May 31, 2023

Contents


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

1 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  0.235411 -0.463874 -0.430522  0.828744  0.440821
2       NaN       NaN       NaN       NaN       NaN
3  1.776615 -1.128182  1.483508 -0.747460 -1.359850
4       NaN       NaN       NaN       NaN       NaN
```

2 Statistiche descrittive variabili 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  0.022990  0.979112 -3.430362 -0.654614  0.022158  0.689883  2.935128
b  1000.0 -0.023109  0.999222 -3.539961 -0.668722 -0.061274  0.644795  2.967478
c  1000.0  0.000803  0.990264 -3.192919 -0.619020  0.023890  0.625251  3.129880
d  1000.0  0.025380  0.991049 -2.893172 -0.616749  0.041925  0.659904  3.311276
e  1000.0  0.003476  1.032341 -3.080920 -0.720428 -0.001180  0.711563  3.504571
```

3 Frequenze univariate

```
>>> 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
```

4 Crosstabs

```
>>> 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
```

5 Statistiche 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  -0.149295 -0.243868
  std    0.086682  0.535244
  min   -0.248234 -0.956210
  25%   -0.180588 -0.432725
  50%   -0.112942 -0.174118
  75%   -0.099825  0.014738
  max   -0.086708  0.328973
y count  3.000000  4.000000
  mean  -0.291136  0.564125
  std    0.169549  1.084302
  min   -0.444591 -0.885877
  25%   -0.382143  0.272660
  50%   -0.319694  0.700991
  75%   -0.214408  0.992455
  max   -0.109121  1.740394
z count  3.000000  4.000000
  mean   0.206153  0.219954
  std    0.738612  0.281829
  min   -0.501237 -0.121585
  25%   -0.176999  0.063653
  50%    0.147238  0.234791
  75%    0.559849  0.391093
  max    0.972459  0.531820
>>> sel = ["x","z"] # descrizione di solo alcune colonne
```

```
>>> spl[sel].describe().transpose()
g            ctrl       trt
x count  3.000000  4.000000
  mean  -0.149295 -0.243868
  std    0.086682  0.535244
  min   -0.248234 -0.956210
  25%   -0.180588 -0.432725
  50%   -0.112942 -0.174118
  75%   -0.099825  0.014738
  max   -0.086708  0.328973
z count  3.000000  4.000000
  mean   0.206153  0.219954
  std    0.738612  0.281829
  min   -0.501237 -0.121585
  25%   -0.176999  0.063653
  50%    0.147238  0.234791
  75%    0.559849  0.391093
  max    0.972459  0.531820
```

6 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  0.006008  0.060552 -0.012971 -0.030461
b  0.006008  1.000000 -0.051051 -0.004034 -0.030964
c  0.060552 -0.051051  1.000000 -0.042863  0.015064
d -0.012971 -0.004034 -0.042863  1.000000 -0.000066
e -0.030461 -0.030964  0.015064 -0.000066  1.000000
>>> df.corr(method='spearman')
          a         b         c         d         e
a  1.000000  0.006481  0.064393 -0.004733 -0.029182
b  0.006481  1.000000 -0.043839  0.007552 -0.022224
c  0.064393 -0.043839  1.000000 -0.043188  0.022743
d -0.004733  0.007552 -0.043188  1.000000 -0.006911
e -0.029182 -0.022224  0.022743 -0.006911  1.000000
```

7 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