Search the docs ...

Input/output
General functions
Series
pandas.Series
pandas.Series.index
pandas.Series.array
pandas.Series.values
pandas.Series.dtype
pandas.Series.shape
pandas.Series.nbytes
pandas.Series.ndim
pandas.Series.size
pandas.Series.T
pandas.Series.memory_usage
pandas.Series.hasnans
pandas.Series.empty
pandas.Series.dtypes
pandas.Series.name
pandas.Series.flags
pandas.Series.set_flags
pandas.Series.astype
pandas.Series.convert_dtypes
pandas.Series.infer_objects
pandas.Series.copy
pandas.Series.bool
pandas.Series.to_numpy

pandas.Series.sample

`Series.sample(n=None, frac=None, replace=False, weights=None, random_state=None, axis=None, ignore_index=False)` [source]

Return a random sample of items from an axis of object.

You can use *random_state* for reproducibility.

Parameters: **n : *int, optional***

Number of items from axis to return. Cannot be used with *frac*. Default = 1 if *frac* = None.

frac : *float, optional*

Fraction of axis items to return. Cannot be used with *n*.

replace : *bool, default False*

Allow or disallow sampling of the same row more than once.

weights : *str or ndarray-like, optional*

Default 'None' results in equal probability weighting. If passed a Series, will align with target object on index. Index values in weights not found in sampled object will be ignored and index values in sampled object not in weights will be assigned weights of zero. If called on a DataFrame, will accept the name of a column when axis = 0. Unless weights are a Series, weights must be same length as axis being sampled. If weights do not sum to 1, they will be normalized to sum to 1. Missing values in the weights column will be treated as zero. Infinite values not allowed.

random_state : *int, array-like, BitGenerator, np.random.RandomState, np.random.Generator, optional*

If int, array-like, or BitGenerator, seed for random number generator. If np.random.RandomState or np.random.Generator, use as given.

Changed in version 1.1.0: array-like and BitGenerator object now passed to np.random.RandomState() as seed

Changed in version 1.4.0: np.random.Generator objects now accepted

axis : *{0 or 'index', 1 or 'columns', None}, default None*

Axis to sample. Accepts axis number or name. Default is stat axis for given data type (0 for Series and DataFrames).

ignore_index : *bool, default False*

If True, the resulting index will be labeled 0, 1, …, n - 1.

New in version 1.3.0.

Returns: **Series or DataFrame**

A new object of same type as caller containing *n* items randomly sampled from the caller object.

See also

`DataFrameGroupBy.sample`
: Generates random samples from each group of a DataFrame object.

`SeriesGroupBy.sample`
: Generates random samples from each group of a Series object.

`numpy.random.choice`
: Generates a random sample from a given 1-D numpy array.

Notes

If *frac* > 1, *replacement* should be set to *True*.

Examples

```
>>> df = pd.DataFrame({'num_legs': [2, 4, 8, 0],
...                    'num_wings': [2, 0, 0, 0],
...                    'num_specimen_seen': [10, 2, 1, 8]},
...                   index=['falcon', 'dog', 'spider', 'fish'])
>>> df
        num_legs  num_wings  num_specimen_seen
falcon         2          2                 10
dog            4          0                  2
spider         8          0                  1
fish           0          0                  8
```

Extract 3 random elements from the `Series` `df['num_legs']`: Note that we use *random_state* to ensure the reproducibility of the examples.

```
>>> df['num_legs'].sample(n=3, random_state=1)
fish      0
spider    8
falcon    2
Name: num_legs, dtype: int64
```

A random 50% sample of the `DataFrame` with replacement:

```
>>> df.sample(frac=0.5, replace=True, random_state=1)
      num_legs  num_wings  num_specimen_seen
dog          4          0                  2
fish         0          0                  8
```

An upsample sample of the `DataFrame` with replacement: Note that *replace* parameter has to be *True* for *frac* parameter > 1.

```
>>> df.sample(frac=2, replace=True, random_state=1)
        num_legs  num_wings  num_specimen_seen
dog            4          0                  2
fish           0          0                  8
falcon         2          2                 10
falcon         2          2                 10
fish           0          0                  8
dog            4          0                  2
fish           0          0                  8
dog            4          0                  2
```

Using a DataFrame column as weights. Rows with larger value in the *num_specimen_seen* column are more likely to be sampled.

```
>>> df.sample(n=2, weights='num_specimen_seen', random_state=1)
        num_legs  num_wings  num_specimen_seen
falcon         2          2                 10
fish           0          0                  8
```

Previous
pandas.Series.rtruediv

Next
pandas.Series.searchsorted