linear_solutions

September 29, 2024

1 Linear Fit Solutions

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import gridspec
%matplotlib inline
from scipy.optimize import curve_fit
import os
```

Directory housekeeping

[2]:
```
basedir = '/home/david/gh/intro_curve_fitting_python'

# literally:
try:
    os.chdir(basedir)
# if there is an exception ('error'):
except:
    print('\n\nproblem changing to the directory you specified; does it exist?
 ↪\nthe kernel will now restart; rerun this program.\n\n')
    quit()
```

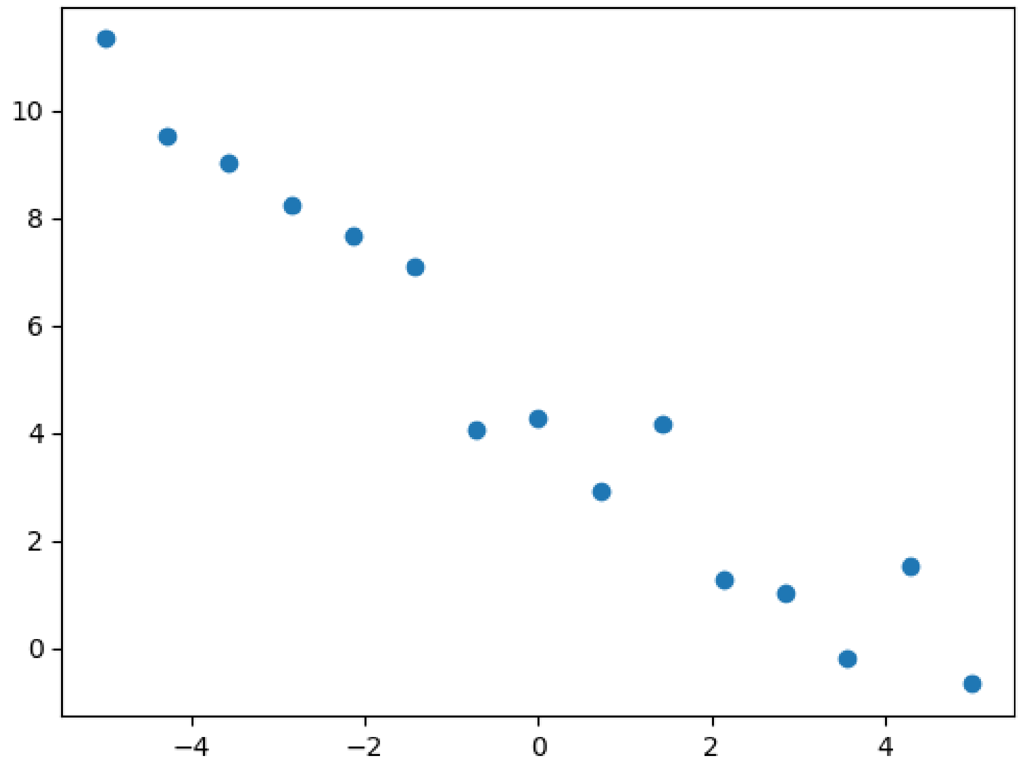
Define the fitting function

[3]:
```
def sl(x, m, b):
    return m*x+b
```

Carry out the fits

[4]:
```
# dataset linear1.csv
fn = basedir+'/linear_data/linear1.csv'

x = []
y = []

inf = open(fn)

for line in inf:
    line = line.rstrip()
```

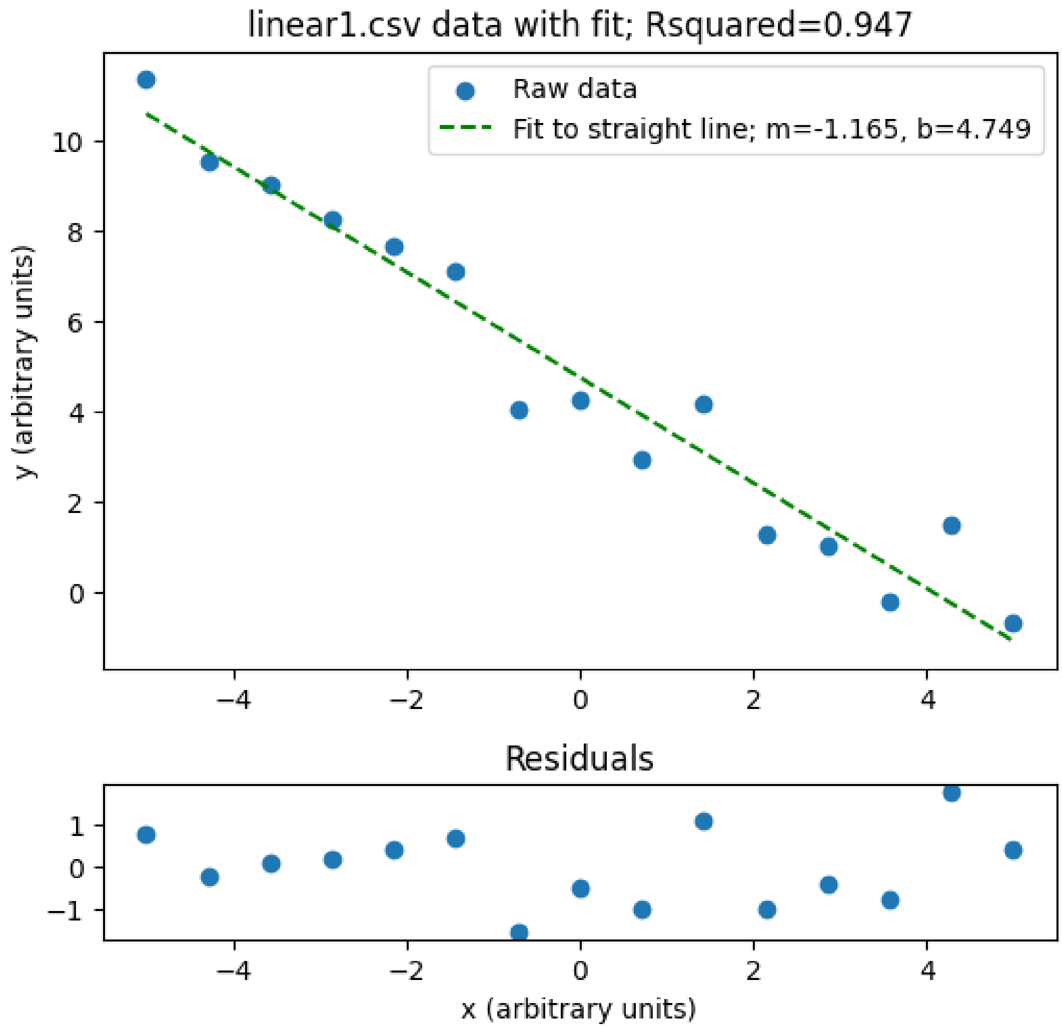
```
    la = line.split(',')
    x.append(float(la[0]))
    y.append(float(la[1]))

inf.close()

x=np.array(x)
y=np.array(y)

plt.scatter(x,y)
plt.show()
```



[5]:

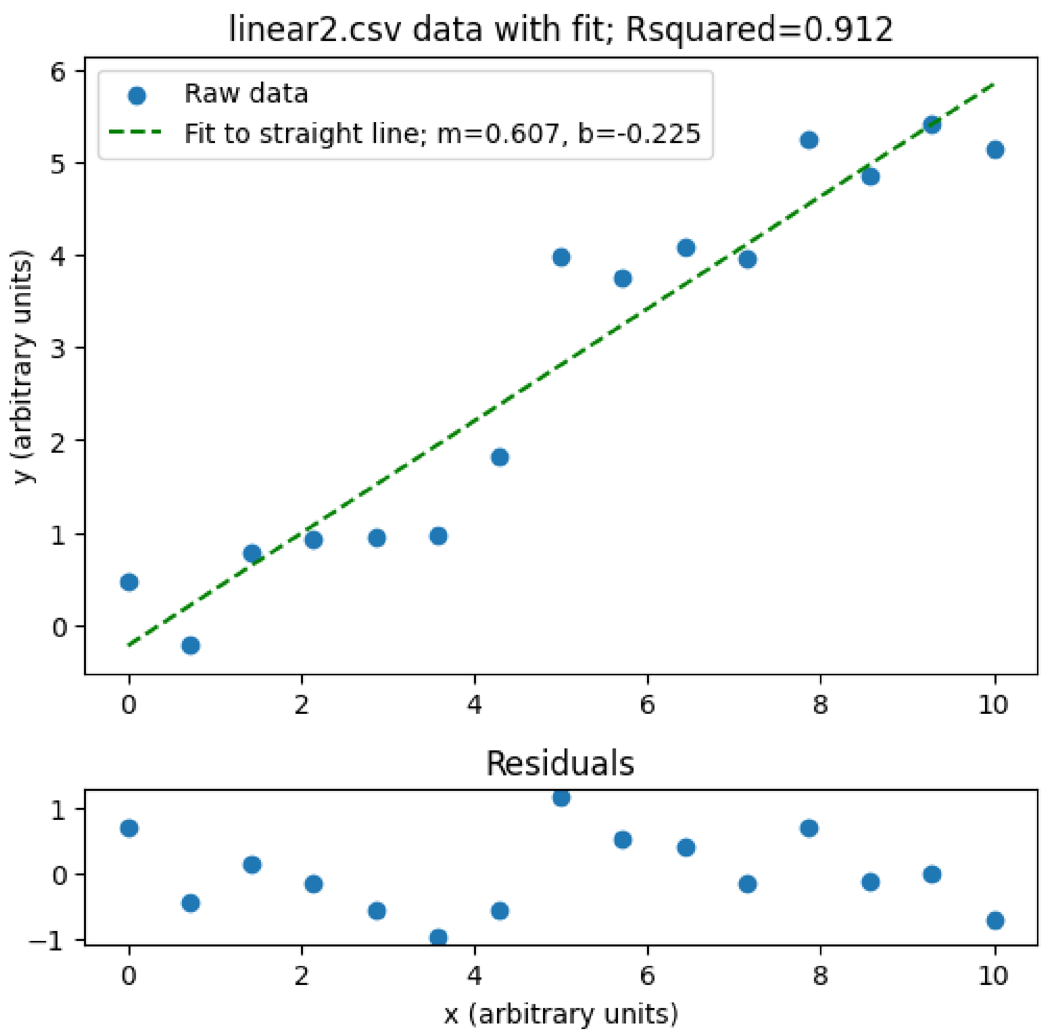
```
popt, pcov = curve_fit(sl, x, y)

residuals = y-sl(x, *popt)

rsq = 1 - np.sum(np.square(residuals))/np.sum(np.square(y-np.mean(y)))

fig = plt.figure()

fig.set_figwidth=(4)
fig.set_figheight(6)
```

```
spec = gridspec.GridSpec(ncols=1, nrows=2,
                                                hspace=0.3, height_ratios=[4, 1])

ax0 = fig.add_subplot(spec[0])
ax0.scatter(x,y, label='Raw data')
ax0.plot(x, sl(x, *popt), 'g--',
         label='Fit to straight line; m=%0.3f, b=%0.3f' % tuple(popt))
ax0.set_ylabel('y (arbitrary units)')
ax0.set_title('linear1.csv data with fit; Rsquared=%0.3f' % rsq)
ax0.legend()

ax1 = fig.add_subplot(spec[1])
ax1.set_title('Residuals')
ax1.set_xlabel('x (arbitrary units)')
ax1.scatter(x, residuals)

# display and save the figure
plt.show()
```



[6]:
```
# dataset linear2.csv
fn = basedir+'/linear_data/linear2.csv'

x = []
y = []

inf = open(fn)

for line in inf:
    line = line.rstrip()
    la = line.split(',')
    x.append(float(la[0]))
    y.append(float(la[1]))

inf.close()
```

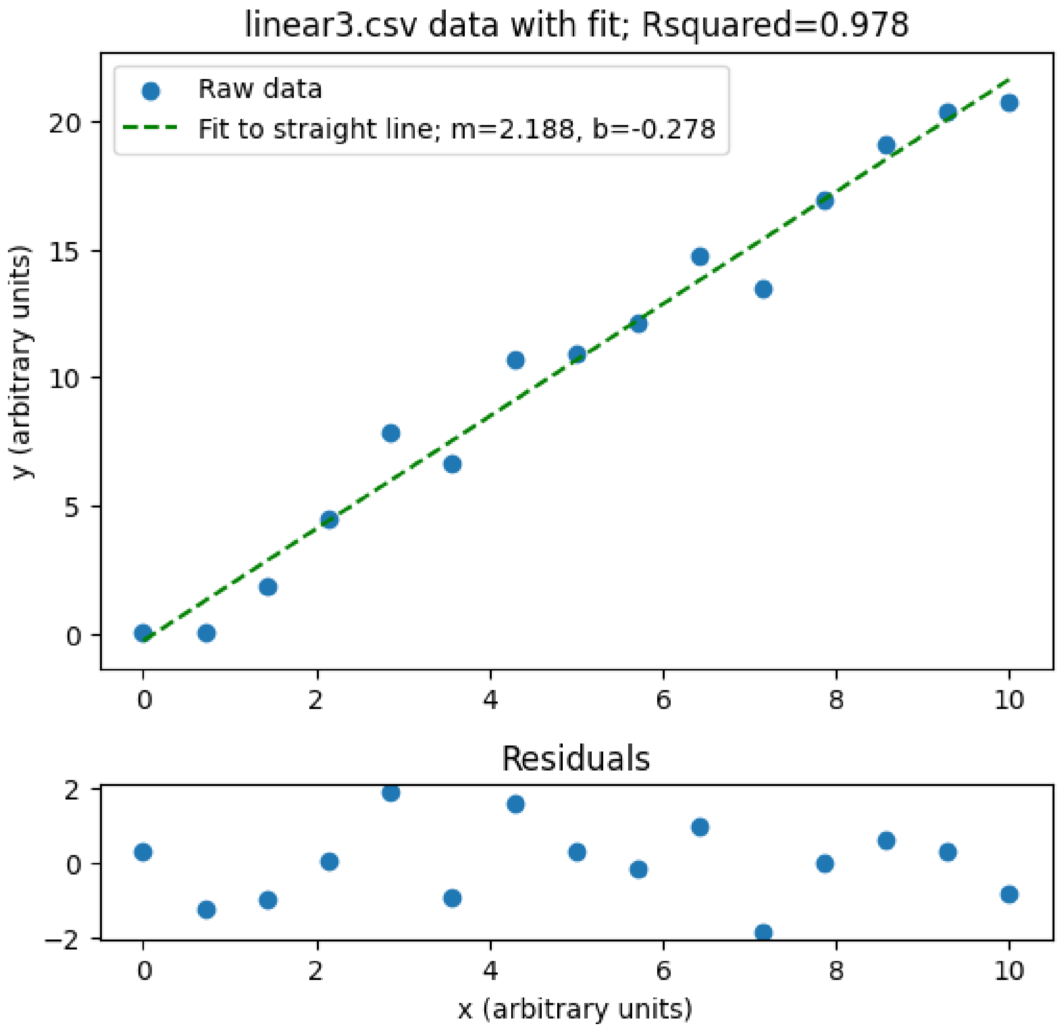
```
x=np.array(x)
y=np.array(y)

popt, pcov = curve_fit(sl, x, y)

residuals = y-sl(x, *popt)

rsq = 1 - np.sum(np.square(residuals))/np.sum(np.square(y-np.mean(y)))

fig = plt.figure()

fig.set_figwidth=(4)
fig.set_figheight(6)

spec = gridspec.GridSpec(ncols=1, nrows=2,
                                                hspace=0.3, height_ratios=[4,␣
 ↪1])

ax0 = fig.add_subplot(spec[0])
ax0.scatter(x,y, label='Raw data')
ax0.plot(x, sl(x, *popt), 'g--',
         label='Fit to straight line; m=%0.3f, b=%0.3f' % tuple(popt))
ax0.set_ylabel('y (arbitrary units)')
ax0.set_title('linear2.csv data with fit; Rsquared=%0.3f' % rsq)
ax0.legend()

ax1 = fig.add_subplot(spec[1])
ax1.set_title('Residuals')
ax1.set_xlabel('x (arbitrary units)')
ax1.scatter(x, residuals)

# display and save the figure
plt.show()
```



[7]:

```
# dataset linear3.csv
fn = basedir+'/linear_data/linear3.csv'

x = []
y = []

inf = open(fn)

for line in inf:
    line = line.rstrip()
    la = line.split(',')
    x.append(float(la[0]))
    y.append(float(la[1]))

inf.close()
```

```
x=np.array(x)
y=np.array(y)

popt, pcov = curve_fit(sl, x, y)

residuals = y-sl(x, *popt)

rsq = 1 - np.sum(np.square(residuals))/np.sum(np.square(y-np.mean(y)))

fig = plt.figure()

fig.set_figwidth=(4)
fig.set_figheight(6)

spec = gridspec.GridSpec(ncols=1, nrows=2,
                                                hspace=0.3, height_ratios=[4,␣
 ↪1])

ax0 = fig.add_subplot(spec[0])
ax0.scatter(x,y, label='Raw data')
ax0.plot(x, sl(x, *popt), 'g--',
         label='Fit to straight line; m=%0.3f, b=%0.3f' % tuple(popt))
ax0.set_ylabel('y (arbitrary units)')
ax0.set_title('linear3.csv data with fit; Rsquared=%0.3f' % rsq)
ax0.legend()

ax1 = fig.add_subplot(spec[1])
ax1.set_title('Residuals')
ax1.set_xlabel('x (arbitrary units)')
ax1.scatter(x, residuals)

# display and save the figure
plt.show()
```



Evidently, it is straightforward to carry out linear fitting.