



PANDAS FOUNDATIONS

Visual exploratory data analysis



The iris data set

- Famous data set in pattern recognition
- 150 observations, 4 features each
 - Sepal length
 - Sepal width
 - Petal length
 - Petal width
- 3 species: setosa, versicolor, virginica





Data import

```
In [1]: import pandas as pd
```

```
In [2]: import matplotlib.pyplot as plt
```

```
In [3]: iris = pd.read_csv('iris.csv', index_col=0)
```

```
In [4]: print(iris.shape)  
(150, 5)
```



Line plot

```
In [5]: iris.head()
```

```
Out[5]:
```

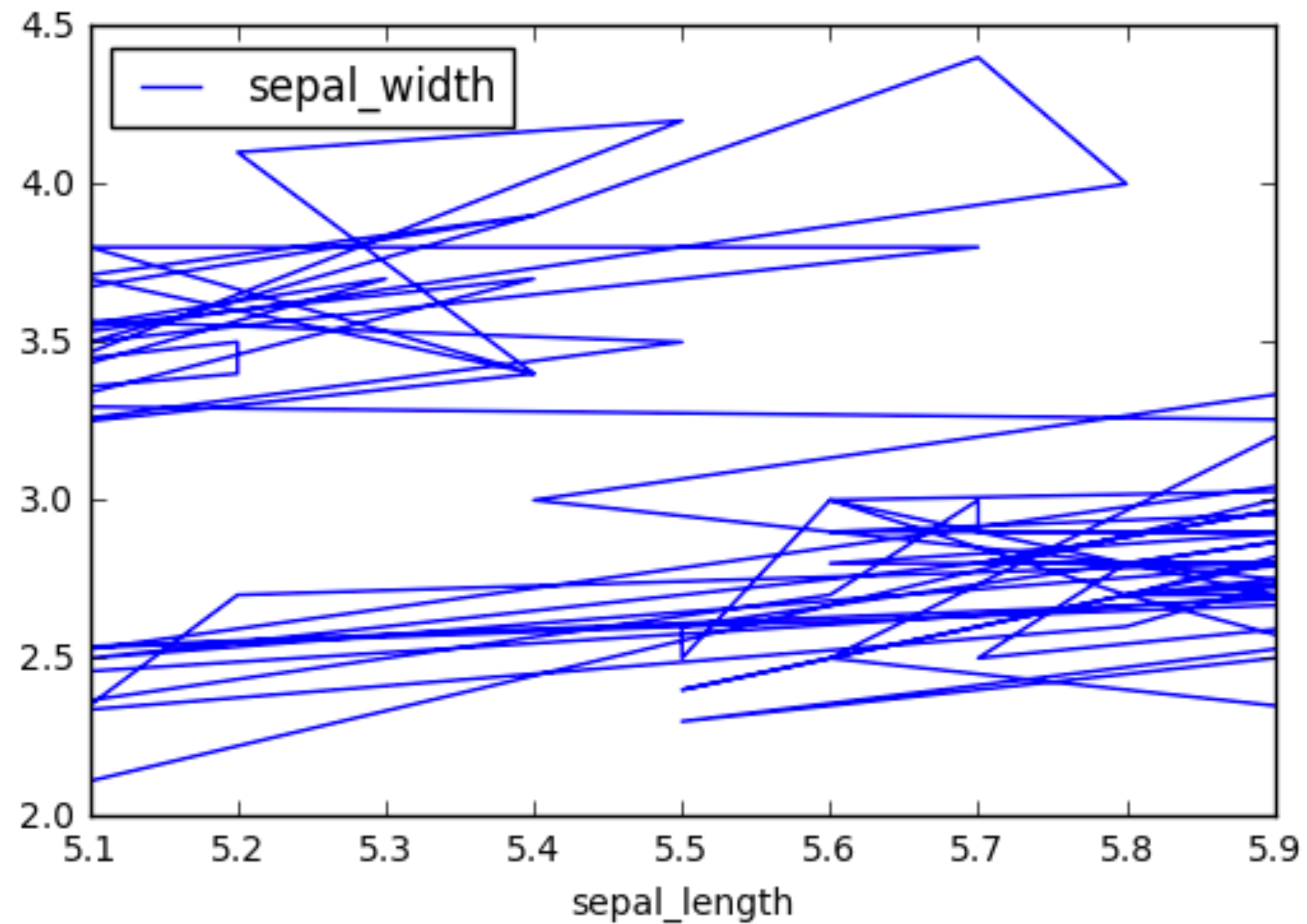
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
In [6]: iris.plot(x='sepal_length', y='sepal_width')
```

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



Line plot





Scatter plot

```
In [8]: iris.plot(x='sepal_length', y='sepal_width',  
....:             kind='scatter')
```

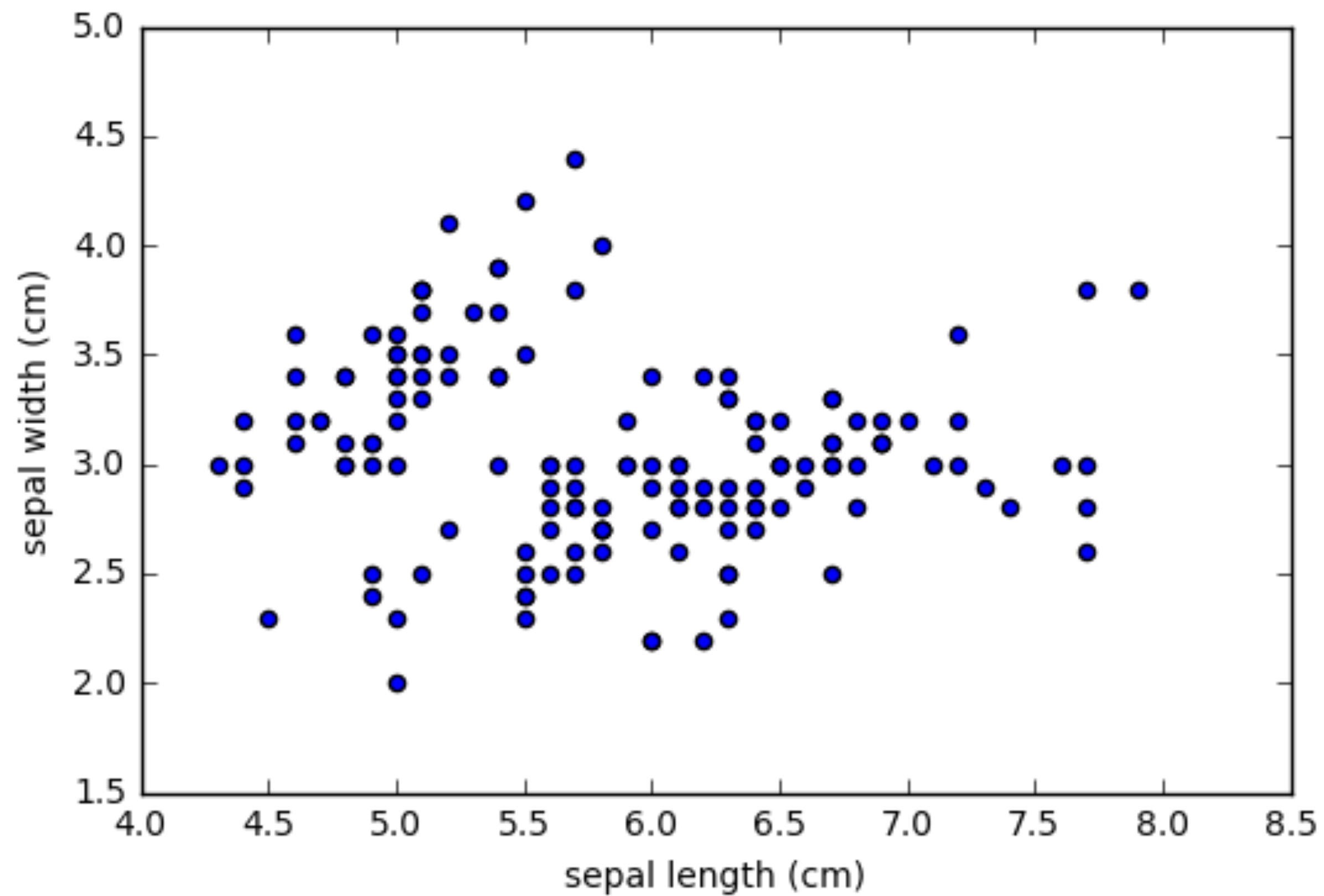
```
In [9]: plt.xlabel('sepal length (cm)')
```

```
In [10]: plt.ylabel('sepal width (cm)')
```

```
In [11]: plt.show()
```



Scatter plot





Box plot

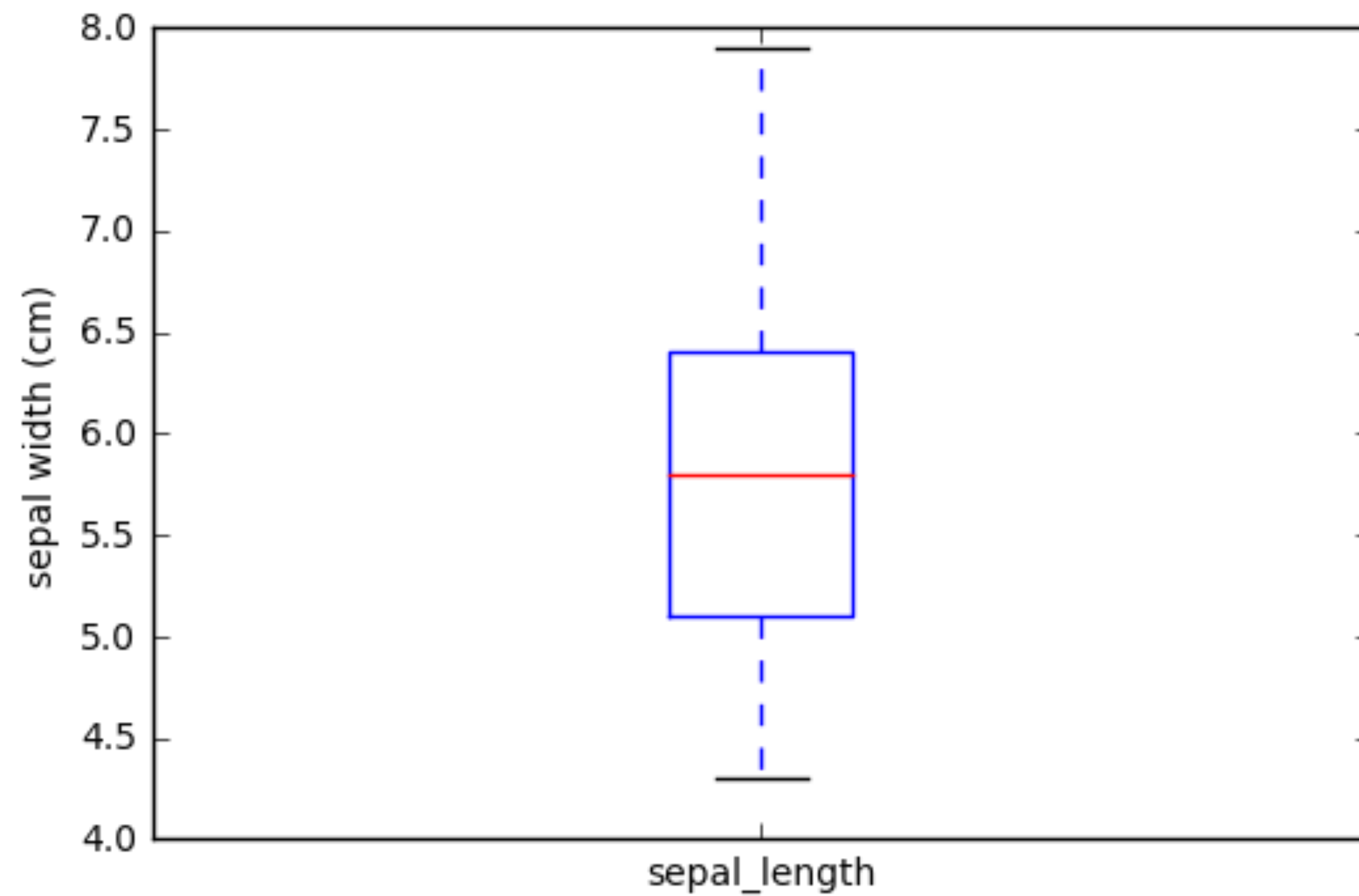
```
In [12]: iris.plot(y='sepal_length', kind='box')
```

```
In [13]: plt.ylabel('sepal width (cm)')
```

```
In [14]: plt.show()
```




Box plot





Histogram

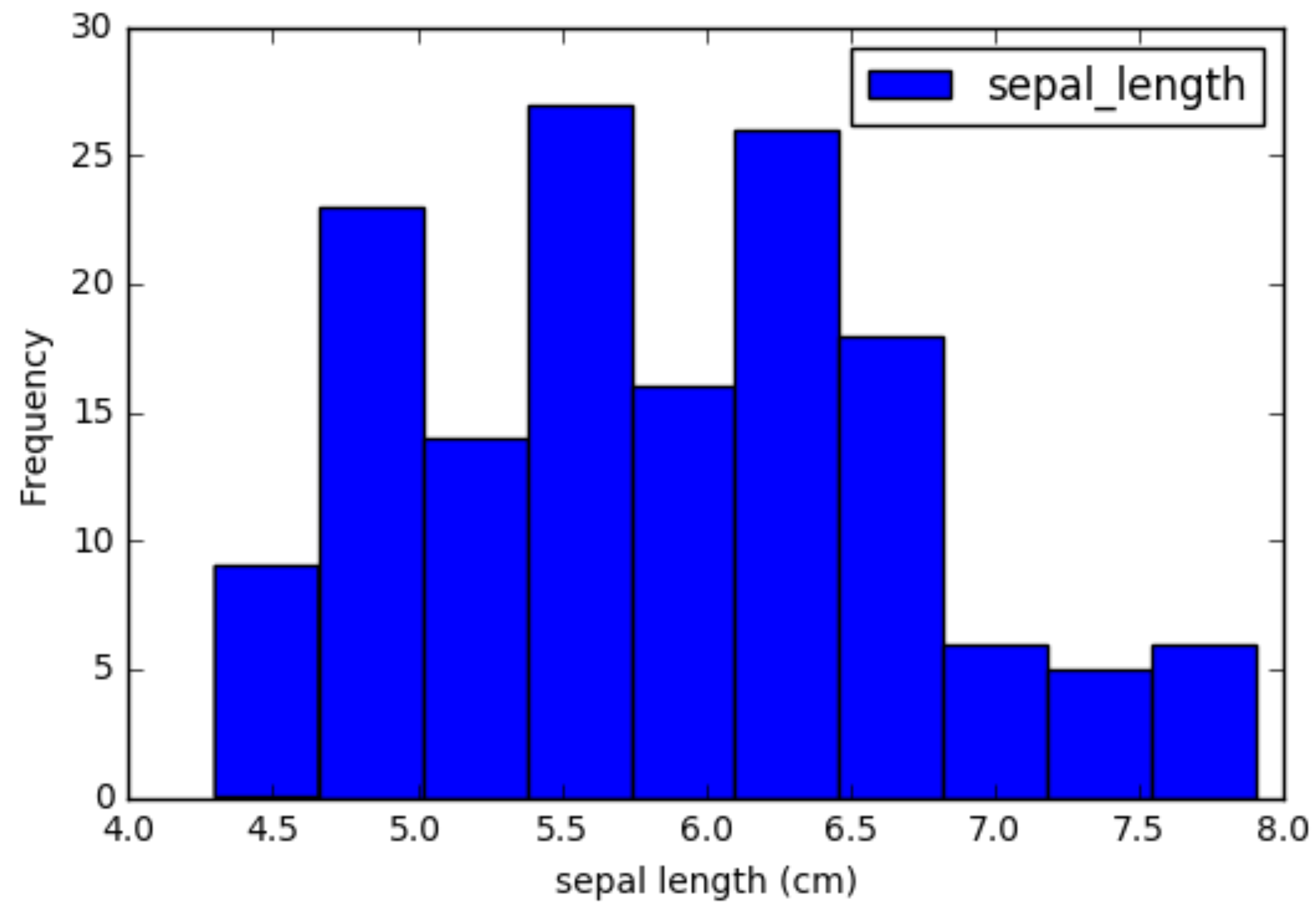
```
In [15]: iris.plot(y='sepal_length', kind='hist')
```

```
In [16]: plt.xlabel('sepal length (cm)')
```

```
In [17]: plt.show()
```



Histogram





Histogram options

- *bins* (integer): number of intervals or bins
- *range* (tuple): extrema of bins (minimum, maximum)
- *normed* (boolean): whether to normalize to one
- *cumulative* (boolean): compute Cumulative Distribution Function (CDF)
- ... more Matplotlib customizations



Customizing histogram

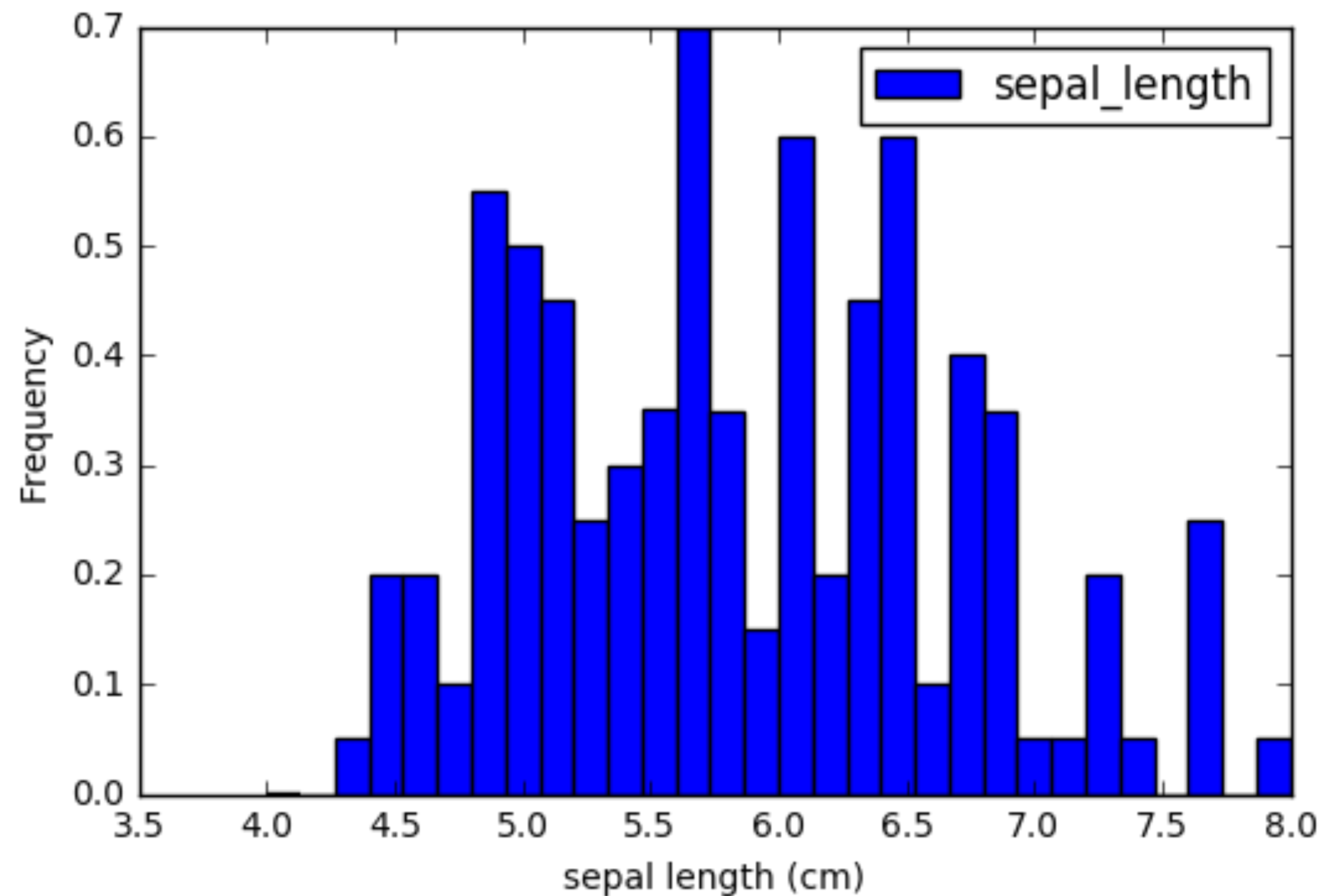
```
In [18]: iris.plot(y='sepal_length', kind='hist',  
....:             bins=30, range=(4,8), normed=True)
```

```
In [19]: plt.xlabel('sepal length (cm)')
```

```
In [20]: plt.show()
```



Customizing histogram





Cumulative distribution

```
In [21]: iris.plot(y='sepal_length', kind='hist', bins=30,  
....:               range=(4,8), cumulative=True, normed=True)
```

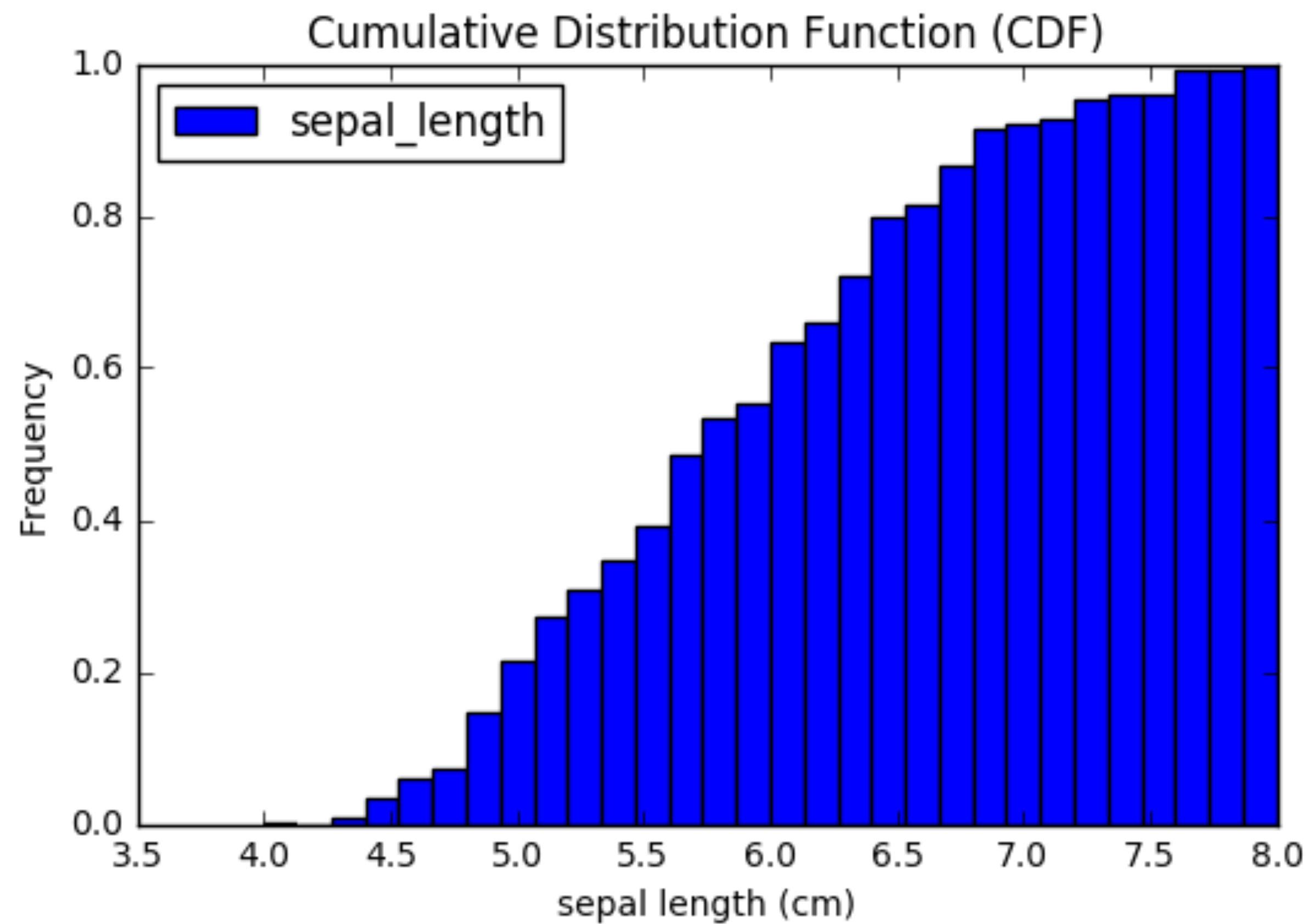
```
In [22]: plt.xlabel('sepal length (cm)')
```

```
In [23]: plt.title('Cumulative distribution function (CDF)')
```

```
In [24]: plt.show()
```




Cumulative distribution





Word of warning

- Three different DataFrame plot idioms
 - *iris.plot(kind='hist')*
 - *iris.plt.hist()*
 - *iris.hist()*
- Syntax/results differ!
- Pandas API still evolving: check documentation!



PANDAS FOUNDATIONS

Let's practice!



PANDAS FOUNDATIONS

Statistical exploratory data analysis



Summarizing with describe()

```
In [1]: iris.describe() # summary statistics
```

```
Out[1]:
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000



Describe

- *count*: number of entries
- *mean*: average of entries
- *std*: standard deviation
- *min*: minimum entry
- *25%*: first quartile
- *50%*: median or second quartile
- *75%*: third quartile
- *max*: maximum entry



Counts

```
In [2]: iris['sepal_length'].count() # Applied to Series
Out[2]: 150
```

```
In [3]: iris['sepal_width'].count() # Applied to Series
Out[3]: 150
```

```
In [4]: iris[['petal_length', 'petal_width']].count() # Applied
....: to DataFrame
Out[4]:
petal_length    150
petal_width     150
dtype: int64
```

```
In [5]: type(iris[['petal_length', 'petal_width']].count()) #
....: returns Series
Out[5]: pandas.core.series.Series
```




Averages

```
In [6]: iris['sepal_length'].mean() # Applied to Series  
Out[6]: 5.8433333333333335
```

```
In [7]: iris.mean() # Applied to entire DataFrame  
Out[7]:  
sepal_length      5.843333  
sepal_width       3.057333  
petal_length      3.758000  
petal_width       1.199333  
dtype: float64
```

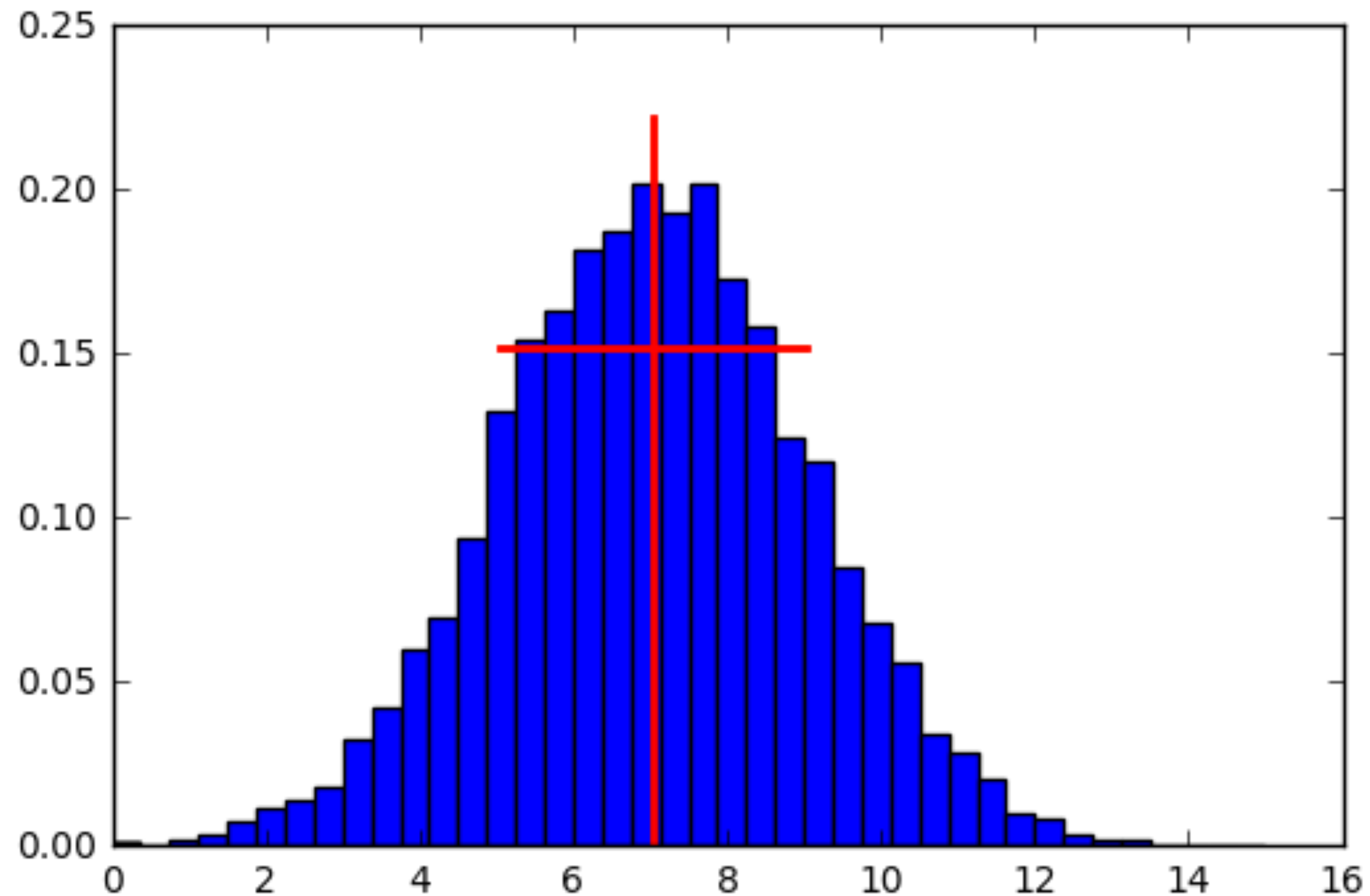


Standard deviations

```
In [8]: iris.std()
Out[8]:
sepal_length    0.828066
sepal_width     0.435866
petal_length    1.765298
petal_width     0.762238
dtype: float64
```



Mean and standard deviation on a bell curve





Medians

```
In [9]: iris.median()  
Out[9]:  
sepal_length    5.80  
sepal_width     3.00  
petal_length    4.35  
petal_width     1.30  
dtype: float64
```



Medians & 0.5 quantiles

```
In [10]: iris.median()
```

```
Out[10]:
```

```
sepal_length    5.80  
sepal_width     3.00  
petal_length    4.35  
petal_width     1.30  
dtype: float64
```

```
In [11]: q = 0.5
```

```
In [12]: iris.quantile(q)
```

```
Out[12]:
```

```
sepal_length    5.80  
sepal_width     3.00  
petal_length    4.35  
petal_width     1.30  
dtype: float64
```



Inter-quartile range (IQR)

```
In [13]: q = [0.25, 0.75]
```

```
In [14]: iris.quantile(q)
```

```
Out[14]:
```

	sepal_length	sepal_width	petal_length	petal_width
0.25	5.1	2.8	1.6	0.3
0.75	6.4	3.3	5.1	1.8



Ranges

```
In [15]: iris.min()
```

```
Out[15]:
```

```
sepal_length    4.3
sepal_width      2
petal_length     1
petal_width     0.1
species         setosa
dtype: object
```

```
In [16]: iris.max()
```

```
Out[16]:
```

```
sepal_length    7.9
sepal_width     4.4
petal_length     6.9
petal_width     2.5
species         virginica
dtype: object
```




Box plots

```
In [17]: iris.plot(kind= 'box')
```

```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x118a3d5f8>
```

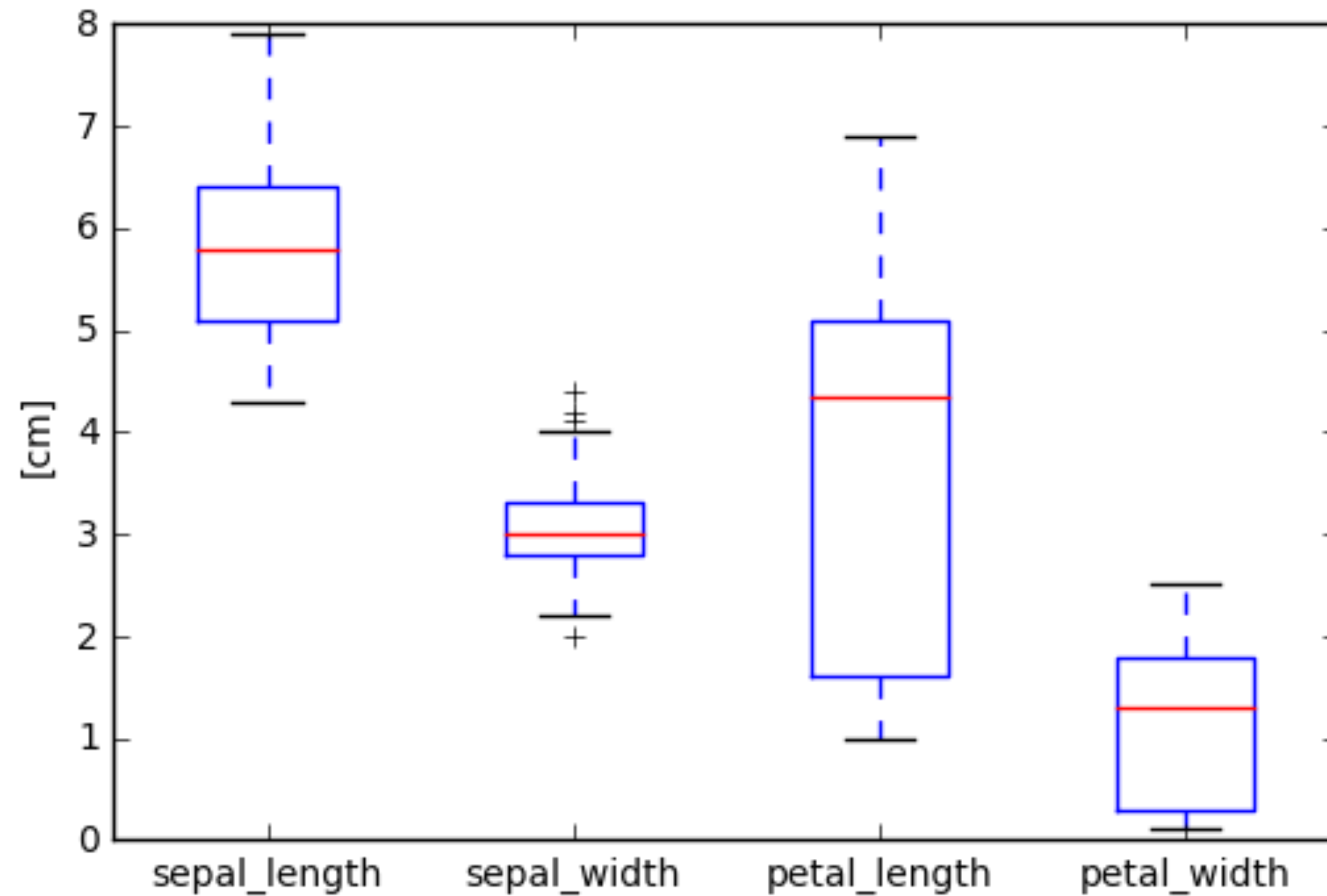
```
In [18]: plt.ylabel(' [cm] ')
```

```
Out[18]: <matplotlib.text.Text at 0x118a524e0>
```

```
In [19]: plt.show()
```



Box plots





Percentiles as quantiles

```
In [20]: iris.describe() # summary statistics
```

```
Out[20]:
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000



PANDAS FOUNDATIONS

Let's practice!



PANDAS FOUNDATIONS

Separating populations



```
In [1]: iris.head()
```

```
Out[1]:
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa



Describe species column

```
In [2]: iris['species'].describe()
```

```
Out[2]:
```

```
count          150  
unique           3  
top          setosa  
freq           50
```

```
Name: species, dtype: object
```

count: # non-null entries

unique: # distinct values

top: most frequent category

freq: # occurrences of top



Unique & factors

```
In [3]: iris['species'].unique()  
Out[3]: array(['setosa', 'versicolor', 'virginica'], dtype=object)
```



Filtering by species

```
In [4]: indices = iris['species'] == 'setosa'
```

```
In [5]: setosa = iris.loc[indices,:] # extract new DataFrame
```

```
In [6]: indices = iris['species'] == 'versicolor'
```

```
In [7]: versicolor = iris.loc[indices,:] # extract new DataFrame
```

```
In [8]: indices = iris['species'] == 'virginica'
```

```
In [9]: virginica = iris.loc[indices,:] # extract new DataFrame
```



Checking species

```
In [10]: setosa['species'].unique()  
Out[10]: array(['setosa'], dtype=object)
```

```
In [11]: versicolor['species'].unique()  
Out[11]: array(['versicolor'], dtype=object)
```

```
In [12]: virginica['species'].unique()  
Out[12]: array(['virginica'], dtype=object)
```

```
In [13]: del setosa['species'], versicolor['species'],  
....:     virginica['species']
```



Checking indexes

```
In [14]: setosa.head(2)
```

```
Out[14]:
```

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2

```
In [15]: versicolor.head(2)
```

```
Out[15]:
```

	sepal_length	sepal_width	petal_length	petal_width
50	7.0	3.2	4.7	1.4
51	6.4	3.2	4.5	1.5

```
In [16]: virginica.head(2)
```

```
Out[16]:
```

	sepal_length	sepal_width	petal_length	petal_width
100	6.3	3.3	6.0	2.5
101	5.8	2.7	5.1	1.9



Visual EDA: all data

```
In [17]: iris.plot(kind= 'hist', bins=50, range=(0,8), alpha=0.3)
```

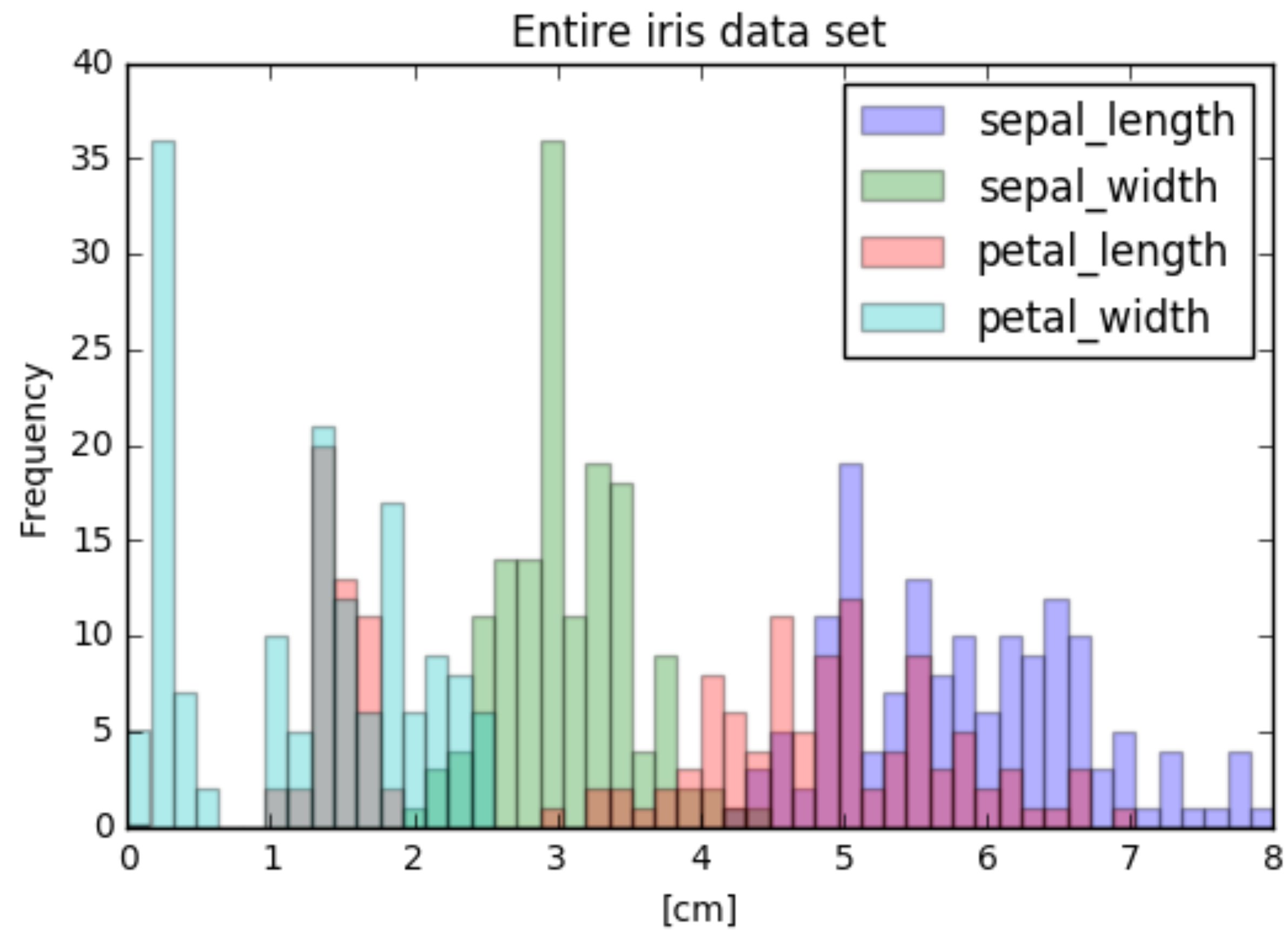
```
In [18]: plt.title('Entire iris data set')
```

```
In [19]: plt.xlabel(' [cm] ')
```

```
In [20]: plt.show()
```



Visual EDA: all data





Visual EDA: individual factors

```
In [21]: setosa.plot(kind='hist', bins=50, range=(0,8), alpha=0.3)
```

```
In [22]: plt.title('Setosa data set')
```

```
In [23]: plt.xlabel('[cm]')
```

```
In [24]: versicolor.plot(kind='hist', bins=50, range=(0,8), alpha=0.3)
```

```
In [25]: plt.title('Versicolor data set')
```

```
In [26]: plt.xlabel('[cm]')
```

```
In [27]: virginica.plot(kind='hist', bins=50, range=(0,8), alpha=0.3)
```

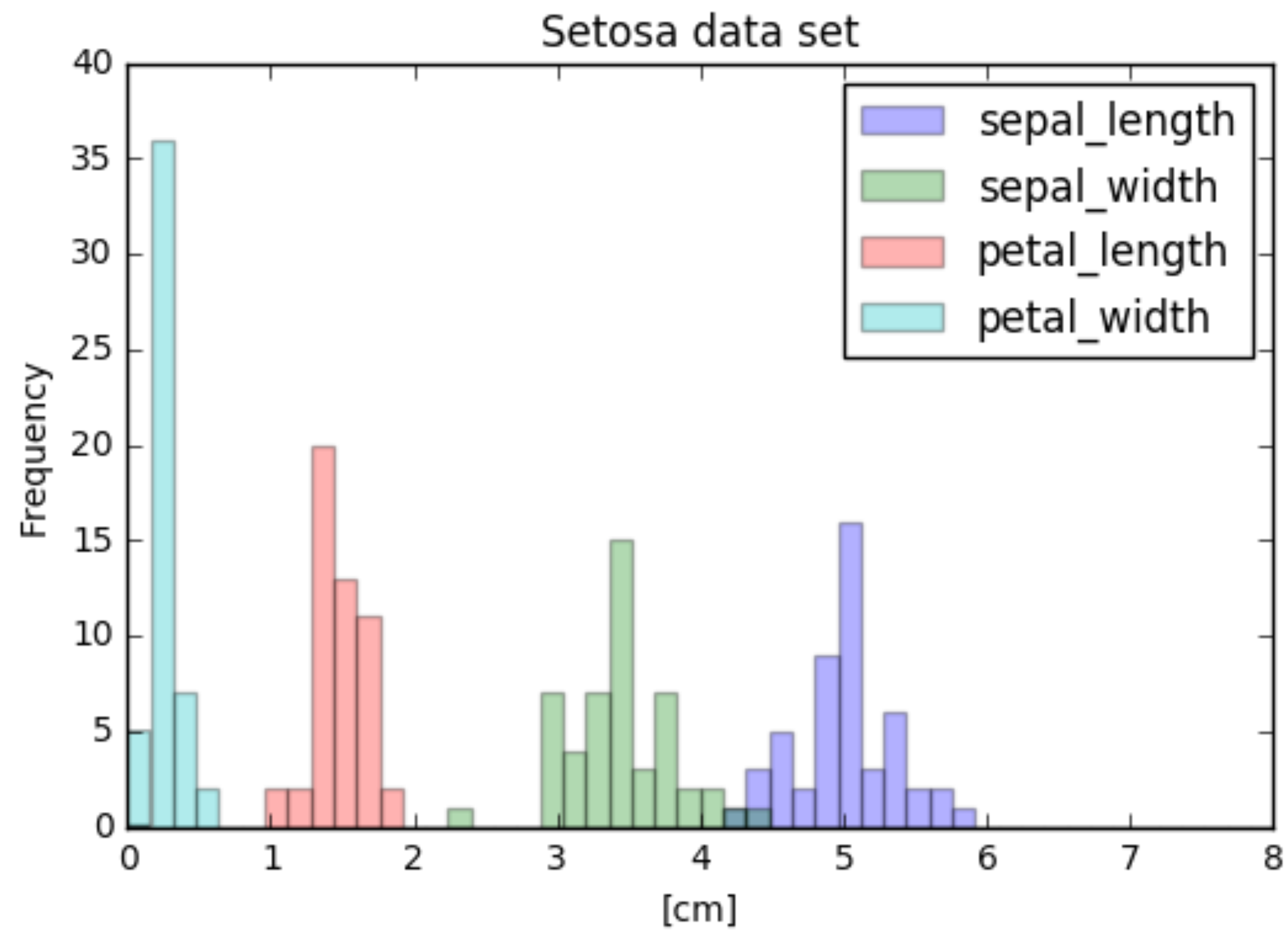
```
In [28]: plt.title('Virginica data set')
```

```
In [29]: plt.xlabel('[cm]')
```

```
In [30]: plt.show()
```

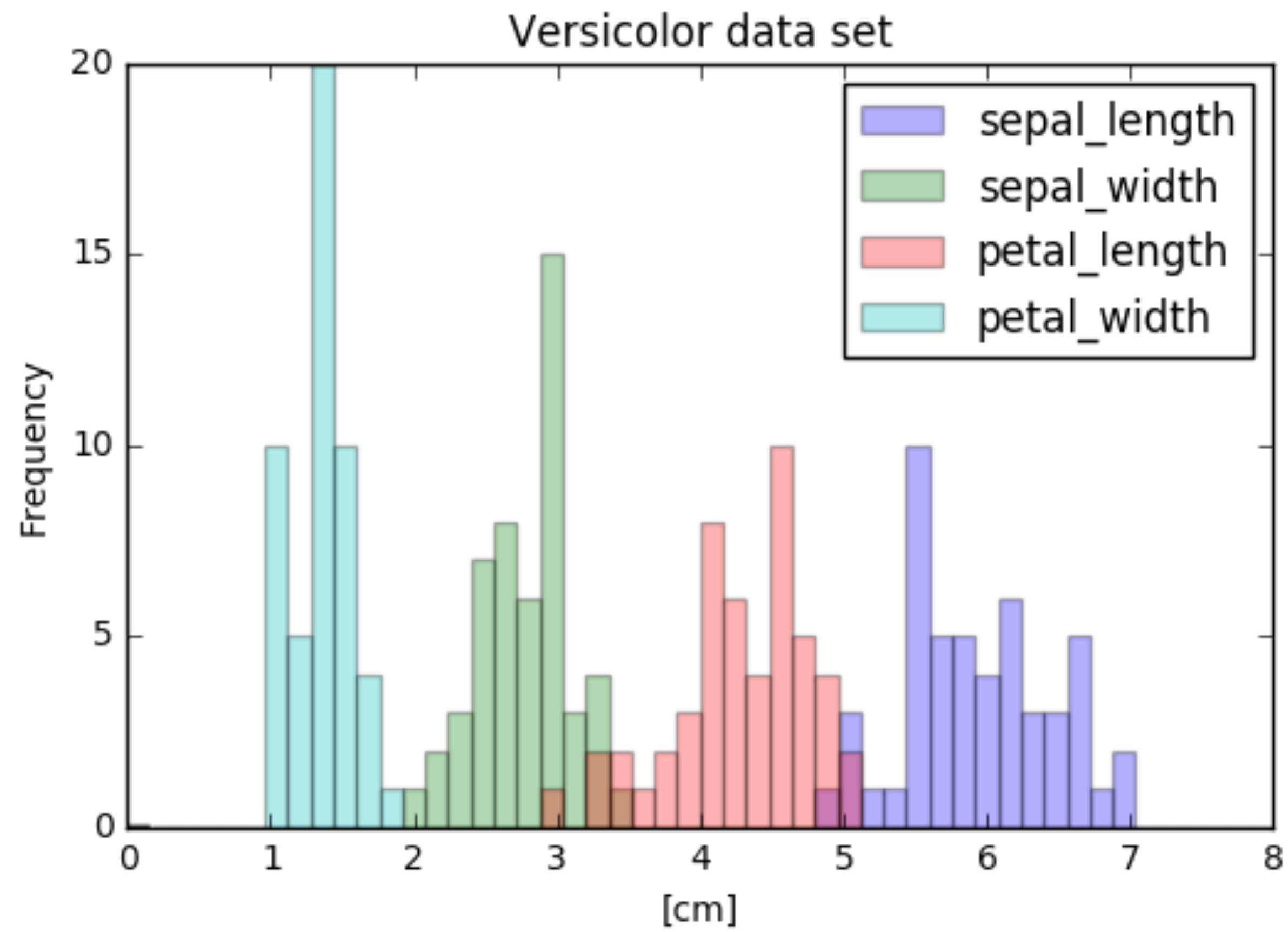


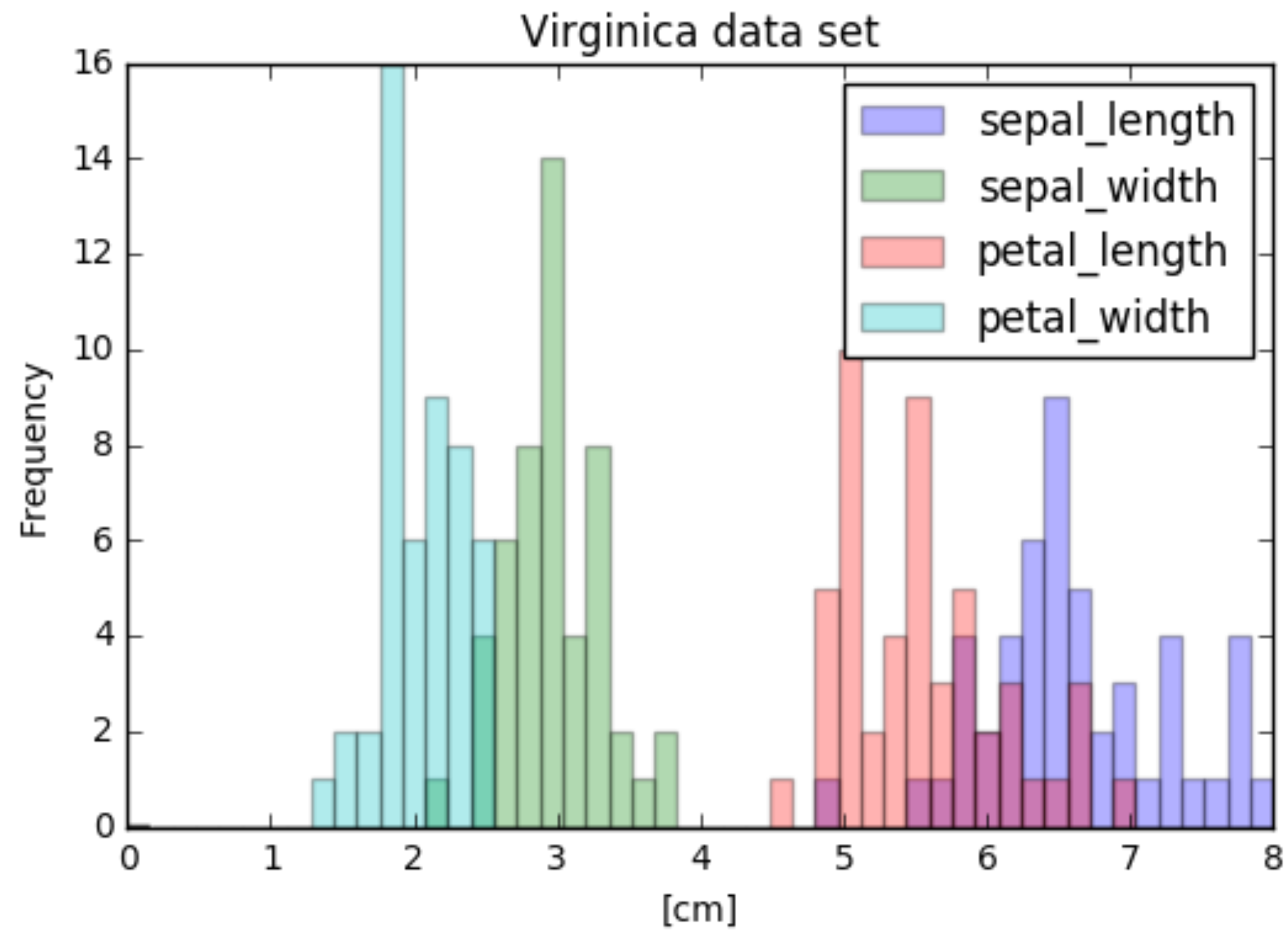
Visual EDA: Setosa data





Visual EDA: Versicolor data







Statistical EDA: describe()

```
In [31]: describe_all = iris.describe()
```

```
In [32]: print(describe_all)
```

```
Out[32]:
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
In [33]: describe_setosa = setosa.describe()
```

```
In [34]: describe_versicolor = versicolor.describe()
```

```
In [35]: describe_virginica = virginica.describe()
```



Computing errors

```
In [36]: error_setosa = 100 * np.abs(describe_setosa -  
    ...: describe_all)
```

```
In [37]: error_setosa = error_setosa/describe_setosa
```

```
In [38]: error_versicolor = 100 * np.abs(describe_versicolor -  
    ...: describe_all)
```

```
In [39]: error_versicolor = error_versicolor/describe_versicolor
```

```
In [40]: error_virginica = 100 * np.abs(describe_virginica -  
    ...: describe_all)
```

```
In [41]: error_virginica = error_virginica/describe_virginica
```



Viewing errors

```
In [42]: print(error_setosa)
```

	sepal_length	sepal_width
count	200.000000	200.000000
mean	16.726595	10.812913
std	134.919250	14.984768
min	0.000000	13.043478
25%	6.250000	12.500000
50%	16.000000	11.764706
75%	23.076923	10.204082
max	36.206897	0.000000

petal_length
200.000000
157.045144
916.502136
0.000000
14.285714
190.000000
223.809524
263.157895

petal_width
200.000000
387.533875
623.284534
0.000000
50.000000
550.000000
500.000000
316.666667



PANDAS FOUNDATIONS

Let's practice!