Сравнение различных библиотек для визуализации данных: Matlotlib, Seaborn и Plotly.

Введение.

Название проекта, как всегда в случае таких учебных работ, на мой взгляд, не совпадает с его целью. Заявленная тема слишком сложна для выполнения даже квалифицированными специалистами.

Поясню на примере:

Наверное, в специализированных вузах на первом курсе задаётся, например, такая тема: «Преимущества и недостатки САД/САМ/САЕ платформ

Solid Works и CATIA.» Мой сын, например, который дока в Solid Works, такую тему за месяц не осилит, наверное, просто потому, что ему понадобится погрузиться в CATIA и исполнить в ней часть своих проектов, не знаю, сколько он за это запросит!

Целью проекта, считаю поэтому, является показать, что студент (смешно в моём случае) что-то умеет в программировании на Python, используя заявленные библиотеки, а, наверное, главное, знает, где найти нужное. Как говорили в моём вузе, цель высшего образования не научить чему-то, а научить, где искать. С тех пор искать стало гораздо проще, с одной стороны, с другой - сложнее, поскольку нужно из лавины информации выбирать действительно полезное. Кстати говоря, поэтому основная информация, используемая в проекте, была почерпнута из официальной документации к библиотекам ( [matplotlib.org](https://matplotlib.org/), [seaborn.pydata.org](https://seaborn.pydata.org), [plotly.com](https://plotly.com/python/)).

Тем не менее, помимо показа таких навыков, я постарался акцентировать внимание именно на различия в лёгкости реализации, наглядности, информативности и в других аспектах использования этих трёх основных библиотек визуализации данных.

Примеры использования различных библиотек для визуализации анализа данных.

Для применения различных методов визуализации анализа данных я взял dataset https://www.kaggle.com/datasets/shreyasg23/life-expectancy-averaged-dataset. Этот набор данных содержит агрегированные данные об ожидаемой продолжительности жизни, усредненные за несколько лет по различным странам, а также связанные с ними социально-экономические факторы и факторы здравоохранения.

Попробуем, например, вывести точечный график зависимости продолжительности жизни от ВВП в разных странах.

Для начала нужно установить необходимые библиотеки. Например, с помощью pip install matplotlib seaborn plotly. После этого импортируем их в проект:

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import seaborn as sns

import plotly.express as px

Pandas необходима для работы с базами данных. В нашем случае, файл в формате csv мы читаем как таблицу, состоящую из строк и столбцов.

Итак, в matplotlib код может быть выглядеть таким образом:

df = pd.read\_csv("Life-Expectancy-Data-Averaged (1).csv")

#scatter matplotlib

plt.figure(figsize=(15, 15))

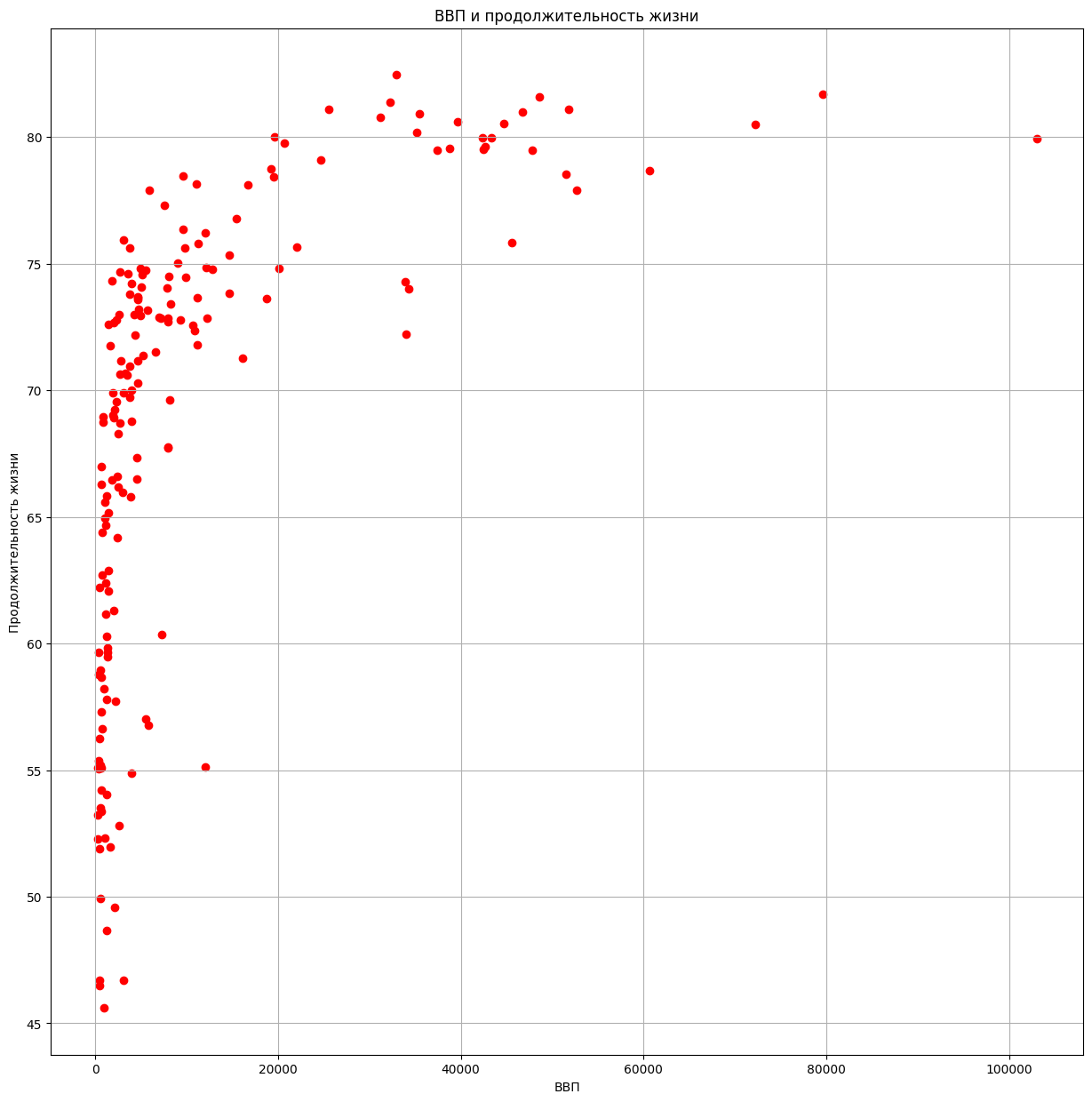
plt.scatter(df['GDP\_per\_capita'], df['Life\_expectancy'], color='r')

plt.xlabel('ВВП')

plt.ylabel('Продолжительность жизни')

plt.title('ВВП и продолжительность жизни')

plt.grid()



В seaborn код и картинка будут выглядеть так:

#scatter seaborn

sns.scatterplot(x=df['GDP\_per\_capita'], y=df['Life\_expectancy'], color='g')

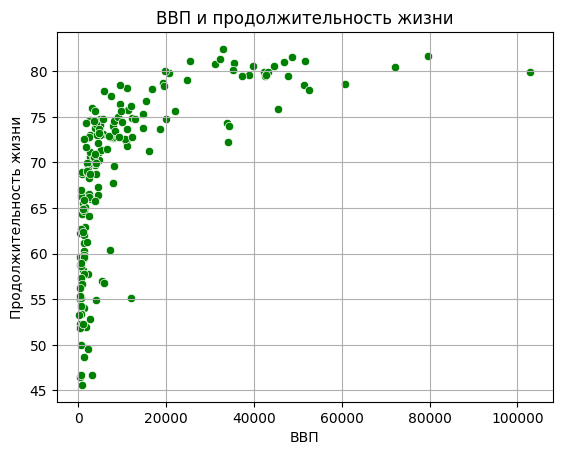
plt.xlabel('ВВП')

plt.ylabel('Продолжительность жизни')

plt.title('ВВП и продолжительность жизни')

plt.grid()

plt.show()



Теперь тоже самое в plotly:

#scatter plotly

fig=px.scatter(x=df['GDP\_per\_capita'], y=df['Life\_expectancy'], hover\_name=df['Country'],

title='ВВП и продолжительность жизни', labels= {'x': 'ВВП', 'y' : 'Продолжительность жизни'})



Что мы видим на этом простом примере?

Код что в matplotlib, что в seaborn. Изображение в seaborn немного приятнее. А вот в plotly и точки не сливаются, и появляется возможность посмотреть на каждую, что она собой представляет. И всё это в одну строку.

Переходим к столбчатым диаграммам. Например, мы хотим вывести показатели стран, где продолжительность жизни больше средней.

Они определяются как life\_exp\_more50 = df[df['Life\_expectancy'] > des['mean']]

Как это выглядит в matplotlib:

#Bar chart matplotlib

plt.figure(figsize=(15, 6))

plt.bar(life\_exp\_more50['Country'], life\_exp\_more50['Life\_expectancy'], color = '#f39c12')

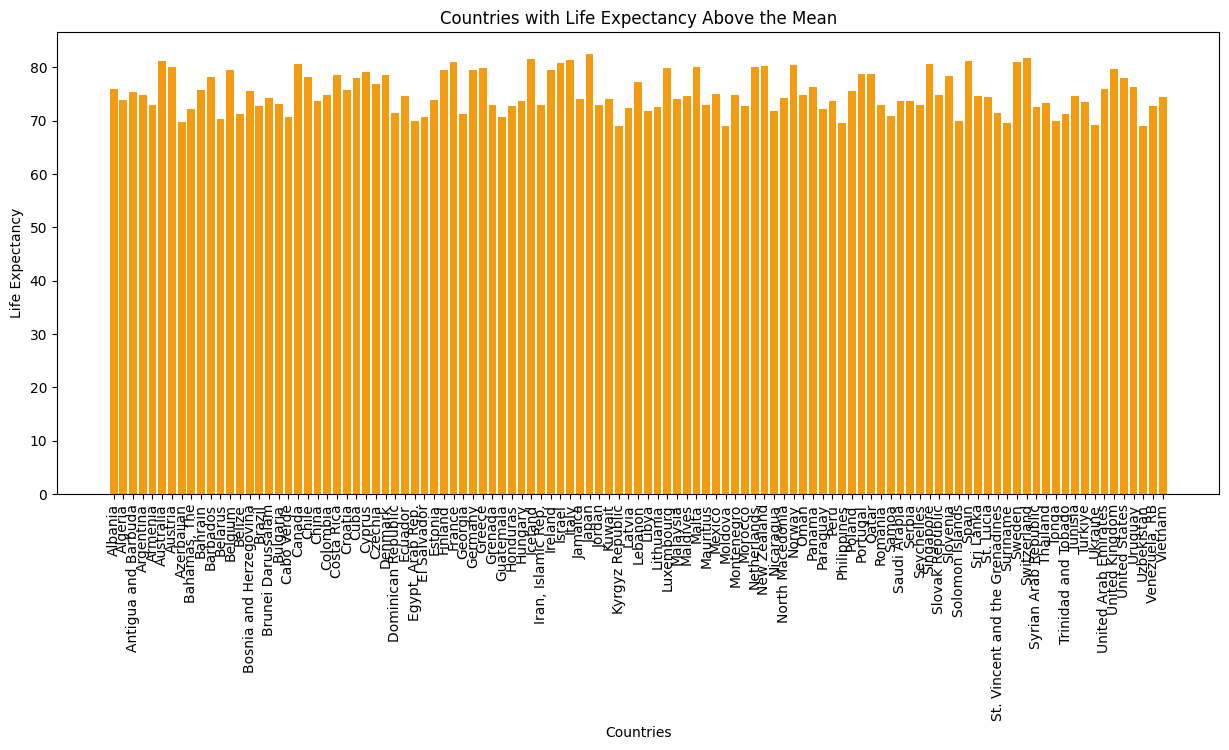
plt.xlabel("Countries")

plt.ylabel("Life Expectancy")

plt.title("Countries with Life Expectancy Above the Mean")

plt.xticks(rotation=90)

plt.show()



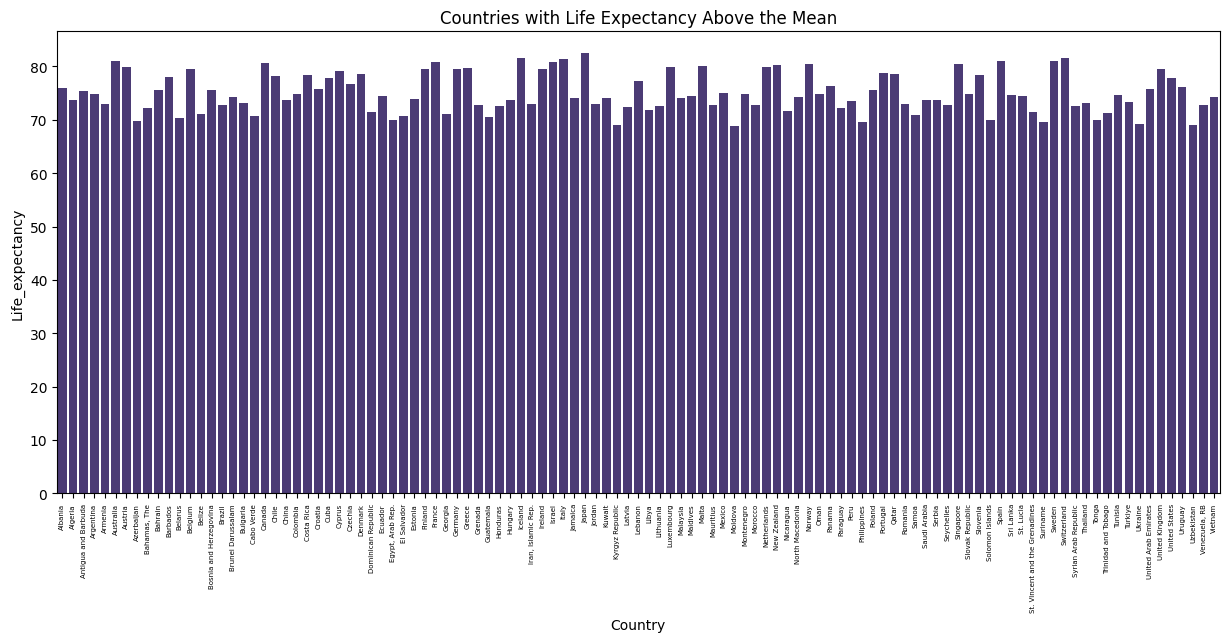
В seaborn:

fig, ax = plt.subplots(figsize=(12, 6))

sns.barplot(x=life\_exp\_more50['Country'],y= life\_exp\_more50['Life\_expectancy']).set(title='Countries with Life Expectancy Above the Mean')

ax.tick\_params(axis='x', labelrotation=90, labelsize=5)

plt.show()



В plotly:

#bar chart plotly

import plotly.express as px

fig = px.bar( x=life\_exp\_more50['Country'], y=life\_exp\_more50['Life\_expectancy'],title='Countries with Life Expectancy Above the Mean',labels={'y':'Life\_expectancy','x':'Country'},height=600)

fig.show()



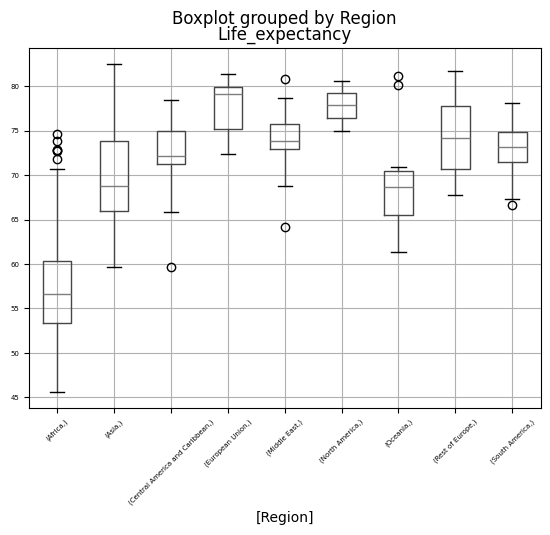
Видим, что в seaborn код даже сложнее для почти такой же картинки, как и в matplotlib. Знающие люди не зря говорят, что не стоит seaborn’ом колоть орехи, он для простого кода для сложных вещей. А в plotly опять всё хорошо – одна строка и всё красиво и информативно!

Попробуем теперь ящики с усиками – boxplot.

Снова matplotlib:

df.boxplot(['Life\_expectancy'], by=['Region'], rot=45, fontsize=5)

plt.show()



Seaborn:

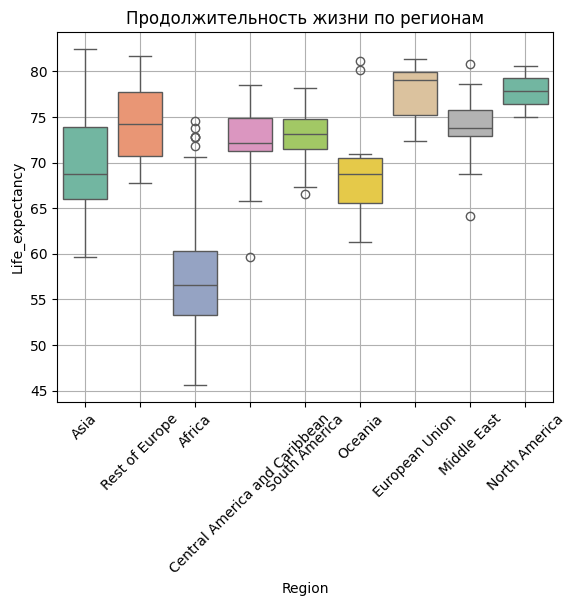
fig, ax = plt.subplots()

sns.boxplot(data=df, x=df['Region'], y=df['Life\_expectancy'], hue='Region', palette='Set2')

plt.grid(True)

plt.xticks(rotation=45)

plt.title('Продолжительность жизни по регионам')

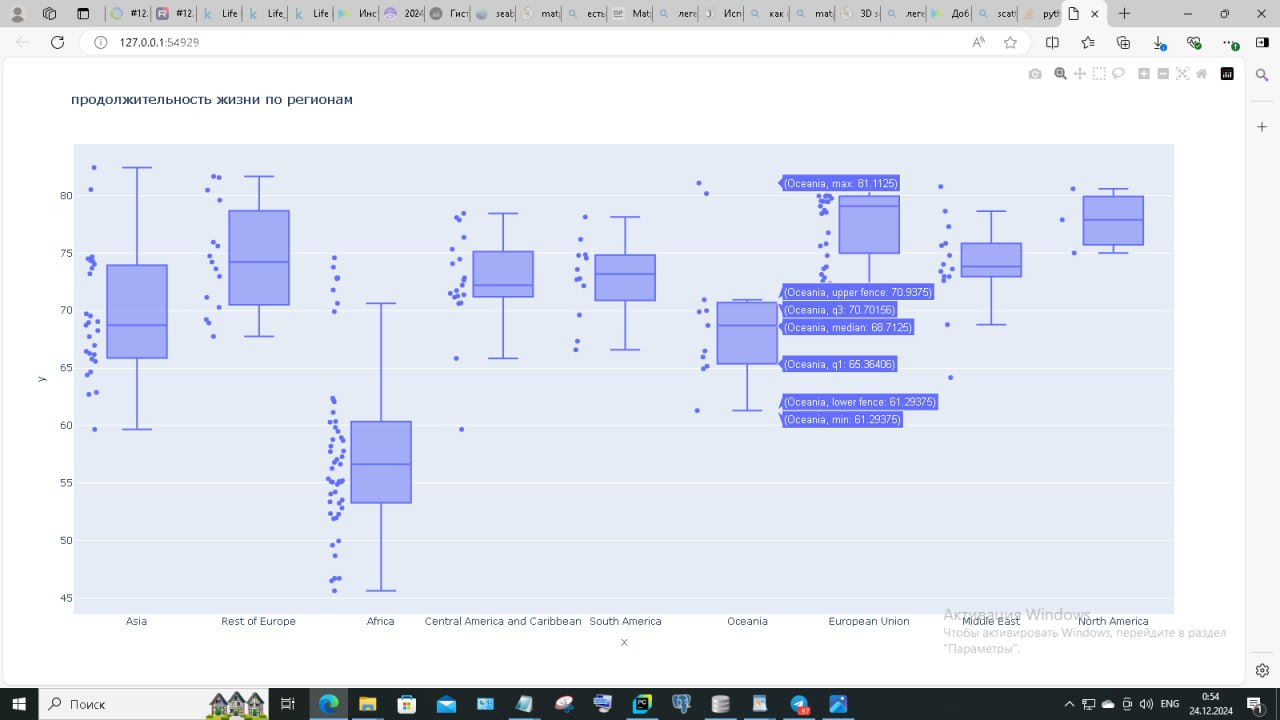


Plotly:

#plotly box

fig\_=px.box(x=df['Region'], y=df['Life\_expectancy'],points='all', title='продолжительность жизни по регионам'height=600)

fig\_.show()



Здесь снова, при почти одинаковых усилиях в matplotlib и seaborn картинки получаются почти одинаковые (изменять цвета, подписи, тексты – отдельная работа для обеих библиотек), а вот в plotly всё сразу хорошо, кроме того, становится зрительно понятно, что означают эти ящики с усами – внутри ящика по 50% и по столько же в усах.

Наконец, из простых вещей рассмотрим гистограммы.

Matplotlib:

# hist matplotlib

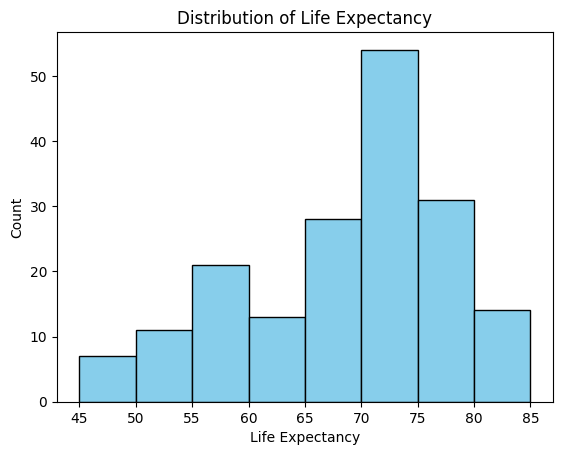
plt.hist(df['Life\_expectancy'], range=(45,85),bins=8, color='skyblue', edgecolor='black')

plt.xlabel('Life Expectancy')

plt.ylabel('Count')

plt.title('Distribution of Life Expectancy')

plt.show()

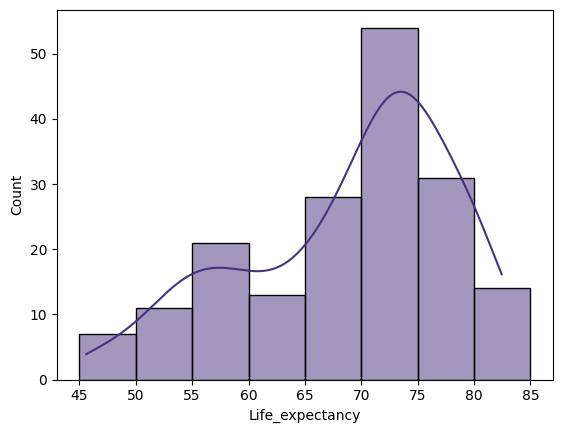


В seaborn приведём даже две картинки:

# hist seaborn

sns.histplot(df, x='Life\_expectancy', palette=sns.set\_palette("viridis"), kde=True,binwidth=5,binrange=(45,85))

plt.show()

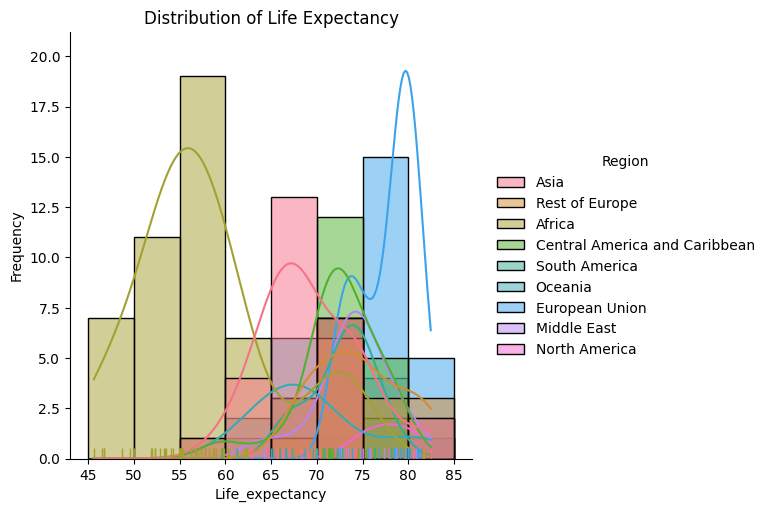


sns.displot(df, x='Life\_expectancy', hue='Region', kde=True, rug=True,bins=8,binrange=(45,85))

plt.ylabel("Count")

plt.title("Distribution of Life Expectancy")

plt.show()



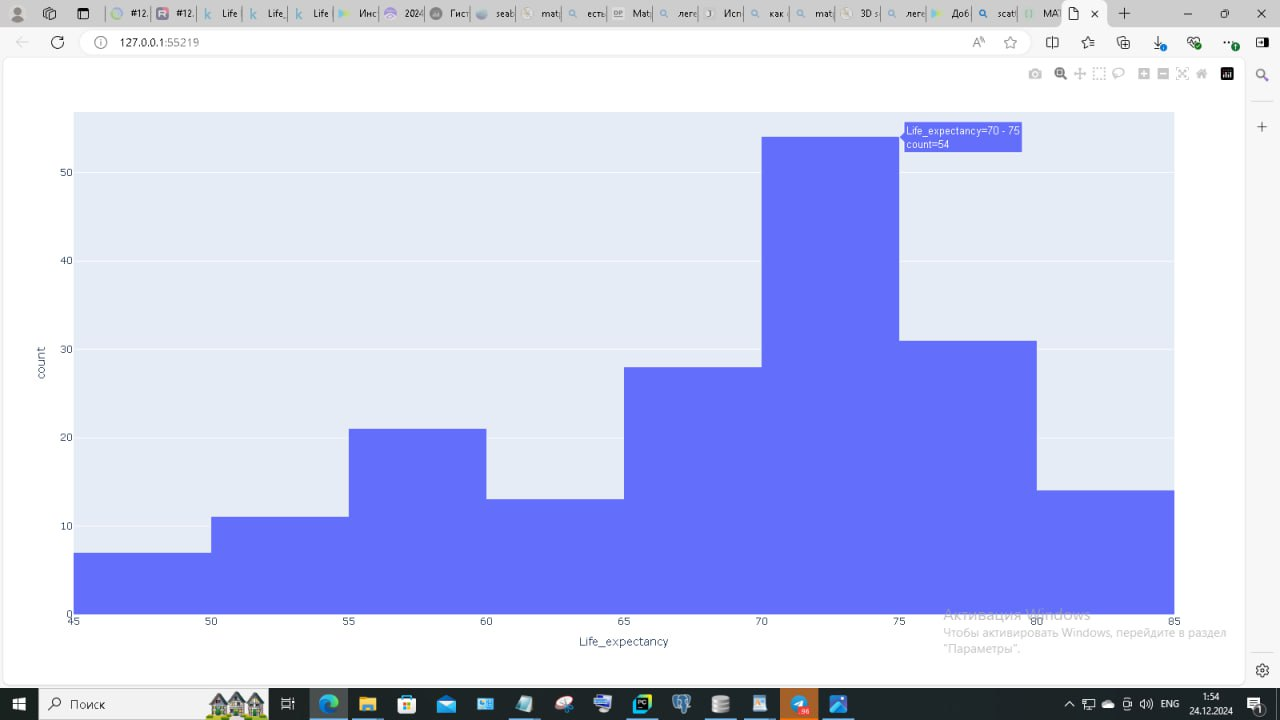
В seaborn оказалось легко добавить график плотности, просто присвоив kde=True, в Matplotlib , быстро найти не получилось. Вторая картинка показывает, как просто в seaborn сгруппировать гистограммы по регионам.

В plotly:

# hist plotpy

fig=px.histogram(df, x='Life\_expectancy', labels={'y':'Frequency'},height=600,width=600)

fig.show()



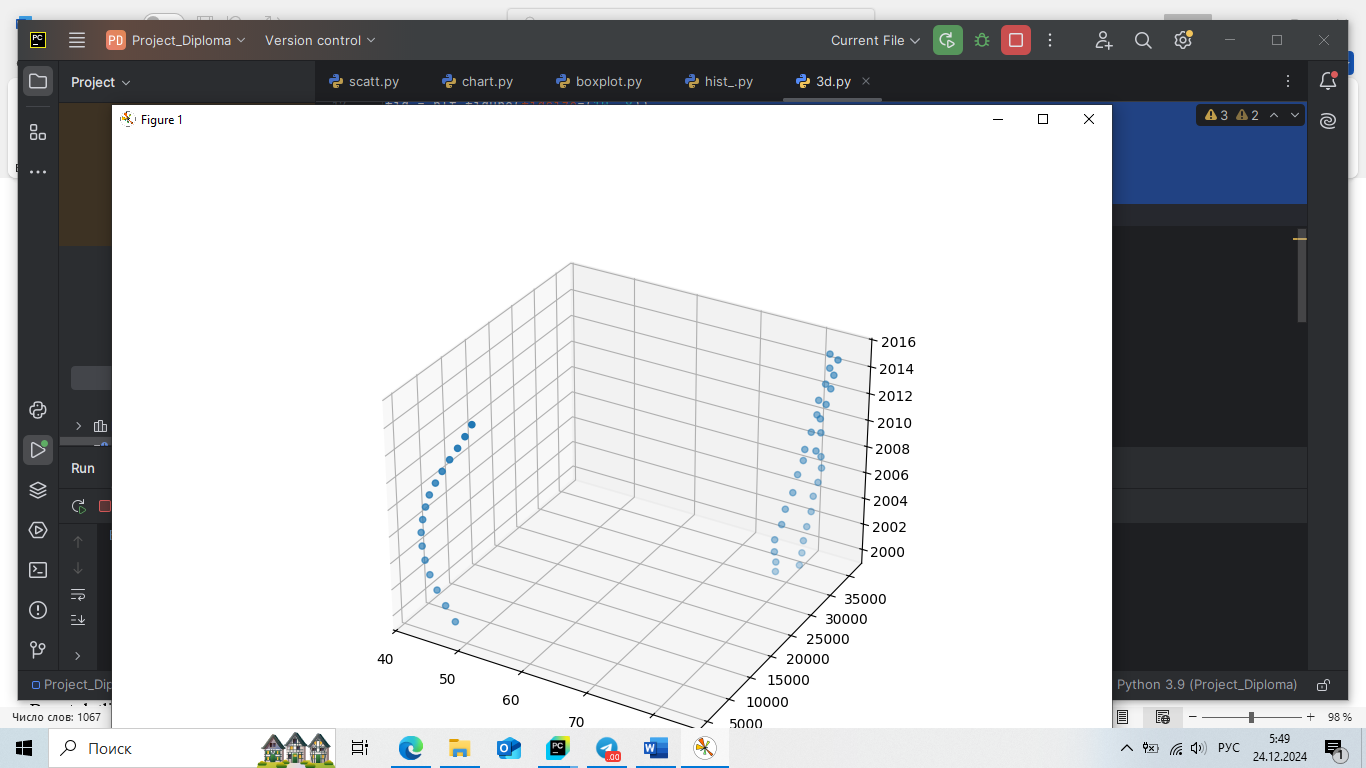
Надо заметить, что plotly сам понял, что бины нужно рисовать от 45 до 85 шириной 5лет, в двух предыдущих случаях над этим пришлось поработать.

Если же нужно получить 3d изображения, то seaborn отпадает, приведу три примера.

На основании тех же данных строится зависимость продолжительности жизни от ВВП по годам для трёх стран – Лесото, Израиля и Японии.

import matplotlib.pyplot as plt  
import pandas as pd  
import numpy as np

df = pd.read\_csv("Life-Expectancy-Data-Averaged (1).csv")  
df\_year = pd.read\_csv("Life-Expectancy-Data-Updated.csv")  
sel = ['Japan', 'Lesotho', 'Israel']  
df\_selected = df\_year[df\_year['Country'].isin(sel)]  
df\_selected = df\_selected.sort\_values(by='Year')  
  
for i in df\_selected['Year']:  
 x = df\_selected['Life\_expectancy']  
 y = df\_selected['GDP\_per\_capita']  
 z = df\_selected['Year']  
fig = plt.figure(figsize=(10, 8))  
ax = fig.add\_subplot(111, projection='3d')  
ax.scatter(x, y, z)  
plt.show()



В matplotlib поверхность, заданная функционально:

import matplotlib.pyplot as plot

def FUNC\_Z(x, y):

return x\*\*2 + y\*\*3

N = 50

X\_VAL = np.linspace(-5, 5, N)

Y\_VAL = np.linspace(-5, 5, N)

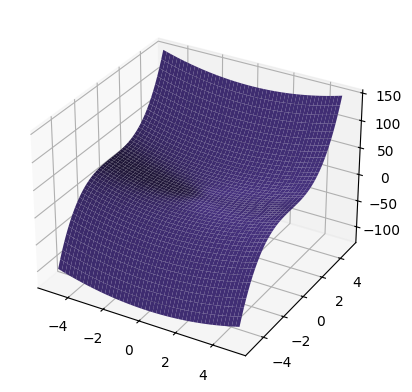
X1, Y1 = np.meshgrid(X\_VAL, Y\_VAL)

Z1 = FUNC\_Z(X1, Y1)

axes = plot.axes(projection="3d")

axes.plot\_surface(X1, Y1, Z1)

plot.show()



И в plotly поверхность, заданная массивом данных (гора Бруно):

import plotly.graph\_objects as go

import pandas as pd

# Read data from a csv

z\_data = pd.read\_csv('https://raw.githubusercontent.com/plotly/datasets/master/api\_docs/mt\_bruno\_elevation.csv')

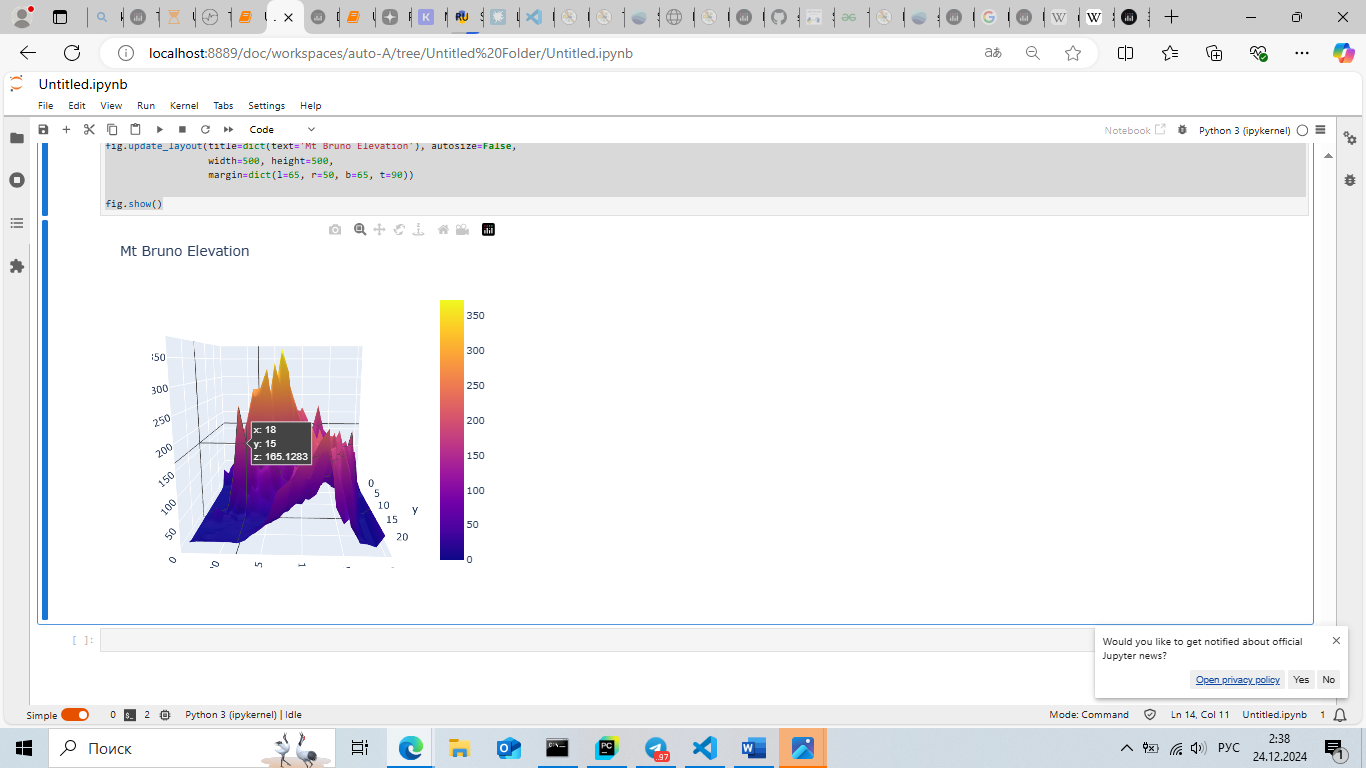
fig = go.Figure(data=[go.Surface(z=z\_data.values)])

fig.update\_layout(title=dict(text='Mt Bruno Elevation'), autosize=False,

width=500, height=500,

margin=dict(l=65, r=50, b=65, t=90))

fig.show()



Здесь я не сильно разобрался, что лучше подходит для 3d-визуализации, но, судя по последней картинке и коду(честно взятого из сети), plotly легче использовать в этих целях.

Заключение.

Здесь могу повторить все характеристики этих библиотек, которые за последнюю неделю я выучил уже наизусть: matplotlib основательный и дотошный(«гибкий»), однако сложный в освоении, seaborn создан на основе matplotlib для облегчения его использования, за счёт вложенных настроек,

с помощью plotly легко выполнить любую визуализацию, да ещё и пользоваться ею интерактивно. На основании только своего опыта очень согласен с последним утверждением, кроме того, объём официальной документации, с которым пришлось столкнуться, убеждает, что matplotlib, наверное, может всё.

Кроме того, бросается в глаза, что plotly просто заточен для сети и публикаций.