

# Statistica descrittiva, visualizzazione

July 29, 2023

## Contents

<b>1</b>	<b>Setup</b>	<b>1</b>
<b>2</b>	<b>Info generali</b>	<b>2</b>
<b>3</b>	<b>Univariate</b>	<b>2</b>
3.1	Variabili numeriche . . . . .	2
3.1.1	Statistiche numeriche . . . . .	2
3.1.2	Istogramma . . . . .	3
3.2	Categoriche . . . . .	3
3.2.1	Frequenze . . . . .	3
3.2.2	Diagramma a barre . . . . .	4
<b>4</b>	<b>Bivariate</b>	<b>5</b>
4.1	Numeriche stratificate . . . . .	5
4.2	Tabelle di contingenza . . . . .	6
4.3	Tabella trial . . . . .	6
4.4	Correlazione . . . . .	7
4.5	Grafici . . . . .	8
4.5.1	Scatterplot . . . . .	8
4.5.2	Matrice di scatterplot . . . . .	10
4.5.3	Boxplot . . . . .	10

## 1 Setup

Importiamo le librerie qui usate

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pylbmisc as lb
```

```
# Per la visualizzazione utilizziamo seaborn. un po di setup
import seaborn as sns
```

```
sns.set_theme()
```

```
# dataset
iris = sns.load_dataset('iris')
```

Queste importazioni doppie servono per gli ambienti pyconsole, isolati dal resto:

```
>>> import numpy as np
>>> import pandas as pd
```

## 2 Info generali

```
>>> df = pd.DataFrame(np.random.randn(1000, 5), columns=["a", "b", "c", "d", "e"])
>>> df[:,2] = np.nan
>>> df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  -
 0    a      500 non-null    float64
 1    b      500 non-null    float64
 2    c      500 non-null    float64
 3    d      500 non-null    float64
 4    e      500 non-null    float64
dtypes: float64(5)
memory usage: 39.2 KB
>>> df.head()
   a      b      c      d      e
0  NaN    NaN    NaN    NaN    NaN
1  0.202498  0.672204 -0.413379  0.220610  0.090997
2  NaN    NaN    NaN    NaN    NaN
3  0.866847 -1.522836  1.010345  0.121939  0.403454
4  NaN    NaN    NaN    NaN    NaN
```

## 3 Univariate

### 3.1 Variabili numeriche

#### 3.1.1 Statistiche numeriche

Usare il metodo `describe` e fare trasposizione.

```
>>> df = pd.DataFrame(np.random.randn(1000, 5), columns=["a", "b", "c", "d", "e"])
>>> df.describe().transpose() # count sono i valori non mancanti (qui tutti)
```

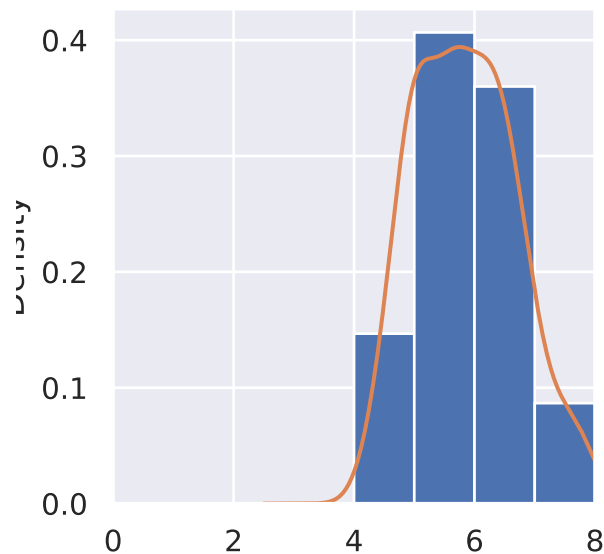


Figure 1: Istogramma (pandas syntax)

	count	mean	std	min	25%	50%	75%	max
a	1000.0	-0.017543	1.007334	-3.523030	-0.705765	0.067024	0.685625	2.744069
b	1000.0	-0.023202	0.949985	-3.217975	-0.681230	0.008212	0.612476	2.893202
c	1000.0	0.014019	0.997395	-2.836318	-0.646395	0.046285	0.727336	2.856179
d	1000.0	0.002391	1.000984	-3.803654	-0.654873	0.060738	0.686495	3.166527
e	1000.0	0.024085	1.037603	-2.913289	-0.689868	0.047890	0.745991	3.563289

### 3.1.2 Istogramma

Utilizzando i metodi di pandas in figura 1

```
ax = iris.sepal_length.plot.hist(density = True, bins = range(9), xlim = [0, 8])
iris.sepal_length.plot.density(ax = ax)
fig = ax.get_figure()
lb.io.fig_dump(fig, label="pandas_histo", caption = 'Istogramma (pandas syntax)')
plt.close()
```

## 3.2 Categorie

### 3.2.1 Frequenze

```
>>> df = pd.DataFrame({"x": ["a", "a", "a", "a", np.nan, "b", "b"],
...                     "y": ["1", "2", "1", "2", "2", "1", "2"]})
```

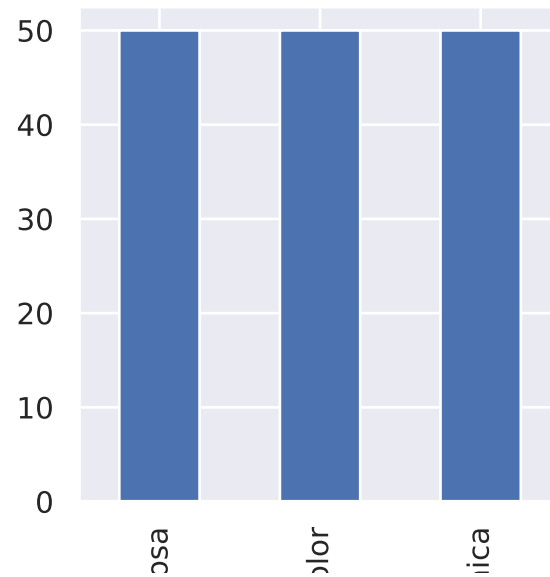


Figure 2: Diagramma a barre

```
>>> df.x.value_counts()
x
a      4
b      2
Name: count, dtype: int64
>>> df.x.value_counts(dropna=False)
x
a      4
b      2
NaN     1
Name: count, dtype: int64
```

### 3.2.2 Diagramma a barre

In figura 2 come farlo con pandas

```
# grafico non molto significativo ma qui guardiamo sintassi
ax = iris.species.value_counts().plot.bar()
fig = ax.get_figure()
lb.io.fig_dump(fig, label="pandas_barre", caption = 'Diagramma a barre')
plt.close()
```

## 4 Bivariate

### 4.1 Numeriche stratificate

```
>>> df = pd.DataFrame({"x": np.random.randn(7),
...                     "y": np.random.randn(7),
...                     "z": np.random.randn(7),
...                     "g": ["trt", "ctrl", "trt", "ctrl", "trt", "ctrl", "trt"]})

>>> spl = df.groupby("g")
>>> spl.describe().transpose() # descrizione complessiva
```

		ctrl	trt
x	count	3.000000	4.000000
	mean	-0.026651	-0.341753
	std	1.358654	1.390846
	min	-1.506627	-1.787266
	25%	-0.622029	-1.315553
	50%	0.262570	-0.431645
	75%	0.713338	0.542155
y	count	3.000000	4.000000
	mean	0.371821	-0.432336
	std	0.590360	1.570421
	min	-0.241647	-2.173654
	25%	0.089741	-1.554124
	50%	0.421129	-0.226430
	75%	0.678555	0.895357
z	count	3.000000	4.000000
	mean	1.584549	-0.300830
	std	0.920410	1.834051
	min	0.522551	-2.692119
	25%	1.301190	-1.217268
	50%	2.079828	0.022242
	75%	2.115547	0.938679
	max	2.151266	1.444315

```
>>> sel = ["x", "z"] # descrizione di solo alcune colonne
>>> spl[sel].describe().transpose()
```

		ctrl	trt
x	count	3.000000	4.000000
	mean	-0.026651	-0.341753
	std	1.358654	1.390846
	min	-1.506627	-1.787266
	25%	-0.622029	-1.315553
	50%	0.262570	-0.431645
	75%	0.713338	0.542155

```

max      1.164105  1.283544
z count  3.000000  4.000000
mean     1.584549 -0.300830
std      0.920410  1.834051
min      0.522551 -2.692119
25%      1.301190 -1.217268
50%      2.079828  0.022242
75%      2.115547  0.938679
max      2.151266  1.444315

```

## 4.2 Tabelle di contingenza

```

>>> df = pd.DataFrame({"x": ["a", "a", "a", "a", "a", "b", "b"],
...                     "y": ["1", "2", "2", "2", "2", "1", "2"],
...                     "g": ["trt", "ctrl", "trt", "ctrl", "trt", "ctrl", "trt"]})

>>> pd.crosstab(df.x, df.y, margins=True) # frequenze schiette con totali
y      1  2  All
x
a      1  4    5
b      1  1    2
All    2  5    7

>>> pd.crosstab(df.x, df.y, margins=True, normalize = 'columns') # percentuali di colonna
y      1    2    All
x
a  0.5  0.8  0.714286
b  0.5  0.2  0.285714

```

## 4.3 Tabella trial

Utilizzare la libreria `tableone`.

```

>>> import tableone
>>> df = tableone.load_dataset('pn2012')
>>> df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Age         1000 non-null   int64
1   SysABP      709 non-null    float64
2   Height      525 non-null    float64
3   Weight      698 non-null    float64
4   ICU         1000 non-null   object
5   MechVent    1000 non-null   int64

```

```

6   LOS          1000 non-null   int64
7   death        1000 non-null   int64
dtypes: float64(3), int64(4), object(1)
memory usage: 62.6+ KB
>>> df.head()
   Age  SysABP  Height  Weight  ICU  MechVent  LOS  death
0   54     NaN     NaN     NaN  SICU         0     5     0
1   76   105.0   175.3    80.6  CSRU         1     8     0
2   44   148.0     NaN    56.7  MICU         0    19     0
3   68     NaN   180.3    84.6  MICU         0     9     0
4   88     NaN     NaN     NaN  MICU         0     4     0
>>> ft = {0: "alive", 1: "dead"}
>>> df["group"] = pd.Categorical(df.death.map(ft))
>>> select = ['Age', 'SysABP', 'Height', 'Weight', 'ICU', 'group']
>>> categ = ['ICU', 'group']
>>> groupby = ['group']
>>> nonnormal = ['Age']
>>> labels={'death': 'mortality'}
>>> tab1 = tableone.TableOne(df,
...                          columns=select,
...                          categorical=categ,
...                          groupby=groupby,
...                          nonnormal=nonnormal,
...                          rename=labels,
...                          pval=False)
>>> tab1

```

		Grouped by group					
		Missing	Overall	alive		dead	
n			1000	864		136	
Age, median [Q1,Q3]		0	68.0 [53.0,79.0]	66.0 [52.8,78.0]		75.0 [62.0,88.0]	
SysABP, mean (SD)		291	114.3 (40.2)	115.4 (38.3)		107.6 (40.2)	
Height, mean (SD)		475	170.1 (22.1)	170.3 (23.2)		168.5 (21.8)	
Weight, mean (SD)		302	82.9 (23.8)	83.0 (23.6)		82.3 (23.2)	
ICU, n (%)	CCU	0	162 (16.2)	137 (15.9)		25 (18.4)	
	CSRU		202 (20.2)	194 (22.5)		8 (6.0)	
	MICU		380 (38.0)	318 (36.8)		62 (45.6)	
	SICU		256 (25.6)	215 (24.9)		41 (30.0)	
group, n (%)	alive	0	864 (86.4)	864 (100.0)			
	dead		136 (13.6)			136 (100.0)	

## 4.4 Correlazione

Si usa il metodo `corr`

```

>>> df = pd.DataFrame(np.random.randn(1000, 5), columns=["a", "b", "c", "d", "e"])
>>> df.corr() # correlazione di pearson

```

```

      a      b      c      d      e
a  1.000000 -0.011736 -0.020552  0.047175  0.035546
b -0.011736  1.000000  0.009603  0.002779 -0.011595
c -0.020552  0.009603  1.000000  0.016710 -0.022531
d  0.047175  0.002779  0.016710  1.000000  0.020769
e  0.035546 -0.011595 -0.022531  0.020769  1.000000
>>> df.corr(method='spearman')
      a      b      c      d      e
a  1.000000  0.004511 -0.027011  0.052206  0.027015
b  0.004511  1.000000 -0.007078  0.014060 -0.001029
c -0.027011 -0.007078  1.000000  0.018606 -0.012641
d  0.052206  0.014060  0.018606  1.000000  0.019836
e  0.027015 -0.001029 -0.012641  0.019836  1.000000

```

## 4.5 Grafici

### 4.5.1 Scatterplot

Uno rapido con pandas in figura 3

```

ax = iris.plot(kind = 'scatter', x = 'sepal_length', y = 'sepal_width')
fig = ax.get_figure()
lb.io.fig_dump(fig, label="pandas_scatter", caption = 'Scatterplot (pandas)')
plt.close()

```

Uno più elaborato con colorazione condizionale, alpha shading e fatto con seaborn in figura 4

```

# data
group1 = pd.DataFrame({'x': np.random.normal(10, 1.2, 2000),
                       'y': np.random.normal(10, 1.2, 2000),
                       'group': np.repeat('A',2000) })
group2 = pd.DataFrame({'x': np.random.normal(14.5, 1.2, 2000),
                       'y': np.random.normal(14.5, 1.2, 2000),
                       'group': np.repeat('B',2000) })
group3 = pd.DataFrame({'x': np.random.normal(9.5, 1.5, 2000),
                       'y': np.random.normal(15.5, 1.5, 2000),
                       'group': np.repeat('C',2000) })
df = pd.concat([group1, group2, group3])

# Plot
ax = sns.scatterplot(x='x', y='y', data=df, hue='group', alpha = 0.30)
fig = ax.get_figure()
lb.io.fig_dump(fig, label="sns_scatter", caption = 'Scatterplot')
plt.close()

```



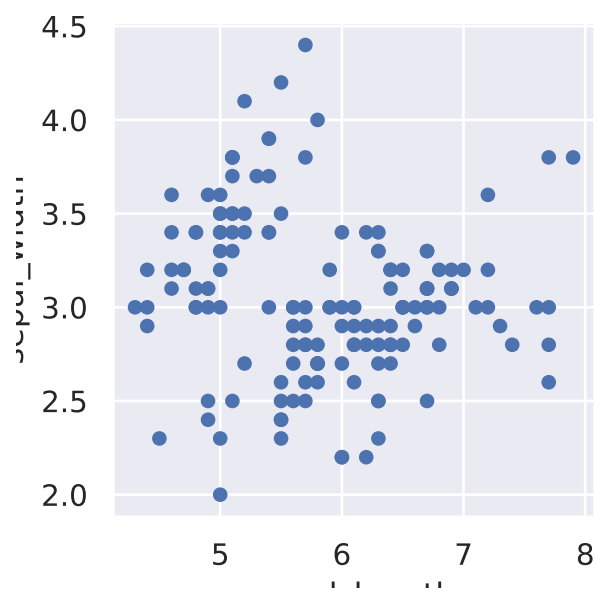


Figure 3: Scatterplot (pandas)

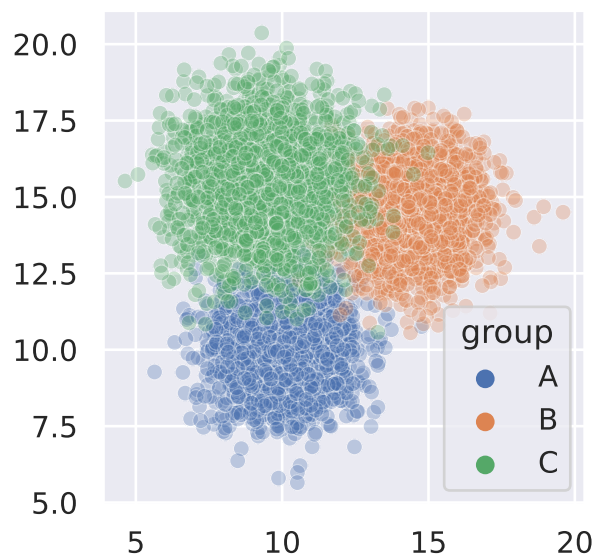


Figure 4: Scatterplot

### 4.5.2 Matrice di scatterplot

Invece per la matrice di scatterplot ne mettiamo senza con regressione (5 e con colorazioni 6.

```
# primo
plot = sns.pairplot(iris, kind="reg")
fig = plot.fig
lb.io.fig_dump(fig, label="sns_pair1", caption = 'Pairplot 1', scale = 0.5)
plt.close()
```

```
# secondo
plot = sns.pairplot(iris, kind="scatter", hue="species", markers=["o", "s", "D"], palette="S")
fig = plot.fig
lb.io.fig_dump(fig, label="sns_pair2", caption = 'Pairplot 2', scale = 0.5)
plt.close()
```

### 4.5.3 Boxplot

In figura 7

```
ax = iris.boxplot(by="species", column="sepal_length")
fig = ax.get_figure()
lb.io.fig_dump(fig, label="pandas_boxplot", caption = 'Boxplot')
plt.close()
```

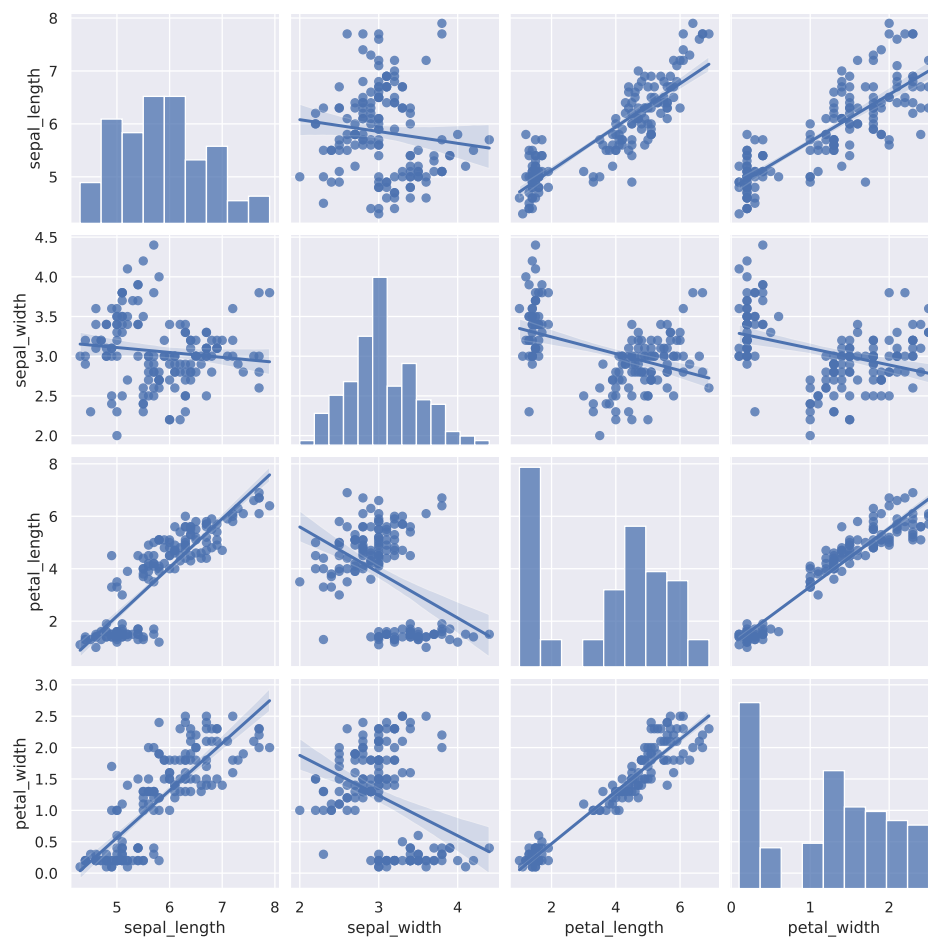


Figure 5: Pairplot 1

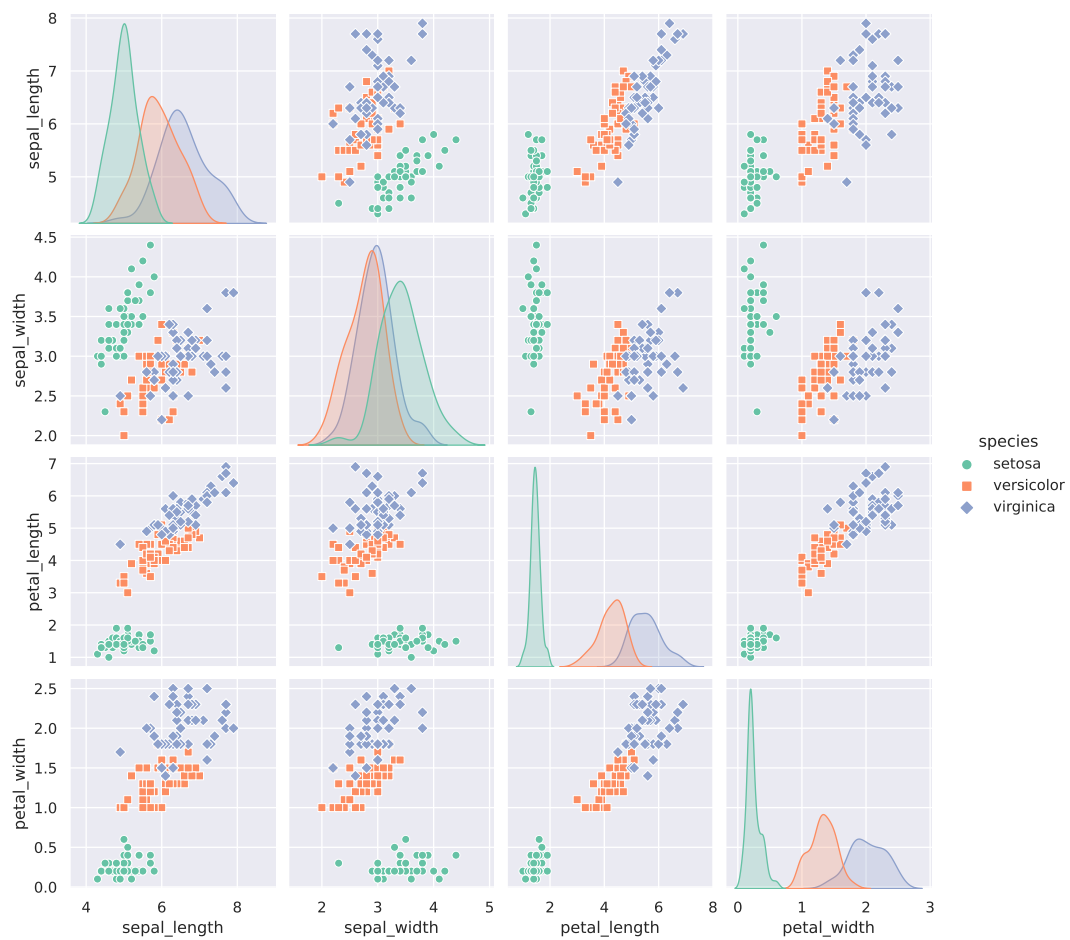


Figure 6: Pairplot 2

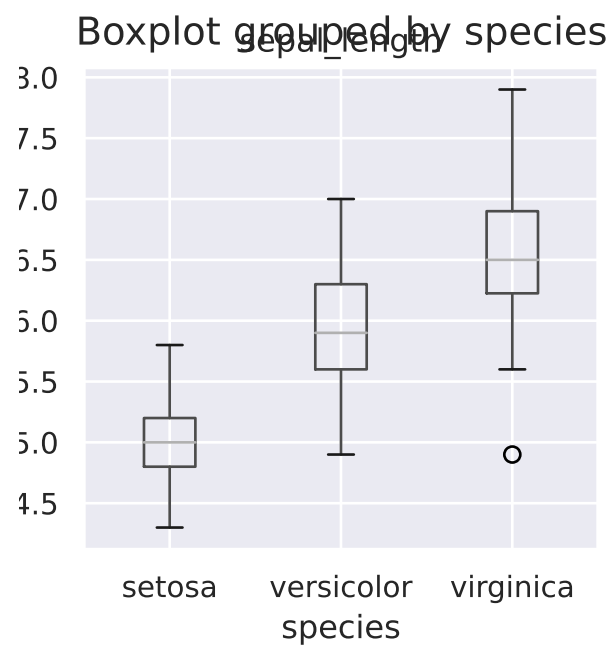


Figure 7: Boxplot