Distimate

Dec 21, 2020

Contents

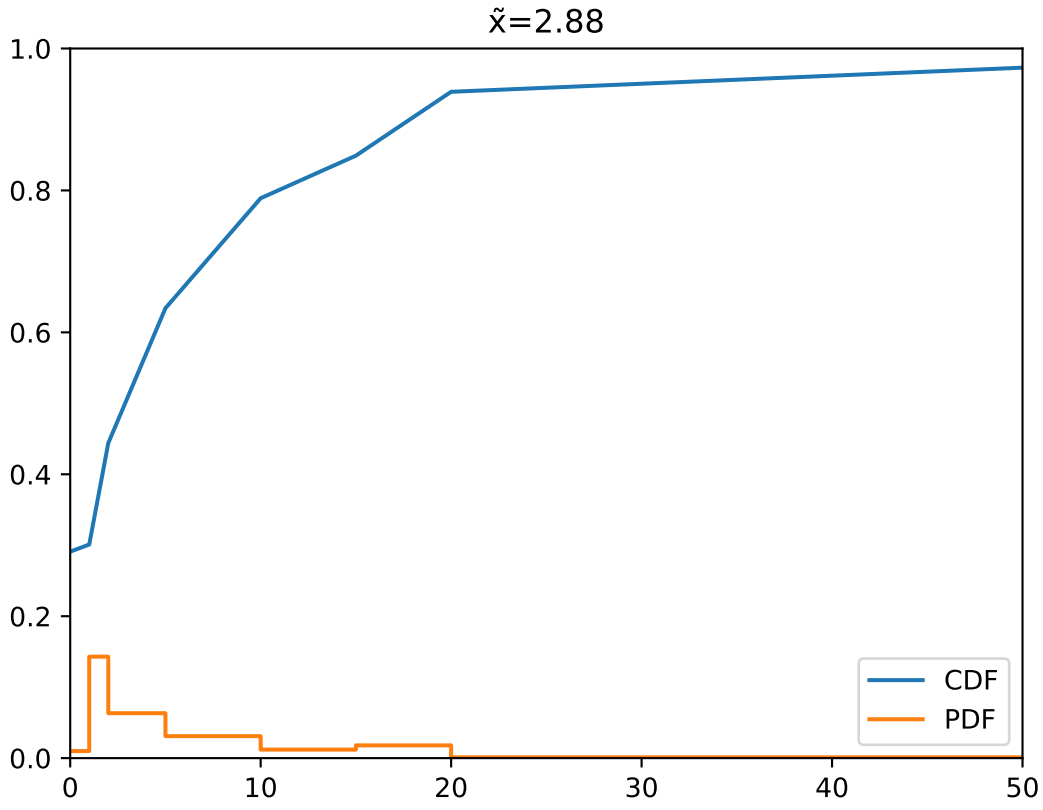
Distimate approximates and plots common statistical functions from histograms.

Distimate can aggregate empirical distributions of random variables. The distributions are represented as histograms with user-defined bucket edges. This is especially useful when working with large datasets that can be aggregated to histograms at database level.

```
import distimate
import matplotlib.pyplot as plt

edges = [0, 1, 2, 5, 10, 15, 20, 50]
values = [291, 10, 143, 190, 155, 60, 90, 34, 27]
dist = distimate.Distribution.from_histogram(edges, values)

plt.title(f"x={dist.quantile(0.5):.2f}")
plt.xlim(0, 50)
plt.ylim(0, 1)
plt.plot(dist.cdf.x, dist.cdf.y, label="CDF")
plt.plot(dist.pdf.x, dist.pdf.y, label="PDF")
plt.legend(loc="lower right")
```



Features:

- Histogram creation and merging
- Probability density function (PDF)
- Cumulative distribution function (CDF or ECDF)

- Quantile (percentile) function
- Pandas integration.

Distimate is hosted at GitHub and it can be installed from PyPI. This documentation is available online at Read the Docs.

CHAPTER 1

Table of Contents

1.1 Installation

Distimate requires Python 3.6 or newer. It can be installed using pip:

```
pip install distimate
```

1.2 Tutorial

1.2.1 Statistical functions

Distimate can approximate common statistical functions from a histogram.

Note: Distimate is most useful in situations when it would be ineffective to retrieve a full dataset.

For example, we can easily aggregate millions of database rows to 100 histogram buckets using SQL. The selected 100 points will provide enough detail for smooth CDF plots.

Distimate supports the following functions:

- *mean()* (ineffective and imprecise, for sanity checks only)
- Probability density function (PDF) - *PDF*
- Cumulative distribution function (CDF) - *CDF*
- Quantile (percentile) function - *Quantile*

Note: In many contexts, distribution functions approximated by Distimate should be called *empirical* distribution functions. They usually aggregate an empirical measure of a random sample.

For example, many libraries implement an empirical cumulative distribution function (ECDF). Distimate calls that function CDF for brevity.

Each of the above functions can be either plotted as an object with `.x` and `.y` attributes, or it can be called to approximate a function value at arbitrary point.

```
import distimate

edges = [0, 10, 50, 100]

cdf = distimate.CDF(edges, [4, 3, 1, 0, 2])
print(cdf.x)
print(cdf.y)
```

```
[  0  10  50 100]
[0.4 0.7 0.8 0.8]
```

The functions accept a number or a NumPy array-like.

```
print(cdf(-7))
print(cdf(0))
print(cdf(5))
print(cdf(107))
print(cdf([-7, 0, 5, 107]))
```

```
0.0
0.4
0.55
nan
[0.   0.4  0.55  nan]
```

Functions are approximated from histograms.

- The first bucket is represented by the first edge.
- We assume that samples are uniformly distributed in inner buckets.
- Outliers in the last bucket cannot be approximated.

```
# The first bucket counts zeros.
mean = distimate.mean(edges, [3, 0, 0, 0, 0])
print(mean)
```

```
0.0
```

```
# The midpoint of the (0, 10] bucket is 5.
mean = distimate.mean(edges, [0, 7, 0, 0, 0])
print(mean)
```

```
5.0
```

```
# The last bucket cannot be approximated.
mean = distimate.mean(edges, [0, 0, 0, 0, 13])
print(mean)
```

```
nan
```

The implementation intelligently handles various corner cases. In the following example, a distribution median can be anything between 10 and 50.

```
quantile = distimate.Quantile(edges, [0, 5, 0, 5, 0])

print(quantile.x, quantile.y)
print(quantile(0.5))
```

```
[0.  0.5 0.5 1. ] [  0.  10.  50. 100.]
30.0
```

A plot will contain a vertical line. When called, the function returns middle of possible values.

1.2.2 Distributions

All approximations from histograms require histogram edges and values. The *Distribution* class is a wrapper that holds both. It provides methods for updating or combining distributions:

```
dist1 = distimate.Distribution(edges)
dist1.add(7)
print(dist1.to_histogram())

dist2 = distimate.Distribution(edges)
dist2.update([0, 1, 1])
print(dist2.to_histogram())

print("----------------")
print((dist1 + dist2).to_histogram())
```

```
[0. 1. 0. 0. 0.]
[1. 2. 0. 0. 0.]
----------------
[1. 3. 0. 0. 0.]
```

- The first histogram bucket counts items less than or equal to the left-most edge.
- The inner buckets count items between two edges. Intervals are left-open, the inner buckets count items greater than their left edge and less than or equal to their right edge.
- The last bucket counts items greater than the right-most edge.

Note: The bucketing implemented in Distimate works best with non-negative metrics.

- The left-most edge should be zero in most cases.
- The right-most edge should be set to highest expected value.

With this setup, the first bucket counts zeros and the last bucket counts outliers.

If Distimate is used with negative metrics, it can return wrong approximation for values bellow the left-most edge.

Optional weights are supported:

```
dist = distimate.Distribution(edges)
dist.update([0, 7, 13], [1, 2, 3])
print(dist.to_histogram())
```

```
[1. 2. 3. 0. 0.]
```

It is common to define histogram edges once and reuse them between distributions. The *DistributionType* class can remember the histogram edges. It can be used as a factory for creating distributions:

```
dist_type = distimate.DistributionType([0, 10, 50, 100])
print(dist_type.edges)

dist = dist_type.from_samples([0, 7, 10, 107])
print(dist.edges, dist.values)
```

```
[  0  10  50 100]
[  0  10  50 100] [1. 2. 0. 0. 1.]
```

1.2.3 Pandas integration

Consider that you load `pandas.DataFrame` with histogram values:

```
import pandas as pd

columns = ["color", "size", "hist0", "hist1", "hist2", "hist3", "hist4"]
data = [
    (  "red", "M", 0, 1, 0, 0, 0),
    ( "blue", "L", 1, 2, 0, 0, 0),
    ( "blue", "M", 3, 2, 1, 0, 1),
]
df = pd.DataFrame(data, columns=columns)
print(df)
```

```
  color size  hist0  hist1  hist2  hist3  hist4
0   red    M      0      1      0      0      0
1  blue    L      1      2      0      0      0
2  blue    M      3      2      1      0      1
```

The histogram data can be converted to `pandas.Series` with *Distribution* instances:

```
hist_columns = df.columns[2:]
dists = pd.Series.dist.from_histogram(edges, df[hist_columns])
print(dists)
```

```
0    <Distribution: weight=1, mean=5.00>
1    <Distribution: weight=3, mean=3.33>
2     <Distribution: weight=7, mean=nan>
dtype: object
```

We can replace histograms in the original DataFrame by the distributions:

```
df["qty"] = dists
df.drop(columns=hist_columns, inplace=True)
print(df)
```

```
  color size                                    qty
0   red    M  <Distribution: weight=1, mean=5.00>
1  blue    L  <Distribution: weight=3, mean=3.33>
2  blue    M   <Distribution: weight=7, mean=nan>
```

The advantage of the new column is that we can use it with the `dist` accessor to compute statistical functions for all DataFrame rows using a simple expression.

```
median = df["qty"].dist.quantile(0.5)
print(median)
```

```
0    5.0
1    2.5
2    2.5
Name: qty_q50, dtype: float64
```

See *DistributionAccessor* for all methods available via the `dist` accessor.

Series of `Distribution` instances can be aggregated:

```
agg = df.groupby("color")["qty"].sum()
print(agg)
```

```
color
blue    <Distribution: weight=10, mean=nan>
red     <Distribution: weight=1, mean=5.00>
Name: qty, dtype: object
```

1.3 API

1.3.1 Statistical functions

distimate.stats.**mean**()

Estimate mean from a histogram.

The approximated mean is for sanity checks only, it is ineffective and imprecise to estimate mean from a histogram.

Return NaN for distributions with no samples.

- Inner bins are represented by their midpoint (assume that samples are evenly distributed in bins).
- The left outer bin is represented by the leftmost edge (assume that there are no samples bellow the supported range).
- Return NaN if the rightmost bin is not empty (because we cannot approximate outliers).

Parameters

- **edges** – 1-D array-like, ordered histogram edges
- **hist** – 1-D array-like, one item longer than edges

Returns float number

class distimate.stats.**PDF**

Probability density function (PDF).

Callable object with `.x` and `.y` attributes. The attributes can be used for plotting, or the function can be called to estimate a PDF value at arbitrary point. The callable accepts a single value or an array-like.

The returned callable takes inputs from a distribution domain and returns outputs between 0 and 1 (inclusive).

The PDF values provides relative likelihoods of various distribution values. It is computed from a histogram by normalizing relative frequencies by bucket widths.

- For inputs less than the first edges, the PDF will always return zero.
- For inputs equal to the first edge (typically zero), the PDF function will return zero or NaN, depending on whether the first histogram bucket is empty. This is because the PDF is not defined for discrete distributions.
- For inputs in each of inner histogram buckets (which are left-open), one value is returned. On a plot, this will form a staircase. To plot a non-continuous distribution, x-values are duplicated.
- For inputs greater than the last edge, the PDF returns either zero or NaN, depending on whether the last histogram bucket is empty.

Parameters

- **edges** – 1-D array-like, ordered histogram edges
- **hist** – 1-D array-like, one item longer than edges‘

__call__(*v*)

Compute function value at the given point.

Parameters **v** – scalar value or Numpy array-like

Returns scalar value or Numpy array depending on *x*

x

Return `numpy.array` of x-values for plotting

y

Return `numpy.array` of y-values for plotting

class distimate.stats.**CDF**

Cumulative distribution function (CDF).

Callable object with `.x` and `.y` attributes. The attributes can be used for plotting, or the function can be called to estimate a CDF value at arbitrary point. The callable accepts a single value or an array-like.

The returned callable takes inputs from a distribution domain and returns outputs between 0 and 1 (inclusive).

`cdf(x)` returns a probability that a distribution value will be less than or equal to `.x`.

- For inputs less than the first edge, the CDF will always return zero.
- Function return exact values for inputs equal to histogram edges. Values inside histogram buckets are interpolated.
- CDF of the first edge can be used to obtain how many samples were equal to that edge (typically zero)
- For inputs greater than the last edge, the PDF returns either one or NaN, depending on whether the last histogram bucket is empty.

Parameters

- **edges** – 1-D array-like, ordered histogram edges

- **hist** – 1-D array-like, one item longer than edges

__call__(*v*)
Compute function value at the given point.

Parameters **v** – scalar value or Numpy array-like

Returns scalar value or Numpy array depending on *x*

x
Return `numpy.array` of x-values for plotting

y
Return `numpy.array` of y-values for plotting

class distimate.stats.**Quantile**
Create a quantile function.

Returns a callable object with `.x` and `.y` attributes. The attributes can be used for plotting, or the function can be called to estimate a quantile value at arbitrary point. The function accepts a single value or an array-like.

The returned callable takes inputs from a range between 0 and 1 (inclusive) and returns outputs from a distribution domain.

`quantile(q)` returns the smallest `.x` for which `cdf(x) >= q`.

- If the first histogram bucket is not empty, the quantile value can return the first edge for many inputs.
- If an inner histogram bucket is empty, then the quantile value can be ambiguous. In that case, duplicate x-values will be plotted. When called, the quantile function will a middle of possible values.
- The function returns NaN for values outside of the <0, 1> range.
- When called with zero, returns the left edge of the smallest non-empty bucket. If the first bucket is not empty, returns the first edge.
- When called with one, returns the right edge of the greatest non-empty bucket. If the last bucket is not empty, returns NaN.

__call__(*v*)
Compute function value at the given point.

Parameters **v** – scalar value or Numpy array-like

Returns scalar value or Numpy array depending on *x*

x
Return `numpy.array` of x-values for plotting

y
Return `numpy.array` of y-values for plotting

1.3.2 Distributions

class distimate.distributions.**Distribution**(*edges*, *values=None*)
Statistical distribution represented by its histogram.

Provides an object interface on top of a histogram array. Supports distribution merging and comparison. Implements approximation of common statistical functions.

Parameters

- **edges** – 1-D array-like, ordered histogram edges

- **values** – 1-D array-like, histogram, one item longer than *edges*

__eq__(*other*)
Return whether distribution histograms are equal.

__add__(*other*)
Combine this distribution with other distribution.

__iadd__(*other*)
Combine this distribution with other distribution inplace.

edges
Edges of the underlying histogram

Returns

class 1-D *numpy.array*, ordered histogram edges

values
Values of the underlying histogram.

Returns 1-D *numpy.array*, histogram values

classmethod from_samples(*edges*, *samples*, *weights=None*)
Create a distribution from a list of values.

Parameters

- **edges** – 1-D array-like, ordered histogram edges
- **samples** – 1-D array-like
- **weights** – optional scalar or 1-D array-like with same length as samples.

Returns a new *Distribution*

classmethod from_histogram(*edges*, *histogram*)
Create a distribution from a histogram.

Parameters

- **edges** – 1-D array-like, ordered histogram edges
- **histogram** – 1-D array-like, one item longer than edges

Returns a new *Distribution*

classmethod from_cumulative(*edges*, *cumulative*)
Create a distribution from a cumulative histogram.

Parameters

- **edges** – 1-D array-like, ordered histogram edges
- **cumulative** – 1-D array-like, one item longer than edges

Returns a new *Distribution*

to_histogram()
Return a histogram of this distribution as a NumPy array.

Returns 1-D `numpy.array`

to_cumulative()
Return a cumulative histogram of this distribution as a NumPy array.

Returns 1-D `numpy.array`

add(*value*, *weight=None*)
Add a new item to this distribution.

Parameters

- **value** – item to add
- **weight** – optional item weight

update(*values*, *weights=None*)
Add multiple items to this distribution.

Parameters

- **values** – items to add, 1-D array-like
- **weights** – optional scalar or 1-D array-like with same length as samples.

weight
Return a total weight of samples in this distribution.

Returns float number

mean
Estimate mean of this distribution.

The approximated mean is for sanity checks only, it is ineffective and imprecise to estimate mean from a histogram.

See *mean()* for details.

Returns float number

pdf
Probability density function (PDF) of this distribution.

See *PDF* for details.

Returns a *PDF* instance

cdf
Cumulative distribution function (CDF) of this distribution.

See *CDF* for details.

Returns a *CDF* instance

quantile
Quantile function of this distribution.

See *Quantile* for details.

Returns a *Quantile* instance

class distimate.types.**DistributionType**(*edges*)
Factory for creating distributions with constant histogram edges.

Parameters **edges** – 1-D array-like, ordered histogram edges

edges
Edges of the underlying histogram

Returns

class 1-D *numpy.array*, ordered histogram edges

empty()
Create an empty distribution.

Returns a new `Distribution`

from_samples(*samples*, *weights=None*)
Create a distribution from a list of values.

Parameters

- **samples** – 1-D array-like
- **weights** – optional 1-D array-like

Returns a new `Distribution`

from_histogram(*histogram*)
Create a distribution from a histogram.

Parameters **histogram** – 1-D array-like

Returns a new `Distribution`

from_cumulative(*cumulative*)
Create a distribution from a cumulative histogram.

Parameters **cumulative** – 1-D array-like

Returns a new `Distribution`

1.3.3 Pandas integration

class distimate.pandasext.**DistributionAccessor**(*series*)
Implements `.dist` accessor on `pandas.Series`.

Allows to easily call *Distribution* methods on all instances in Pandas Series:

```
df[col] = pd.Series.dist.from_histogram(dist_type, histograms)
median = df[col].dist.quantile(0.5)
```

static from_histogram(*dist_type*, *histograms*, *, *name=None*)
Construct a new `pandas.Series` from histograms.

This is a static method that can be accessed as `pd.Series.dist.from_histogram()`.

Parameters
- **dist_type** – *DistributionType* or 1-D array-like with histogram edges
- **histograms** – `pandas.DataFrame` or 2-D array-like
- **name** – optional name of the series.

Returns `pandas.Series`

static from_cumulative(*dist_type*, *cumulatives*, *, *name=None*)
Construct a new `pandas.Series` from cumulative histograms.

This is a static method that can be accessed as `pd.Series.dist.from_cumulative()`.

Parameters
- **dist_type** – *DistributionType* or 1-D array-like with histogram edges
- **histograms** – `pandas.DataFrame` or 2-D array-like
- **name** – Optional name of the series.

Returns `pandas.Series`

to_histogram()
Convert `pandas.Series` of *Distribution* instances to histograms.

Returns `pandas.DataFrame` with histogram values.

to_cumulative()
Convert `pandas.Series` of *Distribution* instances to cumulative histograms.
Returns `pandas.DataFrame` with cumulative values

pdf(*v*)
Compute PDF for `pandas.Series` of *Distribution* instances.
Parameters **v** – input value, or list of them
Returns `pandas.Series`

cdf(*v*)
Compute CDF for series of distribution instances.
Parameters **v** – input value, or list of them
Returns `pandas.Series`

quantile(*v*)
Compute quantile function `pandas.Series` of *Distribution* intances.
Parameters **v** – input value, or list of them
Returns `pandas.Series`

values
Values of the underlying histograms.
Returns 2-D `numpy.array`

1.4 Development

1.4.1 Installation

Distimate can be installed with development dependencies from a Git clone using pip:

```
pip install -e .[dev]
```

1.4.2 Tools

- Code is checked with flake8.
- Tests are run using pytest.
- Documentation is built using Sphinx.

Tox runs all of the above tools automatically:

```
tox
```

1.5 FAQ

What is Distimate useful for? Distimate started as a library for plotting empirical cumulative distribution functions (ECDF).

It can plot CDF not only from an array of samples (similar to statsmodels ECDF), but also from a histogram. The advantage of histograms is that they can be computed by a database, aggregating millions of rows to a few dozens of buckets.

Having CDFs, we can ask for median or other quantiles. Histograms can be easily merged, allowing to estimate quantiles of grouped data.

Distimate helps to write cleaner code. The *Distribution* class wraps two arrays into a simple object with a nice interface. The Pandas *DistributionAccessor* helps to remove unnecessary boilerplate.

How are approximations lossy? CDF values at histogram edges are exact. If you choose the edges wisely at round values, the user may never ask for CDF at other points.

When plotted using Matplotlib, you will hardly notice a difference between a plot with 100 and 1000 points. It is important to use similar scales for both histogram edges and plot axis.

In case of quantile estimates, the error will never be larger than a width of a bucket. If samples are evenly distributed in the bucket, the error will be much smaller.

You can look at a paper explaining a similar approach.

Why does Distimate use more buckets than NumPy? If you define 10 edges, `numpy.histogram()` will create a histogram with 9 buckets. Distimate will need a histogram with 11 buckets, because it also counts items bellow the left-most edge and above the right-most edge.

Counting the out-of-range items is important for plotting CDFs or computing quantiles. It is necessary to know a total count to get relative a density right.

Distimate assumes that edges are defined only once and then reused. With the constant edge, risk the out-of-range items cannot be eliminated.

Why does Distimate use left-open intervals (unlike NumPy)? If you define your buckets as `[0, 10, 100]`, `numpy.histogram()` will insert the number 10 to the `[10, 100)` bucket. Distimate puts 10 to the `(0, 10]` bucket.

The left-open interval correspond to definition of CDF, where `cdf(x)` includes all samples less than or equal to `x`.

This is especially important for `cdf(0)` that should include all samples equal to zero.

Why does Distimate approximate left and right out-of-range values differently?

The first bucket contains items less than or equal to the left-most edge. For approximations, we assume that the items in this bucket are equal to the left-most edge.

The last bucket contains items greater that the right-most edge. We have no approximation for items in this bucket.

For metrics that do not have negative values, the left-most edge is typically set to zero. The first bucket then counts samples that are exactly zero. Because the zero is a prominent value, Distimate is designed to return exact estimates for it.

For example, if most samples are zero, median should return zero. This would not be possible if the first bucket contained negative values.

Sample in the last bucket can be arbitrary high, so no approximation is possible.

1.6 License

(continues on next page)

(continued from previous page)

```
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

1.7 Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index

Symbols

A

C

D

E

F

M

P

Q

T

U

V

W

X

Y