matplotlib で作成する図の調整方法

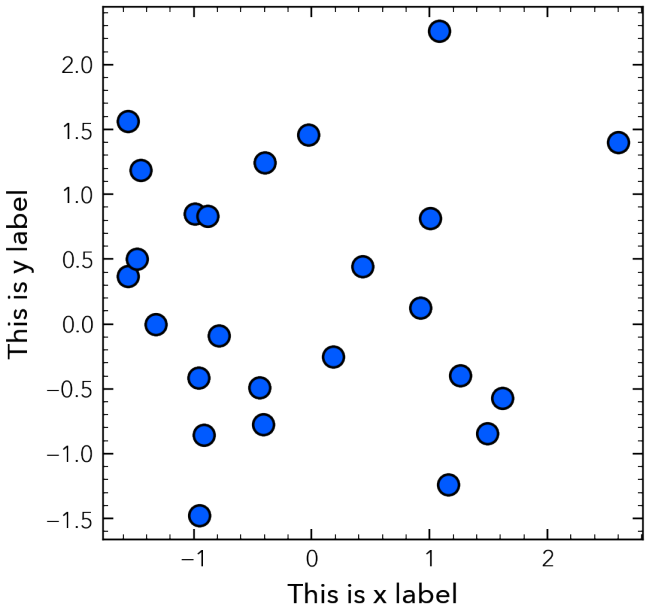
sugayu

2025 年 8 月 2 日

目次

1 基準の図

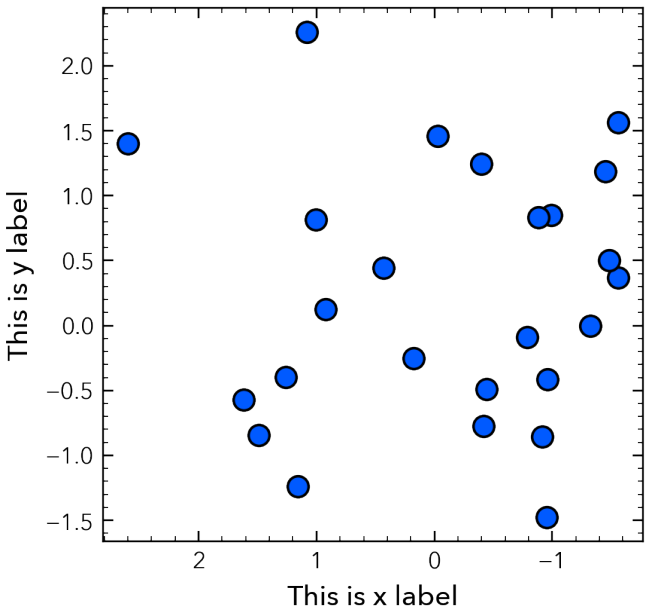
```
from numpy.random import default_rng
from sugayutils.figure import makefig

rng = default_rng(222)
data = rng.standard_normal(50).reshape(2, 25)


def plot_fiducial():
    fig = makefig(figsize=['small', 1.0])
    ax = fig.add_subplot(1, 1, 1)
    ax.scatter(data[0], data[1], c='blue')
    ax.set_xylabels('This is x label', 'This is
    ↪  y label')
    return ax

_ = plot_fiducial()
```



2 目盛り

2.1 目盛り反転

```
ax = plot_fiducial()
_ = ax.invert_xaxis()
```



2.2 目盛りの数字

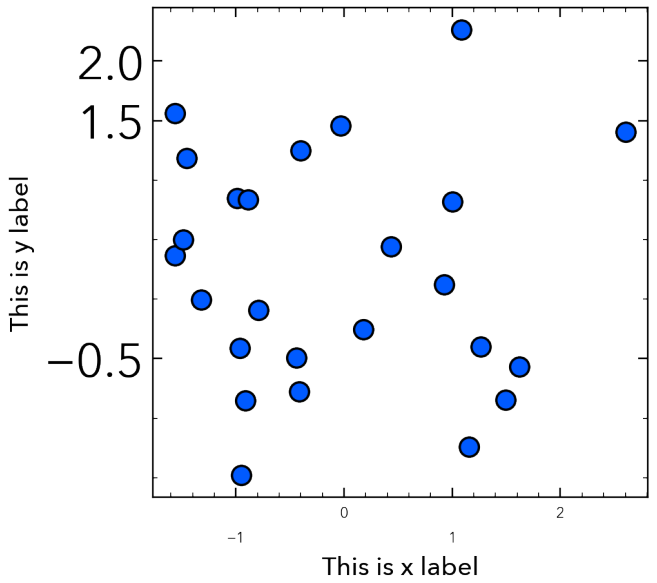
目盛りの情報はテキストとして格納されている。

```
ax = plot_fiducial()
print(ax.get_xticklabels())
plt.close()
```

[Text(-2.0, 0, ' − 2'), Text(-1.0, 0, ' − 1'), Text(0.0, 0, '0'), Text(1.0, 0, '1'), Text(2.0, 0, '2'), Text(3.0, 0, '3')]

よって、このテキスト情報を変更してやればテキスト位置などを細かく変更することが可能である。これを利用して、目盛りラベルのサイズを変更し、縦軸の目盛りを指定する。また、手作業で目盛りの位置を変更する。

```
ax = plot_fiducial()
ax.tick_params(labelsize='x-small') # both axes
ax.tick_params(axis='y', labelsize=20)
_ = ax.set_yticks([-0.5, 1.5, 2.0])
for i, xticklabel in
↪  enumerate(ax.get_xticklabels()):
    if i % 2 == 1:
        _, _y = xticklabel.get_position()
        xticklabel.set_y(_y - 0.05)
```

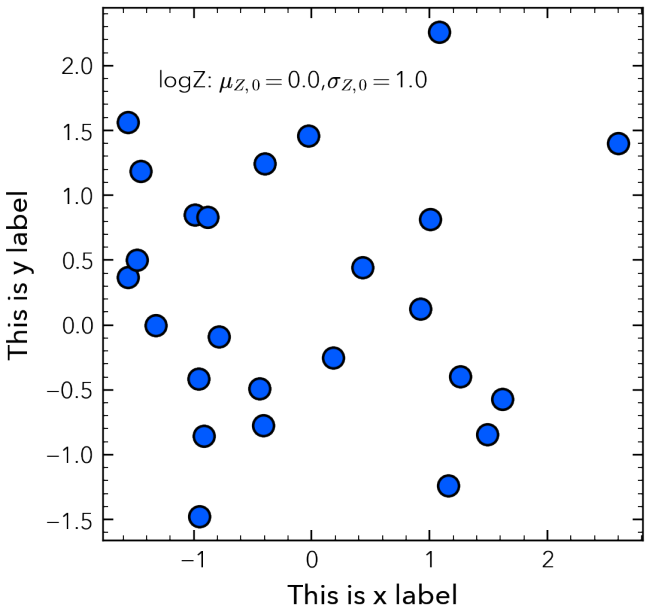


3 テキスト

3.1 f-string と LaTeX

f-string と LaTeX 記法は併存できる。このとき f-string 記法の {} が LaTeX 記法と衝突することにより、LaTeX 記法の {} に変更が加わる。

```
ax = plot_fiducial()
mu, sigma = 0.0, 1.0
_ = ax.text(
    *(0.1, 0.85),
    fr'logZ: $\mu_{{Z, 0}}=${mu},$\sigma_{{Z,
    ↪  0}}=${sigma}',
    transform=ax.transAxes,
)
```

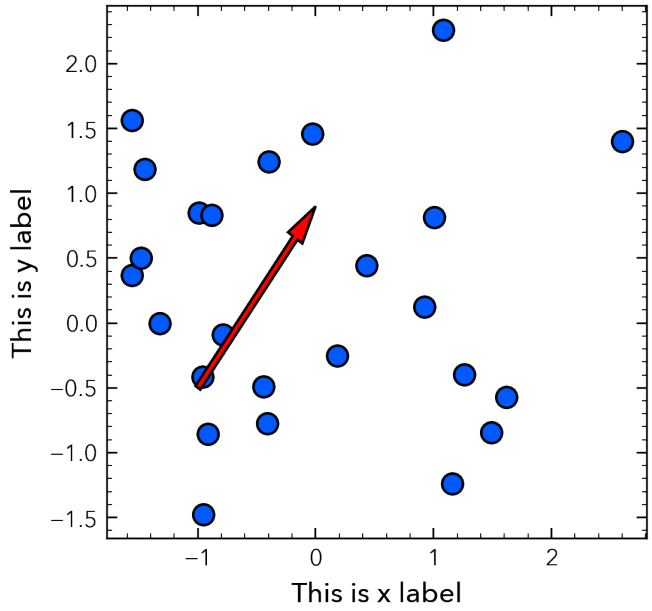


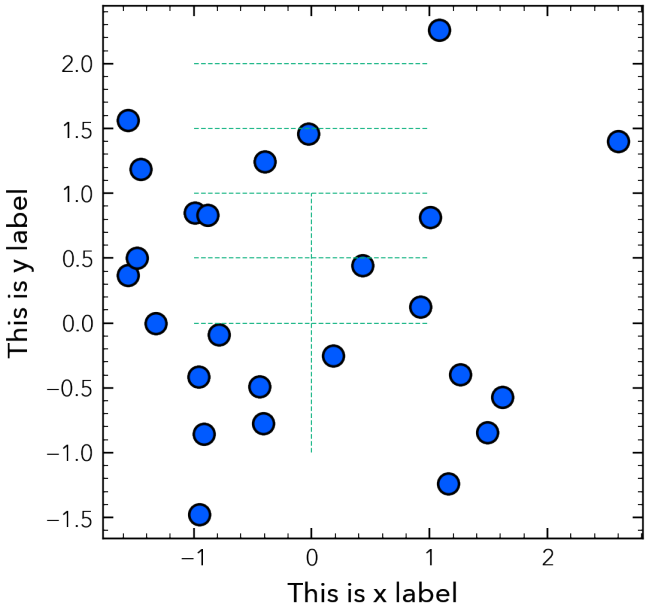
4 矢印

4.1 arrow

データ座標を使って矢印を描く。

```
ax = plot_fiducial()
_ = ax.arrow(
    x=-1.0,
    y=-0.5,
    dx=1.0,
    dy=1.4,
    width=0.05,
    head_length=0.3,
    length_includes_head=True,
    fc='red',
)
```





5 大量の線

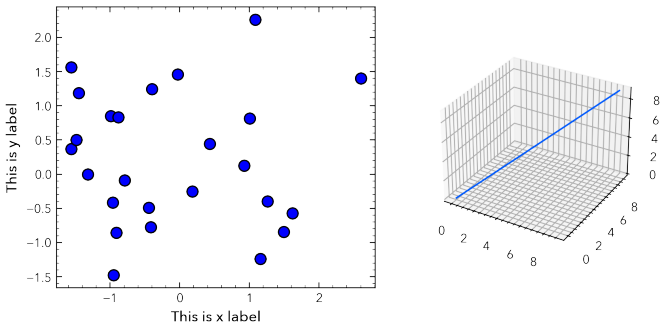
一斉に同じ種類の線をプロットするには `mcoll.LineCollection` を使って、返り値を `ax.add_collection()` で加えると良い。

```
import matplotlib.collections as mcoll
from sugayutils import colors

ax = plot_fiducial()
segments = (
    ((-1.0, 0.0), (1.0, 0.0)),
    ((-1.0, 0.5), (1.0, 0.5)),
    ((-1.0, 1.0), (1.0, 1.0)),
    ((-1.0, 1.5), (1.0, 1.5)),
    ((-1.0, 2.0), (1.0, 2.0)),
    ((0.0, -1.0), (0.0, 1.0)),
)
linecollection = mcoll.LineCollection(segments,
↪  colors=colors.green, lw=0.5, ls='--')
_ = ax.add_collection(linecollection)
```

6 グリッド分け

`fig.subplots()` と `fig.subplots_adjust()` の組み合わせでもグリッドを切れるが、`matplotlib.gridspec.GridSpec` を使うと引数 `width_ratios` などを使ってより柔軟なグリッドを作ることができる。`GridSpec` は `fig.subplots()` に引数として与えることもできるが、`fig.add_subplot()` で個別にパネルを作ることで `projection` などを柔軟に対応させられる。

```
from matplotlib.gridspec import GridSpec
gs = GridSpec(1, 2, width_ratios=(1.5, 1))
fig = plt.figure(figsize=[7.2, 3.5])
ax0 = fig.add_subplot(gs[0])
ax1 = fig.add_subplot(gs[1], projection='3d')
ax0.scatter(data[0], data[1], c='blue')
ax0.set_xlabel('This is x label')
ax0.set_ylabel('This is y label')
_ = ax1.plot(np.arange(0., 10.0), np.arange(0.,
↪  10.0), np.arange(0., 10.0))
```



参照: python - Matplotlib different size subplots - Stack Overflow

- Image nonuniform — Matplotlib 3.10.5 documentation

7 図

7.1 Nonuniform image

ピクセルの形が長方形になるような、各列や行によってピクセル幅が異なる画像を作成する際には **NonUniformImage** を使う。**ax.imshow** は画像 (Image) を定義する以外に内部で様々な設定を同時にしてくれているが、**NonUniformImage** を使う場合には自分で画像の設定をする必要がある。例えば、以下の例では **extent** 自体は **NonUniformImage** を呼ぶ際に設定しているが、画像の縦横サイズ **xlim** と **ylim** は自動では設定されないので、**im.set_extent()** を明示的に呼ぶことで **extent** に合わせて画像サイズを設定している。

```
from matplotlib.image import NonUniformImage

fig = makefig(figsize=['small', 0.6])

x = (np.arange(15) - 7.0)
x = x**3 / 7.0**3 * 3.0
y = (np.arange(9) - 4.0)
image = np.exp(-0.5 * (x[None, ...]**2 + y[...,
↪  None]**2))
extent = (-3.5, 3.5, -4.5, 4.5)
kw = dict(
    extent=extent,
    origin='lower',
    cmap='YlGn',
)

ax = fig.add_subplot(1, 2, 1)
im = NonUniformImage(ax,
↪  interpolation='nearest', **kw)
im.set_data(x, y, image)
im.set_extent(extent)
ax.add_image(im)
ax.set_aspect('auto')

ax = fig.add_subplot(1, 2, 2)
ax.imshow(image, aspect='auto', **kw)
_ = ax.set_title('Uniform (worng scale)')
```