June 2, 2023

Contents

1 Setup

Importiamo le librerie qui usate

```
import numpy as np
import pandas as pd
import pylbmisc as lb

# Per la visualizzazione utilizziamo seaborn. un po di setup
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme()
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  1.570055 -0.393051  0.880055 -0.032189 -0.161610
2       NaN       NaN       NaN       NaN       NaN
3 -1.392931 -0.274947  0.719258  0.093073 -1.661034
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  0.004130  1.062286 -3.621185 -0.643718 -0.040742  0.693380  3.206745
b  1000.0  0.000216  1.007270 -3.297063 -0.666956  0.054856  0.671322  3.084294
c  1000.0 -0.002650  0.994217 -3.316414 -0.623244 -0.020238  0.614408  3.665185
d  1000.0 -0.053436  1.023828 -3.017280 -0.745594 -0.025781  0.628180  3.064176
e  1000.0 -0.038803  1.016394 -3.472263 -0.726240 -0.046140  0.633126  2.721213
```

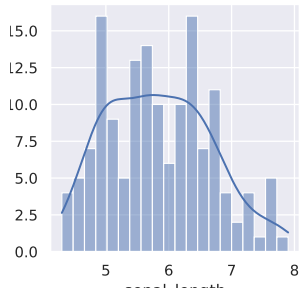

Figure 1: Istogramma

3.1.2 Istogramma

In figura 1.

```
# plot = sns.histplot(data=iris, x="sepal_length", kde=True, bins=20)
plot = sns.histplot(x=iris.sepal_length, kde=True, bins=20)
fig = plot.get_figure()
lb.fig.dump(fig, label="sns_histo", caption = 'Istogramma', scale = 0.5)
plt.clf() # pulire figura se no la ritroviamo sovrapposta
```

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
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

Figura **??**

```
tips = sns.load_dataset("tips")
plot = sns.barplot(
```

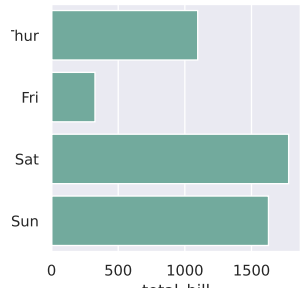

Figure 2: Diagramma a barre

```
    x="total_bill",
    y="day",
    data=tips,
    estimator=sum,
    ci=None,
    color='#69b3a2');
fig = plot.get_figure()
lb.fig.dump(fig, label="sns_barre", caption = 'Diagramma a barre', scale = 0.5)
plt.clf()
```

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   0.232905  0.207106
  std    1.053657  1.007551
  min   -0.790423 -1.142858
  25%   -0.307880 -0.093417
  50%    0.174663  0.340478
  75%    0.744569  0.641000
  max    1.314474  1.290324
y count  3.000000  4.000000
```

```
  mean   0.287371 -0.938949
  std    0.986902  0.778552
  min   -0.794455 -1.824555
  25%   -0.138180 -1.445004
  50%    0.518094 -0.918370
  75%    0.828283 -0.412315
  max    1.138472 -0.094499
z count  3.000000  4.000000
  mean   0.564759 -0.103995
  std    1.168875  1.536222
  min   -0.693046 -2.195190
  25%    0.038344 -0.768697
  50%    0.769734  0.277340
  75%    1.193661  0.942042
  max    1.617589  1.224528
>>> sel = ["x","z"] # descrizione di solo alcune colonne
>>> spl[sel].describe().transpose()
g            ctrl       trt
x count  3.000000  4.000000
  mean   0.232905  0.207106
  std    1.053657  1.007551
  min   -0.790423 -1.142858
  25%   -0.307880 -0.093417
  50%    0.174663  0.340478
  75%    0.744569  0.641000
  max    1.314474  1.290324
z count  3.000000  4.000000
  mean   0.564759 -0.103995
  std    1.168875  1.536222
  min   -0.693046 -2.195190
  25%    0.038344 -0.768697
  50%    0.769734  0.277340
  75%    1.193661  0.942042
  max    1.617589  1.224528
```

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 con colorazione condizionale e alpha shading in figura 3

```
# data
group1 = pd.DataFrame({'x': np.random.normal(10, 1.2, 2000),
```

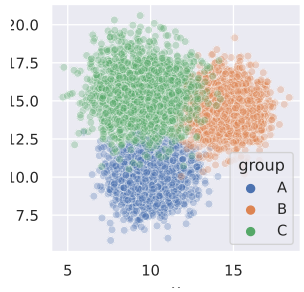

Figure 3: Scatterplot

```
                        '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
plot = sns.scatterplot(x='x', y='y', data=df, hue='group', alpha = 0.30)
fig = plot.get_figure()
lb.fig.dump(fig, label="sns_scatter", caption = 'Scatterplot', scale = 0.5)
plt.clf()
```

Invece per la matrice di scatterplot ne mettiamo senza con regressione (**??**
e con colorazioni **??**

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

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

4.5.2 Boxplot

In figura 4.

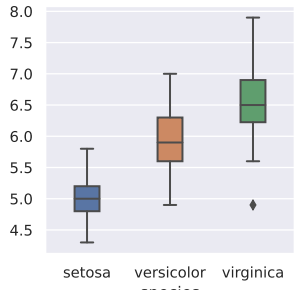

Figure 4: Boxplot

```
plot = sns.boxplot(x=iris.species, y=iris.sepal_length, width=0.3)
fig = plot.get_figure()
lb.fig.dump(fig, label="sns_boxplot", caption = 'Boxplot', scale = 0.5)
plt.clf()
```