INTRODUCTION TO DATA VISUALIZATION WITH PYTHON

Visualizing time series

Datetimes & time series

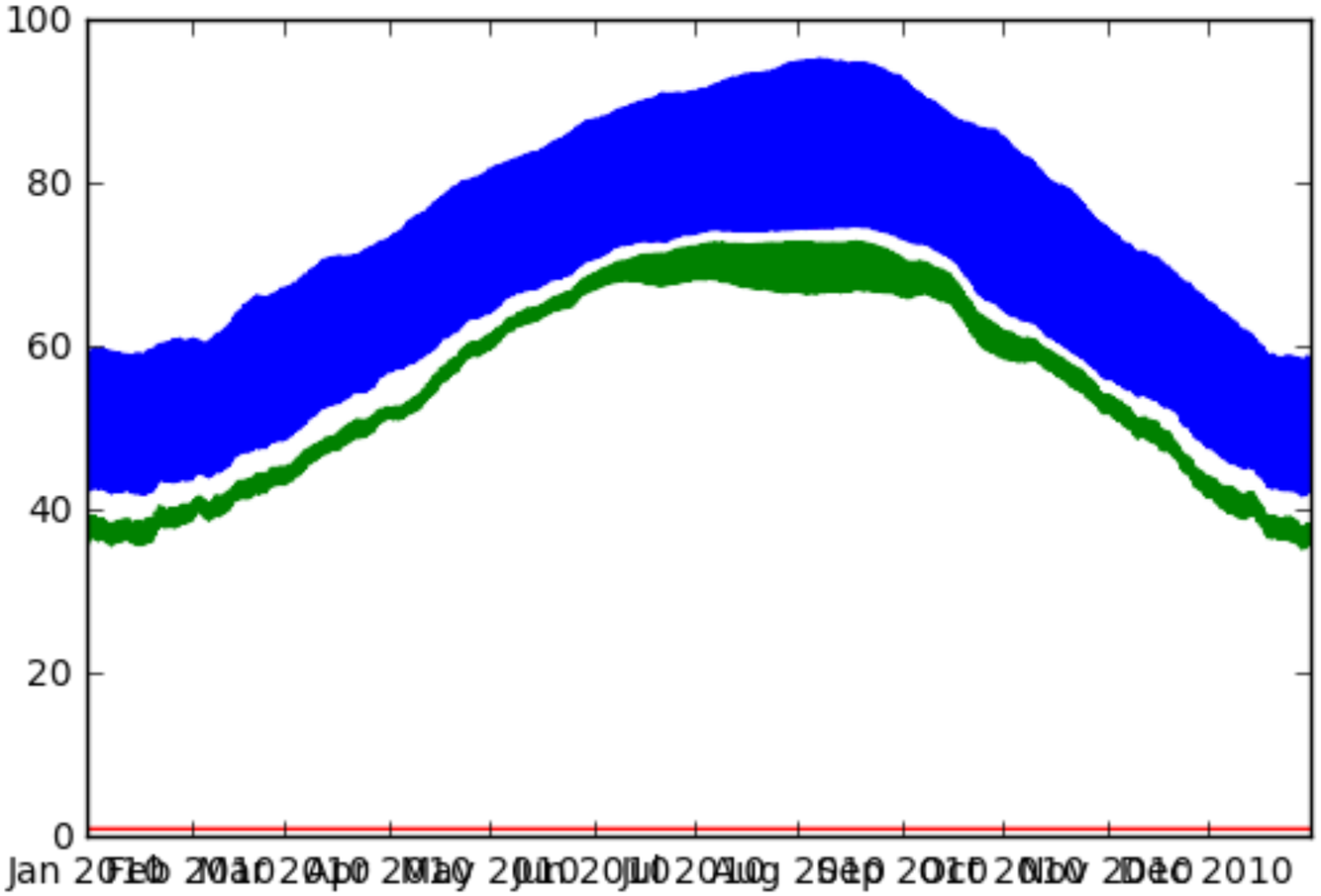
```
In [1]: type(weather)
Out[1]: pandas.core.frame.DataFrame

In [2]: type(weather.index)
Out[2]: pandas.tseries.index.DatetimeIndex
```

Date	Temperature	DewPoint	Pressure
2010-01-01 00:00:00	46.2	37.5	1.0
2010-01-01 01:00:00	44.6	37.1	1.0
2010-01-01 02:00:00	44.1	36.9	1.0
2010-01-01 03:00:00	43.8	36.9	1.0
2010-01-01 04:00:00	43.5	36.8	1.0
...	...	...	...

Plotting DataFrames

```
In [1]: plt.plot(weather)

In [2]: plt.show()
```

Plotting DataFrames



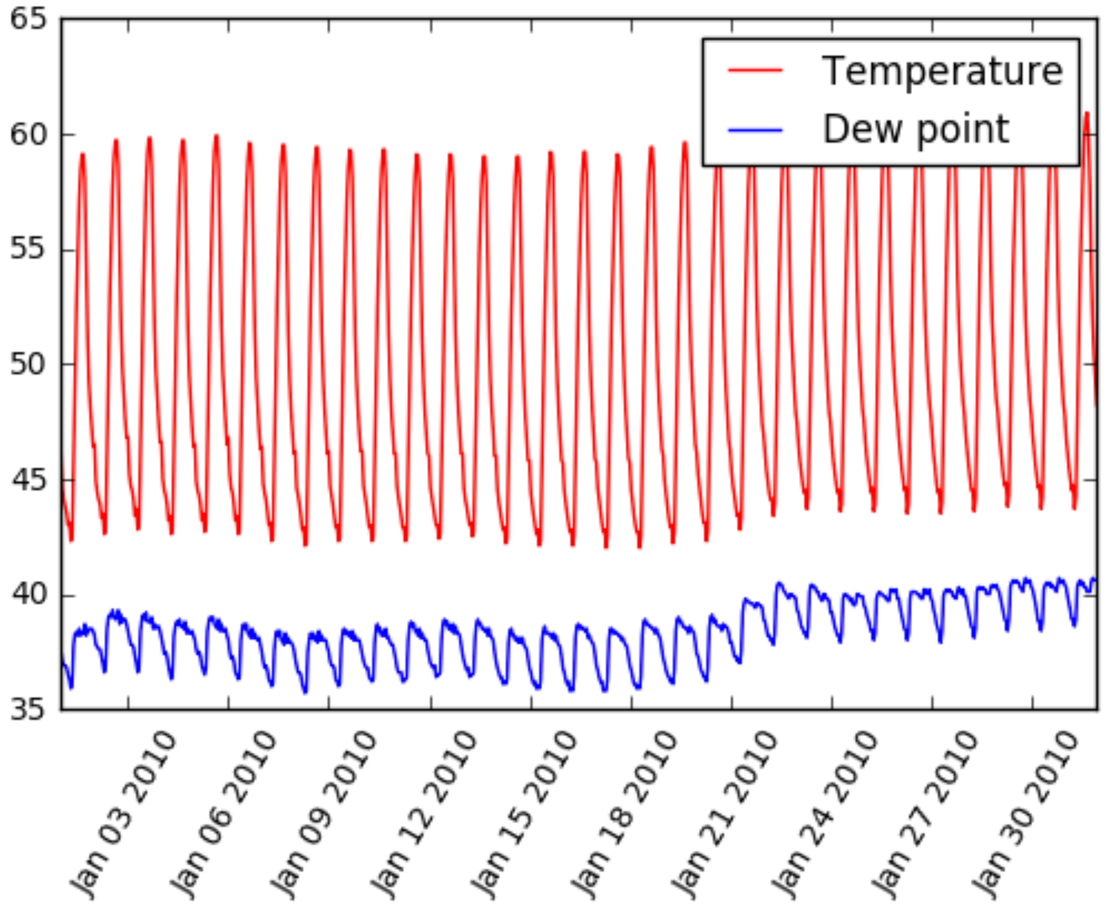
Time series

- Pandas time series: *datetime* as *index*
- Datetime: represents periods or time-stamps
- Datetime index: specialized slicing
  - e.g., weather['2010-07-04']
  - e.g., weather['2010-03':'2010-04']
  - e.g., weather['2010-05']

Slicing time series

```
In [1]: temperature = weather['Temperature']

In [2]: march_apr = temperature['2010-03':'2010-04'] # data of
   ...: March & April 2010 only

In [3]: march_apr.shape
Out[3]: (1463,)

In [4]: march_apr.iloc[-4:] #extract last 4 entries from time
series
Out[4]:
Date
2010-04-30 20:00:00    73.3
2010-04-30 21:00:00    71.3
2010-04-30 22:00:00    69.7
2010-04-30 23:00:00    68.5
Name: Temperature, dtype: float64
```

Plotting time series slices



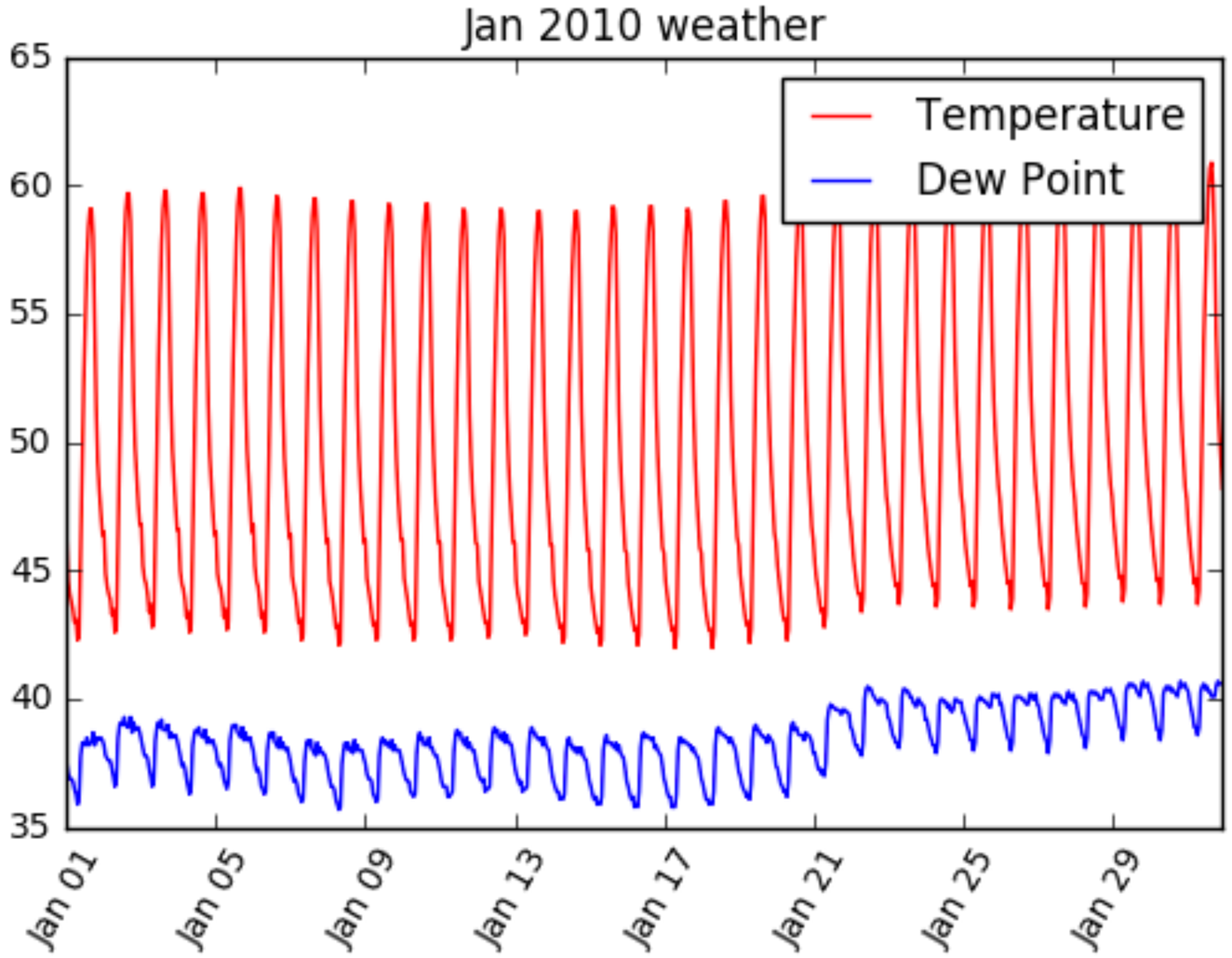
Plotting time series slices

```
In [1]: plt.plot(temperature['2010-01'], color='red',
   ...:          label='Temperature')

In [2] dew point = weather['DewPoint']

In [3]: plt.plot(dewpoint['2010-01'], color='blue',
   ...:          label='Dewpoint')

In [4]: plt.legend(loc='upper right')

In [5]: plt.xticks(rotation=60)

In [6]: plt.show()
```

Selecting & formatting dates

```
In [1]: jan = temperature['2010-01']

In [2]: dates = jan.index[::96]  # Pick every 4th day

In [3]: print(dates)
DatetimeIndex([‘2010-01-01’, ‘2010-01-05’, ‘2010-01-09’,
‘2010-01-13’,‘2010-01-17’, ‘2010-01-21’, ‘2010-01-25’,
‘2010-01-29’], dtype=‘datetime64[ns]’, name=‘Date’, freq=None)

In [4]: labels = dates.strftime('%b %d') # Make formatted labels

In [5]: print(labels)
[‘Jan 01’ ‘Jan 05’ ‘Jan 09’ ‘Jan 13’ ‘Jan 17’ ‘Jan 21’ ‘Jan 25’
‘Jan 29’]
```

Cleaning up ticks on axis



Cleaning up ticks on axis

```
In [1]: plt.plot(temperature['2010-01'], color='red',
   ...:          label='Temperature')

In [2]: plt.plot(dewpoint['2010-01'], color='blue',
   ...:          label='Dewpoint')

In [3]: plt.xticks(dates, labels, rotation=60)

In [4]: plt.legend(loc='upper right')

In [5]: plt.show()
```

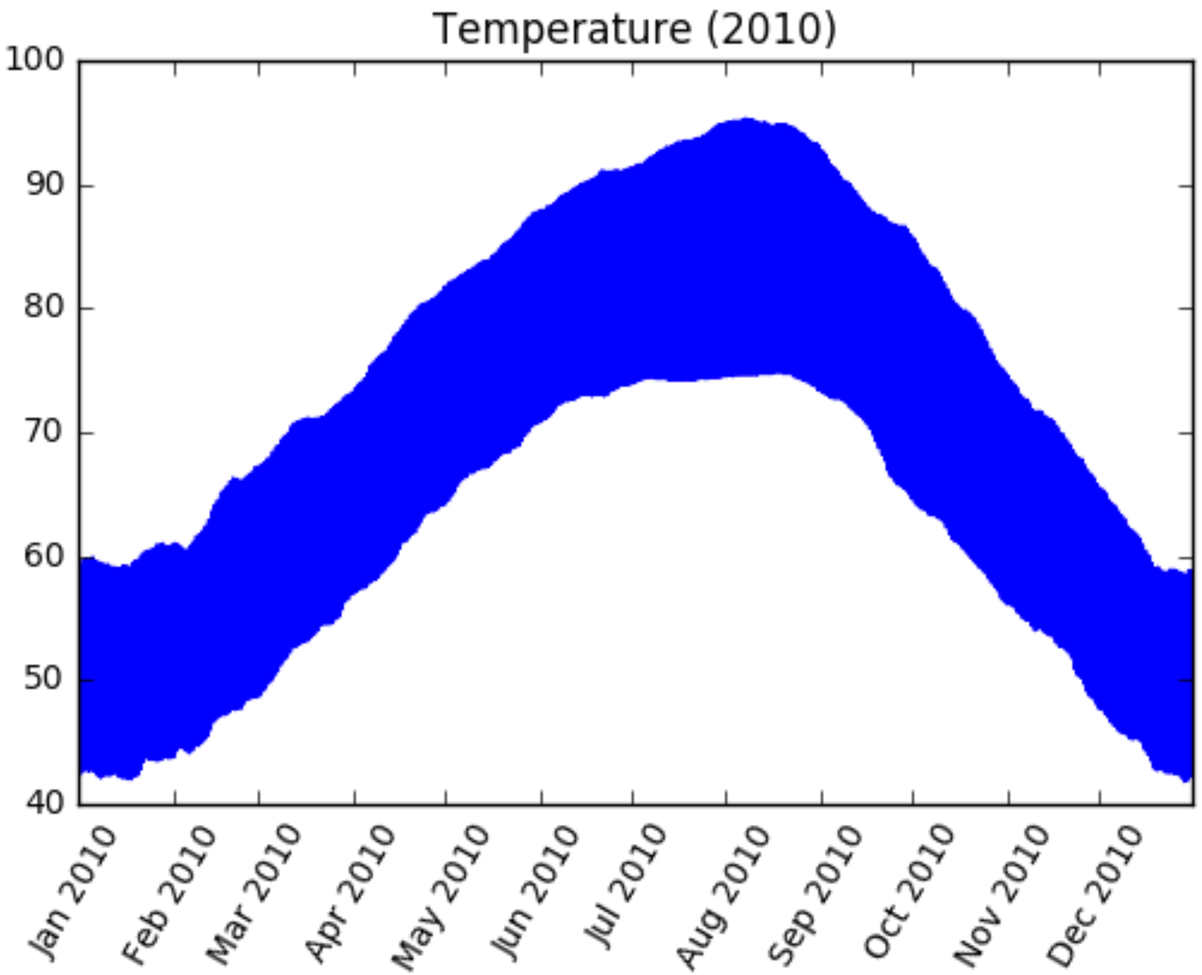
INTRODUCTION TO DATA VISUALIZATION WITH PYTHON

Let’s practice!

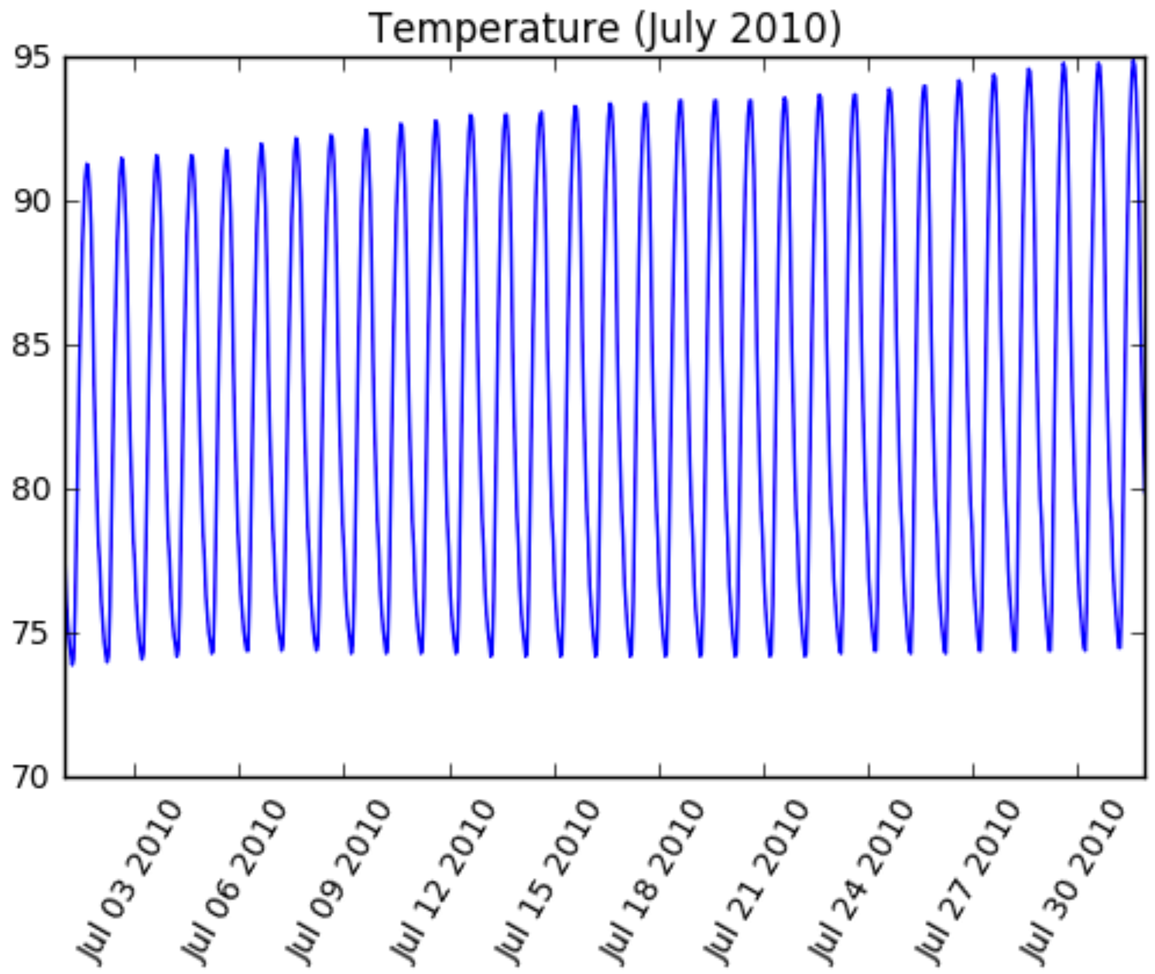
INTRODUCTION TO DATA VISUALIZATION WITH PYTHON

Time series with moving windows

Hourly data over a year



Zooming in


Temperature (July 2010)
95
90
85
80
75
70
Jul 03 2010
Jul 06 2010
Jul 09 2010
Jul 12 2010
Jul 15 2010
Jul 18 2010
Jul 21 2010
Jul 24 2010
Jul 27 2010
Jul 30 2010

Moving windows & time series

- Moving window calculations
  - Averages
  - Medians
  - Standard deviations
- Extracts information on longer time scales
- See Pandas courses on how to compute

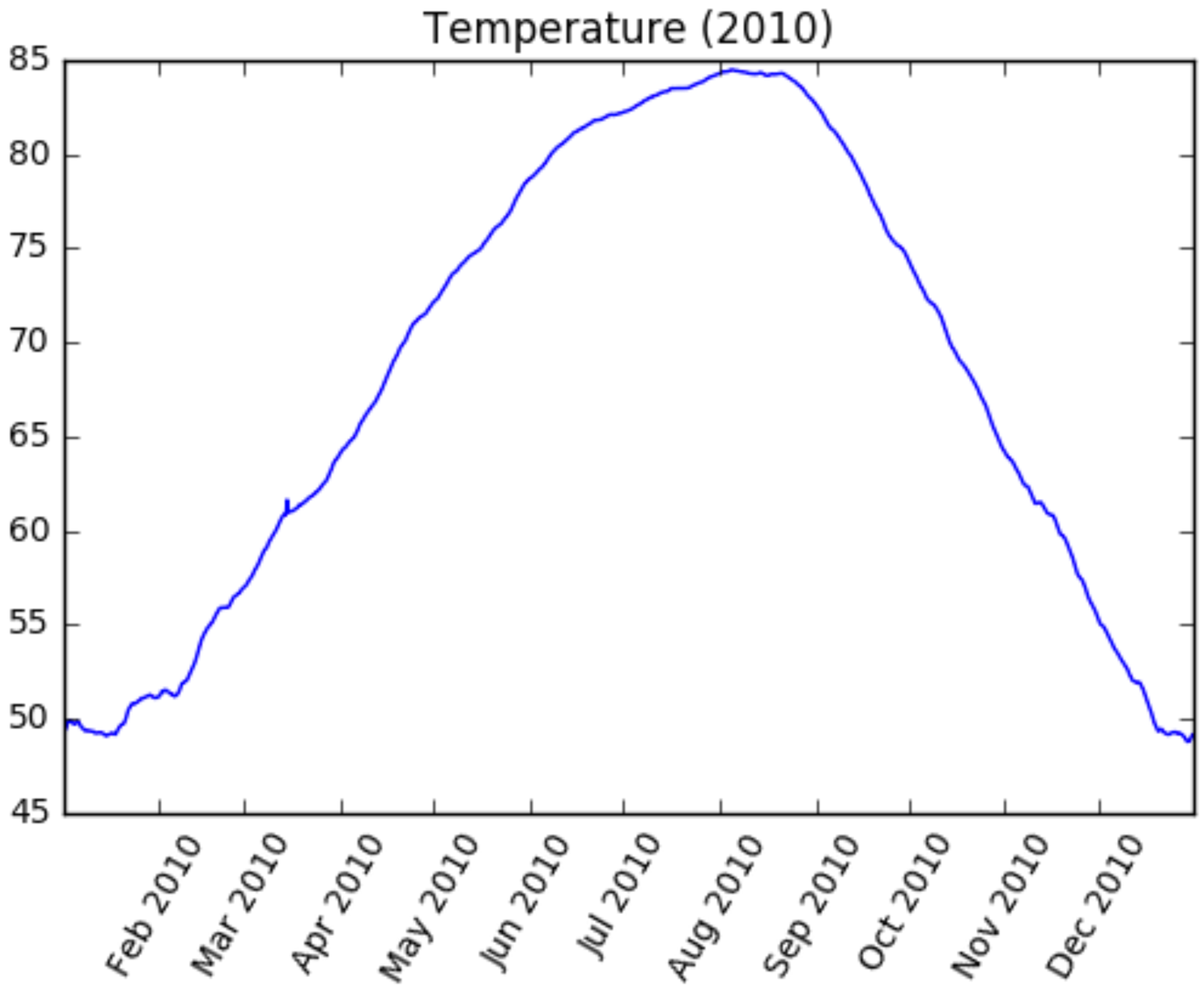
Moving averages

```
In [1]: smoothed.info() # smoothed computing using moving averages
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 8759 entries, 2010-01-01 00:00:00 to 2010-12-31
23:00:00
Data columns (total 5 columns):
14d             8424 non-null float64
1d              8736 non-null float64
3d              8688 non-null float64
7d              8592 non-null float64
dtypes: float64(5)
memory usage: 410.6 KB

In [2]: print(smoothed.iloc[:3,:])
                     14d  1d  3d  7d  Temperature
Date
2010-01-01 00:00:00  NaN NaN NaN NaN         46.2
2010-01-01 01:00:00  NaN NaN NaN NaN         44.6
2010-01-01 02:00:00  NaN NaN NaN NaN         44.1
```

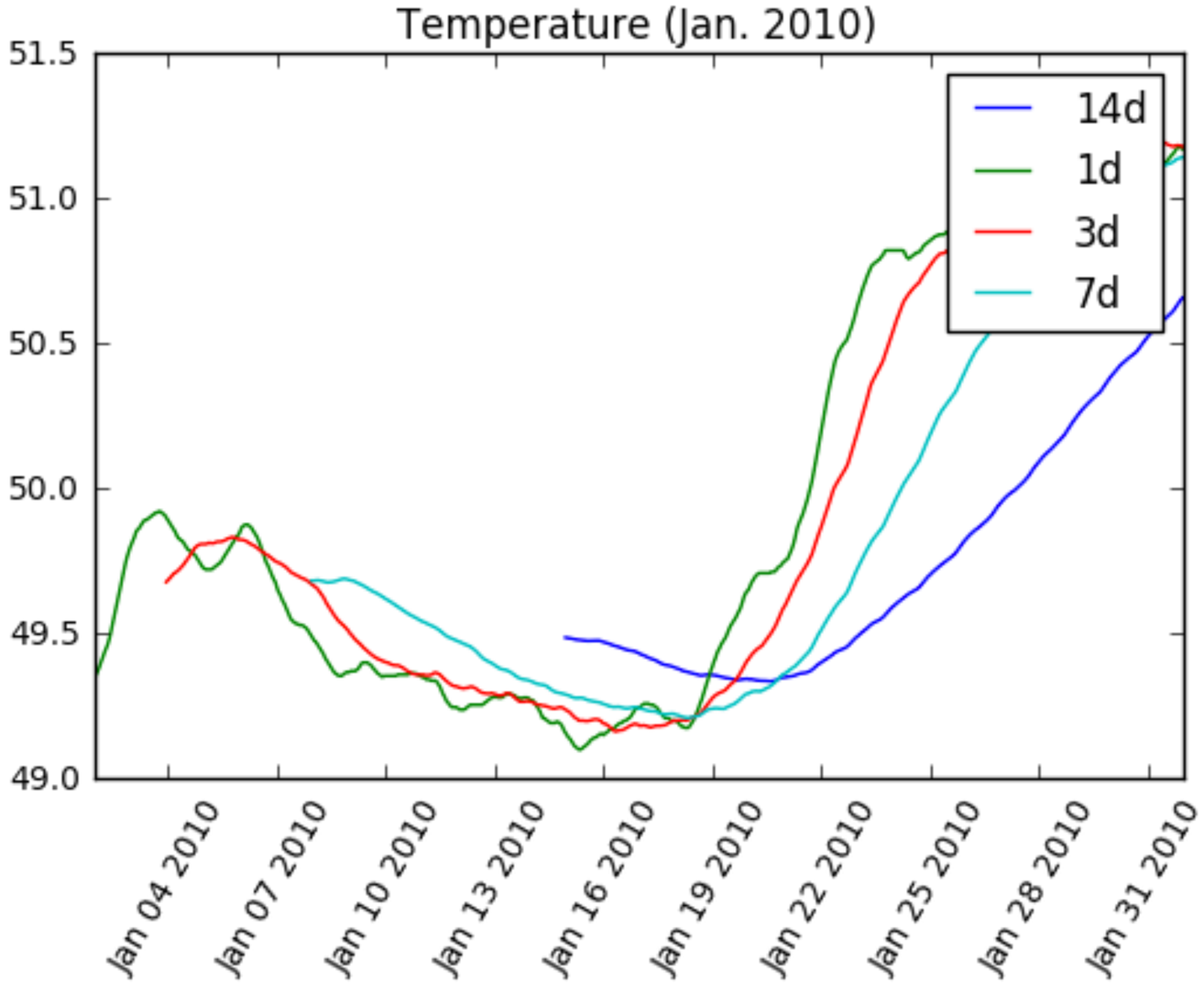
Viewing 24 hour averages

```
In [1]: plt.plot(smoothed['1d']) # moving average over 24 hours

In [2]: plt.title('Temperature (2010)')

In [3]: plt.xticks(rotation=60)

In [4]: plt.show()
```

Viewing 24 hour averages



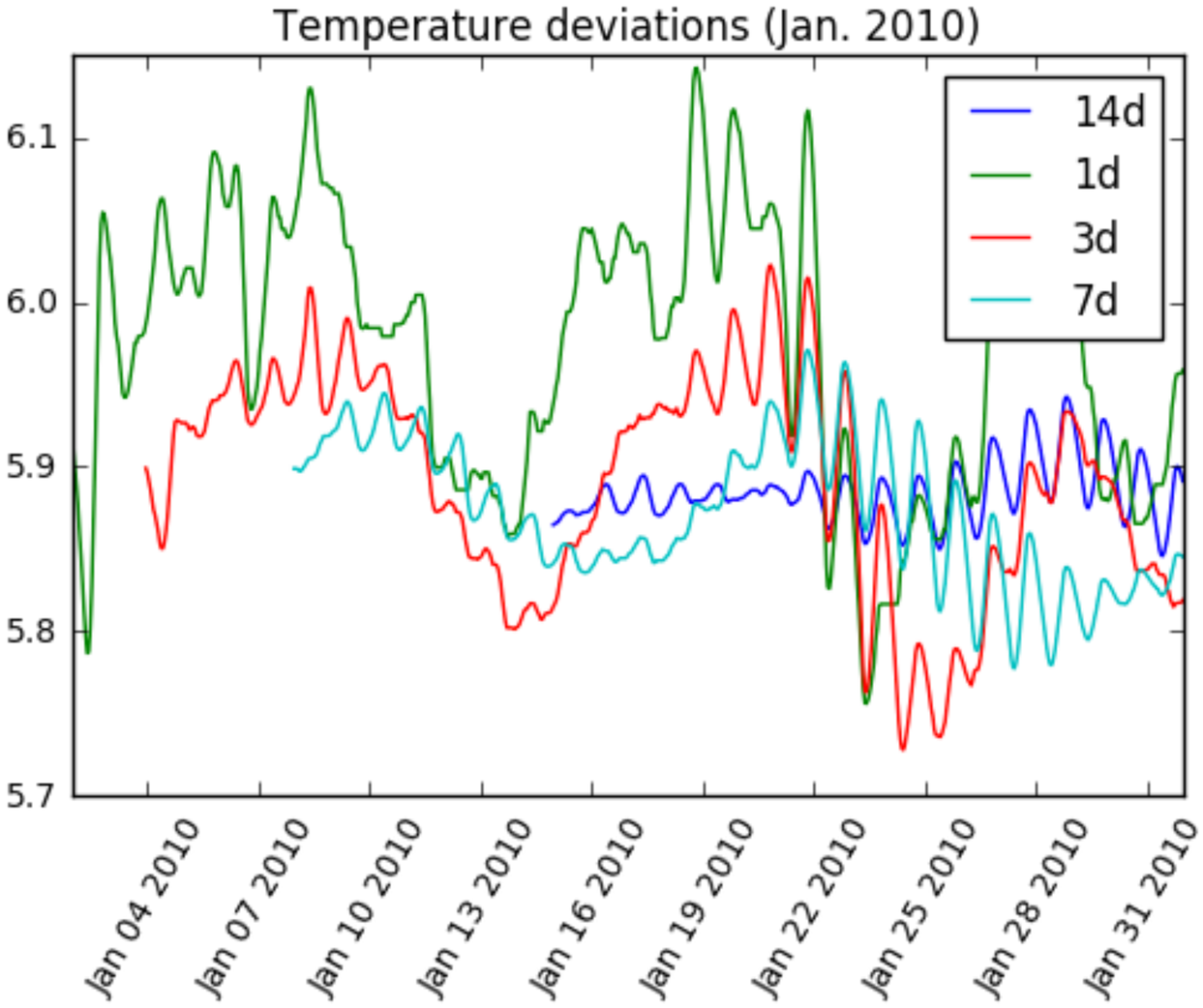
Viewing all moving averages



Viewing all moving averages

```
In [1]: plt.plot(smoothed['2010-01']) # plot
   ...: DataFrame for January

In [2]: plt.legend(smoothed.columns)

In [3]: plt.title('Temperature (Jan. 2010)')

In [4]: plt.xticks(rotation=60)

In [5]: plt.show()
```

Moving standard deviations



Moving standard deviations

```
In [1]: plt.plot(variances['2010-01'])

In [2]: plt.legend(variances.columns)

In [3]: plt.title('Temperature deviations (Jan. 2010)')

In [4]: plt.xticks(rotation=60)

In [5]: plt.show()
```

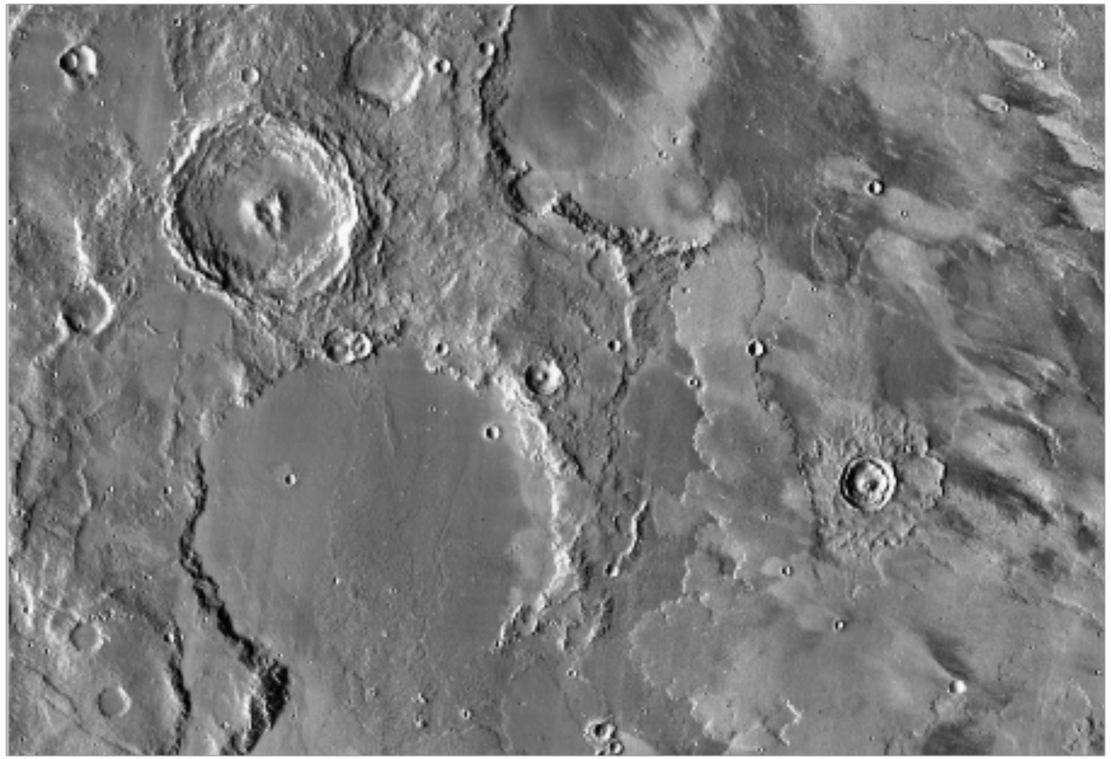
INTRODUCTION TO DATA VISUALIZATION WITH PYTHON

Let’s practice!

INTRODUCTION TO DATA VISUALIZATION WITH PYTHON

Histogram equalization in images

Original image



Equalized image

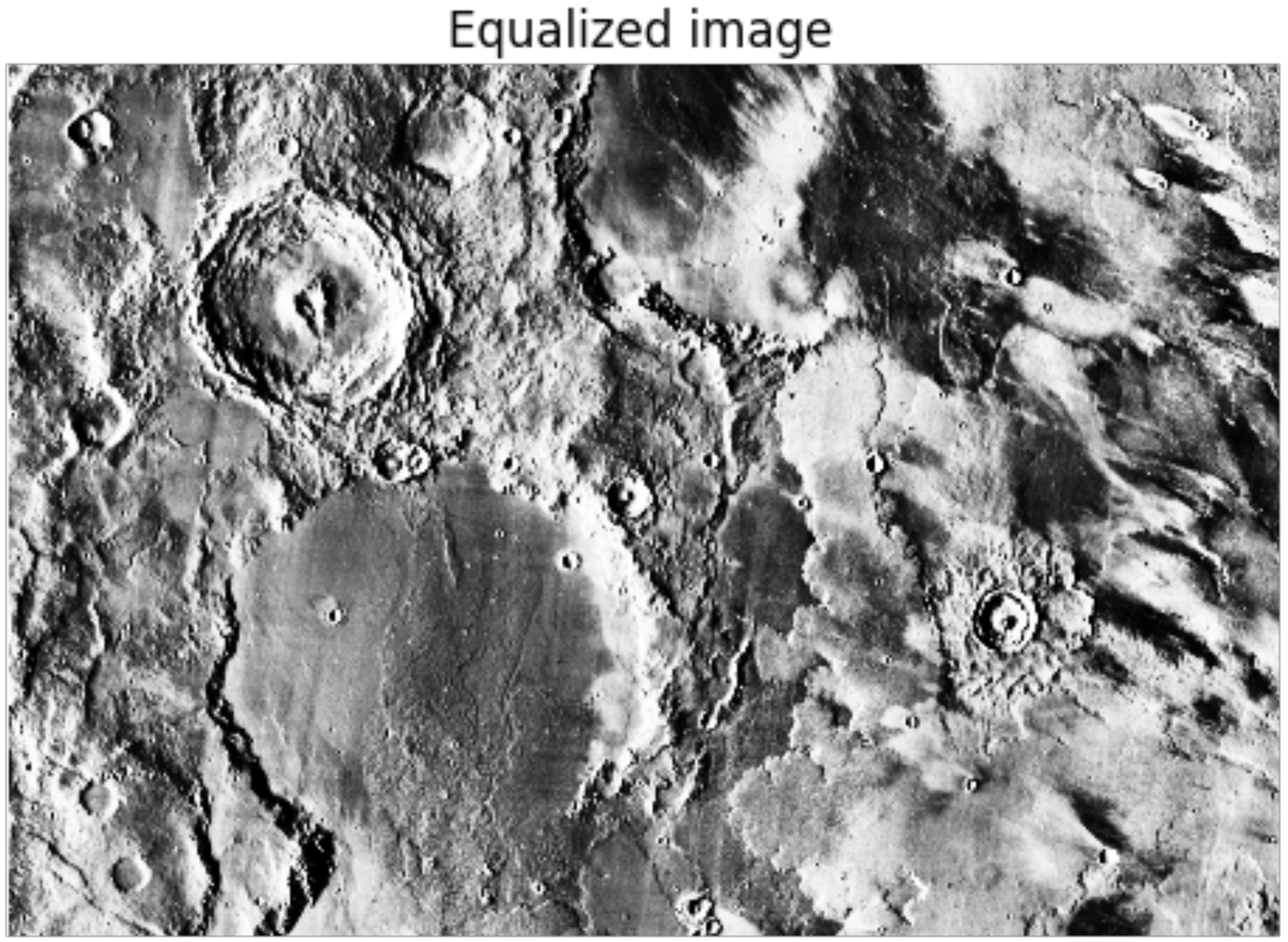


Image histograms

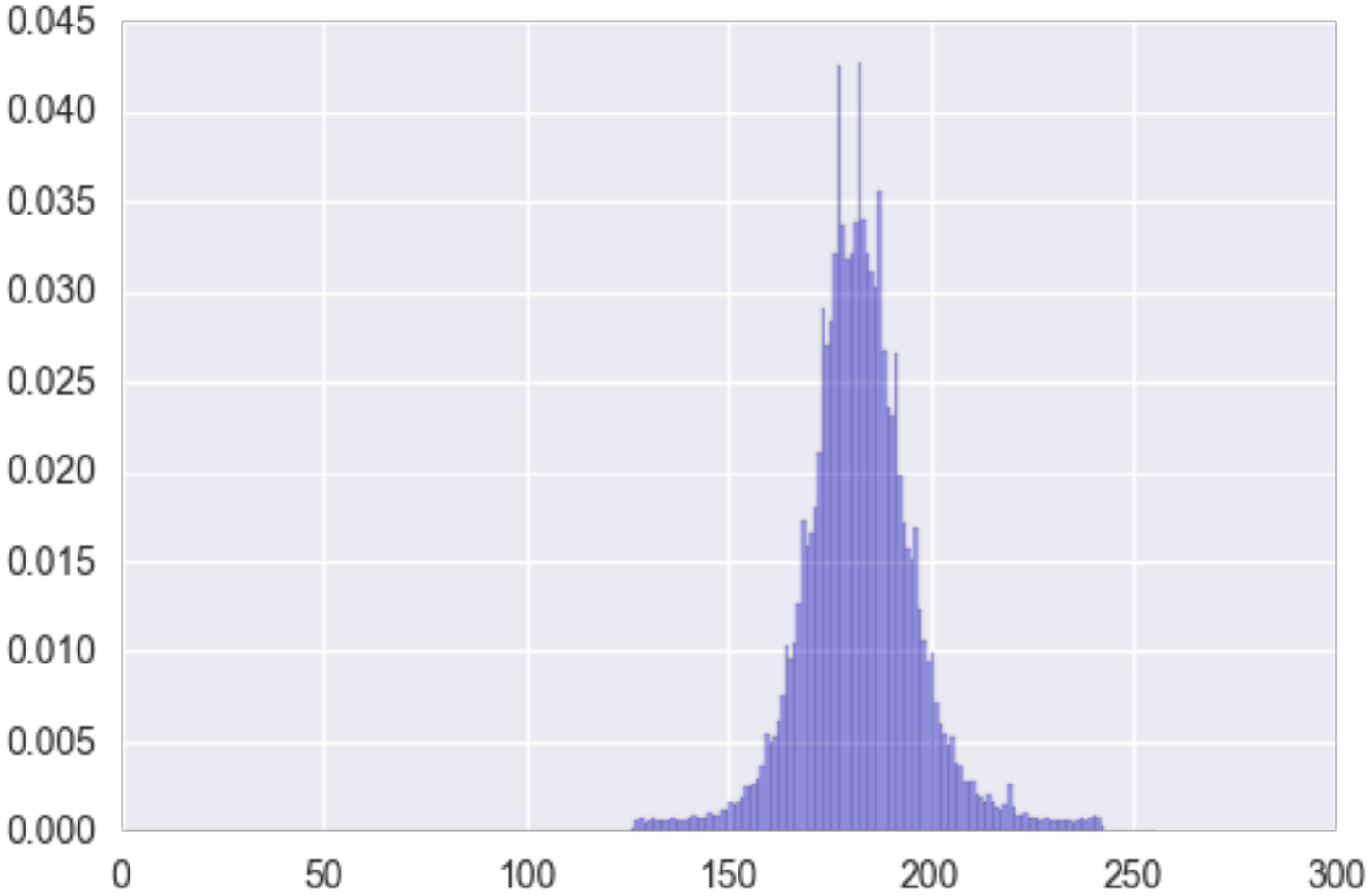


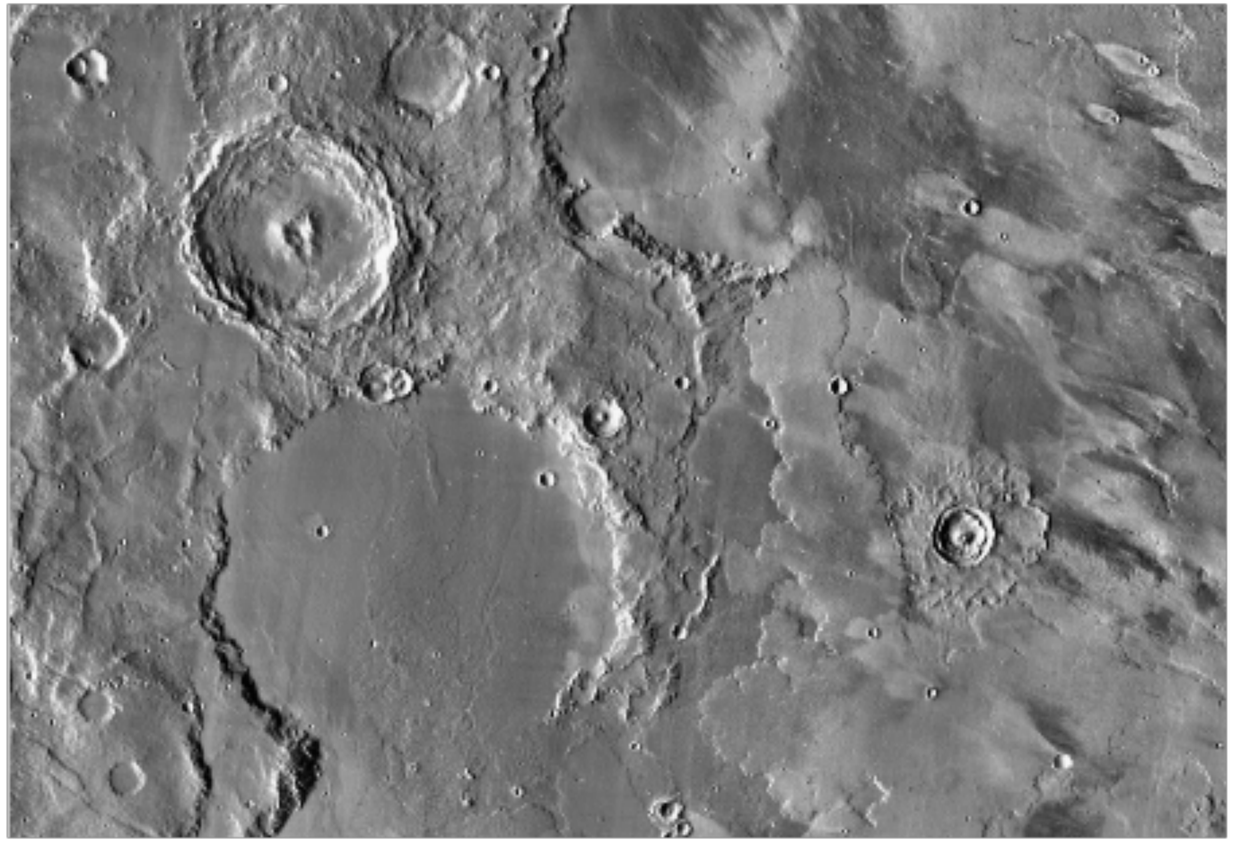
Image histograms

```
In [1]: orig = plt.imread('low-contrast-moon.jpg')

In [2]: pixels = orig.flatten()

In [3]: plt.hist(pixels, bins=256, range=(0,256), normed=True,
   ...:          color='blue', alpha=0.3)

In [4]: plt.show()

In [5]: minval, maxval = orig.min(), orig.max()

In [6]: print(minval, maxval)
125 244
```

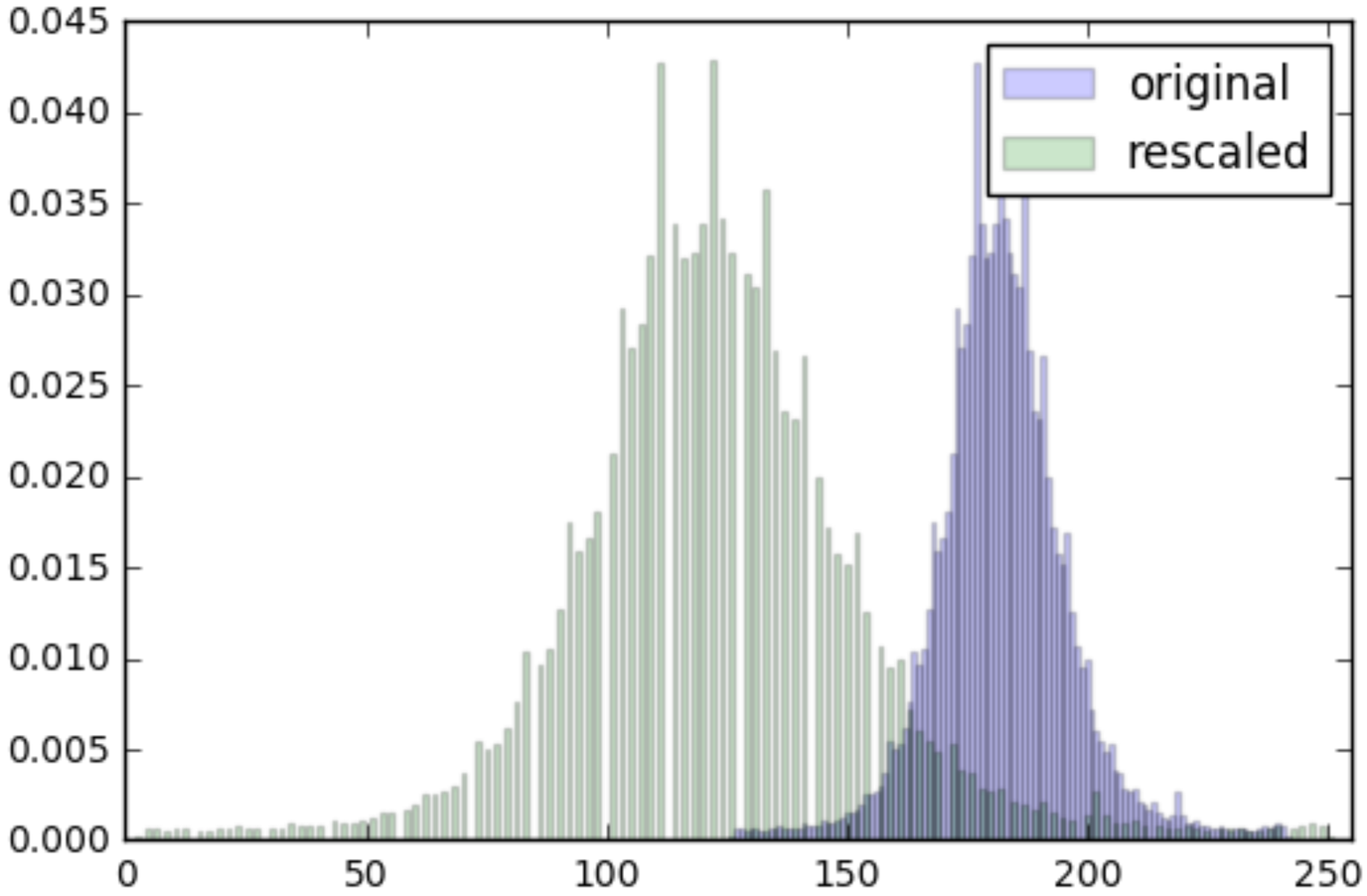
Rescaling the image

```
In [1]: minval, maxval = orig.min(), orig.max()

In [2]: print(minval, maxval)
125 244

In [3]: rescaled = (255/(maxval-minval)) * (pixels - minval)

In [4]: print(rescaled.min(), rescaled.max())
0.0 255.0

In [5]: plt.imshow(rescaled)

In [6]: plt.axis('off')

In [7]: plt.show()
```

Rescaled image



Original & rescaled histograms



Original & rescaled histograms

```
In [1]: plt.hist(orig.flatten(), bins=256, range=(0,255),
   ...:          normed=True, color='blue', alpha=0.2))

In [2]: plt.hist(rescaled.flatten(), bins=256, range=(0,255),
   ...:          normed=True, color='green', alpha=0.2))

In [3]: plt.legend(['original', 'rescaled'])

In [4]: plt.show()
```

Image histogram & CDF

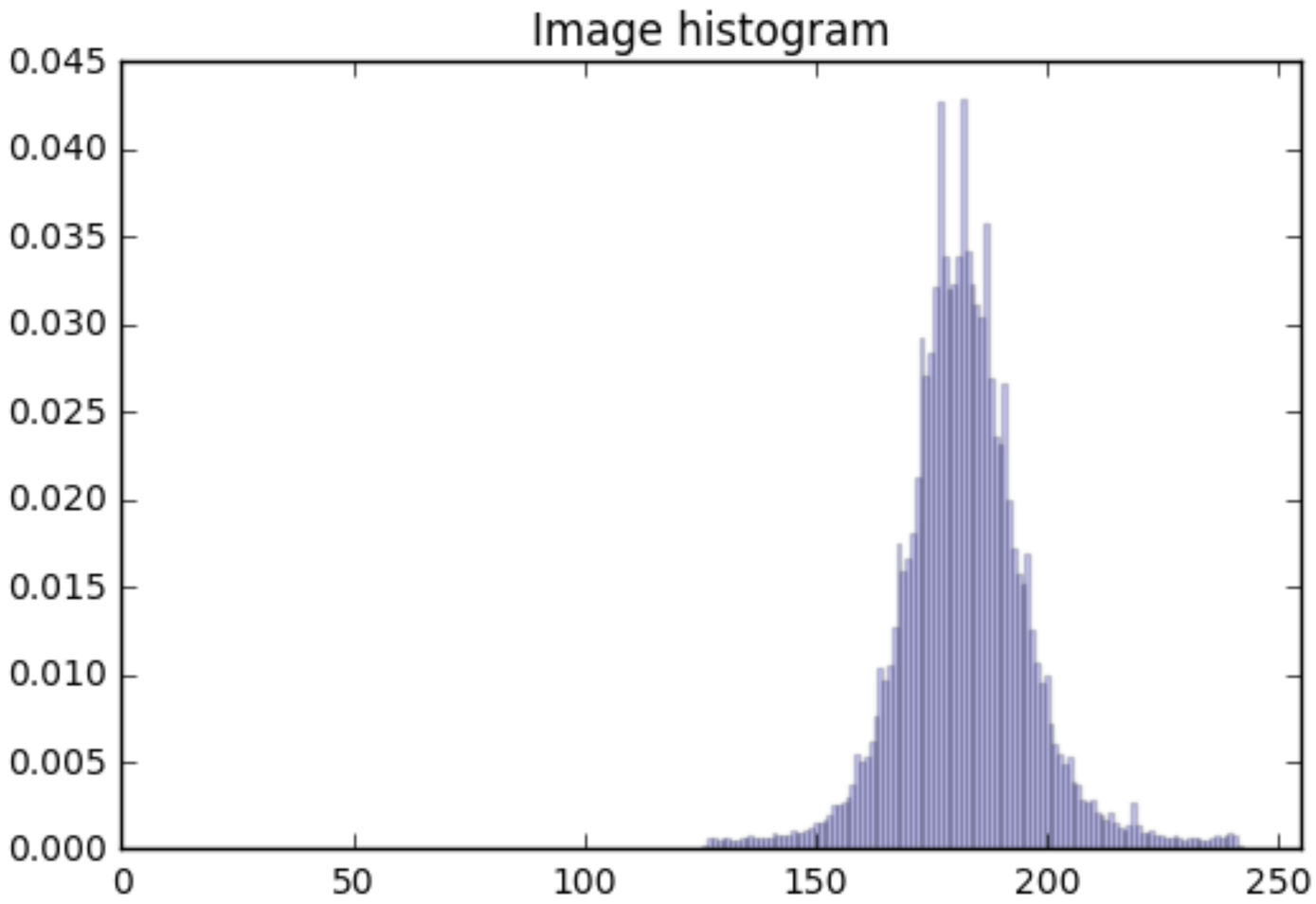


Image histogram & CDF

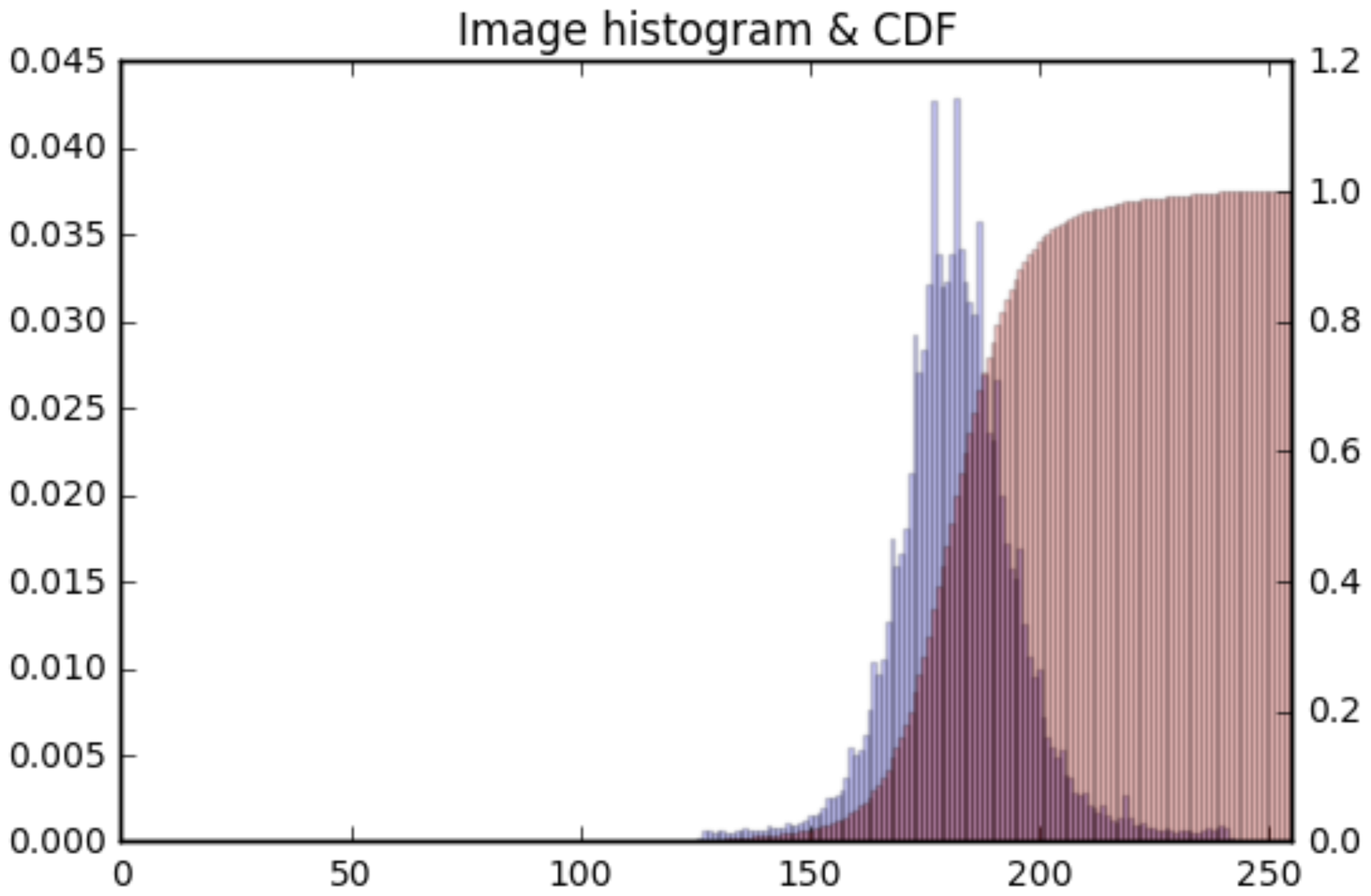


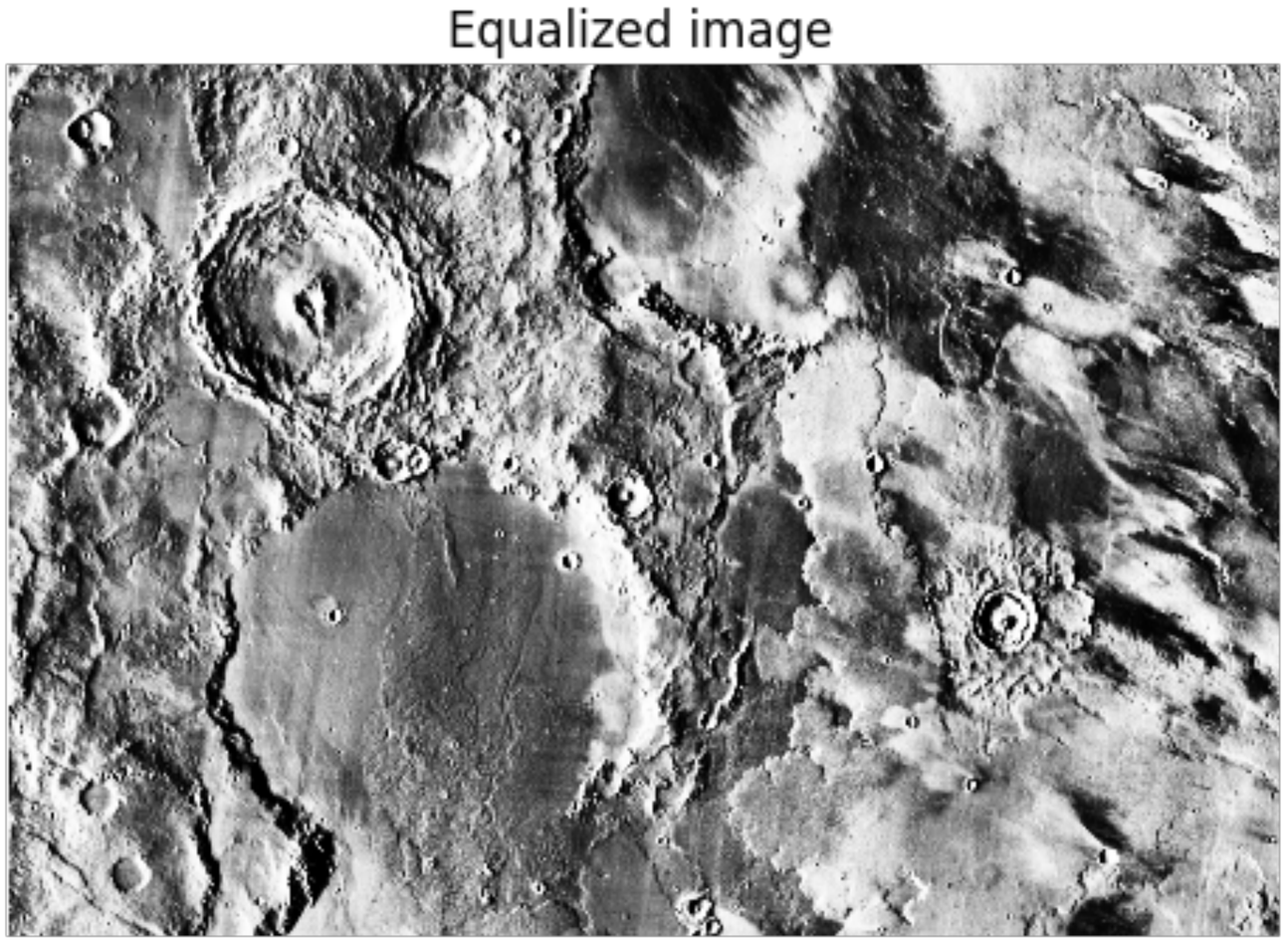
Image histogram & CDF

```
In [1]: plt.hist(pixels, bins=256, range=(0,256), normed=True,
   ...:          color='blue', alpha=0.3)

In [2]: plt.twinx()

In [3]: orig_cdf, bins, patches = plt.hist(pixels,
   ...: cumulative=True, bins=256, range=(0,256), normed=True,
   ...: color='red', alpha=0.3)

In [4]: plt.title('Image histogram and CDF')

In [5]: plt.xlim((0, 255))

In [6]: plt.show()
```

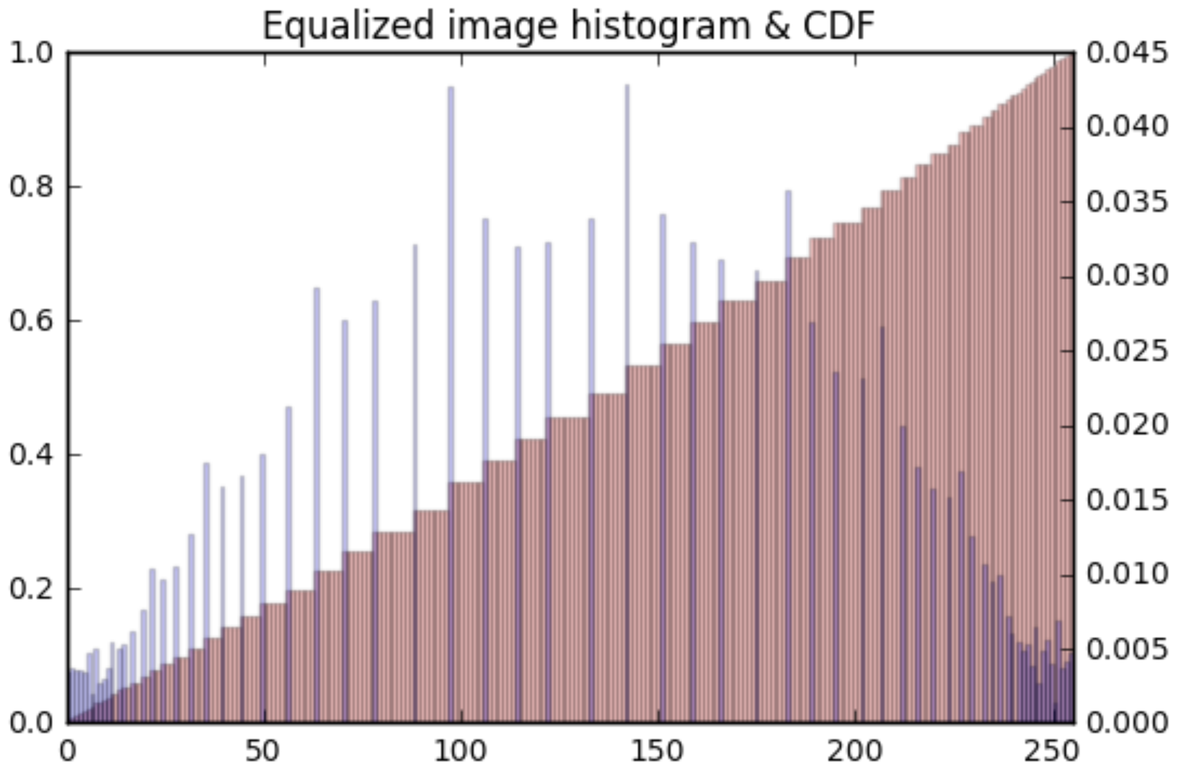
Equalizing intensity values

```
In [1]: new_pixels =  np.interp(pixels, bins[:-1], orig_cdf*255)

In [2]: new = new_pixels.reshape(orig.shape)

In [3]: plt.imshow(new)

In [4]: plt.axis('off')

In [5]: plt.title('Equalized image')

In [6]: plt.show()
```

Equalized image



Equalized histogram & CDF



Equalized histogram & CDF

```
In [1]: plt.hist(new_pixels, bins=256, range=(0,256),
   ...:          normed=True, color='blue', alpha=0.3)

In [2]: plt.twinx()

In [3]: plt.hist(new_pixels, cumulative=True, bins=256,
   ...:          range=(0,256), normed=True, color='red', alpha=0.1)

In [4]: plt.title('Equalized image histogram and CDF')

In [5]: plt.xlim((0, 255))

In [6]: plt.show()
```

INTRODUCTION TO DATA VISUALIZATION WITH PYTHON

Let’s practice!

INTRODUCTION TO DATA VISUALIZATION WITH PYTHON

Congratulations!