Matplotlib による
グラフ描画（データ分析）

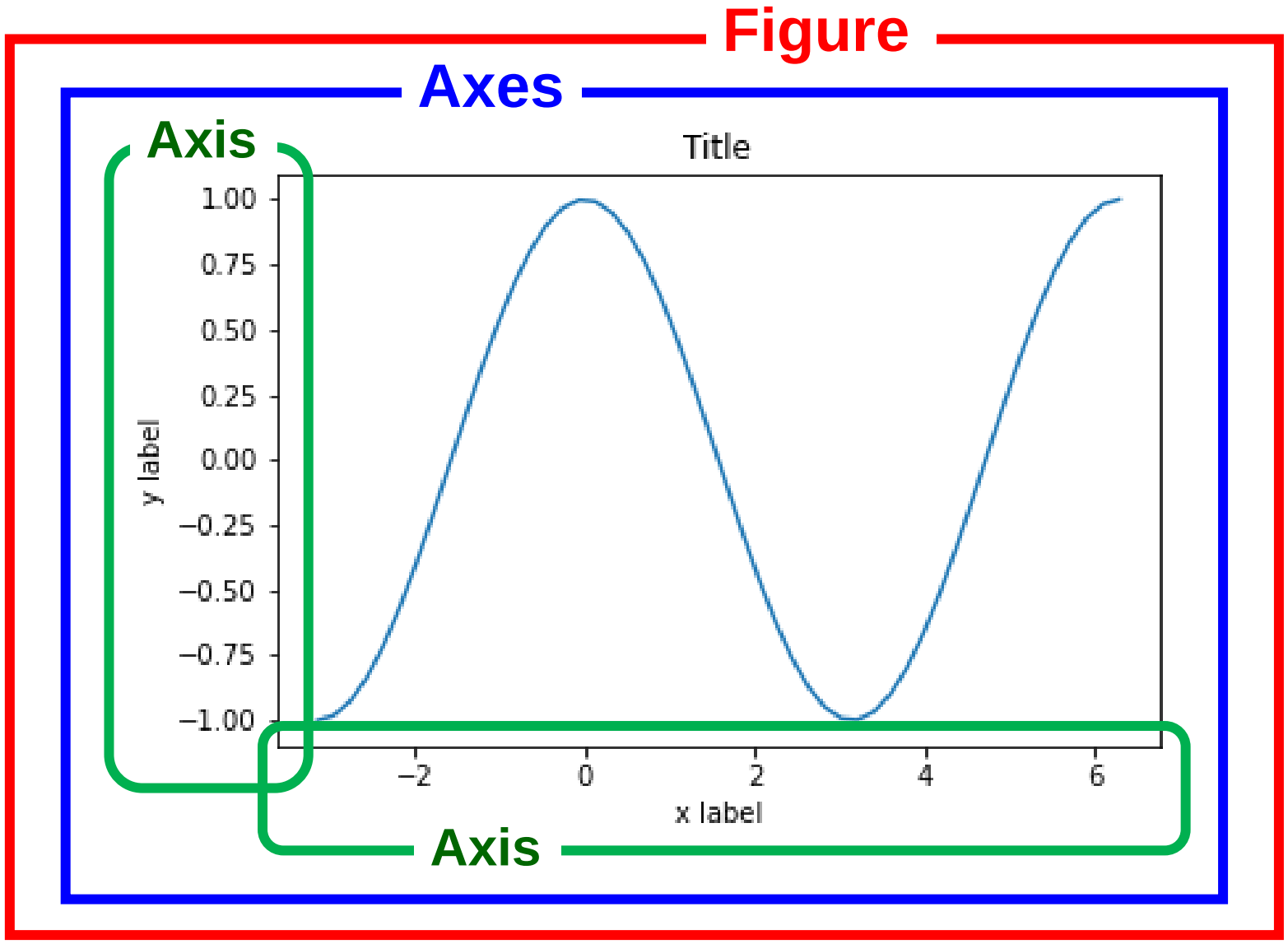
2023年度2Q 5c/6c（IL2） 木曜日

担当: 地引

TA: 増井

はじめに

グラフの描画に必要なデータ構造



1枚のグラフには、例えば左図のような描画データが入れ子状に含まれています(これらは一部)。

Figure: 台紙

Axes: グラフ全体

Axis: 座標軸

matplotlib は、これらの設定を簡潔に行える Python 言語用のライブラリです。

Python はﾌﾟﾛｸﾞﾗﾐﾝｸﾞ言語であり、matplotlib では様々な表記によりグラフを描画できるため、混乱しないようにしましょう。

今回は、np, plt の二つのライブラリだけを使う流儀に統一します。

Matplotlib の基本的な使い方

Matplotlib の基本的な使い方（1）

まずは、先頭に次の 2 行を書きます。

```
import numpy as np
import matplotlib.pyplot as plt
```

Matplotlib はプログラミング言語 Python 用のライブラリなので、前スライドで例示したようにグラフの部品毎にデータを設定する(例えば、これは x 軸、これは凡例という感じに部品毎に変数を用意し、逐一これらに代入する)という表記もできますが(こちらの方が、プログラム的な感じと言えそうです)、多くの場合では、数値解析用アプリとして有名な MATLIB の操作に似せて、上記の np, plt にデータを設定するという表記が使われます。

上の 2 行は、numpy および matplotlib.pyplot ライブラリを np および plt という変数名で利用する、という意味の設定です。

Matplotlib の基本的な使い方（2）

matplotlib では、以下のコマンドで、(x, y) に指定した点の線グラフ データを作成します。

```
plt.plot(x, y)
```

(x, y) は、基本的に変数 np を使って設定します。
例えば、$[-\pi, 2\pi]$ の範囲で、$(x, \cos(x))$ の点 100 個の線グラフ（つまり、$y = \cos(x)$ のグラフ）を書きたい場合（データを作りたい場合）は、下記のように設定します。

```
x = np.linspace(-np.pi, 2*np.pi, 100)
y = np.cos(x)
```

x 座標の範囲指定において、描画する点の個数（上記の例では 100）を省略した場合は、50 個の点が描画されます。
また、linespace ではなく、linspace なので、注意しましょう。

Matplotlib の基本的な使い方（3）

データの解析という意味では、関数をそのまま描画するより、実験などで記録したデータのグラフを描画することの方が多いでしょう。例えば、以下のような形式で、データがファイルに記録されている場合を考えます（以下の例では、このファイルの名前を sample.data とします）。

$x_0,\ y_0$
$x_1,\ y_1$
$x_2,\ y_2,$
…

このようなデータ集合のグラフを描く場合、np を用いた (x, y) の設定は次のよう行ないます。以下では、1 行で x 座標だけのデータ x_data と y 座標だけのデータ y_data を作成し、plt.plot で線グラフ データを作っています（sample.data のパス名に注意）。

```
x_data, y_data = np.loadtxt("sample.data", unpack=True)
plt.plot(x_data, y_data)
```

Matplotlib の基本的な使い方（4）

実際には、様々な実験を行ない記録した複数のデータを比較したいことも多いでしょう。
このような場合は、比較したいデータの種類だけ、

np による (x, y) の作成 + plt.plot(x, y) による描画データの作成

を行ないます。以下は、4 種類のデータを一つのグラフに重ねて描く例です。

```
x0_data, y0_data = np.loadtxt("cnm_etime-size.data", unpack=True)
plt.plot(x0_data, y0_data)
x1_data, y1_data = np.loadtxt("he1_etime-size.data", unpack=True)
plt.plot(x1_data, y1_data)
x2_data, y2_data = np.loadtxt("he2_etime-size.data", unpack=True)
plt.plot(x2_data, y2_data)
x3_data, y3_data = np.loadtxt("./hn_etime-size.data", unpack=True)
plt.plot(x3_data, y3_data)
```

パス名の意味に注意

Matplotlib の基本的な使い方（5）

データの種類や解析の状況に応じて、線グラフではなく棒グラフを描画したい場合もあります。
このような場合は、下記のように plot ではなく bar を使います。但し、棒グラフを描画する場合は、x 座標の範囲やデータの数（棒の数）に応じて、width で棒の太さを指定する必要があります（棒が細過ぎると、何も表示されない白紙のグラフが作成されてしまうので要注意）。
事前に適切な太さを見積もれれば良いのですが、多くの場合は試行錯誤をすることになります。

```
plt.bar(x0_data, y0_data, width = 40000)
```

また、データ集合が複数あり、これらの棒グラフを一つのグラフに重ね描きしたい場合は、
前スライドと同様な表記で描画できます（plt.plot が plt.bar に変わるだけ）。
但し、棒グラフなので、描画する順番は重要です（つまり、plt.bar を呼ぶ順番が重要）。
例えば、先に小さな棒グラフを描いてから、その上に大きな棒グラフを重ね描きしてしまうと、小さい方が隠れてしまいます。

Matplotlib の基本的な使い方（6）

さらに、観測したデータが大量にある場合は、グラフではなく散布図を作成して、傾向を掴みたいこともあるでしょう。このような場合は、下記のように plot ではなく scatter を使います。
散布図の描画では、データの数（点の数）に応じて、s で点の大きさを指定する必要があります。
これも事前に適切な大きさを見積もれれば良いのですが、棒グラフと同様、多くの場合は試行錯誤をすることになります。

```
plt.scatter(x0_data, y0_data, s = 0.01)
```

この例では、s に 1 以下の小数を設定していますが、1 以上の値を設定することもできます。
s に小さい値を設定することで細かい点が描画されます。但し、共通教材に説明があるように、データ上は小さい値を設定するほど描画データは細かくなりますが、画面の表示能力といった外部環境も影響するため、現実的には 0.01 程度が下限のようです。

Matplotlib の基本的な使い方（7）

x 軸, y 軸に名称(ラベル)を入れたい場合は、次のコマンドを使います(“” 内が表示される)。

```
plt.xlabel("x-axis label")
plt.ylabel("y-axis label")
```

凡例を入れたい場合は、plt.plot 等でグラフ データを作成する際に label でデータの名称を指定し(これは、plt.plot 毎に行ないます)、最後に plt.legend コマンドを使います。

```
x0_data, y0_data = np.loadtxt("./cnm_etime-size.data", unpack=True)
plt.plot(x0_data, y0_data, label="CNM")
x1_data, y1_data = np.loadtxt("./he1_etime-size.data", unpack=True)
plt.plot(x1_data, y1_data, label="HE")
x2_data, y2_data = np.loadtxt("./hn_etime-size.data", unpack=True)
plt.plot(x2_data, y2_data, label="HN")
plt.legend()
```

bar や scatter も同じです。

Matplotlib の基本的な使い方（8）

以下に、グラフ データを作成 or 利用するために必要な
その他の基本コマンドを列挙しておきます。

- グラフにタイトルを付ける：　　plt.title("Graph Title")
- グラフ データをファイルに保存する：　plt.savefig("AAA.bbb")
  - . の後に続く bbb の部分で、保存するデータの形式を指定します。
  - bbb に pdf を指定すると（例えば、sample.pdf）PDF 形式で、png を指定すると（例えば、sample.png）画像データ形式で保存されます。
  - 保存先（AAA.bbb のパス名）の指定は、慎重に確認しましょう。
  - plt.savefig は、次の plt.show より先に実行する必要があります。⇒ 要注意
- グラフ データを画面に表示する：　　plt.show()
- 対数軸にする（軸毎に設定する）：　　plt.xscale("log") および plt.yscale("log")

Matplotlib の基本的な使い方（9）

冒頭で簡単に説明しましたが、グラフ データ Axes は台紙データ Figure の中に作成されます。この時、何らかの原因で Axes の描画範囲が Figure の範囲を超えてしまうことがあります。このような場合は、グラフの一部が途切れてしまいます。
本来ならば、Axes や Figure のパラメータを個々に修正するのですが、Matplotlib にはお手軽に修正する方法も用意されています。下記のどちらかを実行することで簡易修正ができます。

```
plt.tight_layout()
```

```
plt.savefig("AAA.bbb", bbox_inches='tight')
```

前者は、savefig() コマンドの前に実行します。また後者は、savefig() コマンドのオプションという扱いです。作成したグラフを確認し、状況に応じて、試してみて下さい。

最後に

チャレンジしたい方を止めるわけではありません。

- 今回は以下の理由より、Matplotlib が自動的に作成するグラフの色やデザイン、目盛りなどを修正せずに、そのまま使うこととします。

  - Matplotlib は Python 言語用のライブラリなので、これまで説明して来た Matplotlib のコマンドとは、実は Python のプログラム コードです。

  - 冒頭で説明したように、グラフを構成する各データ（x 軸とか凡例とか）を直接設定する（もう少し別の言い方をすれば、細かい修正をする）Python コードを作ることはできますが、テーマ1「データの処理と加工」は、Python のプログラム演習ではないので、今回の説明では numpy（np）と matplotlib.pyplot（plt）の二つのライブラリを簡便に使う範囲に止めました。

  - 更に細かい修正をしたい人は、インターネットを検索してみて下さい。但し、上の理由から様々な表記ができるため、np/plt だけを使う流儀をお勧めします。

Matplotlib を用いたデータ分析（1）

演習１

- 補足教材「Google Colaboratory で Matplotlib を動かす準備」に記載のある“データ ファイルの扱い” 以降のスライドを参考に、 Google Colaboratory へ解析用データ（theme1.zip）をアップロードし、Colab 内で解凍（展開）する。
  - ブラウザの種類や設定によっては、zip ファイルをダウンロードした時点で展開してしまうものもあるので、このような場合は、上記スライドにある “注意!!” をよく読んで、対応して下さい。

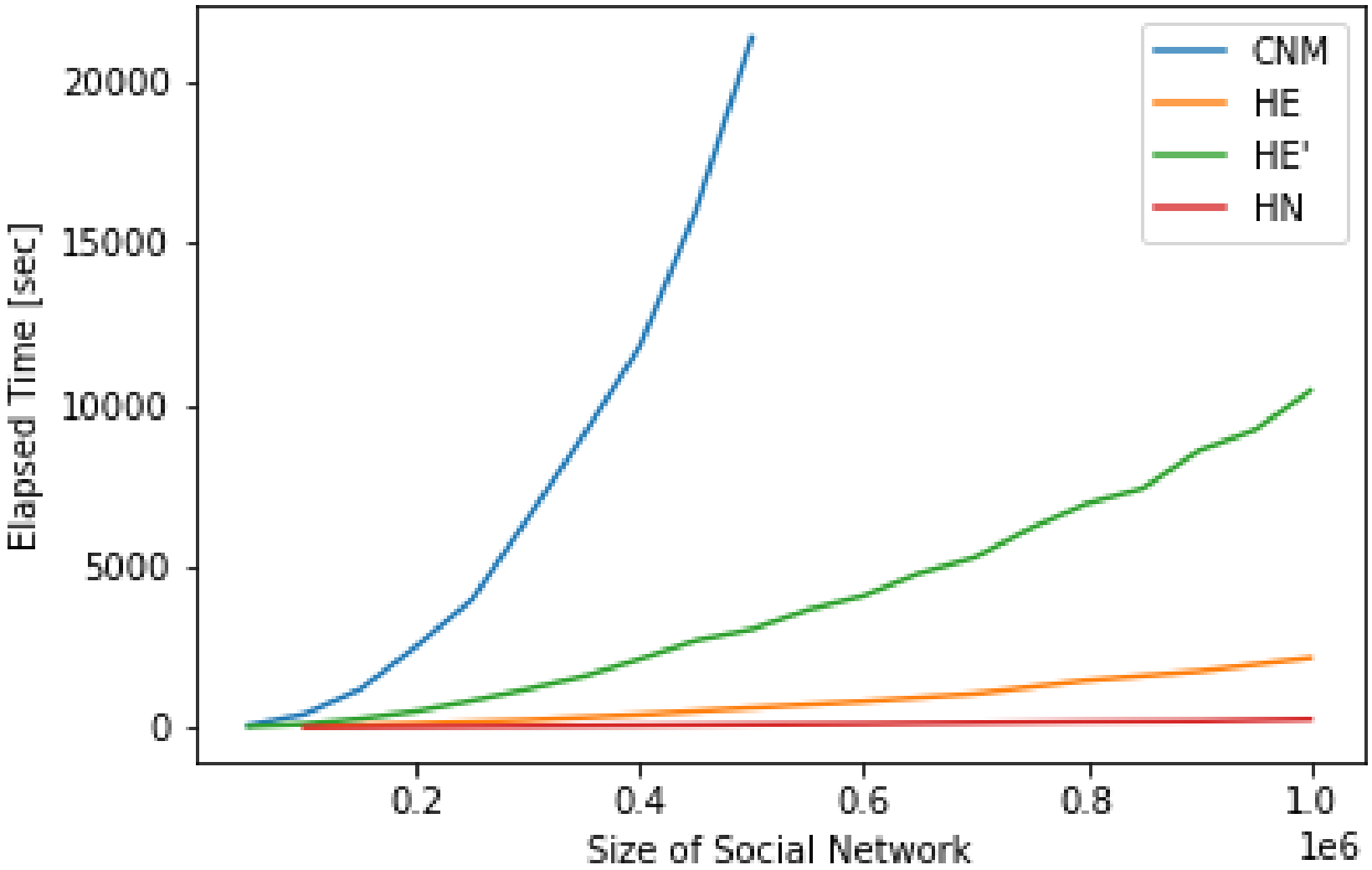
- まずは、次スライドにある図 5 用の Matplotlib コードを Colab のセルに入力し、図 5 のグラフ ファイルを作成する。
  - 最初なので、コピペでもよいですよ。→ とは言え、Colab はひっそりと固まることもあるので、戸惑うかも。

- 作成したグラフ ファイルをダウンロードし、保存しておく。
  - 以降の演習 2, 3 で作成したグラフは、成績に関わる課題として提出を求めるので、その練習です。
  - 課題についての詳細は、5c/6cページの “2. 課題” を参照。

図5のグラフを作成するための Matplotlib コード

```
import numpy as np
import matplotlib.pyplot as plt

# 各データ ファイルのパス名に注意
x0_data, y0_data = np.loadtxt("theme1/data/cnm/etime-size.data", unpack=True)
plt.plot(x0_data, y0_data, label="CNM")
x1_data, y1_data = np.loadtxt("theme1/data/he1/etime-size.data", unpack=True)
plt.plot(x1_data, y1_data, label="HE")
x2_data, y2_data = np.loadtxt("theme1/data/he2/etime-size.data", unpack=True)
plt.plot(x2_data, y2_data, label="HE'")
x3_data, y3_data = np.loadtxt("theme1/data/hn/etime-size.data", unpack=True)
plt.plot(x3_data, y3_data, label="HN")

plt.legend()
plt.xlabel("Size of Social Network")
plt.ylabel("Elapsed Time [sec]")

# 出力ファイルのパス名に注意
plt.savefig("theme1/images/fig5-etime.pdf")
plt.show()
```

Matplotlib で作成した図5のグラフ



演習2

- 図 5 のグラフを作成するための Matplotlib コードを参考に、
  図 6 のグラフ（fig6-etime.pdf）を作成する。
  - 新たなセルを用意し、図 5 用のコードをそこにコピペして修正することを
    お勧めします。
  - 図 6 の作成に必要なデータは、下記の共通教材で説明されています。

    “情報リテラシ第二” → “テーマ1 データの処理と加工” →
    “実習の内容：実習に用いるデータセット”

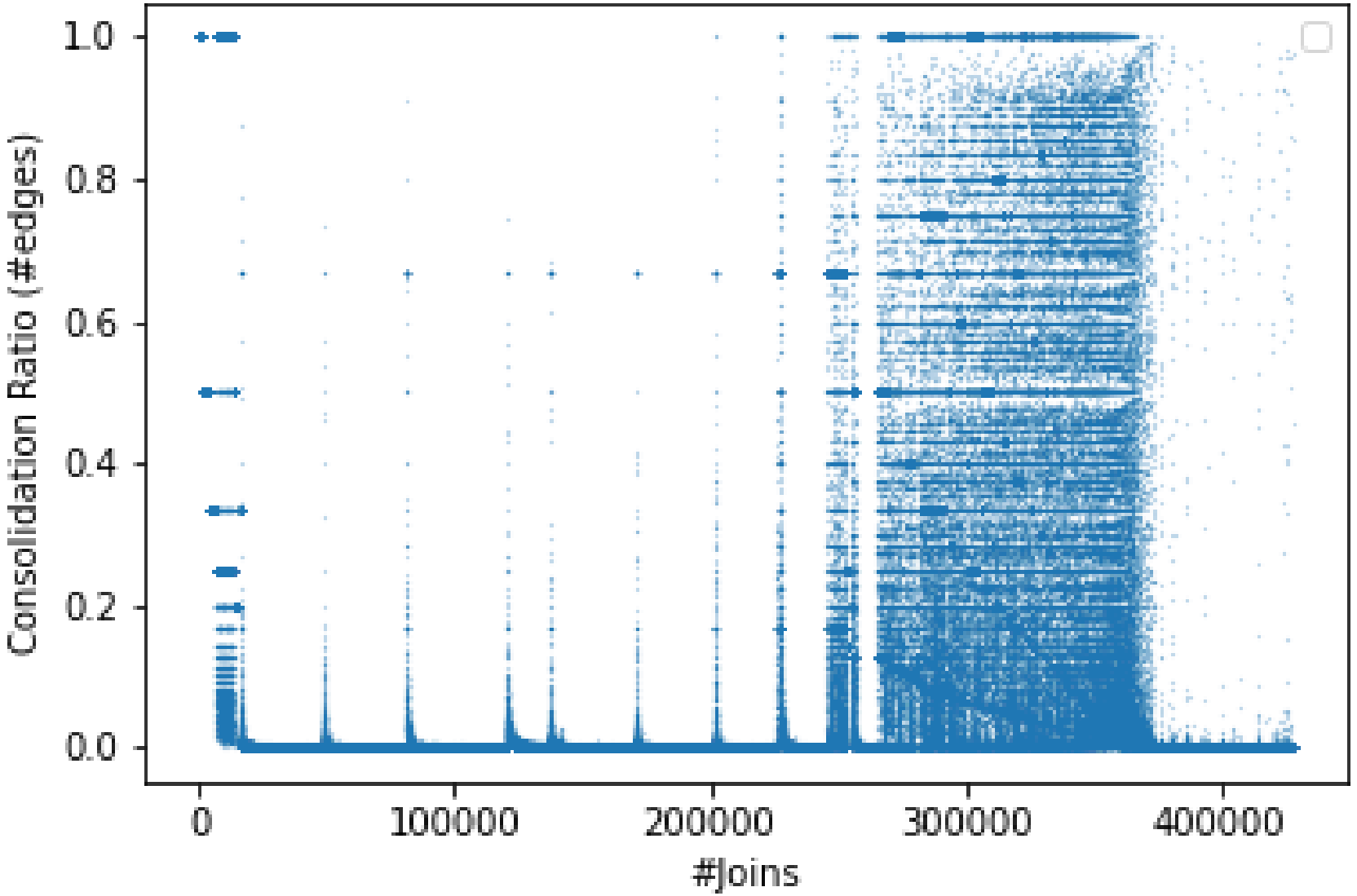
- 作成したグラフ ファイル（fig6-etime.pdf）をダウンロードし、
  保存しておく（課題として提出，詳細は 5c/6c ページの “2. 課題” を参照）。

Matplotlib を用いたデータ分析（2）

図3のグラフを作成するための Matplotlib コード（問題あり）

```
import numpy as np
import matplotlib.pyplot as plt

plt.xlabel("#Joins")
plt.ylabel("Consolidation Ratio (#edges)")

# 各データ ファイルのパス名に注意
x0_data, y0_data = np.loadtxt("theme1/data/cnm/ratio-join.data", unpack=True)
plt.scatter(x0_data, y0_data, s=0.01)

plt.legend()

# 出力ファイルのパス名に注意
plt.savefig("theme1/images/fig3-cnm-ratio-join.pdf")
plt.show()
```

Matplotlib で作成した図3のグラフ



この解析の趣旨

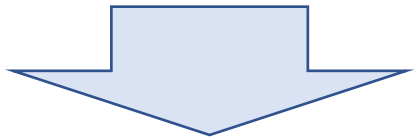
- 詳細は、次のテーマであるオンライン文書で取り上げる論文に記載されていますが、この解析では、ハブ空間（クラスタ）を合併して行く際に長時間を要する理由を調べています。

- そこで、合併回数（横軸）と、合併するクラスタ同士のサイズ比（縦軸）とをグラフ化して、何らかの傾向を掴もうとしました。

- しかしながら、特定の合併回数では合併するクラスタ同士のサイズがほぼ同じである（y 軸の値が 1 に近い）、大半の合併ではサイズが大きく異なる（同、0 に近い）、と見えるだけで、それ以上の（より詳しい）状況は分かりません。

データの特徴を掴むには

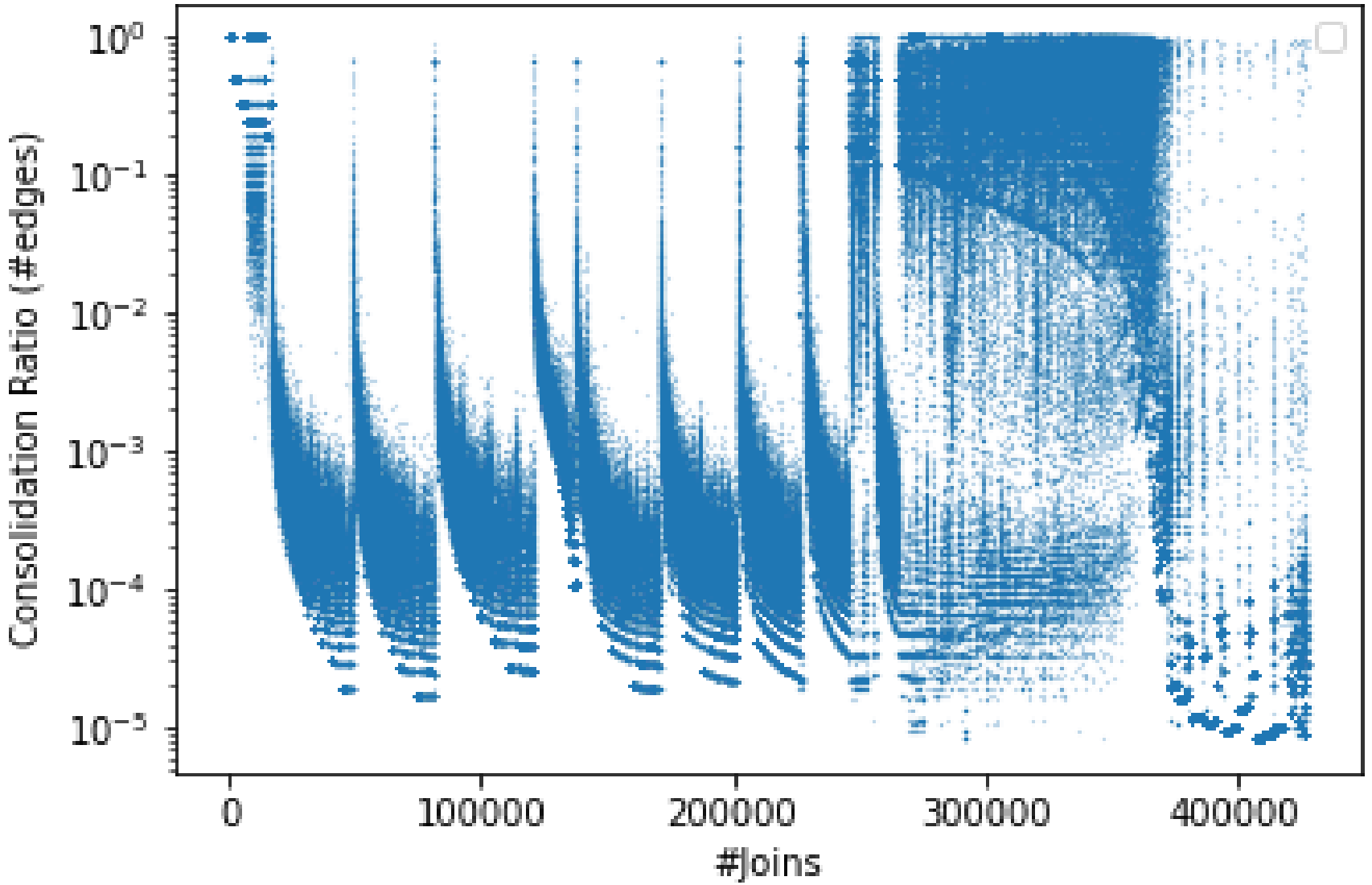
- データ cnm/ration-join.data は、そのまま表示しても特徴をよく掴めない。
  特徴をうまく掴めるような工夫が欲しい。
  - データの分布が、極端に偏っているように見える。
  - 相互関係では、指数法則の影響を受ける場合が多い
    （強い存在が益々/指数的に強くなる：スケール フリー）。



- x, y 軸を対数表示に変えてみる
  （元に戻す場合は、先頭に # を付けてコメントアウトすればよい）。

```
plt.xscale("log")      # x 軸が対数スケールに変わる。
plt.yscale("log")      # y 軸が対数スケールに変わる。
```

y 軸を対数スケールに変えた図3



このグラフを見ると、
合併回数（x軸）に応じて
サイズ比（y軸）も次第に
変化して行く（小さくなって
行く）様子が分かります。

グラフの出力形式について

- pdf 形式：図形を数式で表現
  - 一種のプログラムと言える。
  - 曲線は滑らかに表示される。
  - 拡大/縮小しても、画質は影響されない。
  - 描画するデータが増えると、処理が遅くなる（pdf ファイルのサイズも大きくなる）。

用途やデータ サイズに応じて、適切な出力形式を選択しよう。

- ビット マップ形式：図形を点の集まりとして表現
  - 点（画素）の細かさには限界があるため、曲線の描画や拡大/縮小時に画質が落ちる（掠れたように見える）。

- Matplotlib で出力形式を変える ⇒ 保存するファイルの識別子を変える

Matplotlib で作成した図2のグラフ

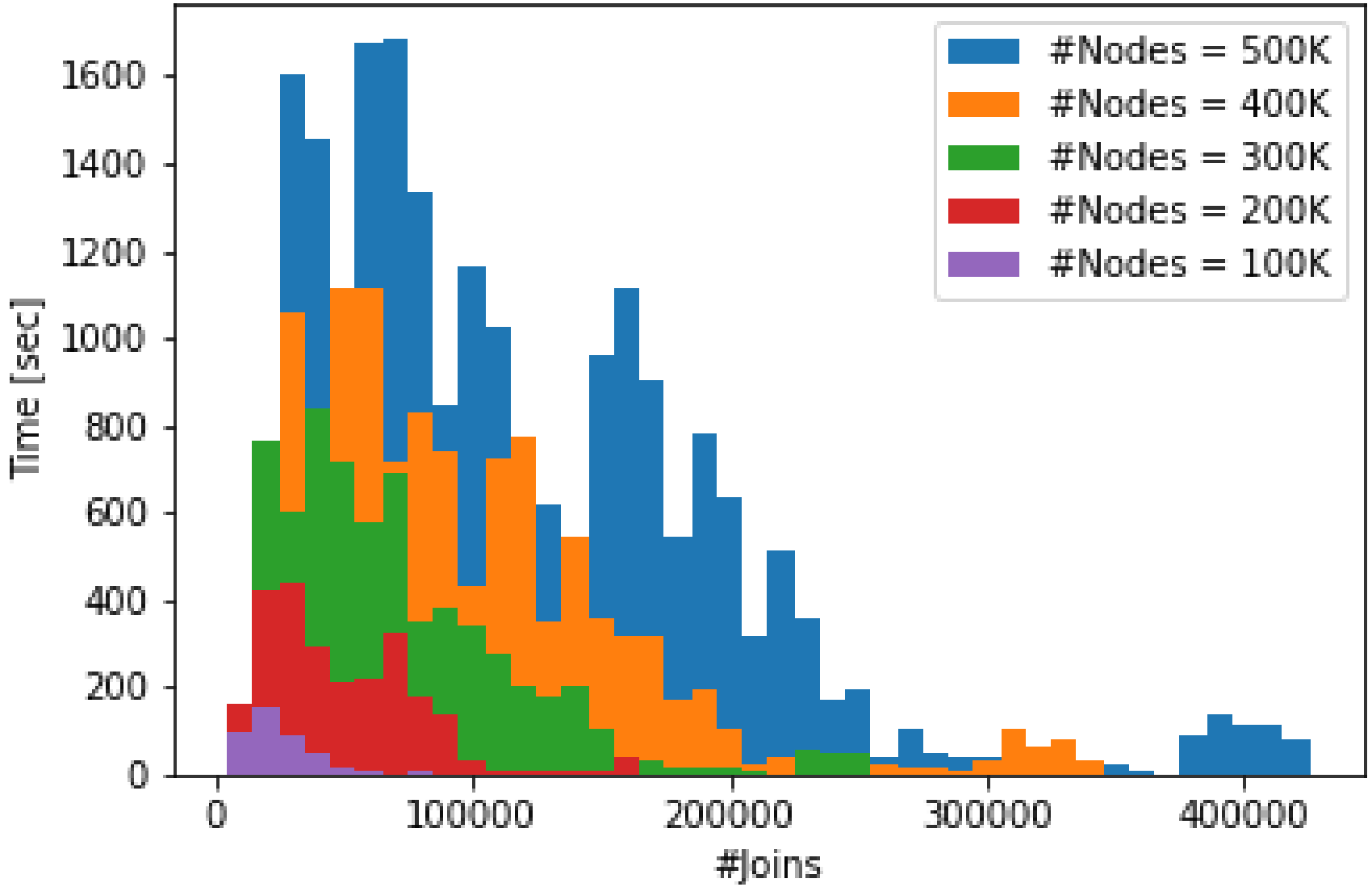


図2のグラフを作成するための Matplotlib コード（未完成）

```
import numpy as np
import matplotlib.pyplot as plt

plt.xlabel("#Joins")
plt.ylabel("Time [sec]")

# 各データ ファイルのパス名に注意
x0_data, y0_data = np.loadtxt("theme1/data/cnm/time-join-500K.data", unpack=True)
plt.bar(x0_data, y0_data, label="#Nodes = 500K", width=10000)

# この部分が未完成

plt.legend()

# 出力ファイルのパス名に注意
plt.savefig("theme1/images/fig2-cnm-joins-time-series.pdf")
plt.show()
```

図 2 のグラフは、線グラフではなく棒グラフです。
Matplotlib で棒グラフを作成する関数は、plt.bar() です。
線グラフ用の plt.plot() とほぼ同じですが、棒グラフの場合は適切な棒の幅を “width” で指定する必要があります。

“width” 適切な値を指定しない場合は、棒グラフが表示されません。
どんな値が適切かは、データの値域を見れば判断できますが、
まずは適当な値で試してみるという試行錯誤も、そう悪くありません。
また、棒グラフの場合は、作成する順番にも注意が必要です（隠れる）。

演習3

- 教材内の Matplotlib コードを参考に、図 3, 7, 2 のグラフを作成する。
  - グラフの出力形式は、データ数などに応じて適切なものを選択しましょう。
  - 作成するグラフ ファイル名は、出力形式（下記など）に合わせましょう。
    図2: fig2-cnm.pdf, 図3: fig3-cnm.png, 図7: fig7-hn.png
  - 各図の作成に必要なデータは、下記の共通教材で説明されています。
    “情報リテラシ第二” → “テーマ1 データの処理と加工” →
    “実習の内容: 実習に用いるデータセット”

- 作成した各グラフ ファイルをダウンロードし、保存しておく
  （各グラフは課題として提出, 詳細は 5c/6c ページの “2. 課題” を参照）。