June 6, 2023

Contents

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 -0.211667  0.229882  1.057124 -0.786242  0.974781
2       NaN       NaN       NaN       NaN       NaN
3 -0.571312 -0.327726 -0.849239  1.312916  0.442238
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)
    count      mean       std       min       25%       50%       75%       max
a  1000.0 -0.092920  0.959500 -3.371739 -0.700707 -0.078666  0.548636  2.991837
```

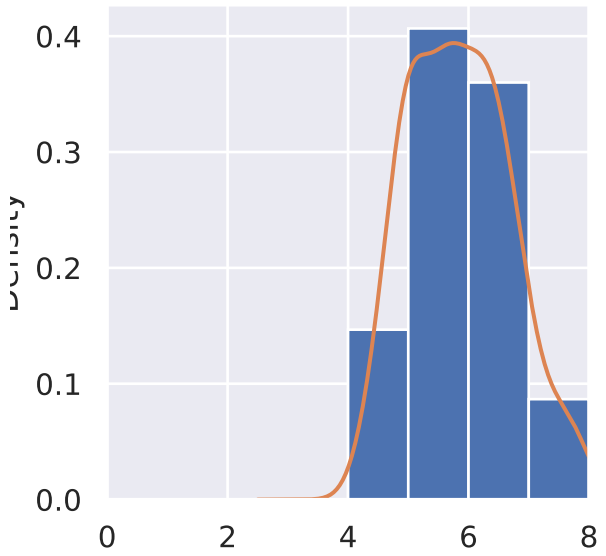

Figure 1: Istogramma (pandas syntax)

```
b  1000.0 -0.002984  0.994890 -4.152744 -0.643738  0.019054  0.659725  3.017753
c  1000.0 -0.003731  0.980569 -2.821368 -0.672108 -0.016166  0.652769  3.676547
d  1000.0  0.025283  0.987914 -2.865702 -0.648626  0.068483  0.717604  3.287291
e  1000.0  0.032527  1.020073 -3.278659 -0.665440  0.030570  0.736676  2.802641
```

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.fig.dump(fig, label="pandas_histo", caption = 'Istogramma (pandas syntax)')
plt.close()
```

3.2 Categoriche

3.2.1 Frequenze

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

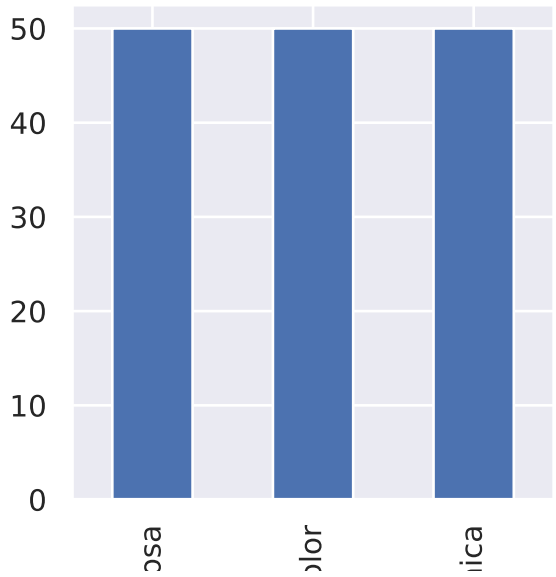

Figure 2: Diagramma a barre

```
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.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 complesiva
g            ctrl       trt
x count  3.000000  4.000000
  mean   0.576626 -0.205472
  std    0.209123  0.924879
  min    0.342701 -1.389003
  25%    0.492204 -0.545312
  50%    0.641707 -0.145197
  75%    0.693588  0.194642
  max    0.745470  0.857508
y count  3.000000  4.000000
  mean  -0.072108  0.383397
  std    0.700354  0.723242
  min   -0.798797 -0.488780
  25%   -0.407434 -0.054193
  50%   -0.016070  0.453803
  75%    0.291237  0.891393
  max    0.598544  1.114763
z count  3.000000  4.000000
  mean   0.897449 -0.273350
  std    0.809657  0.527357
  min    0.147814 -1.032992
  25%    0.468131 -0.415508
  50%    0.788448 -0.104635
  75%    1.272267  0.037522
  max    1.756086  0.148863
>>> sel = ["x","z"] # descrizione di solo alcune colonne
>>> spl[sel].describe().transpose()
g            ctrl       trt
x count  3.000000  4.000000
  mean   0.576626 -0.205472
  std    0.209123  0.924879
  min    0.342701 -1.389003
  25%    0.492204 -0.545312
  50%    0.641707 -0.145197
  75%    0.693588  0.194642
```

```
  max    0.745470  0.857508
z count  3.000000  4.000000
  mean   0.897449 -0.273350
  std    0.809657  0.527357
  min    0.147814 -1.032992
  25%    0.468131 -0.415508
  50%    0.788448 -0.104635
  75%    1.272267  0.037522
  max    1.756086  0.148863
```

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
n                                                      1000               864
Age, median [Q1,Q3]                    0  68.0 [53.0,79.0]  66.0 [52.8,78.0]  75.0 [62.0,8
SysABP, mean (SD)                    291      114.3 (40.2)      115.4 (38.3)      107.6 (4
Height, mean (SD)                    475      170.1 (22.1)      170.3 (23.2)      168.5 (1
Weight, mean (SD)                    302       82.9 (23.8)       83.0 (23.6)       82.3 (2
ICU, n (%)          CCU                0       162 (16.2)        137 (15.9)         25 (1
                    CSRU                       202 (20.2)        194 (22.5)          8 (
                    MICU                       380 (38.0)        318 (36.8)         62 (4
                    SICU                       256 (25.6)        215 (24.9)         41 (3
group, n (%)        alive              0       864 (86.4)       864 (100.0)
                    dead                       136 (13.6)                          136 (10
```

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.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.fig.dump(fig, label="sns_scatter", caption = 'Scatterplot')
plt.close()
```

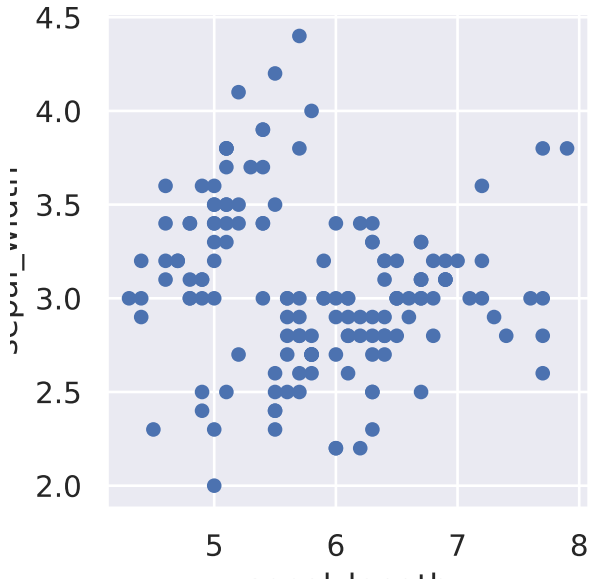

Figure 3: Scatterplot (pandas)

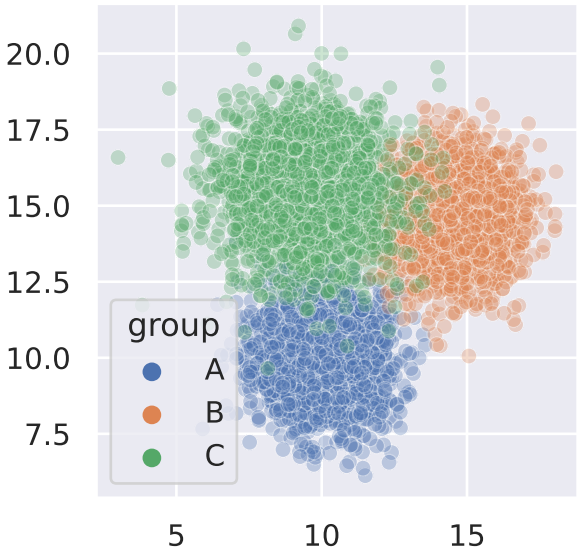

Figure 4: Scatterplot

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.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.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.fig.dump(fig, label="pandas_boxplot", caption = 'Boxplot')
plt.close()
```

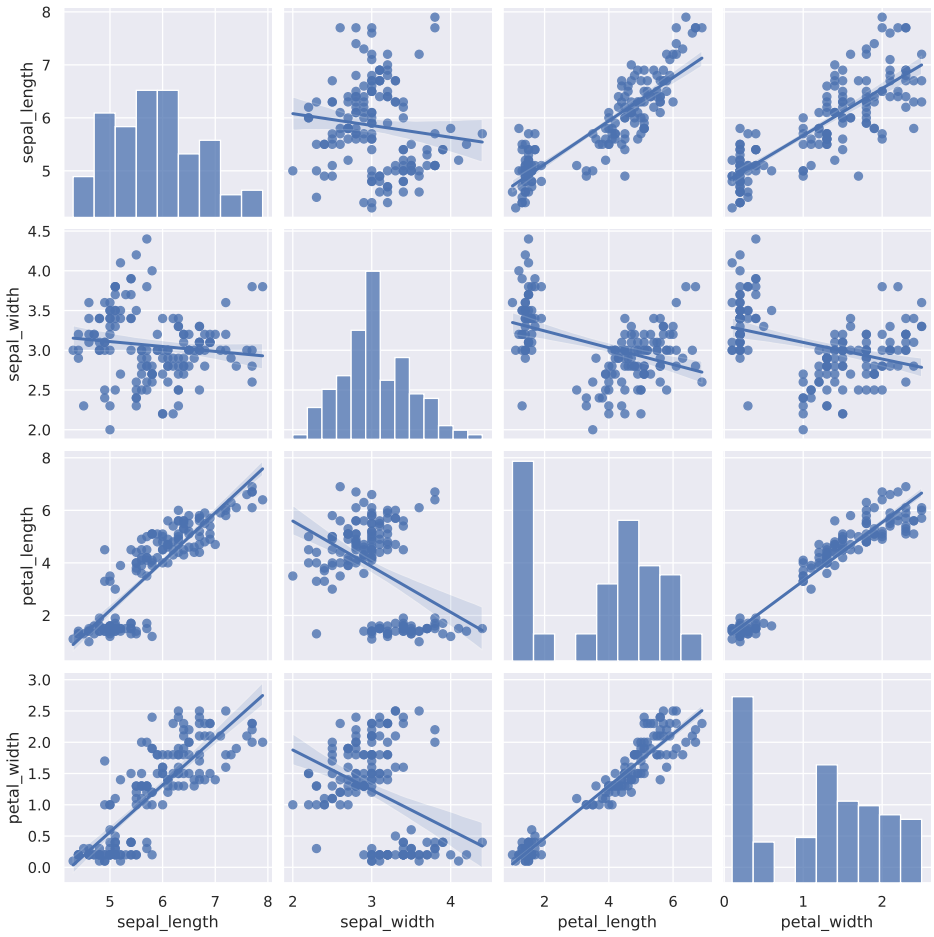

Figure 5: Pairplot 1

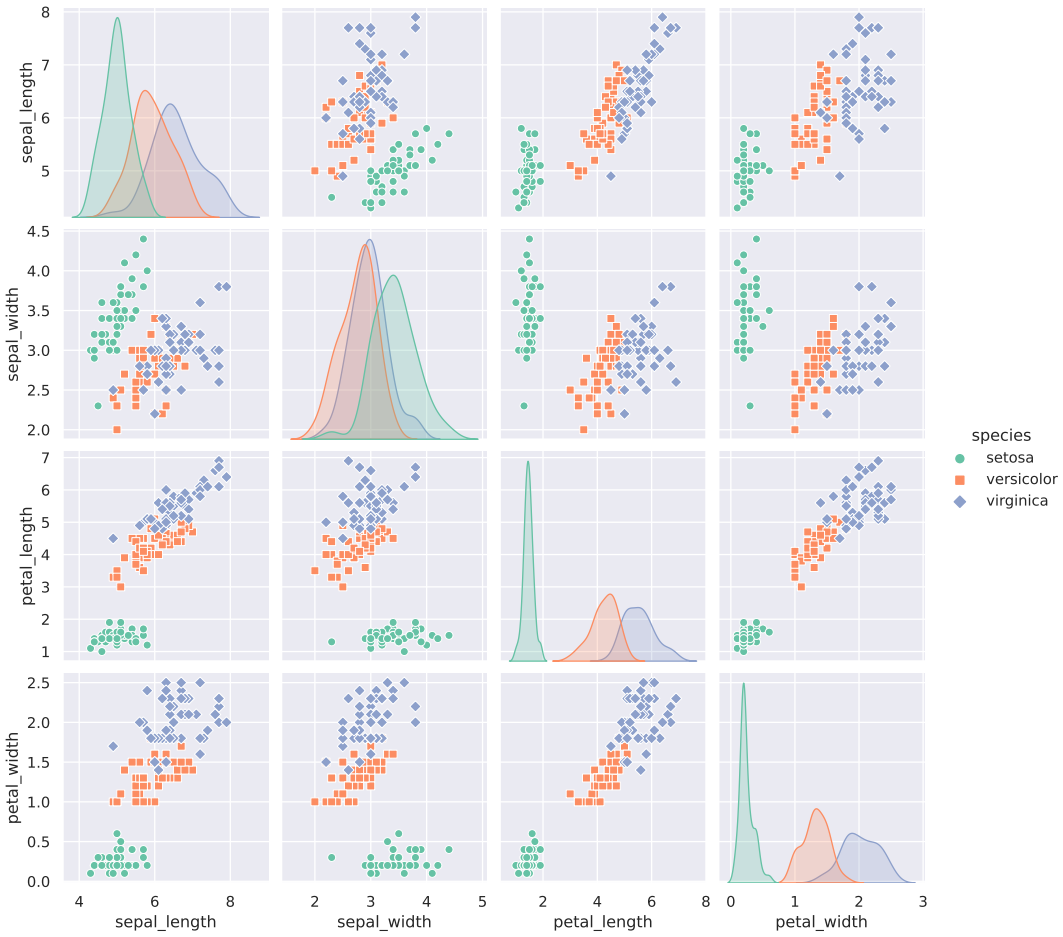

Figure 6: Pairplot 2

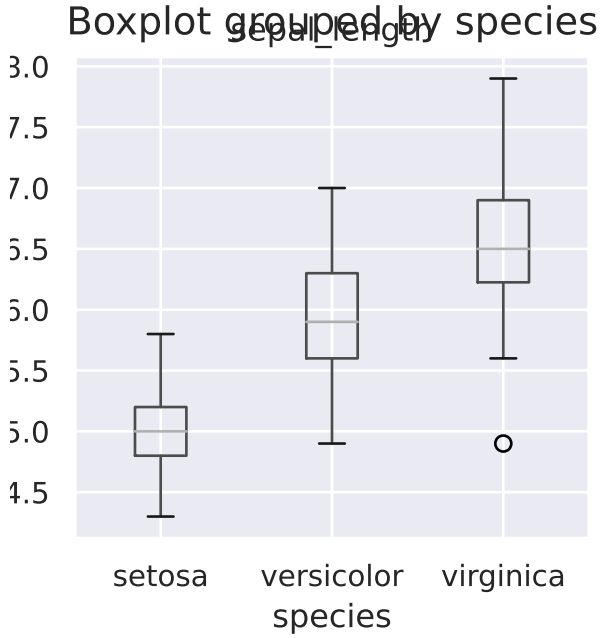

Figure 7: Boxplot