In [71]:
```
from PIL import Image
import numpy as np
from astropy.io import fits
import glob
from PIL import Image as PILImage
import numpy as np
import pylab as pl
pl.rcParams['image.origin'] = 'lower'  # we want to show images, not matrices, sc
pl.matplotlib.style.use('dark_background')  # Optional configuration: if run, thi

from astropy import units as u
from astropy.modeling.polynomial import Polynomial1D
from astropy.modeling.models import Gaussian1D, Linear1D
from astropy.modeling.fitting import LinearLSQFitter
from IPython.display import Image
# astroquery provides an interface to the NIST atomic line database
from astroquery.nist import Nist
import glob
import os
from astropy.io import fits

from astropy.modeling.polynomial import Polynomial1D
from astropy.modeling.fitting import LinearLSQFitter

from astropy.modeling.models import Gaussian1D
from astropy.modeling.fitting import LevMarLSQFitter
```

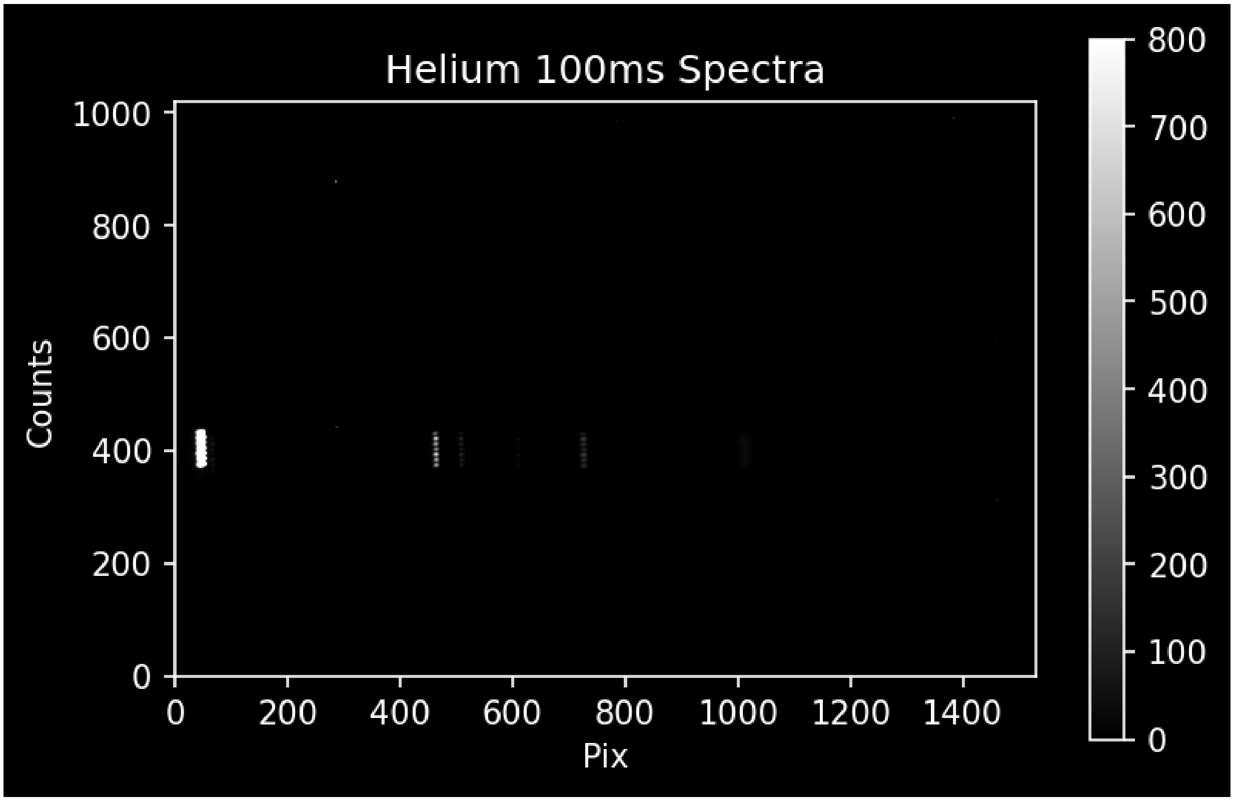
In [72]:
```
he100ms_image_data =  (np.mean([fits.getdata(x) for x in glob.glob("\\Users\\Sydr
                      axis=0)
              - np.mean([fits.getdata(x)
                        for x in glob.glob("\\Users\\Sydnee O'Donnell\\OneDrive\
                       axis=0)
             )
```

In [73]:

```
%matplotlib inline
import pylab as pl
pl.rcParams['image.origin'] = 'lower'
pl.rcParams['figure.dpi'] = 150
pl.matplotlib.style.use('dark_background') # Optional!
pl.imshow(he100ms_image_data, cmap='gray', vmax=0, vmin=800)
pl.colorbar()
pl.xlabel('Pix')
pl.ylabel('Counts')
pl.title('Helium 100ms Spectra')
```

Out[73]:

```
Text(0.5, 1.0, 'Helium 100ms Spectra')
```

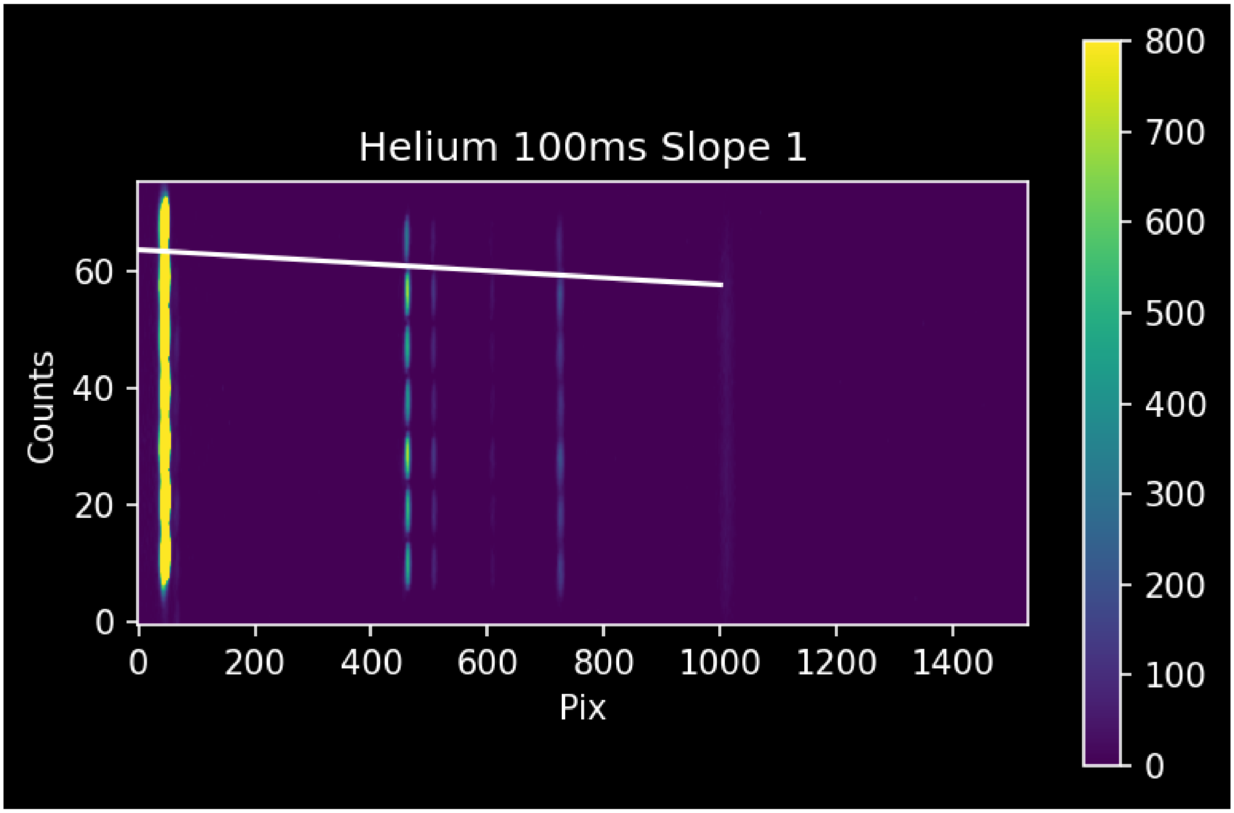


In [74]:

```
# I drew a line between the top two spectra
dy = -6
dx = 1000
slope = dy/dx
ystart = 365
yend = 441

image_array = np.array(he100ms_image_data)
image_array = image_array - np.median(he100ms_image_data)
pl.imshow(he100ms_image_data[ystart:yend,:], vmax=0, vmin=800)
pl.colorbar()
pl.plot([0,1000], 63.8 + np.array([0,1000]) * slope, color='w')
pl.gca().set_aspect(10)
pl.xlabel('Pix')
pl.ylabel('Counts')
pl.title('Helium 100ms Slope 1')
```

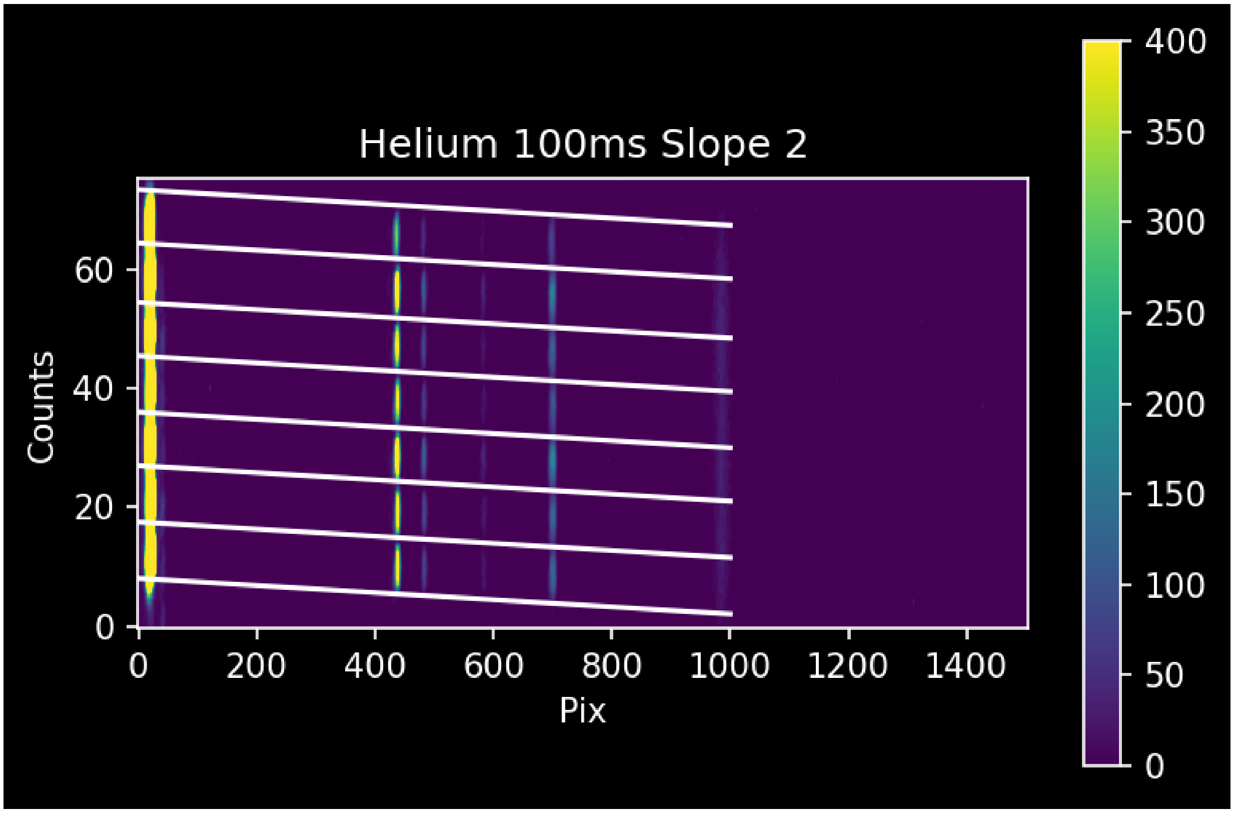
Out[74]:

```
Text(0.5, 1.0, 'Helium 100ms Slope 1')
```



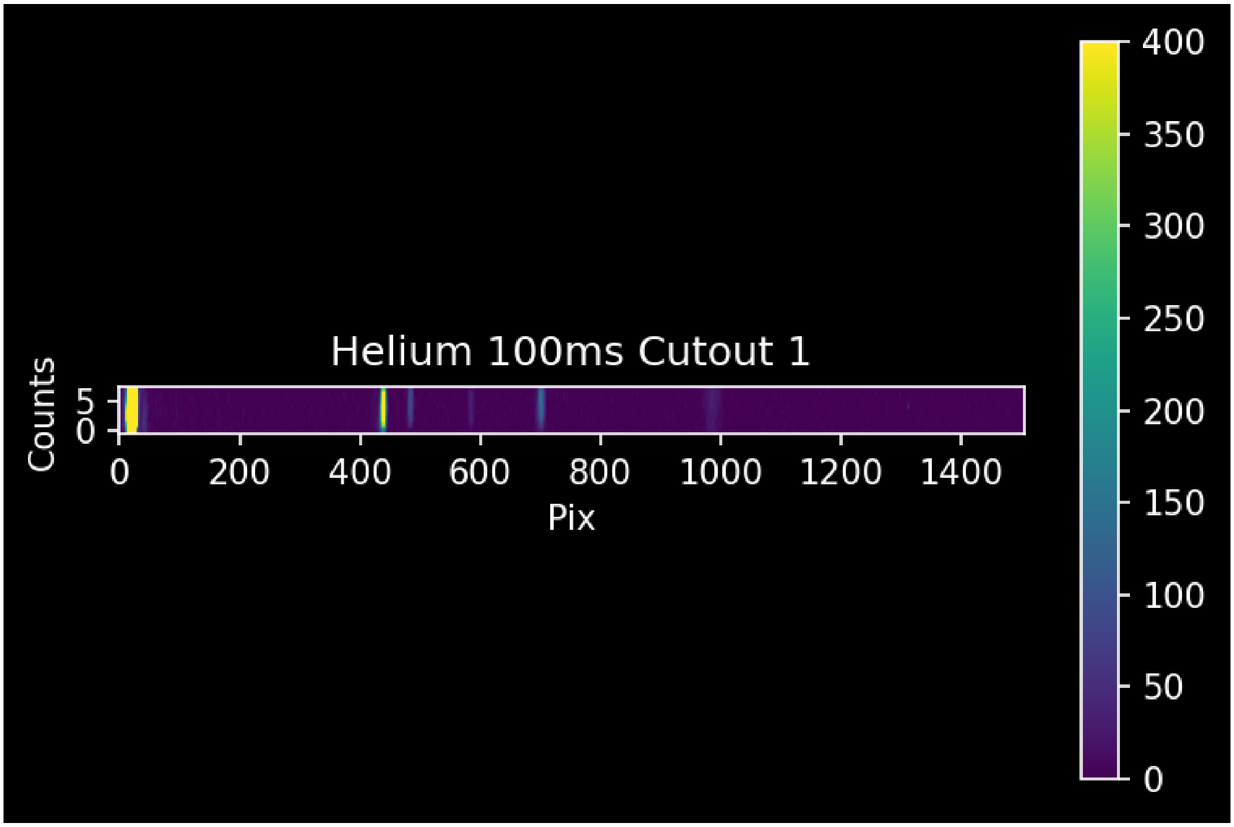
In [75]:
```
intertrace_cuts = np.array([ 8, 17.5, 27, 36, 45.5, 54.5, 64.5, 73.5])
image_array = np.array(he100ms_image_data[:,25:],)
image_array = image_array - np.median(he100ms_image_data[:,25:],)
pl.imshow(he100ms_image_data[ystart:yend,25:], vmax=0, vmin=400)
pl.colorbar()
pl.plot([0,1000], intertrace_cuts + np.array([0,1000])[:,None] * slope, color='w'
pl.gca().set_aspect(10)
pl.xlabel('Pix')
pl.ylabel('Counts')
pl.title('Helium 100ms Slope 2')
```

Out[75]:
```
Text(0.5, 1.0, 'Helium 100ms Slope 2')
```



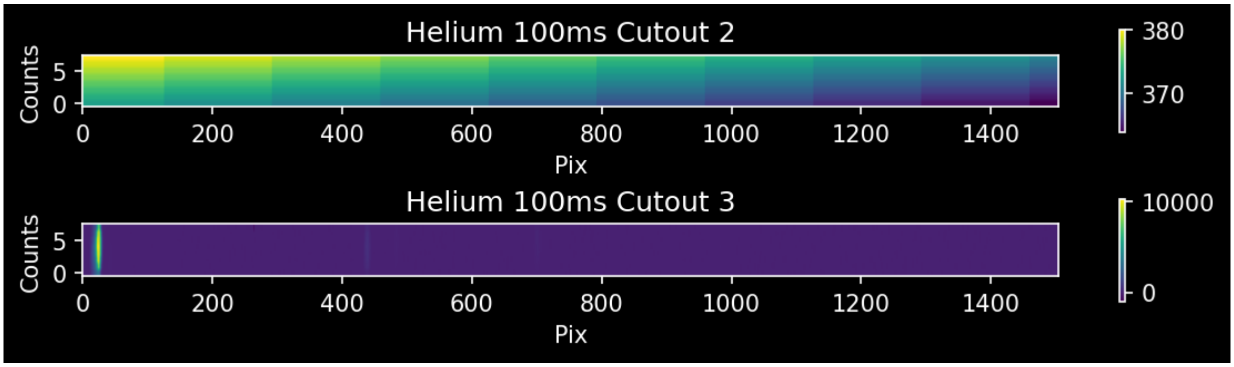
```
In [76]: npixels_to_cut = 4 # very conservative - we'll see why below
         xvals = np.arange(image_array.shape[1])
         trace_center = ystart+(intertrace_cuts[0] + intertrace_cuts[1])/2 + xvals * slope
         cutout_trace = np.array([image_array[int(yval)-npixels_to_cut:int(yval)+npixels_t
                                 for yval, ii in zip(trace_center, xvals)]).T
         cutout_trace.shape

         pl.imshow(cutout_trace, vmax=0, vmin=400)
         pl.colorbar()
         pl.gca().set_aspect(10);
         pl.xlabel('Pix')
         pl.ylabel('Counts')
         pl.title('Helium 100ms Cutout 1')
```

```
Out[76]: Text(0.5, 1.0, 'Helium 100ms Cutout 1')
```



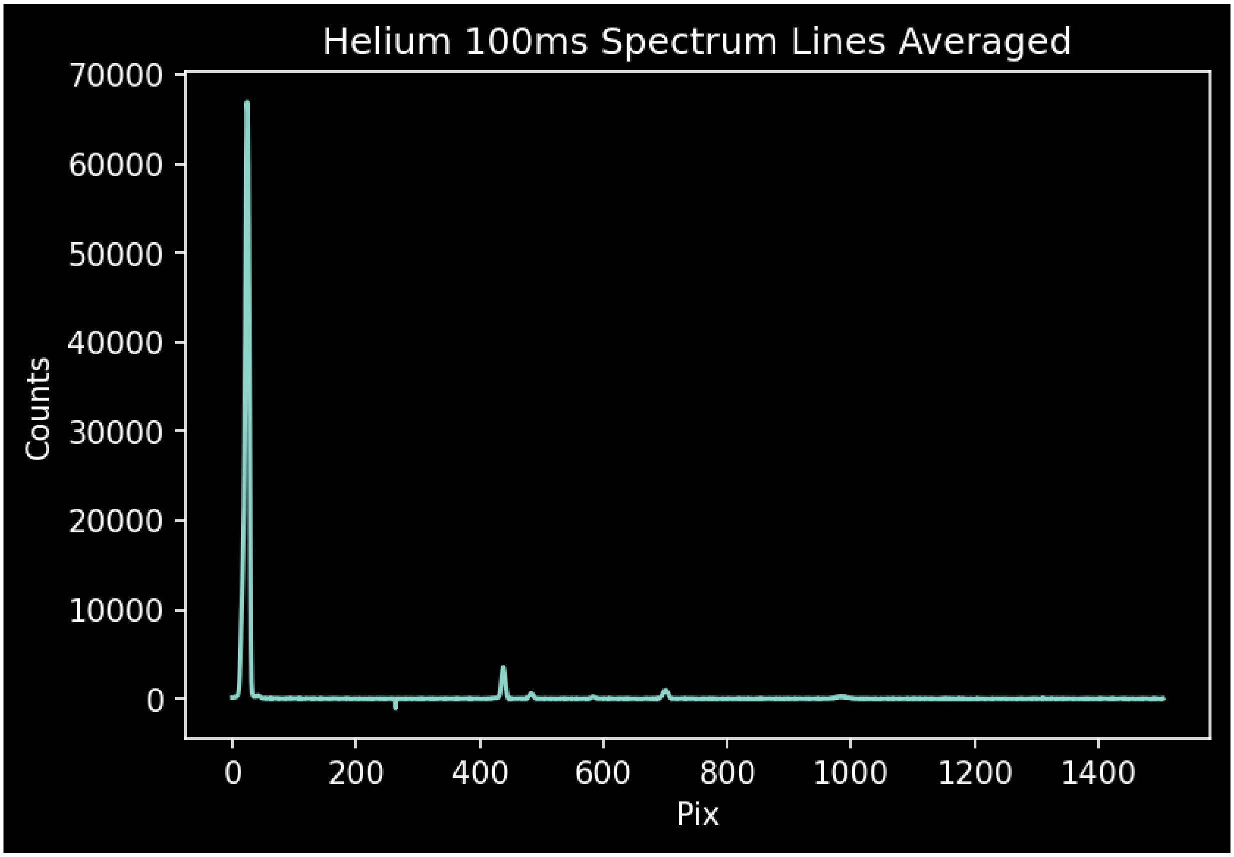
In [77]:
```
# to get the y-axis values corresponding to each part of our cutout trace, we do
yaxis_full = np.arange(image_array.shape[0])
yaxis = np.array([yaxis_full[int(yval)-npixels_to_cut:int(yval)+npixels_to_cut]
                    for yval, ii in zip(trace_center, xvals)]).T

pl.figure(figsize=(8,2))
im = pl.subplot(2,1,1).imshow(yaxis)
pl.colorbar(mappable=im)
pl.gca().set_aspect(10);
pl.title('Helium 100ms Cutout 2')
pl.xlabel('Pix')
pl.ylabel('Counts')
im = pl.subplot(2,1,2).imshow(cutout_trace)
pl.colorbar(mappable=im)
pl.gca().set_aspect(10);
pl.tight_layout()
pl.title('Helium 100ms Cutout 3')
pl.xlabel('Pix')
pl.ylabel('Counts')
```

Out[77]:
```
Text(113.83333333333333, 0.5, 'Counts')
```

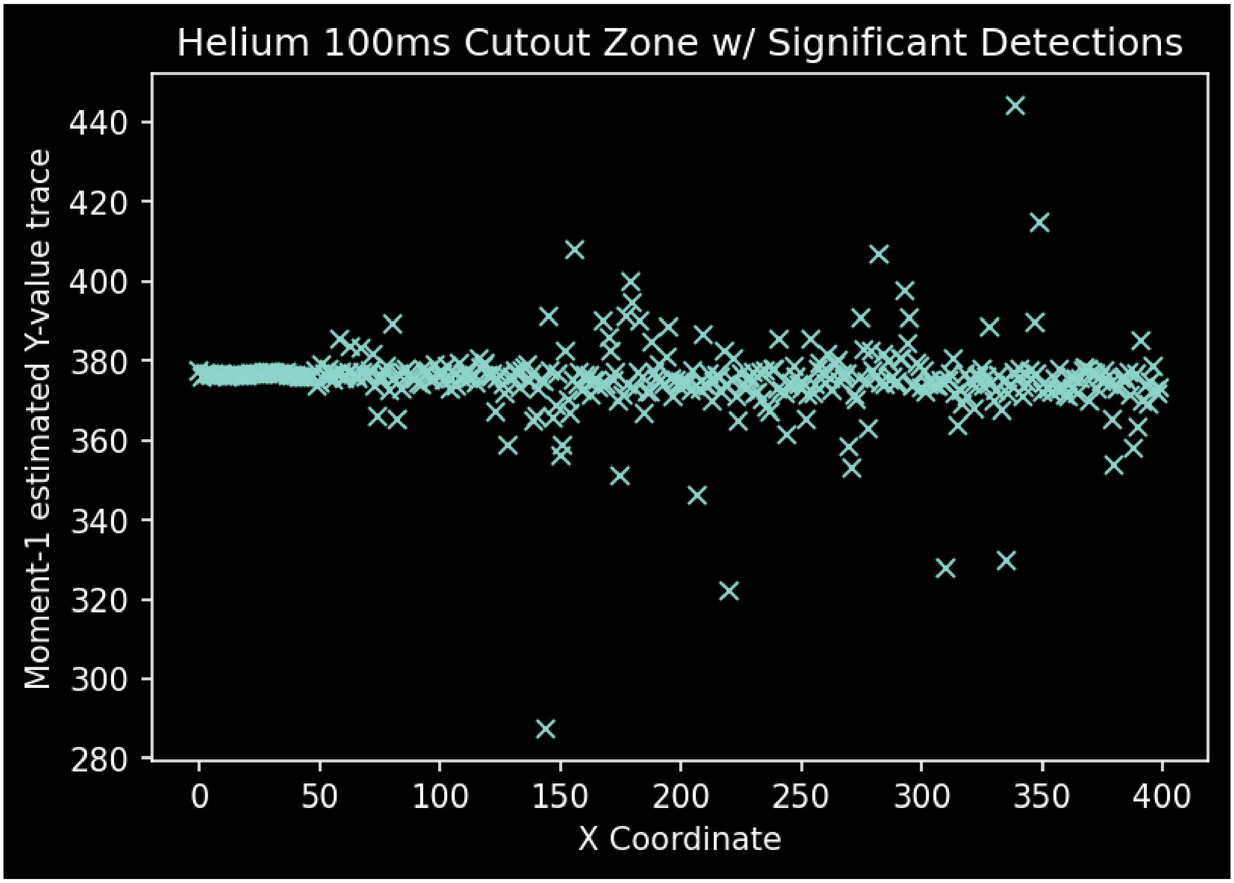


In [78]:
```
pl.plot(cutout_trace.sum(axis=0))
pl.title('Helium 100ms Spectrum Lines Averaged')
pl.xlabel('Pix')
pl.ylabel('Counts')
```

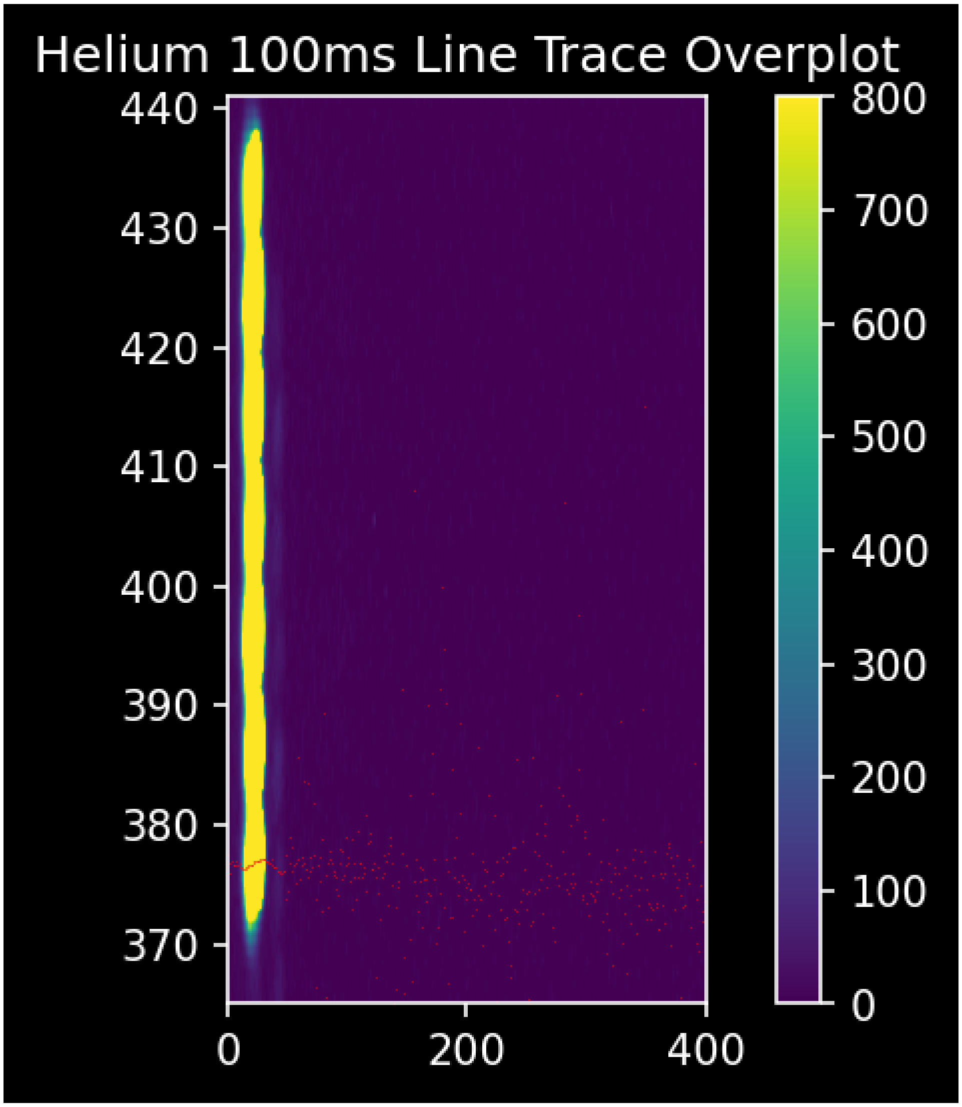
Out[78]:
```
Text(0, 0.5, 'Counts')
```



In [79]:
```python
# moment 1 is the data-weighted average of the Y-axis coordinates
xend = 400
weighted_yaxis_values = np.average(yaxis[:,:xend], axis=0,
                                   weights=cutout_trace[:,:xend])

_=pl.plot(xvals[:xend], weighted_yaxis_values, 'x')
_=pl.xlabel("X Coordinate")
_=pl.ylabel("Moment-1 estimated Y-value trace")
_=pl.title("Helium 100ms Cutout Zone w/ Significant Detections")
```



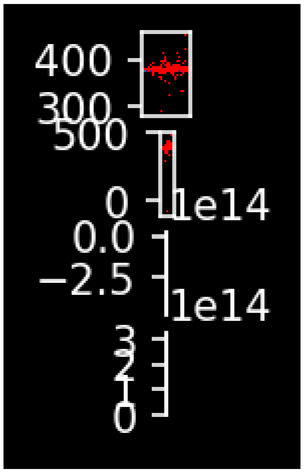
In [80]:
```
# we need to use the 'extent' keyword to have the axes correctly labeled
_=pl.imshow(image_array[ystart:yend, :xend],
            extent=[0,xend,ystart,yend], vmax=0, vmin=800)
_=pl.colorbar()
_=pl.gca().set_aspect(10) # we stretch the image out by 10x in the y-direction
_=pl.plot(xvals[:xend], weighted_yaxis_values[:xend], 'r,', alpha=0.5)
_=pl.axis((0,xend,ystart,yend))
_=pl.title("Helium 100ms Line Trace Overplot")
```



In [81]:

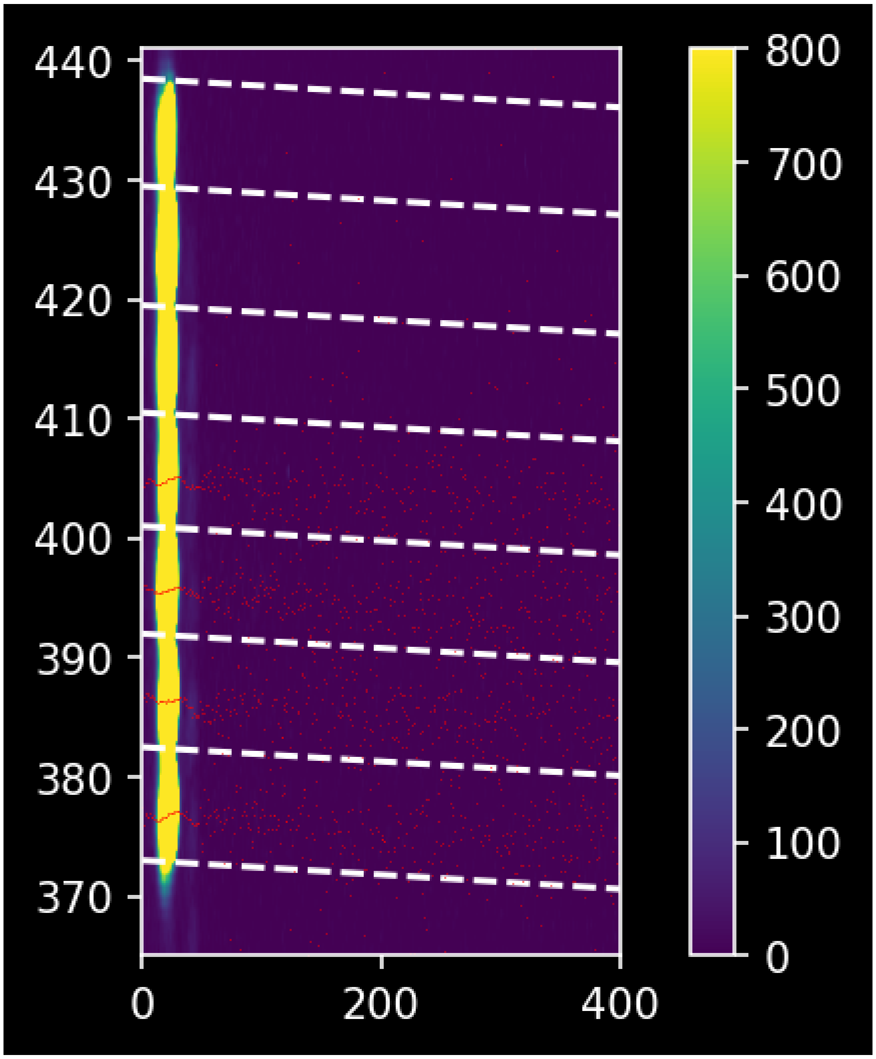
```
## repeated for each figure
pl.figure(figsize=(8,3))
traces = {}
for trace_index in range(len(intertrace_cuts)-1):
    yoffset = ystart + (intertrace_cuts[trace_index] + intertrace_cuts[trace_inde
    trace_center = yoffset + slope * xvals

    cutout_trace = np.array([image_array[int(yval)-npixels_to_cut:int(yval)+npixe
                             for yval, ii in zip(trace_center, xvals)]).T
    yaxis = np.array([yaxis_full[int(yval)-npixels_to_cut:int(yval)+npixels_to_cu
                      for yval, ii in zip(trace_center, xvals)]).T
    weighted_yaxis_values = np.average(yaxis[:,:xend], axis=0,
                                       weights=cutout_trace[:,:xend])

    # it takes a little mental gymnastics to get to this, but: to show the trace
    # we need to calculate the local version
    local_weighted_yaxis_values = np.average(np.arange(npixels_to_cut*2)[:,None]
                                             axis=0, weights=cutout_trace[:,:xend

    traces[trace_index] = weighted_yaxis_values
    ax = pl.subplot(7, 1, trace_index+1)
    ax.imshow(cutout_trace[:,:xend], extent=[0, xend, yoffset-npixels_to_cut, yof
    ax.plot(xvals[:xend], yoffset - npixels_to_cut + local_weighted_yaxis_values|
    ax.set_aspect(4)
    ax.set_xticks([])
pl.tight_layout()
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-81-d63867b73153> in <module>
     10     yaxis = np.array([yaxis_full[int(yval)-npixels_to_cut:int(yval)+npi
xels_to_cut]
     11                       for yval, ii in zip(trace_center, xvals)]).T
---> 12     weighted_yaxis_values = np.average(yaxis[:,:xend], axis=0,
     13                                        weights=cutout_trace[:,:xend])
     14 

<__array_function__ internals> in average(*args, **kwargs)

C:\ProgramData\Anaconda3\lib\site-packages\numpy\lib\function_base.py in averag
e(a, axis, weights, returned)
    407         scl = wgt.sum(axis=axis, dtype=result_dtype)
    408         if np.any(scl == 0.0):
--> 409             raise ZeroDivisionError(
    410                 "Weights sum to zero, can't be normalized")
    411 

ZeroDivisionError: Weights sum to zero, can't be normalized
```


400
300
500
0
1e14
0.0
−2.5
1e14
3
2
1
0

In [82]:
```
# then we can plot the "global" version here
pl.imshow(image_array[ystart:yend, :xend],
          extent=[0,xend,ystart,yend], vmax=0, vmin=800)
pl.colorbar()
pl.plot([0,xend], ystart + intertrace_cuts + np.array([0,xend])[:,None] * slope,
pl.gca().set_aspect(10)
for trace in traces.values():
    pl.plot(xvals[:xend], trace[:xend], 'r,', alpha=0.5)
pl.axis((0,xend,ystart,yend))
```

Out[82]:
```
(0.0, 400.0, 365.0, 441.0)
```



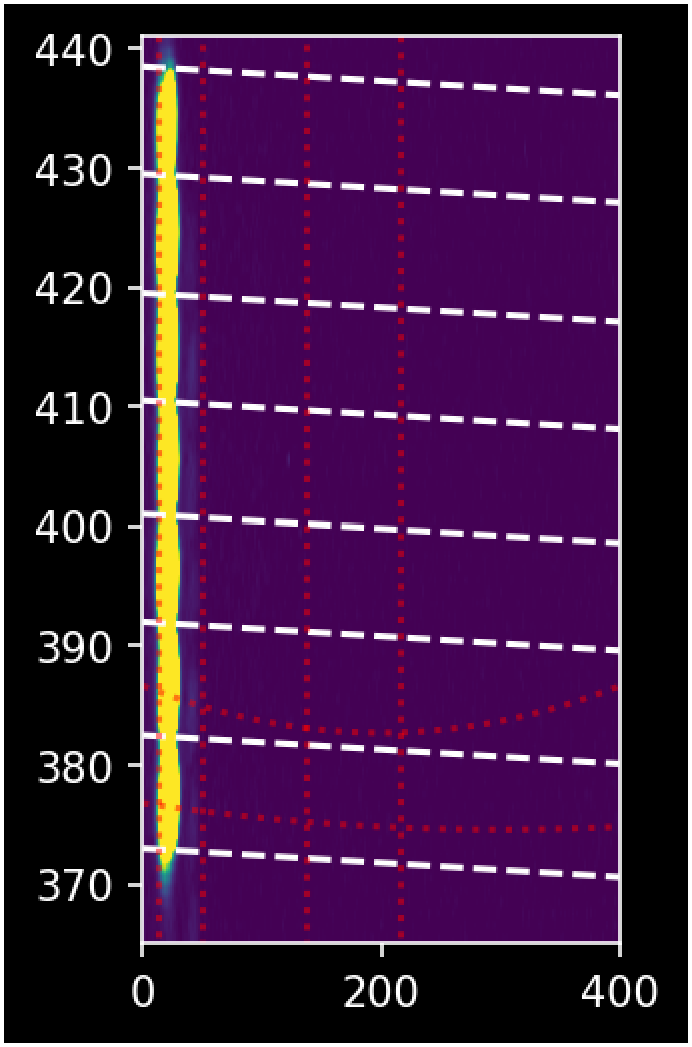
In [83]:
```
# We fit a 2rd-order polynomial
polymodel = Polynomial1D(degree=2)
linfitter = LinearLSQFitter()
fitted_polymodels = {index: linfitter(polymodel, xvals[:xend], weighted_yaxis_val
                     for index, weighted_yaxis_values in traces.items()}
```

In [84]:
```
fitted_polymodels
```

Out[84]:
```
{0: <Polynomial1D(2, c0=376.79598985, c1=-0.01477551, c2=0.00002466)>,
 1: <Polynomial1D(2, c0=386.65609624, c1=-0.03947801, c2=0.0000985)>,
 2: <Polynomial1D(2, c0=-1.15293855e+12, c1=2.81670185e+10, c2=-1.05497957e+08)
>,
 3: <Polynomial1D(2, c0=1.3157346e+11, c1=-1.18887261e+10, c2=79437912.8699785)
>}
```

In [85]:
```
pl.imshow(image_array[ystart:yend, :xend],
          extent=[0,xend,ystart,yend],
          vmin=0, vmax=700,
         )
pl.plot([0,xend], ystart + intertrace_cuts + np.array([0,xend])[:,None] * slope,
pl.gca().set_aspect(10)
for tracefit in fitted_polymodels.values():
    pl.plot(xvals[:xend], tracefit(xvals[:xend]), 'r:', alpha=0.5)
pl.axis((0,xend,ystart,yend))
```

Out[85]:
```
(0.0, 400.0, 365.0, 441.0)
```



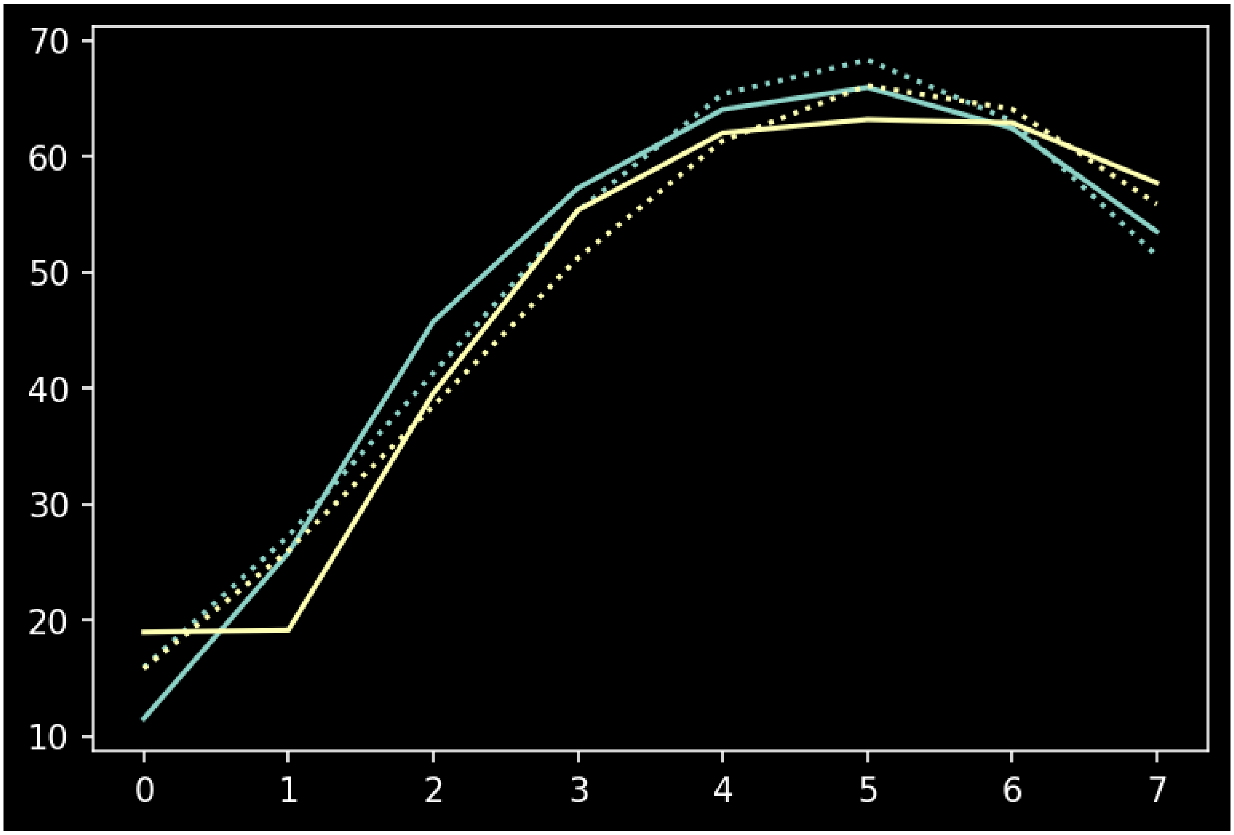
In [86]:
```
lmfitter = LevMarLSQFitter()
guess = Gaussian1D(amplitude=160, mean=0, stddev=5)
```

In [87]:

```
npixels_to_cut_trace = 4

for trace_index, polymodel_trace in fitted_polymodels.items():
    trace_center = polymodel_trace(xvals)

    cutout_trace = np.array([image_array[int(yval)-npixels_to_cut_trace:int(yval)
                            for yval, ii in zip(trace_center, xvals)]).T

    trace_profile = cutout_trace.mean(axis=1)
    trace_profile_xaxis = np.arange(len(trace_profile))
    fitted_trace_profile = lmfitter(model=guess, x=trace_profile_xaxis, y=trace_p
    model_trace_profile = fitted_trace_profile(trace_profile_xaxis)

    line, = pl.plot(trace_profile, label=trace_index)
    pl.plot(trace_profile_xaxis, model_trace_profile, color=line.get_color(), lin
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-87-56e7cbef06c7> in <module>
      9     trace_profile = cutout_trace.mean(axis=1)
     10     trace_profile_xaxis = np.arange(len(trace_profile))
---> 11     fitted_trace_profile = lmfitter(model=guess, x=trace_profile_xaxi
s, y=trace_profile)
     12     model_trace_profile = fitted_trace_profile(trace_profile_xaxis)
     13

~\AppData\Roaming\Python\Python38\site-packages\astropy\modeling\fitting.py i
n wrapper(self, model, x, y, z, **kwargs)
    259         else:
    260
--> 261             return func(self, model, x, y, z=z, **kwargs)
    262
    263     return wrapper

~\AppData\Roaming\Python\Python38\site-packages\astropy\modeling\fitting.py i
n __call__(self, model, x, y, z, weights, maxiter, acc, epsilon, estimate_jac
obian)
   1154             dfunc = self._wrap_deriv
   1155         init_values, _ = _model_to_fit_params(model_copy)
-> 1156         fitparams, cov_x, dinfo, mess, ierr = optimize.leastsq(
   1157             self.objective_function, init_values, args=farg, Dfun=dfu
nc,
   1158             col_deriv=model_copy.col_fit_deriv, maxfev=maxiter, epsfc
n=epsilon,

C:\ProgramData\Anaconda3\lib\site-packages\scipy\optimize\minpack.py in least
sq(func, x0, args, Dfun, full_output, col_deriv, ftol, xtol, gtol, maxfev, ep
sfcn, factor, diag)
    412
    413     if n > m:
--> 414         raise TypeError('Improper input: N=%s must not exceed M=%s' %
(n, m))
    415
    416     if epsfcn is None:
```

```
TypeError: Improper input: N=3 must not exceed M=0
```



```
In [88]: spectra = {}
         for trace_index, polymodel_trace in fitted_polymodels.items():
             trace_center = polymodel_trace(xvals)

             cutout_trace = np.array([image_array[int(yval)-npixels_to_cut_trace:int(yval)
                                     for yval, ii in zip(trace_center, xvals)]).T

             trace_profile = cutout_trace.mean(axis=1)
             trace_profile_xaxis = np.arange(len(trace_profile))
             fitted_trace_profile = lmfitter(model=guess, x=trace_profile_xaxis, y=trace_p
             model_trace_profile = fitted_trace_profile(trace_profile_xaxis)

             trace_avg_spectrum = np.array([np.average(
                     image_array[int(yval)-npixels_to_cut:int(yval)+npixels_to_cut, ii],
                     weights=trace_profile)
                                    for yval, ii in zip(trace_center, xvals)])
             spectra[trace_index] = trace_avg_spectrum
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-88-75e755bd5fa6> in <module>
      8     trace_profile = cutout_trace.mean(axis=1)
      9     trace_profile_xaxis = np.arange(len(trace_profile))
---> 10     fitted_trace_profile = lmfitter(model=guess, x=trace_profile_xaxis,
y=trace_profile)
     11     model_trace_profile = fitted_trace_profile(trace_profile_xaxis)
     12 

~\AppData\Roaming\Python\Python38\site-packages\astropy\modeling\fitting.py in
wrapper(self, model, x, y, z, **kwargs)
    259         else:
    260 
--> 261             return func(self, model, x, y, z=z, **kwargs)
    262 
    263     return wrapper

~\AppData\Roaming\Python\Python38\site-packages\astropy\modeling\fitting.py in
__call__(self, model, x, y, z, weights, maxiter, acc, epsilon, estimate_jacobia
n)
   1154             dfunc = self._wrap_deriv
   1155         init_values, _ = _model_to_fit_params(model_copy)
-> 1156         fitparams, cov_x, dinfo, mess, ierr = optimize.leastsq(
   1157             self.objective_function, init_values, args=farg, Dfun=dfunc
,
   1158             col_deriv=model_copy.col_fit_deriv, maxfev=maxiter, epsfcn=
epsilon,

C:\ProgramData\Anaconda3\lib\site-packages\scipy\optimize\minpack.py in leastsq
(func, x0, args, Dfun, full_output, col_deriv, ftol, xtol, gtol, maxfev, epsfc
n, factor, diag)
    412 
    413     if n > m:
--> 414         raise TypeError('Improper input: N=%s must not exceed M=%s' % (
n, m))
    415 
    416     if epsfcn is None:
```

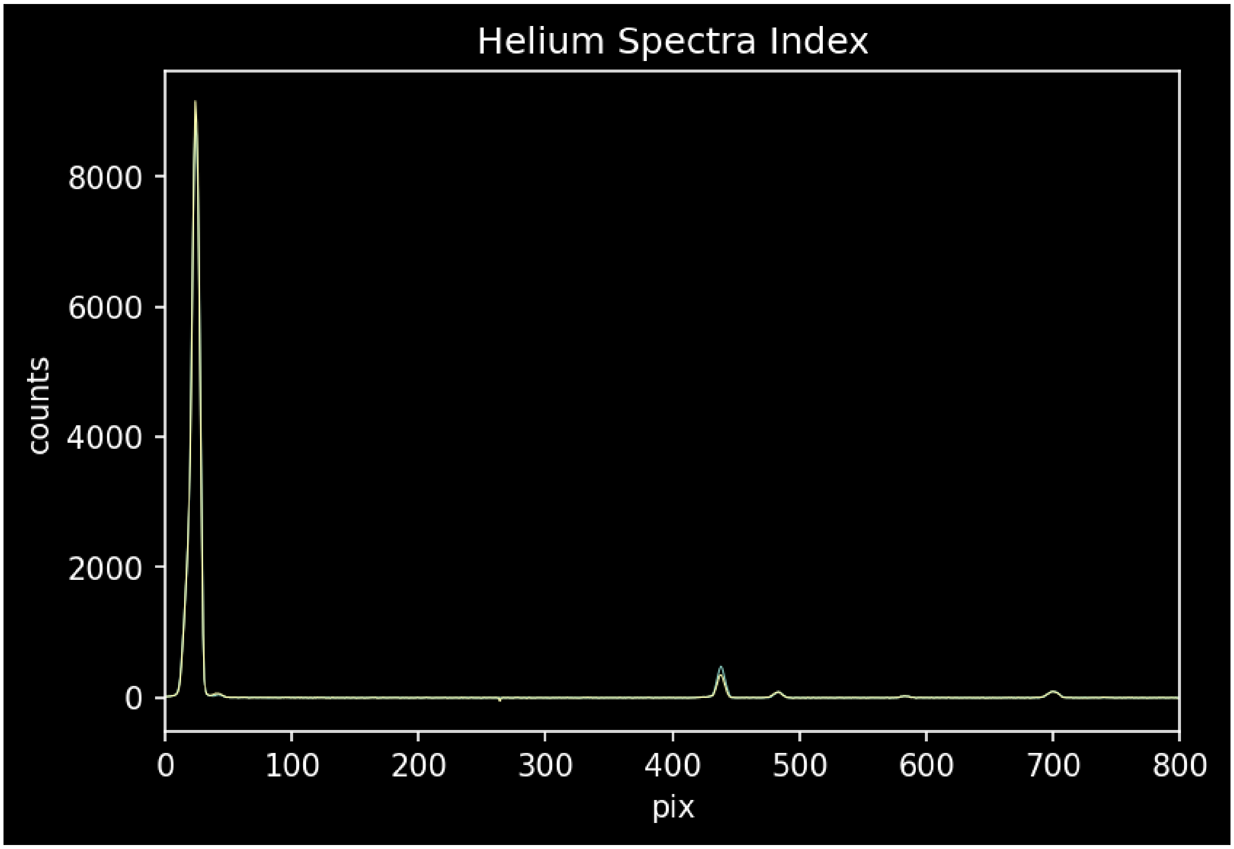
```
TypeError: Improper input: N=3 must not exceed M=0
```

In [89]:

```
for index in spectra:
    pl.plot(spectra[index], linewidth=0.5)
    pl.xlabel('pix')
    pl.ylabel('counts')
    pl.title('Helium Spectra Index')
pl.xlim(0,800)
```

Out[89]:

```
(0.0, 800.0)
```



In [ ]: