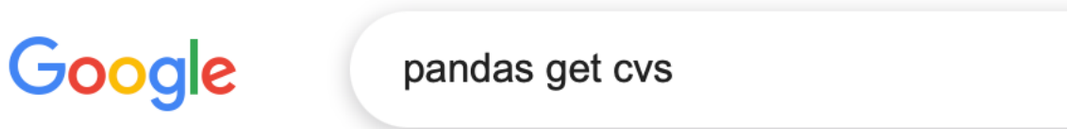
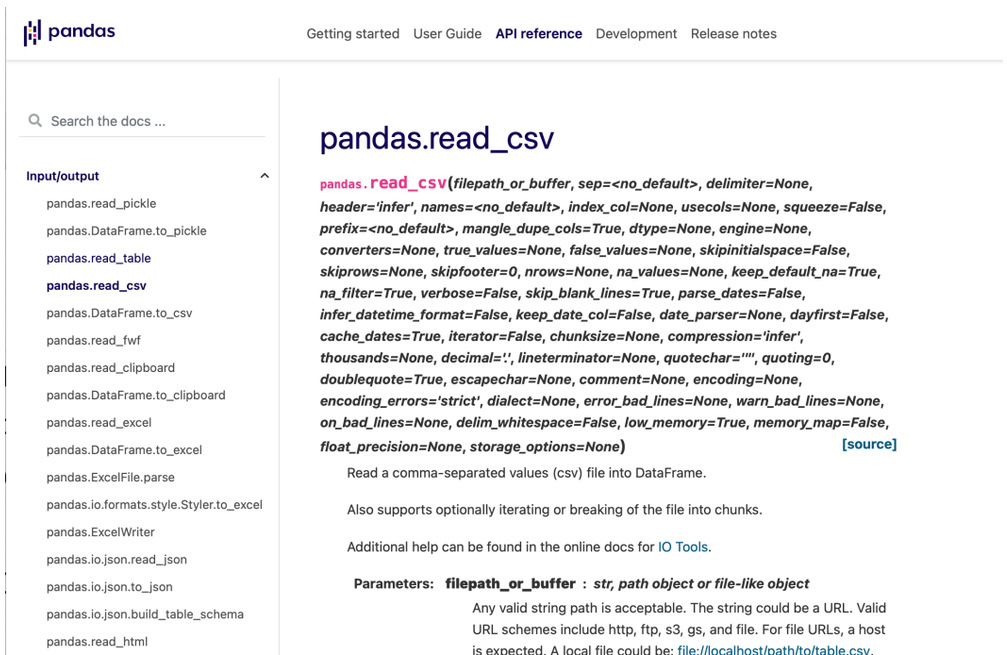
Practice

Penguins

- Let's explore the penguins data set again
  - We now load in Pandas
    - How do we read a csv file?
    - 

    - Go to Pandas documentation



Penguins

Chinstrap Penguin

- Result:
  - Pandas does most of it

```
penguins = pd.read_csv('../Week6Exercises/penguins.csv')
```

- But there are bad values
  - Drop rows that have a single na in them

```
penguins.dropna(inplace=True)
```

Penguins

Adelie Penguin

- An inspection reveals a bad value
  - Can remove the row from the original file
  - Can remove rows with bad values

Gentoo	Biscoe	44.5	15.7	217	4875	.
Gentoo	Biscoe	48.8	16.2	222	6000	MALE
Gentoo	Biscoe	47.2	13.7	214	4925	FEMALE

Penguins

Gentoo Penguin

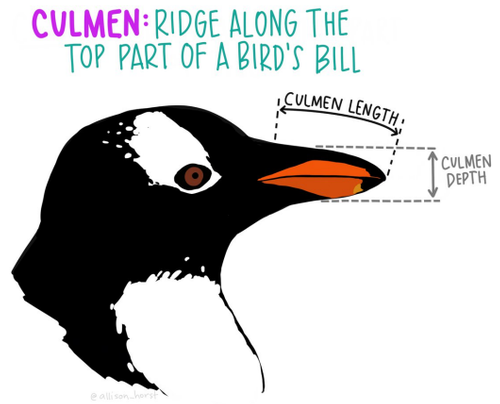
- Need to get ***index*** of the offending row
  - Recall: Use a boolean inside a selection

```
bad_index = penguins[ penguins.sex == '.' ].index
```

- Then remove the row with drop

```
penguins = penguins.drop(bad_index)
```

Penguins



- Grouping: We want to group penguins by species
  - This is done with groupby

```
penguins_by_species = penguins.groupby('species')
```

  - We can group by more than one column name
  - We can aggregate over the group:

```
>>> penguins_by_species.mean()
           culmen_length_mm  culmen_depth_mm  flipper_length_mm  body_mass_g
species
Adelie            38.823973        18.347260         190.102740  3706.164384
Chinstrap         48.833824        18.420588         195.823529  3733.088235
Gentoo            47.568067        14.996639         217.235294  5092.436975
```

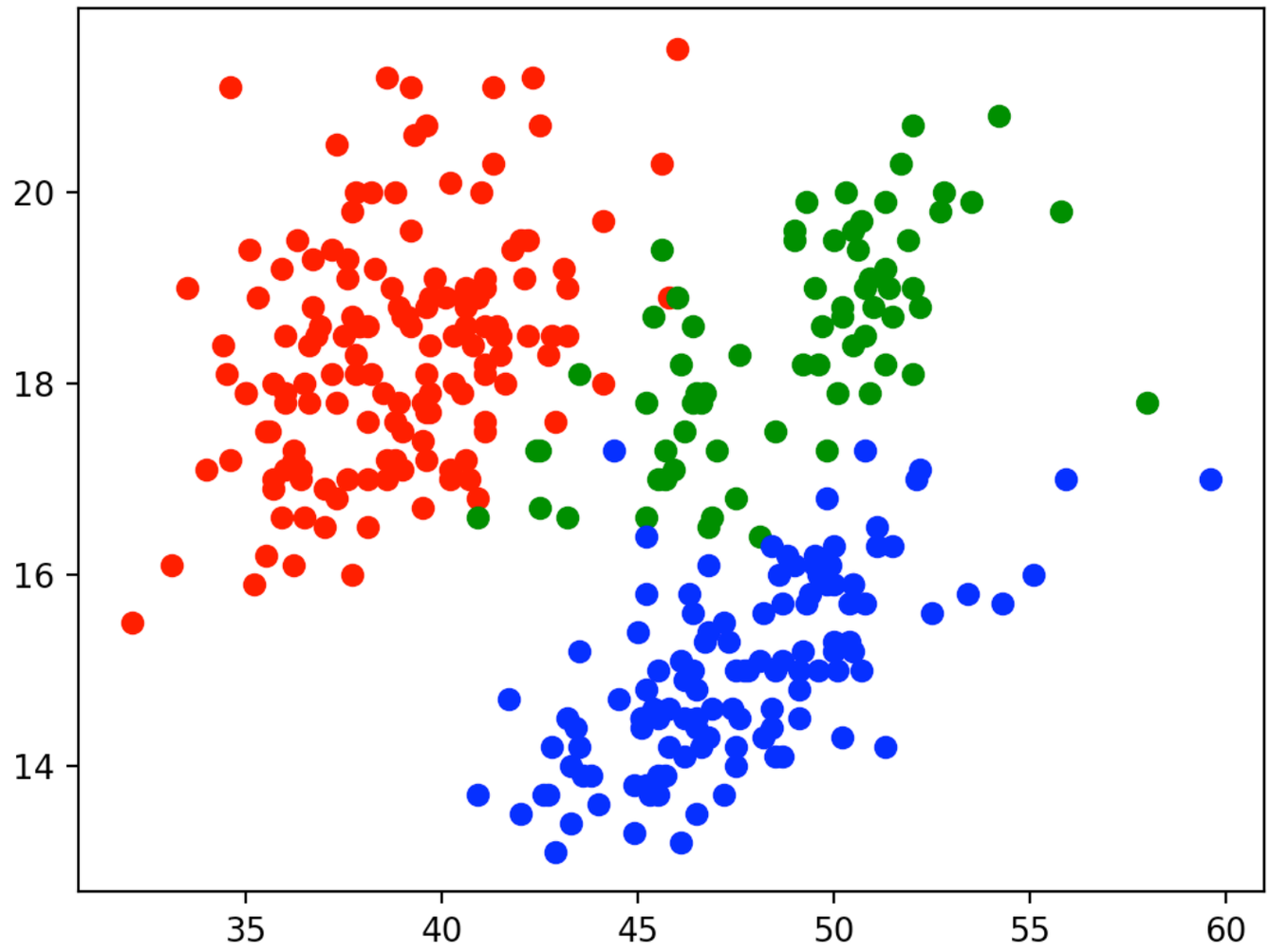
Penguins

- We can plot using scatter
  - Import matplotlib.pyplot as plt
  - Can use color based on species

```
colors = {'Adelie':'red', 'Gentoo': 'blue', 'Chinstrap': 'green'}
```

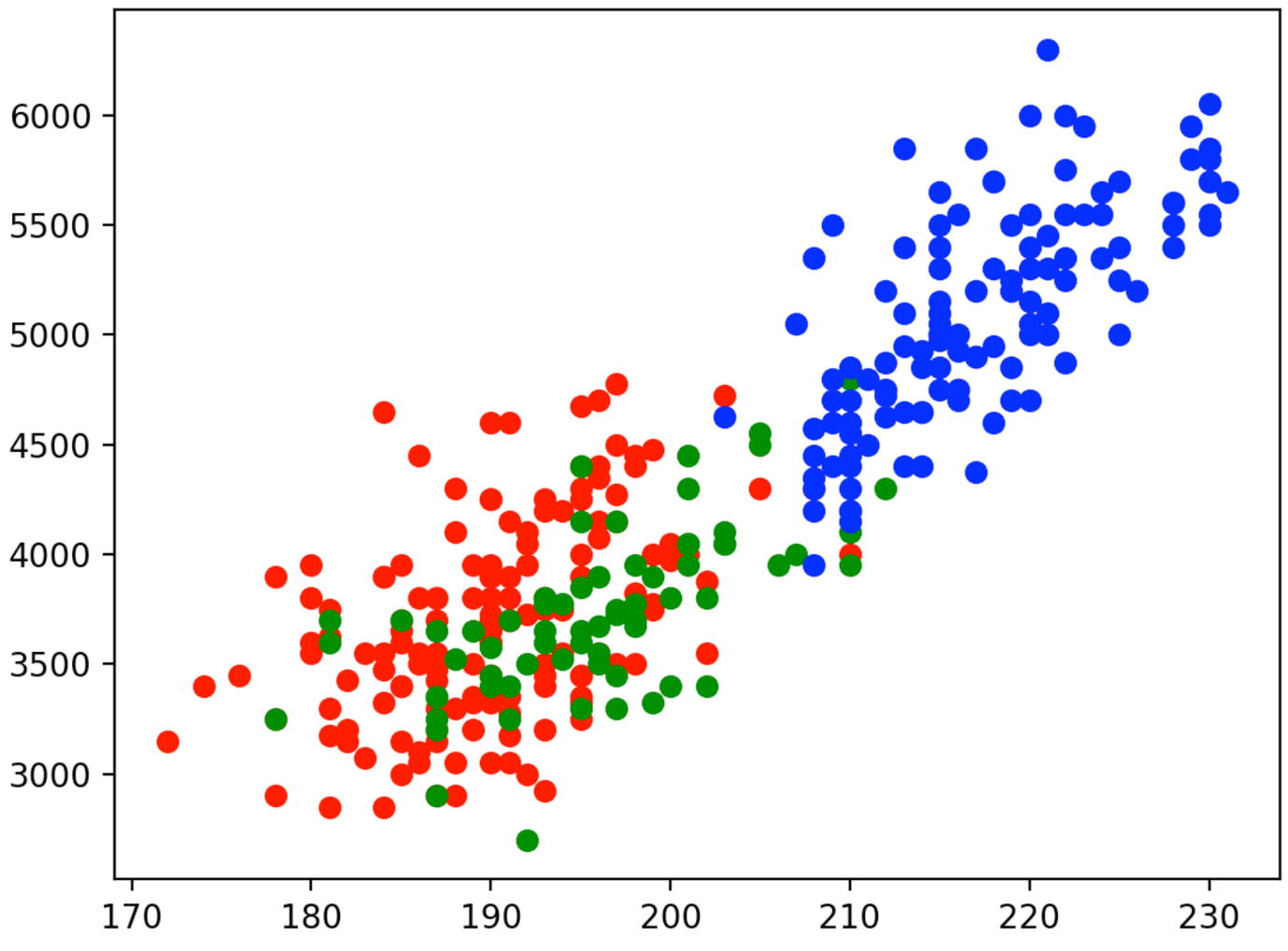
Penguins

```
for key, group in penguins_by_species:
    plt.scatter(group.culmen_length_mm,
                group.culmen_depth_mm,
                color = colors[key])
```



Penguins

```
for key, group in penguins_by_species:
    plt.scatter(group.flipper_length_mm,
                group.body_mass_g,
                color = colors[key])
plt.show()
```



Penguins

- We can group by more than one column

```
penguins_by_species_sex = penguins.groupby(['species', 'sex'])
print(penguins_by_species_sex.mean())
print(penguins_by_species_sex.std())
```

Penguins

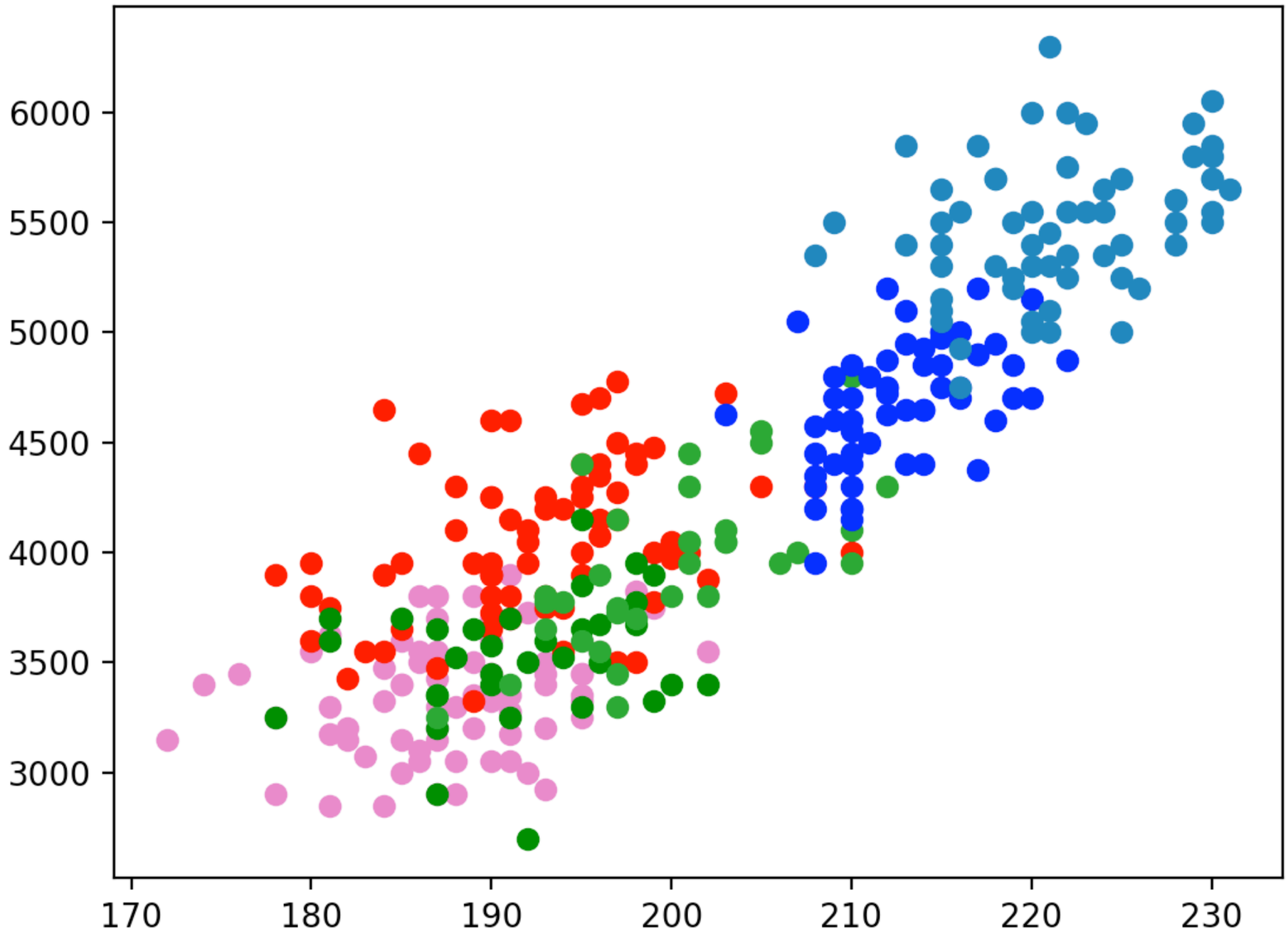
species	sex	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
Adelie	FEMALE	37.257534	17.621918	187.794521	3368.835616
	MALE	40.390411	19.072603	192.410959	4043.493151
Chinstrap	FEMALE	46.573529	17.588235	191.735294	3527.205882
	MALE	51.094118	19.252941	199.911765	3938.970588
Gentoo	FEMALE	45.563793	14.237931	212.706897	4679.741379
	MALE	49.473770	15.718033	221.540984	5484.836066

Penguins

- We can also do scatter diagrams
  - But now we need to look up a few more color names

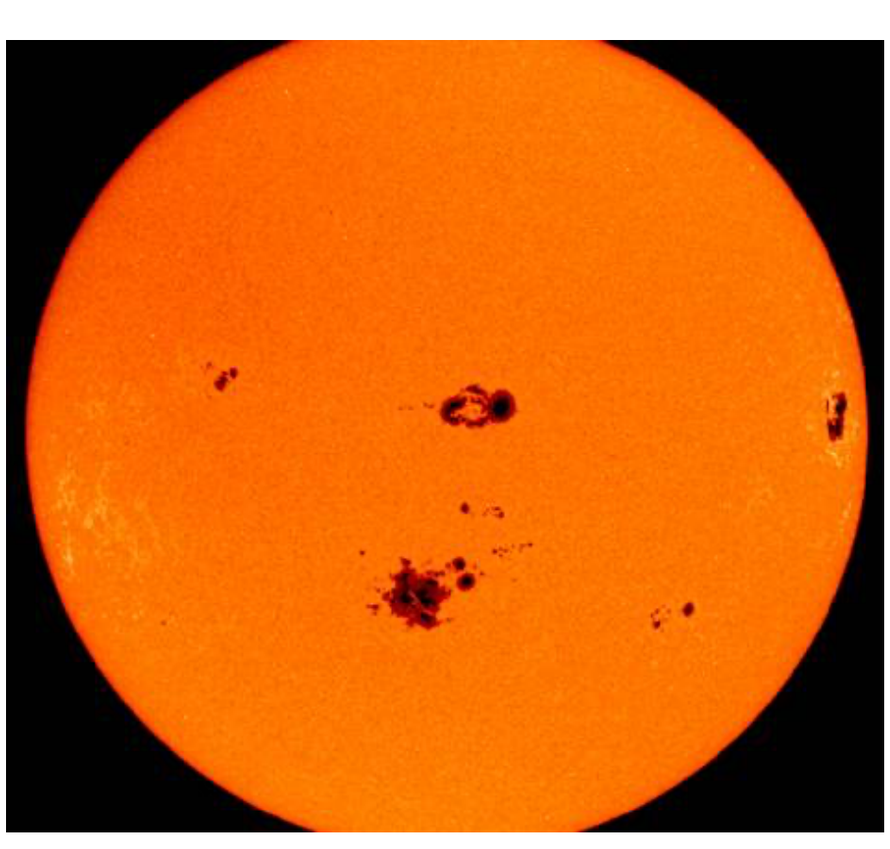
```
colors = {('Adelie','MALE'): 'red',
          ('Adelie','FEMALE'):'tab:pink',
          ('Gentoo', 'MALE'): 'tab:blue',
          ('Gentoo', 'FEMALE'): 'blue',
          ('Chinstrap','MALE'): 'tab:green',
          ('Chinstrap','FEMALE'): 'green'}


for key, group in penguins_by_species_sex:
    plt.scatter(group.flipper_length_mm, group.body_mass_g, color =
colors[key])
plt.show()
```

Penguins



Sunspots

- Observed systematically with telescope in 16th century
  - Activity varies over history



sunspots

Year	Mon	Area
1874	5	365.1
1874	6	415.2
1874	7	1033.5
1874	8	954.1
1874	9	335.3
1874	10	515.0
1874	11	465.4
1874	12	192.5
1875	1	89.8
1875	2	399.2
1875	3	423.0
1875	4	393.8
1875	5	145.4
1875	6	454.0
1875	7	76.8
1875	8	124.2
1875	9	34.0
1875	10	132.4
1875	11	194.0
1875	12	91.0
1876	1	167.5
1876	2	263.6
1876	3	231.4
1876	4	15.8

Sunspots

- Not the easy to make into a pandas data frame via read_csv
  - I found it easier to just build the frame via vanilla python

```
with open('sunspots.csv') as infile:
    line = infile.readline()
    headers = line.split()
    data = []
    for line in infile:
        contents = line.split()
        contents[2] = float(contents[2])
        data.append(contents)
```

Sunspots

- Import from a column name -> data array dictionary

```
sunspots = pd.DataFrame({headers[i]:
    [data[l][i] for l in range(len(data))] for i in range(3)})
```

- Create a new column date

```
sunspots['Date'] = pd.to_datetime(sunspots.Year+"/"+sunspots.Mon)
```

- Drop existing columns Mon and Year

```
sunspots.drop(columns=['Mon', 'Year'], inplace = True)
```

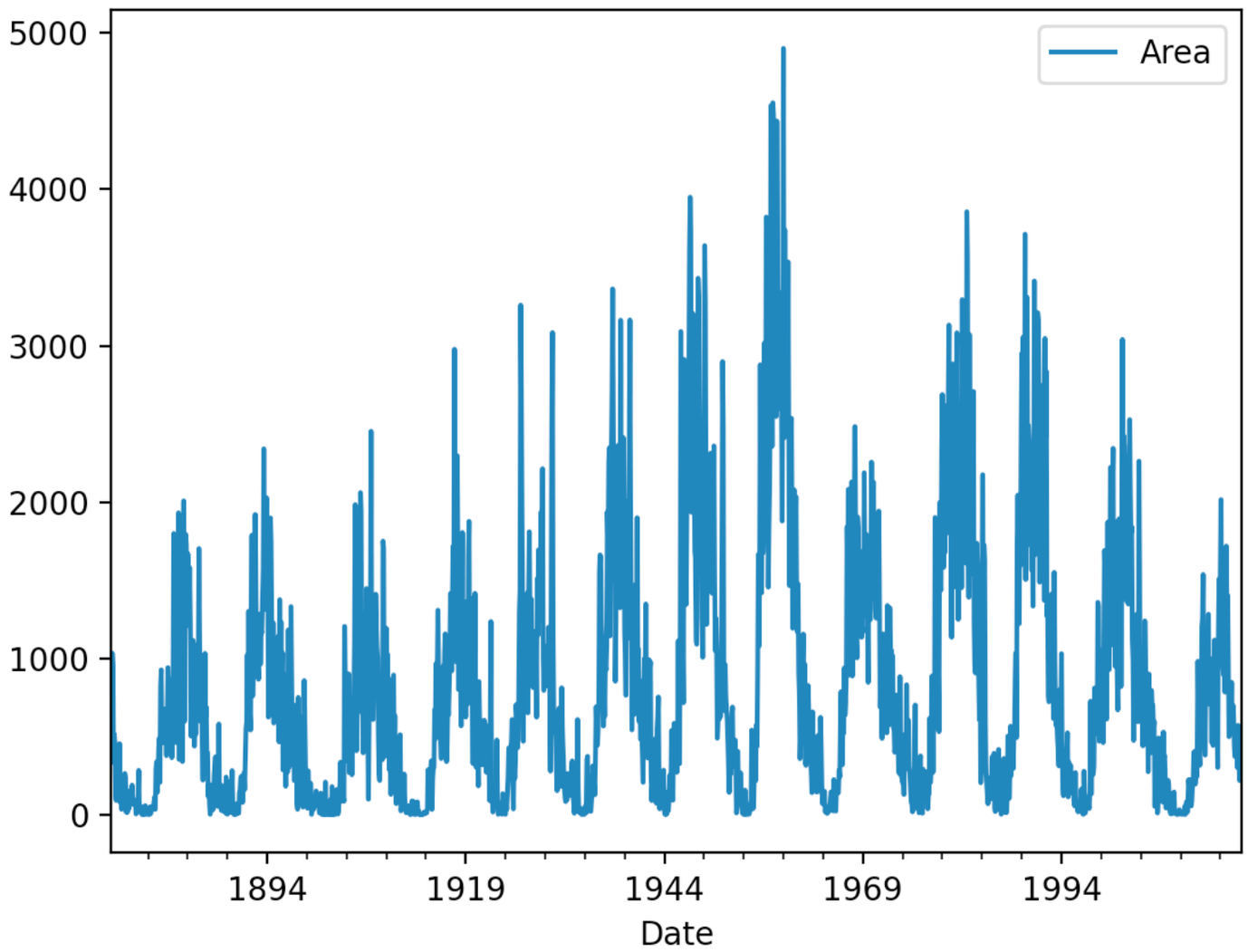
- Reset the index

```
sunspots.reset_index()
sunspots.set_index('Date', inplace=True)
```

Sunspots

- Pandas has very nice automatic graphing

```
sunspots.plot()
plt.show()
```



Sunspots

- Make the area into an np.array

```
X = sunspots['Area'].values
```

- Use two models and try to fit

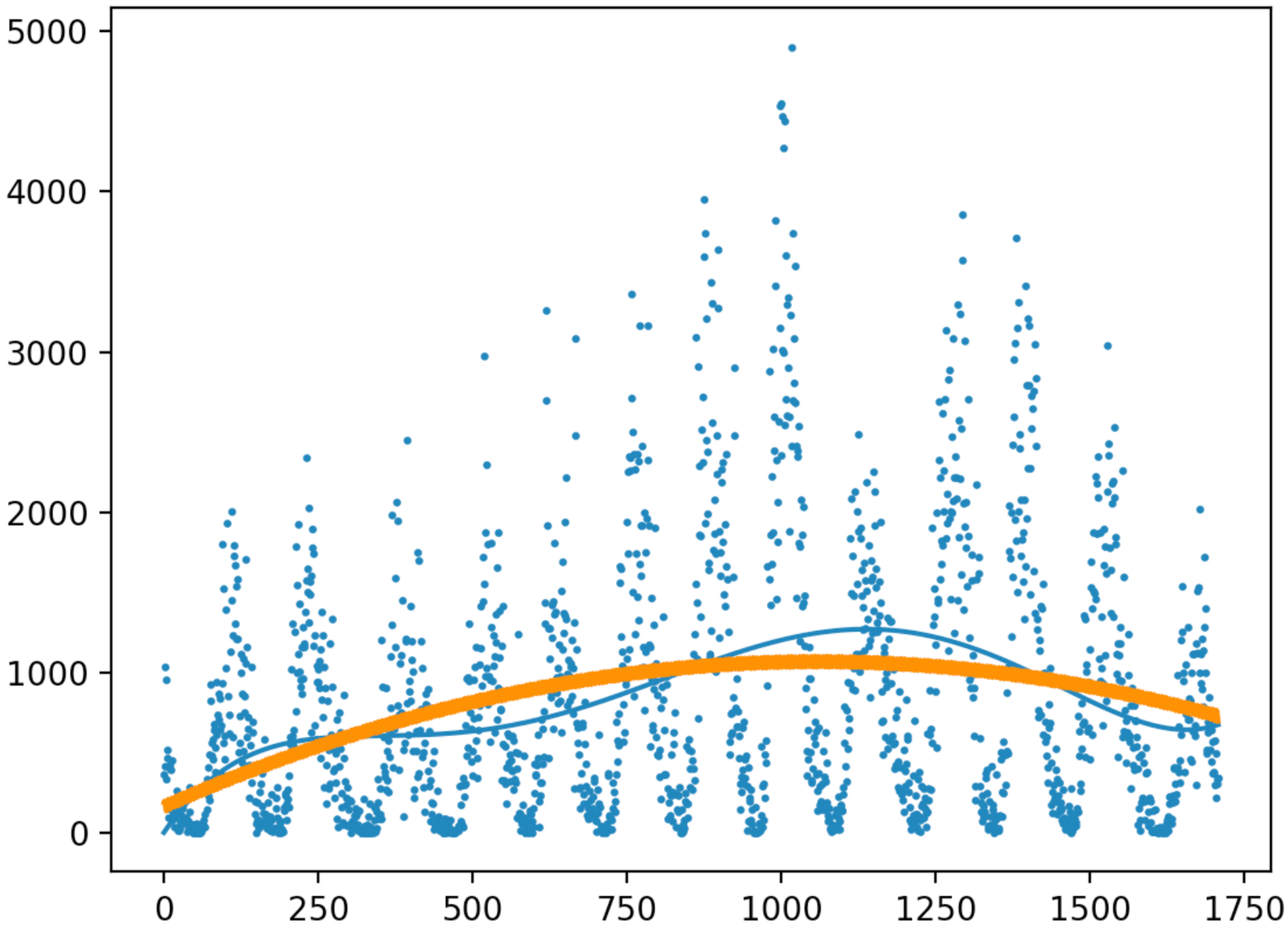
```
def model1(t, alpha, beta, gamma, delta, epsilon, eta):
    return alpha*beta*t+gamma*t**2+delta*t**3+epsilon*t**4+eta*t**5

def model2(t, alpha, beta, gamma, delta, epsilon, eta, phi):
    return alpha+beta*np.sin(gamma*t+epsilon)+eta*t+phi*t**2
```

Sunspots

- But the results of curve-fitting are not good

```
xs = np.arange(0, len(X))
par1, cov1 = opt.curve_fit(model1, xs, X)
par2, cov2 = opt.curve_fit(model2, xs, X)


plt.scatter(xs, X, s=1)
plt.plot(xs, model1(xs, *par1))
plt.plot(xs, model2(xs, *par2))
plt.show()
```

Sunspots



Crop Production in India

- Import as usual

```
crop = pd.read_csv('crop_production.csv')
crop.dropna(inplace=True)
```

- Get the column names

```
print(crop.columns)
```

Crop Production in India

- Pandas has a nice aggregate: describe

```
print(crop.describe())
```

- Gives statistics on what it things are numeric
- Even if they do not make sense

```
          Crop_Year          Area    Production
count  242361.000000  2.423610e+05  2.423610e+05
mean     2005.625773  1.216741e+04  5.825034e+05
std         4.958285  5.085744e+04  1.706581e+07
min      1997.000000  1.000000e-01  0.000000e+00
25%      2002.000000  8.700000e+01  8.800000e+01
50%      2006.000000  6.030000e+02  7.290000e+02
75%      2010.000000  4.545000e+03  7.023000e+03
max      2015.000000  8.580100e+06  1.250800e+09
```

Crop Production in India

- We can get the crop production file into a pandas dataframe

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from math import sin
import scipy.optimize as opt

crop = pd.read_csv('crop_production.csv')
crop.dropna(inplace=True)
```

Crop Production in India

- What are the Indian states according to this data set?
  - Use the unique method

```
print(crop.State_Name.unique())
```

Crop Production in India

- How much rice is produced in Maharashtra?

```
mah = crop[ (crop['State_Name'] == 'Maharashtra') &
            (crop['Crop'] == 'Rice')]
print(mah)
```

```
        State_Name District_Name  Crop_Year  ...  Crop    Area  Production
125200  Maharashtra    AHMEDNAGAR       1997  ...  Rice  5900.0      7200.0
125222  Maharashtra    AHMEDNAGAR       1998  ...  Rice  5600.0      7400.0
125253  Maharashtra    AHMEDNAGAR       1999  ...  Rice  5700.0      8000.0
125285  Maharashtra    AHMEDNAGAR       2000  ...  Rice  5500.0      3800.0
125315  Maharashtra    AHMEDNAGAR       2001  ...  Rice  7300.0      4100.0
...             ...           ...        ...  ...   ...     ...         ...
137704  Maharashtra      YAVATMAL       2007  ...  Rice     6.0         4.0
137721  Maharashtra      YAVATMAL       2008  ...  Rice   500.0       200.0
137737  Maharashtra      YAVATMAL       2009  ...  Rice   200.0       300.0
137766  Maharashtra      YAVATMAL       2011  ...  Rice   100.0       100.0
137782  Maharashtra      YAVATMAL       2012  ...  Rice   100.0        40.0
```

Crop Production in India

- Group by year

```
mah_vals = mah.groupby('Crop_Year').sum()
print(mah_vals)
```

```
                Area  Production
Crop_Year
1997       1339329.0   2330646.0
1998       1483100.0   2467600.0
1999       1498500.0   2549300.0
2000       1511500.0   1929700.0
2001       1514000.0   2651200.0
2002       1523300.0   1854100.0
2003       1529900.0   2834100.0
2004       1508700.0   2147200.0
2005       1503000.0   2665900.0
2006       1529300.0   2571700.0
2007         36836.0     29128.0
2008       1499500.0   2234210.0
2009       1450000.0   2137114.0
2010       1486200.0   2624900.0
2011       1544200.0   2848900.0
2012       1559000.0   3078040.0
2013       1604800.0   3108290.0
2014       1550900.0   2946600.0
```

Crop Production in India

- Create a new column yield
  - Production over area

```
mah_vals['Yield'] = mah_vals.Area / mah_vals.Production
print(mah_vals)
```

Crop Production in India

- Let's approximate yield in dependence on year linearly and quadratically

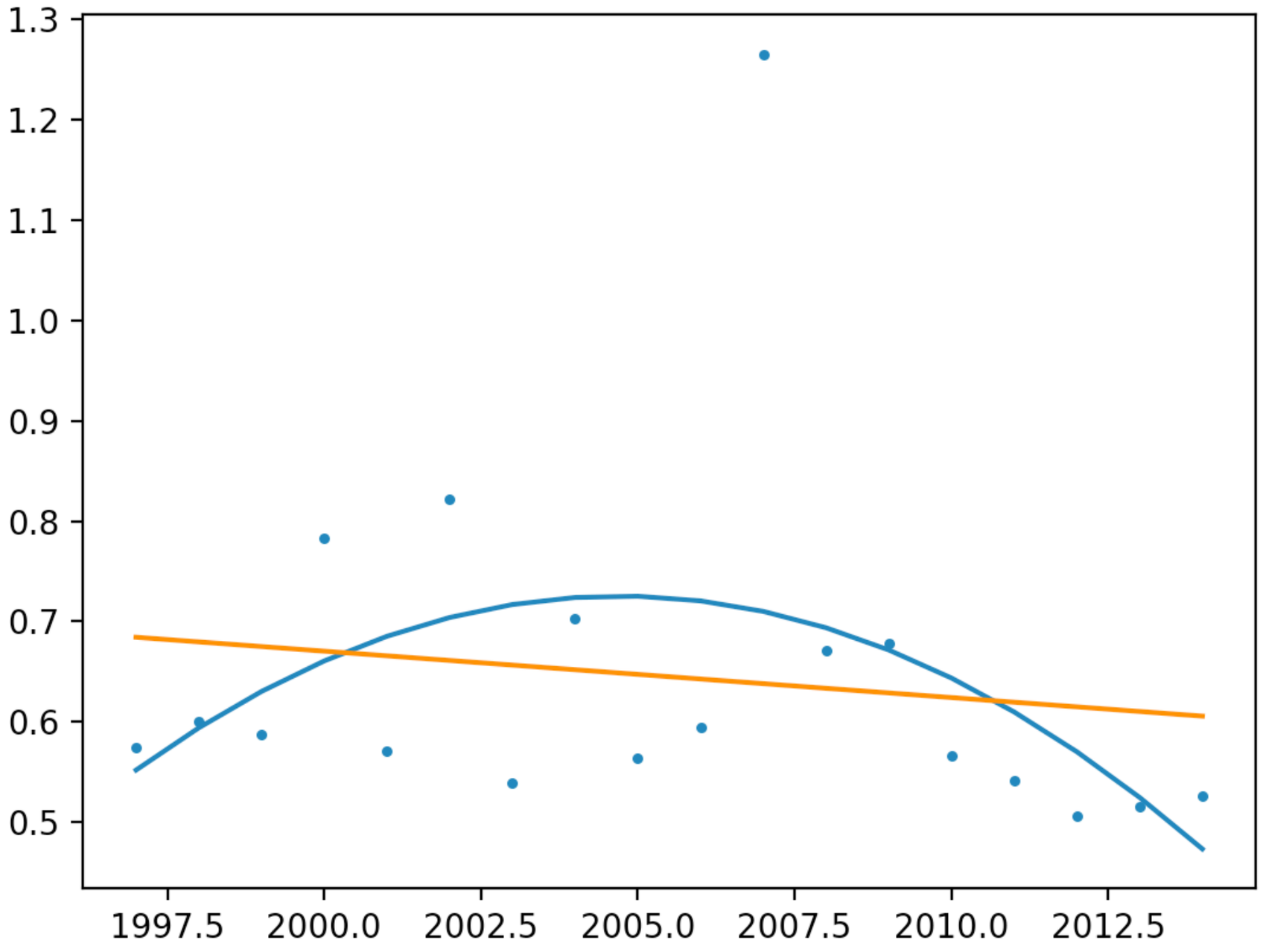
```
def model1(t, alpha, beta, gamma):
    return alpha + beta*t + gamma*t**2

def model2(t, alpha, beta):
    return alpha + beta*t

values = mah_vals.Yield
args = np.arange(1997, 2015)

par1, cov1 = opt.curve_fit(model1, args, values)
par2, cov2 = opt.curve_fit(model2, args, values)
```

Crop Production in India

- And visualize
  - Parameter s stands for size

```
plt.scatter(args, values, s=5)
plt.plot(args, model1(args, *par1))
plt.plot(args, model2(args, *par2))
plt.show()
```

Crop Production in India



Crop Production in India

- There is one outlier for 2007
- Let's get the years without 2007
  - Need to use np.concatenate with a **tuple** as parameter

```
args = np.concatenate((np.arange(1997, 2007),
                       np.arange(2008,2015)))
```

Crop Production in India

- We also need to get rid of the corresponding entrance in the data frame

```
bad_index = mah_vals[mah_vals.Area<1000000].index
mah_vals.drop(bad_index, inplace=True)
print(mah_vals)
```

Crop_Year	Area	Production	Yield
1997	1339329.0	2330646.0	0.574660
1998	1483100.0	2467600.0	0.601029
1999	1498500.0	2549300.0	0.587808
2000	1511500.0	1929700.0	0.783282
2001	1514000.0	2651200.0	0.571062
2002	1523300.0	1854100.0	0.821585
2003	1529900.0	2834100.0	0.539819
2004	1508700.0	2147200.0	0.702636
2005	1503000.0	2665900.0	0.563787
2006	1529300.0	2571700.0	0.594665
2008	1499500.0	2234210.0	0.671154
2009	1450000.0	2137114.0	0.678485
2010	1486200.0	2624900.0	0.566193
2011	1544200.0	2848900.0	0.542034
2012	1559000.0	3078040.0	0.506491
2013	1604800.0	3108290.0	0.516297
2014	1550900.0	2946600.0	0.526335

Crop Production in India

- Now we can repeat the fitting: