numpy function with its update.

3. My setup is astropy with python 2.7 and stsci's ureka. The code is **NOT** python 3 compatible.

To Run on all the objects.

1. cd to the directory where you want the outputs to live. The current setup does not include the Par number in the figures, so you should either make a directory for the field you are working on, or for example, as I am doing, work in the `ParXX_final` directory.

2. the line lists are partially written in SExtractor format, but not correctly. Hence, the built-in astropy.io.ascii sextractor reader will not work right away. Therefore, you must edit the line list header. If it doesn't look like the following, the code will likely break:

```
#1  ParID
#2  NUMBER
#3  NumberofUserIdentifications
#4  NumberofLines>2sigma
#5  RA
#6  Dec
#7  Jmagnitude [99.0 denotes no detection]
#8  Hmagnitude [99.0 denotes no detection]
#9  A_IMAGE
#10 B_IMAGE
#11 redshift
#12 redshift_err
#13 RedshiftFlag [1: Fails to Match That of Other Lines to Within 2.0%, 2: Reviewers Failed To Match Wi
#14 RestFrameFWHM, [Angs]
#15 RestFrameFWHMerror [Angs]
#16 FWHMFlag [1: Deviation by more than factor of 1.5]
#17 Rest-frameEW [Angs]
#18 Lineflux [ergs/s/cm^2]
#19 Linefluxerror [ergs/s/cm^2]
#20 SNR
#21 FluxFlag [1: Error Deviation by more than factor of 1.5; 2: Flux Deviation by more than factor of 1
#22 ContaminationFlag [1=no, 2=yes]
#23 EdgeFlag [1: Overlaps Upper Limit,  1.675 (um), 2: Overlaps Lower Limit,  0.850 (um), 3: Overlaps N
#24 Wavelength
#25 LineID
#26 QualityFlag [+1: Potential Flux Issues, +2: Only multi-line for close, S/N <  5.0, lines (i.e. Ha/S
```

   This header is included in the code distribution.

3. Several adjustable fit parameters are included in a configuration file. I have called it `refit.config`, but it can be renamed however you like. The defaults should be a good place to start, as they seem to work for most objects (85-90%). The details of this file will be described below, as changes will be required to get the last 10-15% of objects.

4. the code is run from ipython; type `ipython` in your terminal. Make sure you've got the ureka python before you type ipython.

5. in ipython, type the following commands

```
import refit_wisplines as wisp
wisp.loop_field('Spectra/',96,'Par096_headerchange_v4.3.dat','Par96_refit','refit.config')
```

   The inputs here are:

   - the path to the .dat files containing the spectra
   - the number of the parallel field
   - the linelist with the modified header
   - the name for the new output catalog (`Par96_refit`).
   - the configuration file

   These can be located anywhere, provided you put the paths into the code call.

6. Outputs:

   - `Par96_refit` – the measurement catalog
   - `Par96_fitfigs.pdf`– Figures showing the fits
   - `Par96_fwhm_z.dat`– fwhm and z guesses used in the fits (this information is elsewhere, but having an abbreviated catalog will be helpful in the next step, I think).
   - Finally, the code makes a directory called `fitfigs`, which contains individual pdf figures for each object. I'm using ghostscript to combine them into the `Par96_fitfigs.pdf` file.

7. **NOTE:** On my computer, there is a strong tendency for ipython to seg fault after executing the last line of code. It doesn't seem to impact results, but it is annoying. I have added a check to make sure it doesn't stop part way– on the last line of code it prints "finished". This behavior started when I upgraded my OS to 10.11. It is beyond me.

Fixing the objects that failed

1. look at the pdf file with the plot figures, and make a list of the ones that you think are bad and should be redone.
2. definitely make a copy of the catalog, and maybe also the fitfigs.pdf file, because we're going to overwrite them. Call them .prelim or .orig or .v0 or whatever suits you.
3. if ipython crashed at the end of the last step, start it again, and `import refit_wisplines as wisp`.
4. edit the configuration file to switch `showfig True`.
5. open up the `ParXX_fwhm_z.dat` because you're probably going to want to reference it.
6. read the configuration file into memory, type `config = wisp.read_config('refit.config')`.
7. if you forgot step 4, you can do it now: `config['showfig'] = True`. config is a python dictionary.
8. refit the object: `results = wisp.fitandplot('Spectra/', 96, 16, 0.5326, 588, config)` ; the inputs here are

   - path to spectra
   - Par
   - Number
   - redshift guess – see the `ParXX_fwhm_z.dat` file!
   - fwhm guess (A). This is the FWHM of the object for lines in G141; the guess from the line-list is given in the same `ParXX_fwhm_z.dat` file. The spectra are fit with a single FWHM *in pixels*; hence the G102 FWHM are half the G141 FWHM. The parameter that is free in the model is the G141 FWHM, even when there are no lines in G141. It is scaled internally in the code. Additionally, the line-list FWHM were converted to rest-frame incorrectly. So the best guesses that we use are line-list/(1+z) = Observed FWM if the line in in G141, and twice that if it is in G102. We are making our best-guess from the line with the highest signal-to-noise in each object.

9. fiddle until you have the fit you like. This may just involve changing the fwhm guess or tweaking the redshift. Or, it may involve changing the configuration file/dictionary. I'll explain this below. Every time you run `wisp.fitandplot` it overwrites the figure (just the one for that object) in the `fitfigs` directory.

10. once you like the line fit, update the catalog: `wisp.update_object('Par96_refit', 16, results)` . Here the inputs are:
    - the catalog to update
    - the beam to update
    - the results from the fitting in steps 8/9.

    This step writes a new ParXX_fitfigs.pdf file, replacing the figure for the object we fixed.
11. Now you are done! (with one object.)

Fiddling with the configuration to get the fits right

1. first, what is in the configuration file?

```
 ########## configuration file ##############################
lambda_min           8500    Angstroms for blue cutoff
lambda_max           17000    Angstroms for red cutoff
transition_wave      11200    Angstroms at which switch between G102 and G141


###### detector things ####################################
#### estimated only required here.
dispersion_red           46.5       Angstroms/pix
dispersion_blue          23

##### line fitting ######
line_mask          250         window in  A to mask out when fitting preliminary continuum
fit_region          1500         window in A around lines to actually fit.
node_wave        9000 10000  11000 12000  13000  14000  15000  16000
mask_region1     0  0
mask_region2     0  0
mask_region3     0  0
fitfluxes        True          False = direct integrate ha+sii and oiii/hb.  TESTING ONLY. BLENDS.
showfig          False
max_fwhm_scl     2.0           fwhm can be up to 2x input guess
min_fwhm_scl     0.3           fwhm can be as small as 0.3 x input guess
```

The parameters can be described as:

- min, max, and transition wavelengths are straight-forward. they can be changed if the ends of the spectra are especially ratty/contaminated. Or, if a line falls in the G102/G141 break, it is best to get it entirely in one or the other. (The continuum may break under the line since the G102 and G141 continua are fit seperately).
- the dispersion is used only as an estimate; it is not very important
- `line_mask`: The model fitting is a two stage process. In step 1, we get an estimate of the continuum model by fitting line-free regions. This mask is what we use to skip over the lines. (Is it too small; I haven't had a chance to think about this yet.) This continuum model gives the initial parameters for continuum in a second step of fitting, when the lines+continuum are fit together. The continuum is a free parameter in the second step; the first step just gives the initial guesses.
- `fit_region`: Is somewhat historic, and not particularly important. The lines are evaluated over +/- `fit_region` in the model, and the spectrum is only fit between [OII] - `fit_region` and HeI 10830 + `fit_region`. One consequence is that some high (low) redshift galaxies may have blue (red) pixels that are not fit, because there is nothing interesting there to fit. This is fine.
- `node_wave` describes the wavelengths, in Å, of the nodes. These can be moved, or removed, or we could even add more.
- `mask_region` up to three regions of the spectrum can just be dropped entirely. If you want to remove between 9000 and 1000, just set any of the three `mask_region` to 9000 1000. no commas or anything.
- `fitfluxes` should be true. Currently it direct integrates around a couple lines, but because they are close there is blending. So it returns Ha + SII as Ha + [NII], or [OIII] + Hb as [OIII].
- `showfig` will bring up a python plot window. when you are running a loop to fit all the lines, you want this to be False; otherwise the loop will not proceed until you close the window. when you are investigating individual objects, you want it to be True.

- `max_fwhm_scl` and `min_fwhm_scl` set the allowed range around the input guess (taken from the line lists).

2. Fix example object 16: above, seemed to be extremely sensitive to the initial guess in z and fwhm.

3. Fix example object 29

```
#### Each time you call fitandplot, a plot window comes up.
results = wisp.fitandplot('Spectra/', 96, 29, 0.7033, 380, config)}
#the line is on the edge of G141/G102;  try a few different options
config['transition_wave']  = 11500.
results = wisp.fitandplot('Spectra/', 96, 29, 0.7033, 380, config)}
config['transition_wave']  = 11600.
results = wisp.fitandplot('Spectra/', 96, 29, 0.7033, 380, config)}
config['transition_wave']  = 11400.
results = wisp.fitandplot('Spectra/', 96, 29, 0.7033, 380, config)}
### liked 11500 the best
config['transition_wave']  = 11500.
results = wisp.fitandplot('Spectra/', 96, 29, 0.7033, 380, config)}
wisp.update_object('Par96_refit', 29, results)
### finally, the config dictionary should be reset to the original
config = wisp.read_config('refit.config')  ### or you can just reset config['transition_wave'] = 11200
```

4. Fix object 33: the spectrum rolls off on the edge of G141, where the [OIII] lines should be. there is also a bad pixel? on the red end of the spectrum slightly messing up the continuum fit under Ha.

```
config['transition_wave']  = 11500.
config['lambda_max'] = 16400
results = wisp.fitandplot('Spectra/', 96, 33, 1.337, 651, config)
wisp.update_object('Par96_refit', 33, results)
config = wisp.read_config('refit.config')
```

5. Fix Object 43: initial guesses on z and fwhm.

6. Fix Object 59: bad continuum contamination on red end of both G102 and G141; solution = decrease `config['lambda_max']` and use `config['mask_region1']` to remove the red end of G102.

```
config['mask_region1'] = [10200, 11200]
config['lambda_max'] = 15300
```

7. Fix Object 75: Lowered fwhm guess because [SII] was looking weird; moved config['transition_wave'] a bit redder, and deleted the spline node at 11000:

```
config['node_wave'] = [9000, 10000, 12000, 13000, 14000, 15000, 16000]
config['transition_wave'] = 11400
out = wisp.fitandplot('Spectra/', 96, 75, 0.65, 100, config)
wisp.update_object('Par96_refit', 75, out)
```

8. Fix Object 87: same as 59, slightly different parameters

Outstanding Issues

- there is no system for handling cases we don't exactly agree with the line list. In theory one could re-fit with a different z-guess without any problem. But this can be subjective; moreover, it can be complicated. For example, BEAM 27 in Par 96 cannot have its Ha and OIII fit with the same FWHM. Either this is scientifically interesting, or the lines are not from the same object. Both the "Ha" and "[OIII]" have a weaker line to the red looking like [SII]. We need to sort out a way to handle such objects.

- I have hard-coded in a delta-z, allowing the redshifts of the lines to vary slightly. There is a dz for [OII], [OIII] + Hb, and [SIII] + He1. The Ha sets the redshift. The dz for Hγ is the average of the dz for [OII] and [OIII] + Hb, in order to keep the number of parameters down. We should decide what maximum dz is allowed, and maybe pull this parameter out into the configuration file. Likewise, the dz for shifting relative to the line-list is hard coded.

- whenever there is no data -e.g. there are some pixels missing from the spectrum or if we masked out the pixels around a line, the line should be returned as -1 to signal "not covered". Currently the fits are returning something, which would lead one to think we have an upper limit.

- we need to decide– to what extent do we want to fuss with the fits. Many times they could be better, but you can tell it will also not change things very much, so is it worth it to double or triple the work to make the fits better?