tueplots

Nicholas Krämer

Apr 23, 2024

GETTING STARTED

TUEplots is a light-weight matplotlib extension that adapts your figure sizes to formats more suitable for scientific publications. It produces configurations that are compatible with matplotlib's *rcParams*, and provides fonts, figure sizes, font sizes, color schemes, and more, for a number of publication formats.

CHAPTER

ONE

SUPPORTED VENUES

The following venues are currently supported by TUEplots out of the box as pre-configured bundles:

- Conference on Neural Information Processing Systems (NeurIPS)
- International Conference on Artificial Intelligence and Statistics (AISTATS)
- International Conference on Learning Representations (ICLR)
- International Conference on Machine Learning (ICML)
- Conference on Uncertainty in Artificial Intelligence (UAI)
- Journal of Machine Learning Research (JMLR)

For further details on the available bundles, check out the *tueplots.bundles API documentation*.

1.1 Installation

Install via pip:

```
pip install tueplots
```

or get the latest version from source:

```
pip install git+https://github.com/pnkraemer/tueplots.git
```

Please be aware that during the early stages of development, some interfaces may be subject to change.

1.2 Usage examples

`tueplots` provides some recipes for scientific plotting. For example, figure sizes can be tailored straightforwardly to some common journal page layouts:

```
>>> from tueplots import figsizes
>>> figsizes.jmlr2001()["figure.figsize"]
(6.0, 1.8541019662496847)
```

within one module, the functions have a unified interface (wherever possible)

```
>>> figsizes.jmlr2001(nrows=2)["figure.figsize"]
(6.0, 3.7082039324993694)
>>>
>>> figsizes.neurips2021(nrows=3)["figure.figsize"]
(5.5, 5.0987804071866325)
>>>
>>> # The full output:
>>> figsizes.icml2022_full(nrows=4)
{'figure.figsize': (6.75, 8.343458848123582), 'figure.constrained_layout.use': True,
↪'figure.autolayout': False, 'savefig.bbox': 'tight', 'savefig.pad_inches': 0.015}
```

There are also predefined color constants. For example, those based on the corporate design of the University of Tuebingen:

```
>>> from tueplots.constants.color import rgb
>>>
>>> rgb.tue_dark
array([0.21568627, 0.25490196, 0.29019608])
>>>
>>> rgb.tue_gray
array([0.68627451, 0.70196078, 0.71764706])
```

Most of the output types of functions in `tueplots` are dictionaries that are directly compatible with matplotlib's `rcParam` language.

```
>>> from tueplots import markers
>>>
>>> markers.inverted()
{'lines.markeredgecolor': 'auto', 'lines.markerfacecolor': 'white', 'lines.
↪markeredgewidth': 0.75}

>>> import matplotlib.pyplot as plt

>>> # Use them as context managers:
>>> with plt.rc_context(markers.inverted()):
...     pass # do your plotting...

>>> # Or change your global configuration
>>> plt.rcParams.update(markers.inverted())
```

For more detailed tutorials, please have a look at the examples in the `examples/` directory.

1.3 Example application: ICML 2022

If you're getting ready to submit your paper to ICML 2022, plug either of the following into your preamble.

1.3.1 In a nutshell

The minimal amount of code required to fit your figure to the ICML 2022 style (source: https://media.icml.cc/Conferences/ICML2022/Styles/example_paper.pdf) is `plt.rcParams.update(bundles.icml2022())`:

```
>>> import matplotlib.pyplot as plt
>>> from tueplots import bundles
>>> bundles.icml2022()
{'text.usetex': True, 'font.family': 'serif', 'text.latex.preamble': '\\usepackage{times}
↪ ', 'figure.figsize': (3.25, 2.0086104634371584), 'figure.constrained_layout.use':␣
↪True, 'figure.autolayout': False, 'savefig.bbox': 'tight', 'savefig.pad_inches': 0.015,
↪ 'font.size': 8, 'axes.labelsize': 8, 'legend.fontsize': 6, 'xtick.labelsize': 6,
↪'ytick.labelsize': 6, 'axes.titlesize': 8}
>>> bundles.icml2022(family="sans-serif", usetex=False, column="full", nrows=2)
{'text.usetex': False, 'font.serif': ['Times'], 'mathtext.fontset': 'stix', 'mathtext.rm
↪': 'Times', 'mathtext.it': 'Times:italic', 'mathtext.bf': 'Times:bold', 'font.family':
↪'sans-serif', 'figure.figsize': (6.75, 8.343458848123582), 'figure.constrained_layout.
↪use': True, 'figure.autolayout': False, 'savefig.bbox': 'tight', 'savefig.pad_inches':␣
↪0.015, 'font.size': 8, 'axes.labelsize': 8, 'legend.fontsize': 6, 'xtick.labelsize': 6,
↪ 'ytick.labelsize': 6, 'axes.titlesize': 8}
>>>
>>> # Plug any of those into either the rcParams or into an rc_context:
>>> plt.rcParams.update(bundles.icml2022())
>>> with plt.rc_context(bundles.icml2022()):
...     pass
```

1.3.2 Some customisation

If you don't want a pre-packaged solution, at least fix your figure- and font-sizes as follows.

```
>>> from tueplots import figsizes, fontsizes, fonts
>>> figsizes.icml2022_full()
{'figure.figsize': (6.75, 2.0858647120308955), 'figure.constrained_layout.use': True,
↪'figure.autolayout': False, 'savefig.bbox': 'tight', 'savefig.pad_inches': 0.015}
>>> figsizes.icml2022_half(nrows=2, constrained_layout=True, tight_layout=False)
{'figure.figsize': (3.25, 4.017220926874317), 'figure.constrained_layout.use': True,
↪'figure.autolayout': False, 'savefig.bbox': 'tight', 'savefig.pad_inches': 0.015}
>>> fontsizes.icml2022()
{'font.size': 8, 'axes.labelsize': 8, 'legend.fontsize': 6, 'xtick.labelsize': 6, 'ytick.
↪labelsize': 6, 'axes.titlesize': 8}
>>> fonts.icml2022()
{'text.usetex': False, 'font.serif': ['Times'], 'mathtext.fontset': 'stix', 'mathtext.rm
↪': 'Times', 'mathtext.it': 'Times:italic', 'mathtext.bf': 'Times:bold', 'font.family':
↪'serif'}
>>> fonts.icml2022(family="serif")
```

(continues on next page)

(continued from previous page)

```
{'text.usetex': False, 'font.serif': ['Times'], 'mathtext.fontset': 'stix', 'mathtext.rm
↪': 'Times', 'mathtext.it': 'Times:italic', 'mathtext.bf': 'Times:bold', 'font.family':
↪'serif'}
>>> fonts.icml2022_tex(family="sans-serif")
{'text.usetex': True, 'font.family': 'sans-serif', 'text.latex.preamble': '\\usepackage
↪{times} \\renewcommand{\\familydefault}{\\sfdefault} \\usepackage{sansmath} \\sansmath
↪'}
```

and if you want to give your plots a makeover (albeit a slightly opinionated one) with a single line of code, consider the `axes.lines()` setting.

```
>>> from tueplots import axes
>>> axes.lines()
{'axes.linewidth': 0.5, 'lines.linewidth': 1.0, 'xtick.major.width': 0.5, 'ytick.major.
↪width': 0.5, 'xtick.minor.width': 0.25, 'ytick.minor.width': 0.25, 'xtick.major.size':␣
↪3.0, 'ytick.major.size': 3.0, 'xtick.minor.size': 2.0, 'ytick.minor.size': 2.0, 'grid.
↪linewidth': 0.5, 'patch.linewidth': 0.5, 'legend.edgecolor': 'inherit', 'axes.axisbelow
↪': True}
>>> axes.lines(base_width=0.5)
{'axes.linewidth': 0.5, 'lines.linewidth': 1.0, 'xtick.major.width': 0.5, 'ytick.major.
↪width': 0.5, 'xtick.minor.width': 0.25, 'ytick.minor.width': 0.25, 'xtick.major.size':␣
↪3.0, 'ytick.major.size': 3.0, 'xtick.minor.size': 2.0, 'ytick.minor.size': 2.0, 'grid.
↪linewidth': 0.5, 'patch.linewidth': 0.5, 'legend.edgecolor': 'inherit', 'axes.axisbelow
↪': True}
```

1.4 FAQ: Frequently asked questions

1.4.1 My version of matplotlib cannot find font XYZ?!

Some fonts that `tueplot` provides (e.g., `Times` or `Roboto`) must be installed on your machine before matplotlib can find them. This means that you need to find a `.ttf` file online (e.g., the `Roboto` family is available at Google fonts: link-to-roboto, download it, and install it. For Ubuntu, this means opening the file (with your font manager) and clicking `install`. There are probably many other ways to do this. Once the font is installed, delete your matplotlib cache (usually: `rm ~/.cache/matplotlib -rf`) and restart your notebook (not just the kernel). See also this question on Stack Overflow.

On a related note: if you want to use the Latex version of the fonts/bundles, your system must include the required tex packages. See this Stack Overflow discussion for information. You may encounter this problem when using Tueplots in combination with a Google Colab notebook.

1.4.2 My fonts are not displayed as expected with `plt.rc_context()`

In our experience, changing fonts works more reliably with `plt.rcParams.update()` than with `plt.rc_context()`. If someone knows why, please let us know. :)

1.4.3 I am still getting 'overfull hbox' errors in my latex code

Even though the figure sizes delivered by Tueplots match the figure sizes in style files exactly, sometimes, figures can be a few points wider than what latex allows. Visually, this does not make any difference, but it might lead to an 'overfull hbox' raised by, e.g., 'pdflatex'. This is not Tueplots' fault but likely due to the optimisation that matplotlib carries out as part of constrained_layout=True or tight_layout=True. Refer to the constrained layout documentation for more info.

Solution: If you really cannot live with this warning, there are the following possible solutions:

- Instead of `\includegraphics(<plot>)`, use `\includegraphics[width=\textwidth](<plot>)`. This fixes the final few pixels, and the visual difference is non-existent.
- Set the `rel_width` in the figsizes to, e.g., `rel_width=0.97` and use `\includegraphics(<plot>)` as usual.

1.4.4 My submission template requires Type 1 fonts

Type 1 fonts are a tricky requirement; for example, because Adobe will disable support for authoring with Type 1 fonts in January 2023 . Matplotlib cannot do Type 1 fonts but uses Type 3 as a default. There is also no way of making matplotlib use Type 1 fonts. The only options are Type 3 and Type 42 (/TrueType) fonts.

Solution: If you *need* a Type 1 font, using TeX typesetting usually does the trick: E.g., via `bundles.icml2022(usetex=True)`, or `fonts.icml2022_tex()`. If the goal, however, is only to avoid type 3 fonts, adding `plt.rcParams.update({"pdf.fonttype": 42})` to your plotting code will create a PDF with `TrueType` fonts. See this issue for more details.

1.5 Styling figure axes

[1]:
```
import matplotlib.pyplot as plt

from tueplots import axes

# Increase the resolution of all the plots below
plt.rcParams.update({"figure.dpi": 150})
```

We can change the axes behaviour via `tueplots.axes`.

[2]:
```
axes.lines()
```

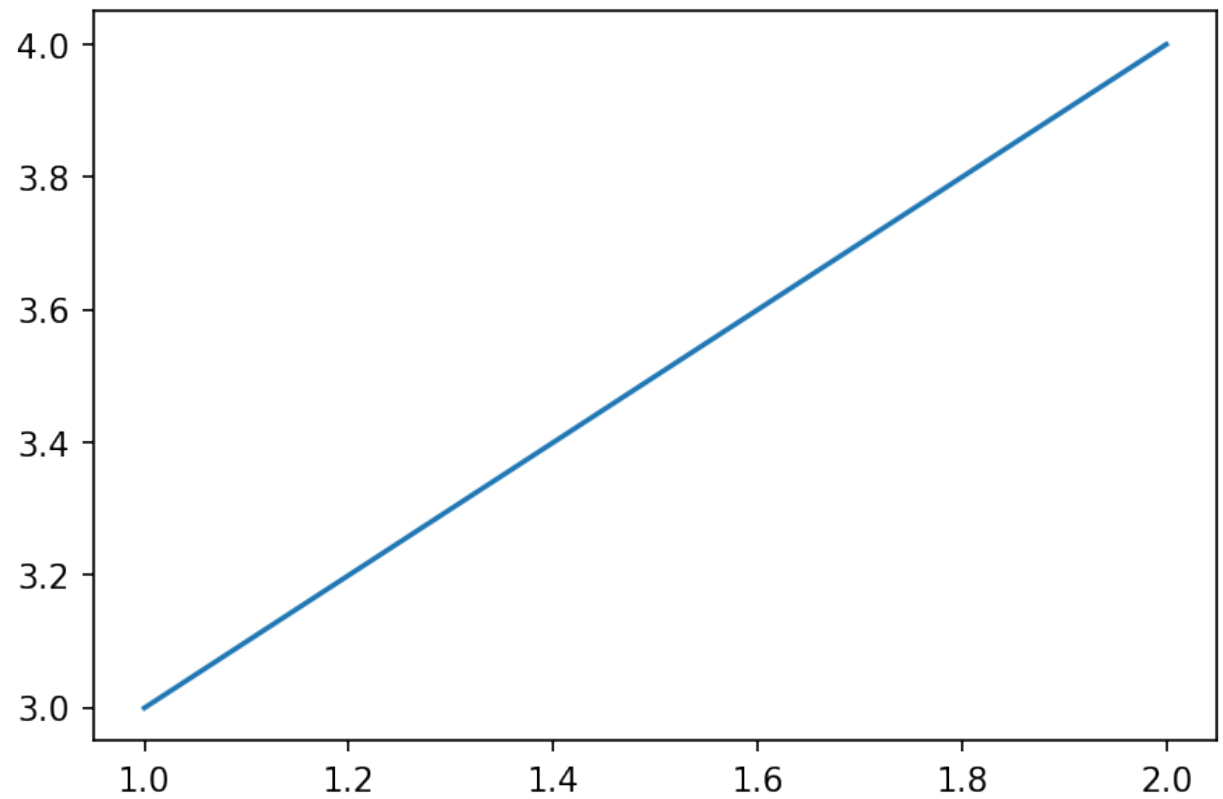
[2]:
```
{'axes.linewidth': 0.5,
 'lines.linewidth': 1.0,
 'xtick.major.width': 0.5,
 'ytick.major.width': 0.5,
 'xtick.minor.width': 0.25,
 'ytick.minor.width': 0.25,
 'xtick.major.size': 3.0,
 'ytick.major.size': 3.0,
 'xtick.minor.size': 2.0,
```

(continues on next page)

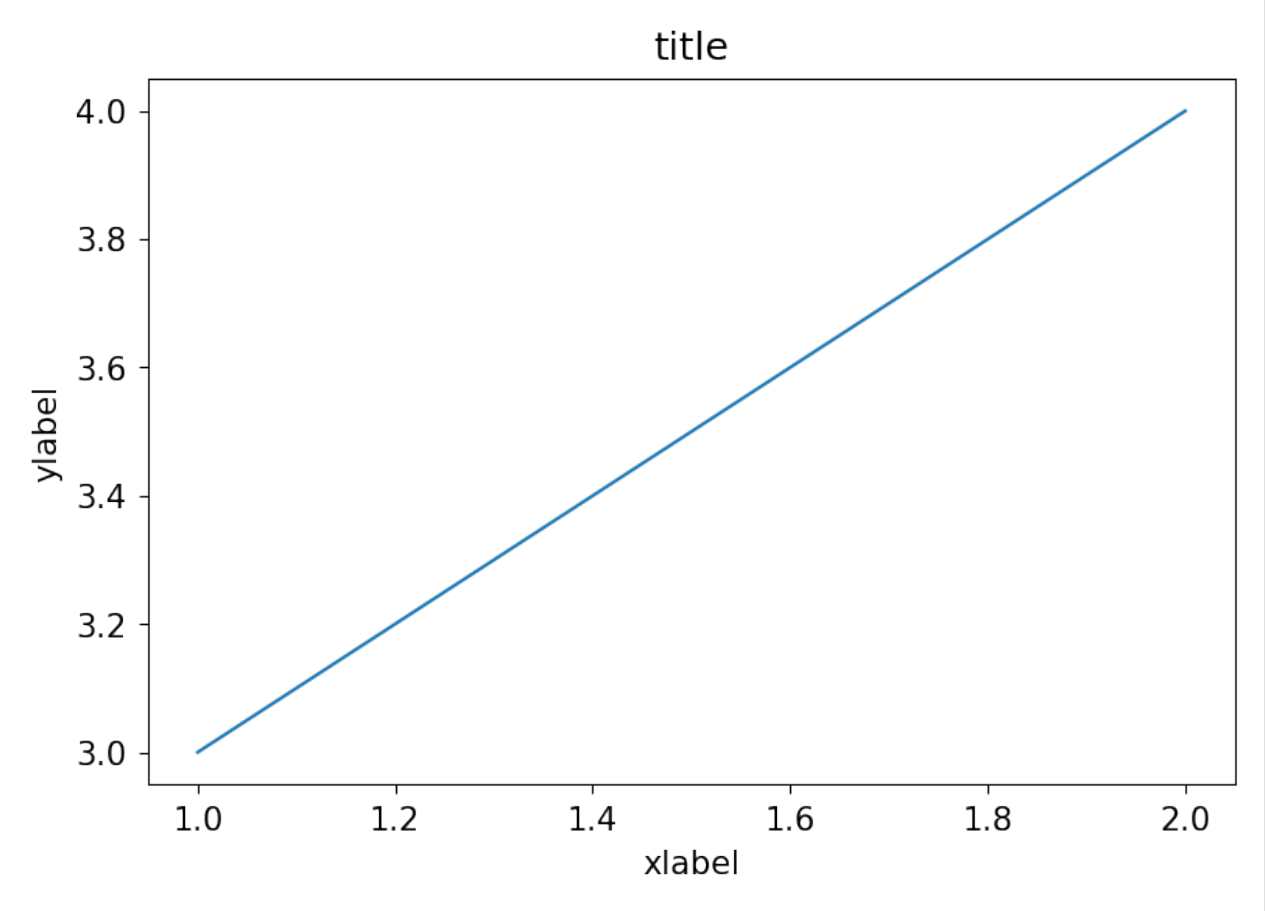
(continued from previous page)

```
 'ytick.minor.size': 2.0,
 'grid.linewidth': 0.5,
 'patch.linewidth': 0.5,
 'legend.edgecolor': 'inherit',
 'axes.axisbelow': True}
```

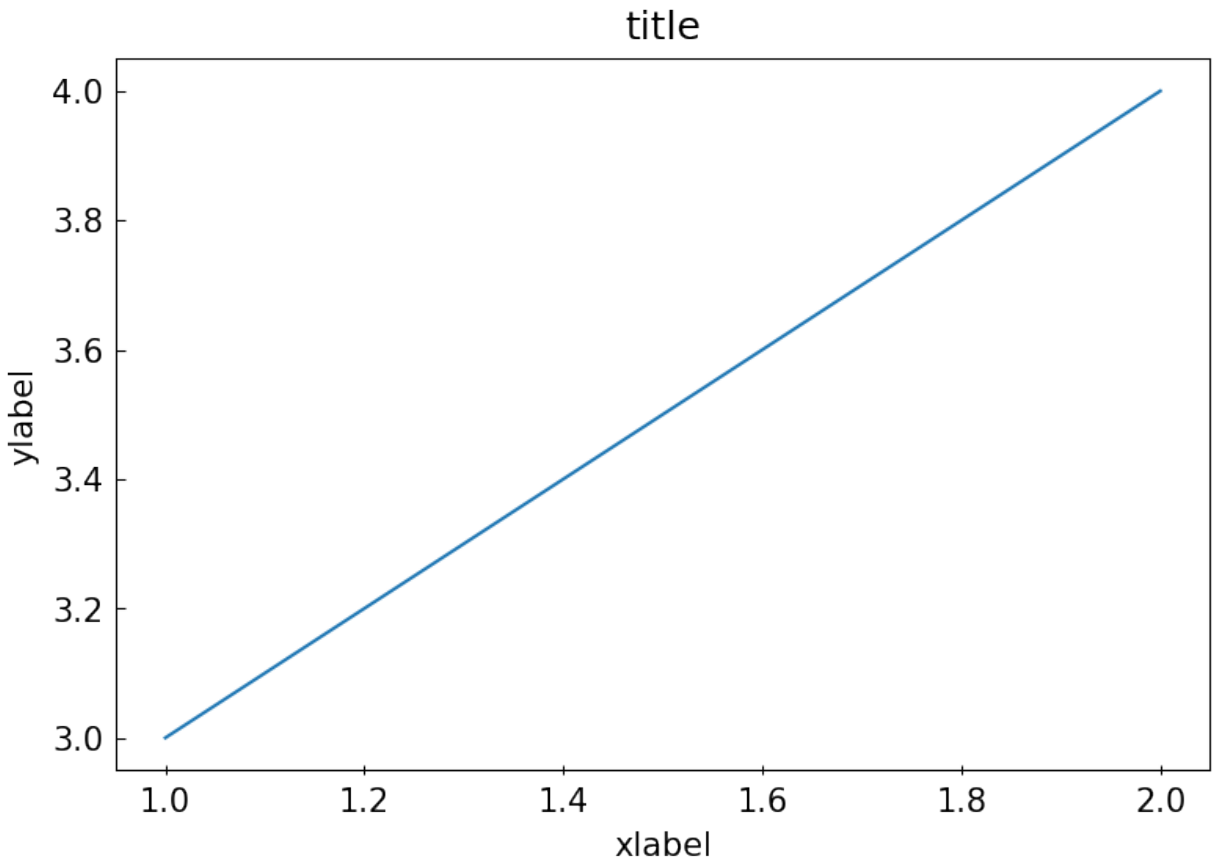
[3]:
```
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
plt.show()
```



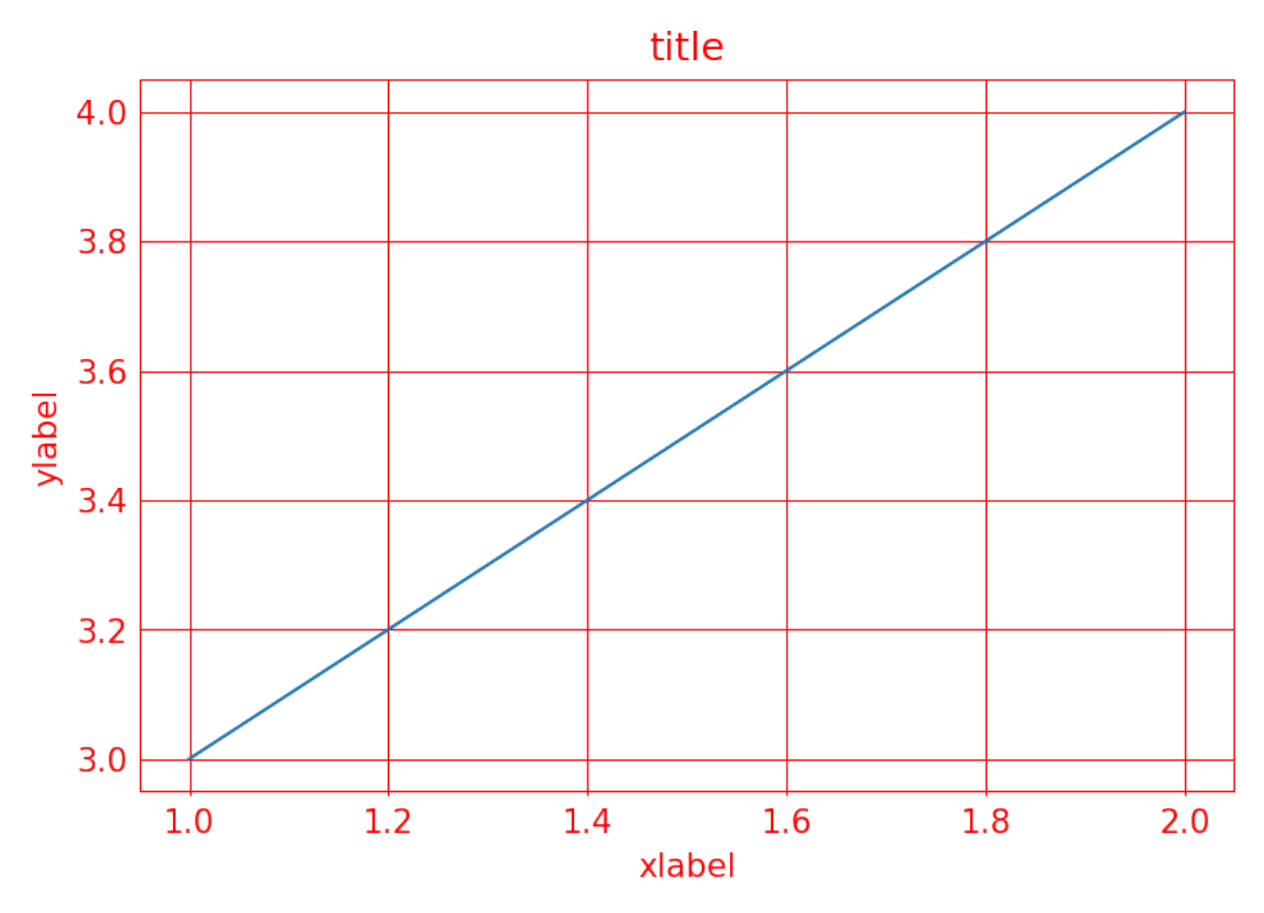
[4]:
```
plt.rcParams.update(axes.lines())
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
ax.set_xlabel("xlabel")
ax.set_ylabel("ylabel")
ax.set_title("title")
plt.show()
```



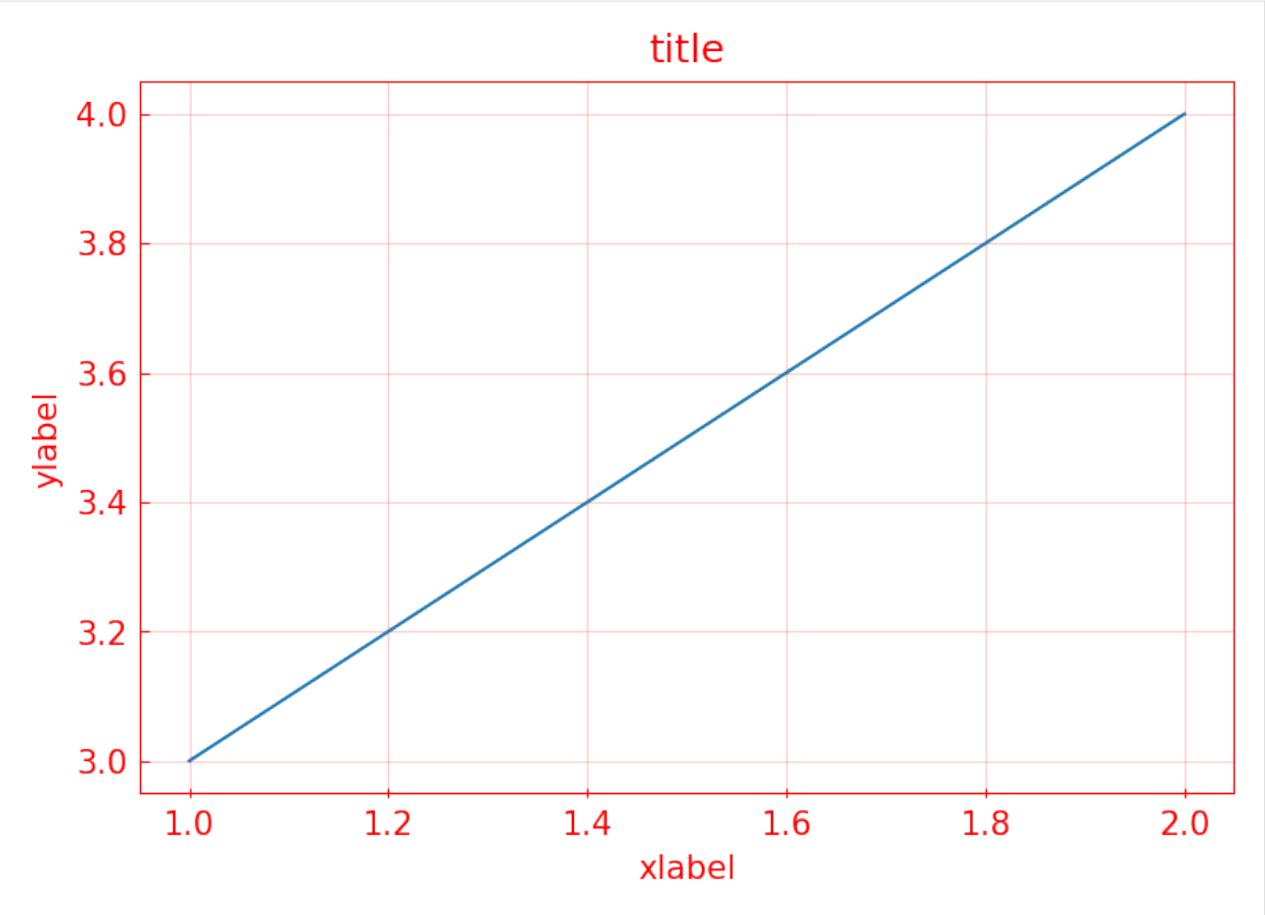
[5]:
```
plt.rcParams.update(axes.tick_direction(x="inout", y="in"))
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
ax.set_xlabel("xlabel")
ax.set_ylabel("ylabel")
ax.set_title("title")
plt.show()
```



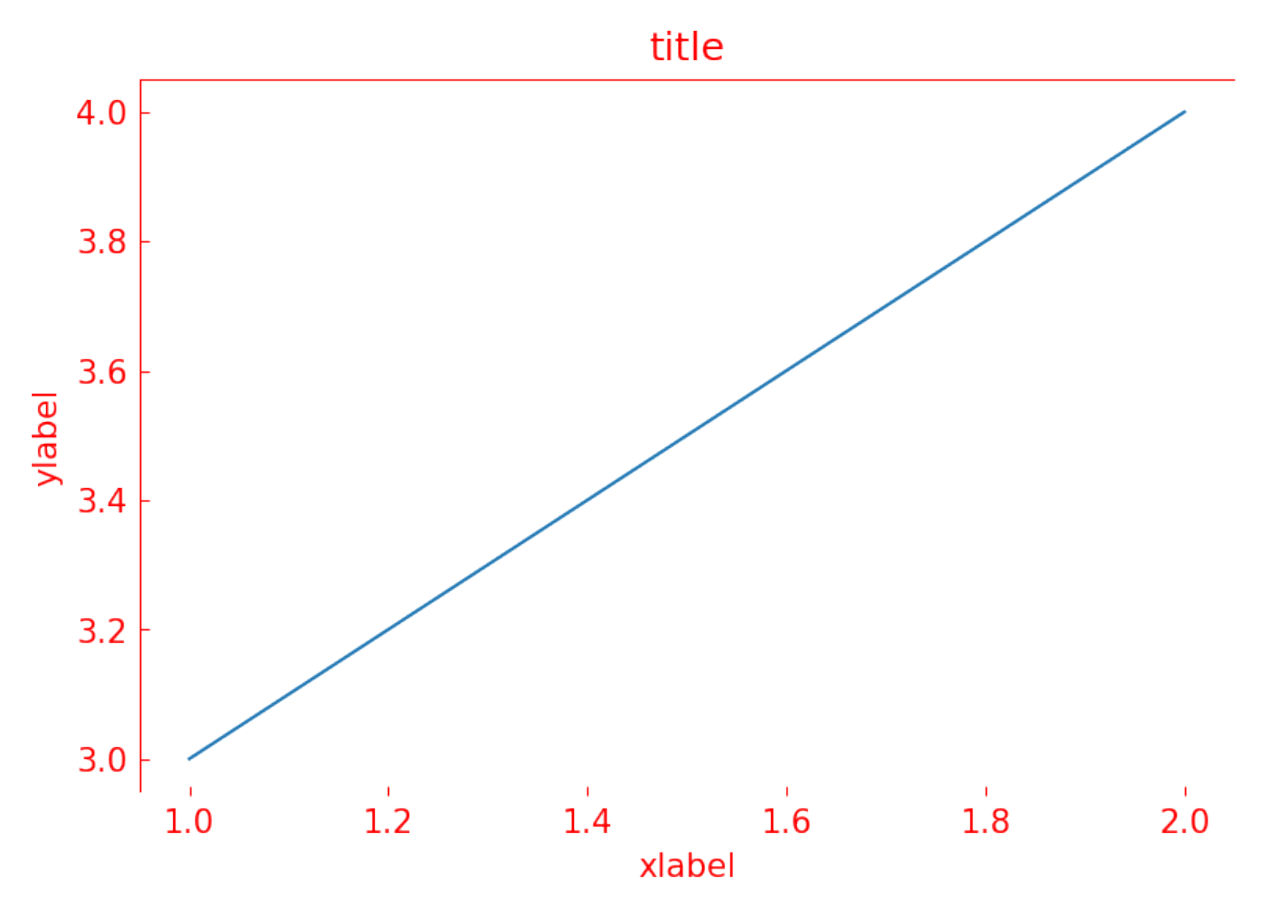
[6]:
```
plt.rcParams.update(axes.color(base="red"))
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0], label="ABC")
ax.set_xlabel("xlabel")
ax.set_ylabel("ylabel")
ax.set_title("title")
plt.grid()
plt.show()
```



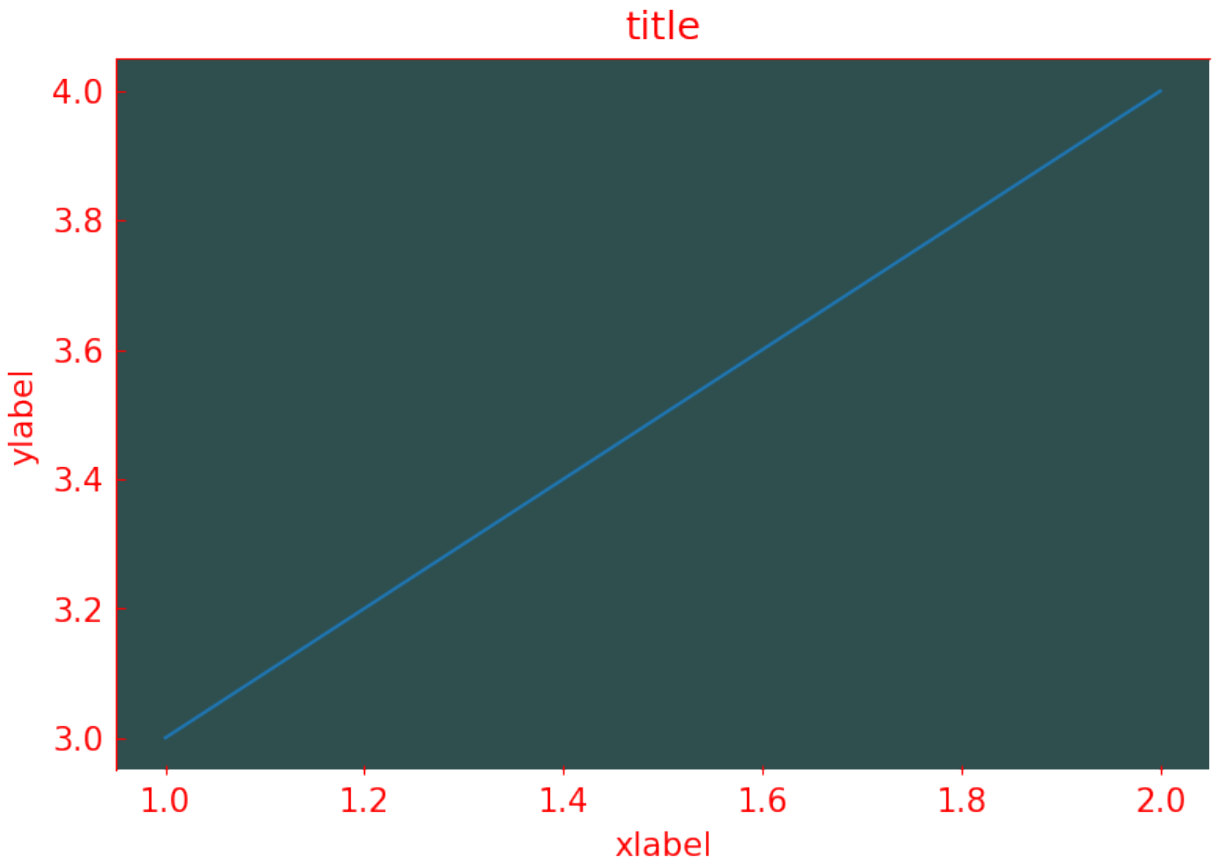
[7]:
```
plt.rcParams.update(axes.grid())
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0], label="ABC")
ax.set_xlabel("xlabel")
ax.set_ylabel("ylabel")
ax.set_title("title")
plt.grid()
plt.show()
```



[8]:
```
plt.rcParams.update(axes.spines(bottom=False, right=False))
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
ax.set_xlabel("xlabel")
ax.set_ylabel("ylabel")
ax.set_title("title")
plt.show()
```



[9]:
```
plt.rcParams.update(axes.color(face="darkslategray", base="red"))
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
ax.set_xlabel("xlabel")
ax.set_ylabel("ylabel")
ax.set_title("title")
plt.show()
```



[ ]:

1.6 Convenient plot-configs with bundles

[1]:
```
import matplotlib.pyplot as plt

from tueplots import axes, bundles

# Increase the resolution of all the plots below
plt.rcParams.update({"figure.dpi": 150})
```

`tueplots` provides a few prepackaged bundles that can be plugged right into matplotlib's context manager.

[2]:
```
bundles.icml2022()
```

[2]:
```
{'text.usetex': True,
 'font.family': 'serif',
 'text.latex.preamble': '\\usepackage{times} ',
 'figure.figsize': (3.25, 2.0086104634371584),
 'figure.constrained_layout.use': True,
 'figure.autolayout': False,
```

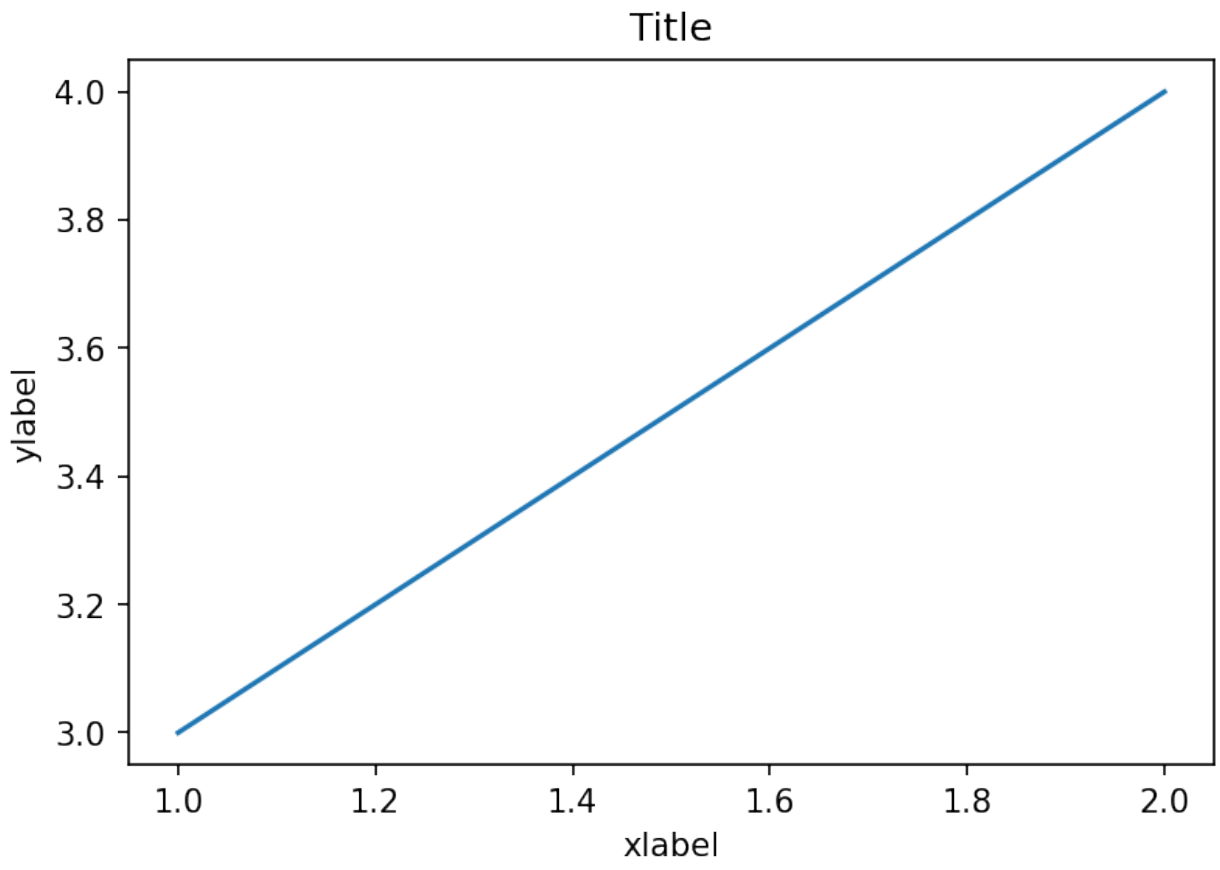
(continues on next page)

(continued from previous page)

```
 'savefig.bbox': 'tight',
 'savefig.pad_inches': 0.015,
 'font.size': 8,
 'axes.labelsize': 8,
 'legend.fontsize': 6,
 'xtick.labelsize': 6,
 'ytick.labelsize': 6,
 'axes.titlesize': 8}
```

Compare the default plots to the context plots below.

[3]:
```
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
ax.set_title("Title")
ax.set_xlabel("xlabel")
ax.set_ylabel("ylabel")
plt.show()
```

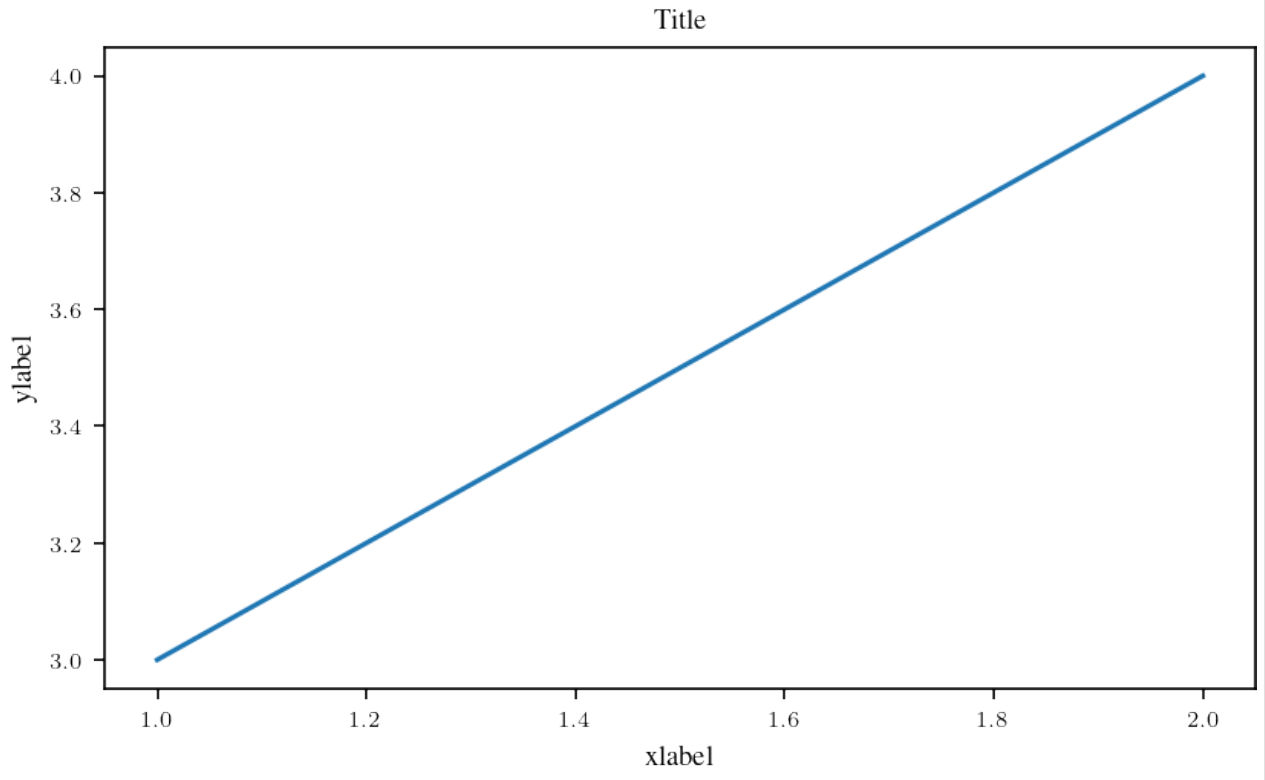


[4]:
```
with plt.rc_context(bundles.neurips2021()):
    fig, ax = plt.subplots()
    ax.plot([1.0, 2.0], [3.0, 4.0])
    ax.set_title("Title")
    ax.set_xlabel("xlabel")
```

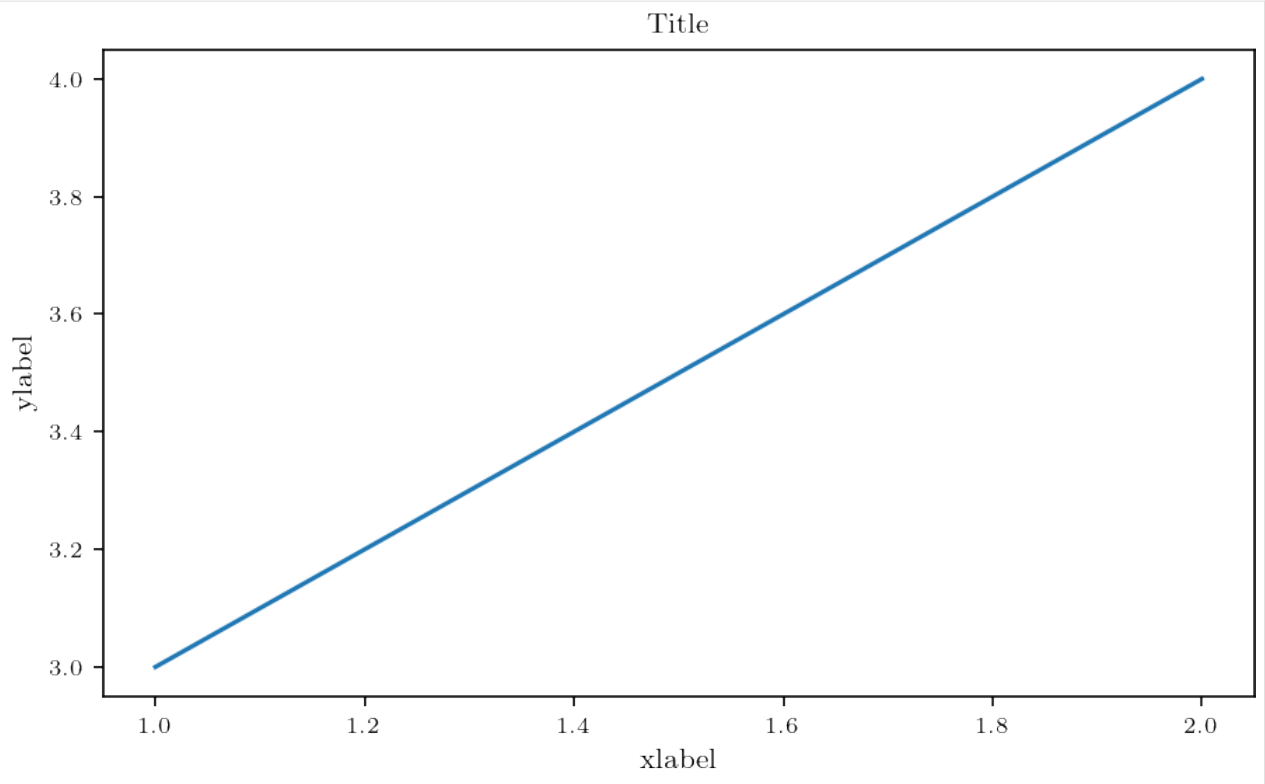
(continues on next page)

(continued from previous page)

```
    ax.set_ylabel("ylabel")
    plt.show()
```

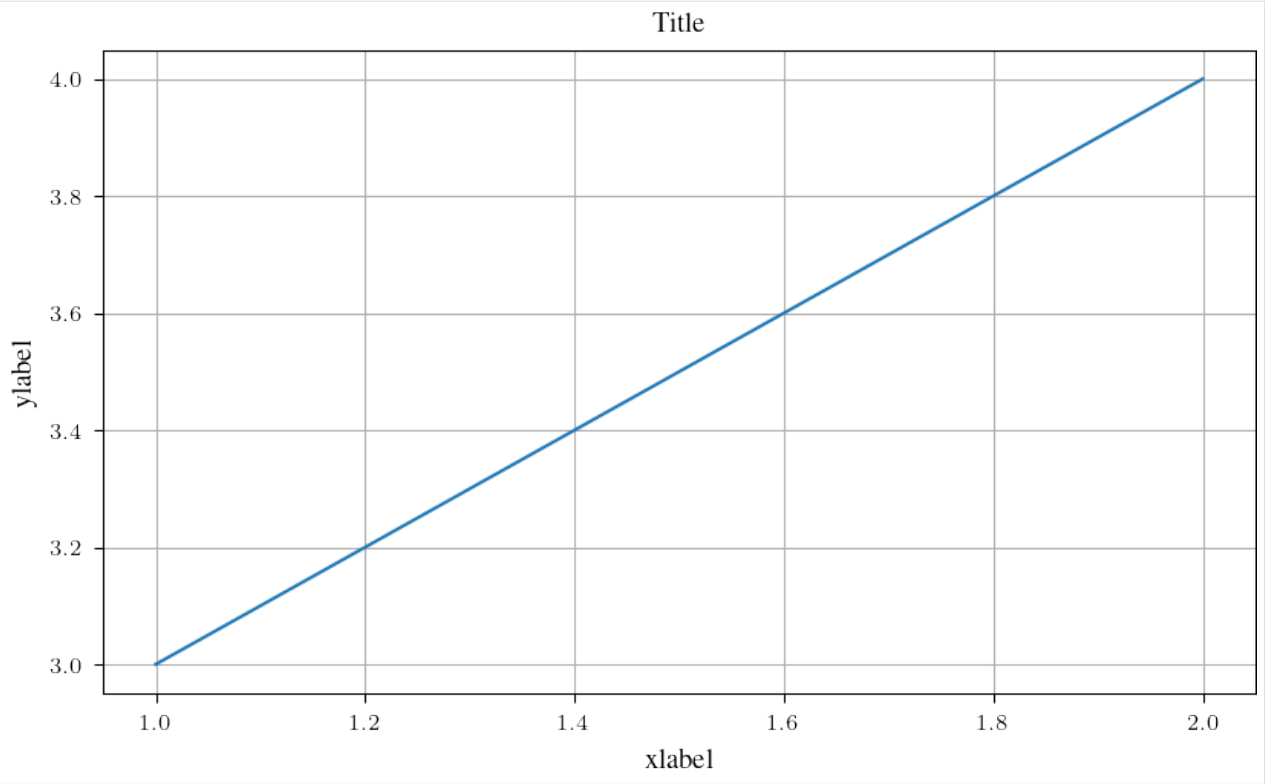


[5]:
```
with plt.rc_context(bundles.jmlr2001(family="serif")):
    fig, ax = plt.subplots()
    ax.plot([1.0, 2.0], [3.0, 4.0])
    ax.set_title("Title")
    ax.set_xlabel("xlabel")
    ax.set_ylabel("ylabel")
    plt.show()
```

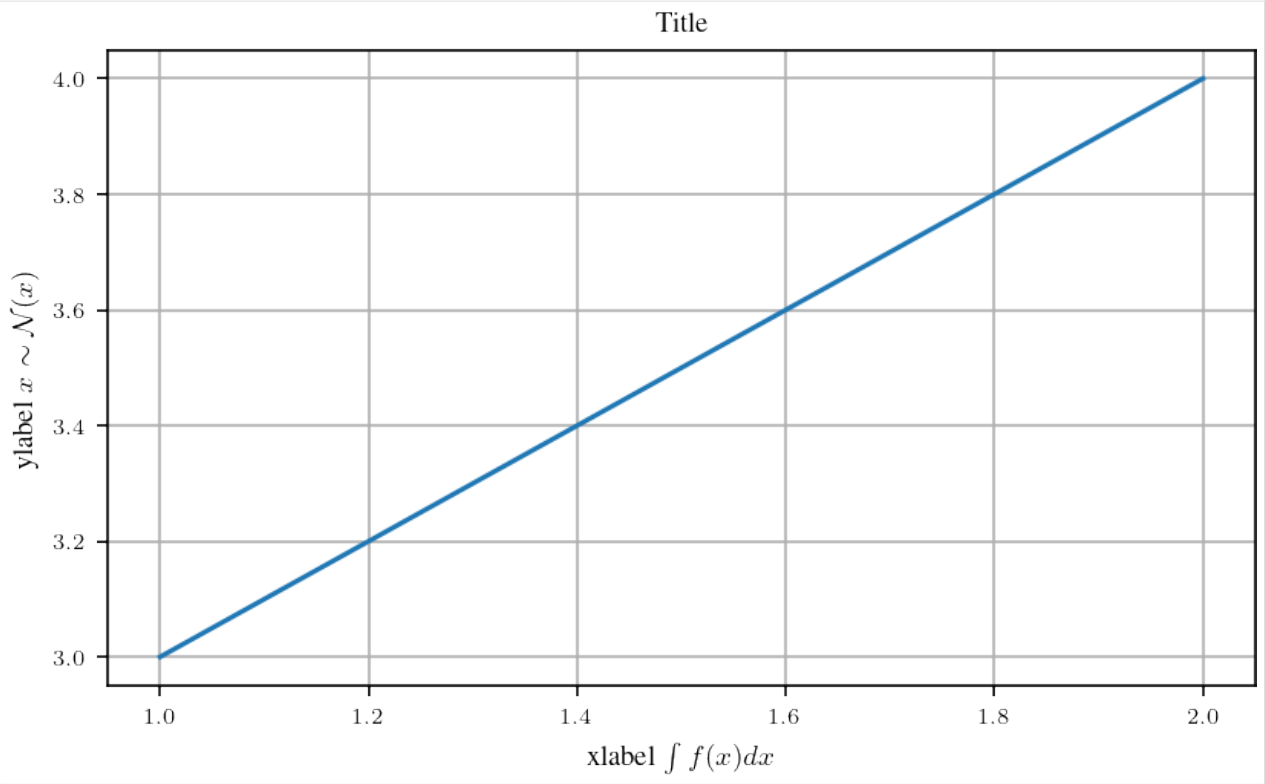


To get some (subjective) default behaviour, combine the bundles with `axes.lines()` (which is highly customisable, but has some opinionated default arguments).

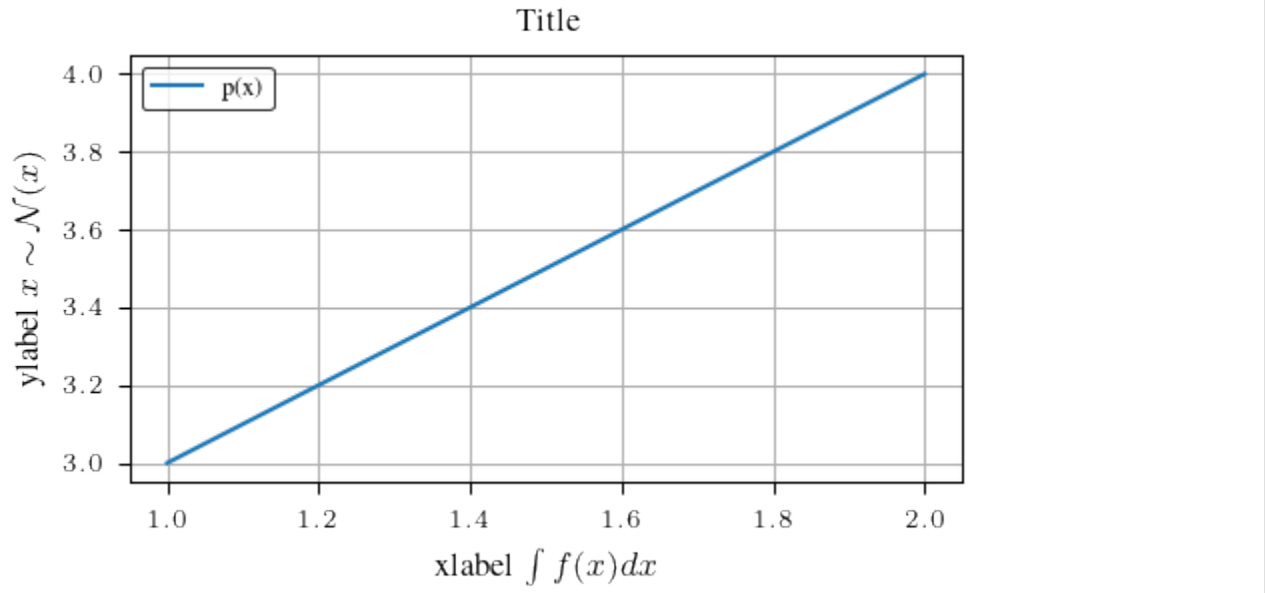
[6]:
```
with plt.rc_context({**bundles.neurips2021(), **axes.lines()}):
    fig, ax = plt.subplots()
    ax.plot([1.0, 2.0], [3.0, 4.0])
    ax.set_title("Title")
    ax.set_xlabel("xlabel")
    ax.set_ylabel("ylabel")
    plt.grid()
    plt.show()
```



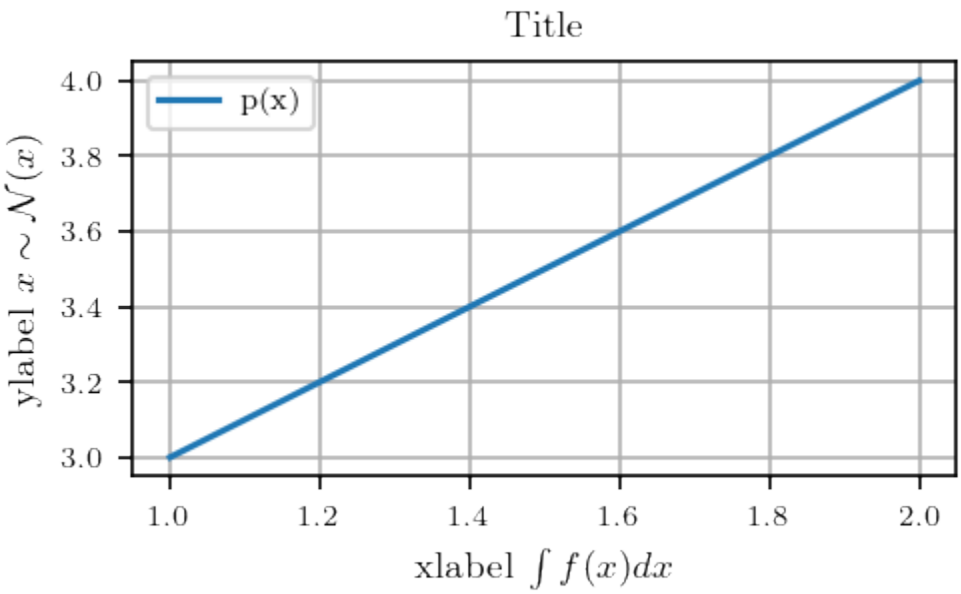
[7]:
```
with plt.rc_context(bundles.neurips2021(usetex=True, family="serif")):
    fig, ax = plt.subplots()
    ax.plot([1.0, 2.0], [3.0, 4.0])
    ax.set_title("Title")
    ax.set_xlabel("xlabel $\int f(x) dx$")
    ax.set_ylabel("ylabel $x \sim \mathcal{N}(x)$")
    plt.grid()
    plt.show()
```



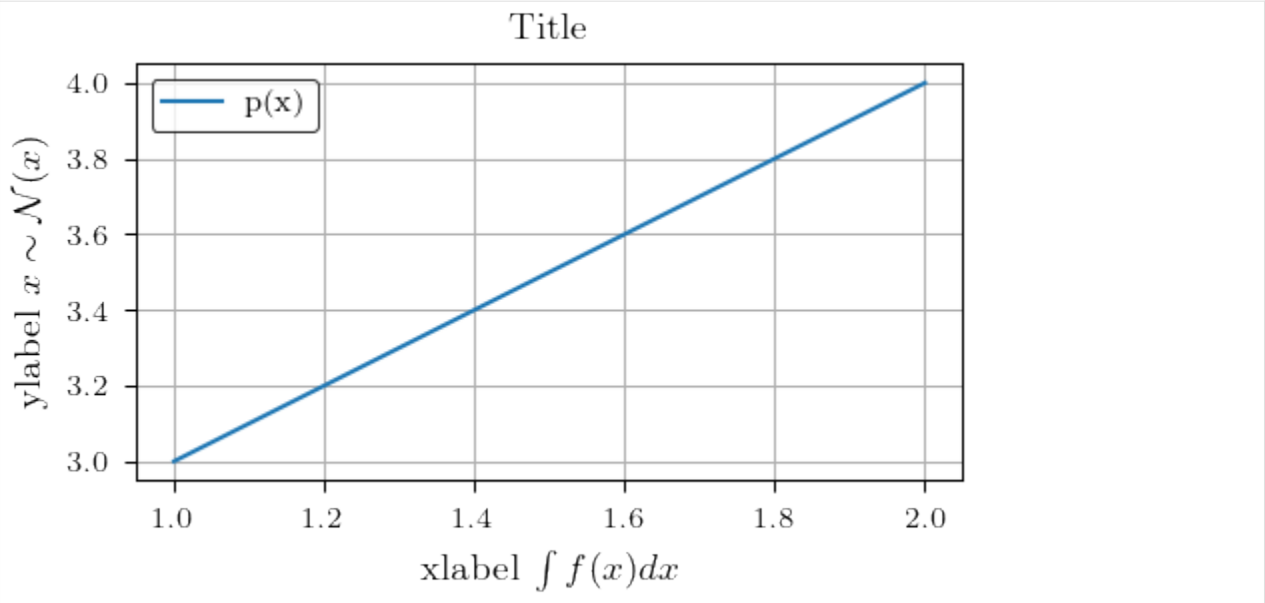
[8]:
```
with plt.rc_context({**bundles.icml2022(), **axes.lines()}):
    fig, ax = plt.subplots()
    ax.plot([1.0, 2.0], [3.0, 4.0], label="p(x)")
    ax.set_title("Title")
    ax.set_xlabel("xlabel $\int f(x) dx$")
    ax.set_ylabel("ylabel $x \sim \mathcal{N}(x)$")
    plt.grid()
    plt.legend()
    plt.show()
```



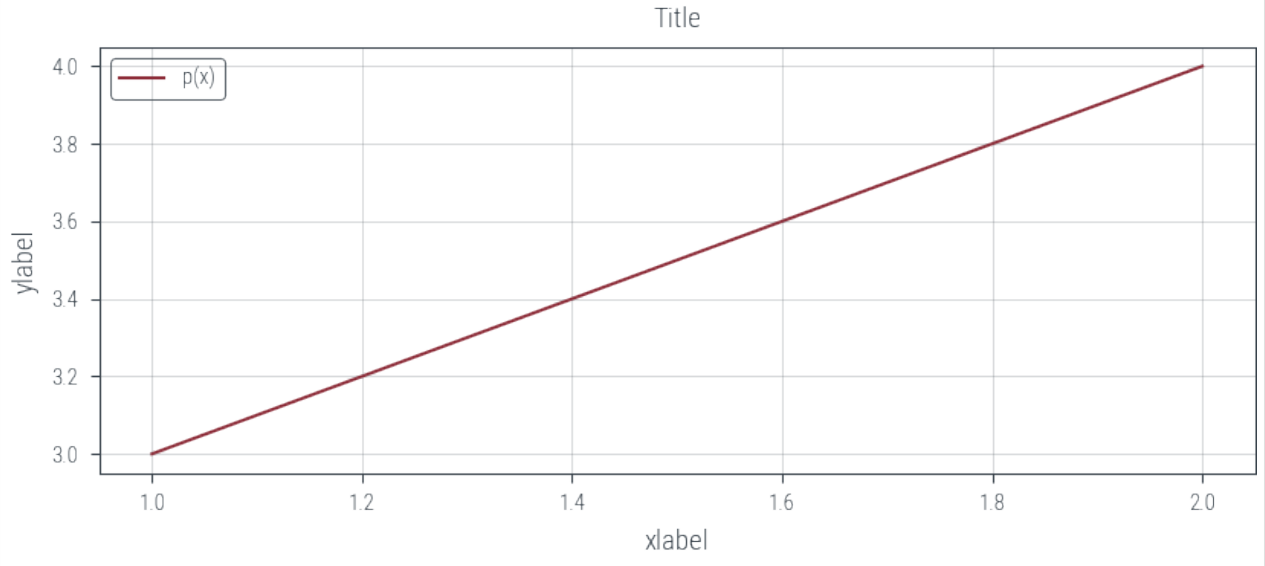
[9]:
```
with plt.rc_context(bundles.aistats2022()):
    fig, ax = plt.subplots()
    ax.plot([1.0, 2.0], [3.0, 4.0], label="p(x)")
    ax.set_title("Title")
    ax.set_xlabel("xlabel $\int f(x) dx$")
    ax.set_ylabel("ylabel $x \sim \mathcal{N}(x)$")
    plt.grid()
    plt.legend()
    plt.show()
```



[10]:
```
with plt.rc_context({**bundles.aistats2022(family="serif"), **axes.lines()}):
    fig, ax = plt.subplots()
    ax.plot([1.0, 2.0], [3.0, 4.0], label="p(x)")
    ax.set_title("Title")
    ax.set_xlabel("xlabel $\int f(x) dx$")
    ax.set_ylabel("ylabel $x \sim \mathcal{N}(x)$")
    plt.grid()
    plt.legend()
    plt.show()
```



[11]:
```
with plt.rc_context(bundles.beamer_moml()):
    fig, ax = plt.subplots()
    ax.plot([1.0, 2.0], [3.0, 4.0], label="p(x)")
    ax.legend()
    ax.set_title("Title")
    ax.set_xlabel("xlabel")
    ax.set_ylabel("ylabel")
    ax.grid()
    plt.show()
```

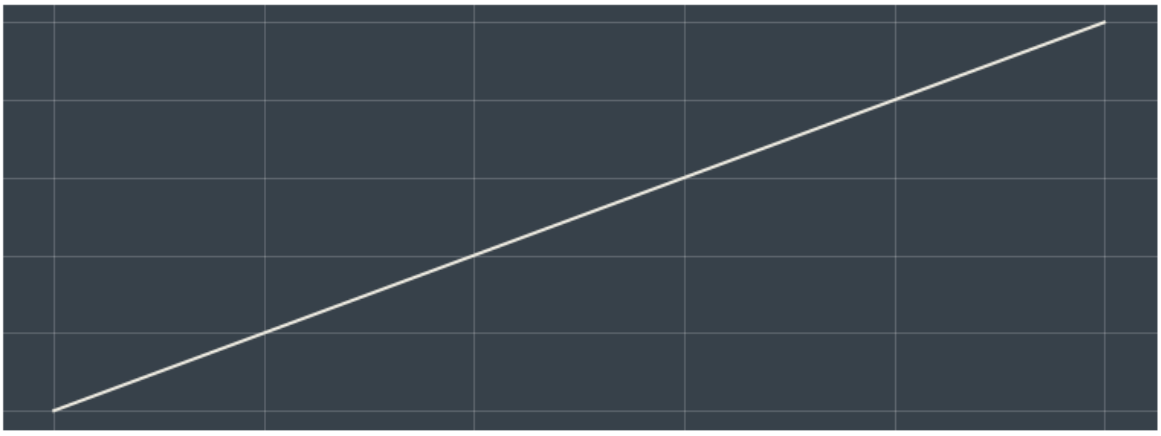


[12]:
```
with plt.rc_context(bundles.beamer_moml_dark_bg()):
    fig, ax = plt.subplots()
    ax.plot([1.0, 2.0], [3.0, 4.0])
    ax.set_title("Title")
    ax.set_xlabel("xlabel")
```

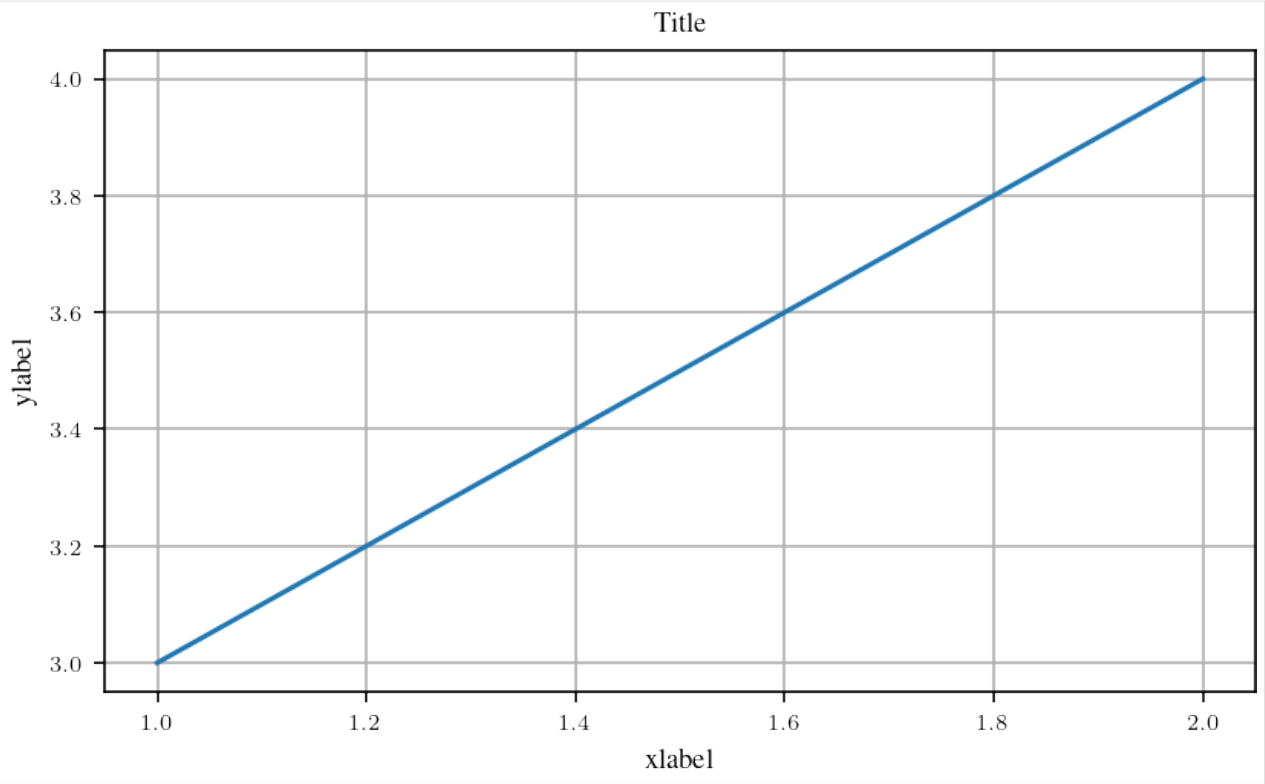
(continues on next page)

(continued from previous page)

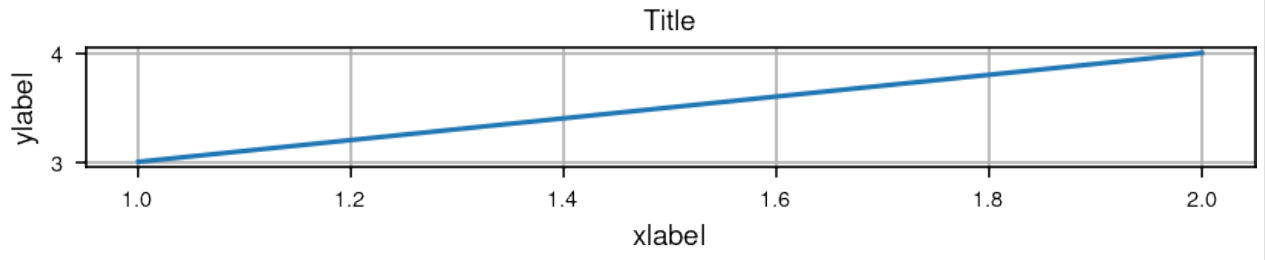
```
    ax.set_ylabel("ylabel")
    ax.grid()
    plt.show()
```



[13]:
```
with plt.rc_context(bundles.iclr2023()):
    fig, ax = plt.subplots()
    ax.plot([1.0, 2.0], [3.0, 4.0])
    ax.set_title("Title")
    ax.set_xlabel("xlabel")
    ax.set_ylabel("ylabel")
    ax.grid()
    plt.show()
```



[14]:
```
with plt.rc_context(
    bundles.iclr2023(usetex=True, nrows=1, ncols=3, family="smallcaps")
):
    fig, ax = plt.subplots()
    ax.plot([1.0, 2.0], [3.0, 4.0])
    ax.set_title("Title")
    ax.set_xlabel("xlabel")
    ax.set_ylabel("ylabel")
    ax.grid()
    plt.show()
```



1.7 Creating color-cycles

[1]:
```
import matplotlib.pyplot as plt
import numpy as np

from tueplots import cycler
from tueplots.constants import markers
from tueplots.constants.color import palettes

# Increase the resolution of all the plots below
plt.rcParams.update({"figure.dpi": 150})
```

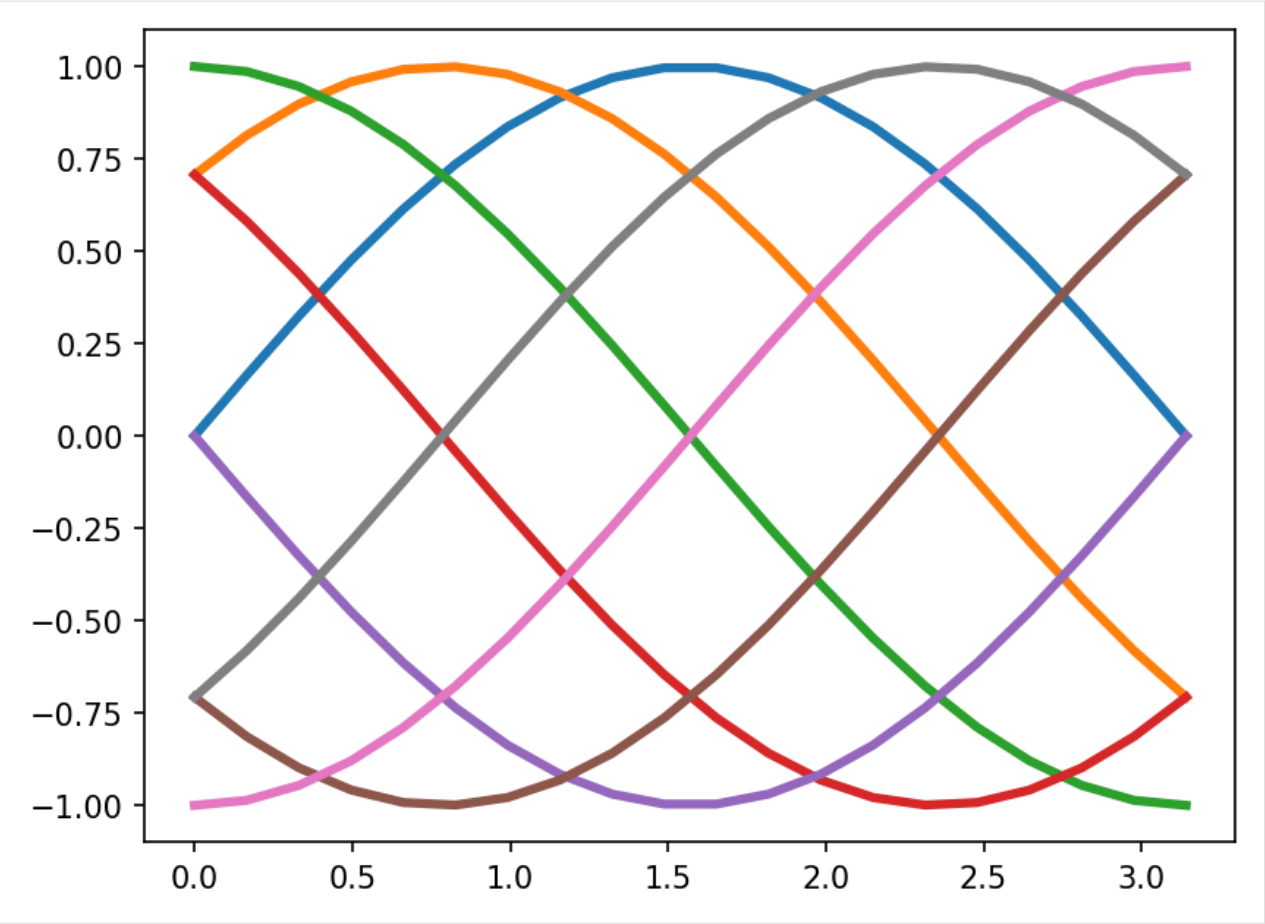
`cycler` objects define the default color choices in matplotlib (e.g., blue-orange-red-...). Like most other settings provided in `tueplots`, the outputs of `cycler` are directly compatible with `plt.rcParams.update()` (which is different to the `cycler` object in matplotlib in that it wraps them into a dictionary).

We can control the cyclers through the constants in `tueplots.constants`. To see this, let us generate some lines to be plotted. (Setup taken from https://matplotlib.org/stable/tutorials/intermediate/color_cycle.html).

[2]:
```
x = np.linspace(0, np.pi, 20)
offsets = np.linspace(0, 2 * np.pi, 8, endpoint=False)
yy = [np.sin(x + phi) for phi in offsets]
```

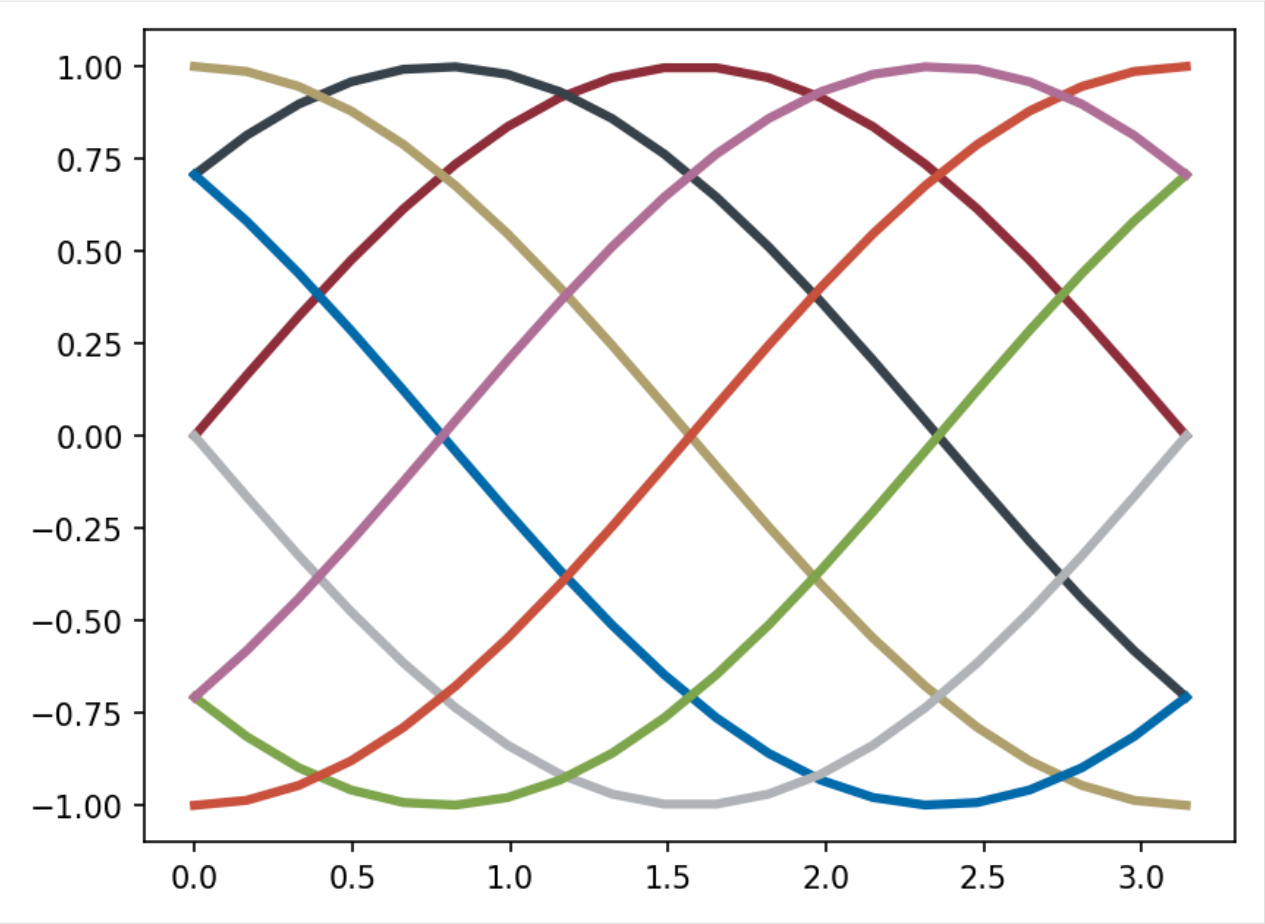
The following are the default colors:

[3]:
```
for y in yy:
    plt.plot(x, y, linewidth=3)
plt.show()
```

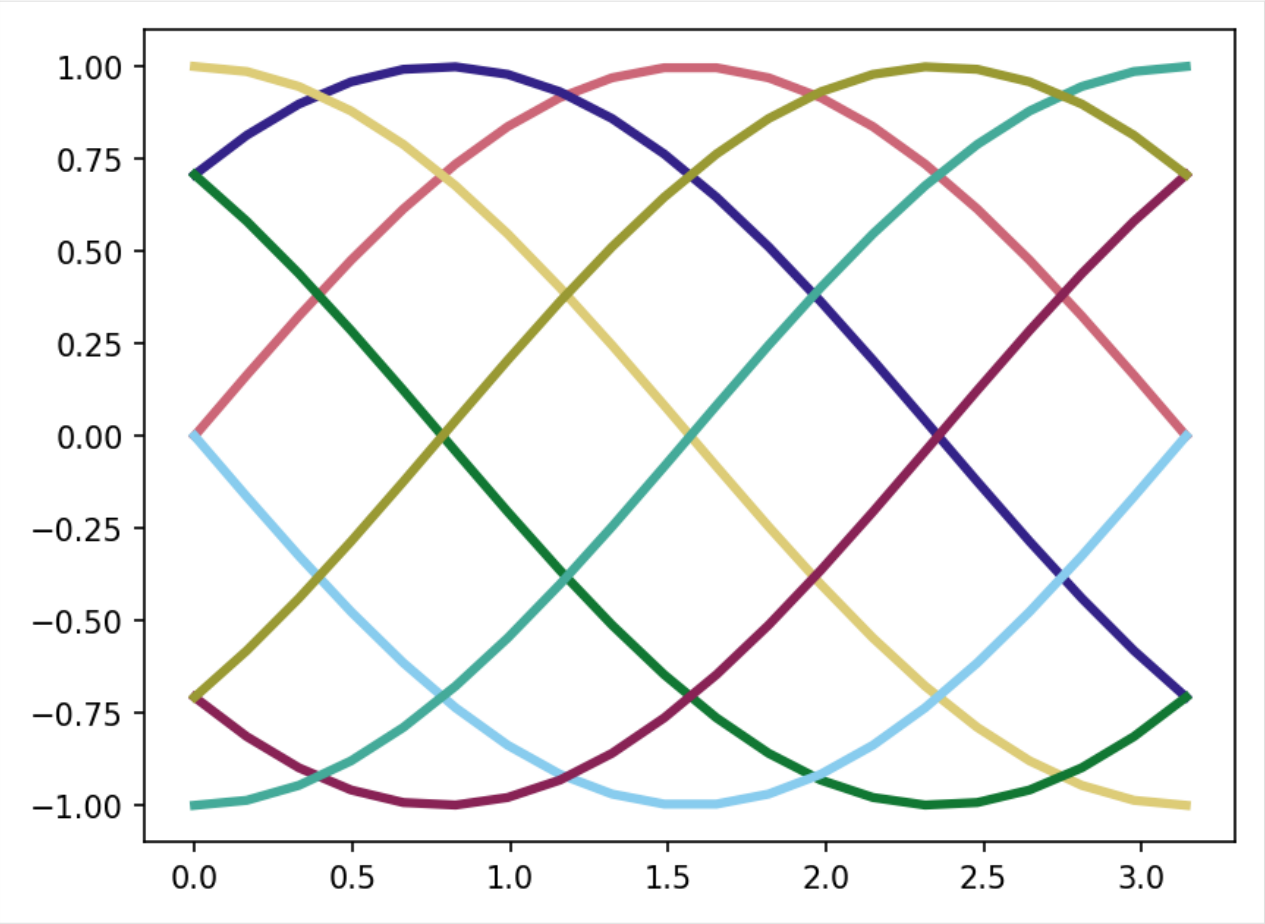


Through the dictionaries provided by `tueplots`, we can change the default color behaviour as follows.

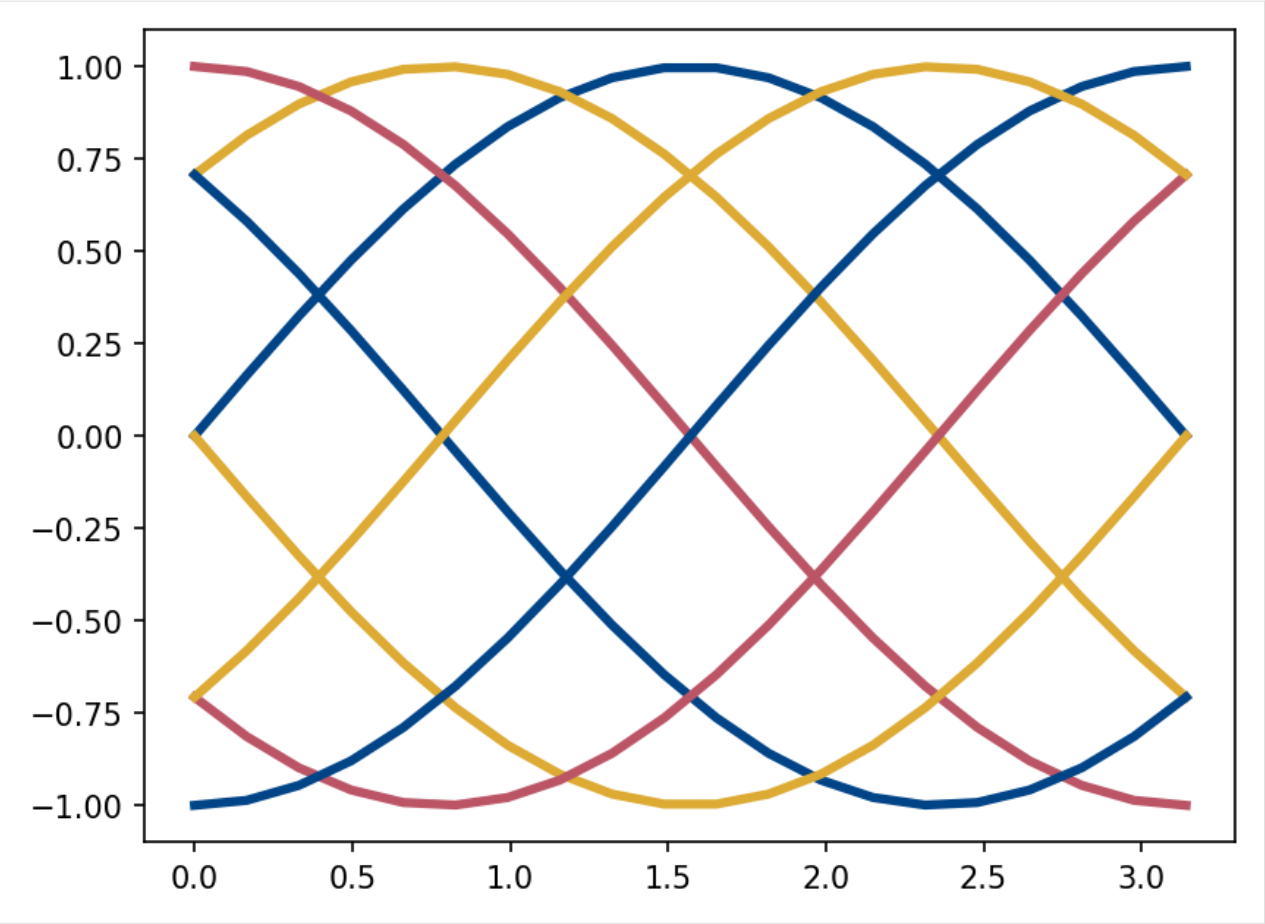
[4]:
```
plt.rcParams.update(cycler.cycler(color=palettes.tue_plot))
for y in yy:
    plt.plot(x, y, linewidth=3)
plt.show()
```



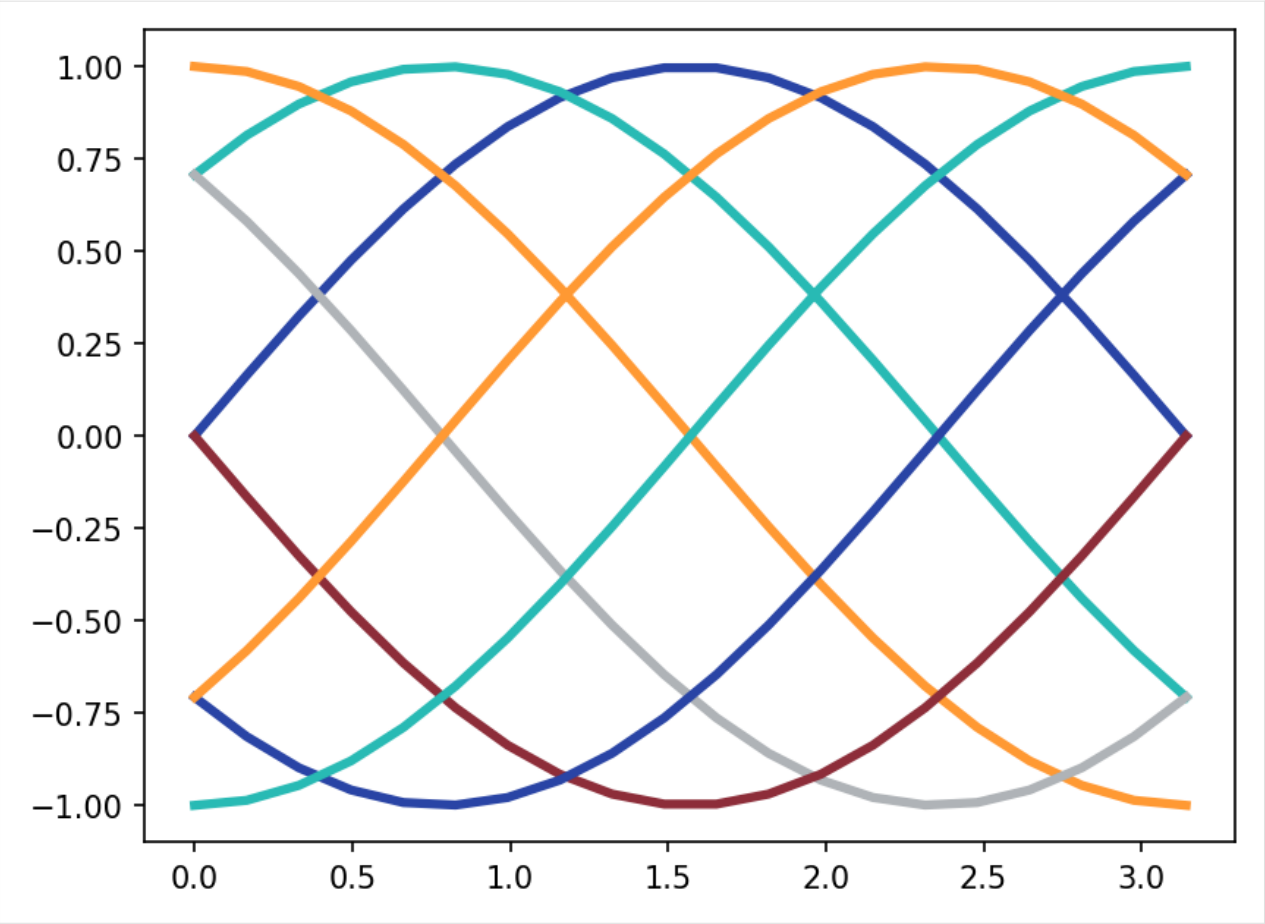
[6]:
```
plt.rcParams.update(cycler.cycler(color=palettes.paultol_muted))
for y in yy:
    plt.plot(x, y, linewidth=3)
plt.show()
```



[7]:
```
plt.rcParams.update(cycler.cycler(color=palettes.paultol_high_contrast))
for y in yy:
    plt.plot(x, y, linewidth=3)
plt.show()
```

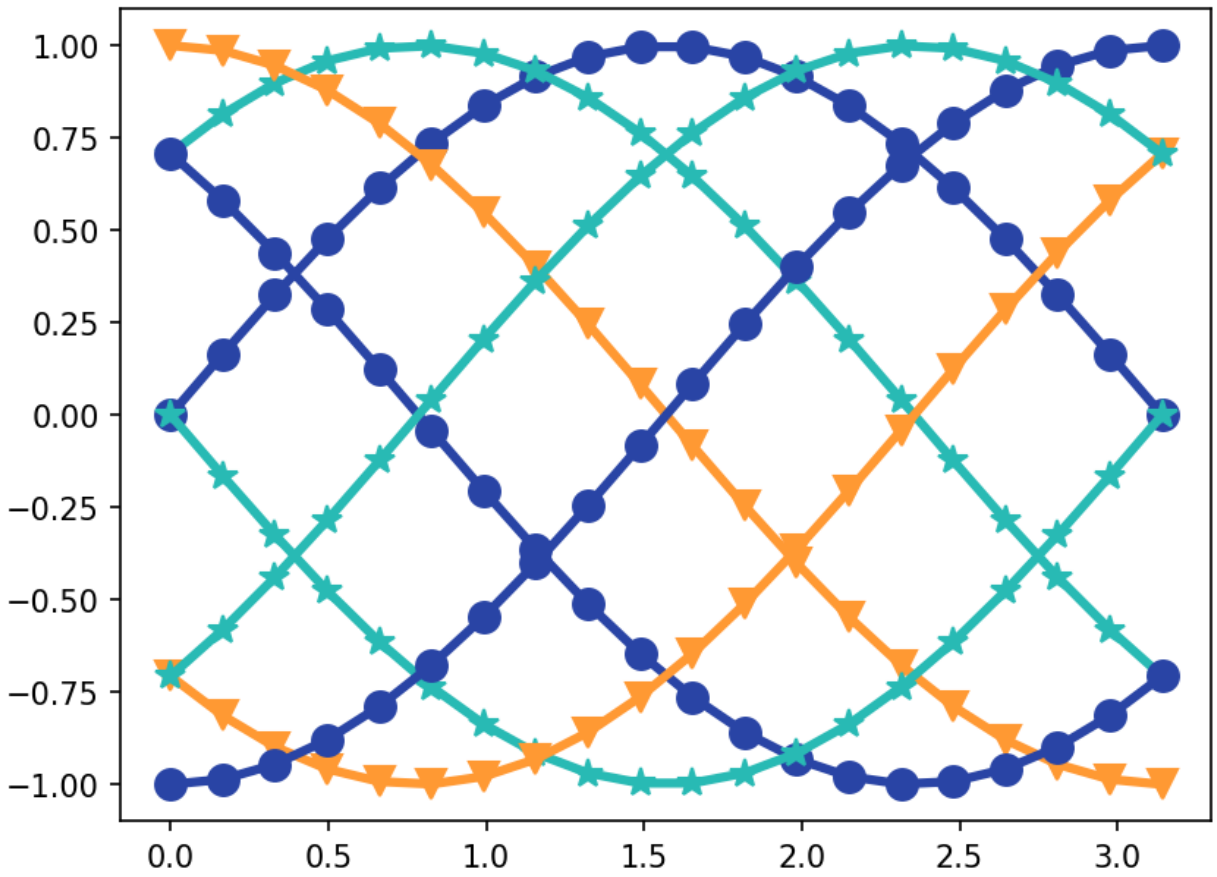


[8]:
```
plt.rcParams.update(cycler.cycler(color=palettes.pn))
for y in yy:
    plt.plot(x, y, linewidth=3)
plt.show()
```

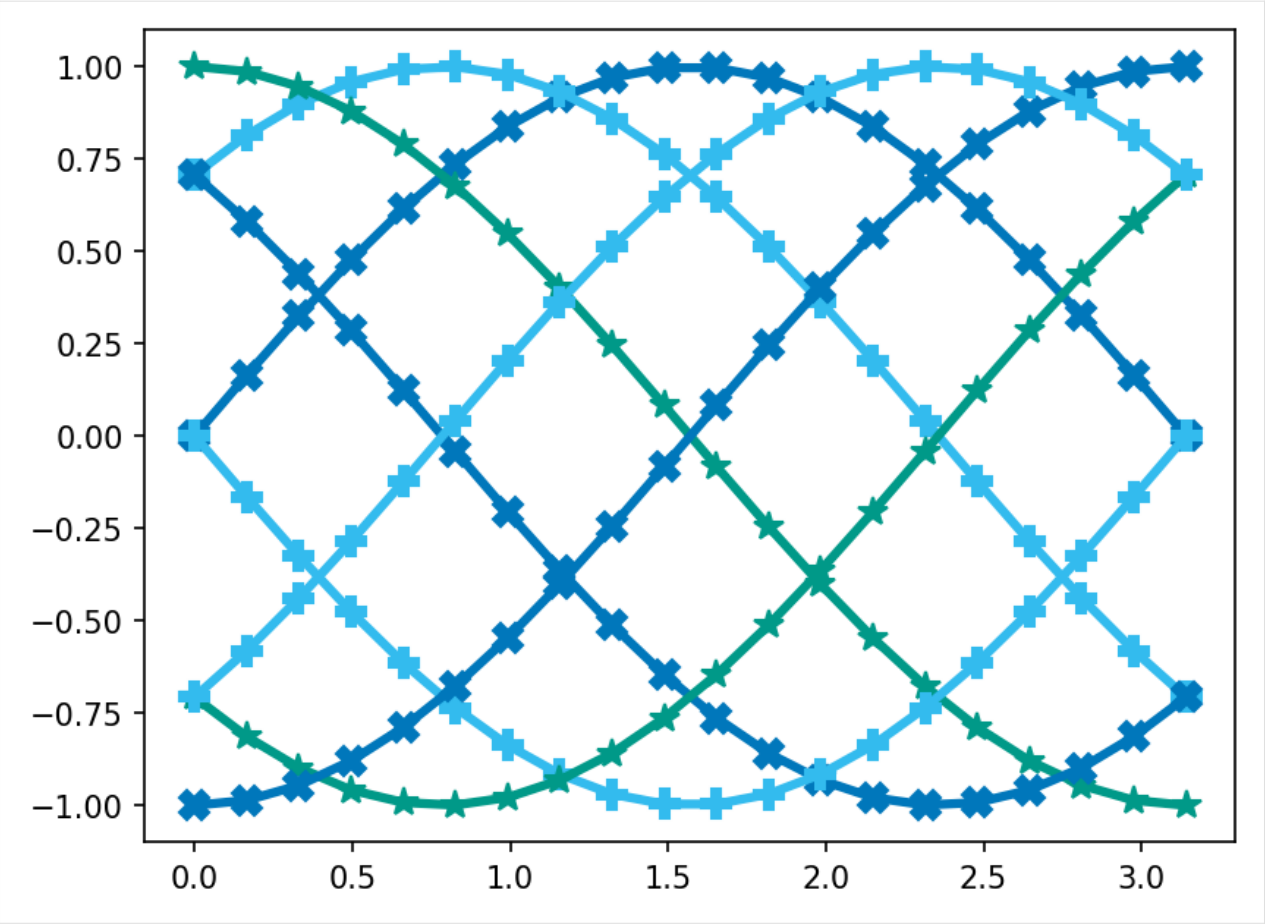


We can also cycle linestyles and markers.

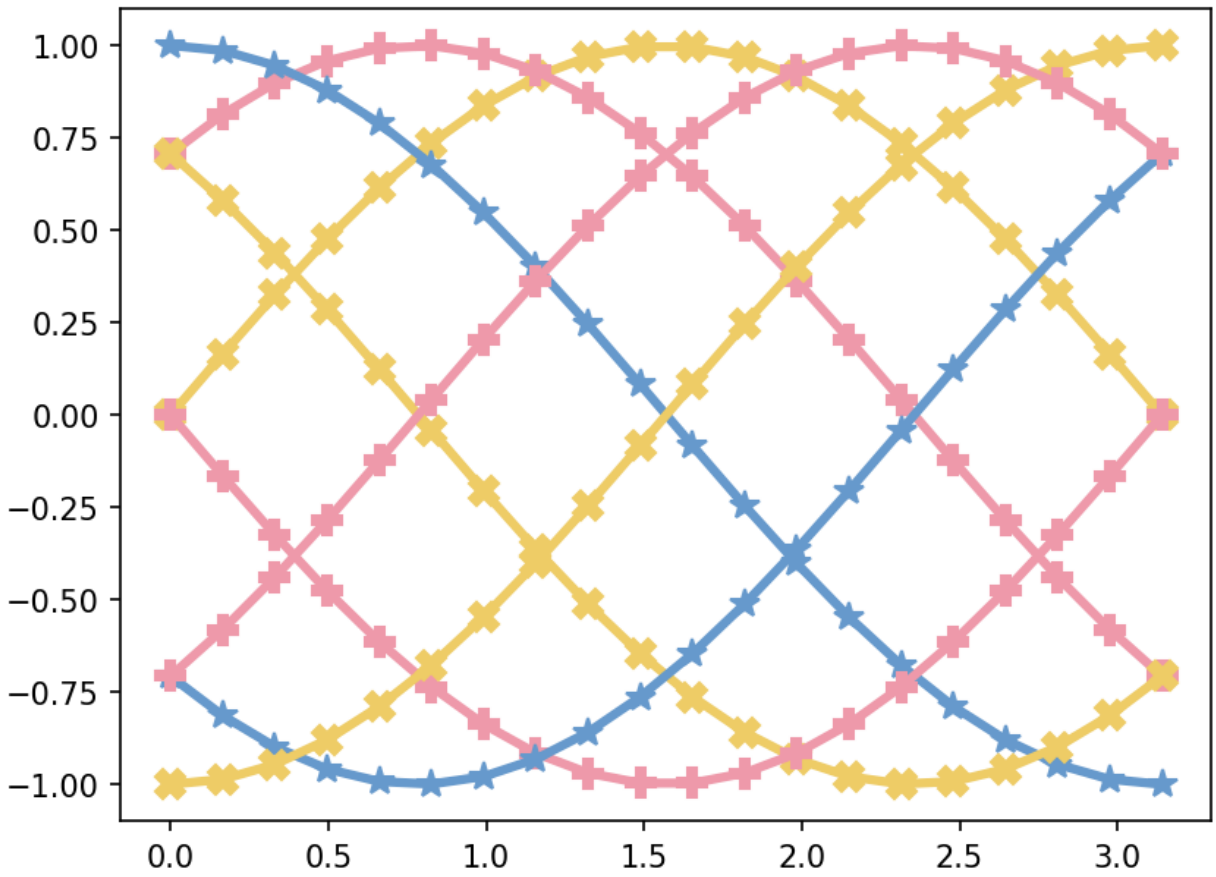
[9]:
```
plt.rcParams.update(cycler.cycler(color=palettes.pn[:3], marker=markers.o_sized[:3]))
for y in yy:
    plt.plot(x, y, linewidth=3, markersize=10)
plt.show()
```



[10]:
```
plt.rcParams.update(
    cycler.cycler(color=palettes.paultol_vibrant[:3], marker=markers.x_like_bold[:3])
)
for y in yy:
    plt.plot(x, y, linewidth=3, markersize=10)
plt.show()
```



[11]:
```
plt.rcParams.update(
    cycler.cycler(
        color=palettes.paultol_medium_contrast[:3], marker=markers.x_like_bold[:3]
    )
)
for y in yy:
    plt.plot(x, y, linewidth=3, markersize=10)
plt.show()
```



[ ]:

1.8 Appropriate figure sizes

[1]:
```
import matplotlib.pyplot as plt

from tueplots import figsizes

# Increase the resolution of all the plots below
plt.rcParams.update({"figure.dpi": 150})
```

Figure sizes are tuples. They describe the figure sizes in `inches`, just like what matplotlib expects.

Outputs of `figsize` functions are dictionaries that match `rcParams`.

[2]:
```
icml_size = figsizes.icml2022_full()
icml_size
```

[2]:
```
{'figure.figsize': (6.75, 2.0858647120308955),
 'figure.constrained_layout.use': True,
 'figure.autolayout': False,
```

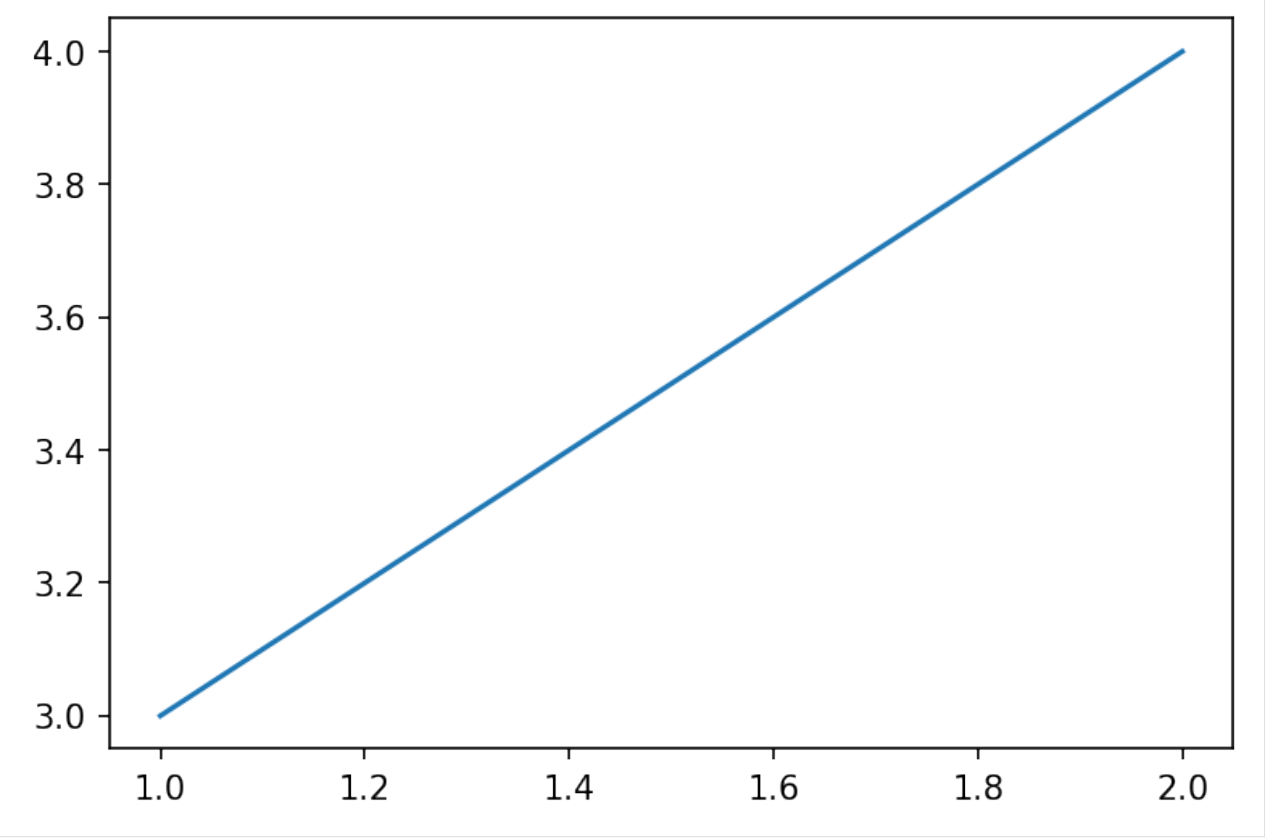
(continues on next page)

(continued from previous page)

```
 'savefig.bbox': 'tight',
 'savefig.pad_inches': 0.015}
```

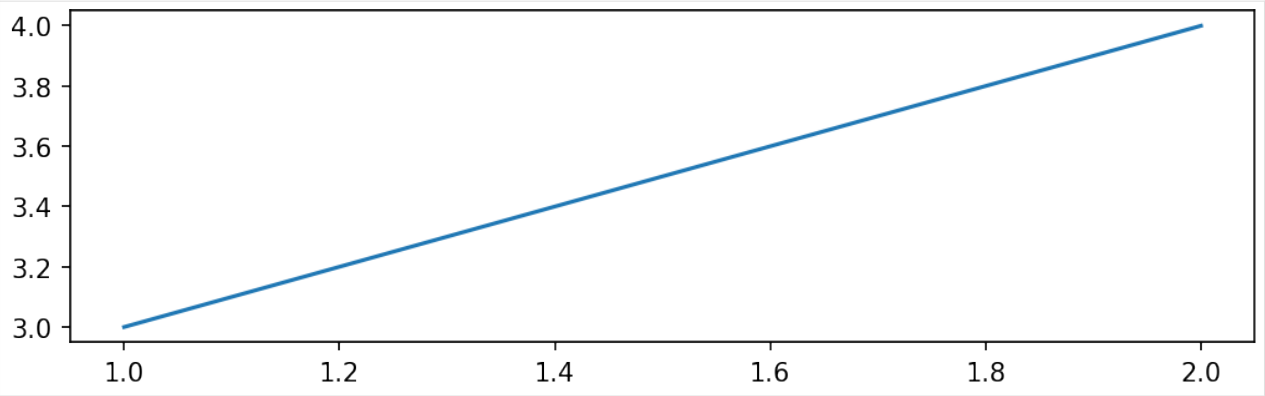
We can use them to make differently sized figures. The height-to-width ratio is (loosely) based on the golden ratio.

[3]:
```
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
plt.show()
```

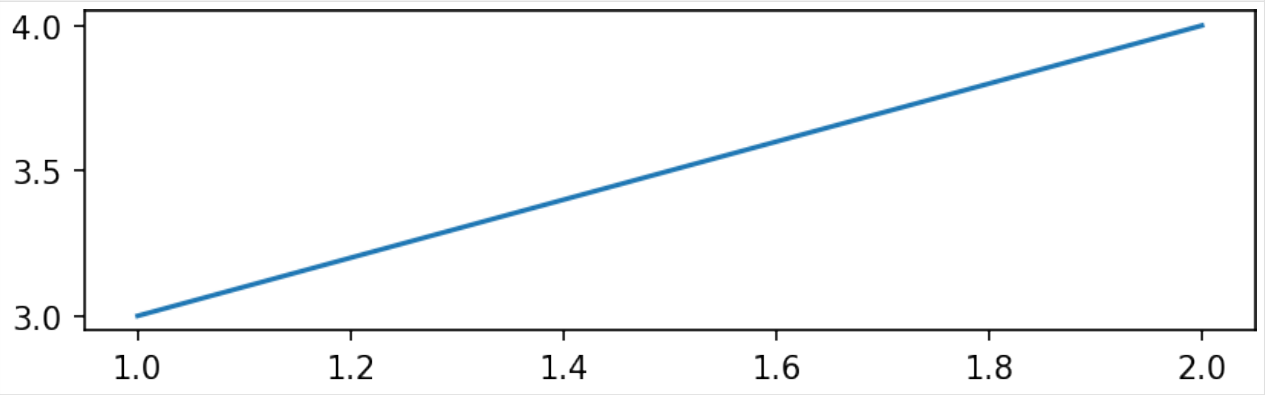


1.8.1 Figure sizes that match your latex template

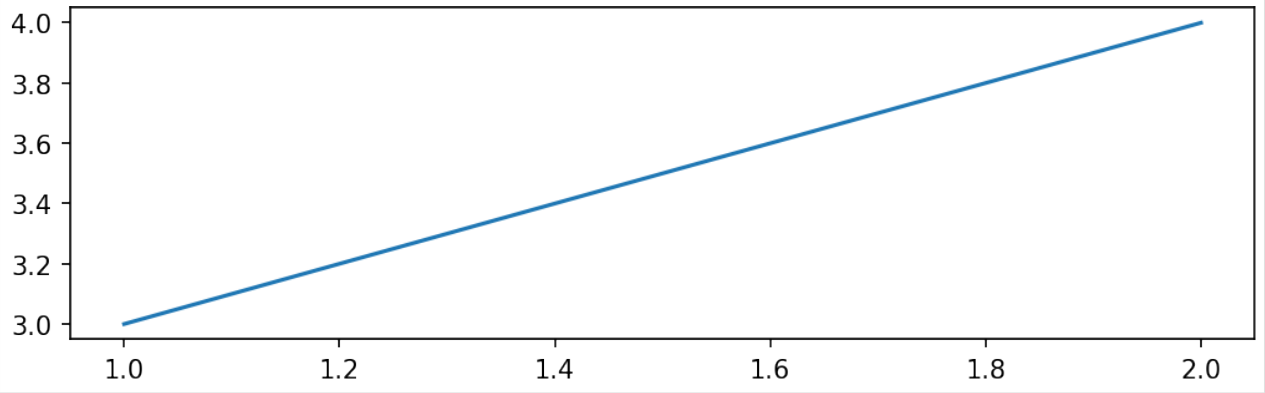
[4]:
```
plt.rcParams.update(figsizes.icml2022_full())

fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
plt.show()
```



[5]:
```
plt.rcParams.update(figsizes.neurips2021())

fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
plt.show()
```

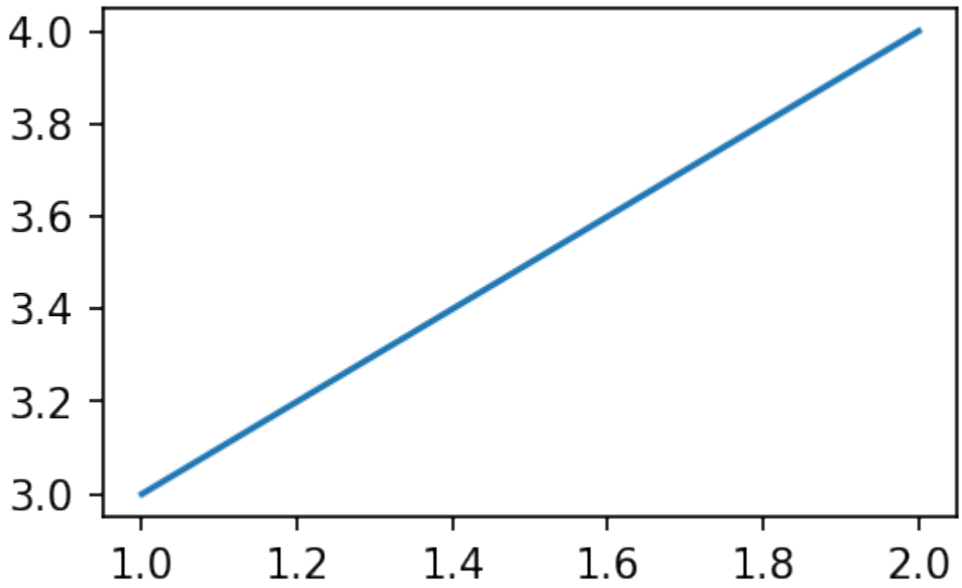


[6]:
```
plt.rcParams.update(figsizes.aistats2022_full())

fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
plt.show()
```

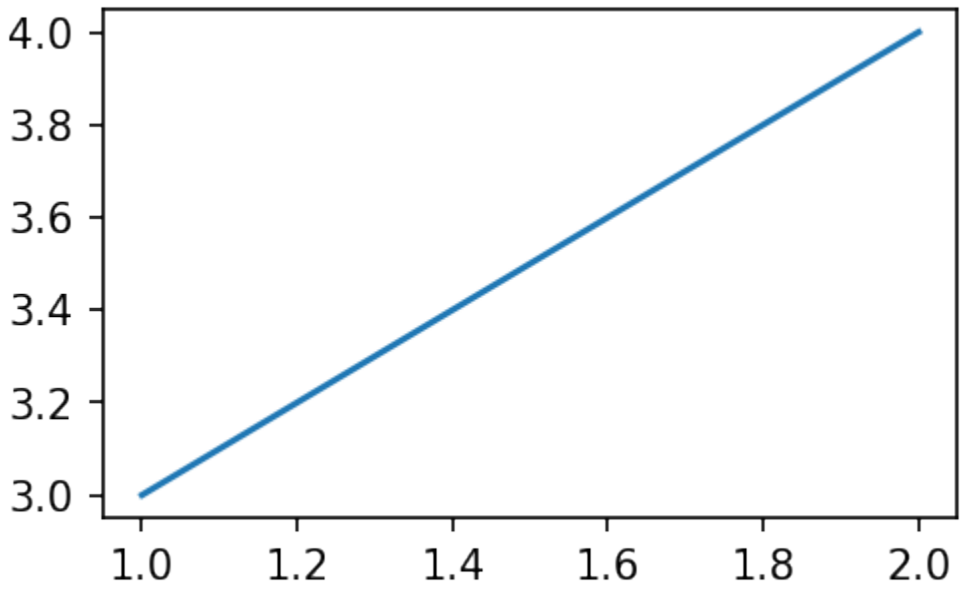


For double-column layouts such as ICML or AISTATS, there is also a single-column (i.e. half-width) version:

[7]:
```
plt.rcParams.update(figsizes.icml2022_half())
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
plt.show()
```



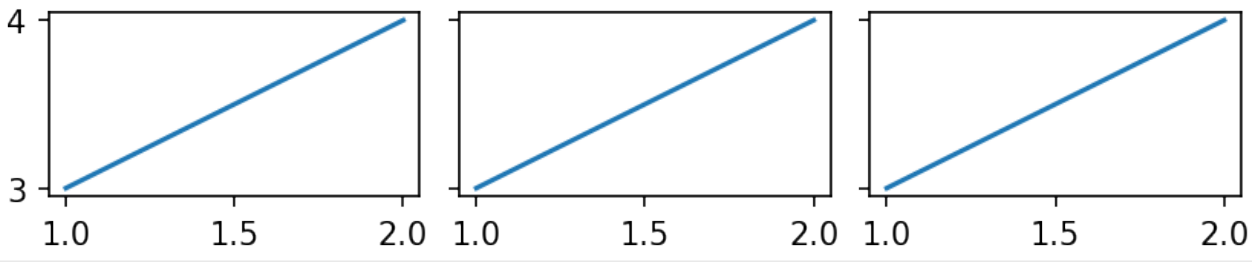
[8]:
```
plt.rcParams.update(figsizes.aistats2022_half())
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
plt.show()
```



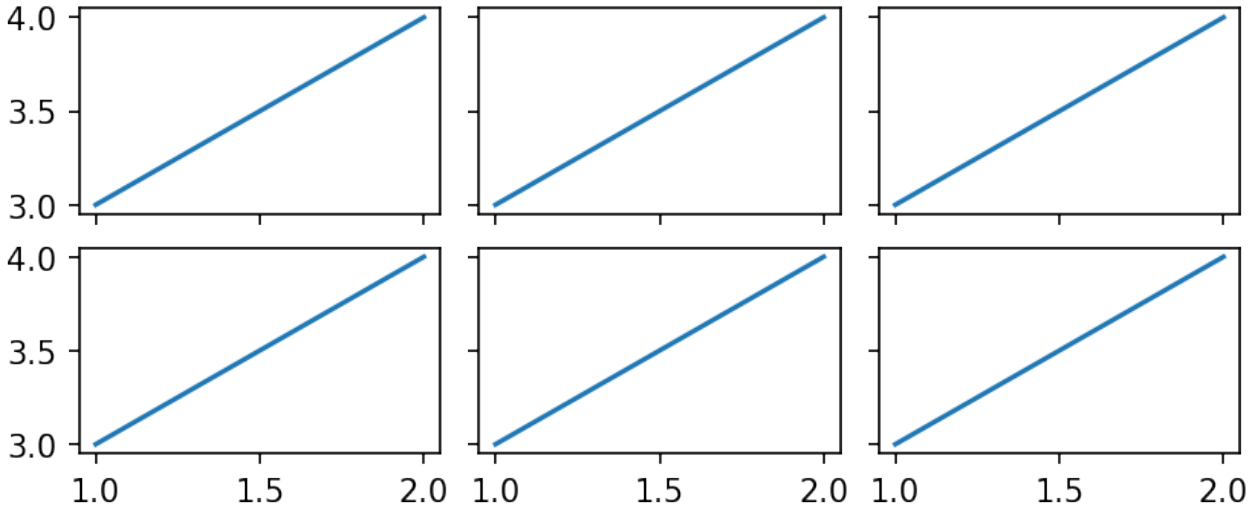
1.8.2 Figure sizes that match your subplot layouts

When working with `plt.subplots`, provide the `nrows` and `ncols` also to the figsize functions to get consistent subplot sizes by adjusting the overall figure height. Why? Because each subplot is fixed to a specific format (usually the golden ratio), and the figure width is commonly tied to the specific journal style. The remaining degree of freedom, the overall figure height, is adapted to make things look clean.

[9]:
```
plt.rcParams.update(figsizes.neurips2021(nrows=1, ncols=3))

fig, axes = plt.subplots(nrows=1, ncols=3, sharex=True, sharey=True)
for ax in axes.flatten():
    ax.plot([1.0, 2.0], [3.0, 4.0])
plt.show()
```

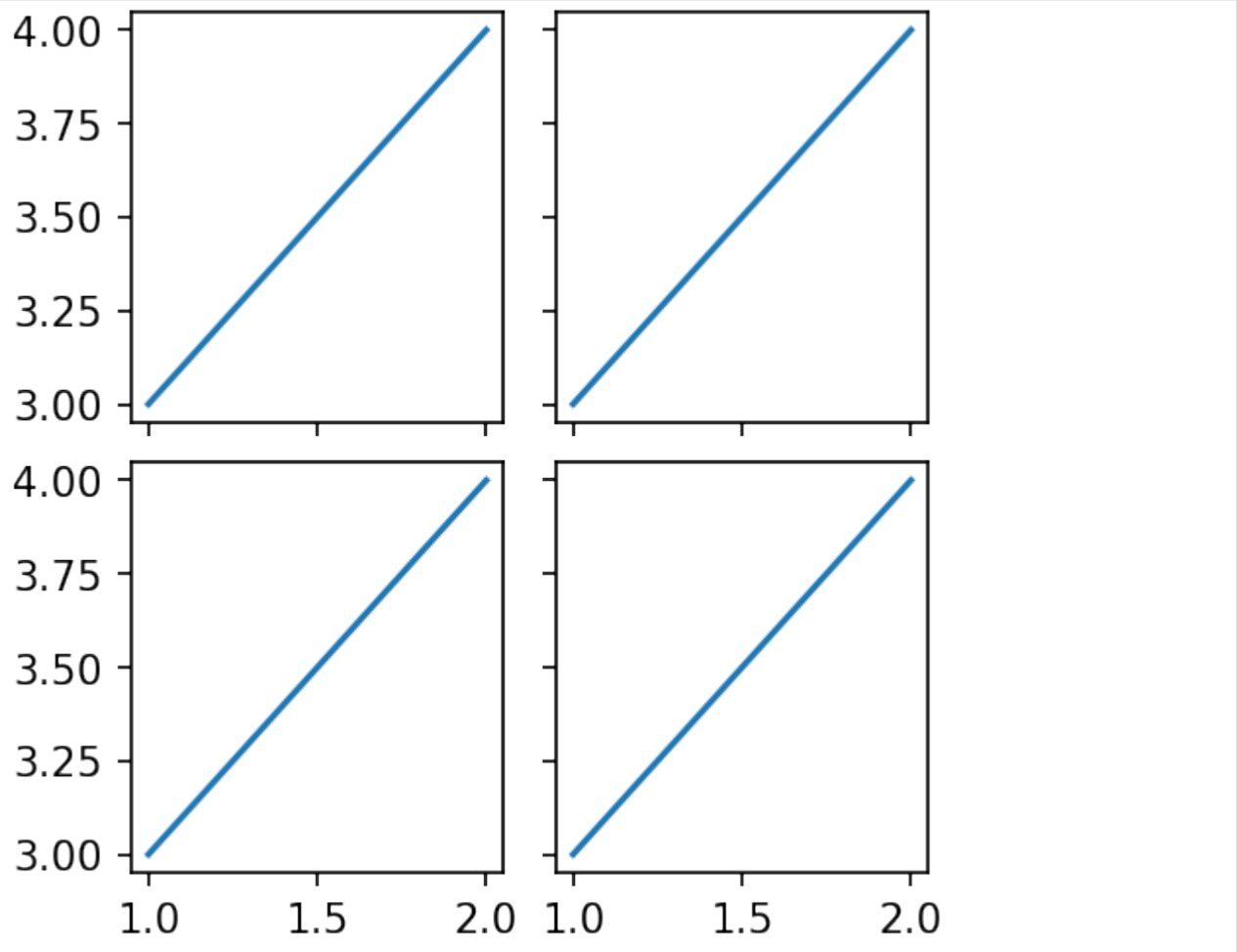


[10]:
```
plt.rcParams.update(figsizes.neurips2021(nrows=2, ncols=3))

fig, axes = plt.subplots(nrows=2, ncols=3, sharex=True, sharey=True)
for ax in axes.flatten():
    ax.plot([1.0, 2.0], [3.0, 4.0])
plt.show()
```



You can also customize the `height_to_width_ratio`:

[11]:
```
plt.rcParams.update(figsizes.icml2022_half(nrows=2, ncols=2, height_to_width_ratio=1.0))

fig, axes = plt.subplots(nrows=2, ncols=2, sharex=True, sharey=True)
for ax in axes.flatten():
    ax.plot([1.0, 2.0], [3.0, 4.0])
plt.show()
```



[ ]:

1.9 Choosing the correct fonts

[ ]:
```
import matplotlib.pyplot as plt

from tueplots import figsizes, fonts

# Increase the resolution of all the plots below
plt.rcParams.update({"figure.dpi": 150})

# "Better" figure size to display the font-changes
plt.rcParams.update(figsizes.icml2022_half())
```

Fonts in `tueplots` follow the same interface as the other settings.

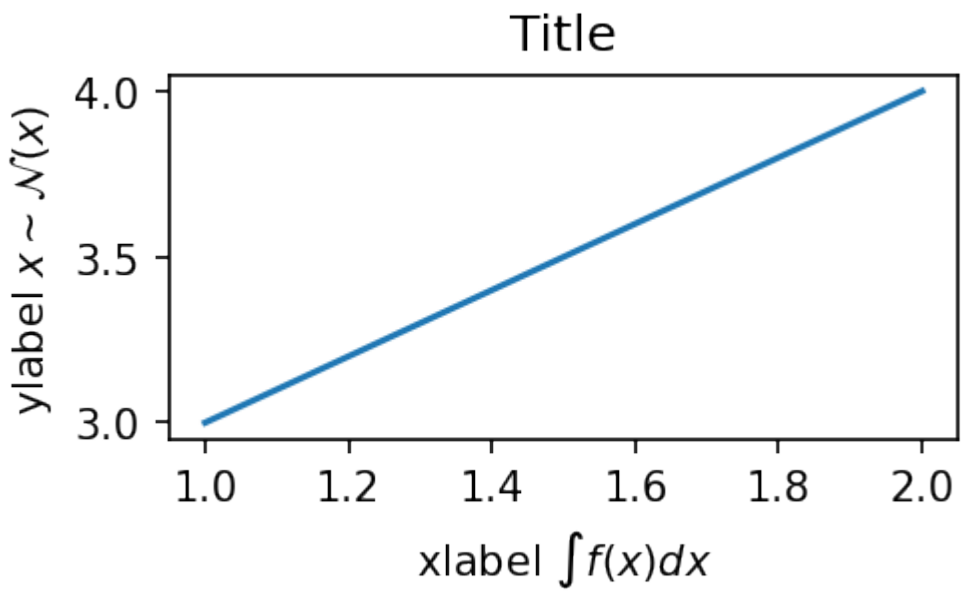
There are some pre-defined font recipes for a few journals, and they return dictionaries that are compatible with `matplotlib.pyplot.rcParams.update()`.

[2]:
```
fonts.neurips2021()
```

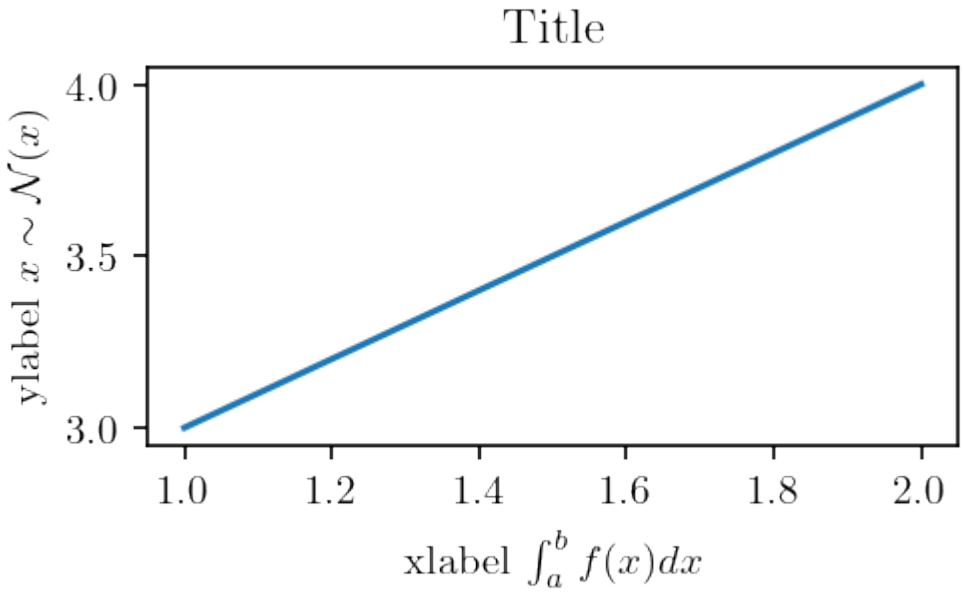
[2]:
```
{'text.usetex': False,
 'font.serif': ['Times New Roman'],
 'mathtext.fontset': 'stix',
 'mathtext.rm': 'Times New Roman',
 'mathtext.it': 'Times New Roman:italic',
 'mathtext.bf': 'Times New Roman:bold',
 'font.family': 'serif'}
```

Compare the following default font to some of the alternatives that we provide:

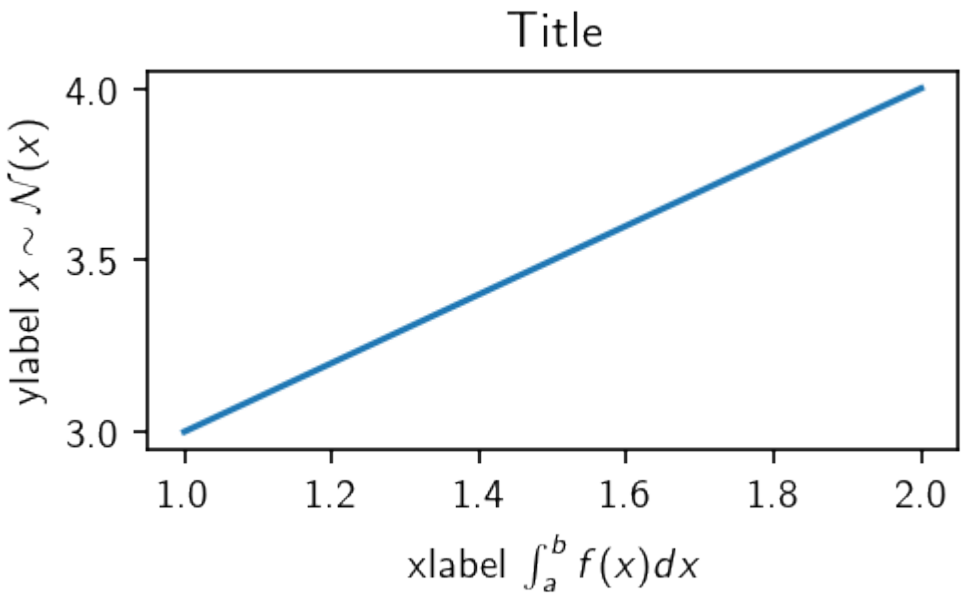
[3]:
```
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
ax.set_title("Title")
ax.set_xlabel("xlabel $\int f(x) dx$")
ax.set_ylabel("ylabel $x \sim \mathcal{N}(x)$")
plt.show()
```



[4]:
```
plt.rcParams.update(fonts.jmlr2001_tex(family="serif"))
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
ax.set_title("Title")
ax.set_xlabel("xlabel $\int_a^b f(x) dx$")
ax.set_ylabel("ylabel $x \sim \mathcal{N}(x)$")
plt.show()
```



[5]:
```
plt.rcParams.update(fonts.jmlr2001_tex(family="sans-serif"))
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
ax.set_title("Title")
ax.set_xlabel("xlabel $\int_a^b f(x) dx$")
ax.set_ylabel("ylabel $x \sim \mathcal{N}(x)$")
plt.show()
```

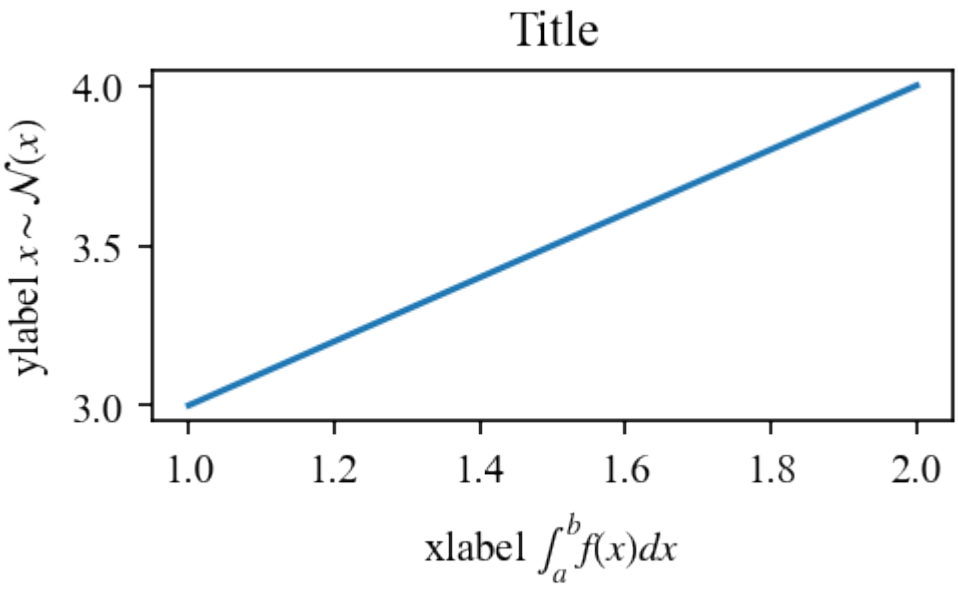


[6]:
```
plt.rcParams.update(fonts.neurips2021())
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
```

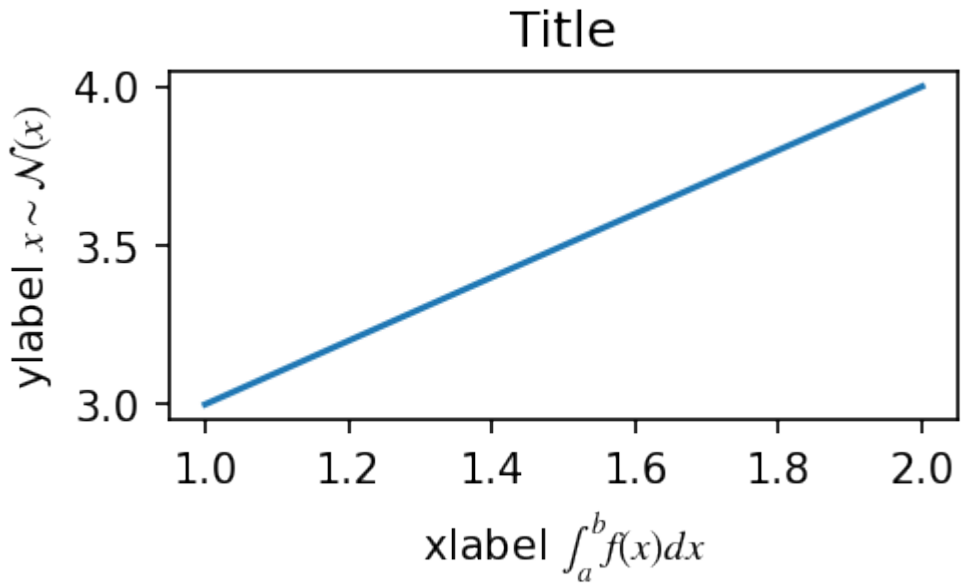
(continues on next page)

(continued from previous page)

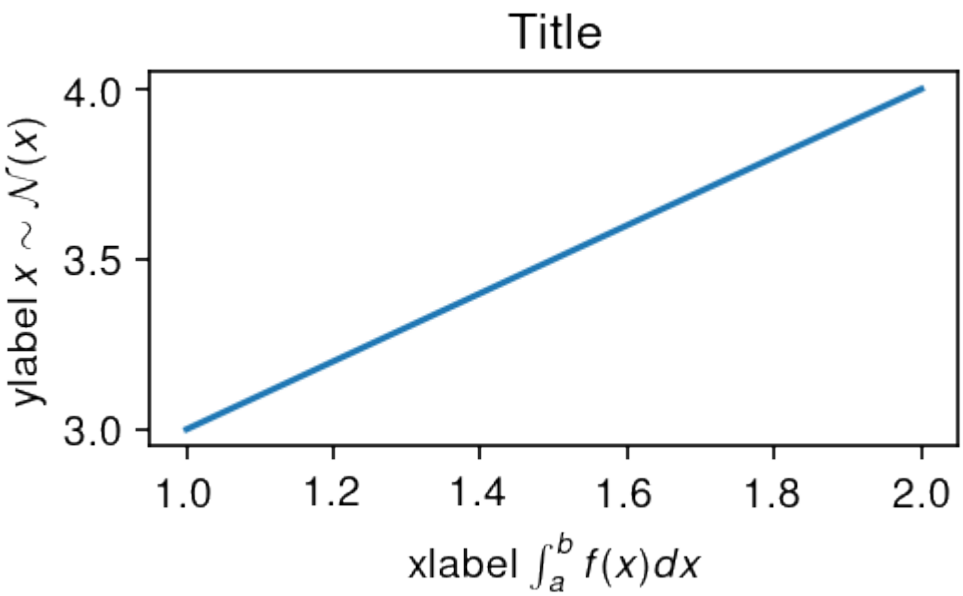
```
ax.set_title("Title")
ax.set_xlabel("xlabel $\int_a^b f(x) dx$")
ax.set_ylabel("ylabel $x \sim \mathcal{N}(x)$")
plt.show()
```



[7]:
```
plt.rcParams.update(fonts.neurips2021(family="sans-serif"))
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
ax.set_title("Title")
ax.set_xlabel("xlabel $\int_a^b f(x) dx$")
ax.set_ylabel("ylabel $x \sim \mathcal{N}(x)$")
plt.show()
```



[8]:
```
plt.rcParams.update(fonts.neurips2021_tex(family="sans-serif"))
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
ax.set_title("Title")
ax.set_xlabel("xlabel $\int_a^b f(x) dx$")
ax.set_ylabel("ylabel $x \sim \mathcal{N}(x)$")
plt.show()
```

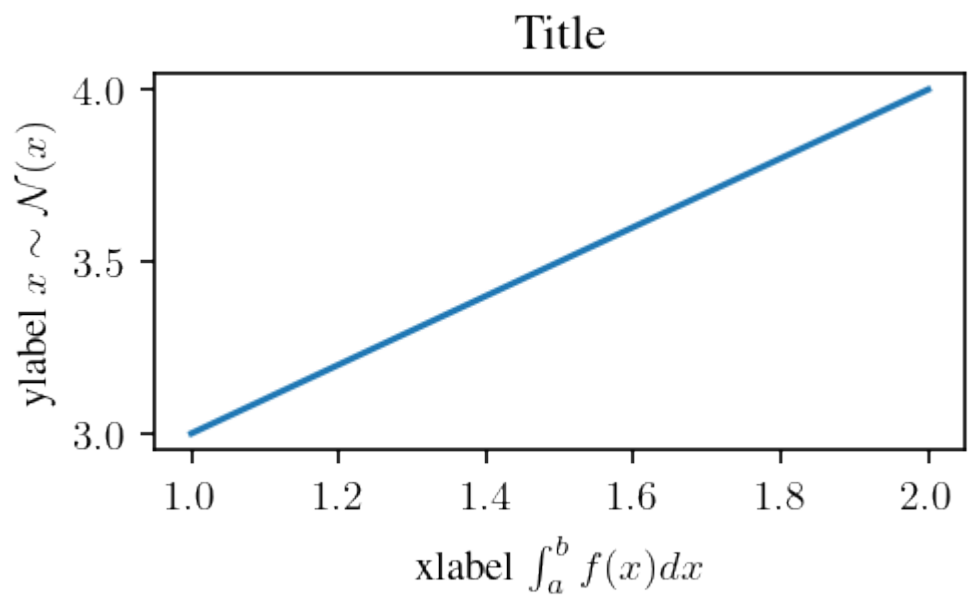


[9]:
```
plt.rcParams.update(fonts.neurips2021_tex(family="serif"))
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
```

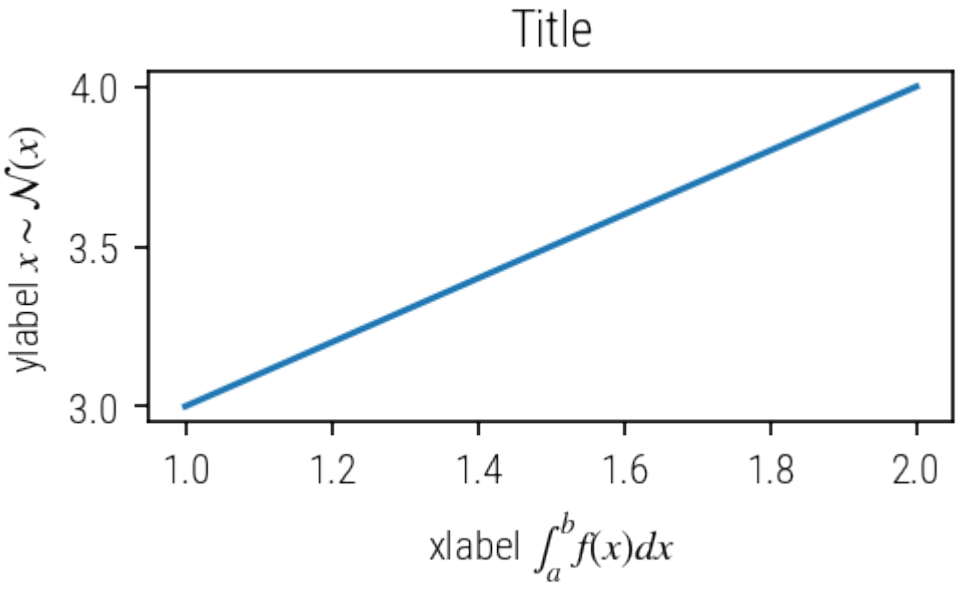
(continues on next page)

(continued from previous page)

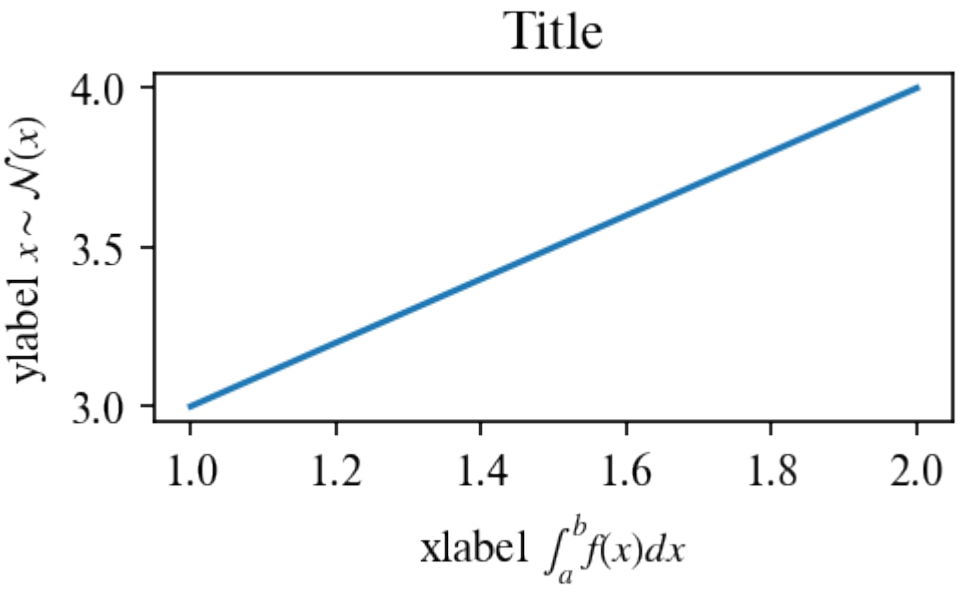
```
ax.set_title("Title")
ax.set_xlabel("xlabel $\int_a^b f(x) dx$")
ax.set_ylabel("ylabel $x \sim \mathcal{N}(x)$")
plt.show()
```



[10]:
```
plt.rcParams.update(fonts.beamer_moml())
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
ax.set_title("Title")
ax.set_xlabel("xlabel $\int_a^b f(x) dx$")
ax.set_ylabel("ylabel $x \sim \mathcal{N}(x)$")
plt.show()
```



[11]:
```
with plt.rc_context(fonts.icml2022()):
    fig, ax = plt.subplots()
    ax.plot([1.0, 2.0], [3.0, 4.0])
    ax.set_title("Title")

    ax.set_xlabel("xlabel $\int_a^b f(x) dx$")
    ax.set_ylabel("ylabel $x \sim \mathcal{N}(x)$")
    plt.show()
```

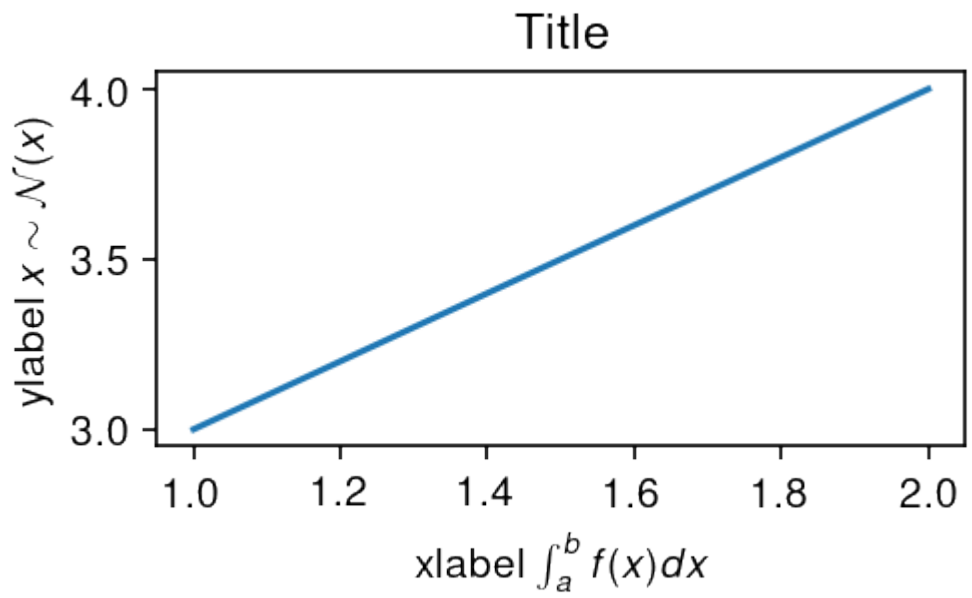


[12]:
```
with plt.rc_context(fonts.icml2022_tex(family="sans-serif")):
    fig, ax = plt.subplots()
```

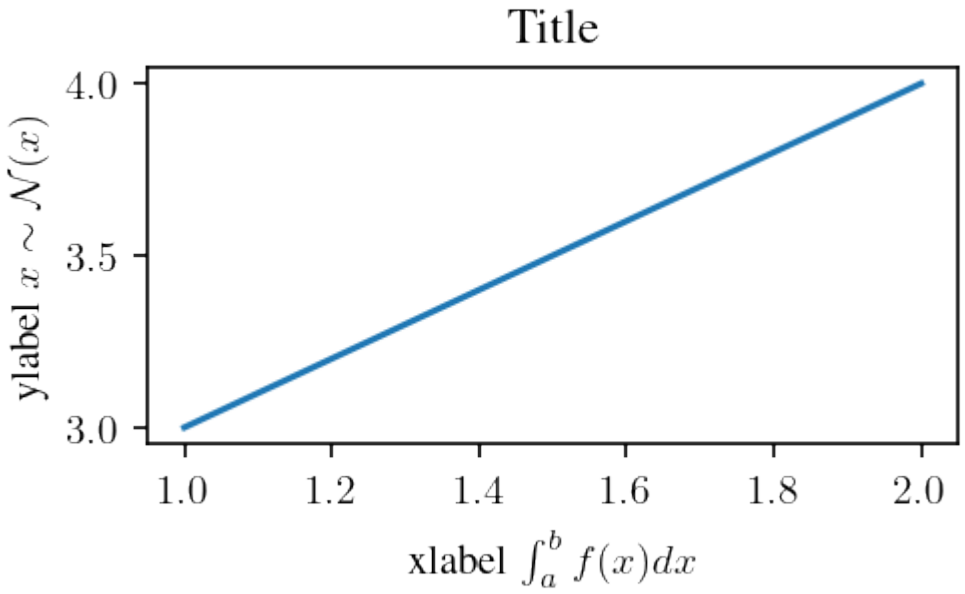
(continues on next page)

(continued from previous page)

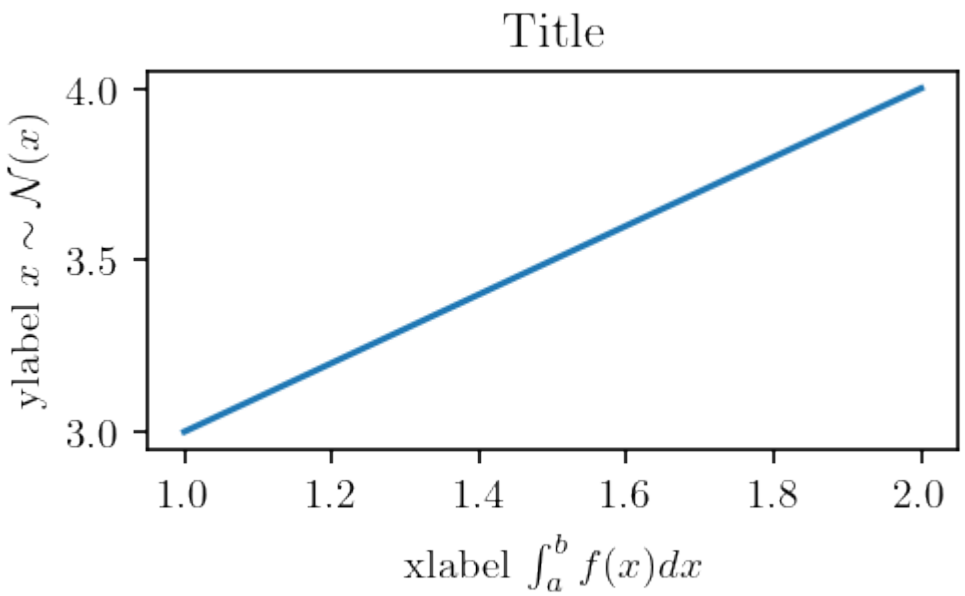
```
    ax.plot([1.0, 2.0], [3.0, 4.0])
    ax.set_title("Title")
    ax.set_xlabel("xlabel $\int_a^b f(x) dx$")
    ax.set_ylabel("ylabel $x \sim \mathcal{N}(x)$")
    plt.show()
```



[13]:
```
with plt.rc_context(fonts.icml2022_tex(family="serif")):
    fig, ax = plt.subplots()
    ax.plot([1.0, 2.0], [3.0, 4.0])
    ax.set_title("Title")
    ax.set_xlabel("xlabel $\int_a^b f(x) dx$")
    ax.set_ylabel("ylabel $x \sim \mathcal{N}(x)$")
    plt.show()
```



[14]:
```
plt.rcParams.update(fonts.aistats2022_tex(family="serif"))
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
ax.set_title("Title")
ax.set_xlabel("xlabel $\int_a^b f(x) dx$")
ax.set_ylabel("ylabel $x \sim \mathcal{N}(x)$")
plt.show()
```

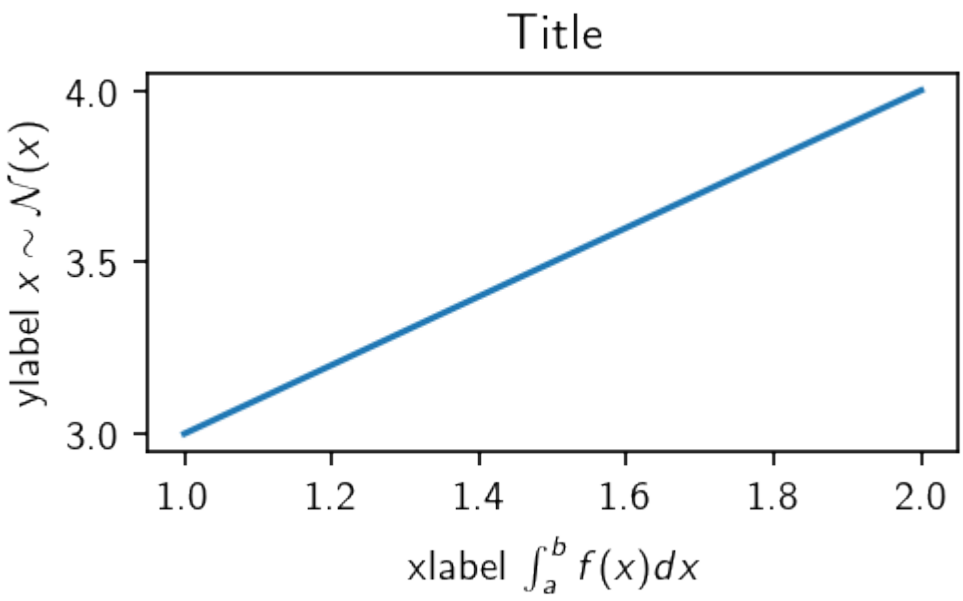


[15]:
```
plt.rcParams.update(fonts.aistats2022_tex(family="sans-serif"))
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
```

(continues on next page)

(continued from previous page)

```
ax.set_title("Title")
ax.set_xlabel("xlabel $\int_a^b f(x) dx$")
ax.set_ylabel("ylabel $x \sim \mathcal{N}(x)$")
plt.show()
```



[ ]:

1.10 Matching fontsizes

[ ]:
```
import matplotlib.pyplot as plt

from tueplots import figsizes, fontsizes

# Increase the resolution of all the plots below
plt.rcParams.update({"figure.dpi": 150})

# "Better" figure size to display the font-changes
plt.rcParams.update(figsizes.icml2022_half())
```

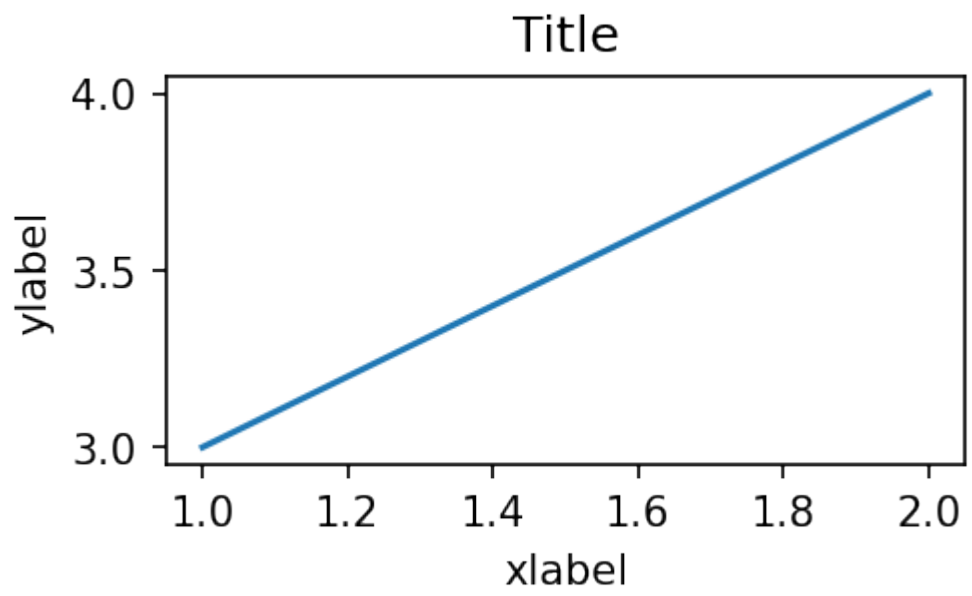
Fontsizes are dictionaries that can be passed to `matplotlib.pyplot.rcParams.update()`.

[2]:
```
fontsizes.icml2022()
```

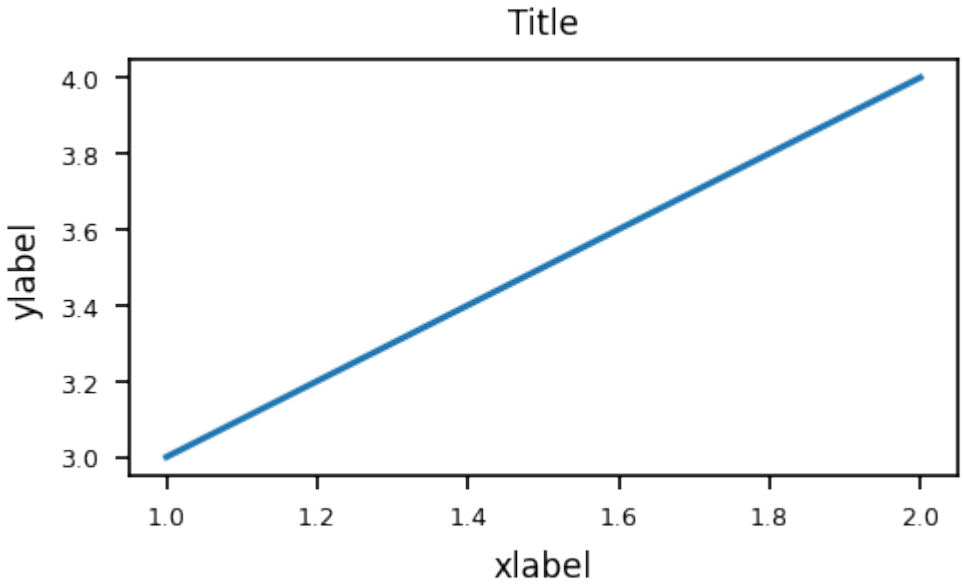
[2]:
```
{'font.size': 8,
 'axes.labelsize': 8,
 'legend.fontsize': 6,
 'xtick.labelsize': 6,
 'ytick.labelsize': 6,
 'axes.titlesize': 8}
```

Compare the default font-sizes to, e.g., the ICML style. (To make differences more obvious, we increase the `dpi` value.)

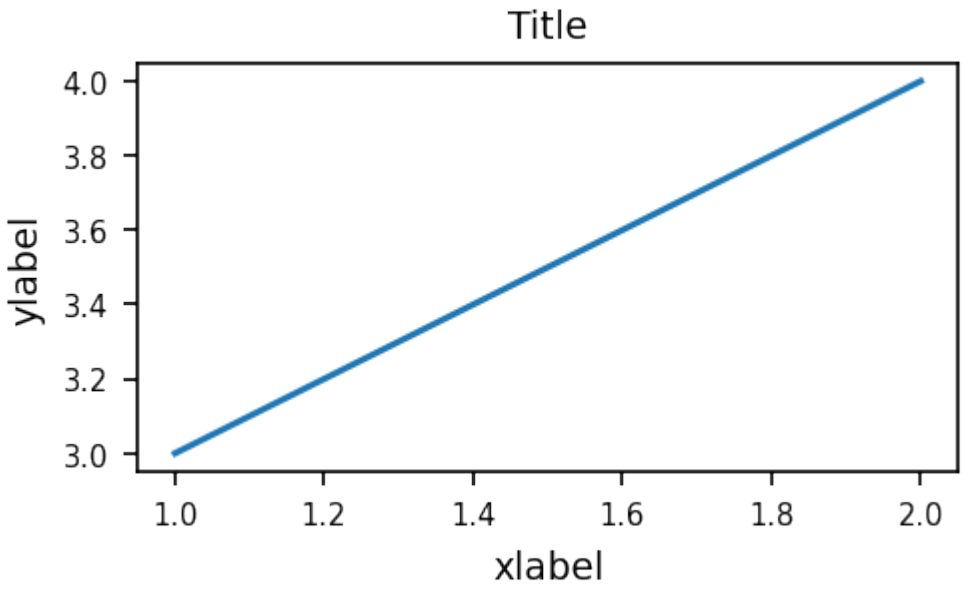
[3]:
```
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
ax.set_title("Title")
ax.set_xlabel("xlabel")
ax.set_ylabel("ylabel")
plt.show()
```



[4]:
```
plt.rcParams.update(fontsizes.icml2022())
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
ax.set_title("Title")
ax.set_xlabel("xlabel")
ax.set_ylabel("ylabel")

plt.show()
```



[5]:
```
plt.rcParams.update(fontsizes.neurips2021())
fig, ax = plt.subplots()
ax.plot([1.0, 2.0], [3.0, 4.0])
ax.set_title("Title")
ax.set_xlabel("xlabel")
ax.set_ylabel("ylabel")

plt.show()
```

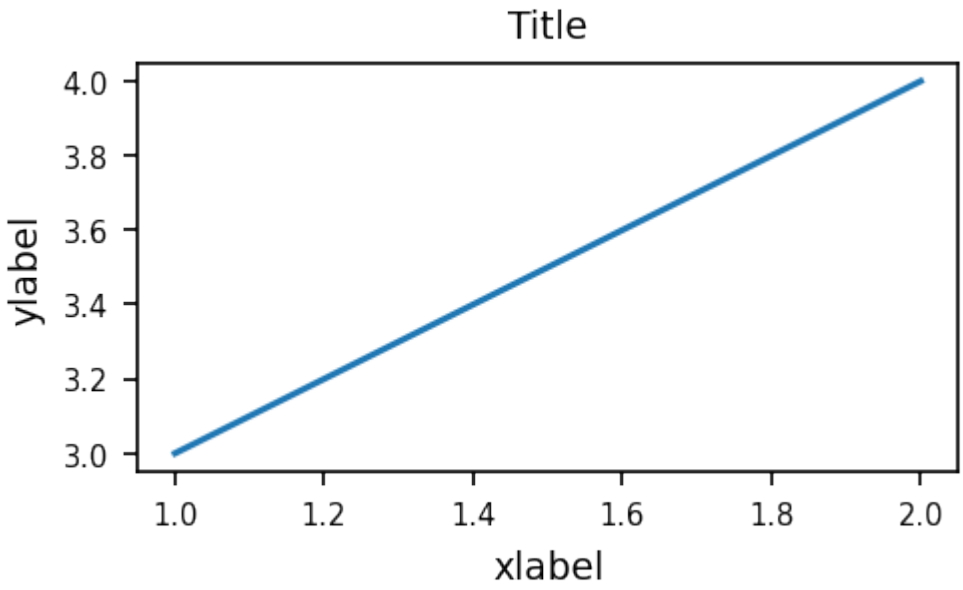


[6]:
```
plt.rcParams.update(fontsizes.aistats2022())
fig, ax = plt.subplots()
```

(continues on next page)

(continued from previous page)

```
ax.plot([1.0, 2.0], [3.0, 4.0])
ax.set_title("Title")
ax.set_xlabel("xlabel")
ax.set_ylabel("ylabel")

plt.show()
```



[ ]:

[ ]:

[ ]:

1.11 Create custom markers

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

from tueplots import cycler, markers
from tueplots.constants import markers as marker_constants
from tueplots.constants.color import palettes

# Increase the resolution of all the plots below
plt.rcParams.update({"figure.dpi": 150})
```

There are a few marker styles to choose from. (Setup taken from https://matplotlib.org/stable/tutorials/intermediate/color_cycle.html). There are rcParam dictionaries including specific marker styles and full marker cycles.

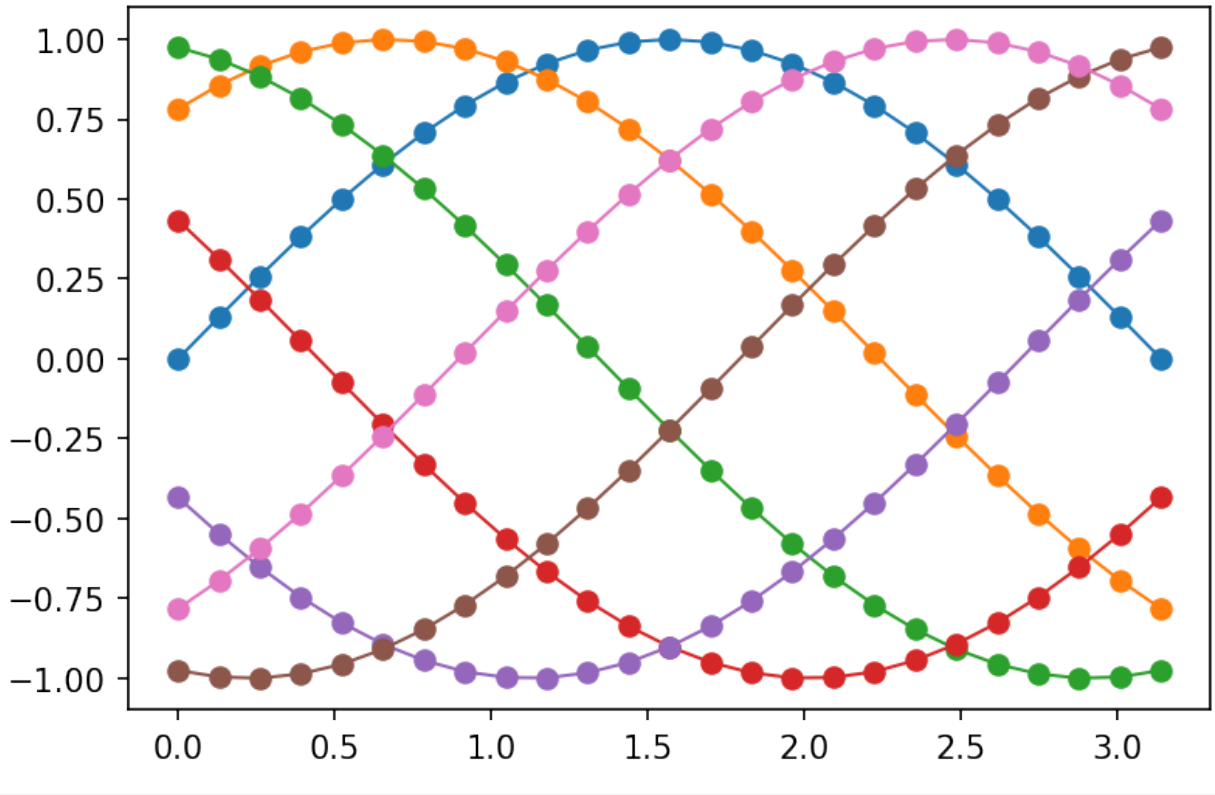
[2]:
```
markers.with_edge()
```

[2]:
```
{'lines.markeredgecolor': 'black',
 'lines.markerfacecolor': 'auto',
 'lines.markeredgewidth': 0.5}
```

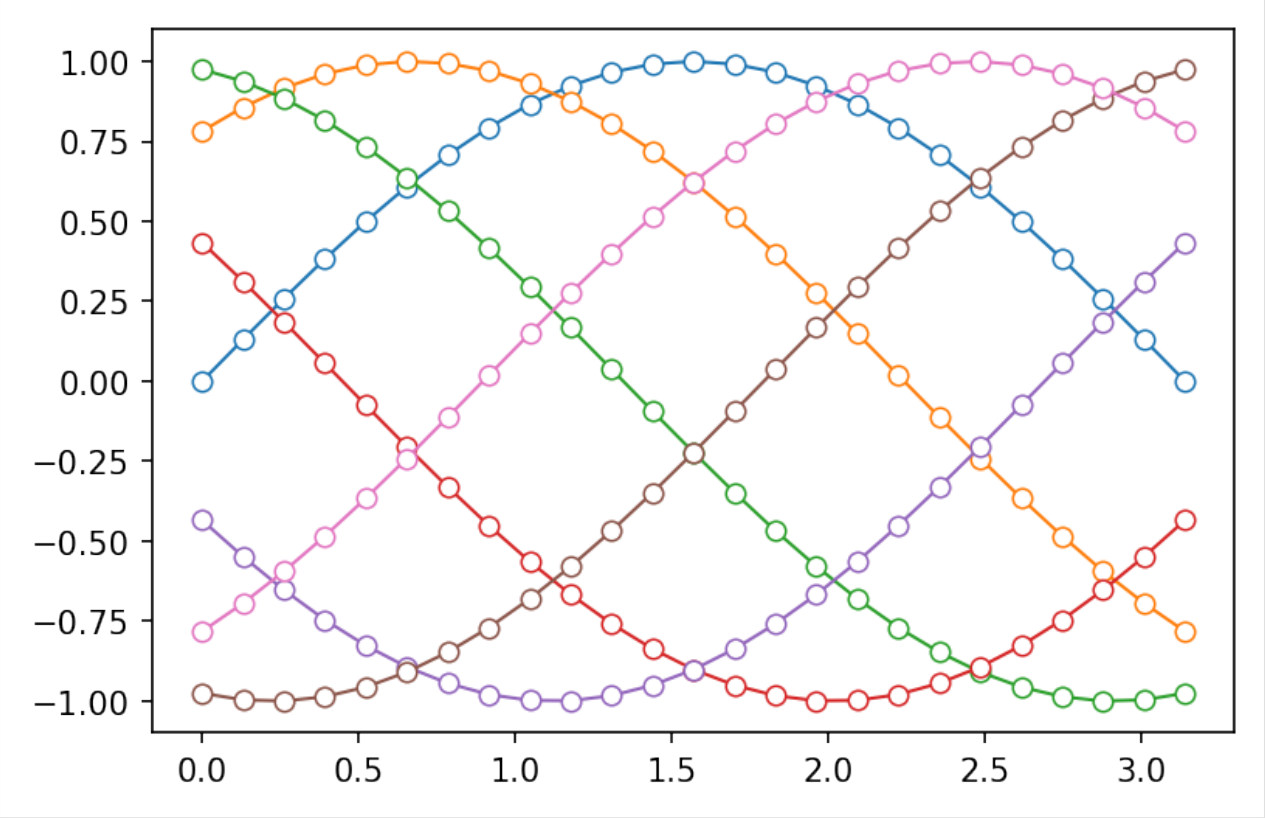
Compare the default options to the specialised markers.

[3]:
```
x = np.linspace(0, np.pi, 25)
offsets = np.linspace(0, 2 * np.pi, 7, endpoint=False)
yy = [np.sin(x + phi) for phi in offsets]
```

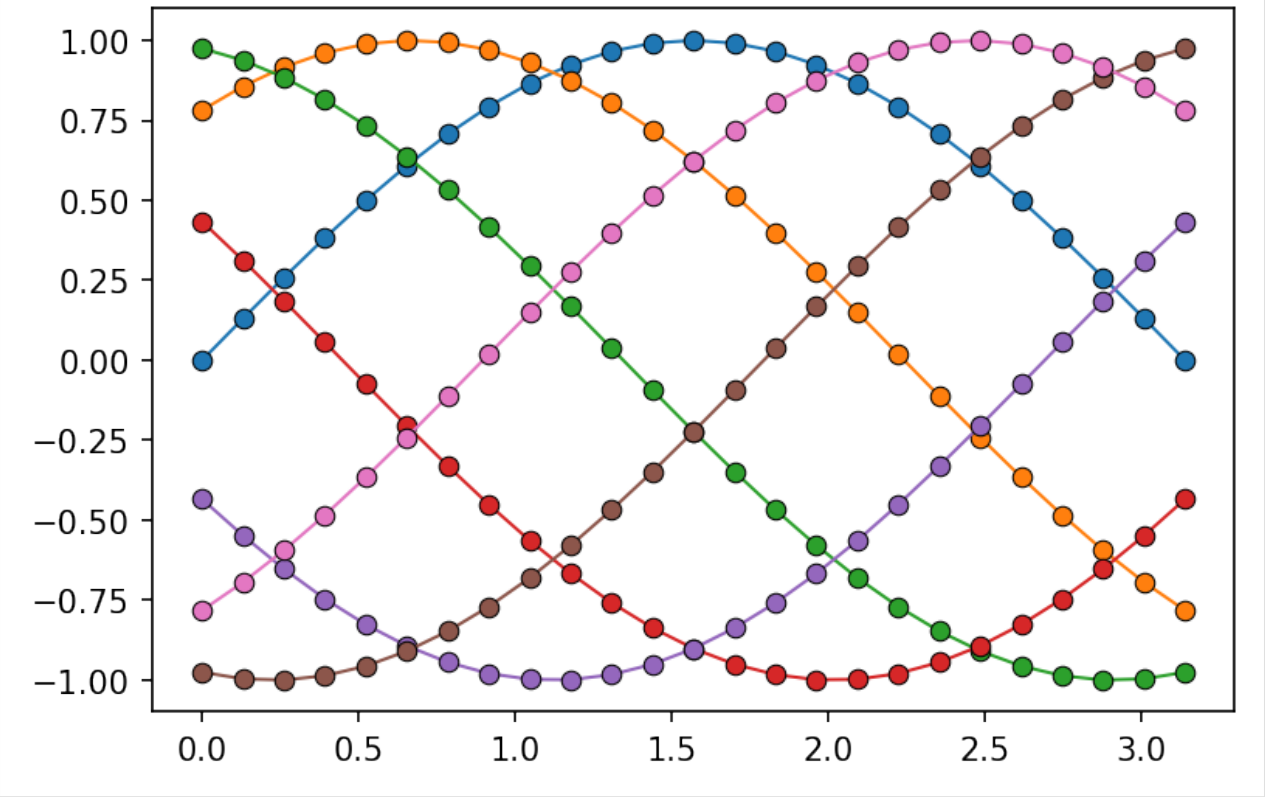
[4]:
```
fig, ax = plt.subplots()
for y in yy:
    ax.plot(x, y, "o-", linewidth=1)
plt.show()
```



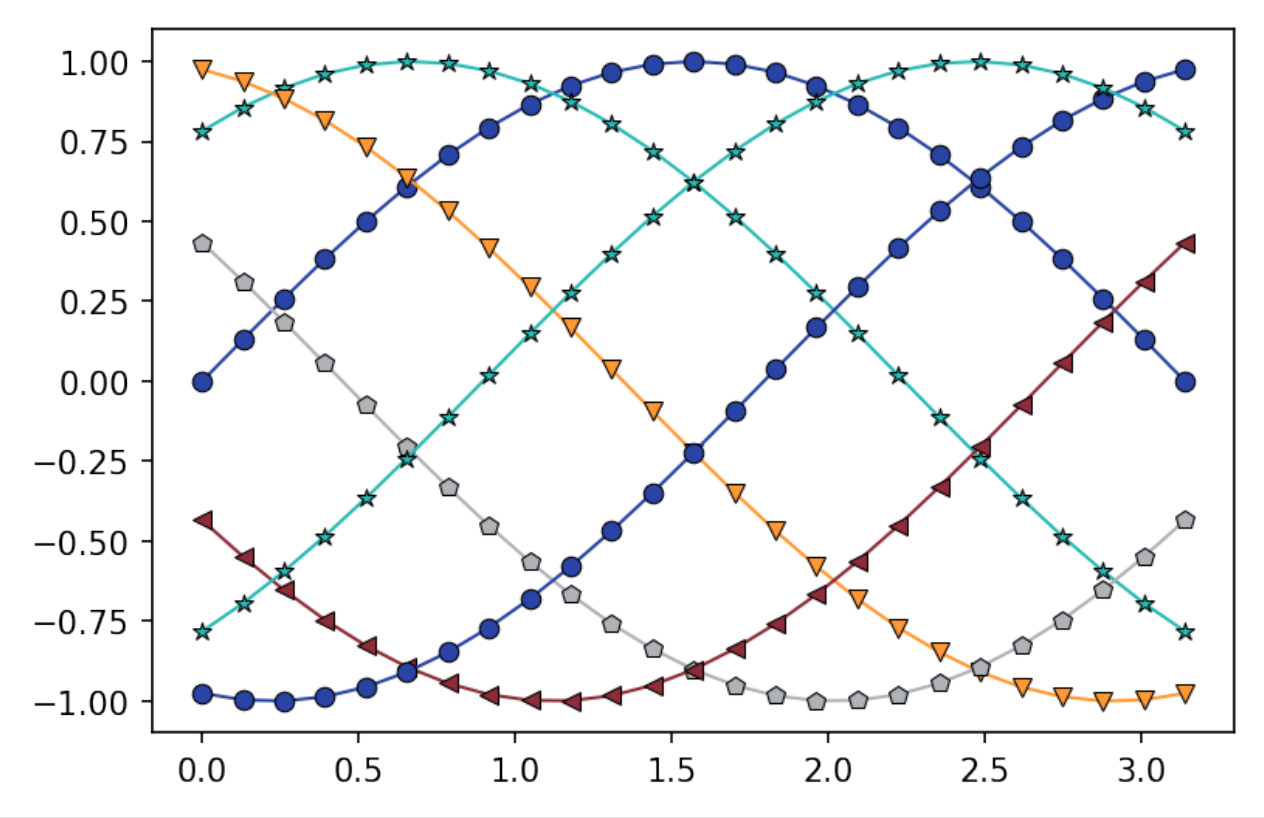
[5]:
```
plt.rcParams.update(markers.inverted())

fig, ax = plt.subplots()
for y in yy:
    ax.plot(x, y, "o-", linewidth=1)
plt.show()
```



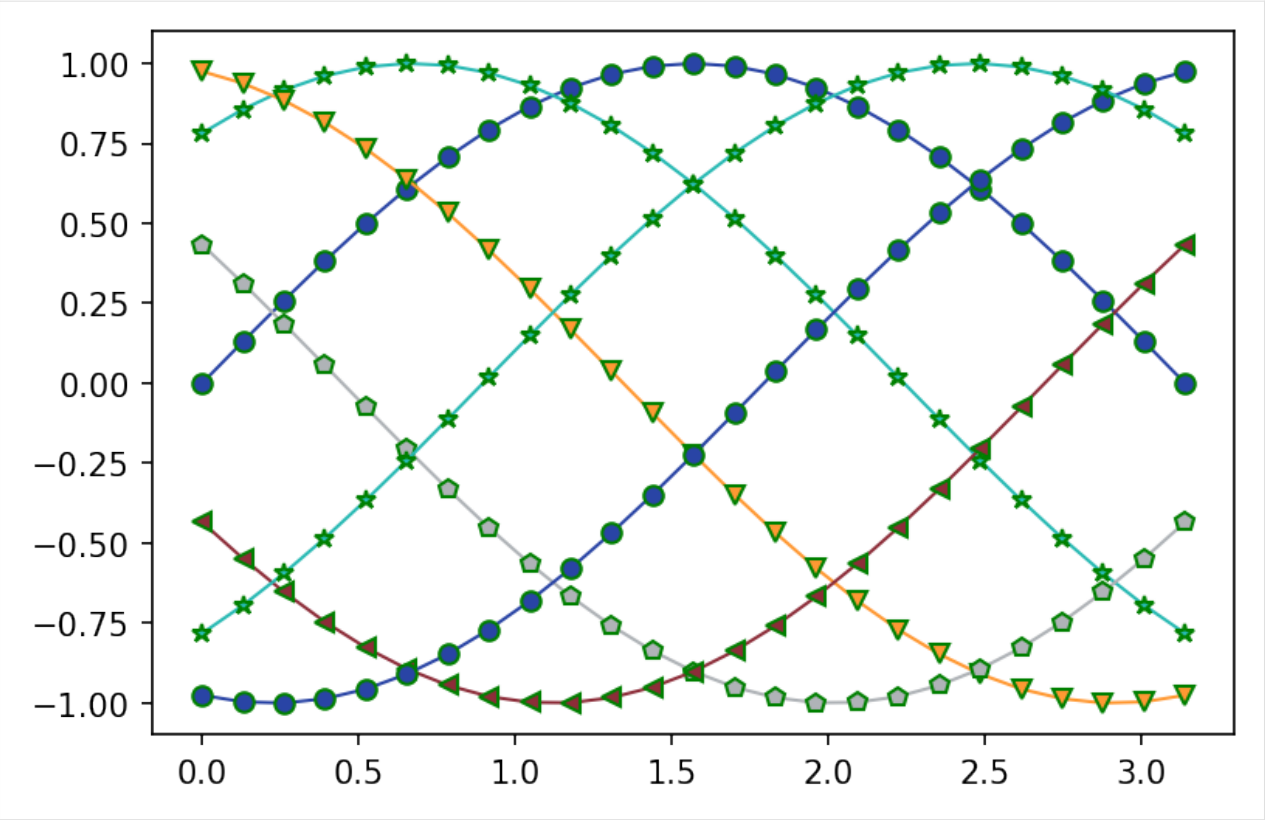
[6]:
```
plt.rcParams.update(markers.with_edge())

fig, ax = plt.subplots()
for y in yy:
    ax.plot(x, y, "o-", linewidth=1)
plt.show()
```



[7]:
```
plt.rcParams.update(
    cycler.cycler(marker=marker_constants.o_sized[:5], color=palettes.pn[:5])
)

fig, ax = plt.subplots()
for y in yy:
    ax.plot(x, y, linewidth=1)
plt.show()
```



[8]:
```
plt.rcParams.update(markers.with_edge(edgecolor="green", edgewidth=1.0))

fig, ax = plt.subplots()
for y in yy:
    ax.plot(x, y, linewidth=1)
plt.show()
```



[ ]:

1.12 API: tueplots package

TUEplots.

1.12.1 tueplots.axes module

Axes behaviour.

`tueplots.axes.`**`color`**(*, *base='black'*, *face='none'*)

Adjust the axes' color.

`tueplots.axes.`**`grid`**(*, *grid_alpha=0.2*, *grid_linestyle='solid'*)

Adjust the grid-style.

`tueplots.axes.`**`legend`**(*, *shadow=False*, *frameon=True*, *fancybox=False*)

Adjust the legend-style.

`tueplots.axes.`**`lines`**(*, *base_width=0.5*, *line_base_ratio=2.0*, *tick_major_base_ratio=1.0*, *tick_minor_base_ratio=0.5*, *tick_size_width_ratio=3.0*, *tick_major_size_min=3.0*, *tick_minor_size_min=2.0*, *axisbelow=True*)

Adjust linewidth(s) according to a base width.

tueplots.axes.**spines**(*, *left=True*, *right=True*, *top=True*, *bottom=True*)

Adjust the visibility of the axes' spines.

tueplots.axes.**tick_direction**(*, *x='inout'*, *y='inout'*)

Adjust the tick direction.

1.12.2 tueplots.bundles module

Bundled configurations.

tueplots.bundles.**aaai2024**(*, *column='half'*, *nrows=1*, *ncols=1*, *family='serif'*, *rel_width=1.0*)

AAAI 2024 bundle.

Source: https://aaai.org/wp-content/uploads/2023/06/AuthorKit24.zip

tueplots.bundles.**aistats2022**(*, *column='half'*, *nrows=1*, *ncols=1*, *family='serif'*)

AISTATS 2022 bundle.

tueplots.bundles.**aistats2023**(*, *column='half'*, *nrows=1*, *ncols=1*, *family='serif'*)

AISTATS 2023 bundle.

tueplots.bundles.**beamer_moml**(*, *rel_width=1.0*, *rel_height=0.8*)

Beamer bundle that matches the template of the method-of-machine-learning group in Tübingen.

tueplots.bundles.**beamer_moml_dark_bg**(*, *rel_width=1.0*, *rel_height=0.8*)

Dark version of *beamer_moml()*.

tueplots.bundles.**cvpr2024**(*, *column='half'*, *nrows=1*, *ncols=1*, *usetex=True*, *family='serif'*)

CVPR 2024 bundle.

tueplots.bundles.**eccv2024**(*, *rel_width=1.0*, *nrows=1*, *ncols=1*, *family='serif'*)

ECCV 2024 bundle.

tueplots.bundles.**iclr2023**(*, *usetex=True*, *rel_width=1.0*, *nrows=1*, *ncols=1*, *family='serif'*)

ICLR 2023 bundle.

tueplots.bundles.**iclr2024**(*, *usetex=True*, *rel_width=1.0*, *nrows=1*, *ncols=1*, *family='serif'*)

ICLR 2024 bundle.

tueplots.bundles.**icml2022**(*, *column='half'*, *nrows=1*, *ncols=1*, *usetex=True*, *family='serif'*)

ICML 2022 bundle.

tueplots.bundles.**icml2024**(*, *column='half'*, *nrows=1*, *ncols=1*, *usetex=True*, *family='serif'*)

ICML 2024 bundle.

tueplots.bundles.**jmlr2001**(*, *rel_width=1.0*, *nrows=1*, *ncols=1*, *family='serif'*)

JMLR 2001 bundle.

tueplots.bundles.**neurips2021**(*, *usetex=True*, *rel_width=1.0*, *nrows=1*, *ncols=1*, *family='serif'*)

Neurips 2021 bundle.

tueplots.bundles.**neurips2022**(*, *usetex=True*, *rel_width=1.0*, *nrows=1*, *ncols=1*, *family='serif'*)

Neurips 2022 bundle.

tueplots.bundles.**neurips2023**(*, *usetex=True*, *rel_width=1.0*, *nrows=1*, *ncols=1*, *family='serif'*)

Neurips 2023 bundle.

tueplots.bundles.**neurips2024**(*, *usetex=True*, *rel_width=1.0*, *nrows=1*, *ncols=1*, *family='serif'*)

Neurips 2024 bundle.

tueplots.bundles.**tmlr2023**(*, *rel_width=1.0*, *nrows=1*, *ncols=1*, *family='serif'*)

TMLR 2023 bundle.

tueplots.bundles.**uai2023**(*, *column='half'*, *nrows=1*, *ncols=1*, *family='serif'*)

UAI 2023 bundle.

1.12.3 tueplots.cycler module

Wrapper for matplotlib cycler.

tueplots.cycler.**cycler**(***kwargs*)

Wrap a matplotlib cycler object into a parameter-dictionary compatible with plt.rcParams.

Please refer to

https://matplotlib.org/cycler/

and

https://matplotlib.org/stable/tutorials/intermediate/color_cycle.html

for information about how to use cyclers.

1.12.4 tueplots.figsizes module

Figure size settings.

tueplots.figsizes.**aaai2024_full**(*, *nrows=1*, *ncols=1*, *constrained_layout=True*, *tight_layout=False*, *height_to_width_ratio=0.6180339887498949*, *pad_inches=0.015*, *rel_width=1.0*)

Double-column (full-width) figures for AAAI 2024.

tueplots.figsizes.**aaai2024_half**(*, *nrows=1*, *ncols=1*, *constrained_layout=True*, *tight_layout=False*, *height_to_width_ratio=0.6180339887498949*, *pad_inches=0.015*, *rel_width=1.0*)

Double-column (half-width) figures for AAAI 2024.

tueplots.figsizes.**aistats2022_full**(***kwargs*)

Single-column (full-width) figures for AISTATS 2022.

tueplots.figsizes.**aistats2022_half**(***kwargs*)

Double-column (half-width) figures for AISTATS 2022.

tueplots.figsizes.**aistats2023_full**(***kwargs*)

Single-column (full-width) figures for AISTATS 2023.

tueplots.figsizes.**aistats2023_half**(***kwargs*)

Double-column (half-width) figures for AISTATS 2023.

tueplots.figsizes.**beamer_169**(*, *rel_width=0.9*, *rel_height=0.6*, *constrained_layout=True*, *tight_layout=False*, *pad_inches=0.015*)

Beamer figure size for *aspectratio=169*.

tueplots.figsizes.**cvpr2022_full**(*, *rel_width=1.0*, *nrows=1*, *ncols=2*, *constrained_layout=True*, *tight_layout=False*, *height_to_width_ratio=0.6180339887498949*, *pad_inches=0.015*)

Single-column (full-width) figures for CVPR 2022.

tueplots.figsizes.**cvpr2022_half**(*, *nrows=1*, *ncols=1*, *constrained_layout=True*, *tight_layout=False*, *height_to_width_ratio=0.6180339887498949*, *pad_inches=0.015*)

Double-column (half-width) figures for CVPR 2022.

tueplots.figsizes.**cvpr2024_full**(*, *rel_width=1.0*, *nrows=1*, *ncols=2*, *constrained_layout=True*, *tight_layout=False*, *height_to_width_ratio=0.6180339887498949*, *pad_inches=0.015*)

Single-column (full-width) figures for CVPR 2024.

tueplots.figsizes.**cvpr2024_half**(*, *nrows=1*, *ncols=1*, *constrained_layout=True*, *tight_layout=False*, *height_to_width_ratio=0.6180339887498949*, *pad_inches=0.015*)

Double-column (half-width) figures for CVPR 2024.

tueplots.figsizes.**eccv2024**(*, *rel_width=1.0*, *nrows=1*, *ncols=2*, *constrained_layout=True*, *tight_layout=False*, *height_to_width_ratio=0.6180339887498949*, *pad_inches=0.015*)

ECCV figure size.

Source: https://eccv2024.ecva.net/Conferences/2024/SubmissionPolicies

tueplots.figsizes.**iclr2023**(*, *rel_width=1.0*, *nrows=1*, *ncols=2*, *constrained_layout=True*, *tight_layout=False*, *height_to_width_ratio=0.6180339887498949*, *pad_inches=0.015*)

ICLR 2023 figure size.

tueplots.figsizes.**iclr2024**(*, *rel_width=1.0*, *nrows=1*, *ncols=2*, *constrained_layout=True*, *tight_layout=False*, *height_to_width_ratio=0.6180339887498949*, *pad_inches=0.015*)

ICLR 2024 figure size.

tueplots.figsizes.**icml2022_full**(**kwargs*)

Single-column (full-width) figures for ICML 2022.

tueplots.figsizes.**icml2022_half**(**kwargs*)

Double-column (half-width) figures for ICML 2022.

tueplots.figsizes.**icml2024_full**(**kwargs*)

Single-column (full-width) figures for ICML 2024.

tueplots.figsizes.**icml2024_half**(**kwargs*)

Double-column (half-width) figures for ICML 2024.

tueplots.figsizes.**jmlr2001**(*, *rel_width=1.0*, *nrows=1*, *ncols=2*, *constrained_layout=True*, *tight_layout=False*, *height_to_width_ratio=0.6180339887498949*, *pad_inches=0.015*)

JMLR figure size.

Source: https://www.jmlr.org/format/format.html

The present format is for US letter format.

tueplots.figsizes.**neurips2021**(*, *rel_width=1.0*, *nrows=1*, *ncols=2*, *constrained_layout=True*, *tight_layout=False*, *height_to_width_ratio=0.6180339887498949*, *pad_inches=0.015*)

Neurips 2021 figure size.

tueplots.figsizes.**neurips2022**(*, *rel_width=1.0*, *nrows=1*, *ncols=2*, *constrained_layout=True*, *tight_layout=False*, *height_to_width_ratio=0.6180339887498949*, *pad_inches=0.015*)

Neurips 2022 figure size.

tueplots.figsizes.**neurips2023**(*, *rel_width=1.0*, *nrows=1*, *ncols=2*, *constrained_layout=True*, *tight_layout=False*, *height_to_width_ratio=0.6180339887498949*, *pad_inches=0.015*)

Neurips 2023 figure size.

tueplots.figsizes.**neurips2024**(*, *rel_width=1.0*, *nrows=1*, *ncols=2*, *constrained_layout=True*, *tight_layout=False*, *height_to_width_ratio=0.6180339887498949*, *pad_inches=0.015*)

Neurips 2024 figure size.

Source: https://nips.cc/Conferences/2024/CallForPapers

tueplots.figsizes.**tmlr2023**(*, *rel_width=1.0*, *nrows=1*, *ncols=2*, *constrained_layout=True*, *tight_layout=False*, *height_to_width_ratio=0.6180339887498949*, *pad_inches=0.015*)

TMLR figure size.

Source: https://www.jmlr.org/tmlr/author-guide.html

tueplots.figsizes.**uai2023_full**(*, *nrows=1*, *ncols=1*, *constrained_layout=True*, *tight_layout=False*, *height_to_width_ratio=0.6180339887498949*, *pad_inches=0.015*)

Double-column (full-width) figures for UAI 2023.

tueplots.figsizes.**uai2023_half**(*, *nrows=1*, *ncols=1*, *constrained_layout=True*, *tight_layout=False*, *height_to_width_ratio=0.6180339887498949*, *pad_inches=0.015*)

Double-column (half-width) figures for UAI 2023.

1.12.5 tueplots.fonts module

Font settings for conference papers and journals.

tueplots.fonts.**aaai2024_tex**(*, *family='serif'*)

Fonts for AAAI 2024. LaTeX version.

tueplots.fonts.**aistats2022_tex**(*, *family='serif'*)

Fonts for AISTATS 2022. LaTeX version.

tueplots.fonts.**aistats2023_tex**(*, *family='serif'*)

Fonts for AISTATS 2023. LaTeX version.

tueplots.fonts.**beamer_moml**()

Fonts that are compatible with the beamer template of the method-of-machine-learning group in Tübingen.

tueplots.fonts.**beamer_moml_dark_bg**()

Fonts for *beamer_moml()* with dark background.

tueplots.fonts.**cvpr2024**(*, *family='serif'*)
Fonts for CVPR 2024. LaTeX version.

tueplots.fonts.**cvpr2024_tex**(*, *family='serif'*)
Fonts for CVPR 2024. LaTeX version.

tueplots.fonts.**eccv2024_tex**(*, *family='serif'*)
Fonts for ECCV 2024. LaTeX version.

tueplots.fonts.**iclr2023**(*, *family='serif'*)
Fonts for ICLR 2023. LaTeX version.

tueplots.fonts.**iclr2023_tex**(*, *family='serif'*)
Fonts for ICLR 2023. LaTeX version.

tueplots.fonts.**iclr2024**(*, *family='serif'*)
Fonts for ICLR 2024. LaTeX version.

tueplots.fonts.**iclr2024_tex**(*, *family='serif'*)
Fonts for ICLR 2024. LaTeX version.

tueplots.fonts.**icml2022**(*, *family='serif'*)
Fonts for ICML 2022.

tueplots.fonts.**icml2022_tex**(*, *family='serif'*)
Fonts for ICML 2022. LaTeX version.

tueplots.fonts.**icml2024**(*, *family='serif'*)
Fonts for ICML 2024.

tueplots.fonts.**icml2024_tex**(*, *family='serif'*)
Fonts for ICML 2024. LaTeX version.

tueplots.fonts.**jmlr2001_tex**(*, *family='serif'*)
Fonts for JMLR. LaTeX version.

tueplots.fonts.**neurips2021**(*, *family='serif'*)
Fonts for Neurips 2021.

tueplots.fonts.**neurips2021_tex**(*, *family='serif'*)
Fonts for Neurips 2021. LaTeX version.

tueplots.fonts.**neurips2022**(*, *family='serif'*)
Fonts for Neurips 2022.

tueplots.fonts.**neurips2022_tex**(*, *family='serif'*)
Fonts for Neurips 2022. LaTeX version.

tueplots.fonts.**neurips2023**(*, *family='serif'*)
Fonts for Neurips 2023.

tueplots.fonts.**neurips2023_tex**(*, *family='serif'*)
Fonts for Neurips 2023. LaTeX version.

tueplots.fonts.**neurips2024**(*, *family='serif'*)
Fonts for Neurips 2024.

tueplots.fonts.**neurips2024_tex**(*, *family='serif'*)
Fonts for Neurips 2024. LaTeX version.

tueplots.fonts.**tmlr2023_tex**(*, *family='serif'*)
Fonts for TMLR. LaTeX version.

tueplots.fonts.**uai2023_tex**(*, *family='serif'*)
Fonts for UAI 2023. LaTeX version.

1.12.6 tueplots.fontsizes module

Fontsize settings.

tueplots.fontsizes.**aaai2024**(*, *default_smaller=1*)
Font size for AAAI 2024.

tueplots.fontsizes.**aistats2022**(*, *default_smaller=1*)
Font size for AISTATS 2022.

tueplots.fontsizes.**aistats2023**(*, *default_smaller=1*)
Font size for AISTATS 2023.

tueplots.fontsizes.**beamer_moml**(*, *default_smaller=1*)
Font size for a beamer slide in aspectratio 16:9 with 10pt font.

tueplots.fontsizes.**cvpr2024**(*, *default_smaller=1*)
Font size for CVPR 2024.

tueplots.fontsizes.**eccv2024**(*, *default_smaller=1*)
Font size for ECCV 2024.

tueplots.fontsizes.**iclr2023**(*, *default_smaller=1*)
Font size for ICLR 2023.

tueplots.fontsizes.**iclr2024**(*, *default_smaller=1*)
Font size for ICLR 2024.

tueplots.fontsizes.**icml2022**(*, *default_smaller=1*)
Font size for ICML 2022.

Source: https://media.icml.cc/Conferences/ICML2022/Styles/example_paper.pdf

tueplots.fontsizes.**icml2024**(*, *default_smaller=1*)
Font size for ICML 2024.

Source: https://media.icml.cc/Conferences/ICML2024/Styles/example_paper.pdf

tueplots.fontsizes.**jmlr2001**(*, *default_smaller=1*)
Font size for JMLR 2021.

tueplots.fontsizes.**neurips2021**(*, *default_smaller=1*)
Font size for Neurips 2021.

tueplots.fontsizes.**neurips2022**(*, *default_smaller=1*)
Font size for Neurips 2022.

tueplots.fontsizes.**neurips2023**(*, *default_smaller=1*)
Font size for Neurips 2023.

tueplots.fontsizes.**neurips2024**(*, *default_smaller=1*)

Font size for Neurips 2024.

tueplots.fontsizes.**tmlr2023**(*, *default_smaller=1*)

Font size for TMLR 2023.

tueplots.fontsizes.**uai2023**(*, *default_smaller=1*)

Font size for UAI 2023.

1.12.7 tueplots.markers module

Marker styles.

tueplots.markers.**inverted**(*, *facecolor='white'*, *edgewidth=0.75*)

The edgecolor is set to the linecolor, the facecolor is changed.

tueplots.markers.**with_edge**(*, *edgecolor='black'*, *edgewidth=0.5*)

The facecolor is set to the linecolor, the edgecolor is changed.

1.12.8 tueplots.constants package

tueplots.constants.markers module

Marker collections.

tueplots.constants.markers.**o_sized = array(['o', '*', 'v', 'p', '<', 'P', '>', 'X', '^', 'D'], dtype='<U1')**

All markers that have a size roughly similar to ‘o’, ‘^’, etc..

tueplots.constants.markers.**triangles = array(['^', 'v', '<', '>'], dtype='<U1')**

All triangular markers.

tueplots.constants.markers.**x_like = array(['x', 'p', '*'], dtype='<U1')**

All markers that look roughly like the bold *X* (i.e. bold +, *, etc.)

tueplots.constants.markers.**x_like_bold = array(['X', 'P', '*'], dtype='<U1')**

All markers that look roughly like X (i.e. +, *, etc.)

tueplots.constants.color package

tueplots.constants.color.palettes module

Color palettes.

tueplots.constants.color.palettes.**paultol_bright = array(['4477AA', 'EE6677', '228833', 'CCBB44', '66CCEE', 'AA3377', 'BBBBBB'], dtype='<U6')**

Bright colors.

From Paul Tol’s website: https://personal.sron.nl/~pault/.

tueplots.constants.color.palettes.**paultol_high_contrast = array(['004488', 'DDAA33', 'BB5566'], dtype='<U6')**

High-contrast colors.

From Paul Tol's website: https://personal.sron.nl/~pault/.

tueplots.constants.color.palettes.**paultol_light = array(['77AADD', '99DDFF', '44BB99', 'BBCC33', 'AAAA00', 'EEDD88', 'EE8866', 'FFAABB', 'DDDDDD'], dtype='<U6')**

Light colors.

From Paul Tol's website: https://personal.sron.nl/~pault/.

tueplots.constants.color.palettes.**paultol_medium_contrast = array(['EECC66', 'EE99AA', '6699CC', '997700', '994455', '004488'], dtype='<U6')**

Medium-contrast colors.

From Paul Tol's website: https://personal.sron.nl/~pault/.

tueplots.constants.color.palettes.**paultol_muted = array(['CC6677', '332288', 'DDCC77', '117733', '88CCEE', '882255', '44AA99', '999933', 'AA4499', 'DDDDDD'], dtype='<U6')**

Muted colors.

From Paul Tol's website: https://personal.sron.nl/~pault/.

tueplots.constants.color.palettes.**paultol_vibrant = array(['0077BB', '33BBEE', '009988', 'EE7733', 'CC3311', 'EE3377', 'BBBBBB'], dtype='<U6')**

Vibrant colors.

From Paul Tol's website: https://personal.sron.nl/~pault/.

tueplots.constants.color.palettes.**pn = array([[0.16078431, 0.26666667, 0.64705882], [0.15686275, 0.72941176, 0.70588235], [1. , 0.6 , 0.2], [0.68627451, 0.70196078, 0.71764706], [0.55294118, 0.17647059, 0.22352941]])**

Colors used for probnum.org.

tueplots.constants.color.palettes.**tue_plot = array([[0.55294118, 0.17647059, 0.22352941], [0.21568627, 0.25490196, 0.29019608], [0.68235294, 0.62352941, 0.42745098], [0. , 0.41176471, 0.66666667], [0.68627451, 0.70196078, 0.71764706], [0.49019608, 0.64705882, 0.29411765], [0.78431373, 0.31372549, 0.23529412], [0.68627451, 0.43137255, 0.58823529], [0.70588235, 0.62745098, 0.58823529], [0.56862745, 0.41176471, 0.2745098]])**

A mix of the primary and secondary colors from tue TUE color palette(s), ordered in a way that provides okay(ish) color contrast.

tueplots.constants.color.palettes.**tue_plot_dark_bg = array([[0.9372549 , 0.9254902 , 0.88627451], [0.55294118, 0.17647059, 0.22352941], [0.31372549, 0.66666667, 0.78431373], [0.68627451, 0.70196078, 0.71764706], [0.50980392, 0.7254902 , 0.62745098], [0.78431373, 0.31372549, 0.23529412], [0.68627451, 0.43137255, 0.58823529], [0.70588235, 0.62745098, 0.58823529], [0.56862745, 0.41176471, 0.2745098]])**

A mix of the primary and secondary colors from tue TUE color palette(s), ordered in a way that provides okay(ish) color contrast when plotting on dark background.

tueplots.constants.color.palettes.**tue_primary = array([[0.55294118, 0.17647059, 0.22352941], [0.21568627, 0.25490196, 0.29019608], [0.68627451, 0.70196078, 0.71764706], [0.68235294, 0.62352941, 0.42745098], [0.9372549 , 0.9254902 , 0.88627451]])**

Primary corporate colors of the University of Tübingen.

tueplots.constants.color.palettes.**tue_secondary = array([[0.25490196, 0.35294118, 0.54901961], [0. , 0.41176471, 0.66666667], [0.31372549, 0.66666667, 0.78431373], [0.50980392, 0.7254902 , 0.62745098], [0.49019608, 0.64705882, 0.29411765], [0.19607843, 0.43137255, 0.11764706], [0.78431373, 0.31372549, 0.23529412], [0.68627451, 0.43137255, 0.58823529], [0.70588235, 0.62745098, 0.58823529], [0.84313725, 0.70588235, 0.41176471], [0.82352941, 0.58823529, 0.], [0.56862745, 0.41176471, 0.2745098]])**

Secondary corporate colors of the University of Tübingen.

tueplots.constants.color.rgb module

RGB color constants.

tueplots.constants.color.rgb.**mps_gray = array([0.86666667, 0.87058824, 0.83921569])**

Color associated with the Max Planck Society. Gray.

tueplots.constants.color.rgb.**mps_green = array([0.06666667, 0.4 , 0.3372549])**

Color associated with the Max Planck Society. Green.

tueplots.constants.color.rgb.**mps_lightgray = array([127.93333333, 127.93529412, 127.91960784])**

Color associated with the Max Planck Society. Light gray. 50% version.

tueplots.constants.color.rgb.**mps_lightgreen = array([127.53333333, 127.7 , 127.66862745])**

Color associated with the Max Planck Society. Light green. 50% version.

tueplots.constants.color.rgb.**pn_blue = array([0.16078431, 0.26666667, 0.64705882])**

probnum.org. Blue.

Type
Color associated with ProbNum

tueplots.constants.color.rgb.**pn_gray = array([0.68627451, 0.70196078, 0.71764706])**

probnum.org. Gray.

Type
Color associated with ProbNum

tueplots.constants.color.rgb.**pn_green = array([0.15686275, 0.72941176, 0.70588235])**

probnum.org. Green.

Type
Color associated with ProbNum

tueplots.constants.color.rgb.**pn_orange = array([1. , 0.6, 0.2])**

probnum.org. Orange.

Type
Color associated with ProbNum

tueplots.constants.color.rgb.**pn_red = array([0.55294118, 0.17647059, 0.22352941])**

probnum.org. Red.

Type
Color associated with ProbNum

tueplots.constants.color.rgb.**tue_blue = array([0. , 0.41176471, 0.66666667])**

Color associated with the University of Tübingen. Blue.

tueplots.constants.color.rgb.**tue_brown = array([0.56862745, 0.41176471, 0.2745098])**
Color associated with the University of Tübingen. Brown.

tueplots.constants.color.rgb.**tue_dark = array([0.21568627, 0.25490196, 0.29019608])**
Color associated with the University of Tübingen. Dark gray.

tueplots.constants.color.rgb.**tue_darkblue = array([0.25490196, 0.35294118, 0.54901961])**
Color associated with the University of Tübingen. Dark blue.

tueplots.constants.color.rgb.**tue_darkgreen = array([0.19607843, 0.43137255, 0.11764706])**
Color associated with the University of Tübingen. Dark green.

tueplots.constants.color.rgb.**tue_gold = array([0.68235294, 0.62352941, 0.42745098])**
Color associated with the University of Tübingen. Gold.

tueplots.constants.color.rgb.**tue_gray = array([0.68627451, 0.70196078, 0.71764706])**
Color associated with the University of Tübingen. Gray.

tueplots.constants.color.rgb.**tue_green = array([0.49019608, 0.64705882, 0.29411765])**
Color associated with the University of Tübingen. Green.

tueplots.constants.color.rgb.**tue_lightblue = array([0.31372549, 0.66666667, 0.78431373])**
Color associated with the University of Tübingen. Light blue.

tueplots.constants.color.rgb.**tue_lightgold = array([0.9372549 , 0.9254902 , 0.88627451])**
Color associated with the University of Tübingen. Light gold.

tueplots.constants.color.rgb.**tue_lightgreen = array([0.50980392, 0.7254902 , 0.62745098])**
Color associated with the University of Tübingen. Light green.

tueplots.constants.color.rgb.**tue_lightorange = array([0.84313725, 0.70588235, 0.41176471])**
Color associated with the University of Tübingen. Light orange.

tueplots.constants.color.rgb.**tue_mauve = array([0.70588235, 0.62745098, 0.58823529])**
Color associated with the University of Tübingen. Mauve.

tueplots.constants.color.rgb.**tue_ocre = array([0.78431373, 0.31372549, 0.23529412])**
Color associated with the University of Tübingen. Ocre.

tueplots.constants.color.rgb.**tue_orange = array([0.82352941, 0.58823529, 0.])**
Color associated with the University of Tübingen. Orange.

tueplots.constants.color.rgb.**tue_red = array([0.55294118, 0.17647059, 0.22352941])**
Color associated with the University of Tübingen. Red.

tueplots.constants.color.rgb.**tue_violet = array([0.68627451, 0.43137255, 0.58823529])**
Color associated with the University of Tübingen. Violet.

1.13 Contribution guide

Contributions from the community drive `tueplots`. Are you thinking about contributing to `tueplots`? If yes, that's fantastic! To make it as easy as possible, follow the steps below.

1.13.1 Is my contribution in scope?

`tueplots` welcomes all contributions that concern style-files of scientific papers. Most obviously, new versions of existing style files (e.g., `tueplots` has `icml2023`, so `icml2025`, `icml2026`, or `icml2027` are in scope), but new venues are also interesting (e.g., `tueplots` has ICML's style files, CVPR is also in scope). For anything else, create an issue, and we will have a look together!

1.13.2 Where do I find the style files?

A good source for the concrete values of, e.g., the figure sizes, are the conference's/journal's instructions for the authors. Some values are easy to find; for others, you have to dig deeper. The figure size and font size are usually mentioned fairly prominently. When implementing the font size in Tueplots, use the Figure captions as a reference, not the main text.

The correct font implementation can sometimes be tricky because most templates use 'Times'; however, there are different ways of achieving a "Times"-like font. One often has to look at the source code for the Latex style file to find the correct font. Look out for the following information:

- If the document uses `\includepackage{times}`, then the template uses the 'Times' font. Use the same font configuration as ICLR-2023.
- If the document uses `\includepackage{ptm}`, the template also uses the 'Times' font but with a different package. Tueplots attempts to mirror this choice as closely as possible (by including the same package as the Latex template) even if it means including an obsolete package (the 'times' package is obsolete) .
- If the document includes no font-related package, it likely uses Computer Modern. Copy the configuration from JMLR-2001.

Note 1: The `times` vs `ptm` usage in Tueplots has been clarified by pull request #125 and does not affect version `v0.0.12` or older.

Note 2: Times and Times New Roman are different fonts. And since there is no native Times New Roman implementation in Latex unless one uses XeLatex (see this link), be sceptical if a template claims to use Times New Roman but provides a Latex style file. Always consult the style file's source code to identify the correct font.

1.13.3 How do I make a pull request on GitHub?

Making a pull request on GitHub can seem complicated if one is not used to doing that, but luckily, there are many guides:

- This is GitHub's guide for contributing to open-source projects, including explanations of the terms "cloning" or "forking".
- Here is information about working with `git`.
- And this is how we can all write better commit messages.
- A virtual environment is generally a good idea when working with Python projects (such as `tueplots`). This guide describes the basics. Be aware that if you place your environment inside the `tueplots` root folder, `tueplots`' auto-formatting might attempt to format the source code inside the environment, which is unnecessary.

A small project like `tueplots` might be the perfect place to start contributing to open-source projects. If you want to contribute but need help knowing where to start, reach out!

1.13.4 Where do I put my code?

Once you've got the basic setup (forks, clones, git) going, it is time to write code. Check out the continuous-integration page for information about installing all dev-related tools.

Then, roughly follow these steps:

1. If you like to add a new style file, it would be ideal if you contributed four things: bundles, figsizes, fontsizes, and fonts. The remaining modules (axes, cycler, markers) are usually unimportant for adding a new style file.
2. Add test cases to `tests/test_rc_params_cases/test_{fontsizes,figsizes,fonts,bundles}.py`. Take the existing cases as a reference. The next time you run `tox`, the tests will fail (which is expected).
3. Fix the failing tests by adding the corresponding functions to `tueplots/{fontsizes, figsizes, fonts, bundles}.py`. If you cannot find something, contribute what you can! Now, all tests should pass again.
4. A lot of this will be copying/pasting existing code. Remember to update the docstrings of the pasted code.
5. If you like, add a corresponding cell to one of the example notebooks. This step is optional, but sometimes it helps to see the configuration "in action".

Commit the results and open a pull request. In the pull request's description, mention what it contributes and where to find the information. For example, write something like this:

```
Title: Style-file for ICML2026

Description: This PR adds the bundles, figsizes, fonts, and fontsizes for ICML 2026.

Most values are the same as in previous years; see <link-to-call-for-papers> for␣
↪reference.
```

Once this is done, we will try to process the pull request as quickly as possible.

Thank you for contributing to tueplots!

1.14 Continuous integration

Install `tueplots` with all ci-related dependencies via

```
pip install .[ci]
```

1.14.1 Tox

Run all checks via

```
tox
```

or only run the tests via

```
tox -e test
```

or only run the linter via

```
tox -e format-and-lint
```

1.14.2 Pre-commit hook

The CI checks for compliance of the code with black and isort, and runs the tests and the notebooks. To automatically satisfy the former, there is a pre-commit that can be used (do this once):

```
pip install pre-commit
pre-commit install
```

From then on, your code will be checked for isort and black compatibility automatically.

Both the pre-commit hook and tox point to isort, black, and so on. We do our best to match their versions. If you run into version conflicts between those two tools, please let us know!

1.15 Running the example notebooks

To run the example notebooks, e.g. **this one**, additional dependencies need to be installed via

```
pip install .[examples]
```

For example: `jupyter`.

PYTHON MODULE INDEX

t

INDEX

M

N

O

P

S

T

U

W

X