


Calculate herfindahl Index in python

An alternative for the herfindahl function in R language. The Herfindahl-Hirschman index (HHI) is a commonly accepted measure of market concentration. It is calculated by squaring the market share of each firm competing in a market, and then summing the resulting numbers, and can range from close to zero to 10,000.



Calculate ECDF using Python

An alternative for the ecdf function in R language. It is really hard to find an alternative for ecdf() function of R in python. There are few online codes available, but I verified this as the most accurate match to the R's ecdf() function. It follows the algorithm behind calculating the ecdf of a given data.



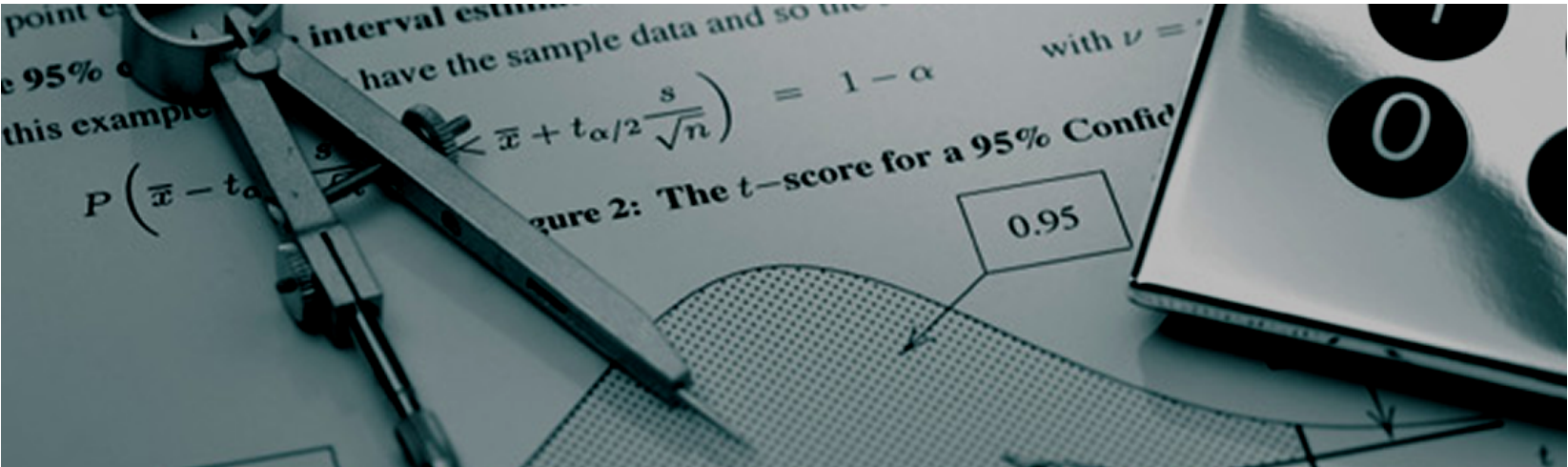
Kripanshu Bhargava

Follow

Software developer intern at Harvard | Graduate Student at UTD

Published on May 14 2018

Calculate ECDF using Python



ECDF: Emperical Cumulative Distribution Function:

An empirical distribution function is the distribution function associated with the empirical measure of a sample. This cumulative distribution function is a step function that jumps up by 1/n at each of the n data points. Its value at any specified value of the measured variable is the fraction of observations of the measured variable that are less than or equal to the specified value. Let (x1, ..., xn) be independent, identically distributed real random variables with the common cumulative distribution function F(t). Then the empirical distribution function is defined as:

$$\hat{F}_n(t) = \frac{\text{number of elements in the sample } \leq t}{n} = \frac{1}{n} \sum_{i=1}^n 1_{x_i \leq t},$$

Coming to my point, it is really hard to find an alternative for `ecdf()` function of R in Python. There are few online codes available, but I verified this as the most accurate match to the R's `ecdf()` function. It follows the algorithm behind calculating the ECDF of a given data.

If you want to calculate it on your own, then it is not a big function (in terms of LOC).

Let's say we are given a 1-D array that we name as data.

```
data = [101, 118, 121, 103, 142, 111, 119, 122, 128, 112, 117,157]
```

We would first convert it into a numpy array.

```
raw_data = np.array(data)
```

Now we would find the respective x and y values that represent the actual cdf of the data.

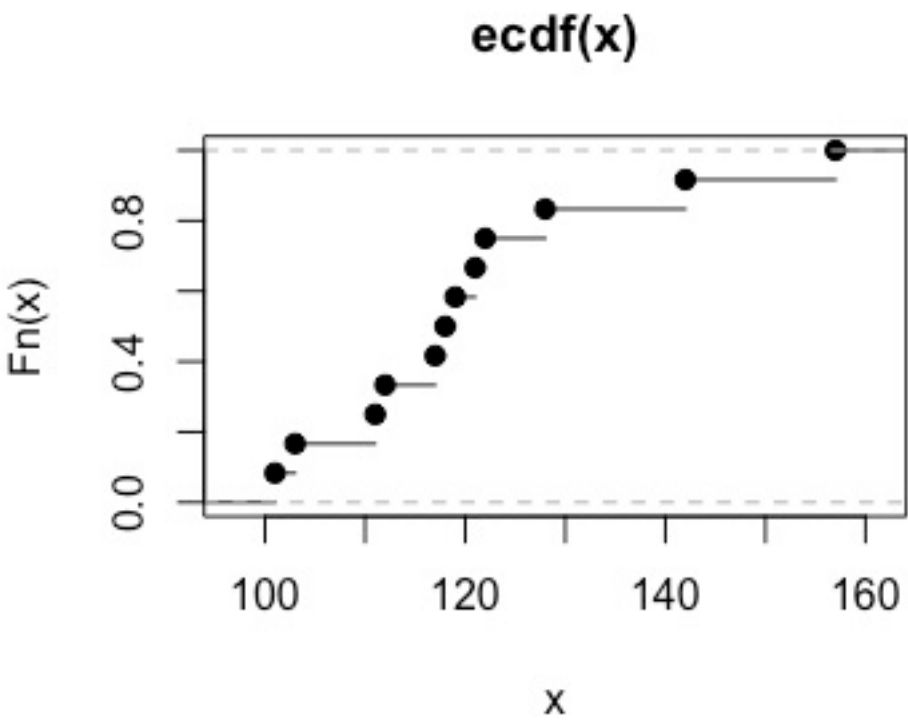
```
# create a sorted series of unique data
cdfx = np.sort(data.unique())
# x-data for the ECDF: evenly spaced sequence of the uniques
x_values = np.linspace(start=min(cdfx),
                        stop=max(cdfx),num=len(cdfx))

# size of the x_values
size_data = raw_data.size
# y-data for the ECDF:
y_values = []
for i in x_value:
    # all the values in raw data less than the ith value in x_values
    temp = raw_data[raw_data <= i]
    # fraction of that value with respect to the size of the x_values
    value = temp.size / size_data
    # pushing the value in the y_values
    y_values.append(value)
# return both x and y values
return x_values,y_values
```

These results are verified with the following function in R:

```
x <- c(101, 118, 121, 103, 142, 111, 119, 122, 128, 112, 117,157)
ecdf(x)
```

Using the output you get in the above R code, you can find the respective x and y values.



Python

Statistics

Pandas

R



leave a comment !

