Seaborn: Cutting Through the Waves

Author: Siyona Arndt
Date: 05/12/2025

[1] Seaborn is a statistical data representation library. Seaborn works similar to matplotlib and is built on top of it. However, seaborn is optimized for different circumstances. Seaborn was created to work well with pandas which would allow for some automatic formatting of data without having to write a different command to change certain visual aspects of the plot. Seaborn is dataset-oriented, and has a declarative API, allowing for the library to do more of the work in creating a readable figure without you having to input certain commands.

[2] I selected this library because I have really enjoyed exploring new ways of data visualization. I have also come across this method being used in some research I am doing, however, I never actually got to use it. I have however looked into seaborn's features a little before this project. Thus, I wanted to explore seaborn more because it looks like a library that will be very helpful in future research I am doing as well as other physics lab courses.

[3] The first release of seaborn on the seaborn website is for v0.2.0 which was in December of 2013. This is the earliest release I can find. I can't find a release date or information for any version below v0.2.0 Seaborn is built on matplotlib, and also requires pandas, and numPy for installation. For certain more advanced statistical features sciPy and/or statsmodels are required. Fastcluster can also be used. There are many libraries that can plot data such as matplotlib, however, they are all slightly different in their strengths. For example seaborn's strength is to work much smoother with pandas. I am using v0.13.2 of seaborn which was released in January of 2024.

[4] Seaborn was written by Michael Waskom, and appears to still be maintained by him. In the seaborn github in the "workflows" directory within the ".github" directory there is a file called "CONTRIBUTING.md" (https://github.com/mwaskom/seaborn/blob/master/.github/CONTRIBUTING.md) which contains information on how to contribute.

[5, 6] The installation requires numPy, matplotlib, and pandas, and certain features also require statsmodels and/or sciPy. To install seaborn I just used "pip install seaborn" in a jupyter notebook cell. This command will install seaborn and its required dependencies. I had no issues in installing it. And it ran pretty fast, within about 10 - 15 seconds. However, I already had most if not all of the requirements satisfied. To install the more advanced statistical features seaborn's website says to use "pip install seaborn[stats]" however, this did not work and I got an error "zsh:1: no matches found: seaborn[stats]." This was able to be fixed by putting "seaborn[stats]" in single quotes. This took about 5 seconds to fully install, however I already had most requirements satisfied. This also installed the "pasty" package which was not listed as a dependent. Seaborn could also be installed through conda and through the conda-forge channel. This library works well for certain subsets of data sets you will come across. For example one of the nice features about this library is that it automatically adds axis labels based on the data being used, however if the key for the data is not descriptive enough to a reader, you still need to add a new command to change the label. The initial formatting for the graphs is also not very clean if there is a lot of categorical data. However, if the data in the pandas dataframe is set up in a clear and organized way, it is very easy to make a quick plot that looks decent. I believe, if you want more control over your plots with more straightforward commands, matplotlib will work better. But with well organized data, seaborn allows you to visualize data with very little code and in many different ways. Overall, seaborn is easier to create a plot, but matplotlib is easier to create polished plots.

[7] The source code is available at https://github.com/mwaskom/seaborn.

[8] One python package that requires seaborn is plotastic, which tries to incorporate ideas from seaborn and pingouin.

[9] Seaborn is mainly used in a Jupyter notebook, or iPython terminal. Python scripts can also be used but you will have to explicitly write matplotlib.pyplot.show() to see the plots. This is also required if the iPython terminal doesn't have matplotlib mode turned on.

[10] The accompanying Jupyter notebook has the code I created. It shows code used to create a seaborn 1D histogram and a matplotlib style look alike. As well as some other seaborn plots showing its strengths and weaknesses.
This was the only code needed to create the seaborn plot:

```
sns.set_palette('colorblind')
fig1 = sns.displot(Diamonds, x = 'price', hue = 'cut', bins = 50)
fig1.set(title = 'Seaborn - Price of diamonds based on cut type', xlabel = 'price [dollars]')
```

Whereas to make the matplotlib plot a lot more code was needed:

```
ideal = Diamonds[Diamonds['cut'] == 'Ideal']
premium = Diamonds[Diamonds['cut'] == 'Premium']
good = Diamonds[Diamonds['cut'] == 'Good']
very_good = Diamonds[Diamonds['cut'] == 'Very Good']
fair = Diamonds[Diamonds['cut'] == 'Fair']
bins = 50
alpha = .7
fig2, ax2 = plt.subplots()
ax2.hist(ideal['price'], bins = bins, alpha = alpha, label = 'Ideal', color = 'lightblue')
ax2.hist(premium['price'], bins = bins, alpha = alpha, label = 'Premium', color = 'yellow')
ax2.hist(fair['price'], bins = bins, alpha = alpha, label = 'Fair', color = 'purple')
ax2.hist(very_good['price'], bins = bins, alpha = alpha, label = 'Very Good', color = 'orange')
ax2.hist(good['price'], bins = bins, alpha = alpha, label = 'Good', color = 'lightgreen')
ax2.set_title('Matplotlib - Price of diamonds based on cut type')
ax2.set_ylabel('Counts')
ax2.set_xlabel('price [dollars]')
ax2.legend();
```

[11] This package does create figures as that is its whole purpose. It is also built upon matplotlib thus it uses matplotlib.

[12]

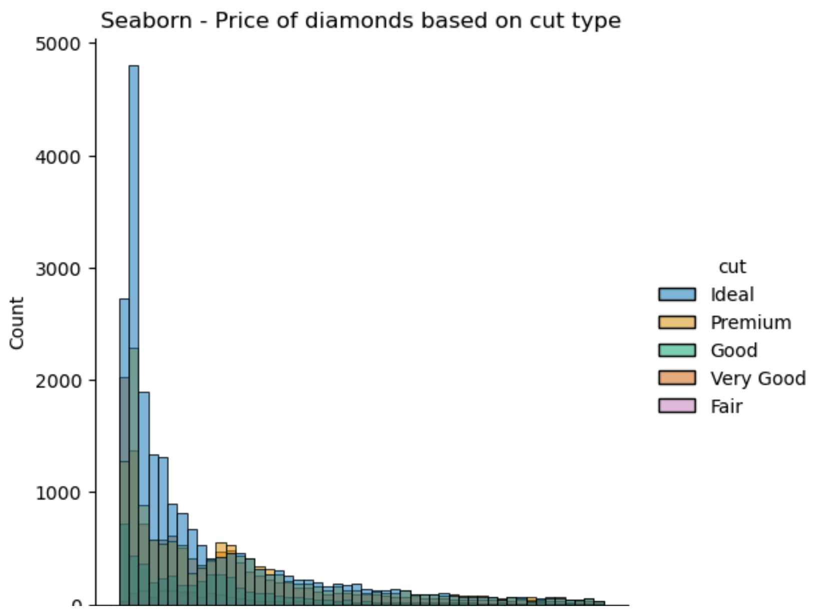

Figure 1: Seaborn plot of price of diamonds based on cut type. This is a seaborn 1D histogram showing the amount of diamonds that cost different amounts. Five different histograms are created for each of the 5 different cut types present in the data. They are all overlaid on top of each other.

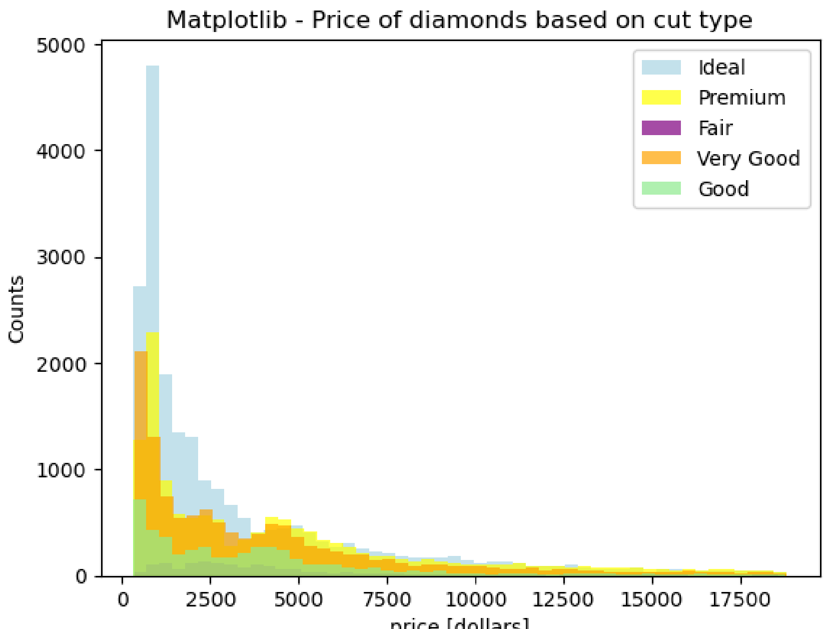

Figure 2: Matplotlib plot of the price of diamonds based on cut type. This is plotting the same data as the seaborn plot in Figure 1 with the same amount of bins. The 5 different histograms for the 5 different cuts present in the data are again overlaid on top of eachother.

[13] Seaborn code is pure python.

[14] The inputs to seaborn are data sets. Usually the input of seaborn is a pandas dataframe or a numPy array. Data created from scratch can be used through using built in python lists and dictionaries.

[15] The output is a figure. Specifically a type of visual representation of data. Many different graph types are available but they are all figures.

[16, 17] The github gives a way to run a test. To run the test more dependencies are required and using the extra
"dev" in the pip install will install those dependencies. To run the test in the source directory you run "make test". This will run unit tests using pytest. It is also recommended to run "make lint" which uses flake8 (which is a tool that uses command line to check the code against the (PEP8) code style) with a setup file in the github. To run automatic checks on code that is being committed for program or style errors (a lint check), "pre-commit" can also be used once pre-commit is installed.

[18] Seaborn uses matplotlib mainly as well as pandas and numPy. I found this on the github page (https://github.com/mwaskom/seaborn?tab=readme-ov-file), as well as the online documentation website (https://seaborn.pydata.org/index.html#), and in the paper "seaborn: statistical data representation" by Michael Waskom (https://joss.theoj.org/papers/10.21105/joss.03021).

[19] Documentation is provided on https://seaborn.pydata.org/index.html#. There is a lot of good information on this website talking about each object that is part of the library. There is also information on the different functions, and statistical tools. There is also information on different aesthetics, as well as many other features. However, while the website does give good information and provide parts of the code that will produce different graphs, I wish each argument in the function were explained more. I also wish the connection to how each command worked with the data was a little more clear from the start. The information is findable but not as easily findable and understandable to someone new to the package as I would have liked. Also while the commands seem simple and straightforward, when it comes to actually writing the code, there are many parameters that are not easy to find if you need to find something small to adjust

[20] The preferred citation is available at https://ui.adsabs.harvard.edu/abs/2021JOSS....6.3021W/exportcitation

[21]

Other package and library links:

- ASCL for seaborn: https://ascl.net/2012.015
- Seaborn github: https://github.com/mwaskom/seaborn?tab=readme-ov-file
- Seaborn website and documentation: https://seaborn.pydata.org/index.html#
- Seaborn paper :https://joss.theoj.org/papers/10.21105/joss.03021
- Github for data: https://github.com/mwaskom/seaborn-data
- Pingouin: https://pingouin-stats.org/build/html/index.html
- numPy: https://numpy.org/
- matplotlib: https://matplotlib.org/
- pandas: https://pandas.pydata.org/, https://pandas.pydata.org/docs/index.html
- sciPy: https://scipy.org/
- Fastcluster: https://danifold.net/fastcluster.html
- Statsmodels: https://www.statsmodels.org/stable/index.html
- Pytest: https://docs.pytest.org/en/stable/
- plotastic: https://github.com/markur4/plotastic
- Flake8: https://flake8.pycqa.org/en/latest/

Links I got information from for answering questions:

- Preferred citation: https://ui.adsabs.harvard.edu/abs/2021JOSS....6.3021W/exportcitation

- Paper 1 that cites seaborn according to ASCL: https://ui.adsabs.harvard.edu/abs/2019ApJ...883L..42F
- Paper 2 that cites seaborn according to ASCL: https://ui.adsabs.harvard.edu/abs/2021MNRAS.503.2380S
- Full list of places that cite seaborn's paper according to ADS: https://ui.adsabs.harvard.edu/abs/2021JOSS....6.3021W/citations
- How I got the seaborn[stats] pip install to work: https://stackoverflow.com/questions/72127673/zsh-no-matches-found-fastapiall/72127733, https://pip.pypa.io/en/stable/user_guide/
- Packages that use seaborn: https://joss.theoj.org/papers/search?q=plotastic&search_button=
- What is Flake8: https://trunk.io/linters/python/flake8
- What is linting: https://www.perforce.com/blog/qac/what-is-linting#:~:text=Linting%20is%20the%20automated%20checking,a%20Unix%20utility%20for%20C.
- What is a declarative API:
- https://www.meshcloud.io/en/blog/should-i-provide-a-declarative-api-you-probably-should/#:~:text=A%20declarative%20API%20is%20a,I%20provide%20will%20be%20there.%22
- Extract rows based on column entry from pandas: https://www.geeksforgeeks.org/how-to-select-rows-from-a-dataframe-based-on-column-values/

[22] Using ASCL only two papers were shown that cited the code https://ui.adsabs.harvard.edu/abs/2019ApJ...883L..42F, and https://ui.adsabs.harvard.edu/abs/2021MNRAS.503.2380S. However, when looking at the ADS citations for the paper "seaborn: statistical data representation" 1614 citations are shown (https://ui.adsabs.harvard.edu/abs/2021JOSS....6.3021W/citations)

[23] I didn't have to learn too much new python code. However, since seaborn works well with pandas, I did have to learn more about pandas. In order to compare the seaborn plot to a matplotlib plot showing something similar I had to learn to extract specific rows based on if a column had a specific value.

[24] Before starting this project I had come across the fact that this library existed before when looking into ways to plot a 2D histogram. I had looked slightly into what seaborn did. However, I never looked into any of the actual code, and I had never actually used it. So this was my first time using it. I did not work in a group.