

Visual exploratory data analysis

PANDAS FOUNDATIONS



Dhavide Aruliah

Director of Training, Anaconda

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

¹ Source: R.A. Fisher, Annual Eugenics, 7, Part II, 179 ² 188 (1936)
(<http://archive.ics.uci.edu/ml/datasets/Iris>)

Data import

```
import pandas as pd
import matplotlib.pyplot as plt
iris = pd.read_csv('iris.csv', index_col=0)
print(iris.shape)
```

```
(150, 5)
```

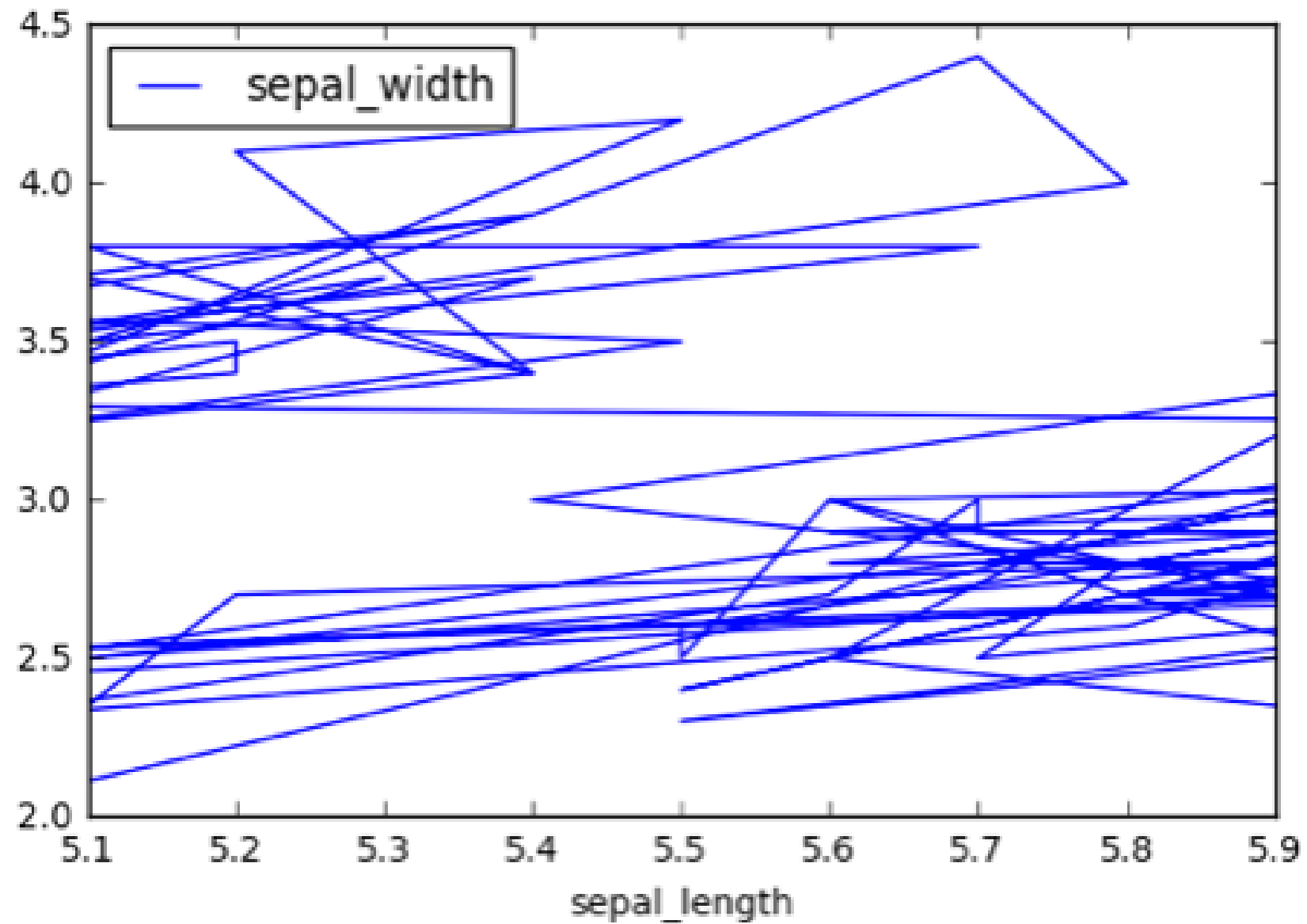
Line plot

```
iris.head()
```

	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

```
iris.plot(x='sepal_length', y='sepal_width')  
plt.show()
```

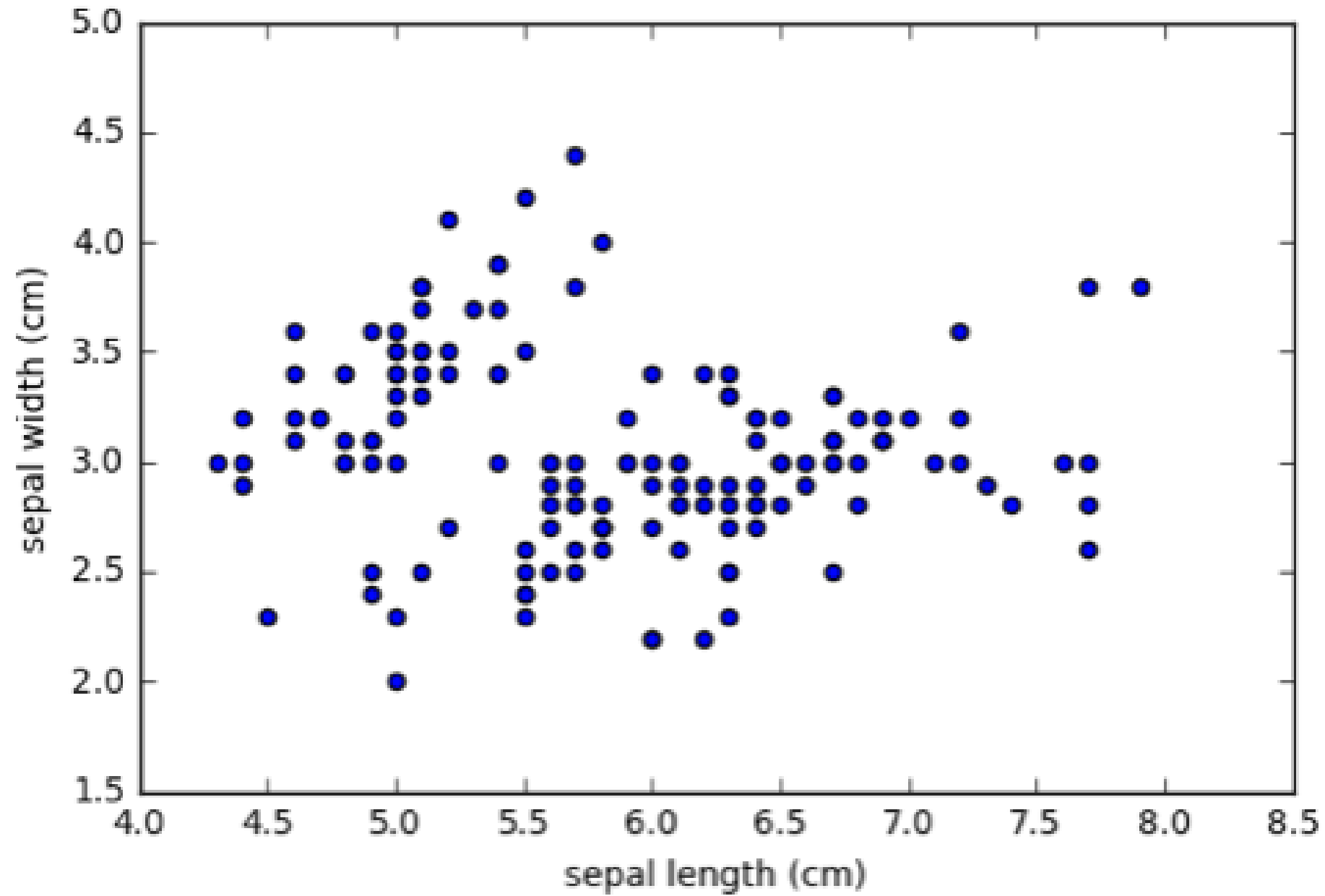
Line plot



Scatter plot

```
iris.plot(x='sepal_length', y='sepal_width',  
          kind='scatter')  
plt.xlabel('sepal length (cm)')  
plt.ylabel('sepal width (cm)')  
plt.show()
```

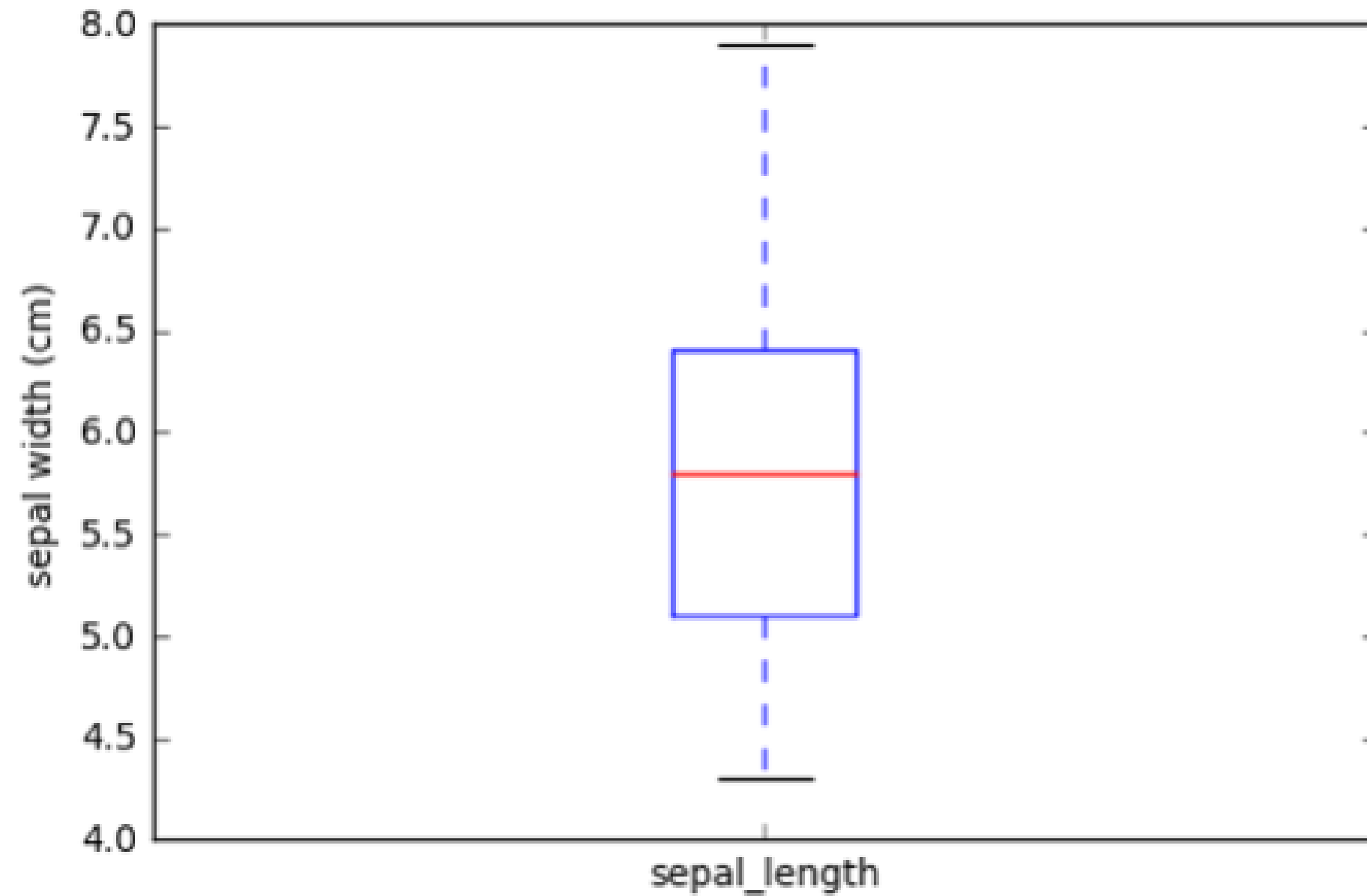
Scatter plot



Box plot

```
iris.plot(y='sepal_length', kind='box')  
plt.ylabel('sepal width (cm)')  
plt.show()
```

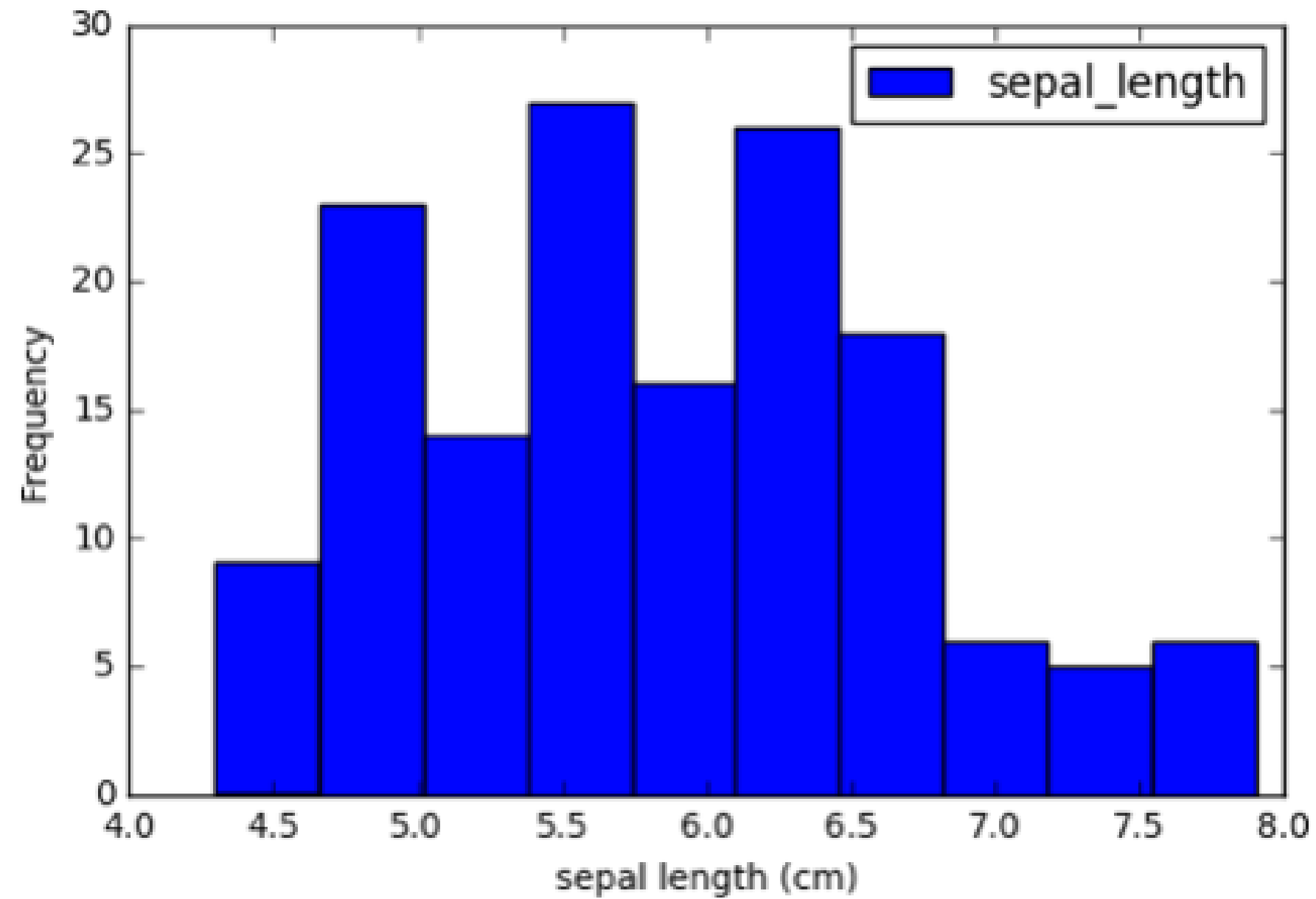

Box plot



Histogram

```
iris.plot(y='sepal_length', kind='hist')  
plt.xlabel('sepal length (cm)')  
plt.show()
```

Histogram



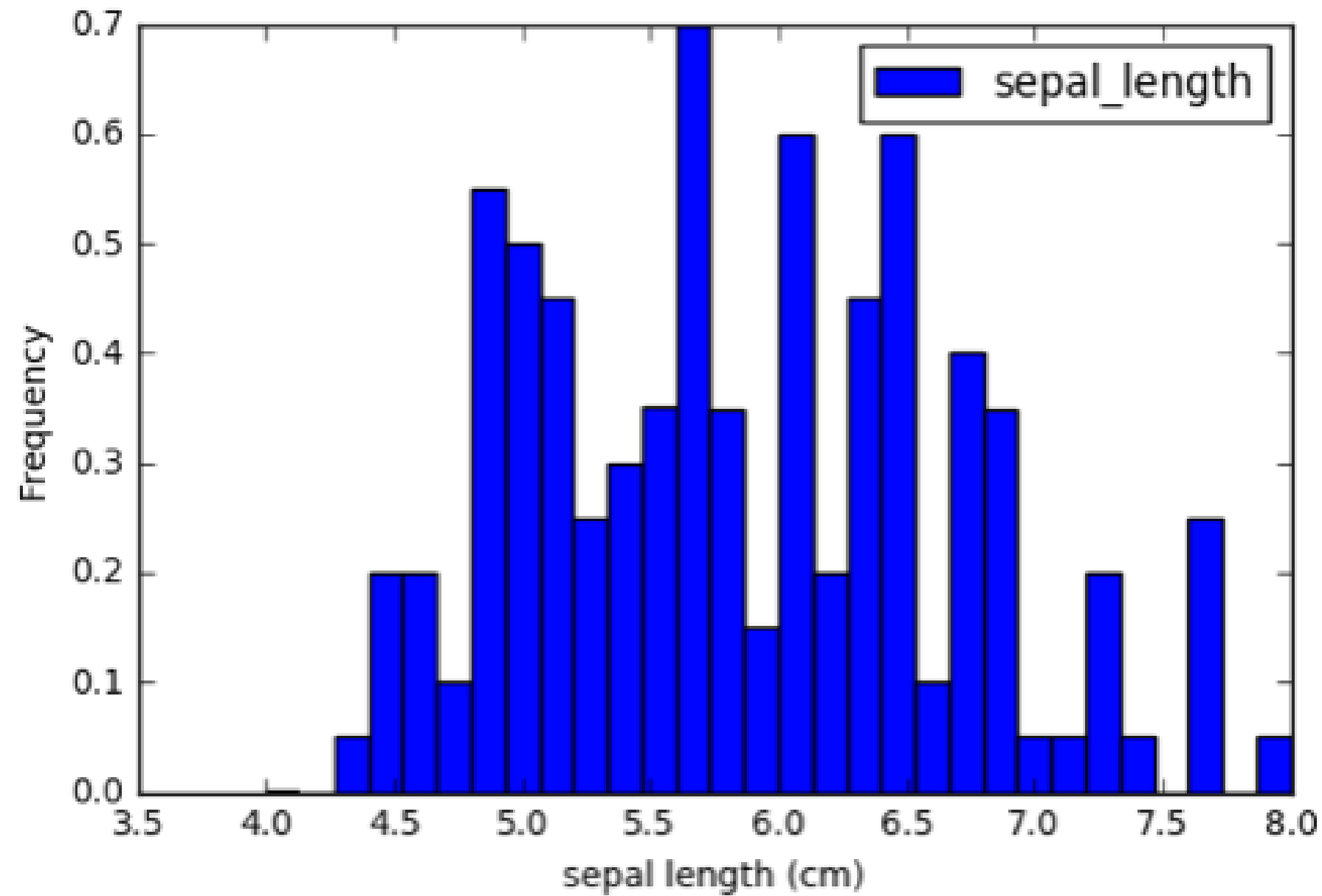
Histogram options

- 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

```
iris.plot(y='sepal_length', kind='hist',  
          bins=30, range=(4, 8), normed=True)  
plt.xlabel('sepal length (cm)')  
plt.show()
```

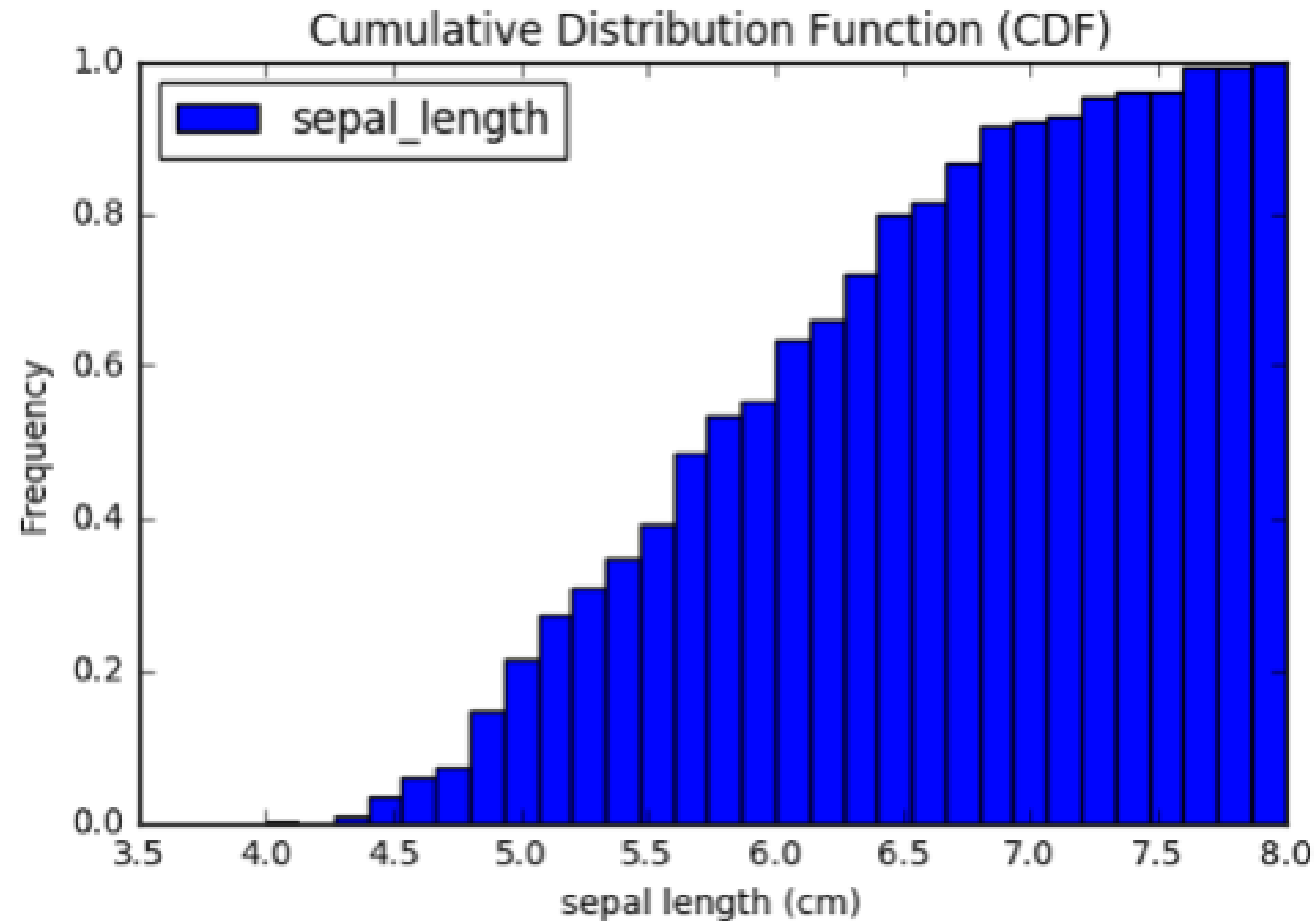
Customizing histogram



Cumulative distribution

```
iris.plot(y='sepal_length', kind='hist', bins=30,  
          range=(4,8), cumulative=True, normed=True)  
plt.xlabel('sepal length (cm)')  
plt.title('Cumulative distribution function (CDF)')  
plt.show()
```

Cumulative distribution



Word of warning

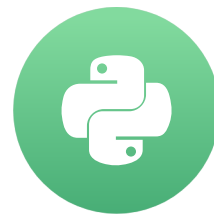
- Three different DataFrame plot idioms
 - `iris.plot(kind='hist')`
 - `iris.plt.hist()`
 - `iris.hist()`
- Syntax/results differ!
- Pandas API still evolving: check documentation!

Let's practice!

PANDAS FOUNDATIONS

Statistical exploratory data analysis

PANDAS FOUNDATIONS



Dhavide Aruliah
Director of Training, Anaconda

Summarizing with describe()

```
iris.describe() # summary statistics
```

	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

```
iris['sepal_length'].count() # Applied to Series
```

```
150
```

```
iris['sepal_width'].count() # Applied to Series
```

```
150
```

```
iris[['petal_length', 'petal_width']].count() # Applied to DataFrame
```

```
petal_length    150  
petal_width     150  
dtype: int64
```

```
type(iris[['petal_length', 'petal_width']].count()) # returns Series
```

```
pandas.core.series.Series
```

Averages

```
iris['sepal_length'].mean() # Applied to Series
```

```
5.8433333333333335
```

```
iris.mean() # Applied to entire DataFrame
```

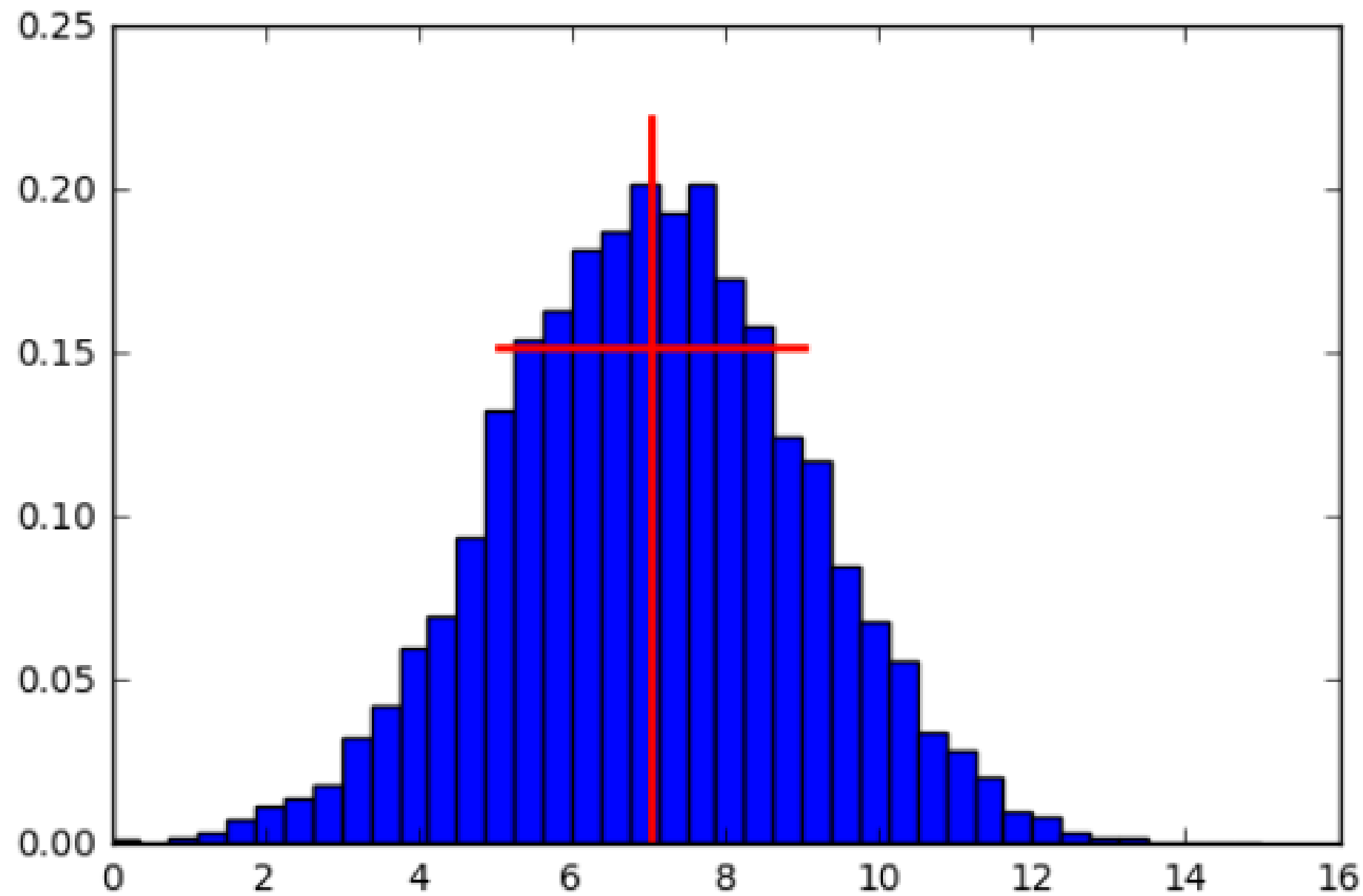
```
sepal_length    5.843333  
sepal_width     3.057333  
petal_length    3.758000  
petal_width     1.199333  
dtype: float64
```

Standard deviations

```
iris.std()
```

```
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

```
iris.median()
```

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

Medians & 0.5 quantiles

```
iris.median()
```

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

```
q = 0.5  
iris.quantile(q)
```

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

Inter-quartile range (IQR)

```
q = [0.25, 0.75]  
iris.quantile(q)
```

	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

```
iris.min()
```

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

```
iris.max()
```

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

Box plots

```
iris.plot(kind= 'box' )
```

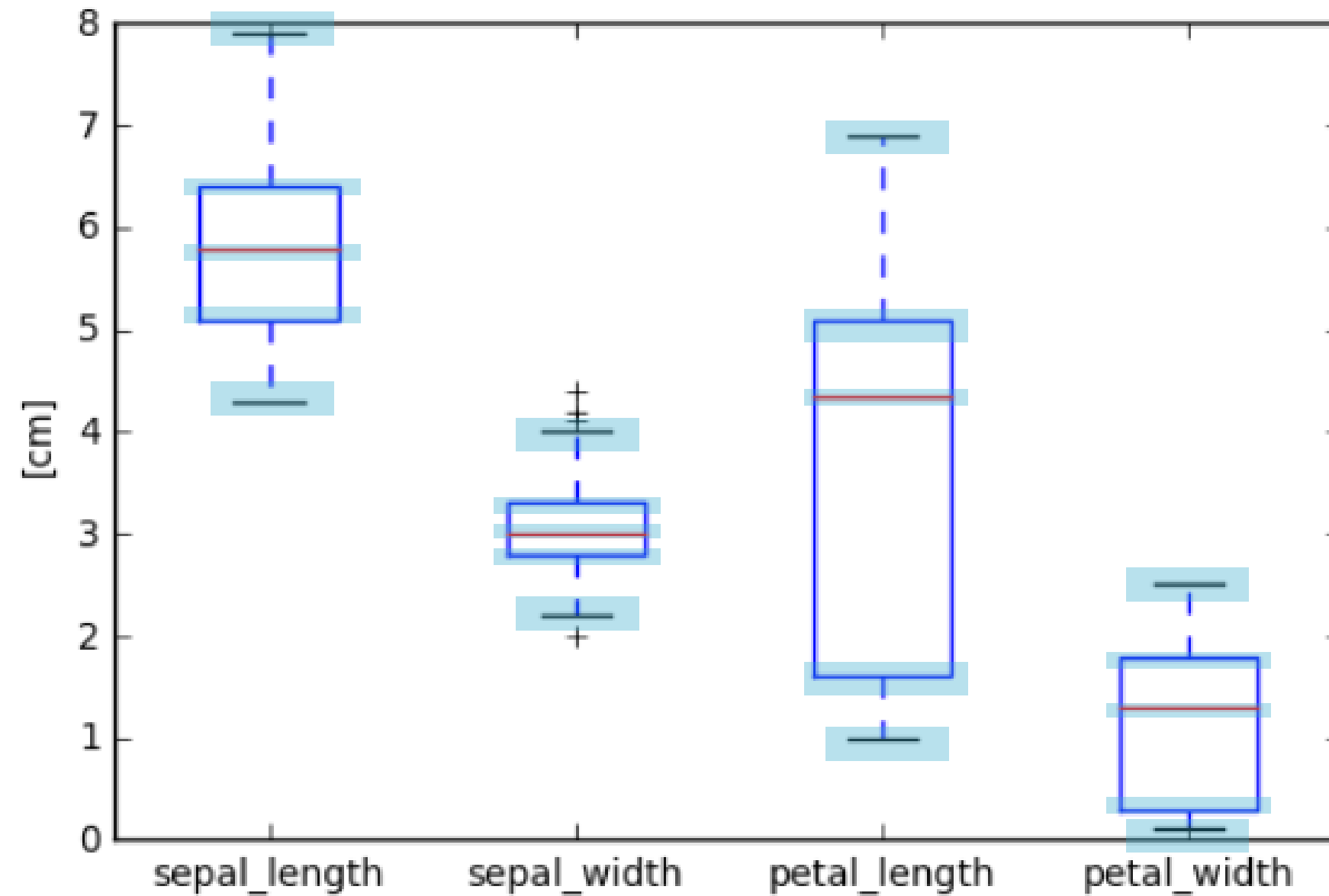
```
<matplotlib.axes._subplots.AxesSubplot at 0x118a3d5f8>
```

```
plt.ylabel( ' [cm]' )
```

```
<matplotlib.text.Text at 0x118a524e0>
```

```
plt.show()
```

Box plots



Percentiles as quantiles

```
iris.describe() # summary statistics
```

	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

Let's practice!

PANDAS FOUNDATIONS

Separating populations

PANDAS FOUNDATIONS



Dhavide Aruliah

Director of Training, Anaconda

```
iris.head()
```

	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

```
iris['species'].describe()
```

```
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

```
iris['species'].unique()
```

```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

Filtering by species

```
indices = iris['species'] == 'setosa'
setosa = iris.loc[indices,:] # extract new DataFrame
indices = iris['species'] == 'versicolor'
versicolor = iris.loc[indices,:] # extract new DataFrame
indices = iris['species'] == 'virginica'
virginica = iris.loc[indices,:] # extract new DataFrame
```

Checking species

```
setosa['species'].unique()
```

```
array(['setosa'], dtype=object)
```

```
versicolor['species'].unique()
```

```
array(['versicolor'], dtype=object)
```

```
virginica['species'].unique()
```

```
array(['virginica'], dtype=object)
```

```
del setosa['species'], versicolor['species'],  
     virginica['species']
```

Checking indexes

```
setosa.head(2)
```

	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

```
versicolor.head(2)
```

	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

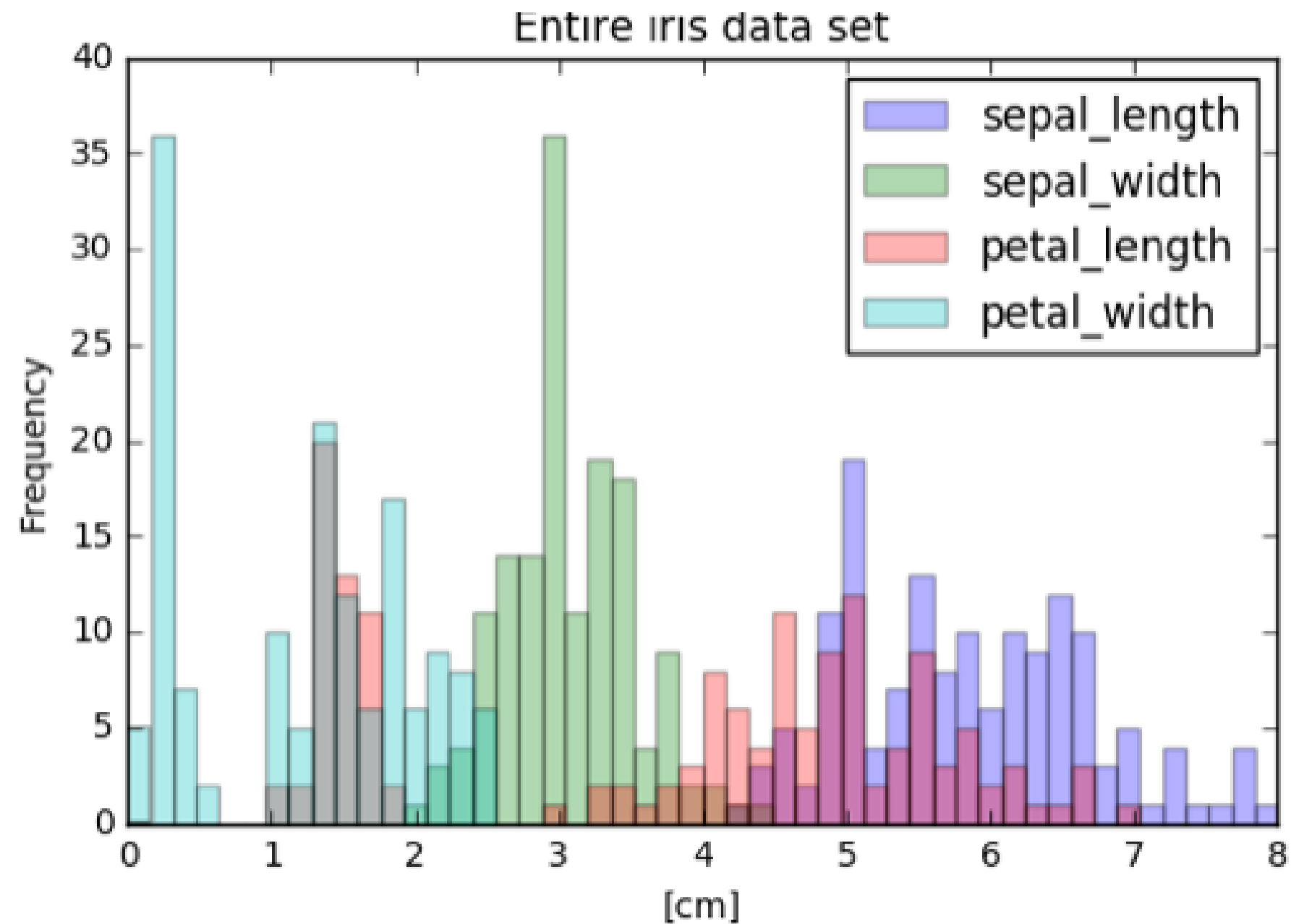
```
versicolor.head(2)
```

	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

```
iris.plot(kind= 'hist', bins=50, range=(0,8), alpha=0.3)  
plt.title('Entire iris data set')  
plt.xlabel('[cm]')  
plt.show()
```

Visual EDA: all data



Visual EDA: individual factors

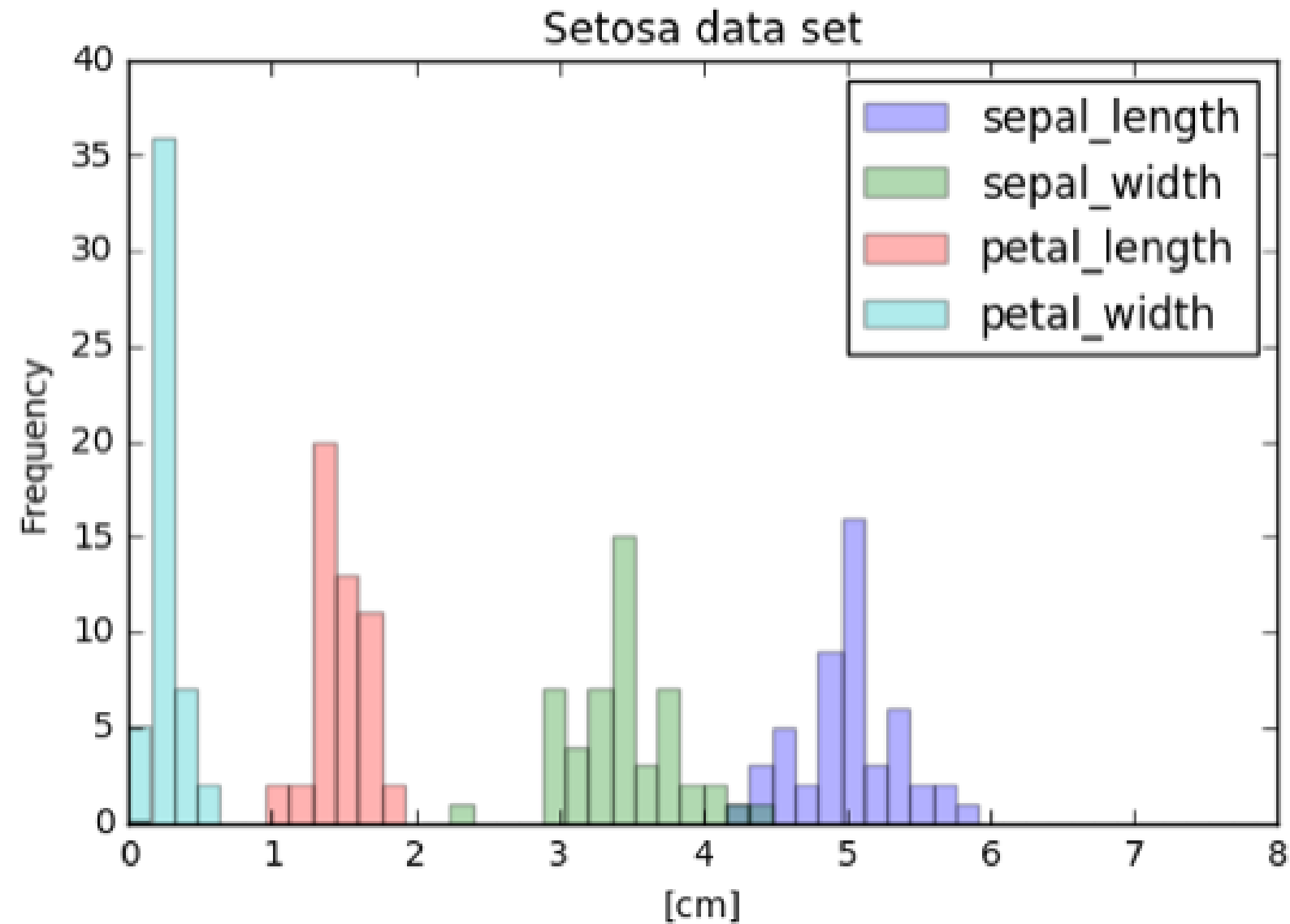
```
setosa.plot(kind='hist', bins=50, range=(0,8), alpha=0.3)
plt.title('Setosa data set')
plt.xlabel('[cm]')

versicolor.plot(kind='hist', bins=50, range=(0,8), alpha=0.3)
plt.title('Versicolor data set')
plt.xlabel('[cm]')

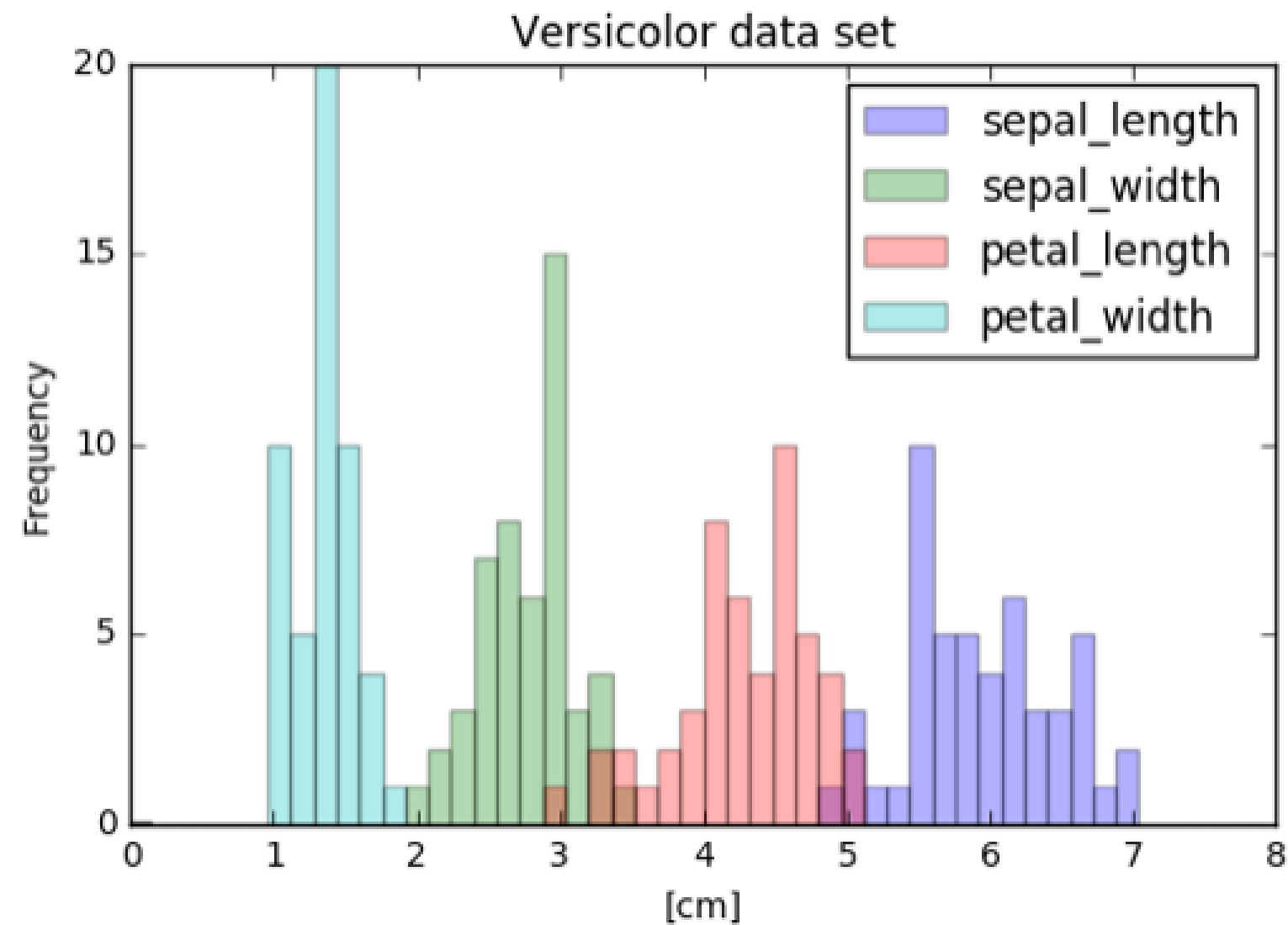
virginica.plot(kind='hist', bins=50, range=(0,8), alpha=0.3)
plt.title('Virginica data set')
plt.xlabel('[cm]')

plt.show()
```

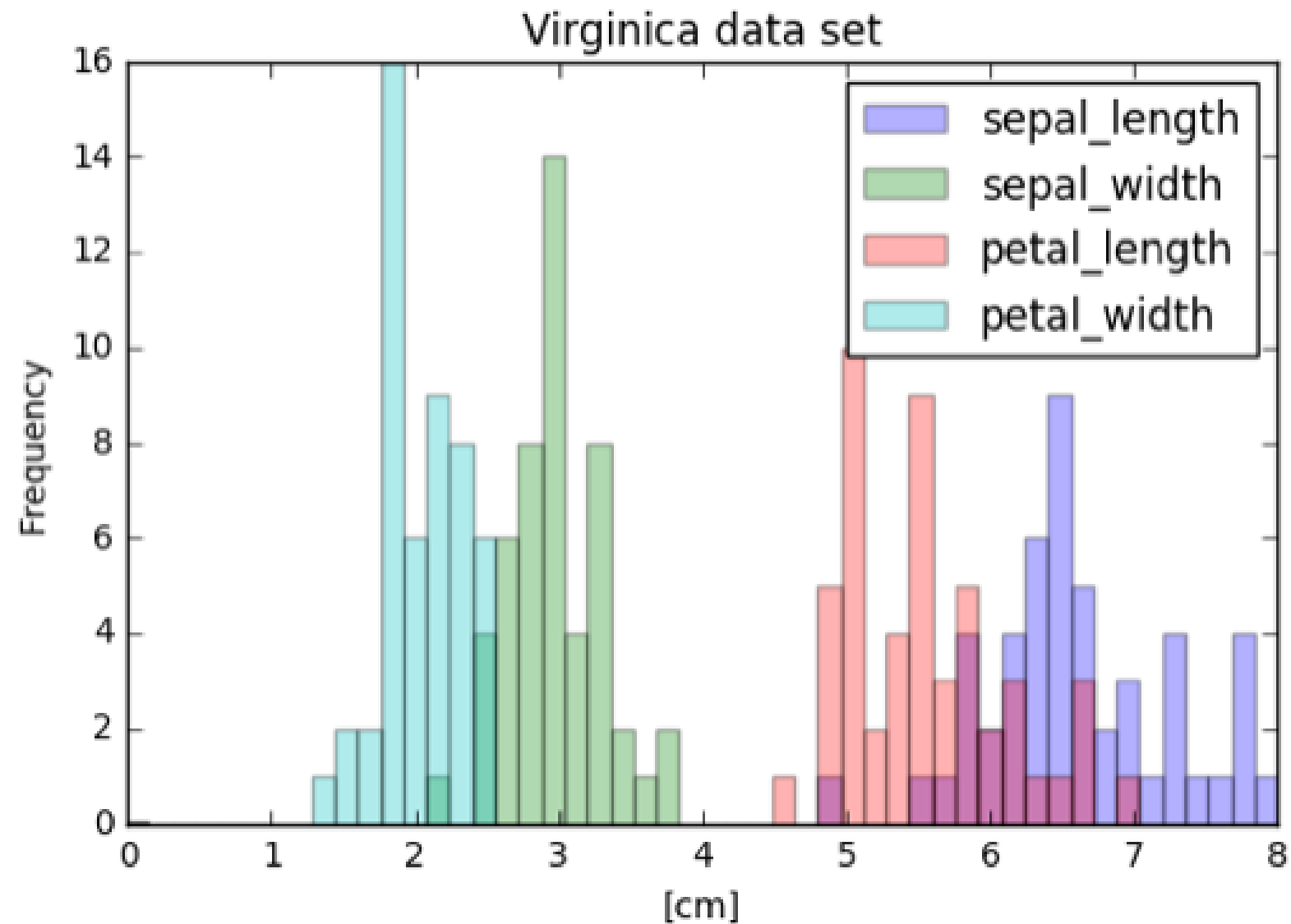
Visual EDA: Setosa data



Visual EDA: Versicolor data



Visual EDA: Virginica data



Statistical EDA: describe()

```
describe_all = iris.describe()  
print(describe_all)
```

	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_setosa = setosa.describe()  
describe_versicolor = versicolor.describe()  
describe_virginica = virginica.describe()
```

Computing errors

```
error_setosa = 100 * np.abs(describe_setosa - describe_all)
error_setosa = error_setosa/describe_setosa

error_versicolor = 100 * np.abs(describe_versicolor - describe_all)
error_versicolor = error_versicolor/describe_versicolor

error_virginica = 100 * np.abs(describe_virginica - describe_all)
error_virginica = error_virginica/describe_virginica
```


Viewing errors

```
print(error_setosa)
```

	sepal_length	sepal_width	petal_length	petal_width
count	200.000000	200.000000	200.000000	200.000000
mean	16.726595	10.812913	157.045144	387.533875
std	134.919250	14.984768	916.502136	623.284534
min	0.000000	13.043478	0.000000	0.000000
25%	6.250000	12.500000	14.285714	50.000000
50%	16.000000	11.764706	190.000000	550.000000
75%	23.076923	10.204082	223.809524	500.000000
max	36.206897	0.000000	263.157895	316.666667

Let's practice!

PANDAS FOUNDATIONS