

# FittingLines\_i

May 11, 2020

```
[46]: import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.optimize import curve_fit

%matplotlib inline
plt.style.use(["fivethirtyeight", "seaborn-paper"])

[4]: # you will most probably have to change this to correctly read in your data
import pandas as pd
table = pd.read_csv(
    "/Users/alix/Desktop/NEP2581_i.cat",
    sep="\s+",
    comment="#",
    index_col=0,
    header="infer",
    names=[
        "Number",
        "MagIso",
        "MagErrIso",
        "MagIsoCor",
        "MagErrIsoCor",
        "MagAper1",
        "MagAper2",
        "MagAper3",
        "MagErrAper1",
        "MagErrAper2",
        "MagErrAper3",
        "FluxAuto",
        "FluxErrAuto",
        "MagAuto",
        "MagErrAuto",
        "X",
        "Y",
        "RA",
        "DEC",
        "Flags",
```

```
] ,  
)
```

```
-----  
FileNotFoundError                                Traceback (most recent call  
last)
```

```
<ipython-input-4-a47b977dcc0d> in <module>  
27         "RA",  
28         "DEC",  
---> 29         "Flags",  
30     ],  
31 )
```

```
~/opt/anaconda3/lib/python3.7/site-packages/pandas/io/parsers.py in  
_parser_f(filepath_or_buffer, sep, delimiter, header, names, index_col,  
usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine, converters,  
true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows,  
na_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates,  
infer_datetime_format, keep_date_col, date_parser, dayfirst, cache_dates,  
iterator, chunksize, compression, thousands, decimal, lineterminator,  
quotechar, quoting, doublequote, escapechar, comment, encoding, dialect,  
error_bad_lines, warn_bad_lines, delim_whitespace, low_memory, memory_map,  
float_precision)  
683     )  
684  
--> 685     return _read(filepath_or_buffer, kwds)  
686  
687     parser_f.__name__ = name
```

```
~/opt/anaconda3/lib/python3.7/site-packages/pandas/io/parsers.py in  
_read(filepath_or_buffer, kwds)  
455  
456     # Create the parser.  
--> 457     parser = TextFileReader(fp_or_buf, **kwds)  
458  
459     if chunksize or iterator:
```

```
~/opt/anaconda3/lib/python3.7/site-packages/pandas/io/parsers.py in  
__init__(self, f, engine, **kwds)  
893         self.options["has_index_names"] = kwds["has_index_names"]
```

```

894
--> 895         self._make_engine(self.engine)
896
897     def close(self):

~/opt/anaconda3/lib/python3.7/site-packages/pandas/io/parsers.py in _
↪ _make_engine(self, engine)
    1133     def _make_engine(self, engine="c"):
    1134         if engine == "c":
-> 1135             self._engine = CParserWrapper(self.f, **self.options)
    1136         else:
    1137             if engine == "python":

~/opt/anaconda3/lib/python3.7/site-packages/pandas/io/parsers.py in _
↪ __init__(self, src, **kwds)
    1915         kwds["usecols"] = self.usecols
    1916
-> 1917         self._reader = parsers.TextReader(src, **kwds)
    1918         self.unnamed_cols = self._reader.unnamed_cols
    1919

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.
↪ _setup_parser_source()

FileNotFoundError: [Errno 2] File b'/Users/alix/Desktop/NEP2581_i.cat'
↪ does not exist: b'/Users/alix/Desktop/NEP2581_i.cat'

```

[137]: table

```

[137]:
      MagIso  MagErrIso  MagIsoCor  MagErrIsoCor  MagAper1  MagAper2  \
Number
1      -13.3767    0.0022   -13.3908         0.0023   -12.2390   -12.8005
2      -9.4826    0.0215    -9.6280         0.0286    -8.4765    -9.0611
3      -8.8818    0.0382    -9.4492         0.0550    -7.1400    -7.8576
4      -8.1896    0.0528    -8.7703         0.0757    -6.9821    -7.7208
5     -11.5036    0.0066   -11.5520         0.0075   -10.3737   -10.8996
...      ...      ...      ...      ...      ...      ...
212    -5.9144    0.1639    -6.8463         0.2006    -6.2306    -6.7739
213    -7.6993    0.0682    -8.3779         0.0944    -6.7552    -7.5006

```

214	-11.7319	0.0055	-11.7652	0.0060	-10.5481	-11.1445
215	-9.5718	0.0207	-9.7192	0.0276	-8.4803	-9.0633
216	-12.5467	0.0036	-12.5723	0.0038	-11.4433	-11.9824

	MagAper3	MagErrAper1	MagErrAper2	MagErrAper3	FluxAuto	\
Number						
1	-13.0661	0.0030	0.0023	0.0021	218483.1000	
2	-9.3449	0.0256	0.0213	0.0208	7008.0870	
3	-8.3640	0.0733	0.0562	0.0467	6151.5810	
4	-8.1035	0.0840	0.0633	0.0586	3906.4750	
5	-11.1428	0.0077	0.0062	0.0059	41285.4000	
...	...	...	...	...	...	
212	-7.1079	0.1625	0.1458	0.1420	660.1039	
213	-7.9312	0.1020	0.0766	0.0681	2045.5990	
214	-11.4207	0.0070	0.0054	0.0050	49678.2000	
215	-9.3752	0.0254	0.0212	0.0203	7523.8860	
216	-12.2182	0.0044	0.0035	0.0032	101974.3000	

	FluxErrAuto	MagAuto	MagErrAuto	X	Y	RA	\
Number							
1	416.5467	-13.3485	0.0021	931.7031	328.2632	275.258573	
2	186.1411	-9.6140	0.0288	737.7443	295.1588	275.290740	
3	299.3925	-9.4725	0.0529	1457.0001	304.5081	275.171347	
4	261.6770	-8.9795	0.0727	1593.2390	299.5876	275.148727	
5	271.0937	-11.5395	0.0071	952.7084	312.8791	275.255071	
...	...	...	...	...	...	...	
212	112.9133	-7.0490	0.1858	1199.2676	1309.1158	275.214927	
213	155.5080	-8.2771	0.0826	634.0111	1243.7678	275.308951	
214	258.0924	-11.7404	0.0056	366.1382	1270.9669	275.353564	
215	171.6220	-9.6911	0.0248	545.5924	1225.6635	275.323646	
216	322.4636	-12.5212	0.0034	847.0334	1207.2085	275.273463	

	DEC	Flags
Number		
1	68.419623	2
2	68.417534	0
3	68.418325	2
4	68.418057	0
5	68.418690	0
...	...	...
212	68.479600	0
213	68.475419	0
214	68.476971	0
215	68.474279	0
216	68.473265	2

[216 rows x 19 columns]

$$R_{PS} = R_{SX} + ZP + c_0 * (R_{PS} - I_{PS})$$

we have  $R_{SX}$  from source extractor catalog, we also have  $R_{PS}, I_{PS}$  from the MAST PanStarrs catalog download

We need to fit for  $ZP$  and  $c_0$

$$y = x + ZP + c_0 * Q = x + \hat{b}$$

$$\hat{b} = ZP + c_0 * Q$$

```
[138]: R_SX = table["MagAuto"]
        # you should be getting these from a table, not making up numbers willy-nilly
        R_PS = R_SX + np.random.rand(*R_SX.shape)
        I_PS = R_SX + 2 * np.random.rand(*R_SX.shape)
```

```
[69]: def calib_R(x, ZP, c0):
        return x + ZP + c0 * (R_PS - I_PS)
```

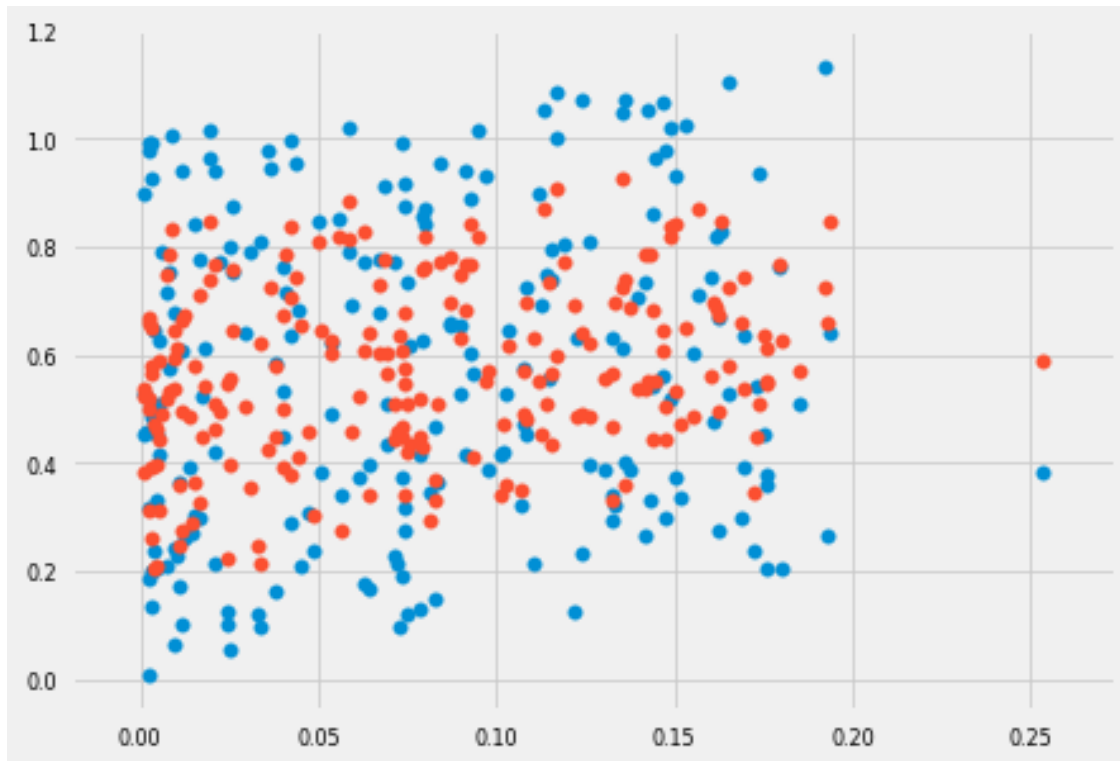
```
[70]: popt, pcov = curve_fit(calib_R, R_SX, R_PS, p0=[1, 1])
```

```
[71]: print(popt)
        print(pcov)
```

```
[0.60865843 0.23365893]
[[0.00041564 0.0002966 ]
 [0.0002966  0.00059265]]
```

```
[73]: plt.scatter(R_SX, R_PS)
        plt.scatter(
            R_SX, calib_R(R_SX, *popt)
        ) # equiv to plt.plot(R_SX, calib(R_SX, popt[0], popt[1]))

        plt.tight_layout();
```



## 0.1 Same c

```
[74]: def calib(X, ZPr, ZPi, ZPz, c):
      R_SX, I_SX, Z_SX = X
      y1 = R_SX + ZPr + c * (R_PS - I_PS)
      y2 = I_SX + ZPi + c * (R_PS - I_PS)
      y3 = Z_SX + ZPz + c * (I_PS - Z_PS)
      return np.hstack([y1, y2, y3])
```

```
[109]: popt, pcov = curve_fit(
      calib, np.vstack([R_SX, I_SX, Z_SX]), np.hstack([R_PS, I_PS, Z_PS]), p0=[0, 0, 0, 1]
      )
```

```
[110]: print(popt)
      print(pcov)
```

```
[ 0.22957457  0.68173358  0.73097073 -0.52380152]
[[ 0.00427717  0.00044218 -0.00040808  0.00088353]
 [ 0.00044218  0.00427717 -0.00040808  0.00088353]
 [-0.00040808 -0.00040808  0.0042116  -0.0008154 ]
 [ 0.00088353  0.00088353 -0.0008154  0.00176542]]
```

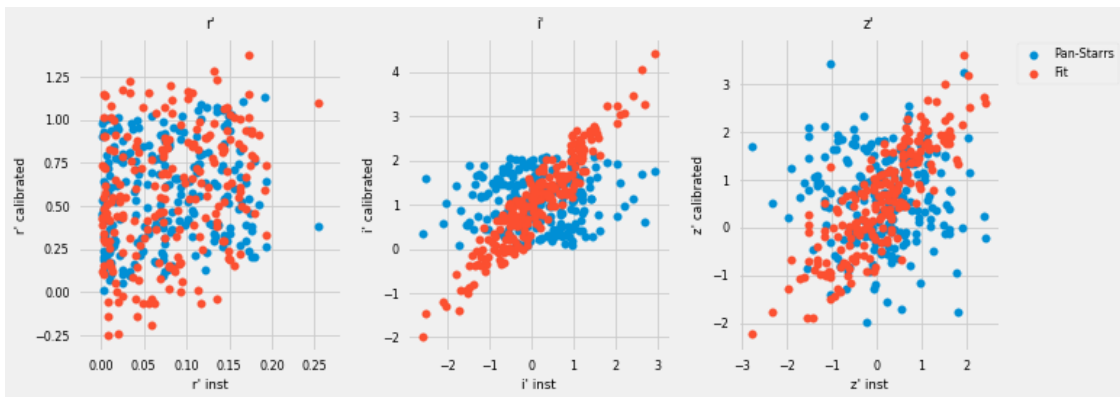
```
[112]: output = calib(
    np.vstack([R_SX, I_SX, Z_SX]), *popt
) # equiv to calib(X, popt[0], popt[1], ...)
y1, y2, y3 = np.reshape(output, (3, -1))

plt.figure(figsize=(11, 4))
plt.subplot(131)
plt.scatter(R_SX, R_PS)
plt.scatter(R_SX, y1)
plt.xlabel("r' inst")
plt.ylabel("r' calibrated")
plt.title("r'")

plt.subplot(132)
plt.scatter(I_SX, I_PS)
plt.scatter(I_SX, y2)
plt.xlabel("i' inst")
plt.ylabel("i' calibrated")
plt.title("i'")

plt.subplot(133)
plt.scatter(Z_SX, Z_PS, label="Pan-Starrs")
plt.scatter(Z_SX, y3, label="Fit")
plt.xlabel("z' inst")
plt.ylabel("z' calibrated")
plt.title("z'")
plt.legend(bbox_to_anchor=(1.04, 1), fancybox=True)

plt.tight_layout();
```



```
[79]: np.reshape([1, 2, 3, 1, 2, 3], (2, 3))
```

```
[79]: array([[1, 2, 3],  
            [1, 2, 3]])
```