NIDA
WISDOM *for* Change

DADS 5001

DATA ANALYTICS AND DATA SCIENCE TOOLS AND PROGRAMMING

MATPLOTLIB & SEABORN

Asst.Prof.Thitirat Siriborvornratanakul, Ph.D. (thitirat@as.nida.ac.th)
Graduate School of Applied Statistics, National Institute of Development Administration

1

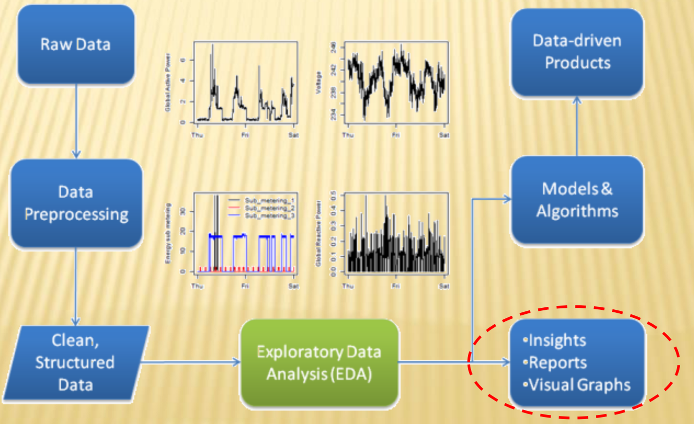
DATA VISUALIZATION

2

2

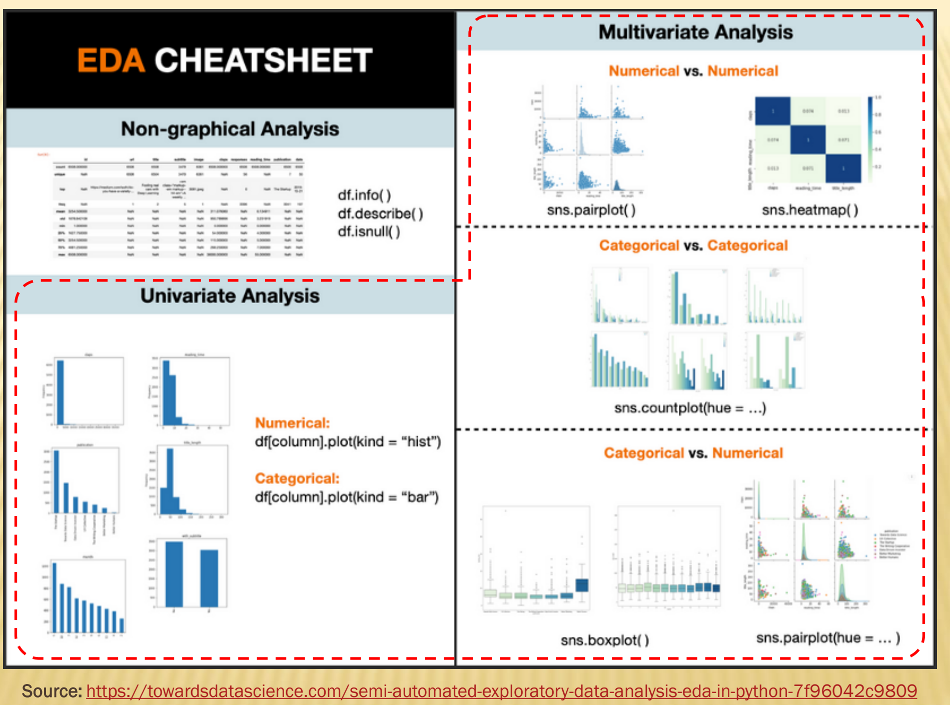
REMINDER

- Originally developed by American mathematician John Tukey in the 1970s, EDA techniques continue to be a widely used method in the data discovery process today.



3

3



Source: https://towardsdatascience.com/semi-automated-exploratory-data-analysis-eda-in-python-7f96042c9809

4

VISUALIZE TO SEE THE NON-OBVIOUS (1/3)

TABLE 1 F-measure evaluation results (d=5) on the four test datasets based on Modified U-Net with different loss functions and normalization layers. The best (maximum) value for each type of normalization is highlighted in bold whereas * highlights the best value in the particular column despite of types of normalization layer.

Model		CrackTree260			CRKWH100			CrackLS315			Stone331		
Norm	**Loss**	**OIS**	**ODS**	**AP**	**OIS**	**ODS**	**AP**	**OIS**	**ODS**	**AP**	**OIS**	**ODS**	**AP**
–	$\mathcal{L}_{bce}$	0.3886	0.2268	0.1312	0.2112	0.1018	0.0545	0.1252	0.0724	0.0666	0.2736	0.1471	0.0932
	$\mathcal{L}_{dice}$	0.2818	0.2111	0.1186	0.1493	0.1157	0.0601	0.1051	0.0724	0.0634	0.1999	0.1028	0.0523
	$\mathcal{L}_{dice\_bce}$	0.3958	0.2225	0.1341	0.2235	0.1019	0.0536	0.1292	0.0731	**0.0674**	0.2739	0.1586	0.0972
	$\mathcal{L}_{iou}$	0.2421	0.1913	0.1155	0.1356	0.1131	0.0592	0.0991	0.0724	0.0626	0.1734	0.0994	0.0508
	$\mathcal{L}_{log\_iou}$	0.1523	0.1516	0.0497	0.1269	0.1099	0.0648	0.1139	**0.0833**	0.0427	0.0724	0.0612	0.0143
	$\mathcal{L}_{bf1}$	**0.6054**	**0.2827**	**0.2081**	**0.4201**	**0.1413**	**0.0833**	**0.1992**	0.0763	0.0432	**0.2938**	**0.1591**	**0.1006**
	$\mathcal{L}_{bf2}$	0.5933	0.2639	0.1843	0.3873	0.1146	0.0669	0.1822	0.0754	0.0398	0.2865	0.1423	0.0881
	$\mathcal{L}_{bf4}$	0.5867	0.2585	0.1840	0.3880	0.1175	0.0696	0.1808	0.0749	0.0401	0.2847	0.1419	0.0870
BN	$\mathcal{L}_{bce}$	0.6498	0.6571	0.2791	0.5674	0.6026	0.2062	0.4379	0.4797	0.1748	0.6602	0.7111	0.6060
	$\mathcal{L}_{dice}$	0.7831	0.7967	0.7593	0.7685	0.8020	0.6180	0.4462	0.6055	0.2956	0.6679	0.7303	0.6850
	$\mathcal{L}_{dice\_bce}$	0.7762	0.7907	0.8010	0.7053	0.7634	0.7647	0.5614	0.6161	0.6119	0.6364	0.6852	0.7320
	$\mathcal{L}_{iou}$	0.7788	0.7906	0.7468	0.7665	0.7982	0.6797	0.4385	0.5822	0.2770	***0.6729***	***0.7415***	***0.7338***
	$\mathcal{L}_{log\_iou}$	0.7827	0.7928	0.8001	0.8160	0.8244	0.8094	0.4586	0.6135	0.4630	0.6362	0.7317	0.7080
	$\mathcal{L}_{bf1}$	**0.8426**	**0.8350**	**0.8688**	**0.8176**	***0.8397***	***0.8314***	***0.7580***	***0.7511***	***0.6856***	0.5579	0.5841	0.2893
	$\mathcal{L}_{bf2}$	0.7666	0.7525	0.6683	0.8066	0.7631	0.6975	0.6378	0.6218	0.4003	0.6067	0.6561	0.5795
	$\mathcal{L}_{bf4}$	0.7547	0.7309	0.6287	0.7810	0.7519	0.7221	0.6085	0.6201	0.4520	0.6253	0.6852	0.6227
IN	$\mathcal{L}_{bce}$	0.8463	0.8568	0.6874	0.7169	0.7516	0.5965	0.5796	0.6148	0.4924	0.3142	0.3080	0.0973
	$\mathcal{L}_{dice}$	0.8625	0.8686	0.8580	0.7353	0.8028	0.7647	0.4617	0.5903	0.4569	0.3475	**0.4540**	**0.4189**
	$\mathcal{L}_{dice\_bce}$	***0.9089***	***0.9120***	***0.9494***	0.7479	0.7303	0.7724	**0.6614**	0.6007	0.4847	0.3471	0.3512	0.1130
	$\mathcal{L}_{iou}$	0.8591	0.8656	0.8535	0.7608	0.8148	**0.7828**	0.4796	0.6150	0.5019	**0.3728**	0.4508	0.4007
	$\mathcal{L}_{log\_iou}$	0.8741	0.8948	0.8721	0.5970	0.7341	0.6594	0.5435	0.6319	0.5160	0.2075	0.2113	0.1021
	$\mathcal{L}_{bf1}$	0.7671	0.7433	0.6562	0.8116	0.7818	0.7251	0.6142	0.5968	0.3967	0.3709	0.3566	0.1648
	$\mathcal{L}_{bf2}$	0.8151	0.8059	0.7721	***0.8352***	**0.8171**	0.7662	0.6597	**0.6578**	**0.5255**	0.2818	0.2607	0.0786
	$\mathcal{L}_{bf4}$	0.7706	0.7445	0.6681	0.8144	0.7566	0.6456	0.6251	0.5997	0.4431	0.3464	0.3200	0.2042

T. Siriborvornratanakul (2023). "Pixel-level thin crack detection on road surface using convolutional neural network for severely imbalanced data," Computer-Aided Civil and Infrastructure Engineering (CACAIE) 5

5

VISUALIZE TO SEE THE NON-OBVIOUS (2/3)

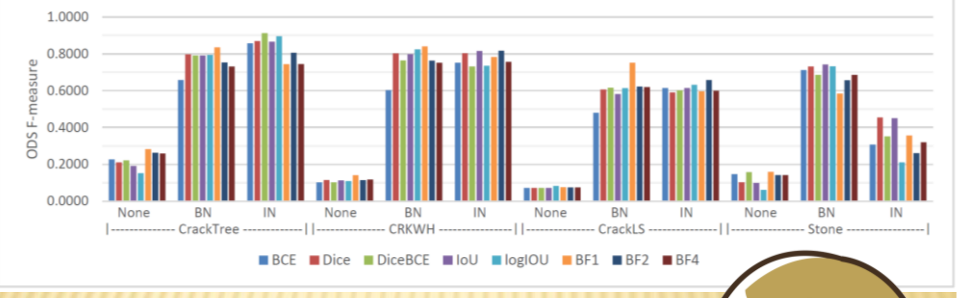
TABLE 1. F-measure evaluation results (d=5) on the four test datasets based on Modified U-Net with different loss functions and normalization layers. The best (maximum) value for each row is highlighted in bold.

NO NORM		$\mathcal{L}_{bce}$	$\mathcal{L}_{dice}$	$\mathcal{L}_{dice\_bce}$	$\mathcal{L}_{iou}$	$\mathcal{L}_{log\_iou}$	$\mathcal{L}_{bf1}$	$\mathcal{L}_{bf2}$	$\mathcal{L}_{bf4}$
CrackTree	**OIS**	0.3886	0.2818	0.3958	0.2421	0.1523	**0.6054**	0.5933	0.5867
	ODS	0.2268	0.2111	0.2225	0.1913	0.1516	**0.2827**	0.2639	0.2585
	AP	0.1312	0.1186	0.1341	0.1155	0.0497	**0.2081**	0.1843	0.1840
CRKWH	**OIS**	0.2112	0.1493	0.2235	0.1356	0.1269	**0.4201**	0.3873	0.3880
	ODS	0.1018	0.1157	0.1019	0.1131	0.1099	**0.1413**	0.1146	0.1175
	AP	0.0545	0.0601	0.0536	0.0592	0.0648	**0.0833**	0.0669	0.0696
CrackLS	**OIS**	0.1252	0.1051	0.1292	0.0991	0.1139	**0.1992**	0.1822	0.1808
	ODS	0.0724	0.0724	0.0731	0.0724	**0.0833**	0.0763	0.0754	0.0749
	AP	0.0666	0.0634	**0.0674**	0.0626	0.0427	0.0432	0.0398	0.0401
Stone	**OIS**	0.2736	0.1999	0.2739	0.1734	0.0724	**0.2938**	0.2865	0.2847
	ODS	0.1471	0.1028	0.1586	0.0994	0.0612	**0.1591**	0.1423	0.1419
	AP	0.0932	0.0523	0.0972	0.0508	0.0143	**0.1006**	0.0881	0.0870
BN		$\mathcal{L}_{bce}$	$\mathcal{L}_{dice}$	$\mathcal{L}_{dice\_bce}$	$\mathcal{L}_{iou}$	$\mathcal{L}_{log\_iou}$	$\mathcal{L}_{bf1}$	$\mathcal{L}_{bf2}$	$\mathcal{L}_{bf4}$
CrackTree	**OIS**	0.6498	0.7831	0.7762	0.7788	0.7827	**0.8426**	0.7666	0.7547
	ODS	0.6571	0.7967	0.7907	0.7906	0.7928	**0.8350**	0.7525	0.7309
	AP	0.2791	0.7593	0.8010	0.7468	0.8001	**0.8688**	0.6683	0.6287
CRKWH	**OIS**	0.5674	0.7685	0.7053	0.7665	0.8160	**0.8176**	0.8066	0.7810
	ODS	0.6026	0.8020	0.7634	0.7982	0.8244	**0.8397**	0.7631	0.7519
	AP	0.2062	0.6180	0.7647	0.6797	0.8094	**0.8314**	0.6975	0.7221
CrackLS	**OIS**	0.4379	0.4462	0.5614	0.4385	0.4586	**0.7580**	0.6378	0.6085
	ODS	0.4797	0.6055	0.6161	0.5822	0.6135	**0.7511**	0.6218	0.6201
	AP	0.1748	0.2956	0.6119	0.2770	0.4630	**0.6856**	0.4003	0.4520
Stone	**OIS**	0.6602	0.6679	0.6364	**0.6729**	0.6362	0.5579	0.6067	0.6253
	ODS	0.7111	0.7303	0.6852	**0.7415**	0.7317	0.5841	0.6561	0.6852
	AP	0.6060	0.6850	0.7320	**0.7338**	0.7080	0.2893	0.5795	0.6227
IN		$\mathcal{L}_{bce}$	$\mathcal{L}_{dice}$	$\mathcal{L}_{dice\_bce}$	$\mathcal{L}_{iou}$	$\mathcal{L}_{log\_iou}$	$\mathcal{L}_{bf1}$	$\mathcal{L}_{bf2}$	$\mathcal{L}_{bf4}$
CrackTree	**OIS**	0.8463	0.8625	**0.9089**	0.8591	0.8741	0.7671	0.8151	0.7706
	ODS	0.8568	0.8686	**0.9120**	0.8656	0.8948	0.7433	0.8059	0.7445
	AP	0.6874	0.8580	**0.9494**	0.8535	0.8721	0.6562	0.7721	0.6681
CRKWH	**OIS**	0.7169	0.7353	0.7479	0.7608	0.5970	0.8116	**0.8352**	0.8144
	ODS	0.7516	0.8028	0.7303	0.8148	0.7341	0.7818	**0.8171**	0.7566
	AP	0.5965	0.7647	0.7724	**0.7828**	0.6594	0.7251	0.7662	0.6456
CrackLS	**OIS**	0.5796	0.4617	**0.6614**	0.4796	0.5435	0.6142	0.6597	0.6251
	ODS	0.6148	0.5903	0.6007	0.6150	0.6319	0.5968	**0.6578**	0.5997
	AP	0.4924	0.4569	0.4847	0.5019	0.5160	0.3967	**0.5255**	0.4431
Stone	**OIS**	0.3142	0.3475	0.3471	**0.3728**	0.2075	0.3709	0.2818	0.3464
	ODS	0.3080	**0.4540**	0.3512	0.4508	0.2113	0.3566	0.2607	0.3200
	AP	0.0973	**0.4189**	0.1130	0.4007	0.1021	0.1648	0.0786	0.2042

T. Siriborvornratanakul (2023). "Pixel-level thin crack detection on road surface using convolutional neural network for severely imbalanced data," Computer-Aided Civil and Infrastructure Engineering (CACAIE) 6

6

VISUALIZE TO SEE THE NON-OBVIOUS (3/3)



T. Siriborvornratanakul (2023). "Pixel-level thin crack detection on road surface using convolutional neural network for severely imbalanced data," Computer-Aided Civil and Infrastructure Engineering (CACAIE)

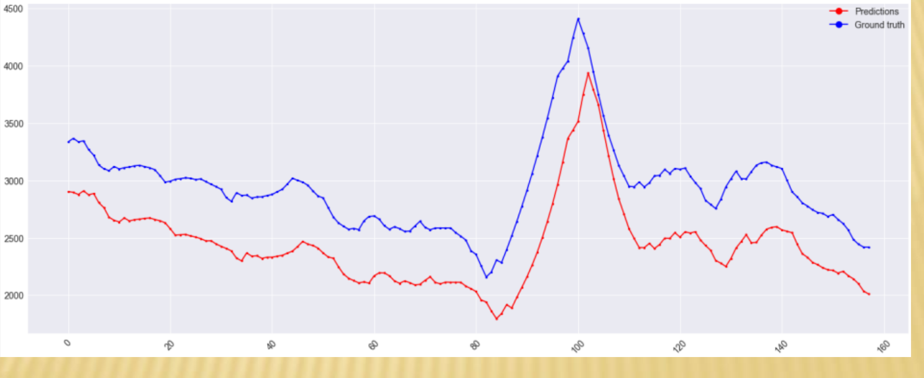
7

7

VISUALIZE TO SEE THE UNEXPECTED

LSTM having a systematic offset between predictions and ground truth

Asked 5 years, 7 months ago   Modified yesterday   Viewed 10k times



https://stackoverflow.com/questions/48286122/lstm-sequence-prediction-in-keras-just-outputs-last-step-in-the-input

8

8

MATPLOTLIB & SEABORN

9

9

REMINDER: PROGRAMMING SEQUENCE IN DADS

The fundamental five of Python's data science stacks

jupyter NumPy pandas matplotlib scikit learn

Python & SQL → Tools → Machine Learning → Text, Image, Real-time → Deep Learning

10

10

MATPLOTLIB (2003) (1/2)

- The first Python data visualization library
- The most popular and widely-used Python package used by data scientists for creating advanced data visualizations
- MATLAB-style 2D plotting library for publication-quality graphics
- A multiplatform (cross-platform) data visualization library built on NumPy arrays.
- Support many OSs, graphics backends, and output types

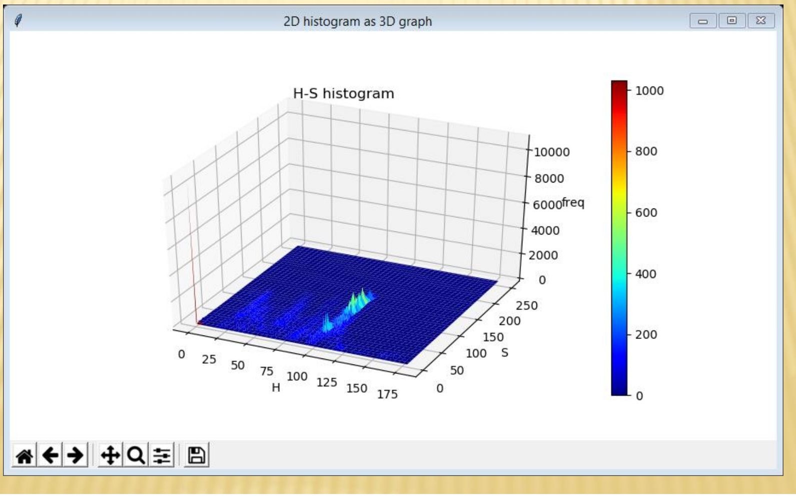
Without a backend explicitly set, Matplotlib automatically detects a usable backend based on what is available on your system and on whether a GUI event loop is already running. The first usable backend in the following list is selected: MacOSX, QtAgg, GTK4Agg, Gtk3Agg, TkAgg, WxAgg, Agg. The last, Agg, is a non-interactive backend that can only write to files. It is used on Linux, if Matplotlib cannot connect to either an X display or a Wayland display.

11

11

MATPLOTLIB (2003) (2/2)

- For 3D graphs, matplotlib does include an alternative toolkit (e.g., mplot3d) that can be used to create 3D graphs.



12

12

MATPLOTLIB'S BACKEND (1/4)

- The "frontend" is the user facing code (i.e., the plotting code) whereas the "backend" does all the hard work behind-the-scenes to make the figure.
- Three types of backends:
  - **Non-interactive/Static backends (hardcopy backends):** the matplotlib renderers, capable of writing to a file.
    - **Renderer:** `AGG`, `PS`, `PDF`, `SVG`, `PGF`, `Cairo`
  - **Interactive backends (user interface backends):** the user interfaces and renderer combinations supported, capable of displaying to the screen and of using appropriate renderers to write to a file:
    - **Backend (case-insensitive):** `QtAgg`, `ipympl`, `GTK3Agg`, `GTK4Agg`, `macosx`, `TkAgg`, `nbAgg`, `WebAgg`, `GTK3Cairo`, `GTK4Cairo`, `wxAgg`

https://matplotlib.org/stable/users/explain/figure/backends.html 13

13

Static backends

Here is a summary of the Matplotlib renderers (there is an eponymous backend for each; these are *non-interactive backends*, capable of writing to a file):

Renderer	Filetypes	Description
AGG	png	raster graphics -- high quality images using the Anti-Grain Geometry engine.
PDF	pdf	vector graphics -- Portable Document Format output.
PS	ps, eps	vector graphics -- PostScript output.
SVG	svg	vector graphics -- Scalable Vector Graphics output.
PGF	pgf, pdf	vector graphics -- using the pgf package.
Cairo	png, ps, pdf, svg	raster or vector graphics -- using the Cairo library (requires pycairo or cairocffi).

To save plots using the non-interactive backends, use the `matplotlib.pyplot.savefig('filename')` method.

https://matplotlib.org/stable/users/explain/figure/backends.html 14

14

Interactive backends

These are the user interfaces and renderer combinations supported; these are *interactive backends*, capable of displaying to the screen and using appropriate renderers from the table above to write to a file:

Backend	Description
QtAgg	Agg rendering in a Qt canvas (requires PyQt or Qt for Python, a.k.a. PySide). This backend can be activated in IPython with `%matplotlib qt`. The Qt binding can be selected via the `QT_API` environment variable; see Qt Bindings for more details.
ipympl	Agg rendering embedded in a Jupyter widget (requires ipympl). This backend can be enabled in a Jupyter notebook with `%matplotlib ipympl` or `%matplotlib widget`. Works with Jupyter `lab` and `notebook>=7`.
GTK3Agg	Agg rendering to a GTK 3.x canvas (requires PyGObject and pycairo). This backend can be activated in IPython with `%matplotlib gtk3`.
GTK4Agg	Agg rendering to a GTK 4.x canvas (requires PyGObject and pycairo). This backend can be activated in IPython with `%matplotlib gtk4`.
macosx	Agg rendering into a Cocoa canvas in macOS. This backend can be activated in IPython with `%matplotlib osx`.
TkAgg	Agg rendering to a Tk canvas (requires TkInter). This backend can be activated in IPython with `%matplotlib tk`.
nbAgg	Embed an interactive figure in a Jupyter classic notebook. This backend can be enabled in Jupyter notebooks via `%matplotlib notebook` or `%matplotlib nbagg`. Works with Jupyter `notebook<7` and `nbclassic`.
WebAgg	On `show()` will start a tornado server with an interactive figure.
GTK3Cairo	Cairo rendering to a GTK 3.x canvas (requires PyGObject and pycairo).
GTK4Cairo	Cairo rendering to a GTK 4.x canvas (requires PyGObject and pycairo).
wxAgg	Agg rendering to a wxWidgets canvas (requires wxPython 4). This backend can be activated in IPython with `%matplotlib wx`.

https://matplotlib.org/stable/users/explain/figure/backends.html

15

15

For non-jupyter-notebook, set the backend by `matplotlib.use("qtagg")` for example.

MATPLOTLIB'S BACKEND (4/4)

Backend	Jupyter notebook	How to display
-	`%matplotlib inline`	Embed a static figure in a Jupyter classic notebook (this is a default mode in Jupyter)
`QtAgg`	`%matplotlib qt`	Render in a Qt canvas
`ipympl`	`%matplotlib ipympl` `%matplotlib widget`	Render embedded in a Jupyter widget (work with Jupyter Lab and notebook>=7)(require additional installation of `pip install ipympl` or `conda install ipympl -c conda-forge`)
`GTK3Agg` `GTK4Agg`	`%matplotlib gtk3` `%matplotlib gtk4`	Render to a GTK 3.x/4.x canvas (require PyGObject and pycairo)
`macosx`	`%matplotlib osx`	Render into a Cocoa canvas in OSX
`TkAgg`	`%matplotlib tk`	Render to a Tk canvas (require TkInter)
`nbAgg`	`%matplotlib notebook` `%matplotlib nbagg`	Embed an interactive figure in a Jupyter classic notebook (work with Jupyter notebook<7 and nbclassic)
`WebAgg`	-	On `show()`, start a tornado server (open a web browser) with an interactive figure
`GTK3Cairo`	-	Render to a GTK 3.x canvas
`wxAgg`	`%matplotlib wx`	Render to a wxWidgets canvas (require wxPython 4)

https://matplotlib.org/stable/users/explain/figure/backends.html

16

16

MATPLOTLIB IN JUPYTER NOTEBOOK (1/2)

Default: inline

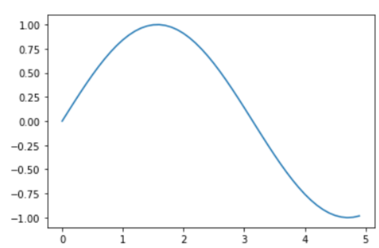
```
# Set matplotlib's backend
%matplotlib inline

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

# Get current backend
print( mpl.get_backend() )

x = np.arange(0,5,0.1)
y = np.sin(x)
plt.plot(x,y)
plt.show()
```

module://ipykernel.pylab.backend_inline



nbAgg

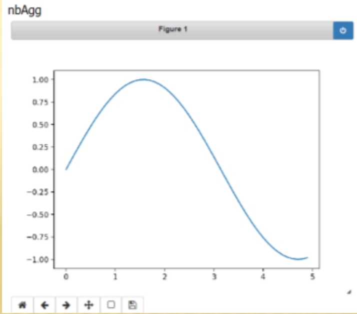
```
# Set matplotlib's backend
%matplotlib notebook

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

# Get current backend
print( mpl.get_backend() )

x = np.arange(0,5,0.1)
y = np.sin(x)
plt.plot(x,y)
plt.show()
```

nbAgg



17

17

MATPLOTLIB IN JUPYTER NOTEBOOK (2/2)

Qt5Agg

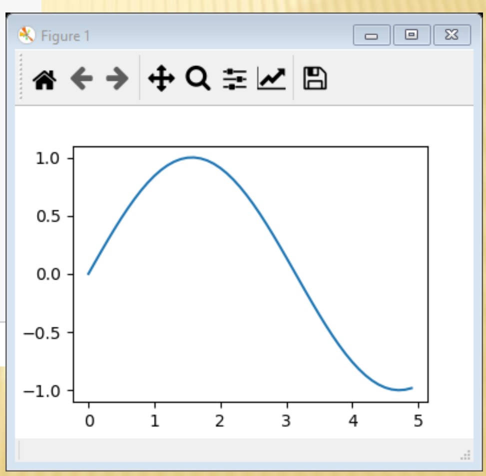
```
# Set matplotlib's backend
%matplotlib qt5

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

# Get current backend
print( mpl.get_backend() )

x = np.arange(0,5,0.1)
y = np.sin(x)
plt.plot(x,y)
plt.show()
```

Qt5Agg

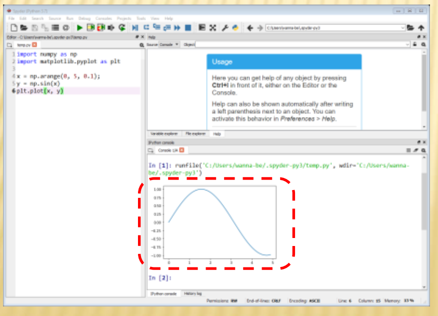


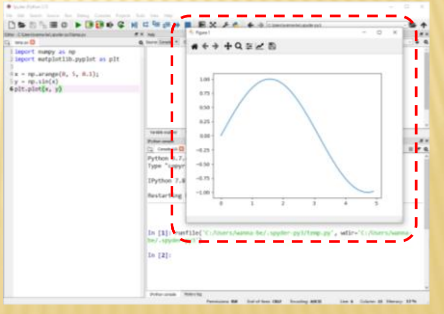
18

18

MATPLOTLIB IN SPYDER

- Inline mode (default).
- Interactive window mode:
  1. *Tools* > *Preferences* > *Ipython console* > *Graphics* tab > change **Graphics backend** from “Inline” to “Automatic” or “Qt” or “Tkinter” > Apply > OK.
  2. Consoles > Restart kernel.





19

19

MATPLOTLIB’S FIGURE

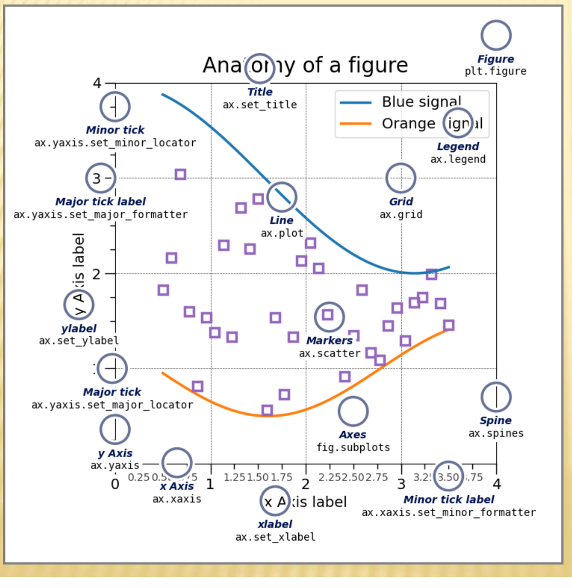
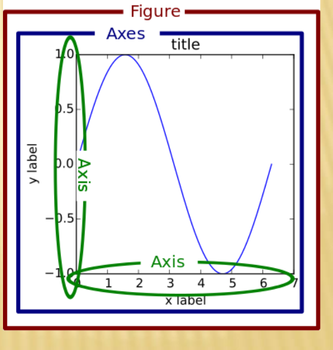


Figure is a single container that contains all the objects (e.g., axes, graphics, text, labels)

Axes is a bounding box with ticks and labels that will contain the plot elements.



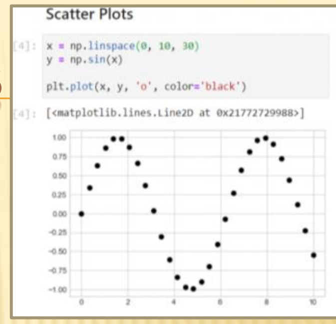
Source: https://matplotlib.org/stable/gallery/showcase/anatomy.html

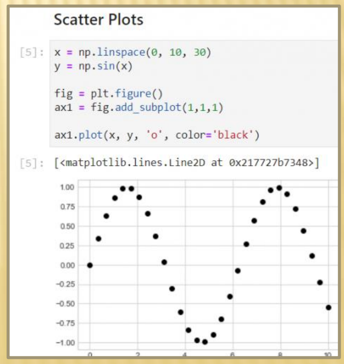
20

20

MATPLOTLIB'S INTERFACES

- The confusion of matplotlib's dual interfaces:
  - MATLAB-style state-based interface:
    - A stateful interface, meaning that it keeps track of the current figure and axes, which are where all matplotlib commands are applied
  - Object-oriented (OO) interface:
    - A more powerful interface for more complicated situations and for when we want more control over our figure
    - Rather than depending on an active figure or axes, the plotting functions in the OO interface are methods of explicit Figure and Axes objects.





21

DRAWBACKS OF MATPLOTLIB

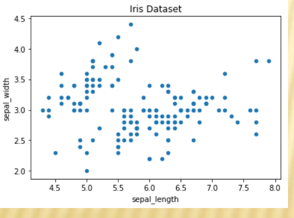
- Matplotlib is extremely powerful but with that power comes complexity.
- Matplotlib's API is relatively low level. Doing sophisticated statistical visualization is possible, but often requires a lot of boilerplate code.
- Matplotlib predated pandas by more than a decade. Thus, it is not directly designed to use with `pandas.DataFrame`.
- To visualize data from `pandas.DataFrame`, we must extract each `pandas.Series` and often concatenate them together into the right format.

22

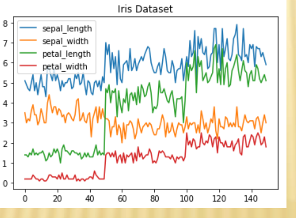
22

MATPLOTLIB (2003) VS. PANDAS (2008)

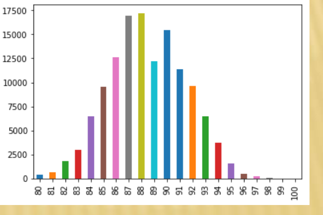
- Pandas visualization makes it really easy to create plots out of `pandas.DataFrame` and `pandas.Series`.
- Pandas visualization has a higher-level API than matplotlib. Therefore, we need fewer codes for the same results.
- The `plot` method on `pandas.Series` and `pandas.DataFrame` is just a simple wrapper around `plt.plot()`, providing a subset of plots as available in matplotlib.



`iris.plot.scatter(x='sepal_length', y='sepal_width', title='Iris Dataset')`



`iris.drop(['class'], axis=1).plot.line(title='Iris Dataset')`



`wine_reviews['points'].value_counts().sort_index().plot.bar()`

Source: https://towardsdatascience.com/introduction-to-data-visualization-in-python-89a54c97fbed

23

23

MATPLOTLIB (2003) VS. SEABORN

- Seaborn = matplotlib + cleaner codes + prettier style + modern APIs
  - Seaborn harnesses the power of matplotlib to create beautiful charts in a few lines of code.
  - The key difference is Seaborn's default styles and color palettes, which are designed to be more aesthetically pleasing and modern.
- Since Seaborn is built on top of matplotlib, you'll need to know matplotlib to tweak Seaborn's defaults.
- Seaborn provides an API on top of matplotlib that
  - Defines simple high-level functions for common statistical plot types
  - Offers sane choices for plot style, color defaults, and beautiful themes
  - Integrates with the functionality provided by pandas
- Both matplotlib and seaborn act as the backbone of data visualization through Python.
  - These two libraries can be used concurrently.
  - However, matplotlib allows more flexible customization.

24

24

END OF THIS CLASS

- News and announcement:
  - Microsoft Teams > General channel
- Class schedule:
  - Microsoft Teams > General channel > Files tab > เอกสารประกอบของคลาส folder > CLASS_SCHEDULE.docx

25

25