0023_pandas_stats

December 24, 2018

1 Basic statistics in Pandas

Like NumPy, Pandas may be used to give us some basic statistics on data.

Let's start by building a very sample dataframe.

```
In [1]: import pandas as pd
        df = pd.DataFrame()

        names = ['Gandolf','Gimli','Frodo','Legolas','Bilbo']
        types = ['Wizard','Dwarf','Hobbit','Elf','Hobbit']
        magic = [10, 1, 4, 6, 4]
        aggression = [7, 10, 2, 5, 2]
        stealth = [8, 2, 5, 10, None]


        df['names'] = names
        df['type'] = types
        df['magic_power'] = magic
        df['aggression'] = aggression
        df['stealth'] = stealth
```

1.1 Overview statistics

We can get an overview with the describe() method.

```
In [2]: print (df.describe())
```

```
       magic_power  aggression  stealth
count     5.000000    5.000000     4.00
mean      5.000000    5.200000     6.25
std       3.316625    3.420526     3.50
min       1.000000    2.000000     2.00
25%       4.000000    2.000000     4.25
50%       4.000000    5.000000     6.50
75%       6.000000    7.000000     8.50
max      10.000000   10.000000    10.00
```

We can modify the percentiles reported:

```
In [3]: print (df.describe(percentiles=[0.05,0.1,0.9,0.95]))
```

```
       magic_power  aggression  stealth
count     5.000000    5.000000     4.00
mean      5.000000    5.200000     6.25
std       3.316625    3.420526     3.50
min       1.000000    2.000000     2.00
5%        1.600000    2.000000     2.45
10%       2.200000    2.000000     2.90
50%       4.000000    5.000000     6.50
90%       8.400000    8.800000     9.40
95%       9.200000    9.400000     9.70
max      10.000000   10.000000    10.00
```

Specific statistics may be returned:

```
In [4]: print (df.mean())
```

```
magic_power    5.00
aggression     5.20
stealth        6.25
dtype: float64
```

1.2 List of key statistical methods

.mean() = mean
.median() = median
.min() = minimum
.max() =maximum
.quantile(x)
.var() = variance
.std() = standard deviation
.mad() = mean absolute variation
.skew() = skewness of distribution
.kurt() = kurtosis
.cov() = covariance
.corr() = Pearson Correlation coefficent
.autocorr() = autocorelation
.diff() = first discrete difference
.cumsum() = cummulative sum
.comprod() = cumulative product
.cummin() = cumulative minimumcs:
.mean() = mean
.median() = median
.min() = minimum

.max() =maximum
.quantile(x)
.var() = variance
.std() = standard deviation
.mad() = mean absolute variation
.skew() = skewness of distribution
.kurt() = kurtosis
.cov() = covariance
.corr() = Pearson Correlation coefficent
.autocorr() = autocorelation
.diff() = first discrete difference
.cumsum() = cummulative sum
.comprod() = cumulative product
.cummin() = cumulative minimum

1.3 Returning the index of minimum and maximum

idxmin and idxmax will return the index row of the min/max. If two values are equal the first will be returned.

```
In [5]: print ('Minimum:', df['aggression'].min())
        print ('Index row:',df['aggression'].idxmin())
        print ('\nFull row:\n', df.iloc[df['aggression'].idxmin()])
```

```
Minimum: 2
Index row: 2

Full row:
 names           Frodo
type           Hobbit
magic_power         4
aggression          2
stealth             5
Name: 2, dtype: object
```

1.4 Removing rows with incomplete data

We can extract only those rows with a complete data set using the dropna() method.

```
In [6]: print (df.dropna())
```

```
     names    type  magic_power  aggression  stealth
0  Gandolf  Wizard           10           7      8.0
1    Gimli   Dwarf            1          10      2.0
2    Frodo  Hobbit            4           2      5.0
3  Legolas     Elf            6           5     10.0
```

We can use this directly in the describe method.

```
In [7]: print (df.dropna().describe())
```

```
       magic_power  aggression  stealth
count     4.000000    4.000000     4.00
mean      5.250000    6.000000     6.25
std       3.774917    3.366502     3.50
min       1.000000    2.000000     2.00
25%       3.250000    4.250000     4.25
50%       5.000000    6.000000     6.50
75%       7.000000    7.750000     8.50
max      10.000000   10.000000    10.00
```

To create a new dataframe with complete rows only, we would simply assign to a new variable name:

```
In [9]: df_na_dropped = df.dropna()
```

1.5 Counting number of different values in a column

```
In [10]: print (df['type'].value_counts())
```

```
Hobbit    2
Wizard    1
Dwarf     1
Elf       1
Name: type, dtype: int64
```