

2018年度「先進ゲノム支援」情報解析講習会（11月20日）

# matplotlib & seaborn による データの視覚化

○ <http://bit.ly/2JgLVKT>

孫 建強

農業・食品産業技術総合研究機構  
農業情報研究センター

# Outline



**Data visualization**



**Visualization libraries**



**Introduction**



**Basic charts**

# Outline



1 Data visualization



2 Visualization libraries



3 Introduction



4 Basic charts

# Biological data visualization

- 視覚化は抽象的なことを、見てわかるような形で提示すること
- データ視覚化の目的
  - データに隠された情報を発見するため
  - データに含まれる情報を説明するため



references

○ Visualizing biological data

<https://www.nature.com/articles/nmeth.2258>

# Outline



1 Data visualization



2 Visualization libraries



3 Introduction



4 Basic charts

# Visualization libraries

## ■ 静止画像

- matplotlib
- pandas
- seaborn
- ggplot

## ■ インタラクティブ

- plotly
- Bokeh

# Visualization libraries

## ■ 静止画像

■ matplotlib

■ pandas

■ seaborn

■ ggplot



- 初期から存在する最も古い視覚化ライブラリー
- 最も使われているため情報入手しやすい
- 複雑なグラフや細かい部分の調整が可能

## ■ インタラクティブ

■ plotly

■ Bokeh

# Visualization libraries

## ■ 静止画像

matplotlib

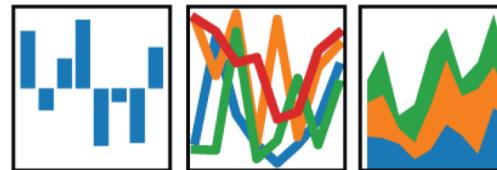
pandas

seaborn

ggplot

## pandas

$$y_i^t = \beta' x_{it} + \mu_i + \epsilon_{it}$$



- matplotlib をベースとしている

- pandas データ構造を簡易にグラフ化できる

- 複雑なグラフや細かい部分の調整が不可

## ■ インタラクティブ

plotly

Bokeh

# Visualization libraries

## ■ 静止画像

matplotlib

pandas

seaborn

ggplot

## seaborn

- matplotlib をベースとしている
- matplotlib を補完する位置付け
- 複雑なグラフも作成可能
- より少ないコーディングで、より美しいグラフを

## ■ インタラクティブ

plotly

Bokeh

# Visualization libraries

## ■ 静止画像

matplotlib

pandas

seaborn

ggplot

## ggplot

- R の ggplot2 をベースとしている
- The Grammar of Graphics
- R の ggplot2 とほぼ同様な仕上がり

## ■ インタラクティブ

plotly

Bokeh

# Visualization libraries

## ■ 静止画像

■ matplotlib

■ pandas

■ seaborn

■ ggplot

## ■ インタラクティブ

■ plotly

■ Bokeh



- ウェブベースのインタラクティブなグラフ
- グラフ化後のグラフに対する調整が可能
- matplotlib に比べ勉強しやすい

# Visualization libraries

## ■ 静止画像

■ matplotlib

■ pandas

■ seaborn

■ ggplot



**Bokeh**

○ ウェブベースのインタラクティブなグラフ

○ The Grammar of Graphics 書き方

## ■ インタラクティブ

■ plotly

■ Bokeh

# Outline



**Data visualization**



**Visualization libraries**



**Introduction**



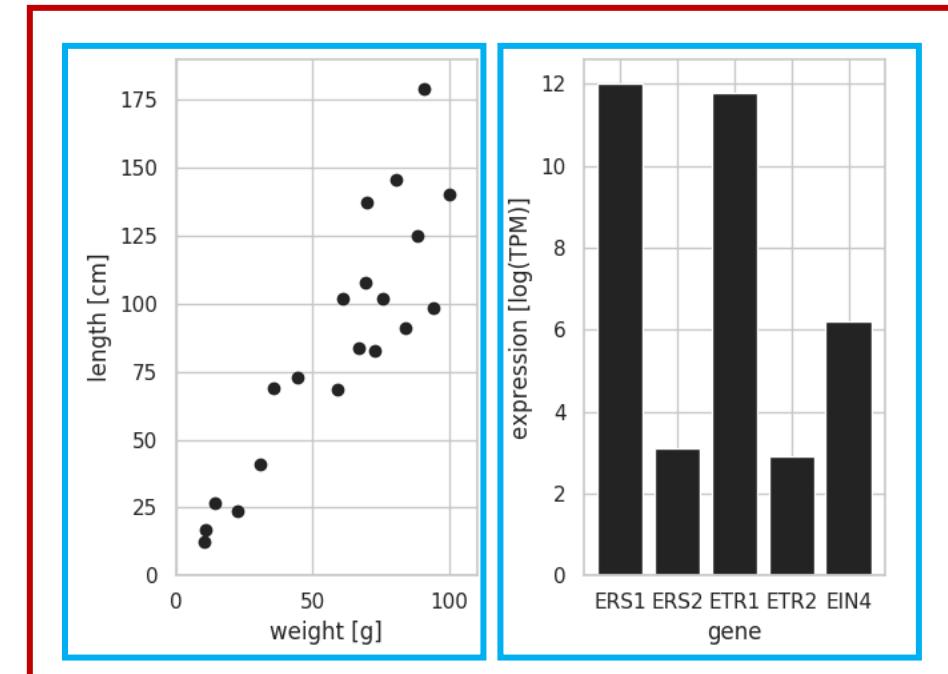
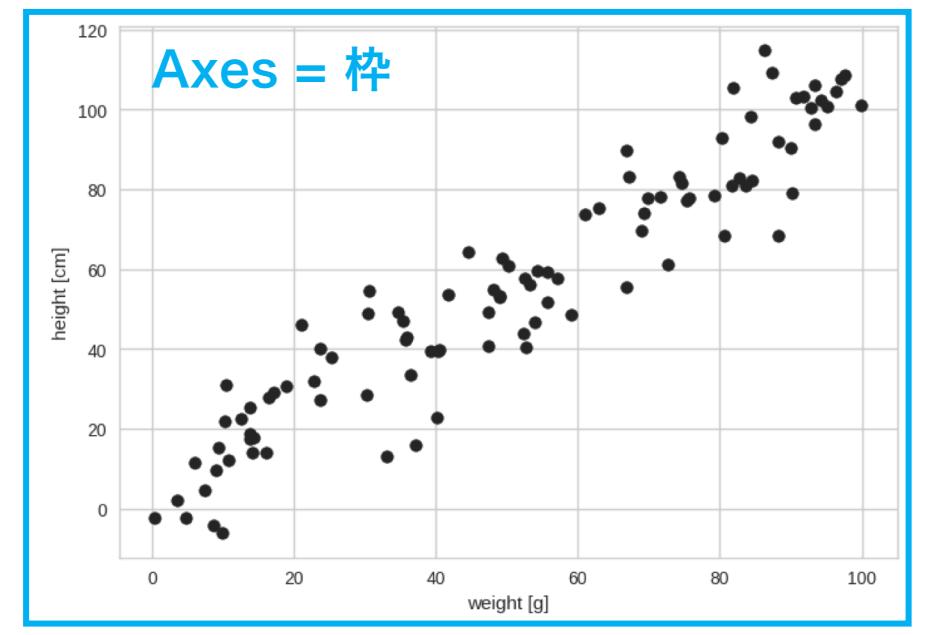
**Basic charts**

# **matplotlib**

# class structure

Pyplot = 落書き帳

Figure = ページ



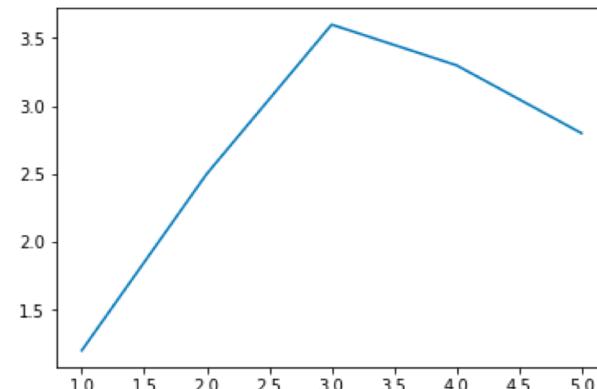
# matplotlib

matplotlib を利用したグラフ作成は、  
pyplot クラスのメソッドを使用する。

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

x = np.array([1.0, 2.0, 3.0, 4.0, 5.0])
y = np.array([1.2, 2.5, 3.4, 3.3, 2.8])

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(x, y)
fig.show()
```



# matplotlib

matplotlib を利用したグラフ作成は、  
pyplot クラスのメソッドを使用する。

matplotlib の pyplot クラスを呼び出し  
て使えるようにする。

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

x = np.array([1.0, 2.0, 3.0, 4.0, 5.0])
y = np.array([1.2, 2.5, 3.4, 3.3, 2.8])

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(x, y)
fig.show()
```

# matplotlib

matplotlib を利用したグラフ作成は、  
pyplot クラスのメソッドを使用する。

Figure クラスのオブジェクトを作成する。

```
import numpy as np  
from matplotlib import pyplot as plt  
  
x = np.array([1.0, 2.0, 3.0, 4.0, 5.0])  
y = np.array([1.2, 2.5, 3.4, 3.3, 2.8])  
  
fig = plt.figure()  
ax = fig.add_subplot(1, 1, 1)  
ax.plot(x, y)  
fig.show()
```

# matplotlib

matplotlib を利用したグラフ作成は、  
pyplot クラスのメソッドを使用する。

Figure 領域を1行1列に分割し、分割領域の1番目の領域でAxesクラスのオブジェクトを作成する。

```
import numpy as np  
from matplotlib import pyplot as plt  
  
x = np.array([1.0, 2.0, 3.0, 4.0, 5.0])  
y = np.array([1.2, 2.5, 3.4, 3.3, 2.8])  
  
fig = plt.figure()  
ax = fig.add_subplot(1, 1, 1)  
ax.plot(x, y)  
fig.show()
```



# matplotlib

matplotlib を利用したグラフ作成は、  
pyplot クラスのメソッドを使用する。

1番目のAxesオブジェクトに線グラフを  
描く。

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

x = np.array([1.0, 2.0, 3.0, 4.0, 5.0])
y = np.array([1.2, 2.5, 3.4, 3.3, 2.8])

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(x, y)
fig.show()
```



# matplotlib

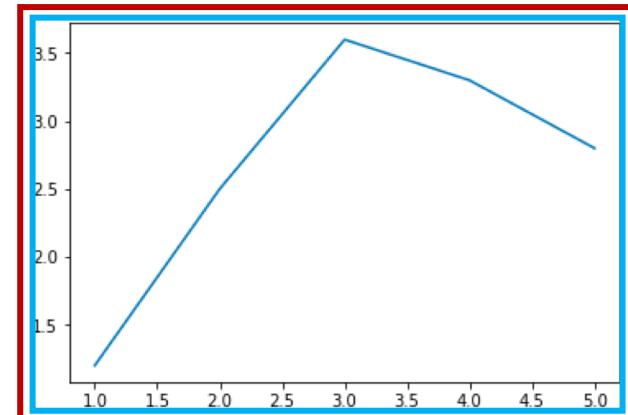
matplotlib を利用したグラフ作成は、  
pyplot クラスのメソッドを使用する。

これまでに描いたグラフをディスプレイ上  
に表示させる。

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

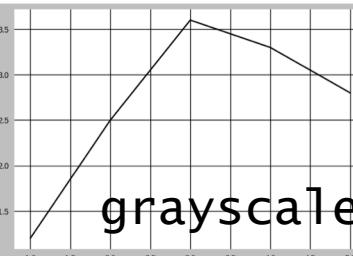
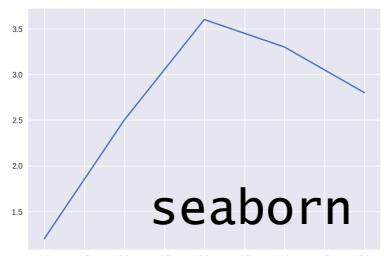
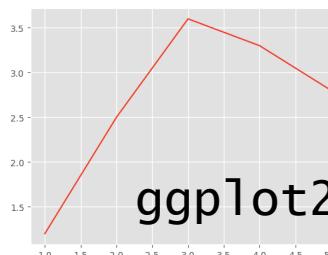
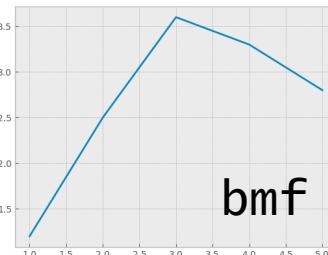
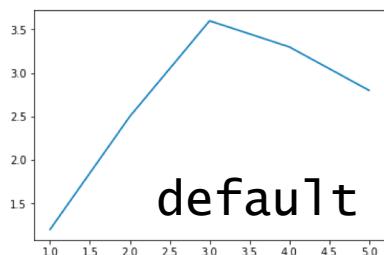
x = np.array([1.0, 2.0, 3.0, 4.0, 5.0])
y = np.array([1.2, 2.5, 3.4, 3.3, 2.8])

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(x, y)
fig.show()
```



# matplotlib – style sheets

グラフのスタイルを設定する。



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

```
plt.style.use('bmh')
```

```
x = np.array([1.0, 2.0, 3.0, 4.0, 5.0])  
y = np.array([1.2, 2.5, 3.4, 3.3, 2.8])
```

```
fig = plt.figure()  
ax = fig.add_subplot(1, 1, 1)  
ax.plot(x, y)  
fig.show()
```

matplotlib を使って作成したグラフは、デフォルトでは、背景が白で、点や線が青色となる。これは、matplotlib のデフォルトでは、そうなるように設定されているからである。matplotlib ではデフォルト以外に bmf、ggplot2 などのスタイルシートも設定されている。

# seaborn

# seaborn

seaborn を利用したグラフ作成は、matplotlib と同様な手順で作成できる。

## seaborn のインポート

seaborn クラスをインポートして、利用可能な状態にする。

## seaborn.set の実行

seaborn.set メソッドを実行して、グラフ描画パラメータなどを seaborn 風に書き換える。

## グラフの作成と表示

matplotlib.pyplot のメソッドをそのまま使用してグラフの作成と表示を行う。

```
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns

sns.set()

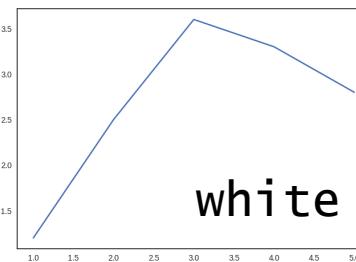
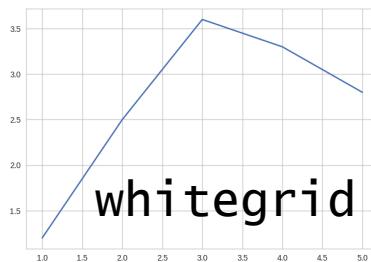
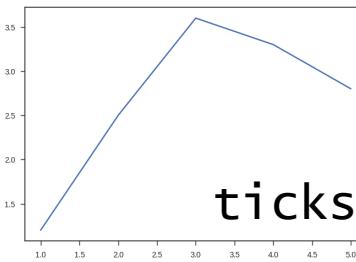
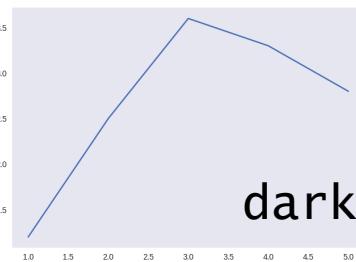
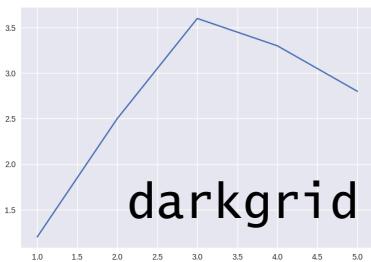
x = np.array([1.0, 2.0, 3.0, 4.0, 5.0])
y = np.array([1.2, 2.5, 3.4, 3.3, 2.8])

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(x, y)
fig.show()
```



# seaborn – style sheets

グラフのスタイルを設定する。



```
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns

sns.set()
sns.set_style('whitegrid')

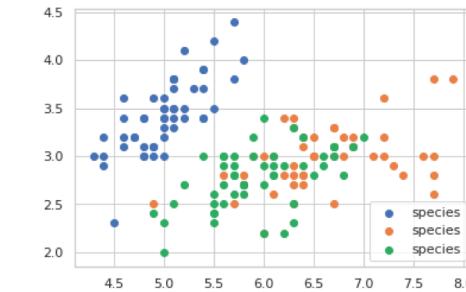
x = np.array([1.0, 2.0, 3.0, 4.0, 5.0])
y = np.array([1.2, 2.5, 3.4, 3.3, 2.8])

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(x, y)
fig.show()
```

seaborn ではスタイルシートが用意されている。スタイルシートを書き換えることで、グラフの描画エリアのスタイルを変更することができる。

# seaborn – color palettes

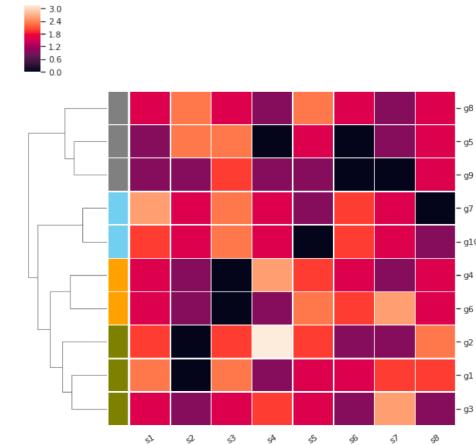
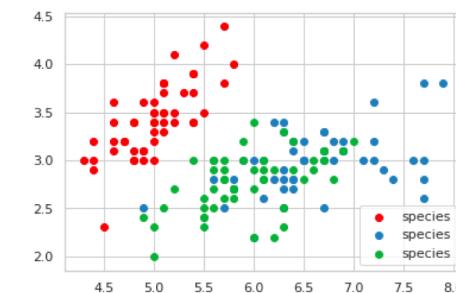
あらかじめ決められた色の組み合わせ。カラー  
パレットを書き換えることで、グラフ全体の配  
色パターンを簡単に変更できる。



default



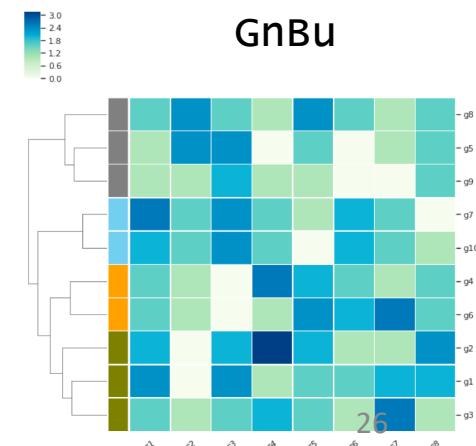
Set1



default



GnBu



# seaborn – color palettes

あらかじめ決められた色の組み合わせ。カラー パレットを書き換えることで、グラフ全体の配色パターンを簡単に変更できる。

seaborn のデフォルトのカラー パレットは `deep` となっている。`set_palette` メソッドを使うことで、カラー パレットの変更を行うことができる。



# seaborn – color palettes

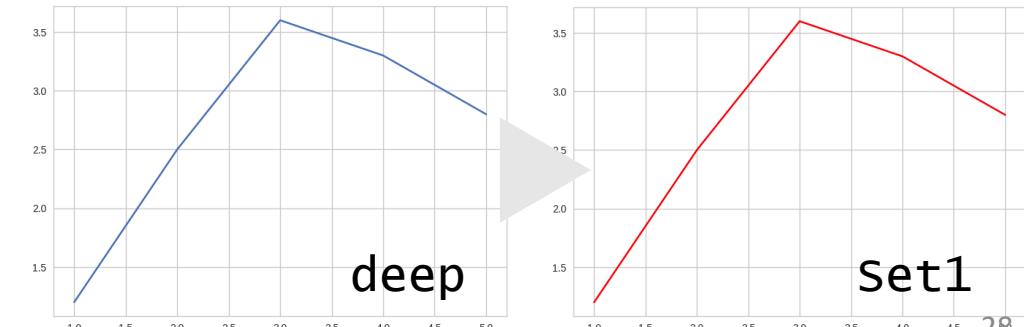
Set1 カラーパレットの色組を使う。

Set1



```
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
sns.set()
sns.set_style('whitegrid')
sns.set_palette('Set1')

x = np.array([1.0, 2.0, 3.0, 4.0, 5.0])
y = np.array([1.2, 2.5, 3.4, 3.3, 2.8])
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(x, y)
fig.show()
```



references

Seaborn style sheet  
<https://bit.ly/2Q3Tnvd>

# Outputs

# Outputs

グラフを画像ファイルへ書き出すとき、  
show メソッドの代わりに savefig メソッドを使用する。画像ファイルのフォーマットは format オプションで指定する。PNG の他に PDF、PS、EPS、SVG を指定できる。

```
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
sns.set()
sns.set_style('whitegrid')
sns.set_palette('Set1')

x = np.array([1.0, 2.0, 3.0, 4.0, 5.0])
y = np.array([1.2, 2.5, 3.4, 3.3, 2.8])

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(x, y)

fig.savefig('fig1.png', format='png')
```

# Outputs

グラフを画像ファイルへ書き出すとき、  
show メソッドの代わりに savefig メソッドを使用する。画像ファイルのフォーマットは format オプションで指定する。PNG の他に PDF、PS、EPS、SVG を指定できる。

グラフのフォントサイズなどを調整する場合は、該当するメソッドを利用する。

```
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
sns.set()
sns.set_style('whitegrid')
sns.set_palette('Set1')

x = np.array([1.0, 2.0, 3.0, 4.0, 5.0])
y = np.array([1.2, 2.5, 3.4, 3.3, 2.8])

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(x, y)

ax.set_xlabel(" xlabel", fontsize=18)
ax.set_ylabel(" ylabel", fontsize=18)
ax.tick_params(labelsize=18)

fig.savefig('fig1.png', format='png')
```

# Outputs

グラフを画像ファイルへ書き出すとき、  
show メソッドの代わりに savefig メソッドを使用する。画像ファイルのフォーマットは format オプションで指定する。PNG の他に PDF、PS、EPS、SVG を指定できる。

グラフのフォントサイズなどを調整する場合は、該当するメソッドを利用する。

画像ファイルのサイズや解像度などを設定したい場合は、描画デバイスを呼び出すときに、設定する。

```
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
sns.set()
sns.set_style('whitegrid')
sns.set_palette('Set1')

x = np.array([1.0, 2.0, 3.0, 4.0, 5.0])
y = np.array([1.2, 2.5, 3.4, 3.3, 2.8])

fig = plt.figure(figsize=[8, 6], dpi=300)
ax = fig.add_subplot(1, 1, 1)
ax.plot(x, y)

ax.set_xlabel(" xlabel", fontsize=18)
ax.set_ylabel(" ylabel", fontsize=18)
ax.tick_params(labelsize=18)

fig.savefig('fig1.png', format='png')
```

# Outline



1 Data visualization



2 Visualization libraries

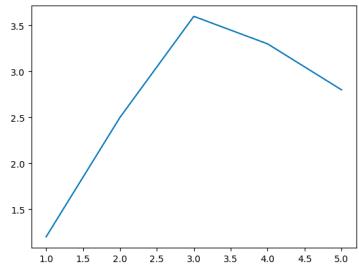


3 Overview



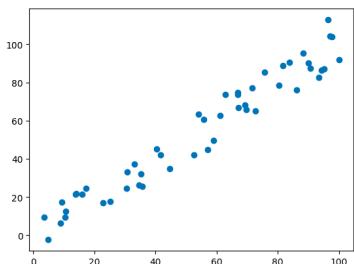
4 Basic charts

# Basic charts



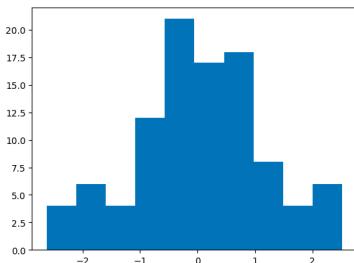
## line chart

データの時系列的な変化傾向を視覚する目的で使われる。



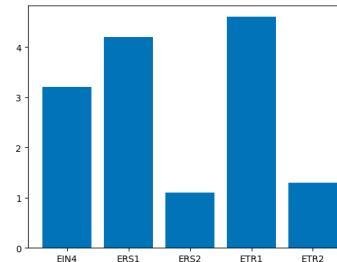
## scatter chart

2変量の連続値データ同士の相関や分布などを視覚する目的で使われる。



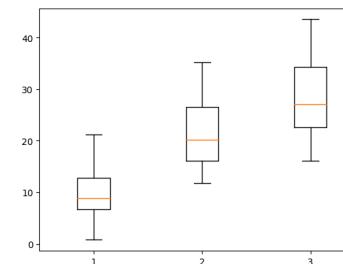
## histogram

1変量の連続値データの分布などを視覚化する目的で使われる。



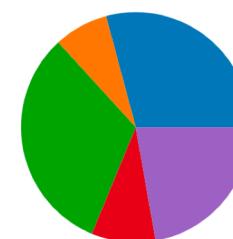
## bar chart

カテゴリ毎に分類されるデータを視覚化する目的で使われる。



## boxplot

カテゴリ毎に分類されるデータの分布などを視覚化する目的で使われる。



## pie chart

割合データを視覚する目的で使われる。

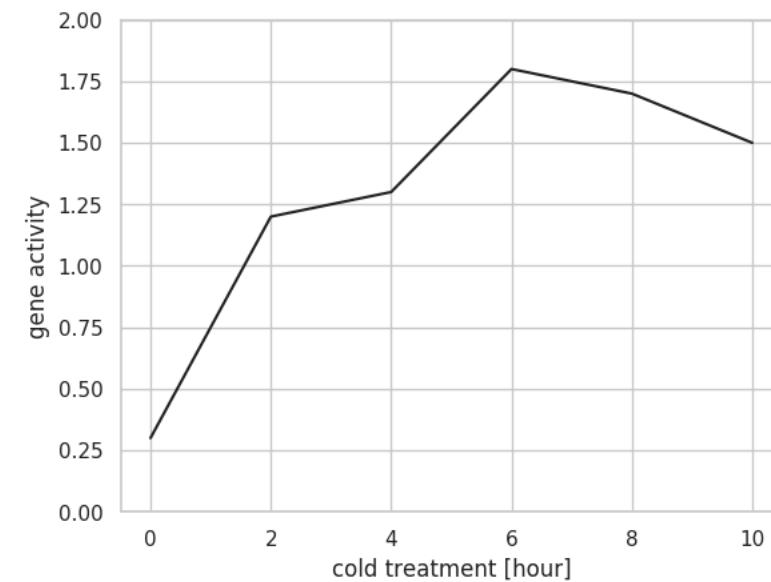
# line chart

# line chart

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set()

x = np.array([0, 2, 4, 6, 8, 10])
y = np.array([0.3, 1.2, 1.3, 1.8, 1.7, 1.5])

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(x, y)
ax.set_xlabel('cold treatment [hour]')
ax.set_ylabel('gene activity')
ax.set_ylim(0, 2)
fig.show()
```



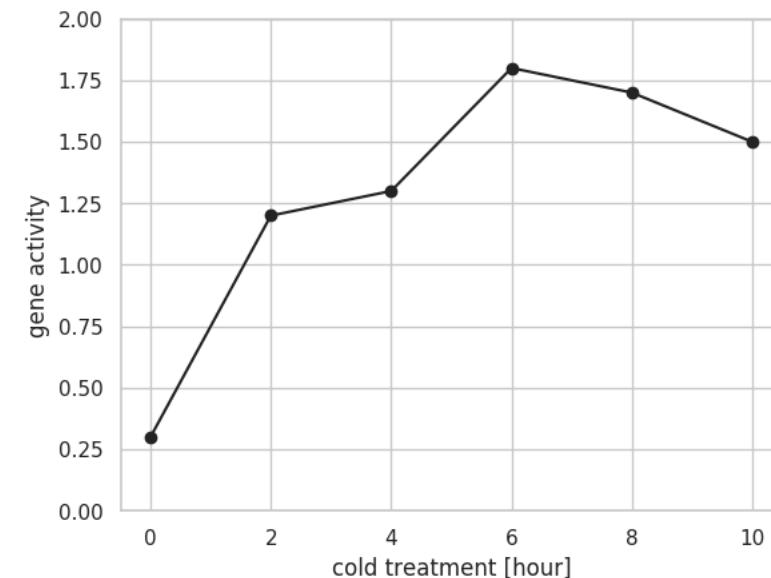
```
set_style='whitegrid'
set_palette='gray'
```

# line chart

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set()

x = np.array([0, 2, 4, 6, 8, 10])
y = np.array([0.3, 1.2, 1.3, 1.8, 1.7, 1.5])

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(x, y, marker='o')
ax.set_xlabel('cold treatment [hour]')
ax.set_ylabel('gene activity')
ax.set_ylim(0, 2)
fig.show()
```



set\_style='whitegrid'  
set\_palette='gray'



references

plot markers

<https://bit.ly/2z5KPOa>

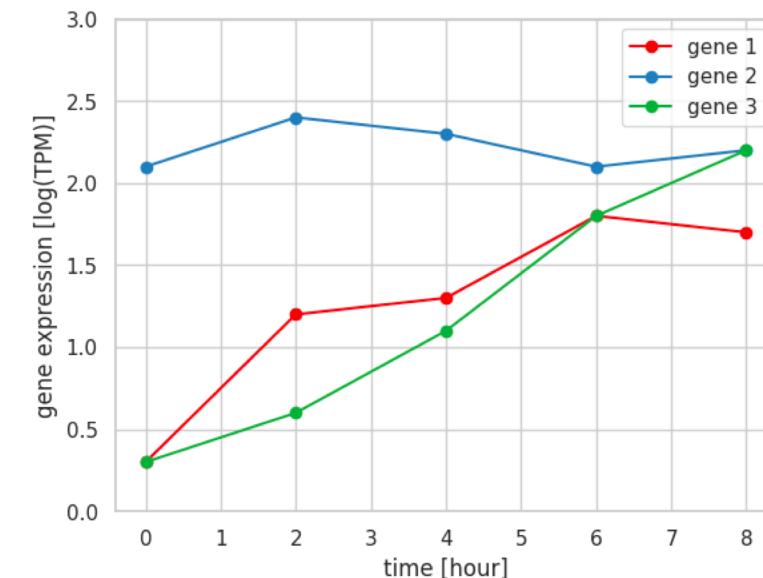
# line chart

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set()

x = np.array([0, 2, 4, 6, 8])
gene_1 = np.array([0.3, 1.2, 1.3, 1.8, 1.7])
gene_2 = np.array([2.1, 2.4, 2.3, 2.1, 2.2])
gene_3 = np.array([0.3, 0.6, 1.1, 1.8, 2.2])

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(x, gene_1, label='gene 1', marker='o')
ax.plot(x, gene_2, label='gene 2', marker='o')
ax.plot(x, gene_3, label='gene 3', marker='o')

ax.legend()
ax.set_title('Gene expression')
ax.set_xlabel('time [h]')
ax.set_ylabel('log10FPKM')
fig.show()
```



```
set_style='whitegrid'
set_palette='Set1'
```



# line chart

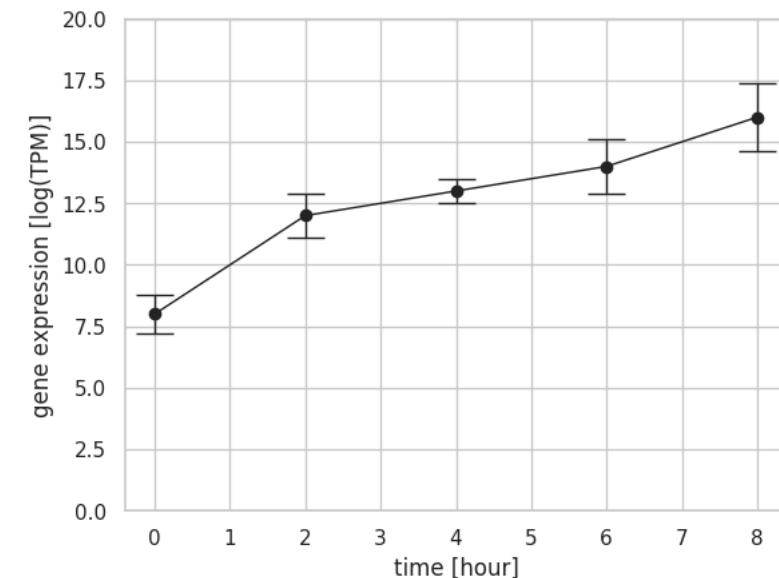
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set()

x = np.array([0, 2, 4, 6, 8])
y = np.array([8, 12, 13, 14, 16])
e = np.array([0.8, 0.9, 0.5, 1.1, 1.4])

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.errorbar(x, y, yerr=e, marker='o',
            capthick=1, capsized=10, lw=1)

ax.set_xlabel('time [hour]')
ax.set_ylabel('gene expression [log(TPM)]')
ax.set_ylim(0, 20)

fig.show()
```

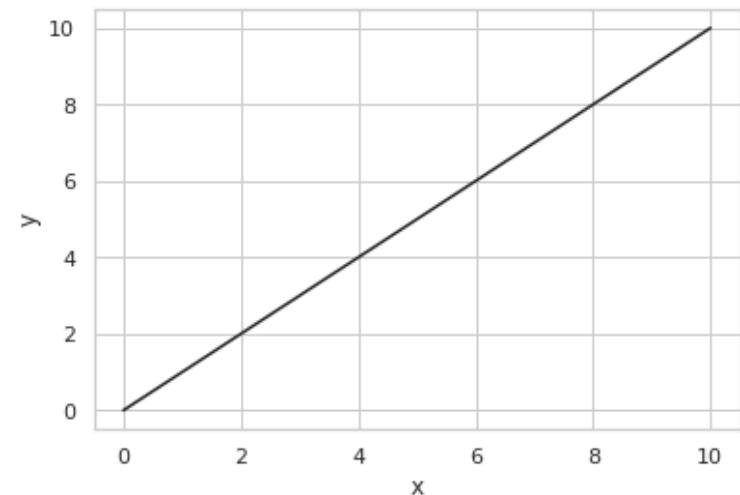


```
set_style='whitegrid'
set_palette='gray'
```

# 練習問題 1

直線  $y = x$  のグラフを描け。ただし、 $x$  軸の範囲を 0~10 とし、 $y$  軸の範囲を 0~10 とする。

## ○ 解答例



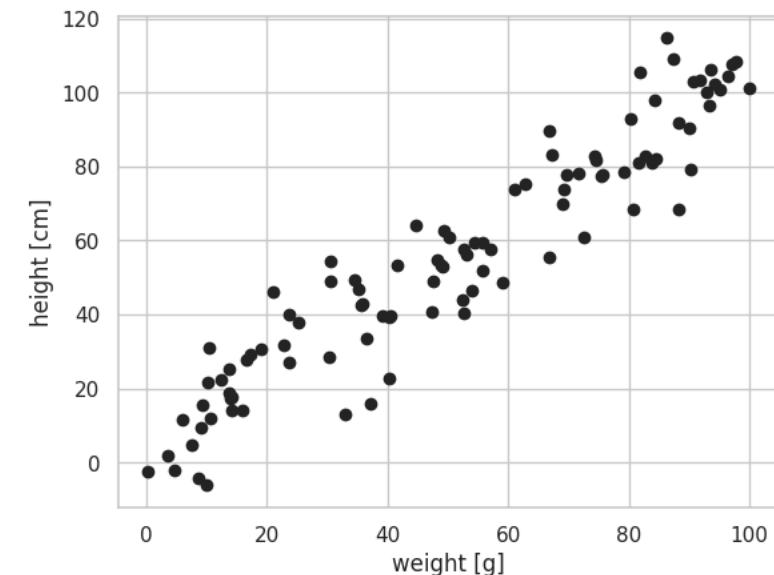
# scatter chart

# scatter chart

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set()
np.random.seed(2018)

x = np.random.uniform(0, 100, 100)
y = x + np.random.normal(5, 10, 100)

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.scatter(x, y)
fig.show()
```



```
set_style='whitegrid'
set_palette='gray'
```

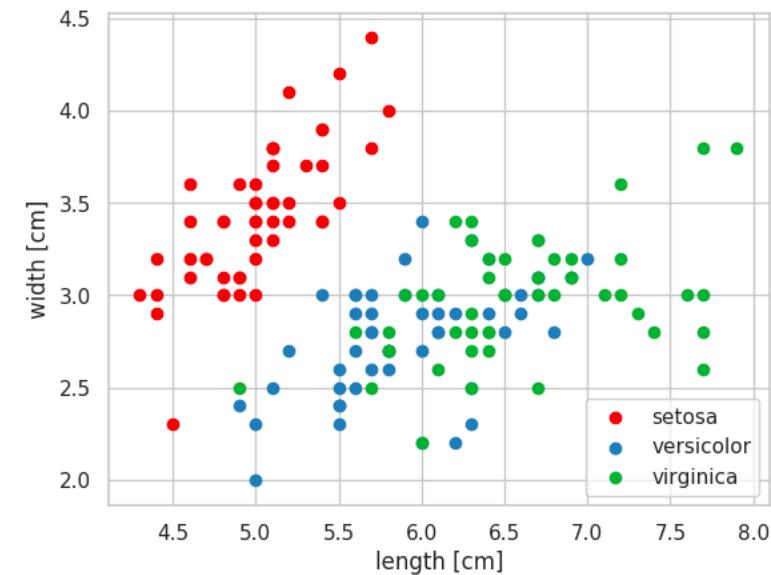
# scatter chart

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set()
df = sns.load_dataset('iris')

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

sp = 'setosa'
ax.scatter(data=df[df['species'] == sp],
           x='sepal_length', y='sepal_width', label=sp)
sp = 'versicolor'
ax.scatter(data=df[df['species'] == sp],
           x='sepal_length', y='sepal_width', label=sp)
sp = 'virginica'
ax.scatter(data=df[df['species'] == sp],
           x='sepal_length', y='sepal_width', label=sp)

ax.legend(loc='lower right')
fig.show()
```



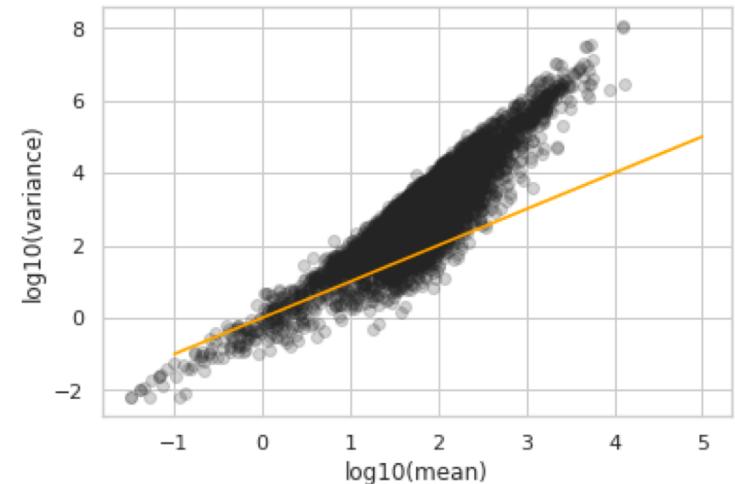
set\_style='whitegrid'  
set\_palette='Set1'

# 練習問題 2-1

遺伝子発現量TPM (count\_raw.tsv) を読み込み、各行について平均と分散を計算し、横軸を平均、縦軸を分散として散布図を描け。また、直線  $y = x$  もわかりやすく描き加えること。

ただし、カウントデータのレンジは大きいので、平均と分散を計算した後に、対数化してから散布図にすること。また、`plt.scatter`メソッドの引数として`alpha=0.2`を渡すと、描かれる点が半透明になる。

## ○ 解答例



# 練習問題 2-2

正規化前の遺伝子発現量 (count\_raw.tsv) を読み込み、batch\_1 と chemostat\_1 サンプルのデータを用いて、MA プロットを描け。

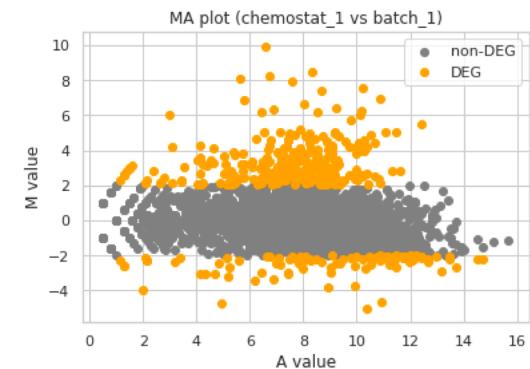
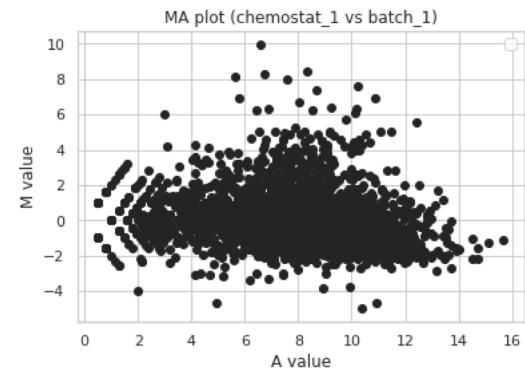
ただし、B 群 vs C 群の場合、M 値と A 値は次のように計算されるものとする。

$$M = \log_2(C + 1) - \log_2(B + 1)$$

$$A = \frac{\log_2(C + 1) + \log_2(B + 1)}{2}$$

余裕があれば、M 値の絶対値が 2 よりも大きい遺伝子を塗り分ける。

## ○ 解答例



# bar chart

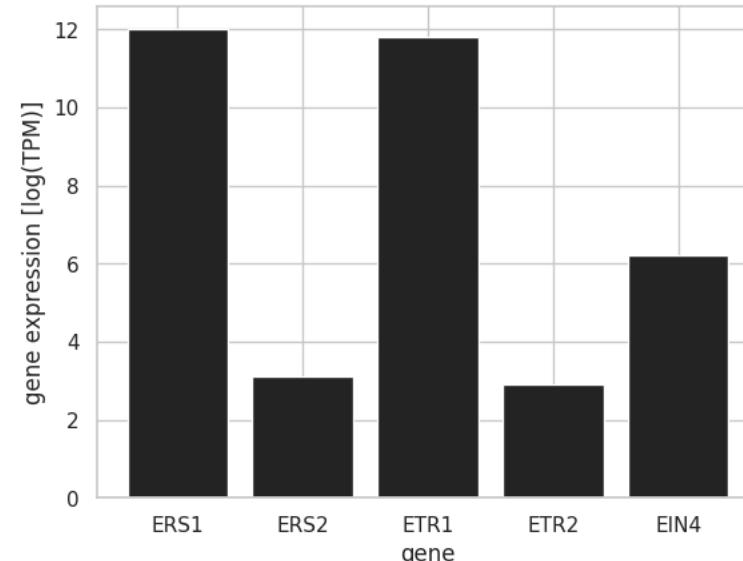
# bar chart

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set()

x = np.array(['ERS1', 'ERS2', 'ETR1',
              'ETR2', 'EIN4'])
y = np.array([12.0, 3.1, 11.8, 2.9, 6.2])
x_position = np.arange(len(x))

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.bar(x_position, y, tick_label=x)
fig.show()
```

`plt.bar(x, y)` でも棒グラフを作成できるが、`x` 軸が自動的にアルファベット順に並び直される。これを防ぐために、ここでは、`x` 軸を数値に変更した上でグラフを作成している。



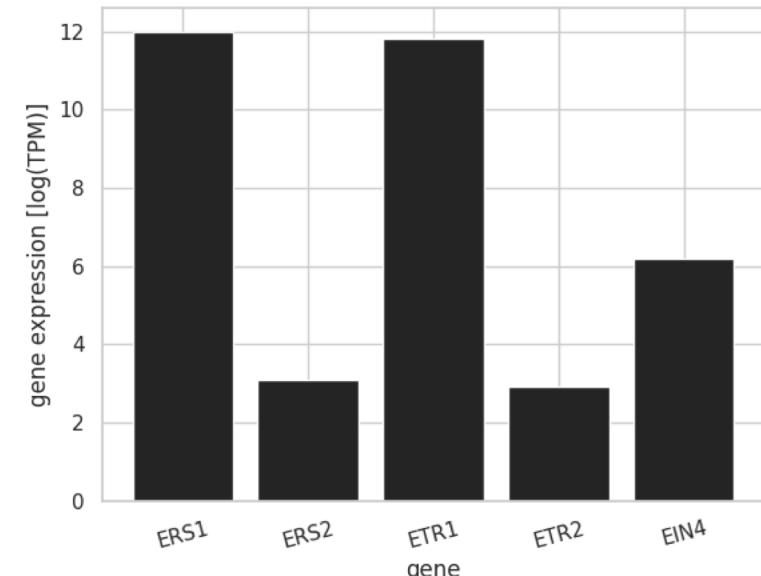
```
set_style='whitegrid'
set_palette='gray'
```

# bar chart

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set()

x = np.array(['ERS1', 'ERS2', 'ETR1',
              'ETR2', 'EIN4'])
y = np.array([12.0, 3.1, 11.8, 2.9, 6.2])
x_position = np.arange(len(x))

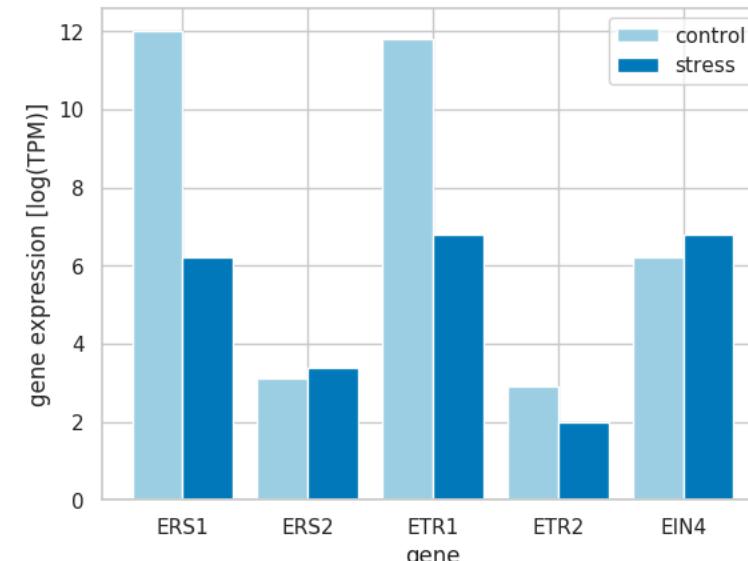
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.bar(x_position, y)
ax.set_xticks(x_position)
ax.set_xticklabels(x, rotation=15)
fig.show()
```



```
set_style='whitegrid'
set_palette='gray'
```

# bar chart

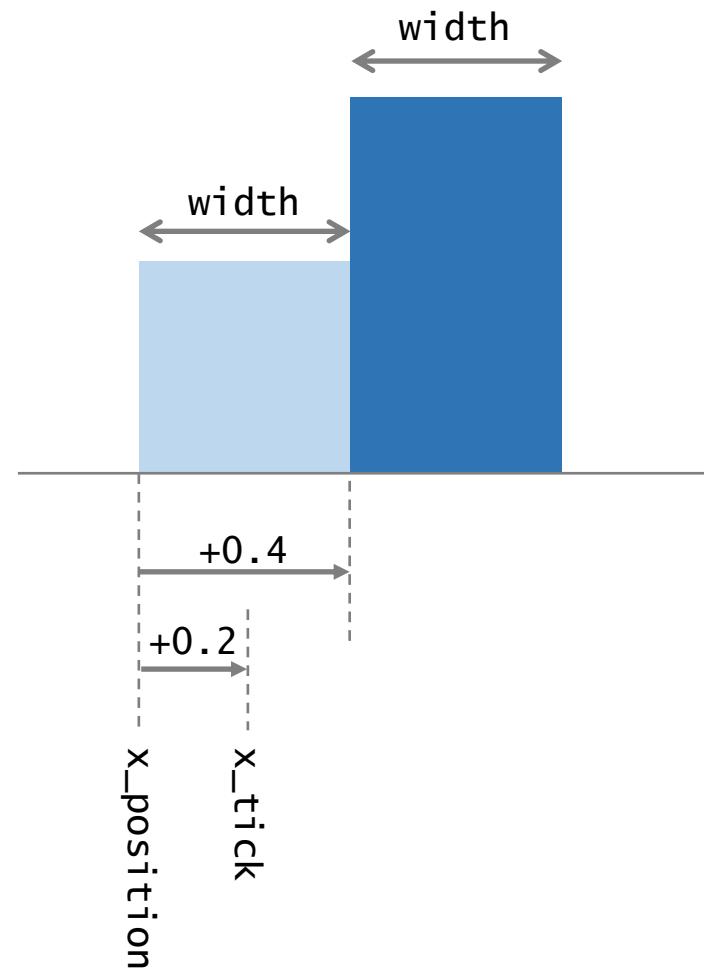
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set()
x = np.array(['ERS1', 'ERS2', 'ETR1',
              'ETR2', 'EIN4'])
x_position = np.arange(len(x))
y_control = np.array([12.0, 3.1, 11.8,
                      2.9, 6.2])
y_stress = np.array([6.2, 3.4, 6.8,
                     2.0, 6.8])
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.bar(x_position, y_control,
       width=0.4, label='control')
ax.bar(x_position + 0.4, y_stress,
       width=0.4, label='stress')
ax.legend()
ax.set_xticks(x_position + 0.2)
ax.set_xticklabels(x)
fig.show()
```



```
set_style='whitegrid'
set_palette='Paired'
```

# bar chart

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set()
x = np.array(['ERS1', 'ERS2', 'ETR1',
              'ETR2', 'EIN4'])
x_position = np.arange(len(x))
y_control = np.array([12.0, 3.1, 11.8,
                      2.9, 6.2])
y_stress = np.array([6.2, 3.4, 6.8,
                     2.0, 6.8])
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.bar(x_position, y_control,
       width=0.4, label='control')
ax.bar(x_position + 0.4, y_stress,
       width=0.4, label='stress')
ax.legend()
ax.set_xticks(x_position + 0.2)
ax.set_xticklabels(x)
fig.show()
```



# bar chart

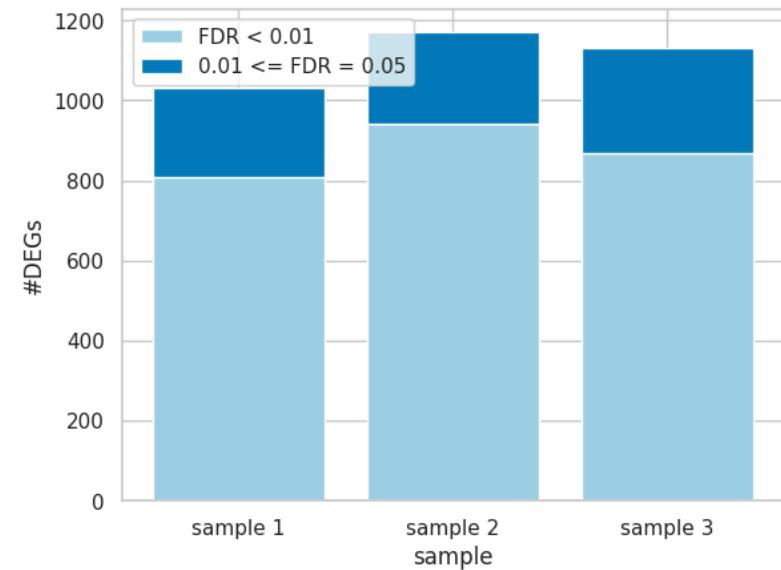
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set()

x = np.array(['sample 1', 'sample 2',
              'sample 3'])
x_position = np.arange(len(x))

y_DEG2 = np.array([220, 230, 260])
y_DEG1 = np.array([810, 940, 870])

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.bar(x_position, y_DEG1,
       label='FDR < 0.01')
ax.bar(x_position, y_DEG2, bottom=y_DEG1,
       label='0.01 <= FDR = 0.05')
```

```
fig.legend()
ax.set_xticks(x_position)
ax.set_xticklabels(x)
ax.set_xlabel('sample')
ax.set_ylabel('#DEGs')
fig.show()
```



```
set_style='whitegrid'
set_palette='Paired'
```

# bar chart

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

sns.set()

x = np.array(['ERS1', 'ERS2'])
x_position = np.arange(len(x))

ers1 = np.array([12.1, 10.9, 11.1, 12.9])
ers2 = np.array([8.6, 7.9, 10.2, 6.2])

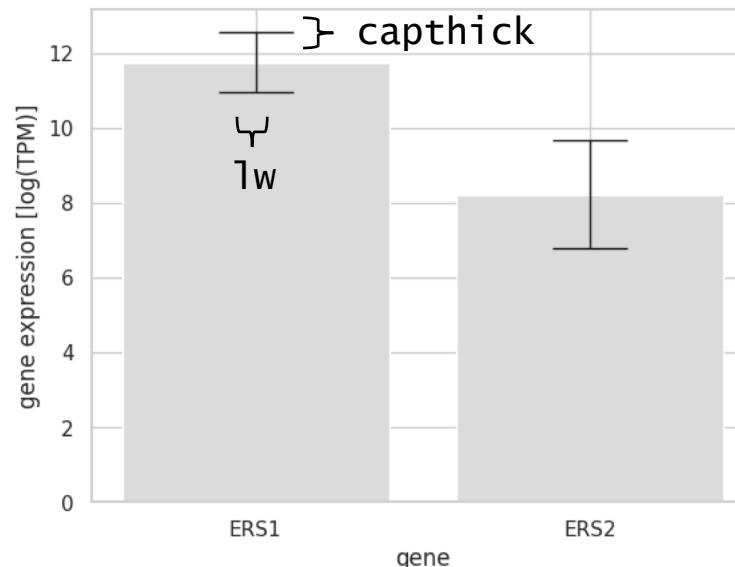
ers1_mu = ers1.mean()
ers2_mu = ers2.mean()
ers1_sd = ers1.std()
ers2_sd = ers2.std()

y = np.array([ers1_mu, ers2_mu])
e = np.array([ers1_sd, ers2_sd])
```

```
error_bar_set = dict(
    lw = 1,
    capthick = 1,
    capszie = 20
)

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.bar(x_position, y, yerr = e,
       tick_label=x,
       error_kw=error_bar_set)
fig.show()
```

capszie  
↔

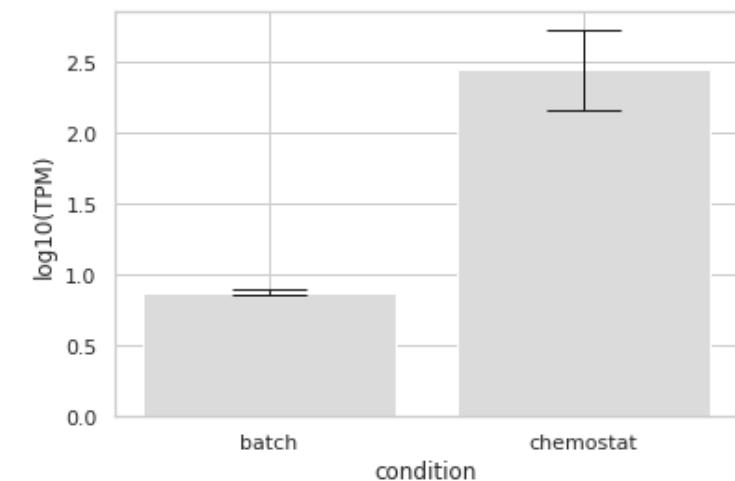


```
set_style='whitegrid'
set_palette='gist_yarg'
```

# 練習問題 3

遺伝子発現量TPM (count\_tpm.tsv) を読み込み、遺伝子 gene\_6042 の発現量を棒グラフで描け。ただし、縦軸は 対数化された TPM とし、横軸はサンプル名 (batch, chemostat) とし、複製実験を考慮し標準偏差をエラー バーとして描きいれること。

## ○ 解答例



# histogram

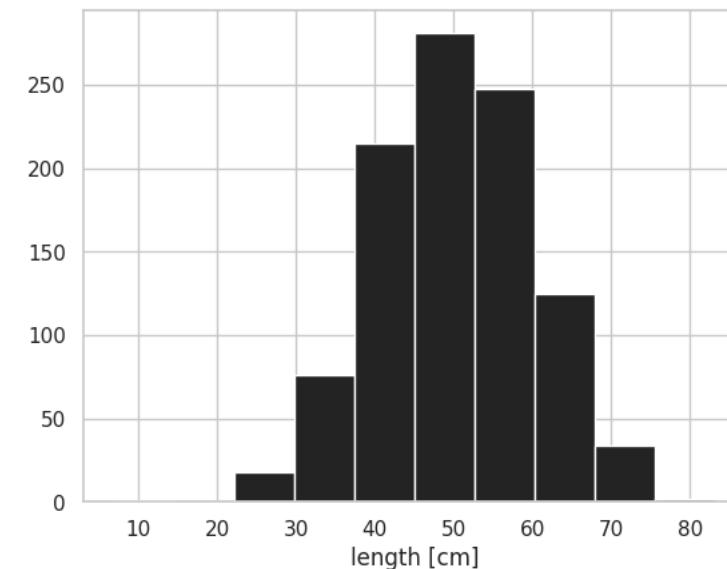
# histogram

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

sns.set()
np.random.seed(2018)

x = np.random.normal(50, 10, 1000)

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.hist(x)
ax.set_xlabel('length [cm]')
fig.show()
```



```
set_style='whitegrid'
set_palette='gray'
```

# histogram

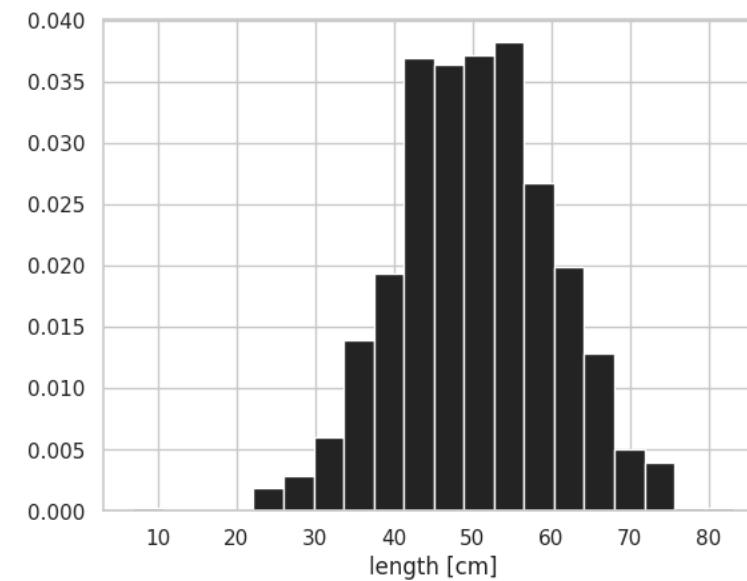
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.set()
np.random.seed(2018)

x = np.random.normal(50, 10, 1000)

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.hist(x, bins=20, density=True)
# ax.hist(x, bins=20, normed=True)

ax.set_xlabel('length [cm]')
fig.show()
```



```
set_style='whitegrid'
set_palette='gray'
```

# histogram

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

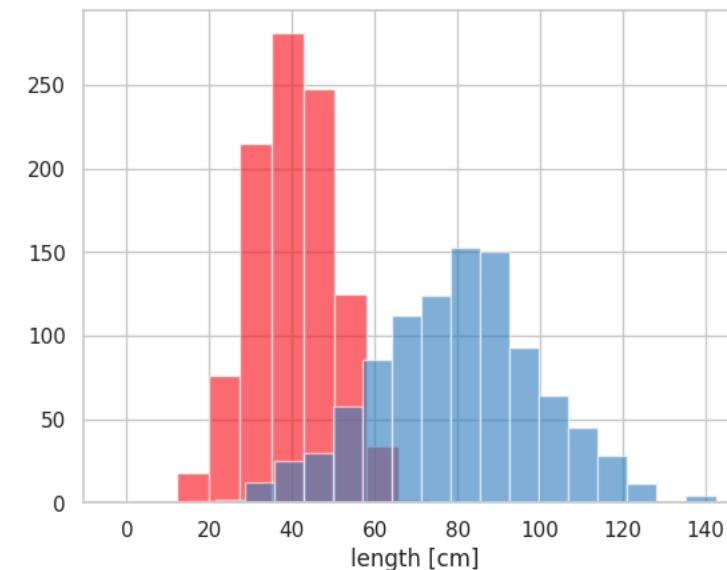
sns.set()
np.random.seed(2018)

x1 = np.random.normal(40, 10, 1000)
x2 = np.random.normal(80, 20, 1000)

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

ax.hist(x1, bins=10, alpha=0.6)
ax.hist(x2, bins=20, alpha=0.6)

ax.set_xlabel('length [cm]')
fig.show()
```

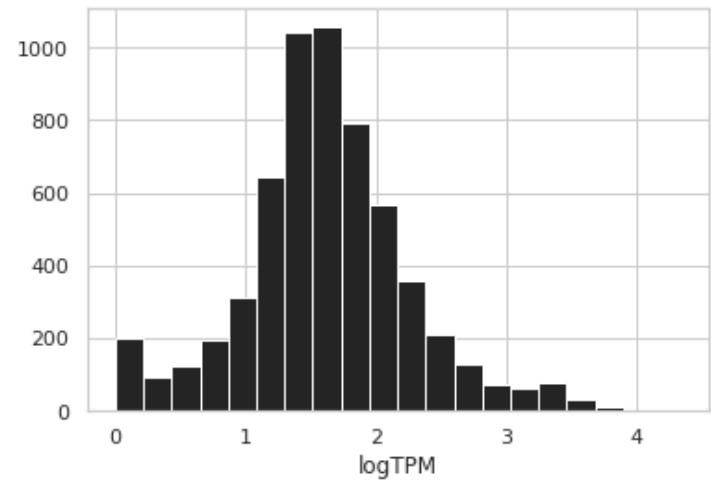


```
set_style='whitegrid'
set_palette='Set1'
```

# 練習問題 4

遺伝子発現量 TPM ファイル (count\_tpm.tsv) を読み込み、batch\_1 サンプルの TPM の分布をヒストグラムで示せ。

## ○ 解答例



# boxplot

# boxplot

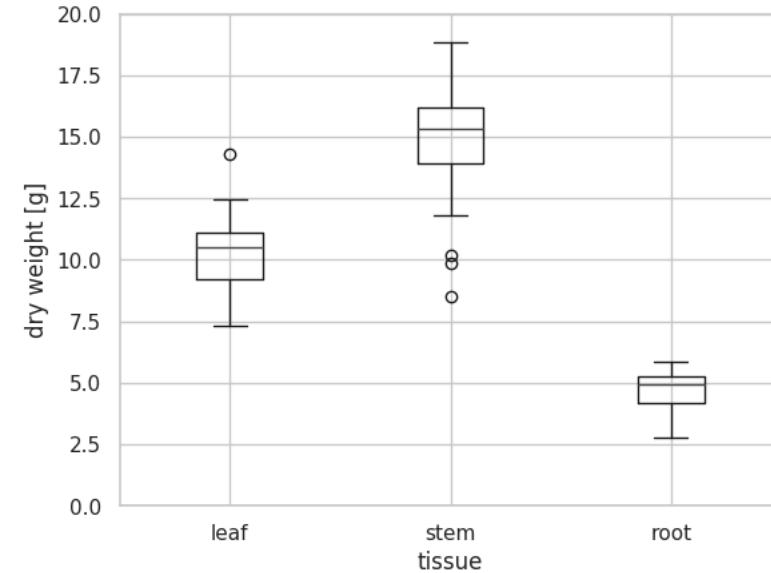
```
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
sns.set()
np.random.seed(2018)

x1 = np.random.normal(10, 2, 20)
x2 = np.random.normal(15, 3, 20)
x3 = np.random.normal(5, 1, 20)

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.boxplot([x1, x2, x3],
           labels=['leaf', 'stem', 'root'])

ax.set_xlabel('tissue')
ax.set_ylabel('dry weight [g]')
ax.set_ylim(0, 20)

fig.show()
```



set\_style='whitegrid'  
set\_palette='gray'

# boxplot

```

import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
sns.set()
np.random.seed(2018)

df = pd.DataFrame({
    'leaf': np.random.normal(10, 2, 20),
    'stem': np.random.normal(15, 3, 20),
    'root': np.random.normal(5, 1, 20)
})

df_melt = pd.melt(df)
print(df_melt.head())
##   variable      value
## 0      leaf  9.446465
## 1      leaf 11.163702
## 2      leaf 14.296799
## 3      leaf  7.441026
## 4      leaf 11.004554

```

```

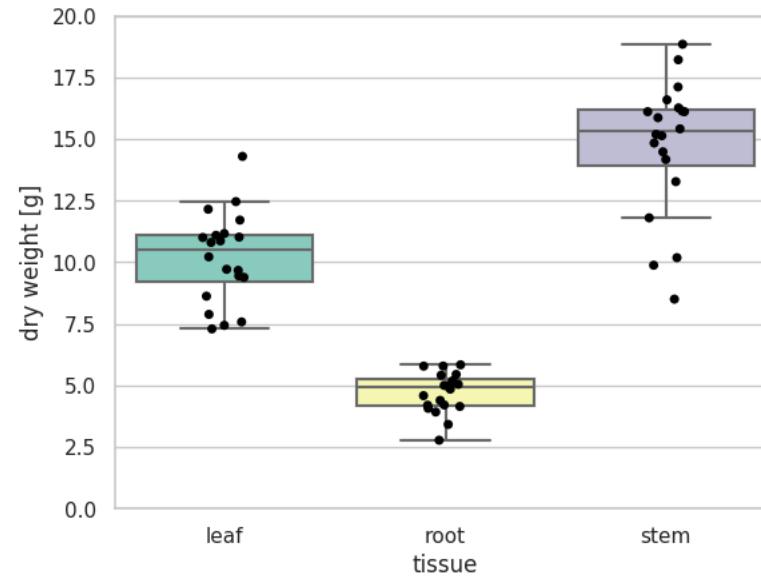
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

sns.boxplot(x='variable', y='value',
            data=df_melt,
            showfliers=False)

sns.stripplot(x='variable', y='value',
              data=df_melt,
              jitter=True,
              color='black')

fig.show()

```



```

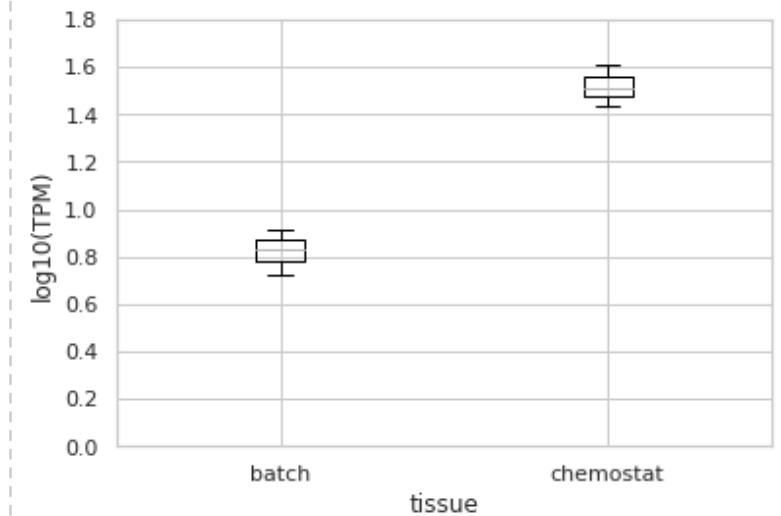
set_style='whitegrid'
set_palette='Set3'

```

# 練習問題 5

遺伝子発現量TPM (count\_tpm.tsv) を読み込み、遺伝子 gene\_6042 の発現量をボックスプロットで描け。

## ○ 解答例



**pie chart**

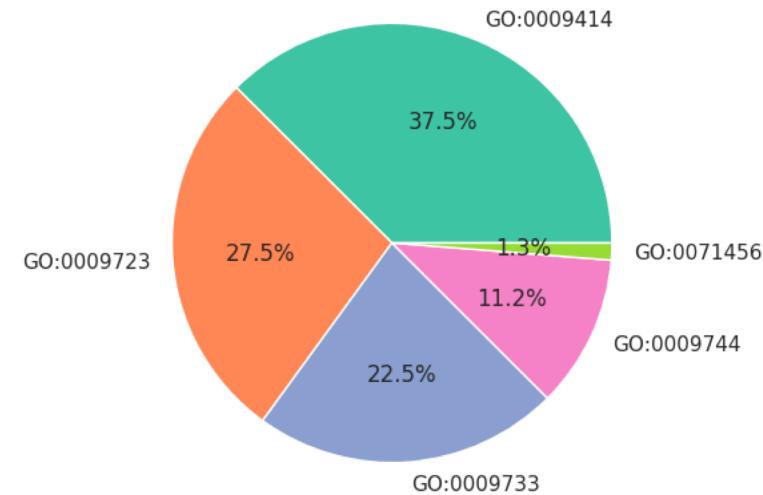
# pie chart

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

sns.set()

x = ['GO:0009414', 'GO:0009723', 'GO:0009733',
      'GO:0009744', 'GO:0071456']
y = np.array([300, 220, 180, 90, 10])

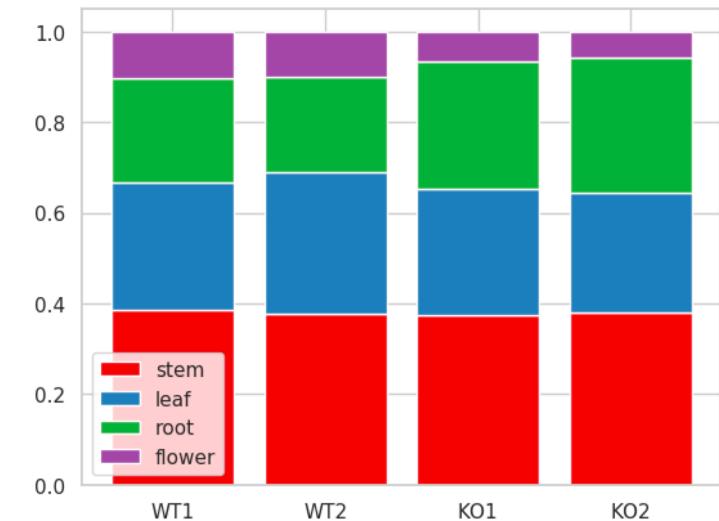
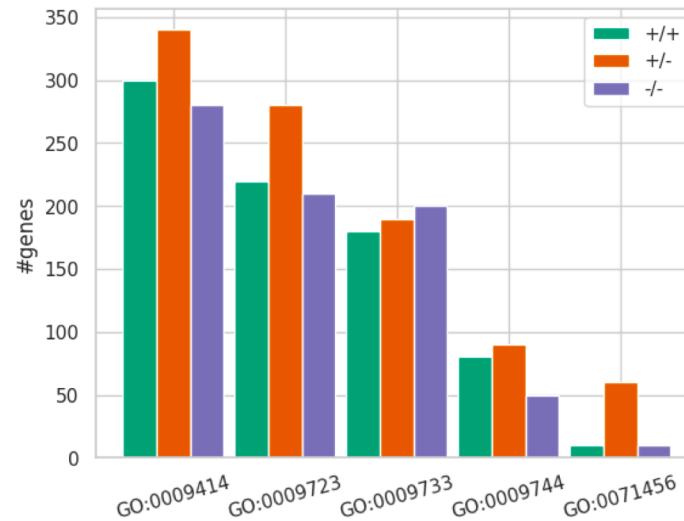
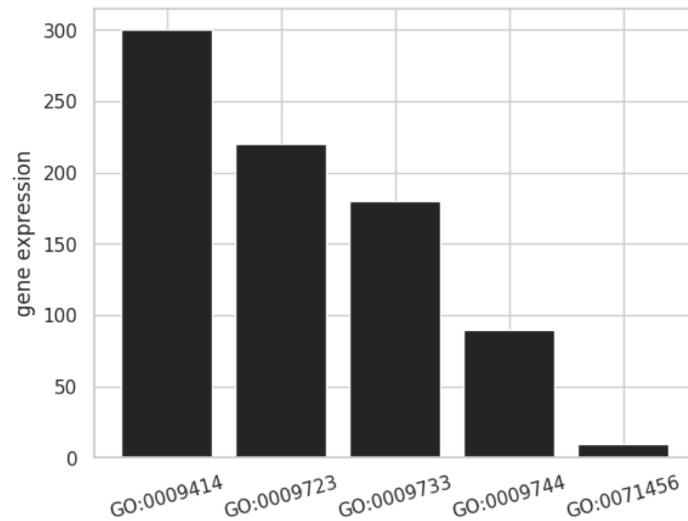
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.pie(y, labels=x, autopct="%1.1f%%")
ax.axis('equal')
fig.show()
```



set\_style='whitegrid'  
set\_palette='Set2'

円グラフは誤解されやすいグラフである。棒グラフなどで代用できるならば、棒グラフで描いた方わかりやすい。

# pie chart



# Outline



1 Data visualization



2 Visualization libraries



3 Overview



4 Basic charts



$+ \alpha$

# heatmap

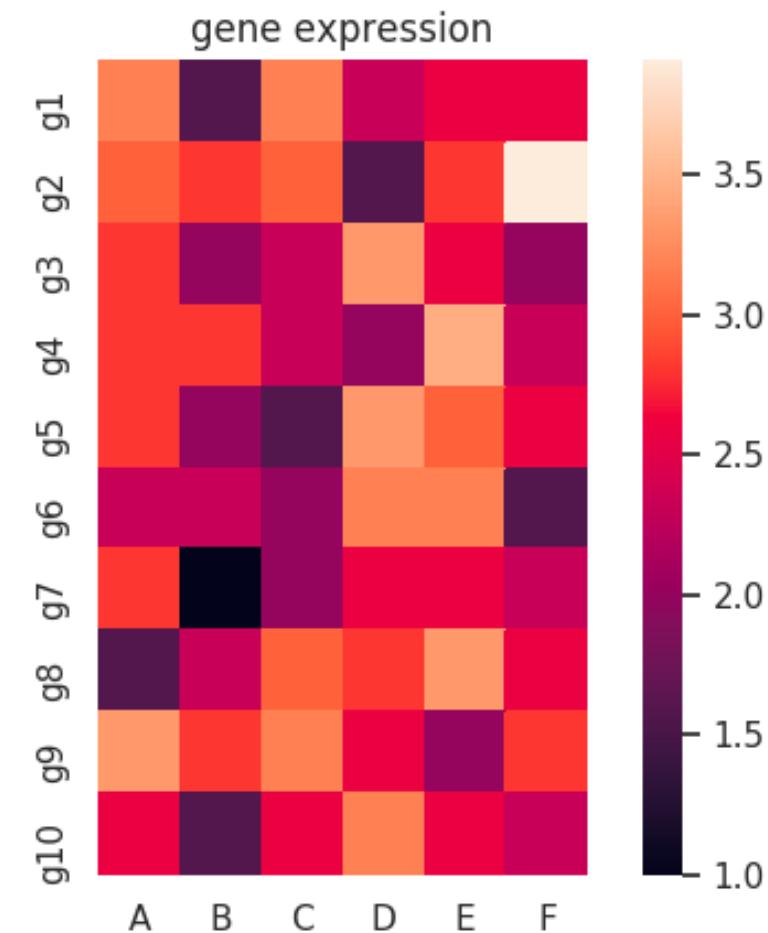
# heatmap

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
sns.set()
np.random.seed(2018)

data = np.random.binomial(100, 0.05, 60)
data = data.reshape((10, 6))
data = np.log2(data + 1)

xlabels = ['g' + str(i + 1) for i in range(10)]
ylabels = ['A', 'B', 'C', 'D', 'E', 'F']
df = pd.DataFrame(data, index=xlabels,
                   columns=ylabels)

sns.heatmap(df, square=True)
plt.title('gene expression')
plt.show()
```



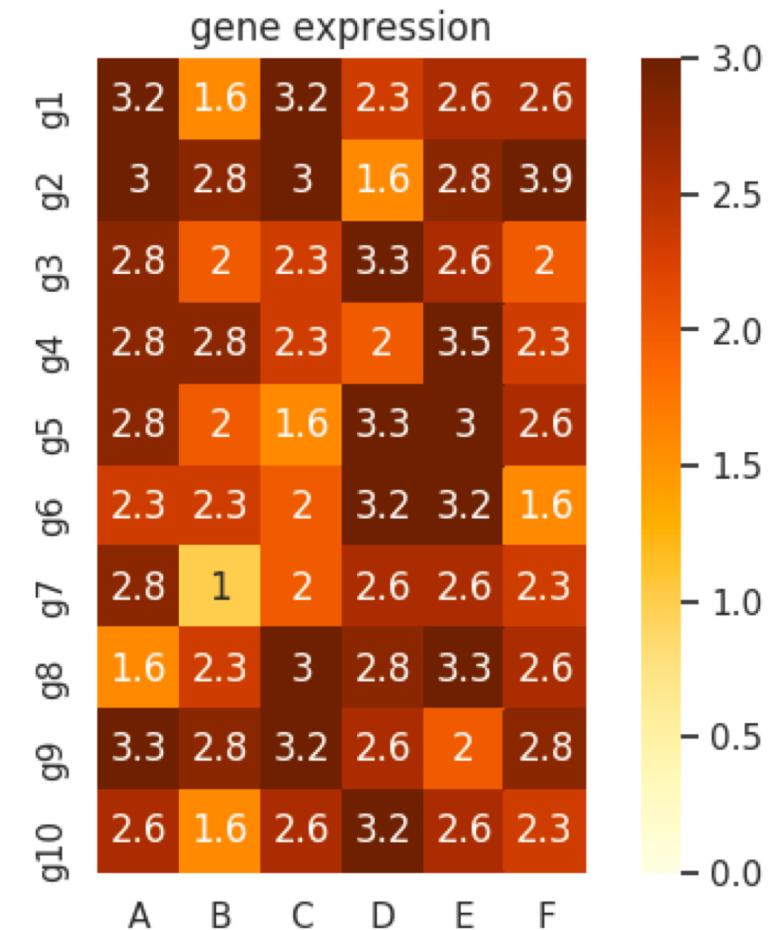
# heatmap

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
sns.set()
np.random.seed(2018)

data = np.random.binomial(100, 0.05, 60)
data = data.reshape((10, 6))
data = np.log2(data + 1)

xlabels = ['g' + str(i + 1) for i in range(10)]
ylabels = ['A', 'B', 'C', 'D', 'E', 'F']
df = pd.DataFrame(data, index=xlabels,
                   columns=ylabels)

sns.heatmap(df, annot=True, square=True,
             cmap='YlOrBr', vmin=0, vmax=3)
plt.title('gene expression')
plt.show()
```



# heatmap

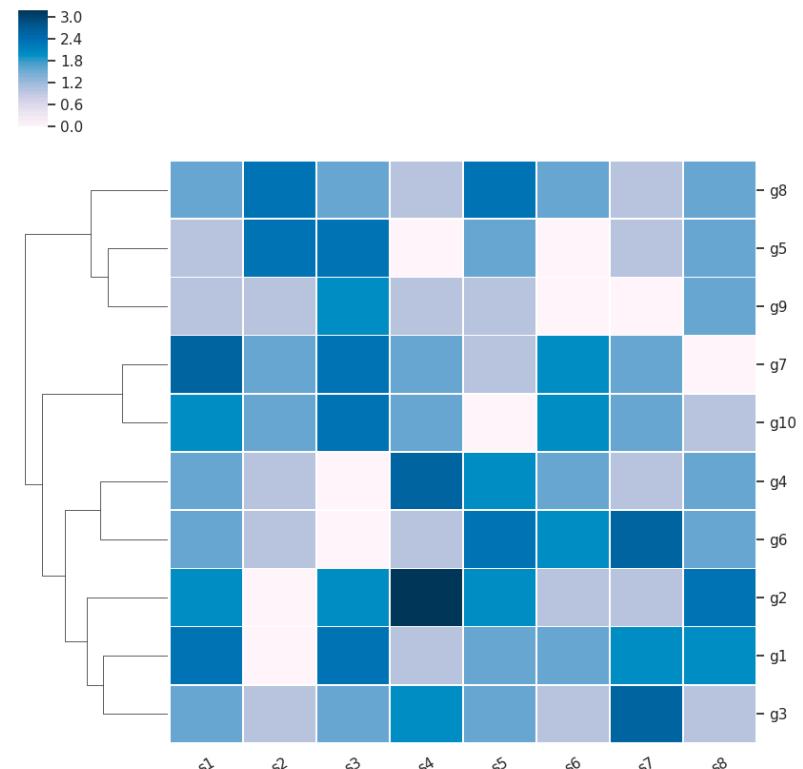
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
np.random.seed(2018)
sns.set()

data = np.random.binomial(100, 0.02, 80)
data = data.reshape((10, 8))
data = np.log2(data + 1)

xlabels = ['g' + str(i + 1) for i in range(10)]
ylabels = ['s' + str(i + 1) for i in range(8)]
df = pd.DataFrame(data, index=xlabels,
                    columns=ylabels)

sns_plot = sns.clustermap(df,
                           method='ward', metric='euclidean',
                           col_cluster = False,
                           cmap='PuBu', linewidths=0.5)
```

```
plt.setp(sns_plot.ax_heatmap.get_yticklabels(),
          rotation=0)
plt.setp(sns_plot.ax_heatmap.get_xticklabels(),
          rotation=30)
plt.show()
```



# heatmap

```

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

np.random.seed(2018)
sns.set()

data = np.random.binomial(100, 0.02, 80)
data = data.reshape((10, 8))
data = np.log2(data + 1)

xlabels = ['g' + str(i + 1) for i in range(10)]
ylabels = ['s' + str(i + 1) for i in range(8)]
df = pd.DataFrame(data, index=xlabels,
                   columns=ylabels)

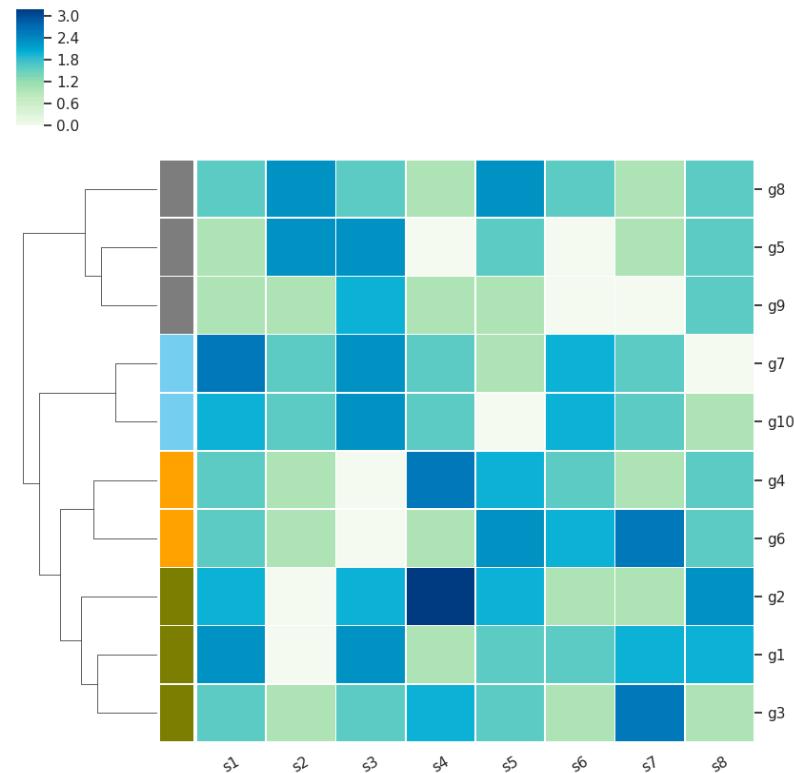
gene_loc = ['olive', 'olive', 'olive',
            'orange', 'gray', 'orange',
            'skyblue', 'gray', 'gray',
            'skyblue']

```

```

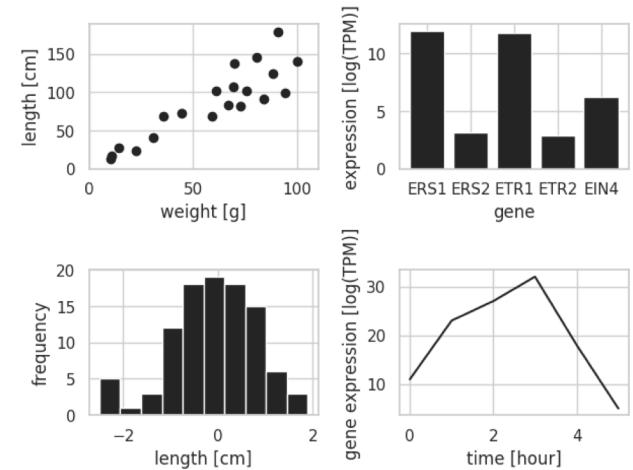
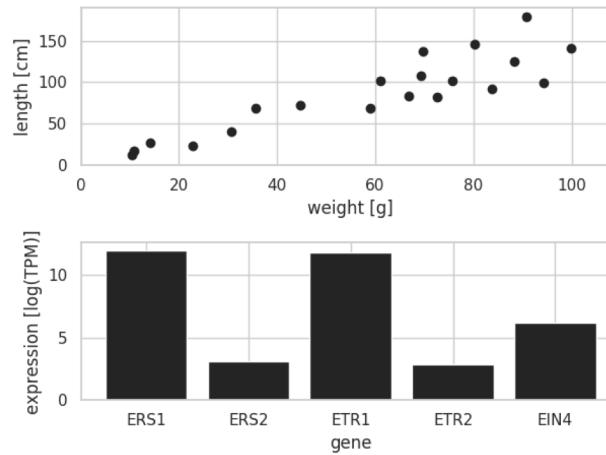
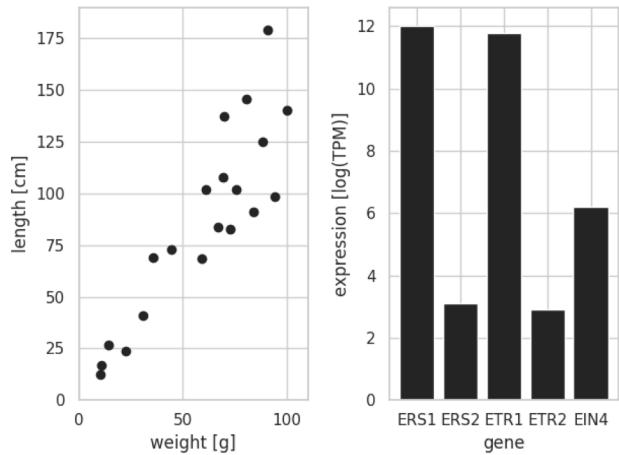
sns_plot = sns.clustermap(df,
                           method='ward', metric='euclidean',
                           col_cluster = False, cmap='GnBu',
                           linewidths=0.5,
                           row_colors = gene_loc)
plt.setp(sns_plot.ax_heatmap.get_yticklabels(),
         rotation=0)
plt.setp(sns_plot.ax_heatmap.get_xticklabels(),
         rotation=30)
plt.show()

```



# subplot

# subplot



# subplot

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

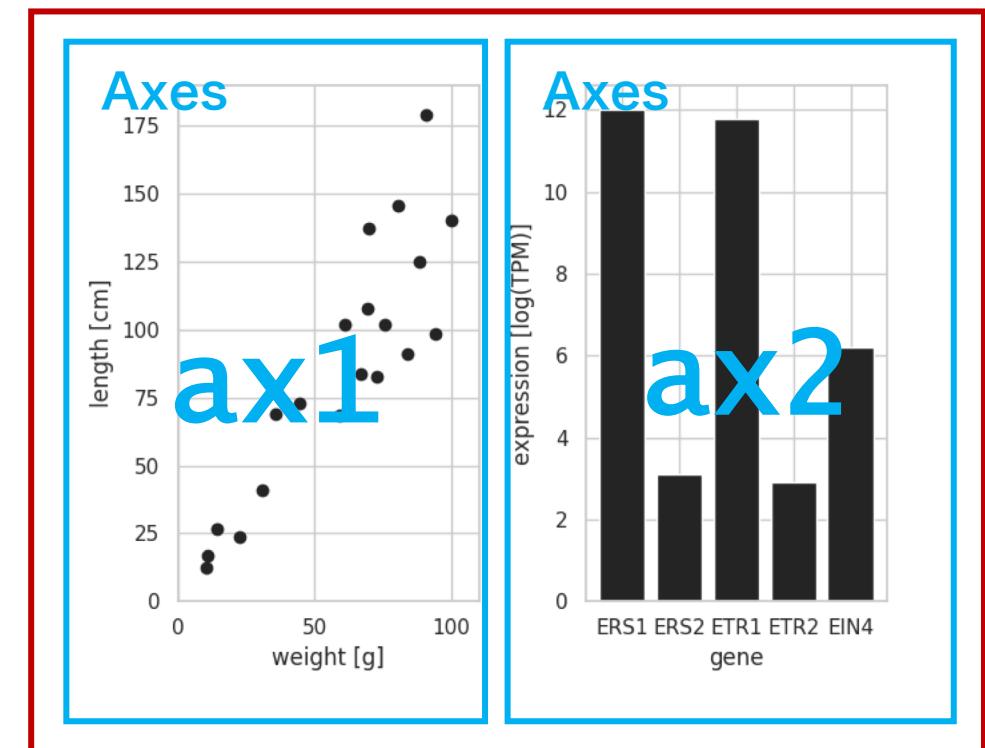
fig = plt.figure()

x1 = np.random.uniform(0, 100, 20)
y1 = x1 * np.random.uniform(1, 2, 20)
ax1 = fig.add_subplot(1, 2, 1)
ax1.scatter(x1, y1)

x2 = np.array(['ERS1', 'ERS2', 'ETR1',
               'ETR2', 'EIN4'])
y2 = np.array([12.0, 3.1, 11.8, 2.9, 6.2])
x2_position = np.arange(len(x2))
ax2 = fig.add_subplot(1, 2, 2)
ax2.bar(x2_position, y2, tick_label=x2)

fig.show()
```

Figure



# subplot

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

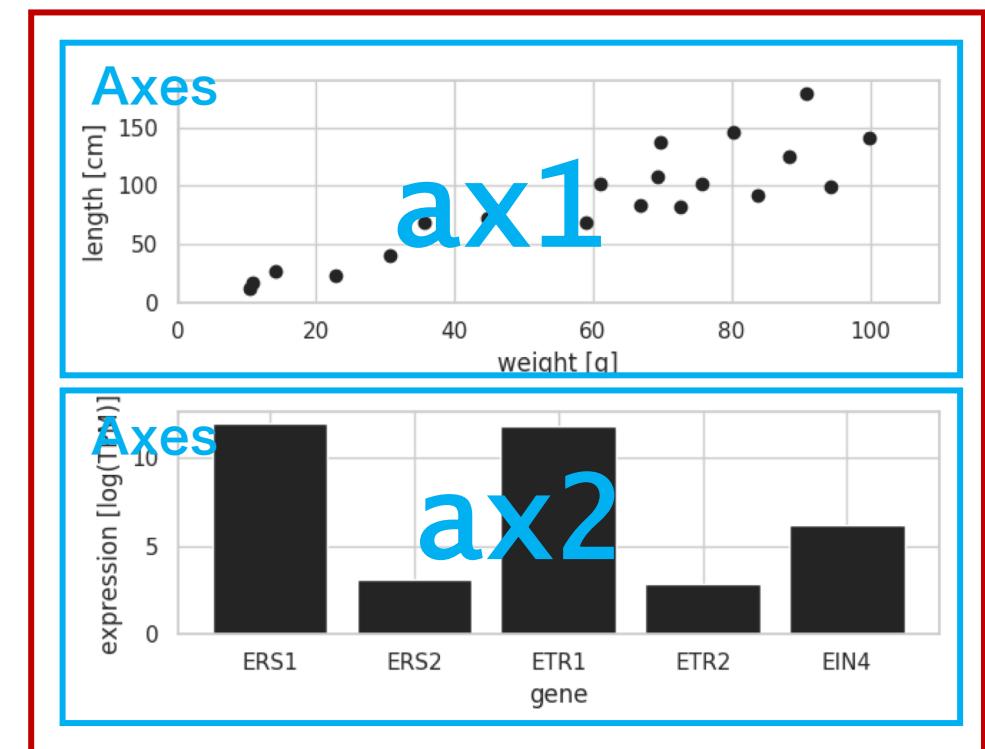
fig = plt.figure()

x1 = np.random.uniform(0, 100, 20)
y1 = x1 * np.random.uniform(1, 2, 20)
ax1 = fig.add_subplot(2, 1, 1)
ax1.scatter(x1, y1)

x2 = np.array(['ERS1', 'ERS2', 'ETR1',
               'ETR2', 'EIN4'])
y2 = np.array([12.0, 3.1, 11.8, 2.9, 6.2])
x2_position = np.arange(len(x2))
ax2 = fig.add_subplot(2, 1, 2)
ax2.bar(x2_position, y2, tick_label=x2)

fig.show()
```

Figure



# subplot

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

fig = plt.figure()

ax1 = fig.add_subplot(2, 2, 1)
ax1.scatter(x1, y1)

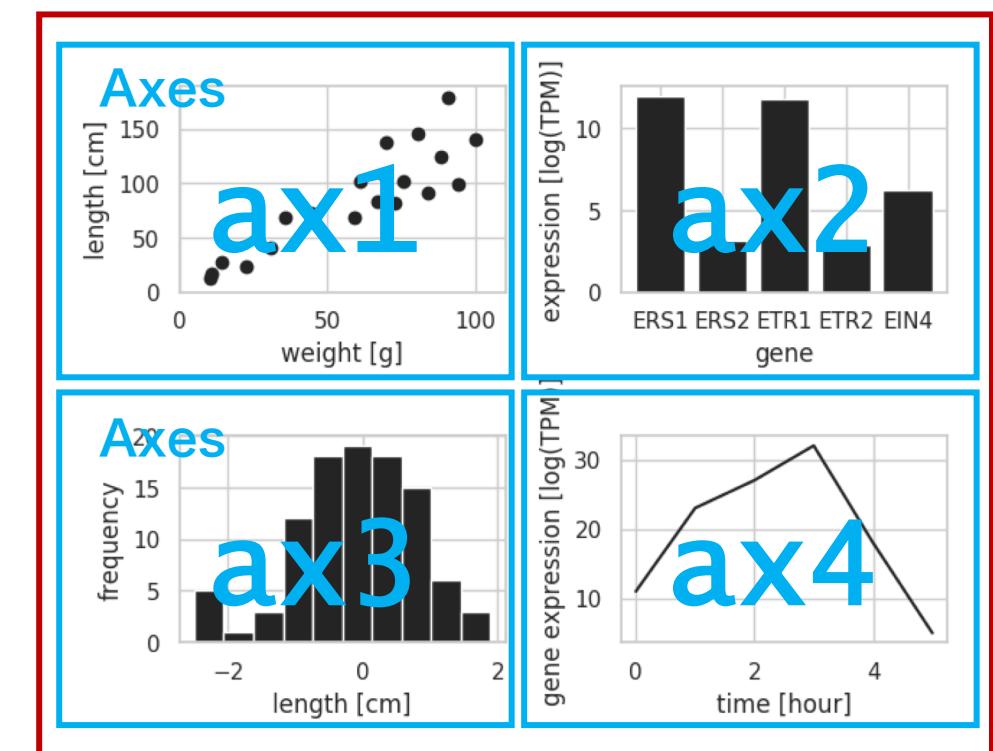
ax2 = fig.add_subplot(2, 2, 2)
ax2.bar(x2, y2)

ax3 = fig.add_subplot(2, 2, 3)
ax3.hist(x3)

ax4 = fig.add_subplot(2, 2, 4)
ax4.plot(x4, y4)

fig.show()
```

Figure





## useful references

- **matplotlib gallery**  
<https://matplotlib.org/gallery.html>

- **seaborn example gallery**  
<https://seaborn.pydata.org/examples/index.html>

- **Python Graph Gallery**  
<https://python-graph-gallery.com/>  
<https://stats.biopapyrus.jp/python/python-graph-gallery.html>