

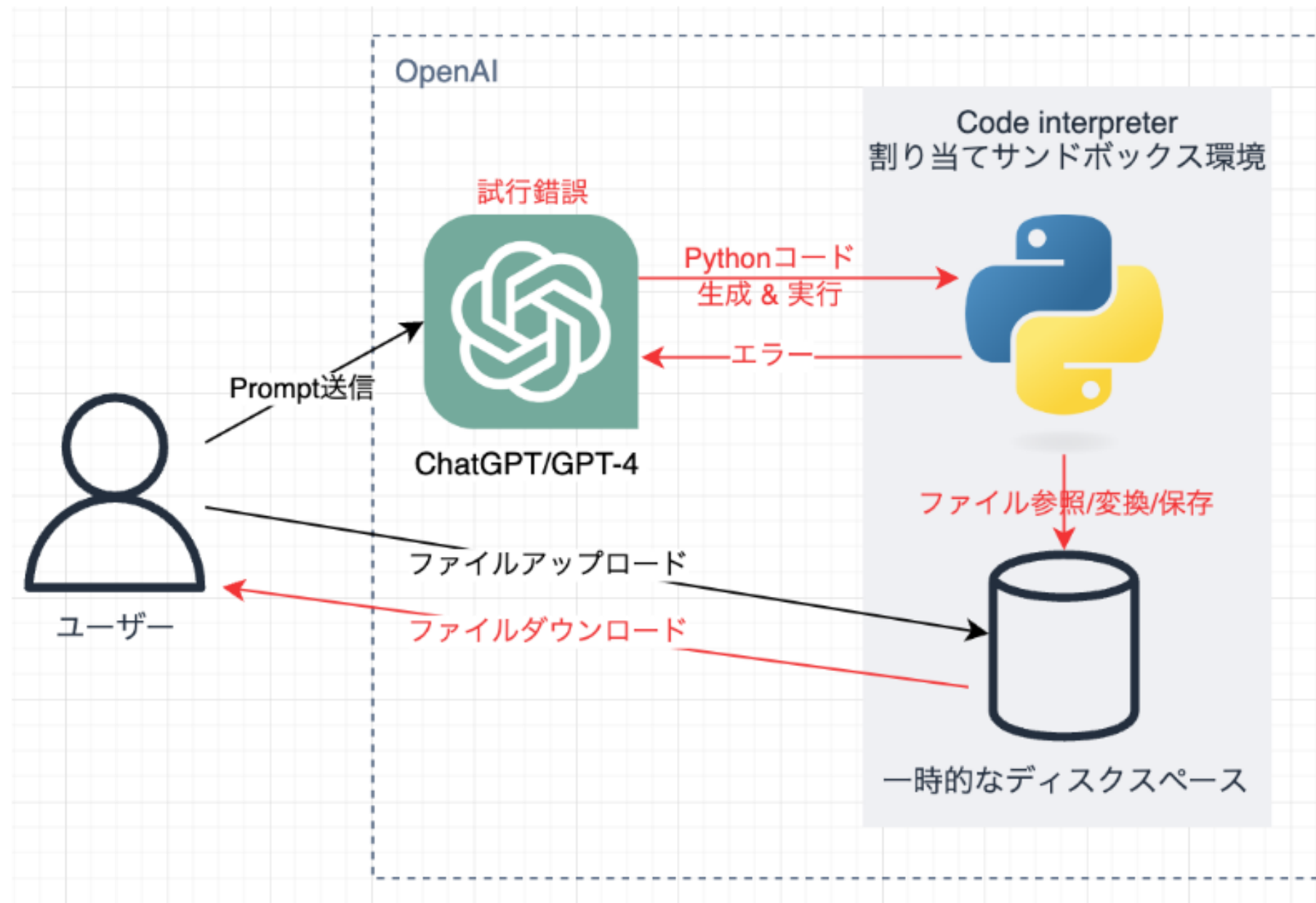
# ChatGPT の Code Interpreterについて

2023/8/30

# ChatGPT の Code Interpreterとは？

- ChatGPT内でpythonコードの生成・実行が可能となるもの
  - コードの生成まではこれまでもできたが、作成したコードを実行し、その結果（グラフや計算結果など）を返してくれるようになった
  - またファイルのアップロードも可能になった（512MBまで）
- 
- 2023年7月7日にOpenAIがリリース
  - 現在は有料プラン「ChatGPT Plus」のユーザーのみが使える

# Code Interpreterの仕組みのイメージ

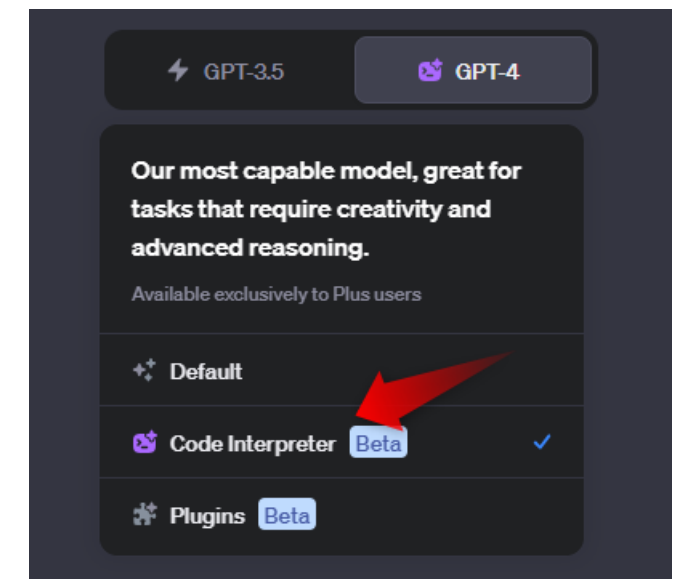
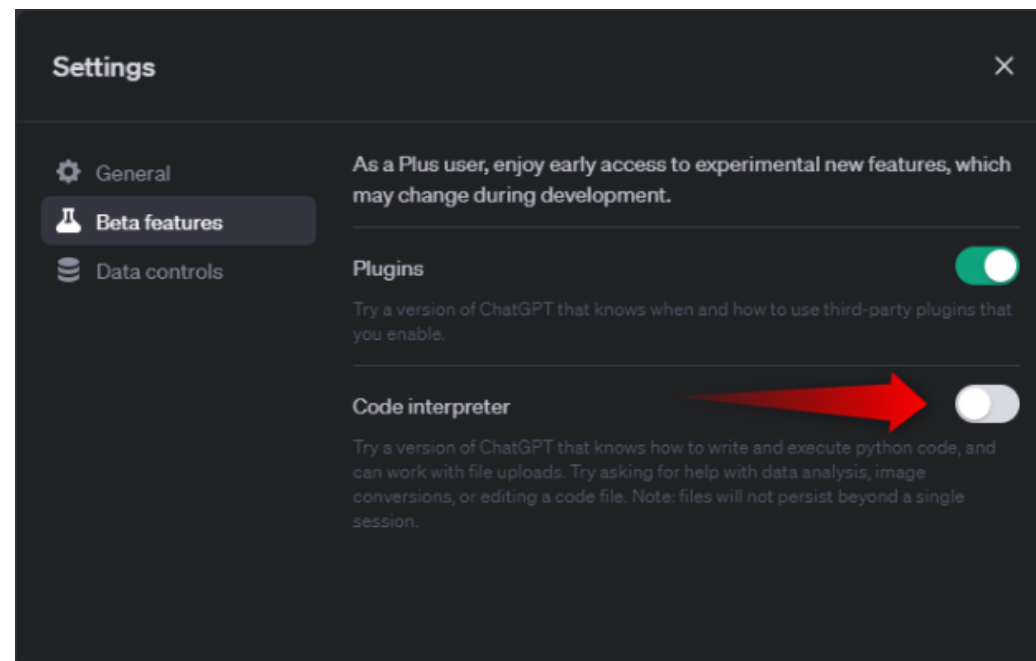
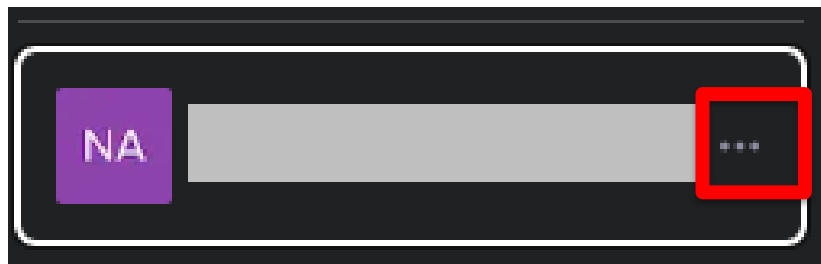


(引用)

<https://www.bioerrorlog.work/entry/code-interpreter-chatgpt#Code-interpreter%E3%81%AE%E4%BB%95%E7%B5%84%E3%81%BF%E3%82%92%E7%90%86%E8%A7%A3%E3%81%99%E3%82%8B>

# ChatGPTを使うには

- ChatGPT内の左下のアカウント情報右の3点リーダーをクリック
- 「Beta features」をクリックし、「Code interpreter」を有効に
- Top画面で新たにChatを開始する際に「GPT-4」の「Code Interpreter」を選択



# (参考) ChatGPT の Code Interpreterで読み込めるファイル

- テキストファイル
- 画像ファイル
- 動画ファイル
- CSVファイル
- Excelファイル
- PDFファイル
- ZIPファイル
- Pythonスクリプト
- Jupyterノートブック
- GIFファイル
- HTMLファイル

# Code Interpreterでできそうなこと

- スライド作成
- 計算の実行
- グラフの作成
- QRコード生成
- 画像処理・音声処理
- コードの解説
- ワードクラウドの作成
- データ分析

# スライド作成

Pythonのライブラリ(python-pptx)を使用することで、パワーポイントのファイルを作成することができる

## プロンプト例

目的: 多変量解析の基礎を説明するための資料

ターゲット: 多変量解析を行っていない学生

主要テーマ: 多変量解析がどのようなものなのか、分析の方法を事例を挙げながら説明

スライド枚数: 30枚

スライド構造: お任せします。

フォーマット: 16:9ワイド

使用するコード: python pptx

# スライド作成

まずはスライドの構成を考えてくれる



了解しました。多変量解析の基礎について説明するためのパワーポイントスライドを作成します。



スライドの構造は以下のように考えています：

1. タイトルスライド
2. 目次
3. はじめに
4. 単変量解析とは？
5. 多変量解析とは？
6. なぜ多変量解析が必要なのか？
7. 多変量解析の種類
  - 主成分分析 (PCA)
  - 因子分析
  - クラスター分析
  - 重回帰分析
  - 識別分析
  - 共分散構造分析

☐ Stop generating



# スライド作成


プログラムを書いてスライドの作成が始まる

それでは、この構造に基づいてパワーポイントスライドを作成していきます。

Finished working

Hide work ^

python

 Copy code

```
from pptx import Presentation
from pptx.util import Inches
from pptx.enum.text import PP_ALIGN
from pptx.util import Pt
from pptx.dml.color import RGBColor

# Initialize a presentation object
prs = Presentation()
prs.slide_width = Inches(16)
prs.slide_height = Inches(9)

# Function to add bullet points
def add_bullets(slide, title, content_list):
    slide.shapes.title.text = title
    bullet_slide = slide.shapes.placeholders[1]
    tf = bullet_slide.text_frame
    for content in content_list:
        p = tf.add_paragraph()
        p.text = content
        p.level = 0
```

# スライド作成

```
-----  
NameError                                Traceback (most recent call last)  
Cell In[3], line 4  
      1 # Reinitialize the presentation object and helper functions due to the interrupte  
      2  
      3 # Initialize a presentation object  
----> 4 prs = Presentation()  
      5 prs.slide_width = Inches(16)  
      6 prs.slide_height = Inches(9)  
  
NameError: name 'Presentation' is not defined
```

エラーが出た場合も内容を判断し、ChatGPTが勝手に再試行してくれる

お手数をおかけして申し訳ありません。必要なライブラリがインポートされていないようです。再度、必要な設定を行い、スライドの作成を続けます。

完成したスライドをダウンロードするだけ

スライドの作成が完了しました。[こちらからダウンロード](#)してください。何か他にお手伝いできることはありますか？

# スライド作成

ファイルを添付することも可能なので、論文紹介資料の作成を依頼

以下のパワーポイントスライドを作成してください。

目的: 添付した論文の内容紹介

ターゲット: 分析チームメンバー

主要テーマ: 論文の概要と結果について紹介

スライド枚数: 10枚

スライド構造: お任せします。

フォーマット: 16:9ワイド

使用するコード: python pptx

伝えたい内容: (添付資料)

Xiv:2201.07006v3 [cs.LG] 19 May 2022

## TIME SERIES GENERATION WITH MASKED AUTOENCODER

**Mengyue Zha**

School of Mathematics

The Hong Kong University of Science and Technology

mzha@ust.hk

**SiuTim Wong**

School of Engineering

The Hong Kong University of Science and Technology

stwongak@ust.hk

**Mengqi Liu**

School of Statistics

Beijing Normal University

mqliu@bnu.edu.cn

**Tong Zhang**

School of Mathematics

The Hong Kong University of Science and Technology

tongzhang@ust.hk

**Kani Chen**

School of Mathematics

The Hong Kong University of Science and Technology

makchen@ust.hk

May 20, 2022

### ABSTRACT

This paper shows that masked autoencoder with extrapolator (ExtraMAE) is a scalable self-supervised model for time series generation. ExtraMAE randomly masks some patches of the original time series and learns temporal dynamics by recovering the masked patches. Our approach has two core designs. First, ExtraMAE is self-supervised. Supervision allows ExtraMAE to effectively and efficiently capture the temporal dynamics of the original time series. Second, ExtraMAE proposes an extrapolator to disentangle two jobs of the decoder: recovering latent representations and mapping them back into the feature space. These unique designs enable ExtraMAE to consistently and significantly outperform state-of-the-art (SoTA) benchmarks in time series generation. The lightweight architecture also makes ExtraMAE fast and scalable. ExtraMAE shows outstanding behavior in various downstream tasks such as time series classification, prediction, and imputation. As a self-supervised generative model, ExtraMAE allows explicit management of the synthetic data. We hope this paper will usher in a new era of time series generation with self-supervised models.

# スライド作成


- タイトルと少量のテキストが出てくるぐらいで、現状そのまま使えるようなレベルスライドは作れない
- テキストのみなので、レイアウト調整も必要
- 正直一から自分で作った方が早い

# 計算の実行

- 積分もしてくれる

NA

(x+5)(x-2)を0から1まで積分してください。



与えられた関数  $(x + 5)(x - 2)$  を  $x = 0$  から  $x = 1$  まで積分するためには、まずその関数を展開し  
てみましょう。

$$(x + 5)(x - 2) = x^2 - 2x + 5x - 10 = x^2 + 3x - 10$$

次に、この展開された関数  $x^2 + 3x - 10$  を  $x = 0$  から  $x = 1$  まで積分します。数学的には以下の  
ように表現されます。

$$\int_0^1 (x^2 + 3x - 10) dx$$




この積分を計算してみましょう。

Finished working

Show work ▼

$x = 0$  から  $x = 1$  までの  $(x + 5)(x - 2)$  の積分結果は約  $-8.17$  です。

何か他に質問や課題はありますか？



# 計算の実行

- 内部でPythonコードを生成し、それを実行して結果を得ている

Finished working Hide work ^

python Copy code

```
from scipy.integrate import quad

# Define the function to be integrated
def func(x):
    return x**2 + 3*x - 10

# Perform the integration from 0 to 1
integral_result, _ = quad(func, 0, 1)

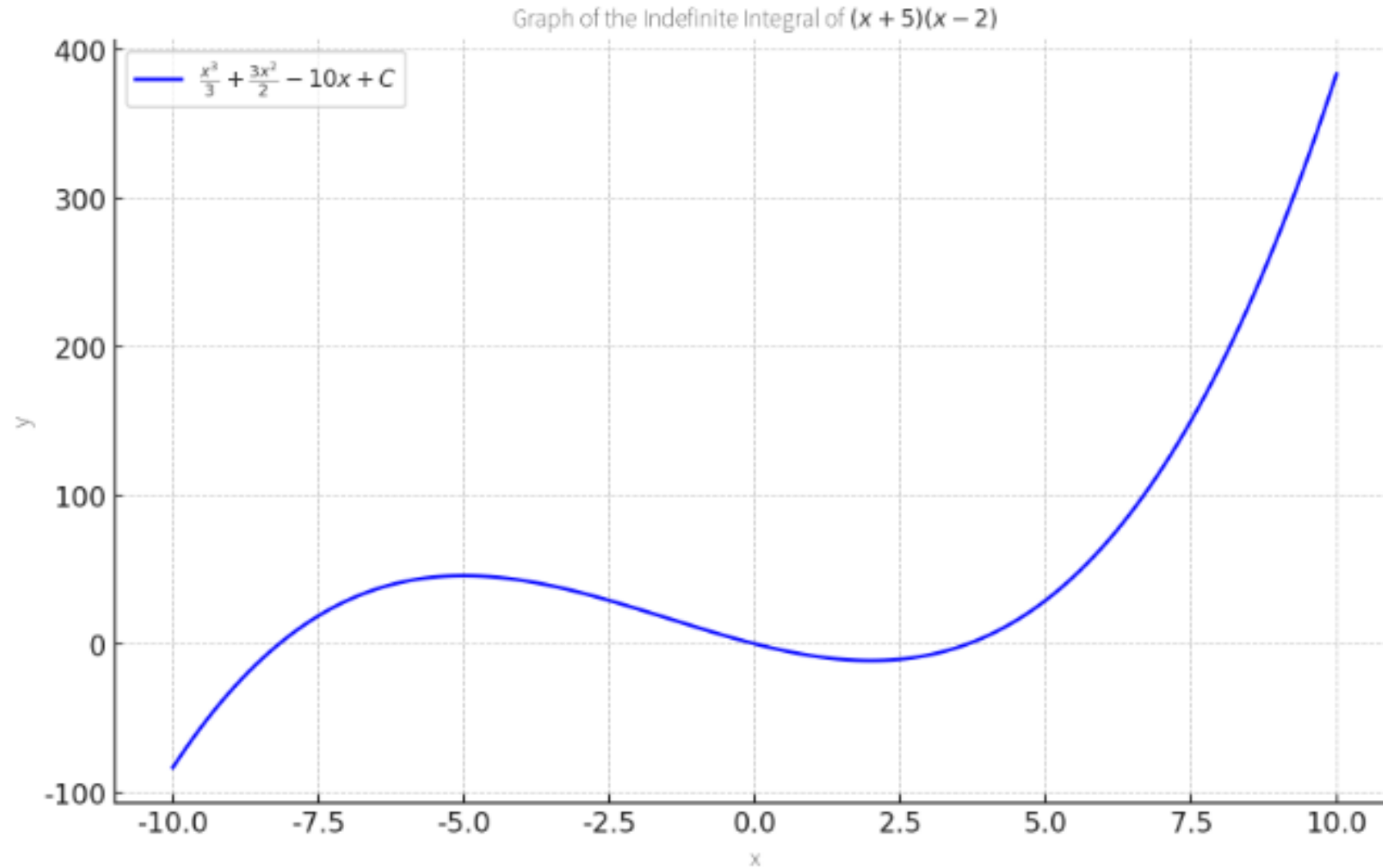
integral_result
```

RESULT

-8.166666666666668

# グラフの作成

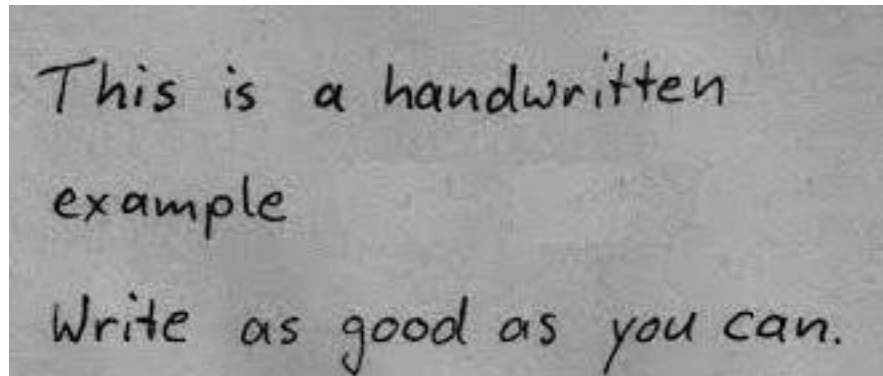
- 先ほどの関数を不定積分した結果をグラフにするよう依頼



$(x + 5)(x - 2)$  の不定積分  $\frac{x^3}{3} + \frac{3x^2}{2} - 10x + C$  のグラフを作成しました。ここで  $C$  は積分定数で、このグラフでは  $C = 0$  としています。

# OCR

- 手書き文字の画像をアップロードして、書かれているテキストの抽出を依頼



※現状日本語には対応していない

添付画像に書かれている英語のテキストは以下のとおりです：

csharp

Copy code

```
This is a handwritten  
example  
  
Write as gooal as you can.
```

注意: "gooal" の部分はおそらく "good" の手書きが誤って読み取られた可能性があります。

pythonでは機械的に文字お越しするだけであるが、文法的におかしい部分など指摘してくれるところはメリットか



# QRコード生成

- QRコードをも作成できる

python

 Copy code

```
import qrcode

# Create a QR code instance
qr = qrcode.QRCode(
    version=1, # controls the size of the QR Code
    error_correction=qrcode.constants.ERROR_CORRECT_L, # controls the error
    box_size=10, # controls how many pixels each "box" of the QR code is
    border=4, # controls how many boxes thick the border should be
)
```



# 画像処理・音声処理

- 拡張子の変換(pngをjpegに変換)
- 画像をグレースケールに変換
- GIFアニメの作成
- スペクトル分析

Original Image



Grayscale Image



# ワードクラウドの作成

- プロンプト: 添付ファイルの内容をもとにWord Cloudを作成してください。  
(添付したのは直近のMinutes)



- PDFからのテキスト抽出がうまくいっていない？
- NLTKも追加ダウンロードが必要なものは使えない

申し訳ありません、NLTK (Natural Language Toolkit) の一部のリソースがインターネットからダウンロードできないため、エラーが発生しました。

# ワードクラウドの作成

- HTMLファイルに変更して実施したところ多少きれいになった。





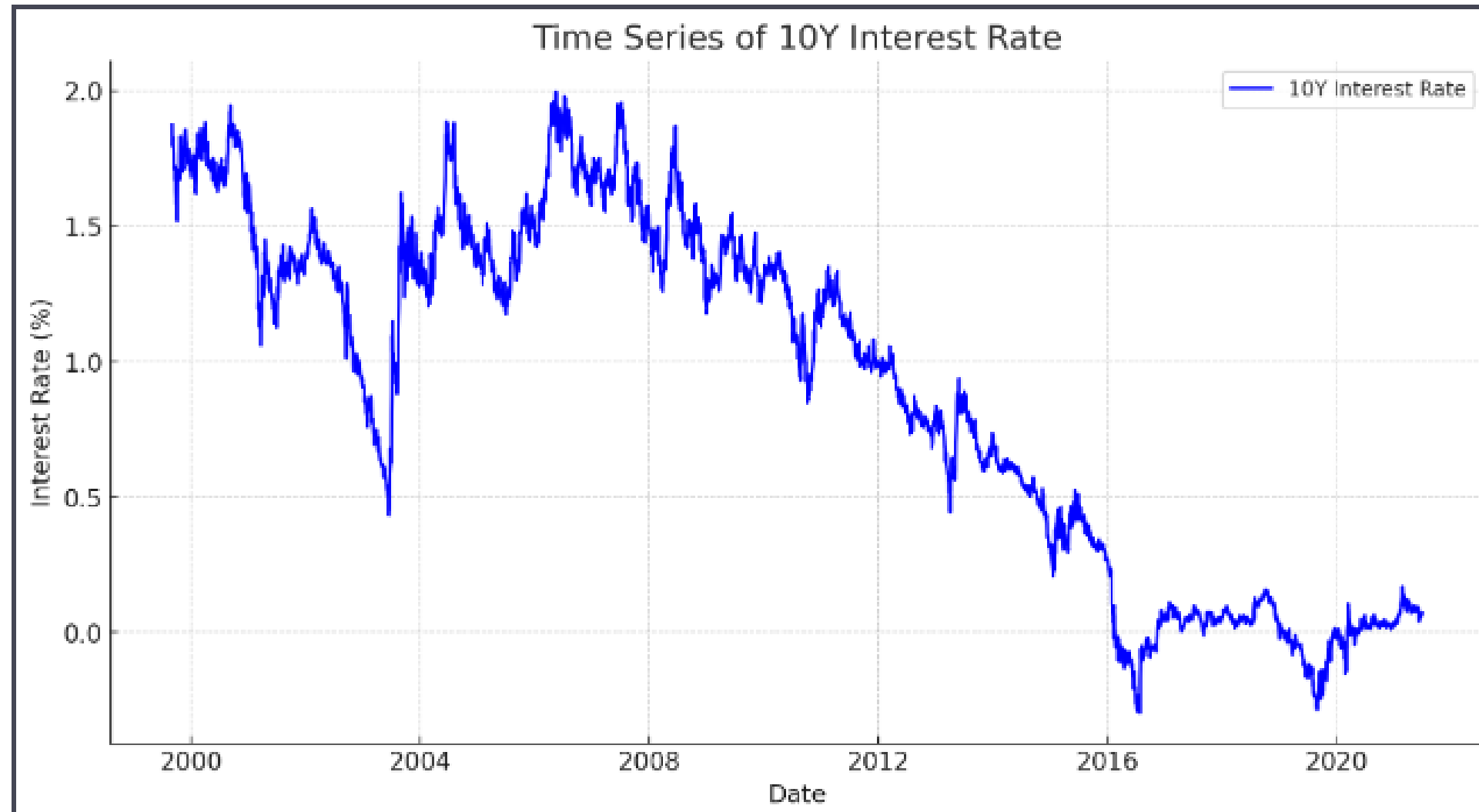
# 分析活用

- サンプルデータ: JGBの金利(日次)

Date	1Y	2Y	3Y	4Y	5Y	6Y	7Y	8Y	9Y	10Y	15Y	20Y	30Y
1999/9/2	0.138	0.358	0.525	0.84	1.105	1.339	1.508	1.688	1.762	1.876	2.409	2.664	2.842
1999/9/3	0.137	0.336	0.495	0.807	1.069	1.303	1.477	1.651	1.725	1.838	2.387	2.649	2.822
1999/9/6	0.134	0.326	0.482	0.775	1.021	1.25	1.422	1.596	1.678	1.795	2.36	2.623	2.808
1999/9/7	0.144	0.351	0.528	0.825	1.065	1.289	1.456	1.626	1.705	1.82	2.392	2.644	2.825
1999/9/8	0.149	0.364	0.543	0.842	1.075	1.296	1.456	1.622	1.703	1.815	2.391	2.64	2.826
1999/9/9	0.159	0.38	0.567	0.866	1.102	1.323	1.483	1.642	1.72	1.829	2.405	2.653	2.834
1999/9/10	0.158	0.375	0.562	0.853	1.084	1.301	1.456	1.608	1.69	1.8	2.37	2.631	2.811
1999/9/13	0.158	0.363	0.543	0.825	1.051	1.263	1.415	1.567	1.655	1.77	2.335	2.605	2.785
1999/9/14	0.142	0.326	0.482	0.763	0.983	1.193	1.346	1.505	1.604	1.72	2.291	2.57	2.748
1999/9/16	0.13	0.31	0.463	0.732	0.945	1.155	1.31	1.476	1.586	1.709	2.286	2.567	2.739
1999/9/17	0.13	0.3	0.441	0.714	0.932	1.15	1.309	1.48	1.596	1.722	2.307	2.585	2.739
1999/9/20	0.12	0.255	0.37	0.636	0.861	1.081	1.243	1.414	1.537	1.668	2.258	2.547	2.688
1999/9/21	0.118	0.26	0.367	0.639	0.88	1.11	1.277	1.442	1.56	1.693	2.285	2.564	2.703
1999/9/22	0.138	0.29	0.422	0.698	0.944	1.172	1.34	1.491	1.595	1.722	2.321	2.594	2.728
1999/9/24	0.126	0.27	0.378	0.643	0.882	1.112	1.274	1.412	1.516	1.63	2.232	2.51	2.656

# 分析活用

- カラム10Yを全期間プロットするよう依頼



※カラム'10Y'だが、おそらくファイル名などから10年債利回りであることを把握して、勝手にタイトルや軸名を付けてくれている

# 分析活用

- 単位根検定も一瞬でやってくれる

NA

各カラムのデータが単位根過程であるかどうかを分析してください。



単位根過程（unit root process）は、時間序列データが定常性（stationarity）を持っているかどうかをテストする一般的な方法です。定常性がある場合、データの統計的特性（平均、分散など）は時間に依存しないので、予測モデルがより安定します。

単位根過程を検定する一般的な方法は、Augmented Dickey-Fuller（ADF）テストです。このテストの帰無仮説（null hypothesis）は、データが単位根過程である（すなわち、非定常である）というものです。したがって、p値が低い（通常は0.05以下）場合、帰無仮説を棄却してデータが定常であると結論づけれます。

各金利カラムに対してADFテストを実施してみましょう。

Augmented Dickey-Fuller（ADF）テストの結果は以下の通りです：


- すべての金利カラム（`1Y`から`30Y`）でのp値が0.05よりも大きいです。
- これにより、帰無仮説（データが単位根過程である、すなわち非定常である）を棄却できません。

以上の結果から、各金利カラムは単位根過程である可能性が高く、定常ではないと考えられます。

何か他に分析したい点や質問はありますか？

# 分析活用

- 差分データへの変換

 確かに、単位根過程のデータはしばしば差分を取ることで定常になります。各金利カラムについて、前日との差分を計算して新しいデータセットを作成しましょう。

Finished working   Hide work ^

python Copy code

```
# Calculate the day-to-day differences for each interest rate column
df_diff = df.iloc[:, 1:].diff().dropna()

# Include the corresponding dates for the differences
df_diff['Date'] = df['Date'].iloc[1:].reset_index(drop=True)

# Show the first few rows of the new DataFrame
df_diff.head()
```

RESULT

	1Y	2Y	3Y	4Y	5Y	6Y	7Y	8Y	9Y	10Y
1	-0.001	-0.022	-0.030	-0.033	-0.036	-0.036	-0.031	-0.037	-0.037	-0.038
2	-0.003	-0.010	-0.013	-0.032	-0.048	-0.053	-0.055	-0.055	-0.047	-0.043
3	0.010	0.025	0.046	0.050	0.044	0.039	0.034	0.030	0.027	0.025
4	0.005	0.013	0.015	0.017	0.010	0.007	0.000	-0.004	-0.002	-0.005
5	0.010	0.016	0.024	0.024	0.027	0.027	0.027	0.020	0.017	0.014

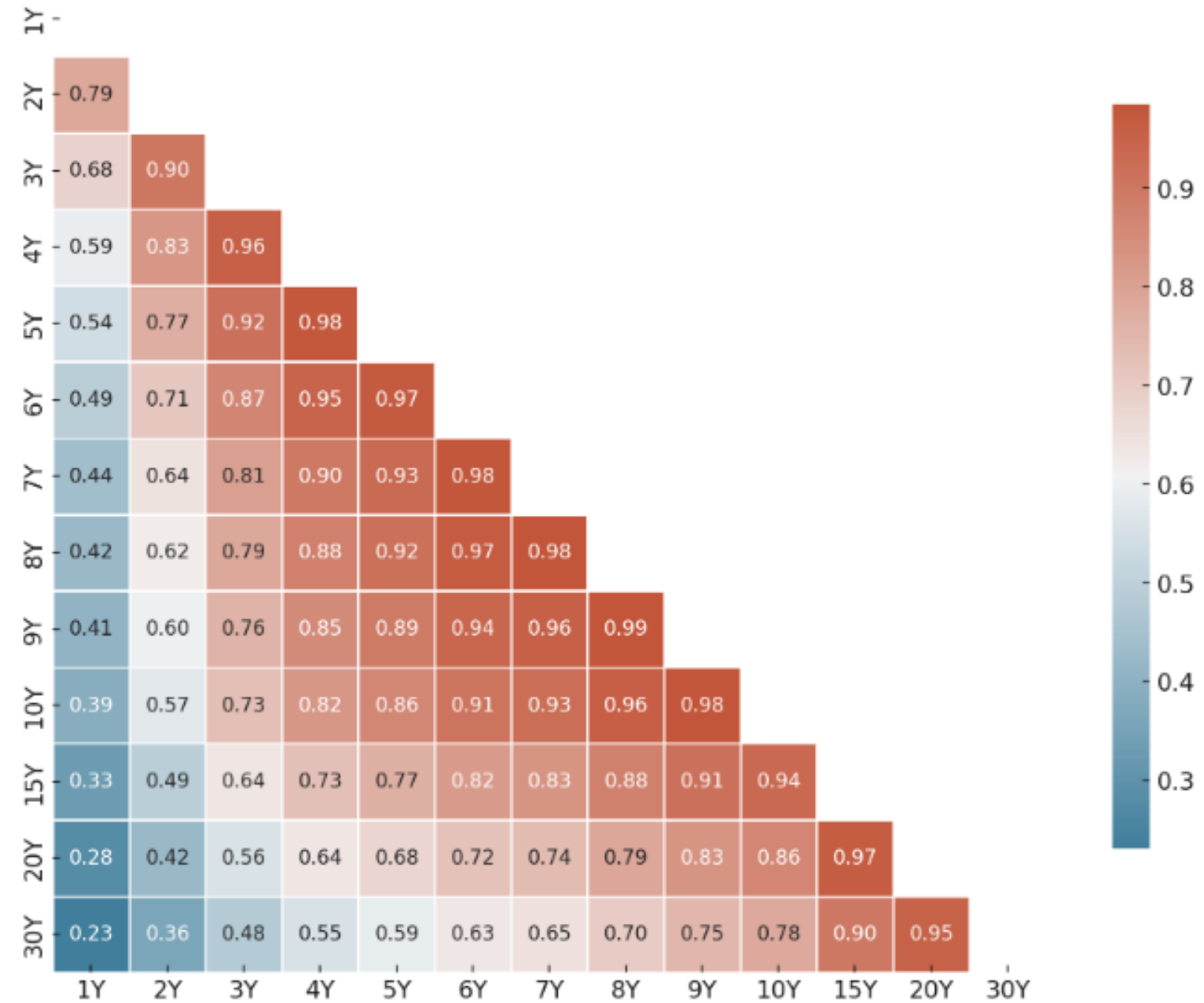
	15Y	20Y	30Y	Date
1	-0.022	-0.015	-0.020	1999-09-06
2	-0.027	-0.026	-0.014	1999-09-07
3	0.032	0.021	0.017	1999-09-08



# 分析活用

- 相関のヒートマップを作成

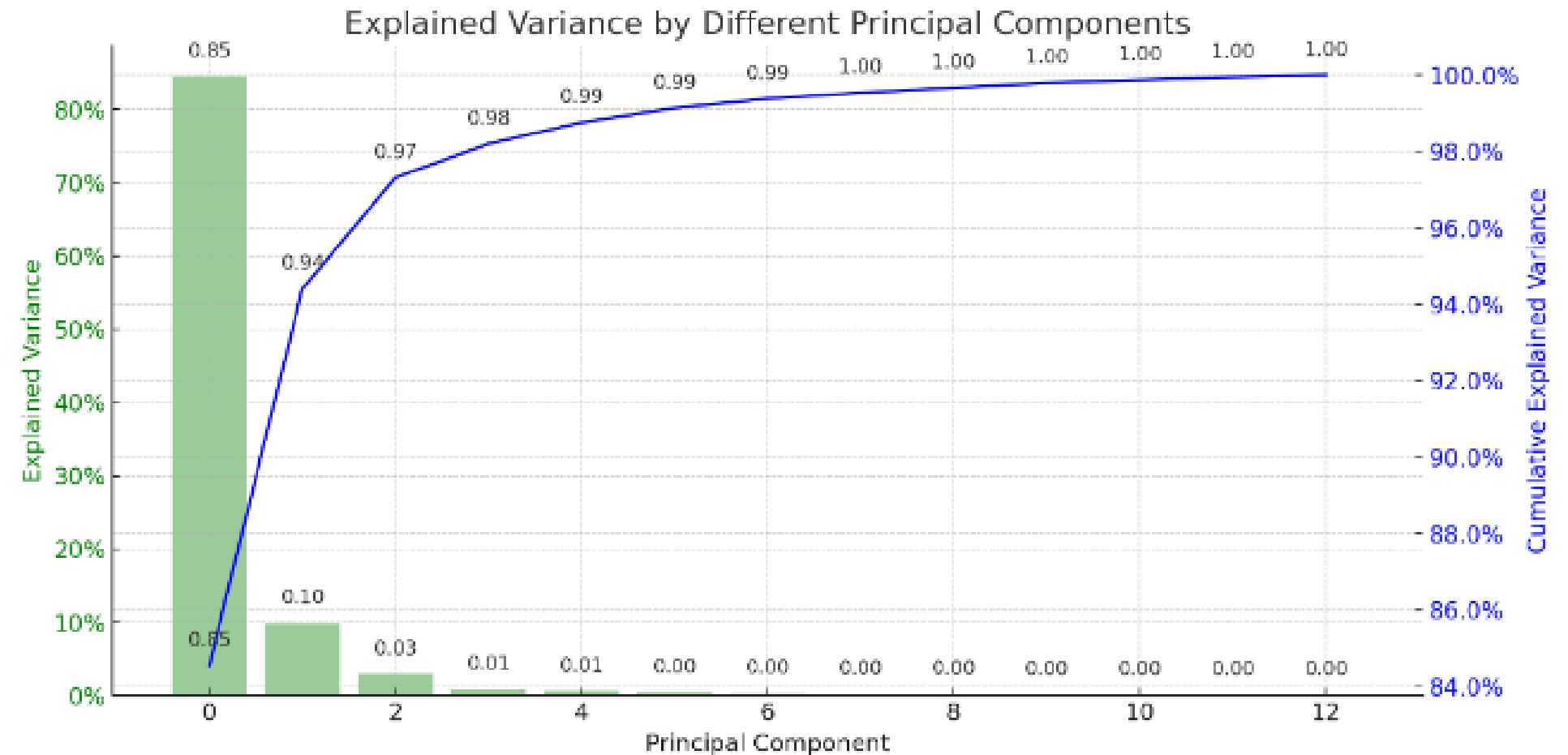
Correlation Matrix of Differenced Interest Rates for Different Periods



# 分析活用

- 主成分分析を依頼

‘主成分分析をしてください’という指示をただけで、細かい指示はしていないが、結果をグラフにしてくれたり、説明を加えてくれたりしてくれた

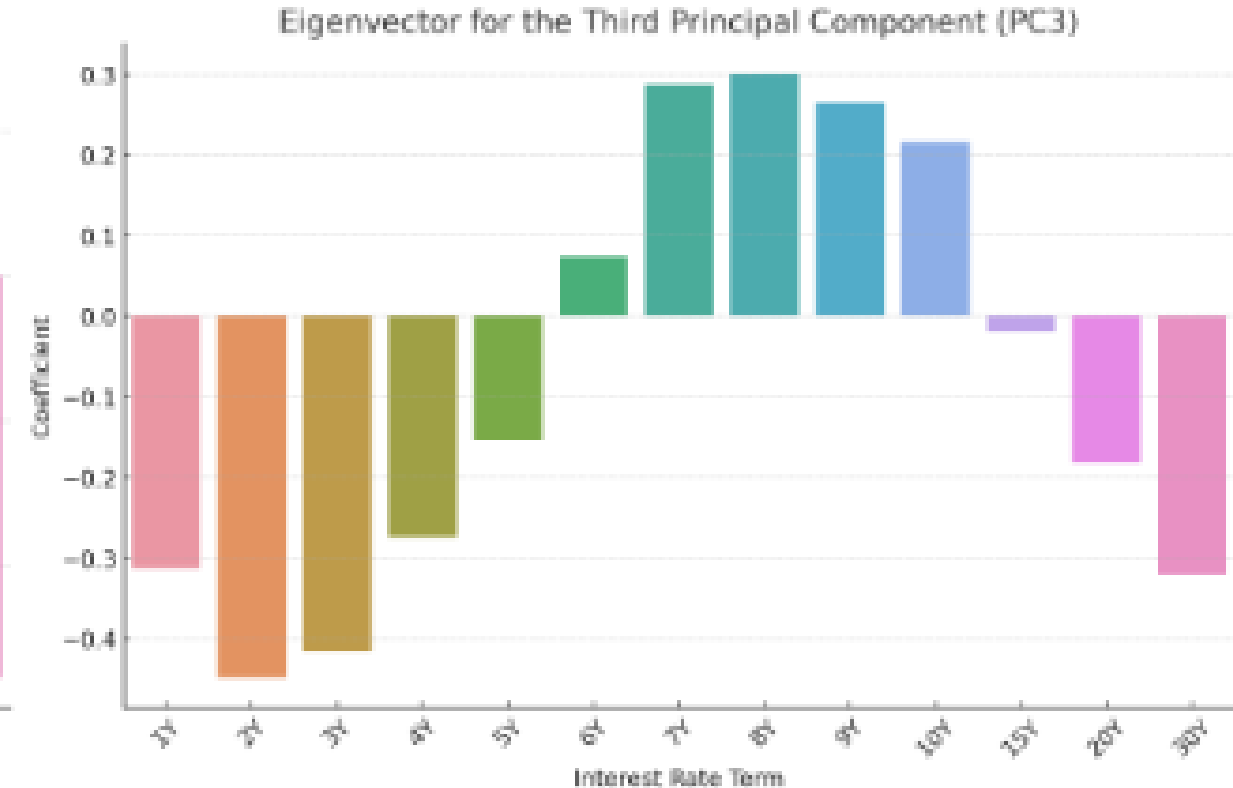
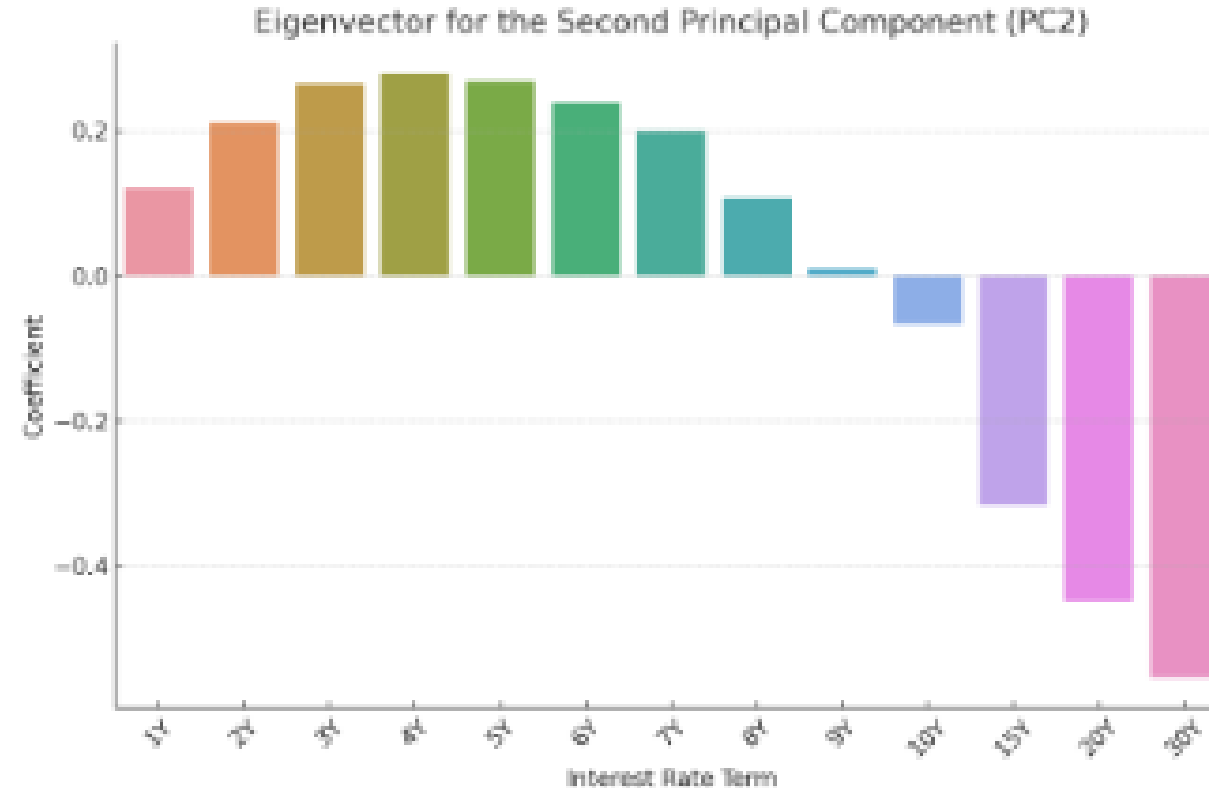
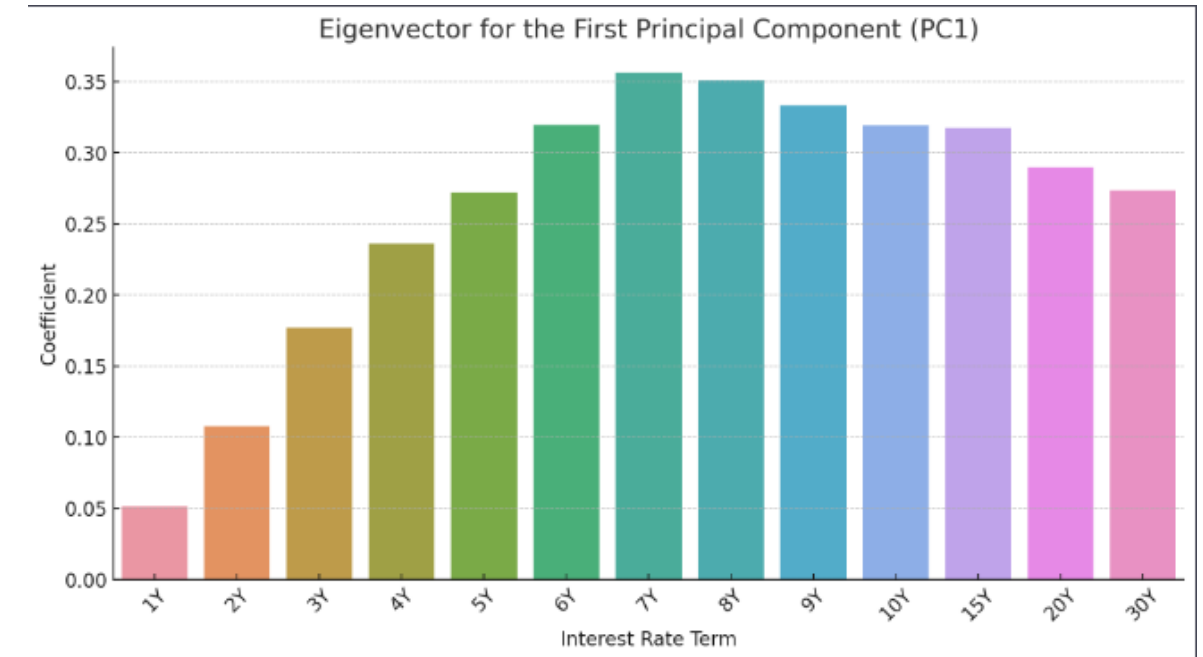


- 最初の主成分だけで大部分の分散（約80%以上）を説明しています。
- 2番目、3番目の主成分を追加すると、累積で90%以上の分散を説明します。

この結果から、元の13の変数（金利の各期間）の情報の大部分は、少数の主成分で表現することが可能であると言えます。

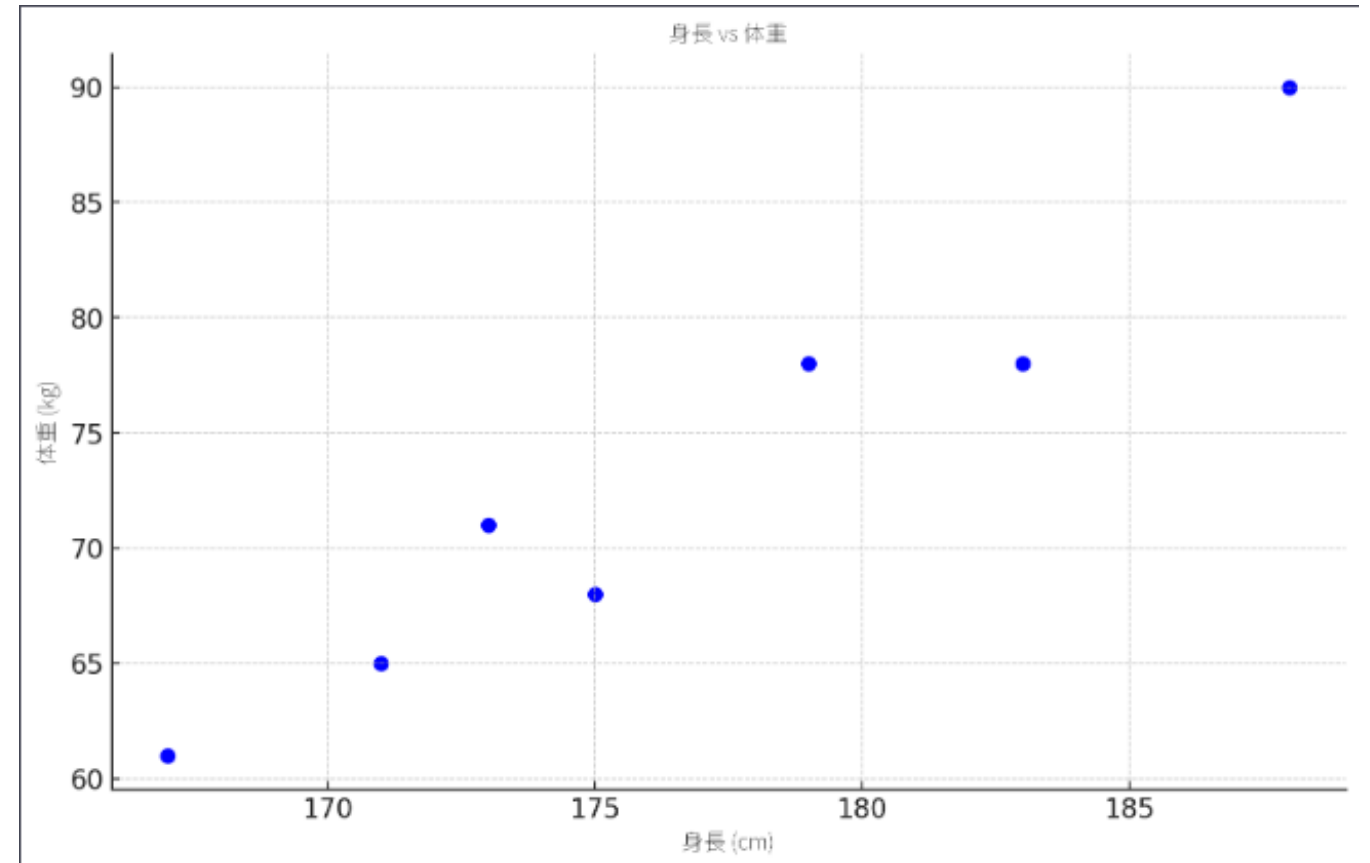
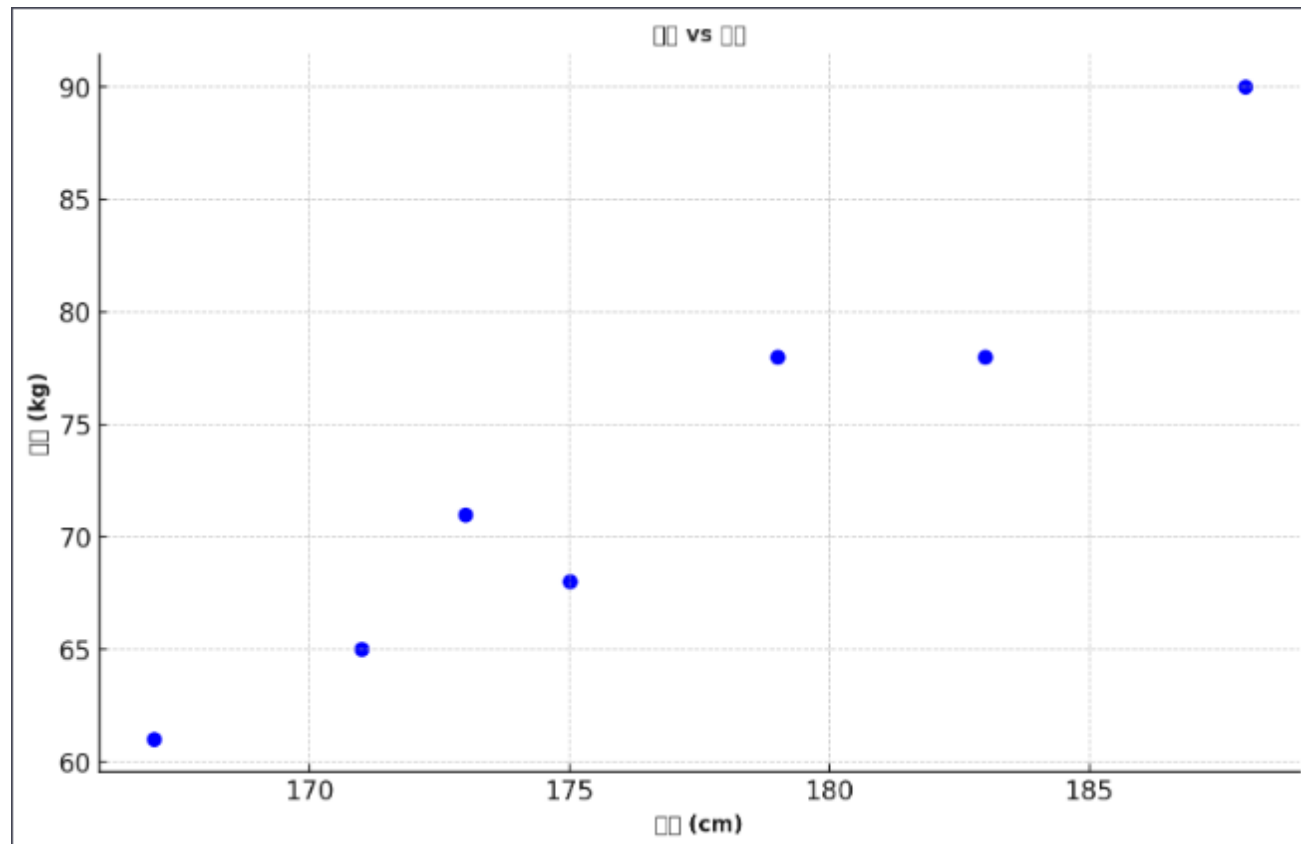
# 分析活用

- 固有ベクトルの可視化
  - 色使いはさておき、
    - 第1主成分: 水準
    - 第2主成分: 傾き
    - 第3主成分: 曲率
- という一般的な結果が得られた



# (補足) 日本語文字化け対応

日本語用のフォントファイル(ttf)と一緒にアップロードすれば、表示可能



日本語対応フォントの例: <https://fonts.google.com/noto/specimen/Noto+Sans+JP>

# まとめ

- Code Interpreterでは、pythonでできることであれば一通りできるイメージ
- ただし、pythonからのインターネットアクセスは無効化されており、外部サイトのスクレイピングなどはできない。
- またライブラリの追加ダウンロードもできない模様
  - ただし352個のライブラリがインストールされており、大概のことはできる

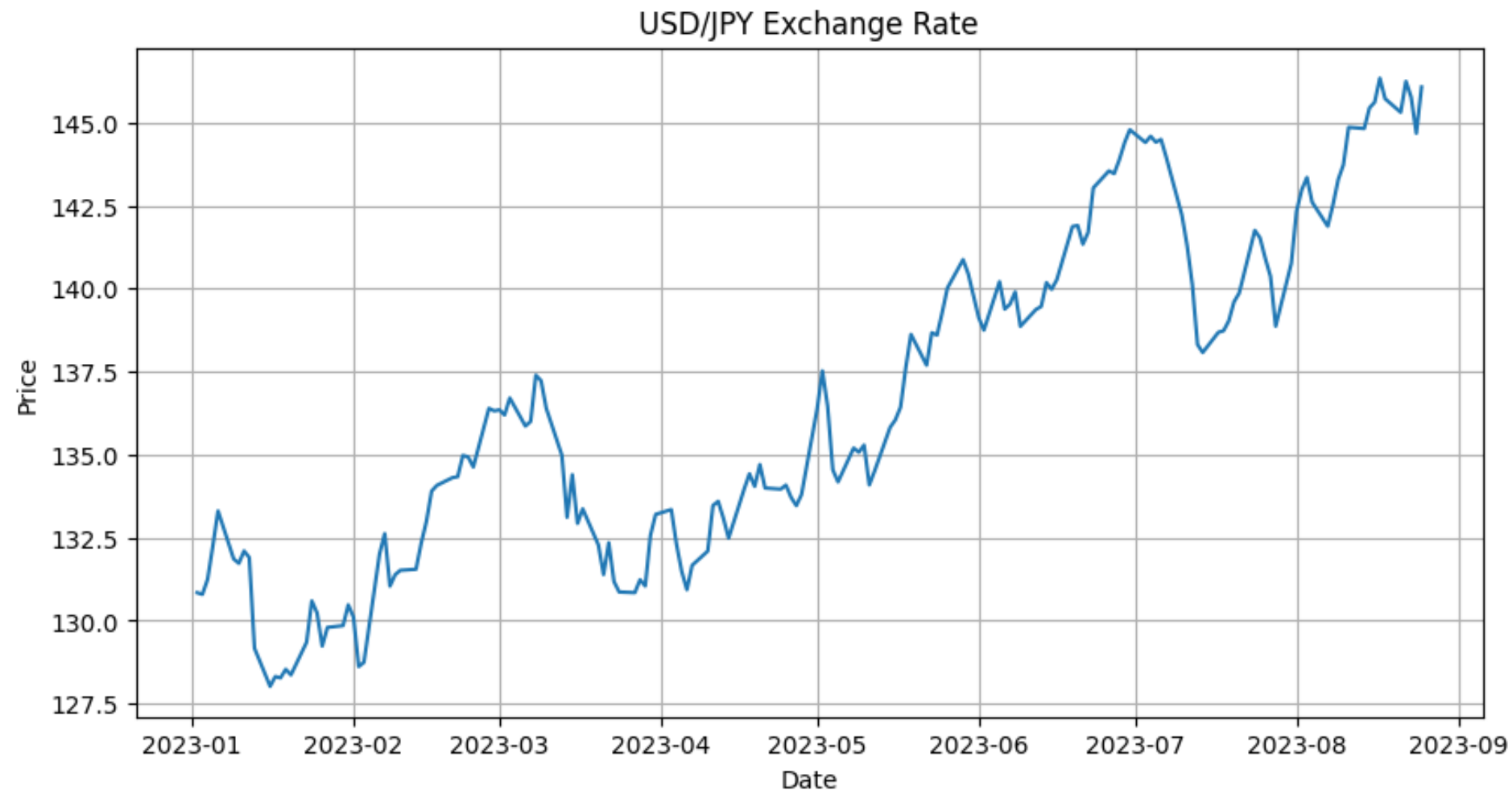
## 使用できる主なライブラリ

- NumPy: 数学的な配列操作
- Pandas: データフレーム操作
- Matplotlib: プロットと可視化
- SciPy: 科学計算
- scikit-learn: 機械学習
- Seaborn: 高度なデータ可視化
- statsmodels: 統計モデル
- Natural Language Toolkit (NLTK): 自然言語処理
- TensorFlow: 機械学習と深層学習
- PyTorch: 機械学習と深層学習
- PIL (Pillow): 画像処理
- SQLAlchemy: データベース操作

# 補足: CodeInterpreter-api

- 非公式ではあるが、LangChainを用いて実装したAPIが存在する  
<https://github.com/shroominic/codeinterpreter-api> (<https://blog.langchain.dev/code-interpreter-api/>)
- こちらはインターネットからデータを取得できる

‘2023年の昨日までのドル円のチャートをプロットしてください。’とお願いしたらプロットしてくれた



# 補足:CodeInterpreter-api

追加で、'ローソク足で'とお願ひした場合の出力



# 補足: CodeInterpreter-api

irisデータファイルを与え、'データを分析して、それについて興味深いプロットをして'と依頼

