Week 5

Solution

In [1]:

```
import numpy as np
import os
import fnmatch
from osgeo import gdal
from matplotlib import pyplot as plt
from matplotlib import patches as mpatches
from matplotlib import colors
```

In [2]:

```
def plotDisturbanceYearMap(img):

    values = np.unique(gdyear.ravel())
    values = values[values > 0]

    cmap = plt.get_cmap('jet', values.size)
    cmap.set_under('grey')
    norm = colors.BoundaryNorm(list(values), cmap.N)

    plt.figure(figsize=(8, 4))
    im = plt.imshow(img, interpolation='none', cmap=cmap, norm=norm)

    dist_colors = [im.cmap(im.norm(value)) for value in values]
    # create a patch (proxy artist) for every color
    patches = [mpatches.Patch(color=dist_colors[i], label=values[i]) for i in range(len(values))]
    # put those patched as legend-handles into the legend
    plt.legend(handles=patches, bbox_to_anchor=(1.05, 1), loc=2,
               borderaxespad=0., ncol=2, frameon=False)

    plt.grid(True)
    plt.show()
```

1 Create greatest disturbance image

Create a single layer raster where each pixel has the year of the greatest disturbance (largest positive change magnitude) if it was disturbed or 0 if it was undisturbed or non-forest. Undisturbed are all pixels with a change magnitude less than 500.

Read forest mask. The mask has the same x- and y- dimension.

In [3]:

```
for_ds = gdal.Open('data/gpy_poland_forestmask.tif')
formask = for_ds.ReadAsArray()
for_ds = None
formask.shape
```

Out[3]:

```
(461, 514)
```

Read vertex image. There are 22 bands in the following order: 7 layers denoting the vertex year, 7 layers denoting the original reflectance at the corresponding vertex years, 7 layers denoting the fitted reflectance at the corresponding vertex years, and 1 layer of the mean square error of the fit.

In [4]:

```
src_ds = gdal.Open('data/gpy_poland_landtrendr_tcw_vertex_8618.tif')
img = src_ds.ReadAsArray()
src_ds = None
img.shape
```

Out[4]:

```
(22, 461, 514)
```

Split vertex array into time-axis (year) and spectral-axis (fit). There are 7 years (index 0 to 6) and 7 fitted values (index 14 to 20).

In [5]:

```
year = img[0:7, :, :]
fit = img[14:21, :, :]
```

Since the fill/missing values are 0s, we can avoid them in the difference calculation by replacing them with NAN values. In this specific example, this is not necessary, since we are looking for maximum change difference/magnitude greater than 500 and differences (with following 0s) will become either negative or zero - and. But let's do this anyway.

In [7]:

```
fillMask = year == 0
fit = fit.astype(np.float32)
fit[fillMask] = np.NAN
```

Now let's calculate the change magnitude using the np.diff() function. np.diff() calculates the difference between t_x and t_x+1 as follows: delta = t_x+1 - t_x.

In [8]:

```
delta = np.diff(fit, axis=0)
delta.shape
```

Out[8]:

```
(6, 461, 514)
```

With so many NaN, I got an error in the subsequent step. To avoid the error in nanargmax, I replaced NaN with 0 again.

In [9]:

```
delta[np.isnan(delta)] = 0
```

To retrieve the array indices for maximum magnitude pixels along the first axis (time axis), you can use the function np.nanargmax(). The resulting index array as the same x- and y-dimension, and the pixel values represent the index/position in delta along the time axis.

In [10]:

```
inds = np.nanargmax(delta, axis=0)
inds.shape
```

Out[10]:

```
(461, 514)
```

Now, we can extract the year of greatest disturbance. Here, I pick the starting vertex year of each segment (6 segments -> 6 starting vertices). This picks the year prior to disturbance, which is OK for this excercise. Alernatively, we could add +1 to obtain the following year or pick the end vertex of each segmment (year[1:7]) (not done here).

In [11]:

```
gdyear = np.take_along_axis(year[0:6, :, :], inds[np.newaxis, :, :], 0).sque
eze()
gdyear.shape
```

Out[11]:

```
(461, 514)
```

We only want to include pixels with a greatest disturbance change magnitude greater than 500. In order to create a magnitude mask later we need to create an image of the greatest disturbance magnitude.

In [12]:

```
#gdmag = np.take_along_axis(delta, inds[np.newaxis, :, :], 0).squeeze()
gdmag = np.max(delta, axis=0)
```

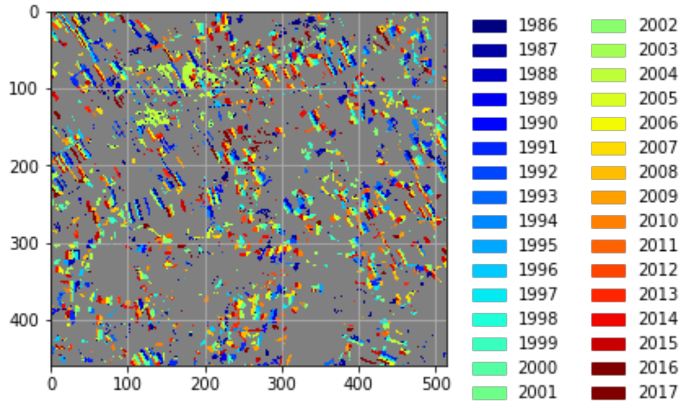
Filter out pixels with change magnitude less than 500 and non-forest pixels

In [13]:

```
gdyear[gdmag < 500] = 0
gdyear[formask == 0] = 0
```

In [14]:

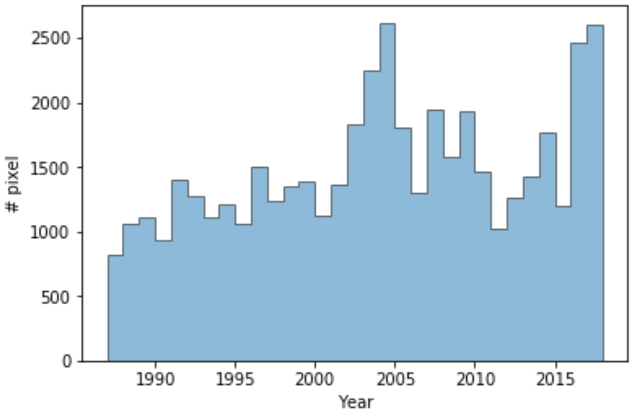
```
plotDisturbanceYearMap(gdyear)
```



2 Create a histogram that shows the annual forest disturbance frequency between 1987 and 2018 for the study area.

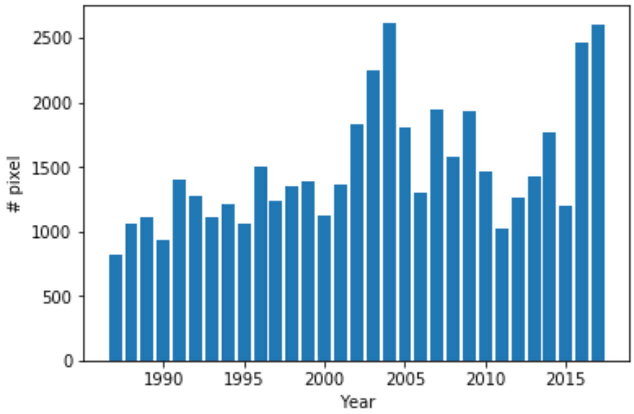
In [15]:

```
# flatten array
data = gdyear.ravel()
plt.hist(data, range=(1987, 2018), bins=range(1987, 2019),
         histtype='stepfilled', alpha=0.5, edgecolor='black')
plt.xlabel('Year')
plt.ylabel('# pixel')
plt.savefig('greatest_disturbance_freq1.png')
plt.show()
```



In [16]:

```
year, count = np.unique(gdyear, return_counts=True)
plt.bar(year[2:], count[2:])
plt.xlabel('Year')
plt.ylabel('# pixel')
plt.savefig('greatest_disturbance_freq2.png')
plt.show()
```


# pixel
Year