Fitting empirical distribution to theoretical ones with Scipy (Python)?

Asked 9 years, 11 months ago Active today Viewed 151k times

156

174

INTRODUCTION: I have a list of more than 30,000 integer values ranging from 0 to 47, inclusive, e.g. `[0,0,0,0,..,1,1,1,1,...,2,2,2,2,...,47,47,47,...]` sampled from some continuous distribution. The values in the list are not necessarily in order, but order doesn't matter for this problem.

PROBLEM: Based on my distribution I would like to calculate p-value (the probability of seeing greater values) for any given value. For example, as you can see p-value for 0 would be approaching 1 and p-value for higher numbers would be tending to 0.

I don't know if I am right, but to determine probabilities I think I need to fit my data to a theoretical distribution that is the most suitable to describe my data. I assume that some kind of goodness of fit test is needed to determine the best model.

Is there a way to implement such an analysis in Python (`Scipy` or `Numpy`)? Could you present any examples?

Thank you!

python numpy statistics scipy distribution

Share Improve this question Follow

edited Aug 21 '19 at 4:47

Amit Kumar Gupta
13.8k 5 38 55

asked Jul 8 '11 at 6:00

s_sherly
1,937 4 17 14

2 You have only discrete empirical values but want a continuous distribution? Do I understand that correctly? – Michael J. Barber Jul 8 '11 at 10:25

1 It seems nonsensical. What do the numbers represent? Measurements with limited precision? – Michael J. Barber Jul 8 '11 at 10:44

1 Michael, I explained what the numbers represent in my previous question: stackoverflow.com/questions/6615489/… – s_sherly Jul 8 '11 at 11:01

6 That's count data. It's not a continuous distribution. – Michael J. Barber Jul 8 '11 at 11:08

1 Check the accepted answer to this question stackoverflow.com/questions/48455018/… – Ahmad Senousi Apr 13 '19 at 5:48

10 Answers

Active | Oldest | Votes

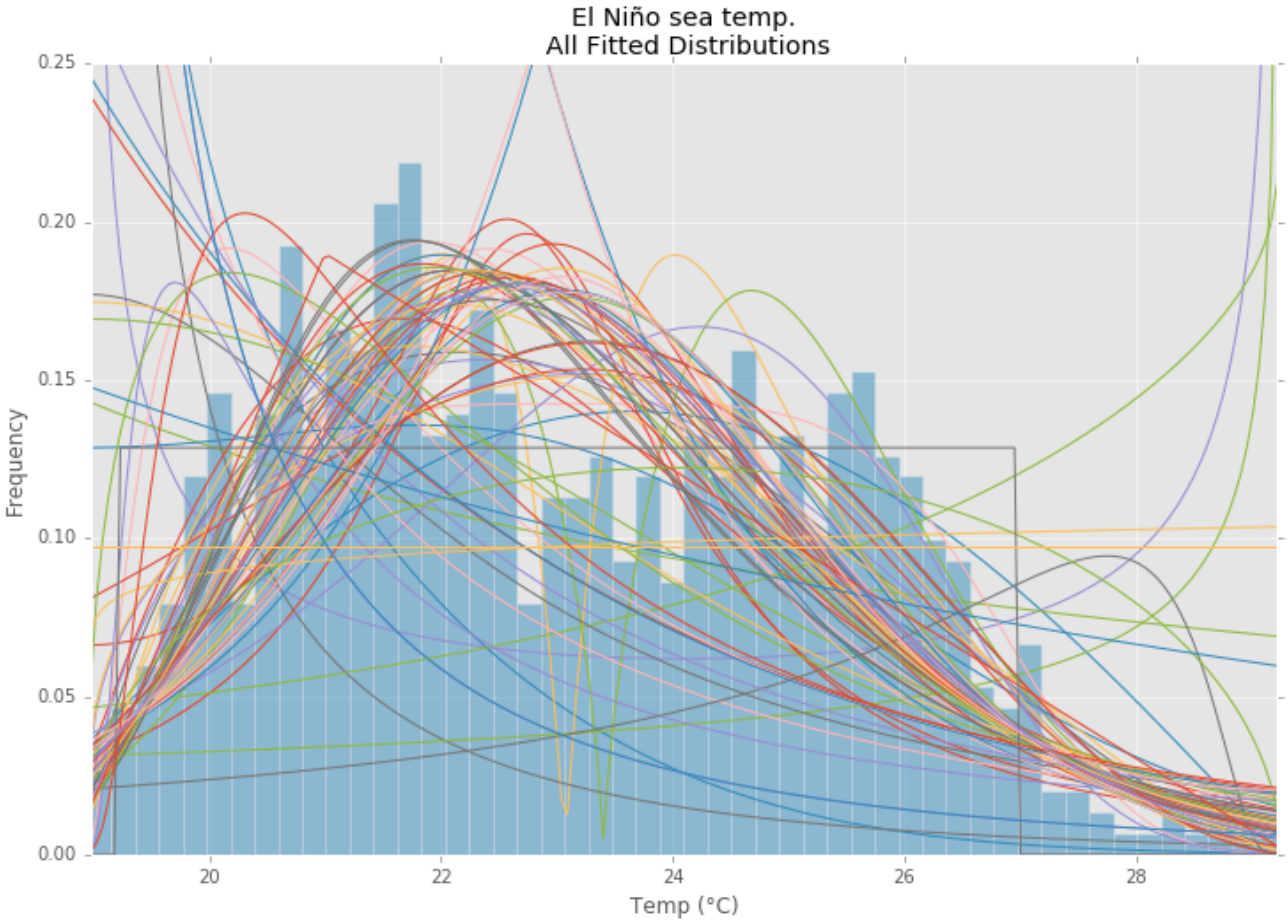
Distribution Fitting with Sum of Square Error (SSE)

This is an update and modification to Saullo's answer, that uses the full list of the current `scipy.stats` distributions and returns the distribution with the least SSE between the distribution's histogram and the data's histogram.

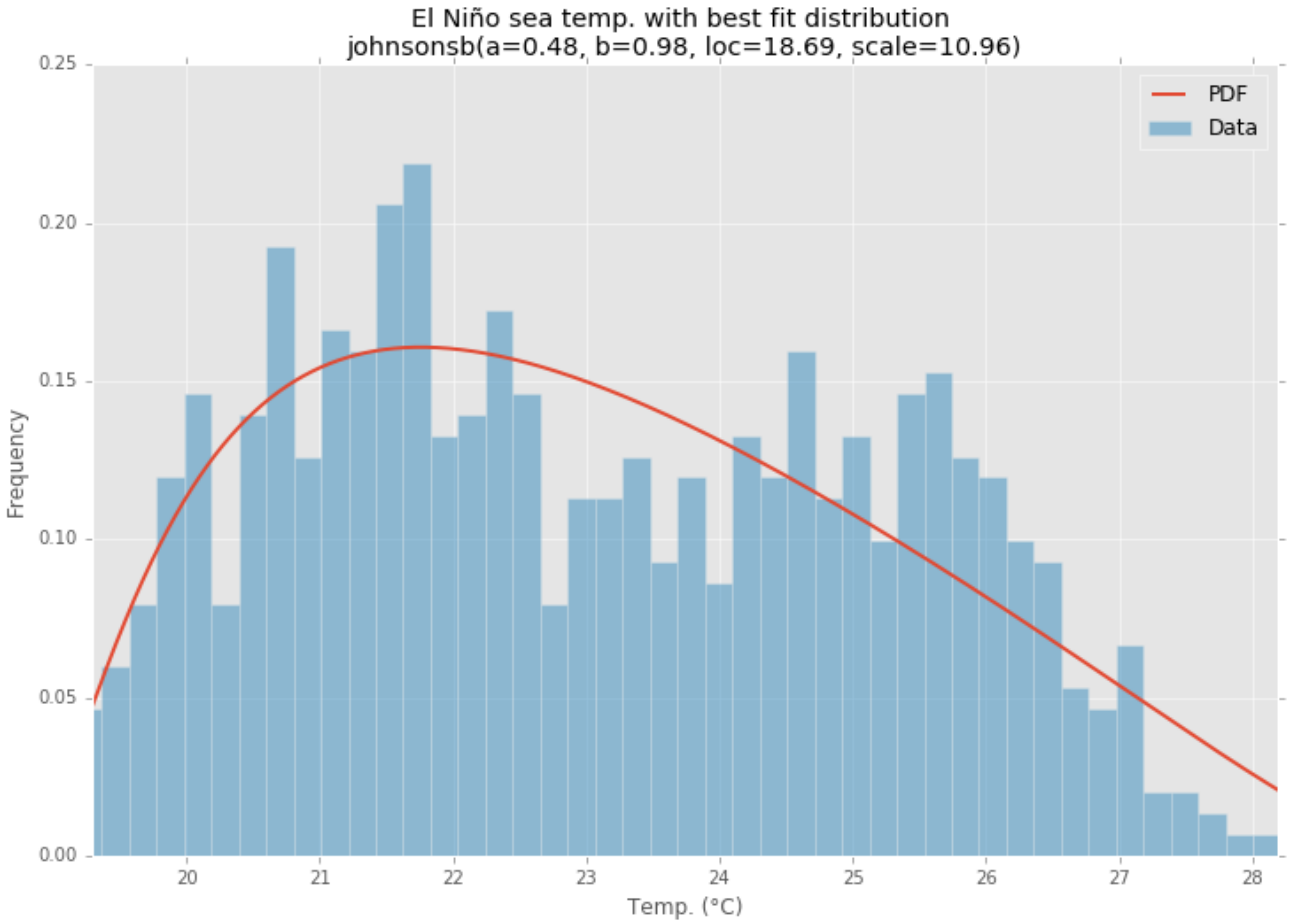
Example Fitting

Using the El Niño dataset from `statsmodels` , the distributions are fit and error is determined. The distribution with the least error is returned.

All Distributions



Best Fit Distribution



Example Code

```
%matplotlib inline

import warnings
import numpy as np
import pandas as pd
import scipy.stats as st
import statsmodels as sm
import matplotlib
import matplotlib.pyplot as plt

matplotlib.rcParams['figure.figsize'] = (16.0, 12.0)
matplotlib.style.use('ggplot')

# Create models from data
def best_fit_distribution(data, bins=200, ax=None):
    """Model data by finding best fit distribution to data"""
    # Get histogram of original data
    y, x = np.histogram(data, bins=bins, density=True)
    x = (x + np.roll(x, -1))[:-1] / 2.0

    # Distributions to check
    DISTRIBUTIONS = [

st.alpha,st.anglit,st.arcsine,st.beta,st.betaprime,st.bradford,st.burr,st.cauchy,s


st.dgamma,st.dweibull,st.erlang,st.expon,st.exponnorm,st.exponweib,st.exponpow,st.
```

```
st.foldcauchy,st.foldnorm,st.frechet_r,st.frechet_l,st.genlogistic,st.genpareto,st

st.genextreme,st.gausshyper,st.gamma,st.gengamma,st.genhalflogistic,st.gilbrat,st.

st.gumbel_l,st.halfcauchy,st.halflogistic,st.halfnorm,st.halfgennorm,st.hypsecant,

st.invweibull,st.johnsonsb,st.johnsonsu,st.ksone,st.kstwobign,st.laplace,st.levy,s

st.logistic,st.loggamma,st.loglaplace,st.lognorm,st.lomax,st.maxwell,st.mielke,st.

st.nct,st.norm,st.pareto,st.pearson3,st.powerlaw,st.powerlognorm,st.powernorm,st.r

st.rayleigh,st.rice,st.recipinvgauss,st.semicircular,st.t,st.triang,st.truncexpon,

st.uniform,st.vonmises,st.vonmises_line,st.wald,st.weibull_min,st.weibull_max,st.w

    ]

    # Best holders
    best_distribution = st.norm
    best_params = (0.0, 1.0)
    best_sse = np.inf

    # Estimate distribution parameters from data
    for distribution in DISTRIBUTIONS:

        # Try to fit the distribution
        try:
            # Ignore warnings from data that can't be fit
            with warnings.catch_warnings():
                warnings.filterwarnings('ignore')

                # fit dist to data
                params = distribution.fit(data)

                # Separate parts of parameters
                arg = params[:-2]
                loc = params[-2]
                scale = params[-1]

                # Calculate fitted PDF and error with fit in distribution
                pdf = distribution.pdf(x, loc=loc, scale=scale, *arg)
                sse = np.sum(np.power(y - pdf, 2.0))

                # if axis pass in add to plot
                try:
                    if ax:
                        pd.Series(pdf, x).plot(ax=ax)
                    end
                except Exception:
                    pass

                # identify if this distribution is better
                if best_sse > sse > 0:
                    best_distribution = distribution
                    best_params = params
                    best_sse = sse
```

```
            except Exception:
                pass

    return (best_distribution.name, best_params)

def make_pdf(dist, params, size=10000):
    """Generate distributions's Probability Distribution Function """

    # Separate parts of parameters
    arg = params[:-2]
    loc = params[-2]
    scale = params[-1]

    # Get sane start and end points of distribution
    start = dist.ppf(0.01, *arg, loc=loc, scale=scale) if arg else
dist.ppf(0.01, loc=loc, scale=scale)
    end = dist.ppf(0.99, *arg, loc=loc, scale=scale) if arg else dist.ppf(0.99,
loc=loc, scale=scale)

    # Build PDF and turn into pandas Series
    x = np.linspace(start, end, size)
    y = dist.pdf(x, loc=loc, scale=scale, *arg)
    pdf = pd.Series(y, x)

    return pdf

# Load data from statsmodels datasets
data =
pd.Series(sm.datasets.elnino.load_pandas().data.set_index('YEAR').values.ravel())


# Plot for comparison
plt.figure(figsize=(12,8))
ax = data.plot(kind='hist', bins=50, normed=True, alpha=0.5,
color=plt.rcParams['axes.color_cycle'][1])
# Save plot limits
dataYLim = ax.get_ylim()

# Find best fit distribution
best_fit_name, best_fit_params = best_fit_distribution(data, 200, ax)
best_dist = getattr(st, best_fit_name)

# Update plots
ax.set_ylim(dataYLim)
ax.set_title(u'El Niño sea temp.\n All Fitted Distributions')
ax.set_xlabel(u'Temp (°C)')
ax.set_ylabel('Frequency')

# Make PDF with best params
pdf = make_pdf(best_dist, best_fit_params)

# Display
plt.figure(figsize=(12,8))
ax = pdf.plot(lw=2, label='PDF', legend=True)
data.plot(kind='hist', bins=50, normed=True, alpha=0.5, label='Data',
legend=True, ax=ax)

param_names = (best_dist.shapes + ', loc, scale').split(', ') if
best_dist.shapes else ['loc', 'scale']
param_str = ', '.join(['{}={:0.2f}'.format(k,v) for k,v in zip(param_names,
best_fit_params)])
dist_str = '{}({})'.format(best_fit_name, param_str)

ax.set_title(u'El Niño sea temp. with best fit distribution \n' + dist_str)
```

```
ax.set_xlabel(u'Temp. (°C)')
ax.set_ylabel('Frequency')
```

Share Improve this answer Follow

edited Sep 17 '18 at 20:55
Melquíades Ochoa
197 2 13

answered Jun 3 '16 at 14:26

tmthydvnprt
8,852 7 49 66

4 Awesome. Consider using `density=True` instead of `normed=True` in `np.histogram()` . ^^ – Peque Apr 28 '17 at 16:13

What version of pandas is required for the 'density' keyword? I get 'AttributeError: Unknown property density' from matplotlib with matplotlib 2.0.2 and pandas 0.20.2 when executing this code snippet because of 'data.plot(..., density=True, ...)'. – aleneum Jun 28 '17 at 16:15

I do not remember what version I used to originally answer the question or what version contained @Peque's recommendation. You can check for version issues/compatibility in pandas/matplotlib docs. There should be docs for each version. If you want, you can use `normed=True` instead. – tmthydvnprt Jun 28 '17 at 18:13

@tmthydvnprt I think you changed `normed` to `density` in `np.histogram()` (which was my recommendation as the former is deprecated), but you also changed it in the `.plot()` methods (I think Matplotlib's `pyplot.hist()` uses `normed` and never supported `density`). ;-) – Peque Jun 29 '17 at 7:27

1 @tmthydvnprt Maybe you could undo the changes in the `.plot()` methods to avoid future confusion. ^^ – Peque Jun 30 '17 at 8:45

12 To get distribution names: `from scipy.stats._continuous_distns import _distn_names` . You can then use something like `getattr(scipy.stats, distname)` for each `distname` in _distn_names`. Useful because the distributions are updated with different SciPy versions. – Brad Solomon Aug 29 '17 at 19:49

That's a good method, but it's not working correctly. Please, consider this CSV file as an input for the Series: drive.google.com/open?id=1kOa35v0NHjWHK8VmMDY-sKiGZyeJzw6z – Alexandre V. Feb 16 '18 at 12:28

1 Can you please explain why this code checks for best fit of continuous distributions only, and cannot check for discrete or multivariate distributions. Thank you. – Adam Schroeder Mar 16 '18 at 20:56

Maybe? Can you explain a specific case? – tmthydvnprt Mar 17 '18 at 0:40

1 Hi @tmthydvnprt Yes. Suppose I have a dataset that has a binomial distribution and another one that is a poisson. When I add scipy.stats.binom or scipy.stats.poisson to your example code, they are not generated on the plot. The example code (which I'm still trying to comprehend) doesn't check for discrete distributions (docs.scipy.org/doc/scipy/reference/stats.html). Is there a way for me to check for best fit of discrete distributions? – Adam Schroeder Mar 20 '18 at 20:38

14 Very cool. I had to update the color parameter - `ax = data.plot(kind='hist', bins=50, normed=True, alpha=0.5, color=list(matplotlib.rcParams['axes.prop_cycle'])[1]['color'])` – basswaves May 10 '18 at 19:43

@tmthydvnprt Sir, why do you do this trick with the `roll()` a the start? – Ladenkov Vladislav Jul 25 '18 at 21:20

2 I do not understand why do you put this line: x = (x + np.roll(x, -1))[:-1] / 2.0. Can you explain me the objective of this operation, please? – jartymcfly Oct 30 '18 at 10:55

1 Hey! I don't know if it's too late, but I'm struggling to generate random numbers with the best fit found by this code. I know I should use the `rvs()` method, but I'm not sure what input is needed for the `c` argument. Any ideas? – Juan C Mar 18 '19 at 15:25

@JuanC you could do `from scipy.stats import beta` `beta.rvs(a=1.74, b=2.69, loc=18.91, scale=10.62)`. Optionally, you can include the `size` parameter for how many samples you want to generate. Docs here – Riebeckite Apr 3 '19 at 14:14

@tmthydvnprt Hello, would you please re write the code to the newest version of python libs? or at least mention the best version of matplotlib to use this code? there are error with the new library. regards. – Abdulaziz Al Jumaia Sep 5 '19 at 11:51

2 @jartymcfly not sure if you worked out why `np.roll()` was used? It's just a moving average with a window size of `2.0`. `x` in this case, represents the `bin_edges`, so performing this calculation returns the centre of each bin. – Josmoor98 Nov 16 '19 at 18:02

Thank you for your answer! I'm not sure I understand how to calculate the probability of a number from the pdf. For example, what is the probability to get two? I think this code creates CDF from this PDF: `pdf.cumsum() / pdf.sum()` and then, I can calculate `cdf(2+epsilon)-cdf(2)`. What do you think? – user5746421 Dec 28 '19 at 14:04

Yeah, is there a specific reason why discrete distributions are not checked? I know we can model a discrete data with a continuous distribution, but it is not optimal. – NoName Feb 25 '20 at 23:25

I had to add at the end plt.show() to see the results. – Gouz Oct 2 '20 at 0:22

11 just in case someone in 2020 is wondering how to make this run, change `import statsmodel as sm` to `import statsmodel.api as sm` – Rafael Silva Oct 9 '20 at 14:49

Thanks a lot. Is this suitable for continuous variables if the Distributions list replaced by continuous distributions by obtaining the names of the continuous distributions as @BradSolomon suggested in his comment? – Paulo Feb 15 at 13:04

A supplement to Rafael's comment: the package name is `statsmodels`, not `statsmodel` – dyluns Apr 12 at 1:38

165

There are more than 90 implemented distribution functions in SciPy v1.6.0. You can test how some of them fit to your data using their `fit()` method. Check the code below for more details: