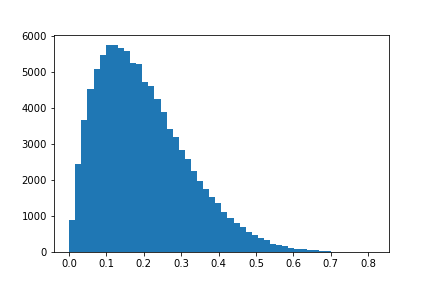
pythonのmatplotlibライブラリを使った簡単なヒストグラム生成方法を紹介します。matplotlibのヒストグラム生成メソッド”plt.hist()”の代表的な機能（オプション、引数）を扱っています。

目次

1. [**この記事でできること**](https://python-ai-learn.com/2021/02/06/histogram/#toc1)
2. [**疑似乱数の生成**](https://python-ai-learn.com/2021/02/06/histogram/#toc2)
3. [**ヒストグラムの生成と画像の保存**](https://python-ai-learn.com/2021/02/06/histogram/#toc3)
4. [**横軸(bin)の変更**](https://python-ai-learn.com/2021/02/06/histogram/#toc4)
   1. [bin数の指定](https://python-ai-learn.com/2021/02/06/histogram/#toc5)
   2. [値の範囲の指定](https://python-ai-learn.com/2021/02/06/histogram/#toc6)
   3. [bin数と幅の指定](https://python-ai-learn.com/2021/02/06/histogram/#toc7)
5. [**縦軸の変更**](https://python-ai-learn.com/2021/02/06/histogram/#toc8)
6. [**縦軸をlogスケールに変更**](https://python-ai-learn.com/2021/02/06/histogram/#toc9)
7. [**縦軸を確率密度に変更**](https://python-ai-learn.com/2021/02/06/histogram/#toc10)
8. [**縦軸をデータ数の割合に変更**](https://python-ai-learn.com/2021/02/06/histogram/#toc11)
9. [**おわりに**](https://python-ai-learn.com/2021/02/06/histogram/#toc12)

**この記事でできること**

pythonのmatplotlibライブラリを使って、pythonのリスト型に格納されたデータからヒストグラムを作成し、画像ファイルとして保存します。

ヒストグラムの例

ヒストグラムとは、横軸を値、縦軸を個数としたデータの分布を可視化した棒グラフです。縦軸を度数とよび、棒グラフの各棒が示す値の範囲にどのくらいの数のデータがあるかを棒の高さで表示します。

ヒストグラムを作成するデータが数値の場合、棒グラフの各棒のことをビン(bin)と呼び、各棒が示す値の範囲をbin幅と呼んだりします。

次のURLを参考にしました。  
<https://matplotlib.org/3.3.3/api/_as_gen/matplotlib.pyplot.hist.html>

**疑似乱数の生成**

まずはヒストグラム生成の説明のため、ヒストグラムの生成の対象とするリスト(list、配列)を作成します。

今回はベータ分布に従う疑似乱数生成器random.betavariate()を使って10万個の要素からなるリスト”data”を作成しました。ベータ分布を使ったのに特に意味がありません。一次元のリストであればどんなデータでもよいです。

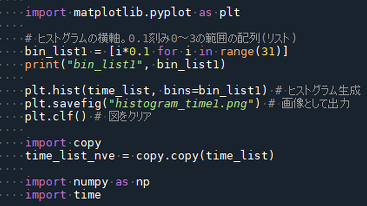
**import** random

random.seed(314) # 乱数シードの設定

# 疑似乱数でリストを作成。ここではbeta分布を使用

data = [random.betavariate(2, 8) **for** i **in** range(100000)]

この記事とは直接関係ないですが、乱数のシードの設定については次の記事で紹介しました。

[[](https://python-ai-learn.com/2021/02/05/seed/)](https://python-ai-learn.com/2021/02/05/seed/" \o "pythonで乱数シード(seed)を設定する３種の方法" \t "_blank)

**[pythonで乱数シード(seed)を設定する３種の方法](https://python-ai-learn.com/2021/02/05/seed/" \o "pythonで乱数シード(seed)を設定する３種の方法" \t "_blank)**

[pythonで乱数を生成するとき、pythonのrandomや、numpyのnp.random、scipyのscipy.statsを使用することがあると思います。乱数生成の再現性で重要となる乱数シード(seed, 種)の設定方法を紹介します。](https://python-ai-learn.com/2021/02/05/seed/" \o "pythonで乱数シード(seed)を設定する３種の方法" \t "_blank)

[[https://www.google.com/s2/favicons?domain=https://python-ai-learn.com/2021/02/06/seed/](https://python-ai-learn.com/2021/02/05/seed/)](https://python-ai-learn.com/2021/02/05/seed/" \o "pythonで乱数シード(seed)を設定する３種の方法" \t "_blank)

[python-ai-learn.com](https://python-ai-learn.com/2021/02/05/seed/" \o "pythonで乱数シード(seed)を設定する３種の方法" \t "_blank)

**ヒストグラムの生成と画像の保存**

matplotlibライブラリの”plt.hist()”を使えば、簡単に次のようなソースコードでリスト”data”のヒストグラムを生成できます。

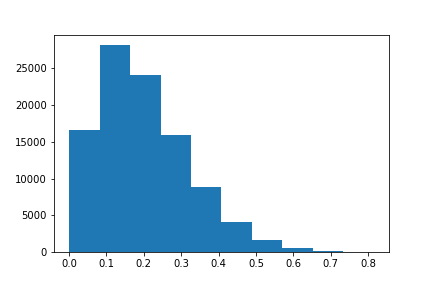
**import** matplotlib.pyplot **as** plt

plt.hist(data) # default

plt.savefig("histogram.png") # 画像ファイルとして保存

plt.clf() # 図をクリア

matplotlib.pyplotのメソッド”plt.hist(data)”でヒストグラムを生成し、”plt.savefig(“histogram.png”)”で画像ファイル”histogram.png”として保存するというサンプルコードになっています。画像ファイル”histogram.png”は次のような画像となります。

ヒストグラム（default設定）

**横軸(bin)の変更**

関数”plt.hist()”で生成するヒストグラムはカスタマイズできます。まずは横軸を変更する方法を説明します。

“plt.hist()”に何もパラメータ(オプション、引数)を指定しないと、入力されたリストの最小値、最大値を値の範囲として、それを等間隔に10個に分割したbinとなります。棒グラフが示す値の幅は等間隔で、10個の棒グラフができるということです。

**bin数の指定**

binの数は”plt.hist()”に引数”bins”を指定することで変更が可能です。次のように”bins=50″と指定すれば、等間隔に50個に分割されたbin、つまり50個の棒からなるグラフが作成されます。defaultは”bins=10″です。

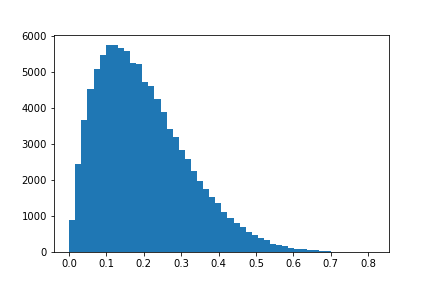
サンプルコードや生成するヒストグラムの画像は次の通りです。

**import** matplotlib.pyplot **as** plt

plt.hist(data, bins=50) # bin数50

plt.savefig("histogram\_bins50.png")

plt.clf()

bin数を指定したヒストグラム

**値の範囲の指定**

次に、横軸の値の範囲を指定するには、”plt.hist()”の引数”range”を使います。指定したい最小値minと最大値maxをタプル(min, max)で引数”range”に指定します。

指定した範囲をはみ出した値の個数は、小さい値は最も左のbin、大きい値は最も右のbinに加算されることになります。

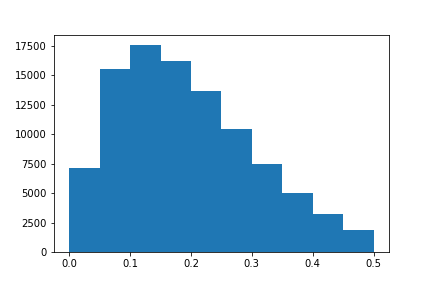
サンプルコードや生成するヒストグラムの画像は次の通りです。

**import** matplotlib.pyplot **as** plt

plt.hist(data, range=(0.2, 0.4)) # ヒストグラムを表示する値の範囲

plt.savefig("histogram\_range.png")

plt.clf()

値の範囲を指定したヒストグラム

**bin数と幅の指定**

“plt.hist()”の引数”bins”に整数ではなくリストを指定することで、binの数や幅をよりカスタマイズして直接指定することができます。

次のサンプルコードでは、0～0.2の値については細かく0.01刻みの狭いbin幅とし、0.2～0.9は0.1刻みの粗いbin幅としたヒストグラムを生成しています。

**import** matplotlib.pyplot **as** plt

# 0～0.2までを0.01刻み、0.2～0.9を0.1刻みのリストを作成

bin\_list = [i\*0.01 **for** i **in** range(20)] + [0.2 + i\*0.1 **for** i **in** range(8)]

# bin\_list = [0.0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09,

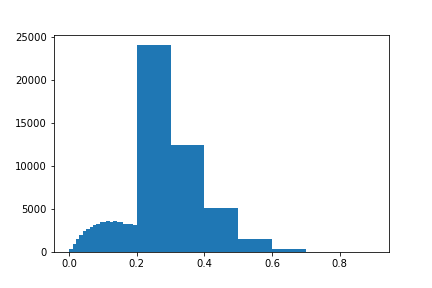
# 0.1, 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19,

# 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]

plt.hist(data, bins=bin\_list) # bin数と幅を等幅でなくlistで指定

plt.savefig("histogram\_binslist.png")

plt.clf()

binの数や幅を直接指定したヒストグラム

横軸の0～0.2の値については、bin幅が狭くなる分だけ、各binに対応するdataの要素も減りますので、棒グラフの高さが低くなりました。少し歪んだヒストグラムになってしまいました。

**縦軸の変更**

ここまでに紹介したヒストグラムの例の縦軸は、各binに対応するデータの個数となっていました。データの個数やデータの偏りは、ヒストグラムを生成するデータによって変わりますので、縦軸を変更した方が見やすいヒストグラムになる場合もあります。

以降は、縦軸を変更する方法を説明します。

**縦軸をlogスケールに変更**

“plt.hist()”の引数”log”をTrueに指定することで、縦軸をlogスケールに変更できます。ただし、制約があり引数”bins”はdefaultでは駄目で、何らかの値を指定しないといけません。binsのdefaultである10を指定したい場合でも、”bins=10”を引数として指定することが必要です。

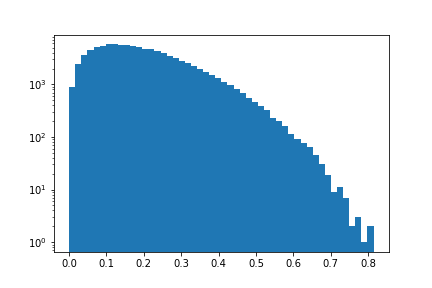
サンプルコードや生成するヒストグラムの画像は次の通りです。

**import** matplotlib.pyplot **as** plt

plt.hist(data, bins=50, log=**True**) # 縦軸をlogスケールに

plt.savefig("histogram\_log.png")

plt.clf()

縦軸をlogスケールにしたヒストグラム

**縦軸を確率密度に変更**

“plt.hist()”の引数”density”をTrueに指定することで、縦軸を各binに対応するデータの個数ではなく確率密度に変更できます。少し前のバージョンのmatplotlibでは、”normed”という名前の引数だったようです。

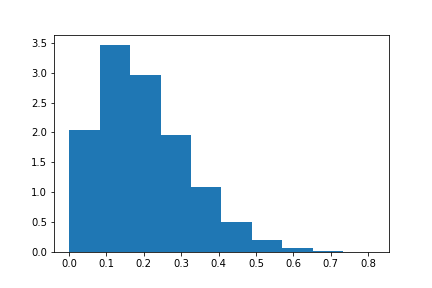
確率密度は、ヒストグラムの全体の棒グラフの面積が1となるように、縦軸を調整した値です。各棒グラフの高さは、(binに対応するデータの個数)/(全データ数\*bin幅)となります。

**import** matplotlib.pyplot **as** plt

plt.hist(data, density=**True**) # 縦軸を確率密度

plt.savefig("histogram\_density.png")

plt.clf()

density=Trueとしたヒストグラム

このヒストグラムではbin幅はdefaultの等間隔です。棒グラフの高さはbin幅に依存するため、上述したようにbinに直接リストを指定して等間隔ではないようにした場合、引数”density=True”でのヒストグラムの形状は、”density=False”(default)とは異なるものになります。

次のヒストグラムの形状と、上に掲載した”binの数や幅を直接指定したヒストグラム”という画像と比較してみてください。上で「歪んだヒストグラム」と記載した画像です。

**import** matplotlib.pyplot **as** plt

# 0～0.2までを0.01刻み、0.2～0.9を0.1刻みのリストを作成

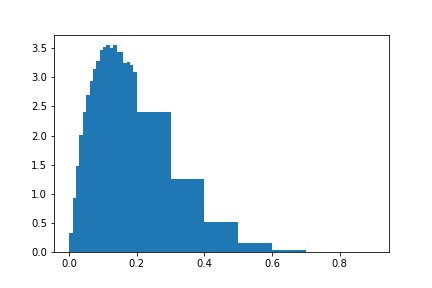
bin\_list = [i\*0.01 **for** i **in** range(20)] + [0.2 + i\*0.1 **for** i **in** range(8)]

# bin幅を等幅でなくlistで指定、かつ縦軸を確率密度

plt.hist(data, density=**True**, bins=bin\_list)

plt.savefig("histogram\_density\_binslist.png")

plt.clf()

density=Trueとしたヒストグラム（bin幅が非等間隔）

上の方に掲載した”binの数や幅を直接指定したヒストグラム”と、この画像は、同じデータを同じbin幅でヒストグラムにしたものです。違いは引数”density=True”としたことです。”density=True”とすることで棒グラフの高さがbin幅に応じて調整されますので、”density=False”(またはdefault)で歪んだヒストグラムであっても、見た目が改善されることがあります。

**縦軸をデータ数の割合に変更**

上記で説明したように、”plt.hist()”の引数を”density=True”とした場合の縦軸は確率密度で、各棒グラフの高さはbin幅に依存してしまいます。

縦軸をbin幅には依存させずに、全データ数に対する割合にしたい場合もあるかと思います。言い換えると、全棒グラフの縦軸の値の合計をbin幅に依らずに1としたい場合です。

bin幅には依存させずに、全棒グラフの高さの合計が1となるように調整するには、一度numpyのヒストグラム関数”np.histogram”等を使って度数(各binのデータ数)を計算し、度数を全データ数で割れば実現可能です。

次のソースコードは、各棒グラフの高さを、(binに対応するデータの個数)/(全データ数)となるようにしたヒストグラムの生成例です。

**import** matplotlib.pyplot **as** plt

**import** numpy **as** np

# 0～0.2までを0.01刻み、0.2～0.9を0.1刻みのリストを作成

bin\_list = [i\*0.01 **for** i **in** range(20)] + [0.2 + i\*0.1 **for** i **in** range(8)]

# numpyのnp.histogramでヒストグラムの度数(縦軸)であるcountsを計算

counts, bins = np.histogram(data, bins=bin\_list)

# 度数countsを全データ数で割り、全データ数の割合を計算

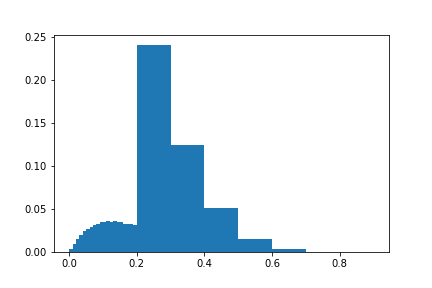
counts\_normed = counts / sum(counts)

# weightsに縦軸の値を指定

plt.hist(bins[:-1], bins=bins, weights=counts\_normed)

plt.savefig("histogram\_normed\_binslist.png")

plt.clf()

縦軸を全データ数に対する度数の割合としたヒストグラム（bin幅が非等間隔）

このヒストグラムの形状は、上の方に掲載した”binの数や幅を直接指定したヒストグラム”と同じです。今回は縦軸をbin幅に依らない値としたので、bin幅が狭い横軸の0～0.2は棒グラフの高さが低くなりました。これらはbin幅が狭いため、各binに対応するdataの割合も小さいからです。

1つ上の画像”density=Trueとしたヒストグラム（bin幅が非等間隔）”と比較すると、違いがわかると思います。

**おわりに**

この記事では、pythonのmatplotlibライブラリを使った簡単なヒストグラム生成方法を紹介しました。matplotlibのヒストグラム生成メソッド”plt.hist()”の代表的な引数の使い方を扱いました。

記事中でも少し登場しましたが、numpyにもヒストグラム生成関数”np.histogram”があります。”np.histogram”と”plt.hist()”の引数や機能は似ていて、”np.histogram”を使用する際にも本記事が参考になるのではと思っています。