Task 1: Comparing Visualizations

COURSE NAME

CSDS 413 Introduction to Data Analysis

Authors:

Jacob Anderson
Wiam Skakri

September 22, 2025

Contents

Context

In data science, the choice of visualization plays a critical role in shaping how insights are derived and communicated. Different visual encodings can highlight or obscure structure in data — including spread, skew, outliers, modality, or differences between categories. In this task, you will explore how three different types of visualizations can be used to compare distributions across categories, and evaluate which is most appropriate depending on the dataset context.

For this task, you are provided with three datasets, each containing categorical grouping variables and a numerical measurement. Each dataset comes with a research scenario/question. Your task is to clean the data, visualize the distribution across categories using multiple plotting techniques, and discuss which visualization is the most appropriate in addressing the research question for each dataset.

1 Best-Selling Albums Dataset

Attributes: Year, Ranking, Artist, Album, Genre, Worldwide Sales, Tracks, Album Length

Scenario: A media analytics firm is interested in understanding whether certain genres consistently produce top-selling albums or if success is more scattered across genres.

Research Question: How does the distribution of album sales vary across music genres for albums in the previous decade (released after 2015), and are high-sales outliers concentrated in certain genres?

1.1 Part A: Data Cleaning and Preprocessing

First, filter your dataset so that only the variables critical for your analysis remain. Then clean your data so that there is consistency in variable types, capitalization, and handle any missing or invalid values.
—

The album data set had multiple issues that needed to be addressed. First, we pre-processed the data points by thresholding them based on their 'Year' attribute to include only those released after 2015 because that is the window of interest to this hypothetical firm. Then we removed the features not critical to the analysis of album sales by genre, leaving only 'Worldwide Sales (Est.)' and 'Genre'. On that note, we standardized the column names under snake case as 'album_sales' and 'genres', respectively.

In terms of cleaning the attribute values themselves, we standardized the casing of the genre values and then checked this step by printing a dictionary of genres as keys, mapped to their respective counts. Before this casing step was enforced, printing the dictionary revealed that one genre value was written as "HIp Hop" and had defined two separate keys for the one genre.

The final cleaning step expresses each of the album sale values as an integer data type without any commas so that they would not be conflated as delimiters in the CSV file.

We wrote this utility function to accomplish the task:

```
def clean_preprocess_albums_data(input_csv: str, output_csv: str) -> pd.DataFrame:
    """
    Cleans and preprocesses the albums dataset by removing irrelevant
    features and data points, handling missing/invalid values, standardizing
    capitalization, and converting data types.

    :param input_csv:: Path to the input CSV file containing the albums dataset.
    :type input_csv: str
    :param output_csv: Filename for the clean data.
    :type output_csv: str
    :returns: pd.DataFrame
    :rtype: pd.DataFrame
    """
    df = pd.read_csv(input_csv)

    # Keeps critical variables, relevant data points, removes missing value rows,
    # and renames columns more appropriately
    df = df[df['Year'] > 2015]
    df = df.iloc[:, [4, 7]]
    df.dropna(inplace=True)
    df.columns = ['album_sales', 'genre']

    # Confirms there aren't duplicate genres due to misspelling/invalid vals
    # and standardizes capitalization
    df['genre'] = df['genre'].str.lower()
    print(df['genre'].value_counts().to_dict())

    # Reformats albums sales as integers
    df['album_sales'] = df['album_sales'].str.replace(',', '').astype(int)
```

```
    os.makedirs(os.path.dirname(f'../datasets/clean/{output_csv}'), exist_ok=True)
    df.to_csv(f'../datasets/clean/{output_csv}', index=False)
    return df
```

Figure 1: Best-Selling Albums data pre-processing function.

Below is a comparison of the head of each version to illustrate the changes that were made:

Top_10_Albums_By_Year.csv

```
Year,Ranking,Artist,Album,Worldwide Sales (Est.),Tracks,Album Length,Genre
1990,1,Madonna,The Immaculate Collection,"30,000,000",17,73:32,Pop
1990,2,New Kids On The Block,Step By Step,"20,000,000",12,47:44,Pop
1990,3,Garth Brooks,No Fences,"18,770,000",10,34:34,Country
1990,4,MC Hammer,Please Hammer Don't Hurt Em,"18,000,000",13,59:04,Hip Hop
1990,5,Mariah Carey,Mariah Carey,"15,000,000",11,46:44,Pop
1990,6,Movie Soundtrack,Aashiqui,"15,000,000",12,58:13,World
1990,7,Whitney Houston,I'm Your Baby Tonight,"10,000,000",11,53:45,Pop
1990,8,Phil Collins,Serious Hits... Live!,"9,956,520",15,76:53,Rock
1990,9,Enigma,MCMXC A.D.,"8,838,000",7,40:16,Pop
1990,10,The Three Tenors,Carreras Domingo Pavarotti In Concert 1990,"8,533,000",17,67:55,Classical
...
```

album_sales_by_genre.csv

```
album_sales,genre
7657000,hip hop
6111355,hip hop
4421666,r&b
4207235,pop
4170954,pop
3661560,pop
3462374,pop
3418440,pop
3189149,pop
2727078,pop
...
```

Figure 2: Raw vs. Cleaned Best-Selling Albums dataset.

1.2 Part B: Generate Three Visualizations

Produce the following types of plots:

- **Error Bar Plot:** Show the mean and variability (e.g., standard error or 95% confidence intervals) of the numerical variable across each category.
- **Barcode Chart:** Also known as a strip plot or rug plot. Shows individual data points across categories.
- **Histogram:** Plot the distribution of the numerical variable, grouped by the categorical variable (using hue or facet).

___

Error Bar Plot

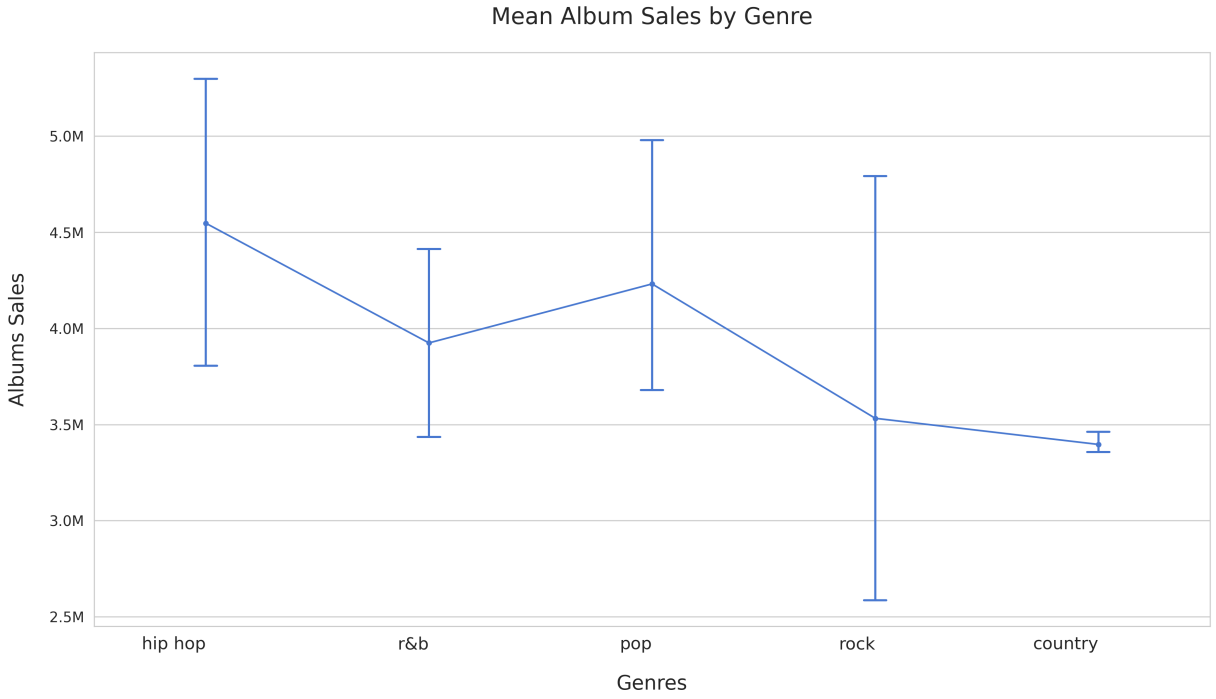

Figure 3: Mean Album Sales by Genre with 95% confidence intervals.

Barcode Chart

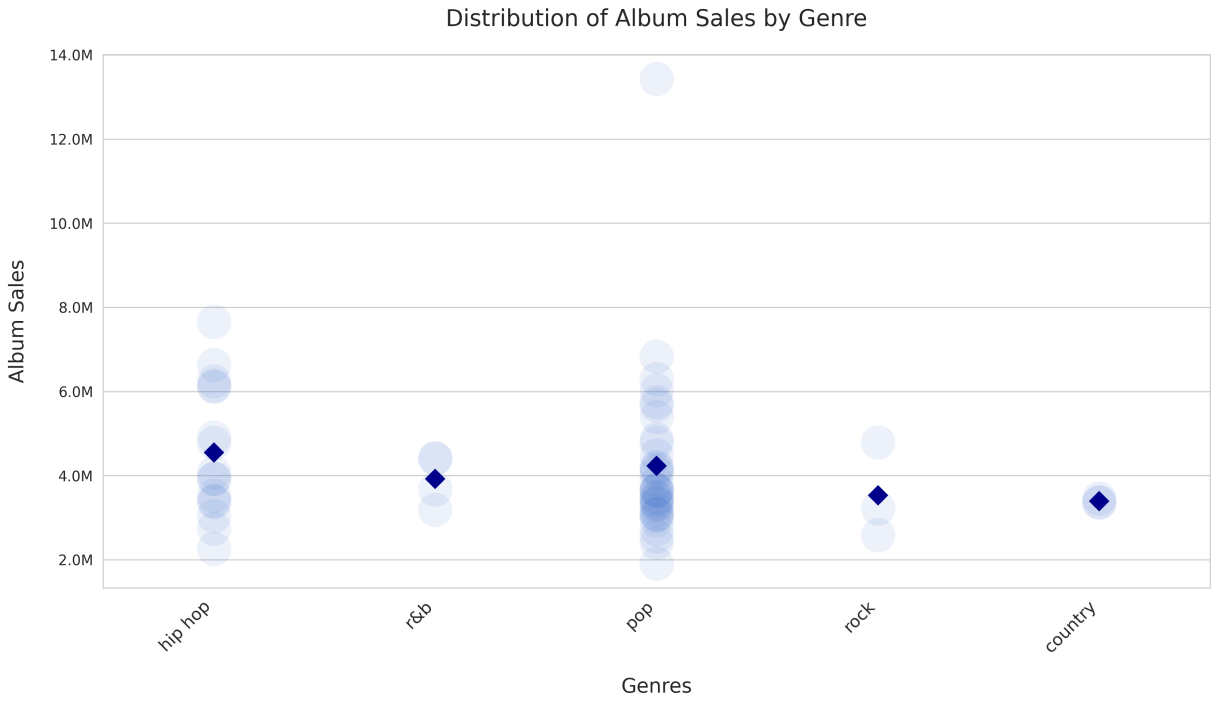

Figure 4: Average Distribution of Album Sales by Genre.

Histogram

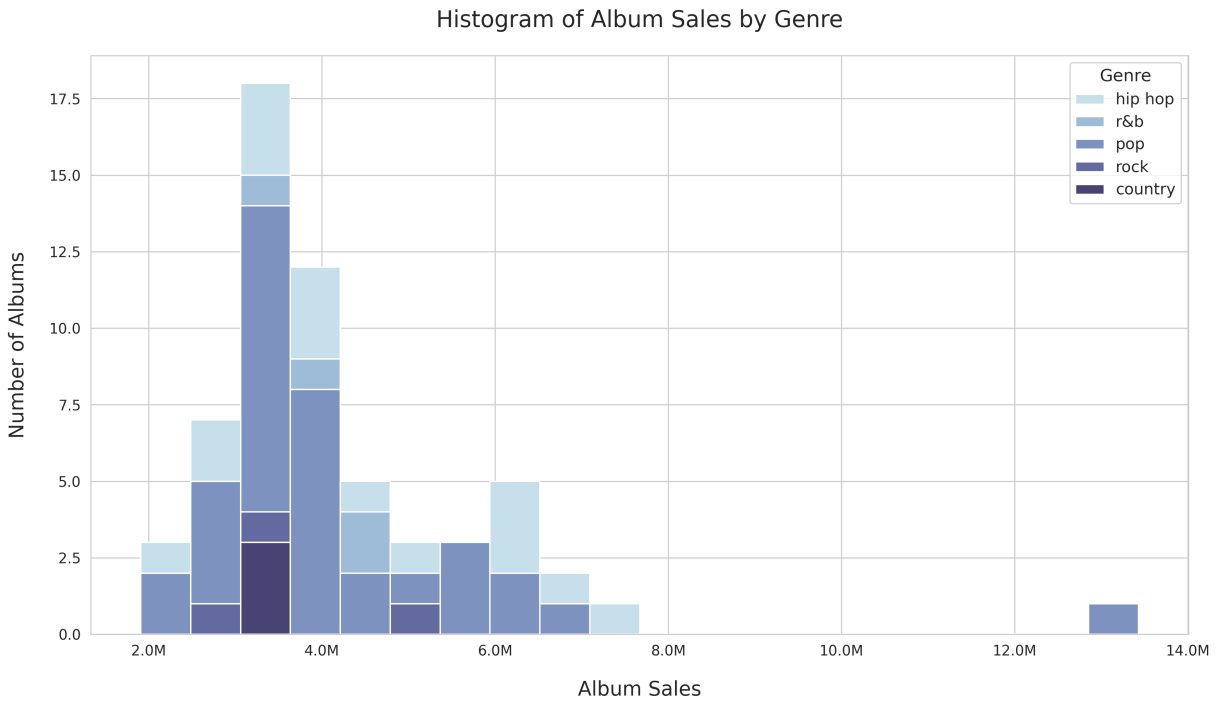

Figure 5: Histogram of Album Sales by Genre.

1.3 Part C: Evaluate and Justify Visualization

For the dataset:

- Discuss the advantages and disadvantages of each visualization type.
- Decide which visualization is best for the research question.
- Support your answer with evidence from the plots and reasoning based on dataset size, shape, or structure.

—

Advantages and Disadvantages

The error bar plot does well to represent comparisons between the sample mean and gives us a notion of uncertainty in population mean. Where the error bar plot loses out, however, is first in its ability to represent outliers/non-general cases. This plot gives us no knowledge of how the mass for each genre is distributed. We can also see in later visualizations that the mass distribution for each genre does look a bit different and so there is necessary context relative to the research question that is lost for this plot. Additionally, while error bars are informative in characterizing general case behavior for many data points, they are not as useful for certain genres. In the cases of genres like Rock or Hip Hop, the representation of the population mean from this plot could very well be anywhere in a rather large range of values, so this plot does not do that well to compare population mean across genres.

The strip plot does a great job at contending with the spread of the data; as we saw in class, the density of the points is a very natural indicator of how the mass of the distribution is laid out across its domain, and we also get on top of that direct knowledge of what outliers exist for each genre and what that behavior looks like comparatively for each genre. We can see in Fig. 4 a relatively extreme, high-sale outlier in the Pop genre that was not obvious before with the error bar plot; an Ed Sheeran album called "Divide" from 2017, achieving just north of 13.4 million sales. This plot also tends to include a mean indicator, which we have included as a distinct, opaque symbol overtop the data points, and this is very helpful in understanding the difference between the extreme behavior and the typical behavior for each genre. The sole issue we have with this visualization is that it does not provide a direct view of the distribution for each genre, as in there is still some amount of interpretation done by the viewer to understand how mass is distributed for each genre. Nevertheless, for this particular context, we are examining a fairly small number of examples (only releases after 2015), so any issues with too much overlap of data points or a lack of clarity in how the mass is distributed is not a huge issue for this dataset.

The histogram is primarily helpful in the exact area where the strip plot can fail, where perhaps the mass of the distribution is hard to interpret due to this overlapping quality of its visuals, and so we get the benefit of most directly viewing how the mass is distributed over the different bins of album sales for each genre. Regarding each genre though, one concern we have is that, regardless of every color palette we tried, reading the behavior of each genre individually is obfuscated due to the shapes of these distributions being not very smooth because of how few samples we are analyzing. We also made the decision to stack the values for each distribution instead of allowing them to overlap because of how many different genres we are observing in the data; allowing them to overlap made the plot very unreadable.

Where we benefit much more with histograms is in examining the extreme values and in understanding the general tendencies of the data. Again, we would argue that strip plots represent these behaviors better for this data, but nonetheless histograms do well to provide us with this context, certainly well enough to make informed claims about the samples. The main struggle here is just with comparing the genres as effectively. We should also mention that histograms, strictly speaking, don't define where the sample mean is nor give a signal to where the population mean may be. Because our data for each genre generally resembles a bell curve, we can take a decent guess as to where the sample mean might be, but it is not directly readable as it is with other plots.

Which visualization is best for the research question

How does the distribution of album sales vary across music genres for albums in the previous decade (released after 2015), and are high-sales outliers concentrated in certain genres? The strip plot is best for answering this research question.

Evidence-based Support for this answer

First, let's compare the information we collect from the error bar plot versus the strip plot. Both give us very clearly the sample mean values for each genre, so when contending with the question of album sales varying across genres, we can equally speak to the general behavior of each genre within this dataset using either plot. Both tell us that Hip-Hop and Pop sells the most albums on average, having a noticeably higher mean, followed by R&B and Rock, and trailed by Country with a noticeably lower average sale performance. We will concede though that the error bar plot does do marginally better to comparatively visualize the mean values across genres due to the fact that it does not have to scale to extreme outliers in the data, allowing it to be more expressive in this respect. Nonetheless, one can still look at the strip plot and immediately make the same comparisons between mean album sales across genres, it is just not as obvious.

Regrading the variance of the data, the strip plot gives us information more relevant to our research question, as we are concerned with high-sales outliers in our data. It is obvious from the shape of our data for each genre that the error bar plot completely misses the primary point to take away from the data in this respect; that being the high-sale outlier in the Pop genre. The strip plot makes it immediately clear and is the obvious choice as far as understanding the whole second part of our research question. Additionally, the error bars, while informative in the case of genres with a lot of exposure and consistence performance in the top 10 over the past decade, struggle to provide as much leverage to make claims about the general behavior of sparser genres that have greater variance. In other words, error bar plots for genres with less data and greater variance in album sales can be quite sensitive/uninformative in terms of characterizing where the population mean may be with 95% confidence. That being said, as the strip plot is the obvious choice over error bar plot for answering the research question.

As far as choosing between the strip plot and the histogram, we confirmed in class and previously from the discussion of advantages and disadvantages that the strip plot mainly loses value in the cases that there is so much data, or perhaps the data is so concentrated toward the distribution's center, that it could be hard to understand the variance for each genre. That is not the case here and the strip plot actually does quite a good job at representing the relative spread in the data across genres; in this case, it's like staring at the distributions from a bird's-eye view, uninhibited by any overlapping masses across genres and expressive enough to readily show qualities like the extra mass of the left side of the pop genre distribution of album sales and the relative difference in variance between rock, pop, and hip hop.

2 Anime Dataset

Attributes: Rank, Name, Japanese_name, Type, Episodes, Studio, Release_season, Tags, Rating, Release_year, End_year, Description, Content_Warning, Related_Mange, Related_anime, Voice_actors, staff

Scenario: A streaming service is considering expanding its short anime series catalog ($<$ 25 episodes) and wants to understand how viewer ratings differ between anime TV series and movies released after 2015. The goal is to determine which format generally receives better audience reception to inform licensing and promotion strategies.

Research Question: How do audience ratings compare between anime TV series and movies released after 2015, and which format generally receives higher ratings?

2.1 Part A: Data Cleaning and Preprocessing

First, filter your dataset so that only the variables critical for your analysis remain. Then clean your data so that there is consistency in variable types, capitalization, and handle any missing or invalid values.
—
We first filtered by Year to keep everything released after 2015, then removed all of the columns irrelevant to the research question, leaving 'Type' and 'Rating'. After doing a drop of all rows with missing values and renaming the columns so they are consistent to how the albums data was set up, there was much whitespace left in the attribute values of the type column, so that was stripped away.

At that point we could force all of the type values to lowercase and read out a dictionary to ensure that there were not any misspelling concerns. Below is the corresponding utility function:

```
def clean_preprocess_anime_data(input_csv: str, output_csv: str) -> pd.DataFrame:
    """
    Cleans and preprocesses the anime dataset by filtering out the irrelevant features
    and datapoints and reformatting the columns names and attribute values

    :param input_csv:: Path to the input CSV file containing the anime dataset.
    :type input_csv: str
    :param output_csv: Filename for the clean data.
    :type output_csv: str
    :returns: pd.DataFrame
    :rtype: pd.DataFrame
    """
    df = pd.read_csv(input_csv)

    # Keeps relevant variables, relevant data poitns, removes the
    # missing value rows, renames the columns, and strips the whitespace
    # out for the type col
    df = df[df['Release_year'] > 2015]
    df = df.loc[:, ['Type', 'Rating']]
    df.dropna(inplace=True)
    df.columns = ['type', 'rating']
    df['type'] = df['type'].str.strip()

    # Standardizes to lower case and filters out irrelevant types
    df['type'] = df['type'].str.lower()
    df = df[df['type'].isin(['tv', 'movie'])]
    print(df['type'].value_counts().to_dict())

    os.makedirs(f'../datasets/clean/', exist_ok=True)
    df.to_csv(f'../datasets/clean/{output_csv}', index=False)
    return df
```

Figure 6: Anime data pre-processing function.

Below is a comparison for this dataset to illustrate the changes:

Anime.csv

```
Rank,Name,Japanese_name,Type,Episodes,Studio,Release_season,Tags,Rating,Release_year,End_year,Descripti
1,Demon Slayer: Kimetsu no Yaiba - Entertainment District Arc, Kimetsu no Yaiba: Yuukaku-hen,TV   ,,ufo
Original Creator, Haruo Sotozaki
Director, Akira Matsushima
Character Design, Aimer
Song Performance","Koyoharu Gotouge : Original Creator, Haruo Sotozaki : Director, Akira Matsushima : C
2,Fruits Basket the Final Season, Fruits Basket the Final,TV   ,13.0,TMS Entertainment,Spring,"Drama, F
Original Creator, Yoshihide Ibata
Director & Episode Director & Storyboard, Taku Kishimoto
Screenplay & Series Composition, Masaru Yokoyama
Music, Masaru Shindou
Character Design & Chief Animation Director, Baek-Ryun Chae
Photography Director, Youko Koyama
Art Director, Mika Sugawara
Color Design","Natsuki Takaya : Original Creator, Yoshihide Ibata : Director & Episode Director & Story
3,Mo Dao Zu Shi 3, The Founder of Diabolism 3,Web  ,12.0,B.C MAY PICTURES,,"Fantasy, Ancient China, Chi
Original Creator, Xiong Ke
Chief Director, Ma Chendi
Chief Director, Sun Yujing
Music, Weng Teng
Music, Feng Shuo
Music, Shen Lin
Character Design & Chief Animation Director, Liang Sha
Screenplay","Mo Xiang Tong Xiu : Original Creator, Xiong Ke : Chief Director, Ma Chendi : Chief Directo
4,Fullmetal Alchemist: Brotherhood, Hagane no Renkinjutsushi: Full Metal Alchemist,TV   ,64.0,Bones,Spr
Original Creator, Yasuhiro Irie
Director, Akira Senju
Music, Hiroki Kanno
2Nd Key Animator & Animation Director & Assistant Animation Director & Character Design & Key Animator,
Producer, Ryou Ooyama
Producer, Nobuyuki Kurashige
Producer, Noritomo Yonai
Producer","Hiromu Arakawa : Original Creator, Yasuhiro Irie : Director, Akira Senju : Music, Hiroki Kan
5,Attack on Titan 3rd Season: Part II, Shingeki no Kyojin Season 3: Part II,TV   ,10.0,WIT Studio,Sprin
Original Creator, Tetsurou Araki
Chief Director, Masashi Koizuka
Director, Tetsuya Wakano
Assistant Director, Yasuko Kobayashi
Series Composition, Hiroyuki Sawano
Music, Kyouji Asano
Character Design, Kazuhiro Yamada
Photography Director","Hajime Isayama : Original Creator, Tetsurou Araki : Chief Director, Masashi Koiz
...
```

anime.csv

```
type,rating
tv,4.6
tv,4.6
tv,4.57
tv,4.56
tv,4.56
...
```

Figure 7: Raw vs. Cleaned Anime dataset.

2.2 Part B: Generate Three Visualizations

Produce the following types of plots:

- **Error Bar Plot:** Show the mean and variability (e.g., standard error or 95% confidence intervals) of the numerical variable across each category.
- **Barcode Chart:** Also known as a strip plot or rug plot. Shows individual data points across categories.
- **Histogram:** Plot the distribution of the numerical variable, grouped by the categorical variable (using hue or facet).

___

Error Bar Plot

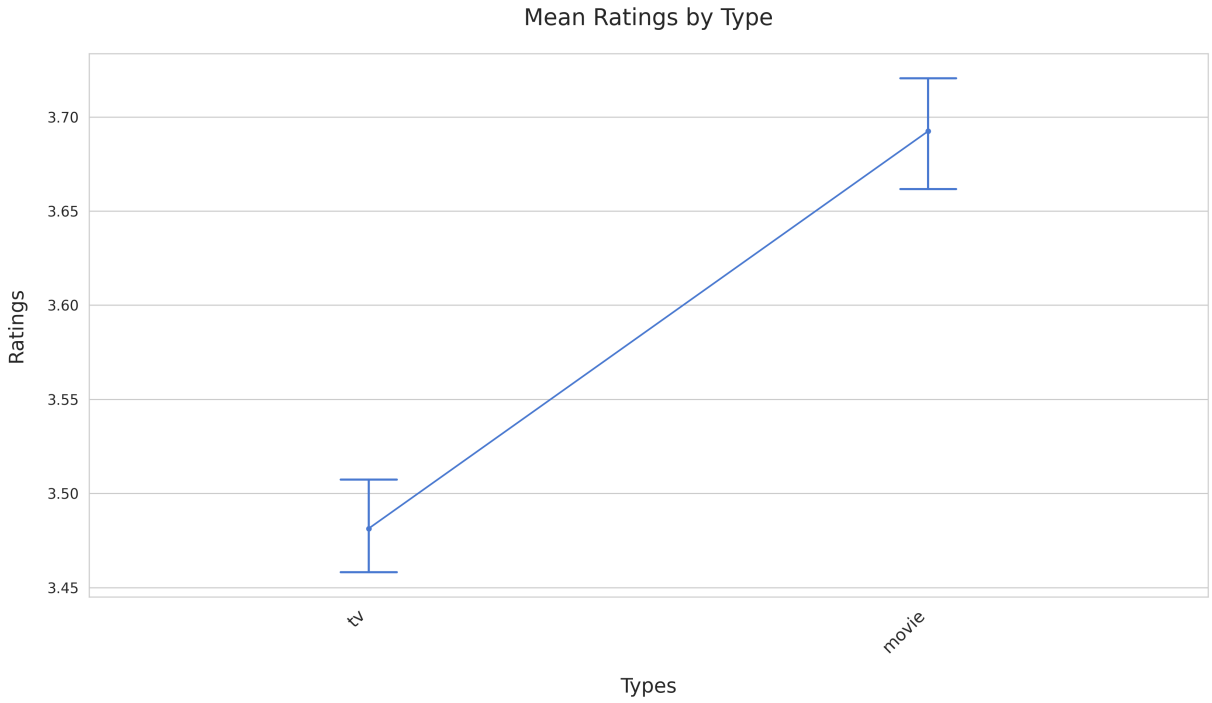

Figure 8: Mean Ratings by Type with 95% confidence intervals.

Barcode Chart

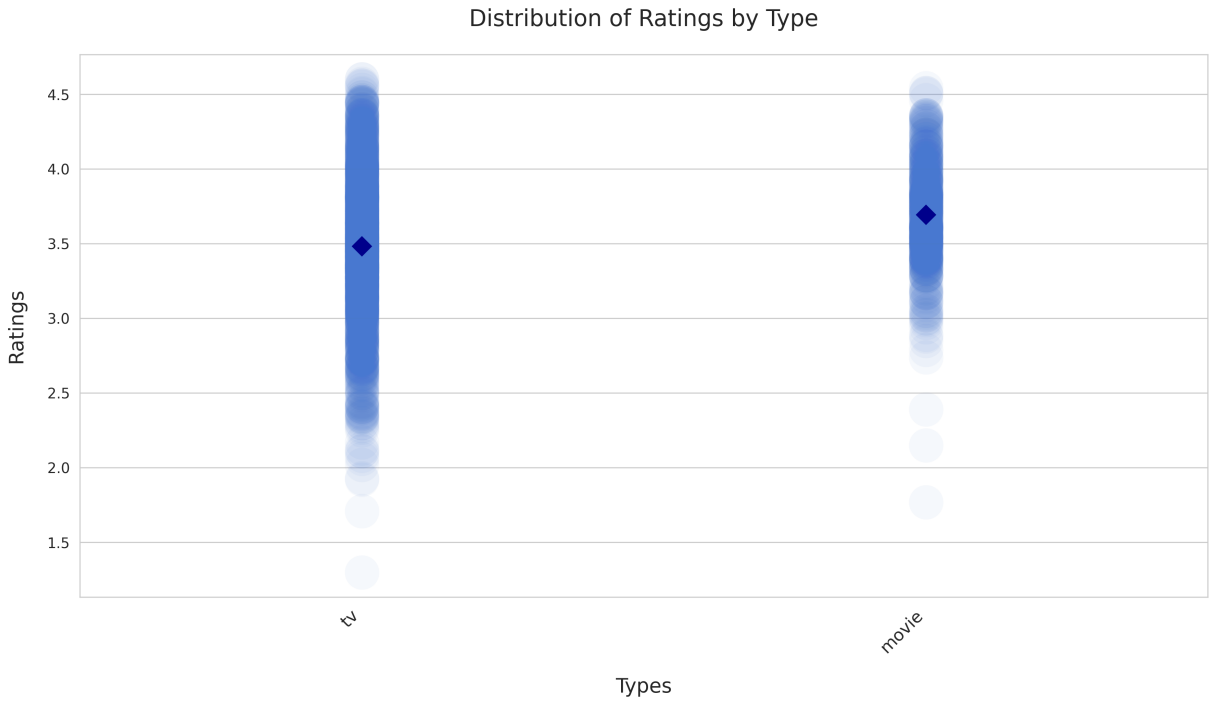

Figure 9: Average Distribution of Ratings by Type.

Histogram

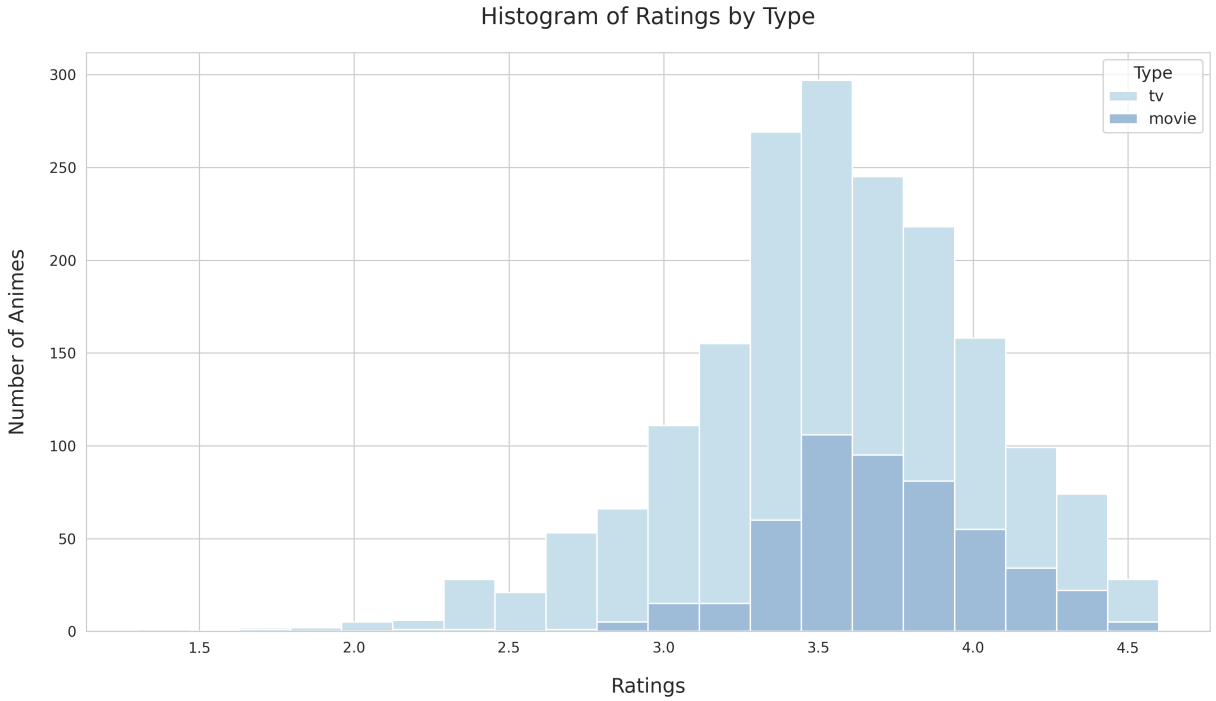

Figure 10: Histogram of Ratings by Type.

2.3 Part C: Evaluate and Justify Visualization

For the dataset:

- Discuss the advantages and disadvantages of each visualization type.
- Decide which visualization is best for the research question.
- Support your answer with evidence from the plots and reasoning based on dataset size, shape, or structure.

—

Advantages and Disadvantages
For this dataset, the confidence intervals for the two types of media are much more comparable in size, so error bar plots in this case give a much better comparison between population mean along with sample mean. Like we established earlier, this plot does not characterize how the mass is distributed for each media type and so it does not scale to extreme values, meaning the comparison of these means is much more expressive in this plot.

The strip plot is facing the exact issue we discussed for the previous dataset, namely the concern that for many data points, the distribution of the mass can be hard to read from the plot at values close to the sample mean or otherwise where many of the points are concentrated. There is also the issue of scale, as this plot visualizes the ratings across all of the sample points in the dataset, which means that reading how the sample means compare for this plot is a bit more difficult as the visual difference between them is not as obvious. It does at least show the range of ratings for each media type and it does give us an idea of the mass distribution at more extreme values.

The histogram plot for this dataset suffers the challenge of gauging average rating just from signals that the plot gives rather than the plot communicating it directly, whereas the other plots give explicit metrics of it. For that reason, comparing the categories in this way is quite challenging, though we do at least get to see the distribution of mass for each category, and for a bell curve, that gives us a decent signal of where the sample means may be.

Which visualization is best for the research question

How do audience ratings compare between anime TV series and movies released after 2015, and which format generally receives higher ratings? The error bar plot is best for answering this research question.

Evidence-based Support for this answer
The error bar plot is most suitable for this research question because this question is concerned with comparing the general rating behavior of anime TV series and anime movies, and which one yields higher ratings. We are given the averages for each media type for the dataset, which shows a clear preference by audiences toward anime movies, with an average of around 3.69 versus an average of 3.48. The error bars representing a 95% confidence interval show that with near certainty the population mean for anime movies is higher; it is a near definitive answer to the second part of our research question. Like we mentioned earlier, it has a y-scale that gives us a very clear comparison of these sample means, and given how relatively tight the error bars are, we can extract immediately from this visual which one is greater.

The strip plot gives us the sample means for each genre, but since it is scaled to the whole range of ratings seen in the dataset; from around 1.0 to 4.8. The difference in sample means between these media types is not as easy to read as those in the error bar plot. However, we are provided with a view of the mass distribution for each media type, but what remains is the issue of data scale, so even then this benefit is ruined by the fact that discerning how the data is massed between different points is challenged around the sample means. You can see in Fig. 9 that for anime TV series the mass is hard to discern from around 2.5 to 4.5, and for anime movies it is hard to discern from 3.4 to 4.2.

As far as making any claims about the data as it relates to the research question, we have from the strip plot an unstable foundation from which to speak on how the audience ratings compare across the media formats, and since strip plots don't give us a notion of confidence in the sample means like the error bar does, we also have an unstable foundation from which to speak on which generally receives higher ratings. All we can speak on is what this specific data says, and given how close the means are across the media types relative to the range of ratings that were observed, this plot alone cannot refute the argument that perhaps we are only seeing a

higher average rating for anime movies from this observation, whereas the population might tell a different story.

Regarding the histogram, we can see very clearly how the mass is distributed for each media format, allowing us to make claims such as how each one is slightly left-skewed, so we know people are probably mentally rating more from 2.0 to 5.0 rather than using the whole scale, or how ratings peak for each media type at around 3.5, or how they both have similar spreads about their means.

A similar fault in this visualization is that discerning the means of each distribution and gauging how representative these samples are to their respective populations is not something that this histogram is good for. We are not told directly what the sample means are for each media format, and even then, the sample means are so similar that telling which is generally receiving higher ratings is infeasible. Not to mention, even if we had the samples means from this plot, the histogram also lacks any notion of uncertainty regarding how representative the respective samples means are of the mean over the whole population. We would argue then that the histogram is not suitable for answering either part of this research question.

3 Algorithm Performance Dataset

Attributes: Algorithm, Epoch, Accuracy, Trial Number

Scenario: You are testing two reinforcement learning (RL) algorithms on a sequential decision task. To avoid overfitting and simulate real-world noise, you shuffle the dataset for each trial and run 10 independent trials per algorithm. For each trial, you track the accuracy across 10 training epochs (one pass through a dataset). Due to how you shuffle your data and algorithmic stochasticity, accuracy results vary across trials.

Research Question: Which algorithm performs more accurately on average across epochs, and how does the use of a visualization help you assess reliability and variation of each algorithm?

3.1 Part A: Data Cleaning and Preprocessing

First, filter your dataset so that only the variables critical for your analysis remain. Then clean your data so that there is consistency in variable types, capitalization, and handle any missing or invalid values.

—

The scenario specified that we are interested in epochs 1-10 over 10 trials, so we first filtered out any epochs and/or runs beyond that, for which there was one epoch 11 in the raw data. After also dropping any rows with missing accuracies, for which there were a few, we then finished by standardizing the format of the Algorithms columns and renaming the columns, to be consistent with the other two clean datasets.

The Algorithms column had inconcsistent casing of the word 'Algorithm', which is also irrelevant entirely to grabbing the information of which algorithm we are observing, so the casing was standardized and the term was removed from all of the values in this row. There was also whitespace in some cases, so each record was stripped, and we were left with clean 'a'/'b' labels to work with.

Below is the corresponding utility function:

```
def clean_preprocess_algorithms_data(input_csv: str, output_csv: str) -> pd.DataFrame:
    """
    Cleans and preprocesses the algorithms dataset by constraining the trials and epochs
    to the first 10 and then cleans the inconsistent entry of algorithm labels.

    :param input_csv:: Path to the input CSV file containing the algorithm trials dataset.
    :type input_csv: str
    :param output_csv: Filename for the clean data.
    :type output_csv: str
    :returns: pd.DataFrame
    :rtype: pd.DataFrame
    """
    df = pd.read_csv(input_csv)

    df.dropna(inplace=True)
    df = df.loc[:, ['Epoch', 'Algorithm', 'Run', 'Accuracy']]

    # Contrains to trial and epoch values within 1-10
    df = df[(df['Epoch'] >= 1) & (df['Epoch'] <= 10)]
    df = df[(df['Run'] >= 1) & (df['Run'] <= 10)]

    # Standardizes algorithms att value format
    df['Algorithm'] = df['Algorithm'].str.strip()
    df['Algorithm'] = df['Algorithm'].str.lower()
    df['Algorithm'] = df['Algorithm'].str.replace('algorithm ', '')

    df.columns = ['epoch', 'algorithm', 'run', 'accuracy']

    os.makedirs(f'../datasets/clean/', exist_ok=True)
    df.to_csv(f'../datasets/clean/{output_csv}', index=False)
    return df
```

Figure 11: Algorithm Performance data pre-processing function.

Below is a comparison for this dataset to illustrate the changes:

algorithm_trials.csv

```
,Epoch,Algorithm,Run,Accuracy
0,1, algorithm a ,1,0.0464
1,2, algorithm a ,1,0.0069
2,3, algorithm a ,1,0.0992
3,4, algorithm a ,1,0.241
4,5, algorithm a ,1,
5,6, algorithm a ,1,0.4813
6,7, algorithm a ,1,0.8574
7,8, algorithm a ,1,0.9422
8,9, algorithm a ,1,0.915
9,10, algorithm a ,1,1.0
...
```

algo_accuracy_by_epoch.csv

```
epoch,algorithm,run,accuracy
1,a,1,0.0464
2,a,1,0.0069
3,a,1,0.0992
4,a,1,0.241
6,a,1,0.4813
7,a,1,0.8574
8,a,1,0.9422
9,a,1,0.915
10,a,1,1.0
1,a,2,0.0
...
```

Figure 12: Raw vs. Cleaned Algorithm Performance dataset.

3.2 Part B: Generate Three Visualizations

Produce the following types of plots:

- **Error Bar Plot:** Show the mean and variability (e.g., standard error or 95% confidence intervals) of the numerical variable across each category.
- **Barcode Chart:** Also known as a strip plot or rug plot. Shows individual data points across categories.
- **Histogram:** Plot the distribution of the numerical variable, grouped by the categorical variable (using hue or facet).

Error Bar Plot

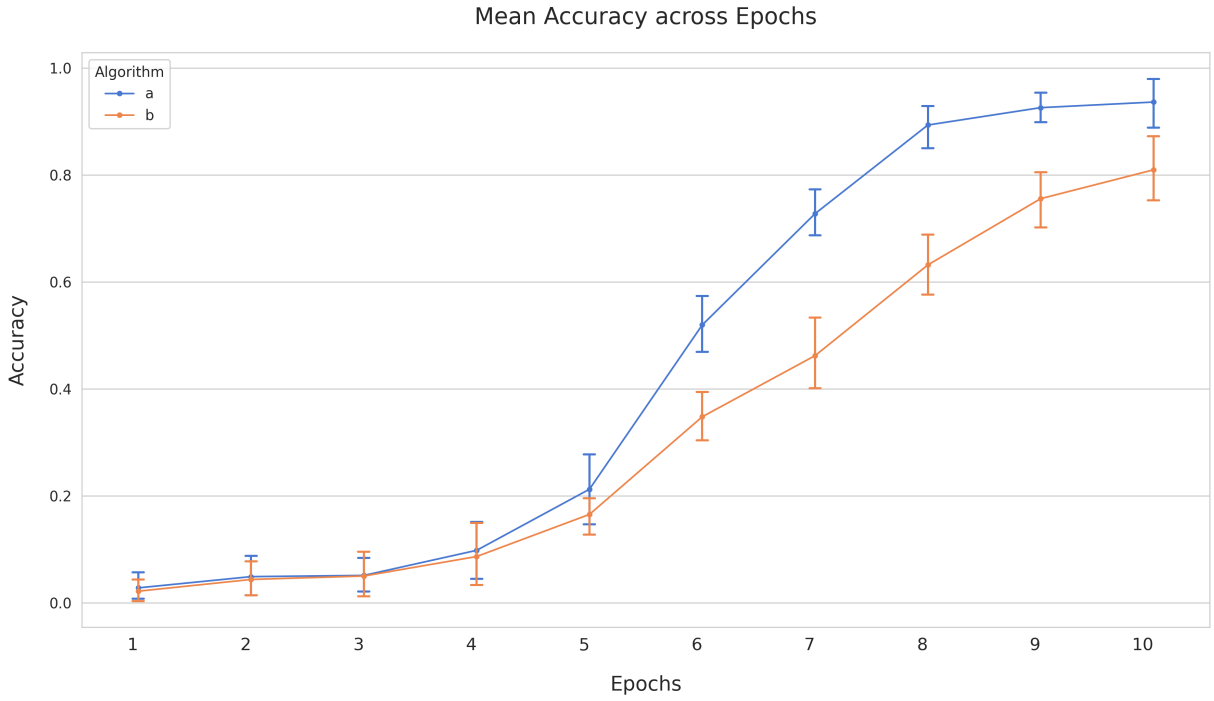

Figure 13: Mean Accuracy across Epochs by Algorithm with 95% confidence intervals.

Barcode Chart

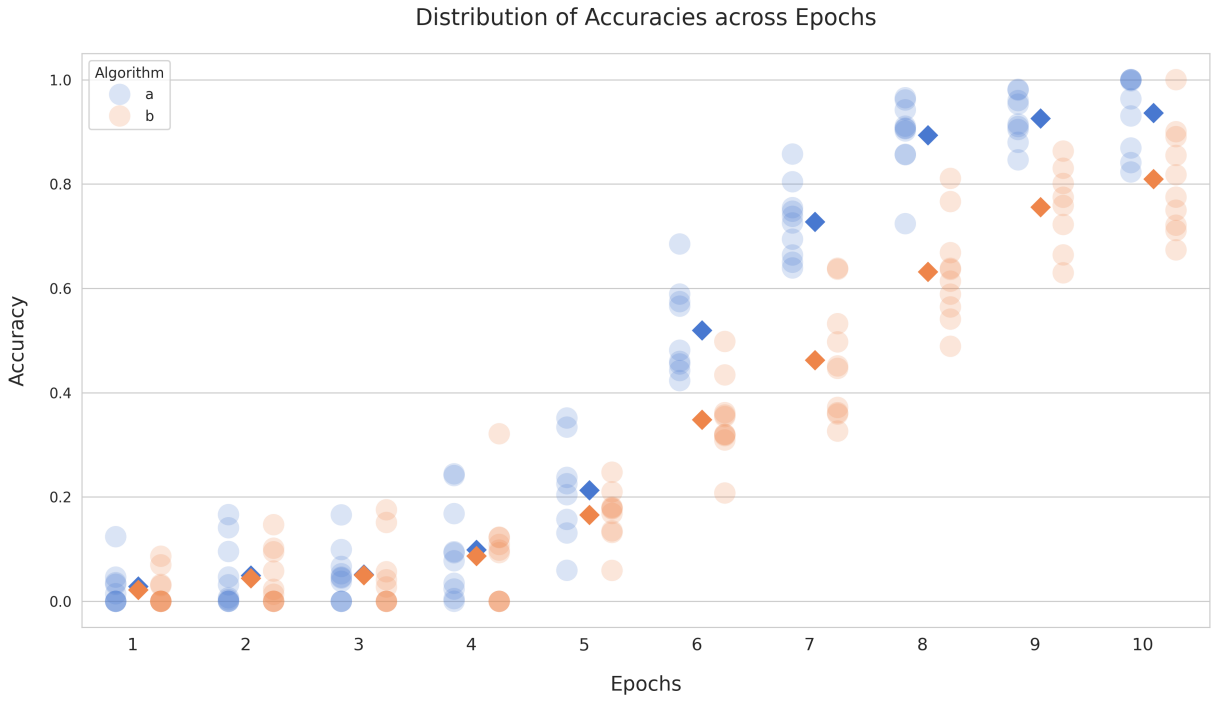

Figure 14: Average Distribution of Accuracy across Epochs by Algorithm.

Histogram

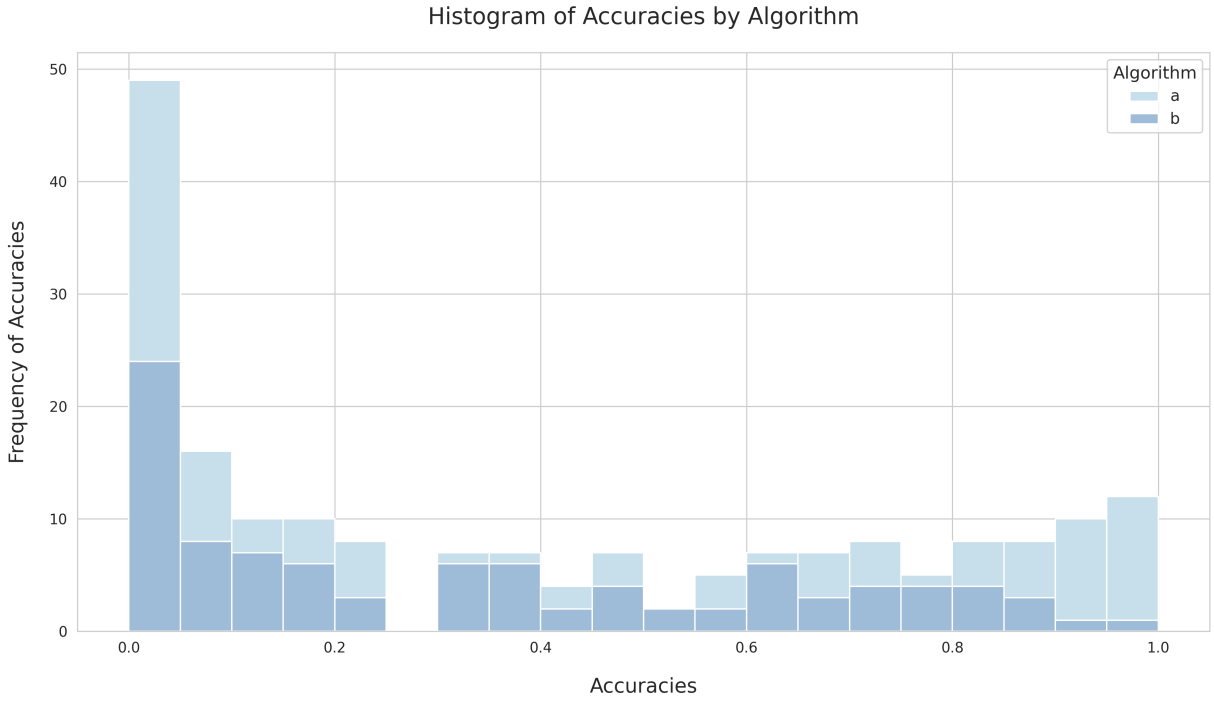

Figure 15: Histogram of Accuracies by Algorithm.

3.3 Part C: Evaluate and Justify Visualization

For the dataset:

- Discuss the advantages and disadvantages of each visualization type.
- Decide which visualization is best for the research question.
- Support your answer with evidence from the plots and reasoning based on dataset size, shape, or structure.

—

Advantages and Disadvantages
The error bar plot shows sample means plotted across each epoch, one plot for each algorithm. This gives us a clear comparison of the average accuracy between the two algorithms over the training cycle, and in some of the later epochs, gives an informative suggestion of how much more accurate algorithm A becomes over algorithm B. The error bar plot is quite uninformative in the whole first half of the training however. The comparable size of the error bars between the two algorithms means that we are not told much at all in terms of their difference from epochs 1-5. Nonetheless, we do get a decent representation of each algorithm's reliability and variation in performance, as the errors bar signal how volatile the performance can be across separate training runs.

The strip plot does well to show us the variation in performance for the specific training runs observed in the dataset, providing very similar information to the error bar plot. Like the album sales dataset, we are not concerned with too many observations at each epoch, so the mass distribution is quite readable from the plot, and we can easily read the difference in average performance across epochs. One key benefit of this visualization though is the fact that we have knowledge of extreme outliers in the training runs for each epoch, which for a sparse dataset, allow us to speak on the average performance across epoch for each algorithm with an increased level of nuance i.e. algorithm A at epoch x shows a higher/lower accuracy than B, but possesses extreme outliers at y% accuracy, so we may require additional data to make a more sound comparison of each algorithm at this stage of the training.

The histogram plot was quite challenging to visualize for this dataset, as binning by algorithm and plotting accuracy frequency by accuracy bins loses any context of how performance develops over the training cycle, which leaves us with a weak comparison of the performance of the algorithms by themselves. This plot is overall quite weak for this context, but we can extract from the distribution of mass for each algorithm ideas about where inflection points and extreme values in the accuracy development over the epochs may have occurred.

For some arbitrary algorithm plotted in this manner, let's say its data showed a generally linear increase in accuracy over 10 epochs. Then, we can expect its histogram visualization to show a very uniform distribution of mass. Now, consider there was an inflection in its accuracy; likely in the earlier stages of training before a learned pattern has emerged for the model. We may see accuracy bin(s) where there is no mass in the histogram, as they were skipped over in the training cycle. Additionally, if for some trials there were some number of extreme outliers, we would see additional mass distributed to the appropriate bins roughly corresponding to other epochs. However, the fact remains that we are left without direct context of performance across epochs, so even the claims we could try to support from this visualization would be weak.

Which visualization is best for the research question

Which algorithm performs more accurately on average across epochs, and how does the use of a visualization help you assess reliability and variation of each algorithm? The strip plot is best for answering this research question.

Evidence-based Support for this answer
The error bar plot is quite a strong visualization for this data with respect to the research question: we can assess the reliability and performance variation of each algorithm from the samples means and the somewhat tight errors bars at every stage of the training. We can speak to qualities such as how there is strong support that algorithm A very reliably achieves an accuracy of around 0.9 and levels out starting at around its seventh epoch, really converging after the eighth. We can also make claims about how the data confidently suggests that algorithm A performs more accurately in the latter half of training, and that the error bars at the last epoch suggest that there at least some possibility that the true means of each may be converging or that algorithm B could surpass A given a longer training, and so we have insights with which to motivate further analysis. It's also worth mentioning that the error bars are a very direct signal of reliability of the algorithms and are

powerful in letting us make claims about the comparative accuracies over the training, which is very relevant to the research question.

An issue that remains with the error bar plot however is the small scale of our observations for each epoch. We saw with the album sales dataset that for a small number of examples the variance in those observations can have a great amount of influence over how large the error bars have to be to remain 95% confident, and so our understanding of average case behavior of each algorithm at each epoch can be potentially misguided. It is not so much of an issue here as the observations do not contain too many outliers, but there are enough to suggest that there is necessary context missing that would help us answer our research question more effectively.

This is context we are provided by the strip plots. For example, in epoch 4 for algorithm B (orange) we see that the observations are separated into three different groupings, and the accuracy of algorithm A is just above that of B. The highest outlier at this epoch for algorithm B is reflected in a relatively wide error bar, but now equipped with this additional context, we can make a more accurate claim about how the average case accuracy compares at this epoch. With more examples, we may see a temporary higher average accuracy for algorithm B before the inflection of A hits in epoch 6, or we may confirm the lower mass densities generally seen for this epoch. Getting to the directly the mass distribution in this scenario provides so much leverage in understanding the reliability of each algorithm as well, as color density directly relates to a notion of how often the algorithm yields nearly the same accuracy at each epoch. For this reason, it is very similar to the abilities of the error bar plot, but with a higher degree of granularity befitting a sparse dataset.

The histogram plot as mentioned earlier is a very clear loser for this data context. There is nothing to directly identify from this plot; even the strong claims we can make do not come from metrics directly identified by the visualization, but are only suggested by how the mass is distributed for each accuracy bin. For example, we could identify the lack of any mass in the bin 0.25-0.3, for which we can see a clear inflection point if we reference one of the other visualizations, occurring between epoch 5 and 6. So we can at least identify potentially a mutual inflection point in the training performance for both algorithms, but we cannot speak to exactly when it occurs without the context of one of the other visualizations. We can also suggest that each algorithm's performance increases slowly to begin with due to the increase mass for each concentrated in the lowest bin, but it remains to be seen that this plot provides any useful information regarding how the accuracy of each algorithm compares over the course of the training.

Task 2: Fitting and Comparing Distributions

COURSE NAME

CSDS 413 Introduction to Data Analysis

Authors:

Wiam Skakri
Jacob Anderson

September 22, 2025

Contents

Context

In this task, you will explore how different types of real-world datasets may follow different distributions. You will need to develop a set of hypotheses and perform experiments to validate your own hypotheses.

1 Normal Distribution Dataset

1.1 Part A: Developing Hypotheses

Identify and collect a real-world dataset that you hypothesize follows a Normal distribution. Please be clear about the reasoning behind your hypothesis and be specific about the source of the dataset.

—

We have collected the Iris dataset for this example. Specifically, we will use the sepal widths of the Setosa species in the dataset. We hypothesize this data to follow a Normal distribution because the sepal widths of Setosas ought to be influenced by many underlying environmental factors. These factors could be anything from the composition of the soil, the level of shade, the composition of the surrounding ecosystem, the agricultural practices of surrounding human civilizations, etc. More generally, notions of performance in a homogenous population often approximate a Gaussian, as most observations will tend to distribute about the mean, symmetrically deviating from the general behavior on each side of the spectrum, becoming less frequent at extreme conditions. The Iris dataset was originally produced by Ronald Fisher in 1936 and was sourced from this page on Kaggle.

As our hypothesize relates to the Gaussian distribution PDF:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

The model defines a mean parameter μ that the distribution mass is centered around, reflecting the idea that the sepal widths are going to average to some value over the observations. The exponential term $e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$ defines the decay behavior for the Gaussian, dictated by the standard deviation parameter σ. The difference between the observation x and μ is squared so that deviations from the mean decay exponentially and symmetrically about the center, which are then scaled relative to the variance σ^2, i.e. the general proclivity of a population to deviate from its mean.

In other words, if the distribution of mass for some population tends to deviate from the mean naturally (higher σ^2), then the decay for observations far from the mean is much slower. This captures the natural variance of the observations in our data, and the bell shape reflects the fact that many Setosas ought to have generally similar sepal widths, and extreme cases ought to be less and less common the more extreme the observations get.

1.2 Part B: Fitting Distributions

For this exercise, we will call each of the four different theoretical distributions (normal, uniform, power law, exponential) a “model". Fit the dataset (i.e., estimate the model parameters) against each model (not just the one you hypothesized) using maximum likelihood estimation (or using any technique you think is appropriate; make sure to comment on the validity of your approach). This should result in a total of **4 parameter sets**. Report the estimated parameters in the following tabular format:

		Model			
Dataset	***# Observations***	**Normal**	**Uniform**	**Power law**	**Exponential**
Dataset 1	n_1	μ_1, σ_1	a_1, b_1	$\alpha_1, x_{\min_1}$	λ_1

Be sure to show the code you used to arrive at your final estimates clearly.

—

Below is the code that produced the parameter estimates and the tabulated estimates for this dataset (for the full tabulation described in the original assignment TeX, see Appendix):

```
def dists_fit(input_csv: str) -> tuple:
    """
    Fits the obs dataset to each model using MLE.

    :param input_csv: Path to input data to fit paramater(s) to
    :type input_csv: str
    """
    obs = pd.read_csv(input_csv).iloc[:, 0].to_numpy()

    mu = np.mean(obs)
    std = np.sqrt(np.sum((obs - mu) ** 2) / len(obs))

    a, b = obs.min(), obs.max()

    alpha = 1 + len(obs) / np.sum(np.log(obs / a))

    lamb = 1 / np.mean(obs)

    return (mu, std, a, b, alpha, lamb)
```

Figure 1: Parameter estimation function for the four models.

		Model			
Dataset	***# Observations***	**Normal**	**Uniform**	**Power law**	**Exponential**
Iris Sepal Widths (Setosa)	50	3.4, 0.4	2.3, 4.4	3.6, 2.3	0.3

Figure 2: Parameter estimates for each model on Iris Sepal Widths (Setosa).

1.3 Part C: Comparing Real and Synthetic Data

For each fitted distribution (there will be 4 of them for this dataset, each corresponding to a different model), generate a synthetic sample of data points equal to the sample size of the real dataset using the respective model parameters you inferred from the real dataset.

Compare the real vs. synthetic data distributions using methods you think are the most appropriate, including visualizations. So, for this dataset, we compare the original dataset to four synthetic datasets, all with equal number of observations, but each synthetic dataset is generated using a different model.

For this dataset, identify the synthetic dataset (which corresponds to a model) that is most similar to the original data in terms of its distribution.

Now revisit your initial hypothesis. For this dataset: Did the dataset behave as expected, or was another model (assumed distribution) a better fit to the dataset? Reflect on why the observed results may differ from your expectations.

—

Below is the code that produced the synthetic datasets for each fitted distribution:

```
def dists_generate(input_csv: str, params: tuple, dist_type: str):
    """
    Generates N synthetic examples

    :param input_csv: Path to input dataset
    :type input_csv: str
    :param params: tuple of parameters returned by dists_fit (mu, std, a, b, alpha, lamb)
    :type params: tuple
    :param dist_type: name of the folder correponsding to a type of distribution
    :type dist_type: str
    """
    obs_df = pd.read_csv(input_csv)
    obs_name = obs_df.columns[0]
    obs = obs_df.iloc[:, 0].to_numpy()
    n = len(obs)

    mu, std, a, b, alpha, lamb = params

    gaussian_samples = int(np.random.normal(loc=mu, scale=std, size=n))
    uniform_samples = int(np.random.uniform(low=a, high=b, size=n))
    powerlaw_samples = int((a * (1 - np.random.uniform(0, 1, n)) ** (-1 / (alpha - 1))))
    exponential_samples = int(np.random.exponential(scale=1/lamb, size=n))

    gaussian_df = pd.DataFrame({f'{obs_name}_gaussian': gaussian_samples})
    uniform_df = pd.DataFrame({f'{obs_name}_uniform': uniform_samples})
    powerlaw_df = pd.DataFrame({f'{obs_name}_powerlaw': powerlaw_samples})
    exponential_df = pd.DataFrame({f'{obs_name}_exponential': exponential_samples})

    gaussian_df.to_csv(f'../datasets/{dist_type}/synth/{obs_name}_gaussian.csv', index=False)
    uniform_df.to_csv(f'../datasets/{dist_type}/synth/{obs_name}_uniform.csv', index=False)
    powerlaw_df.to_csv(f'../datasets/{dist_type}/synth/{obs_name}_powerlaw.csv', index=False)
    exponential_df.to_csv(f'../datasets/{dist_type}/synth/{obs_name}_exponential.csv', index=False)
```

Figure 3: Sampling function for fitted distributions.

We believe an appropriate test for these synthetic distributions would be the K-S statistic because we can equip ourselves with a notion of disagreement between the distributions in terms of their maximum difference between the cumulative probability structure of the actual dataset. The K-S test implemented in SciPy also provides the p-value for our comparisons, so before evening looking at the distributions we can gauge the disagreement between synthetic and real and the extent to which that disagreement is structurally significant or if it's just a product of random noise.

We can see in the table below that immediately the Power Law and Exponential models stand out tremendously with K-S statistics above and around 0.5, both of which are disagreements that are very structural in nature. On the other hand, the Uniform and Normal models show much less disagreement in their CDFs relative to the real sample, however the difference in p-value is very telling of which is actually representative of the data. The 0.04 p-value for the Uniform distribution tells us immediately that there is almost no chance, a 3.9% probability, that such as disagreement could come from two samples in the same distribution, whereas the p-value for the Normal suggests there you could see this K-S statistic with about a 96.7% chance if the two samples were pulled from the same distribution.

Without even observing the other two visualizations we prepared, we can make quite strong claims about which models do well to represent that Sepal Widths dataset and which do not, though it is always helpful to pair these statistics with a visualization that makes the comparison much more readily obvious. In Fig 5, we observe that the Power Law and Exponential models at least begin to converge, climbing a little bit of the probability mass of the real data, but the Uniform and Normal models clearly outmatch them with central tendencies much closer to the real sample.

Distribution	K-S Statistic	p-value	Significant
Normal	0.10	0.967	No
Uniform	0.28	0.039	Yes
Power Law	0.48	1.39×10^{-5}	Yes
Exponential	0.64	6.08×10^{-10}	Yes

Figure 4: Kolmogorov-Smirnov test results between Sepal Widths dataset and synthetic samples.

The histogram also makes it readily obvious the disparity between the Uniform and Normal. While the Uniform performs in a different class than the previous two models, it remains apparent from the mass distribution that the Uniform distribution is more right-skewed and not as centered about its mean as the real sample.

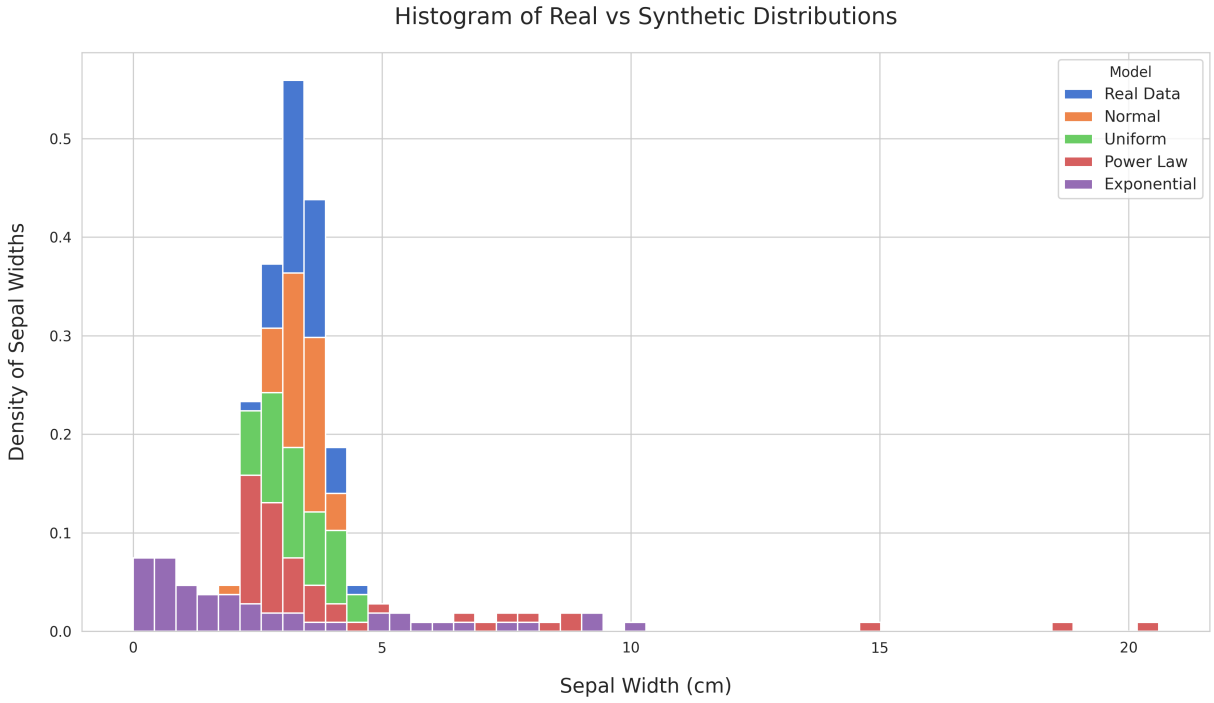

Figure 5: Histogram of Real vs Synthetic Distributions on Sepal Widths dataset.

To better confirm the comparison between the Uniform and Normal Distribution, we also curated a Q-QPlot of the data over 25 quantiles to gauge the agreement between the real and synthetic samples across the entire distribution. What we are left with is a general agreement of each model with the real sample's quantiles for average-case behavior, but the Uniform shows an obvious "snaking" pattern about the agreement line that worsens toward the extremes of the distributions.

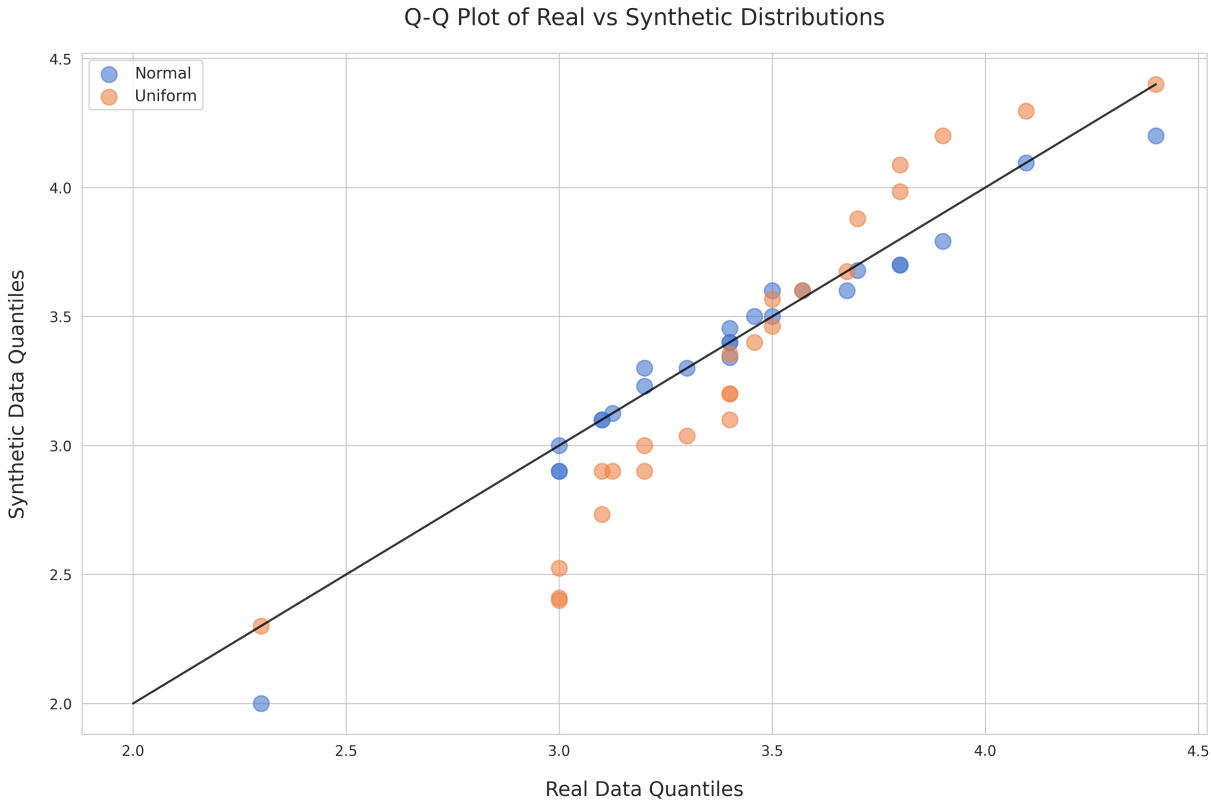

Figure 6: Q-Q Plot comparing Real vs Synthetic quantiles on Sepal Widths dataset.

Given these comparisons, we can make a strong claim that the sepal_widths_gaussian synthetic dataset is most similar to the real data in terms of its distribution. This aligns with our original claim that the Sepal Widths dataset contains notions of central tendency and a spread about its mean that decays toward extreme values.

2 Uniform Distribution Dataset

2.1 Part A: Developing Hypotheses

Identify and collect a real-world dataset that you hypothesize follows a Uniform distribution. Please be clear about the reasoning behind your hypothesis and be specific about the source of the dataset.

—

For this example, we sourced a dataset of 20-sided (d20) die rolls from Kaggle. The dataset contains the outcomes of 145 rolls of a single, unweighted d20. We hypothesize that this data follows a Uniform distribution because an unweighted die ought to have an equal probability of landing on any of its faces.

That would make repeated d20 rolls a clear example of a discrete uniform distribution, where all possible values in its range are equally massed. The probability mass function for a discrete uniform distribution from a to b (inclusive) is given by:

$$P(X = x) = \frac{1}{b - a + 1}, \quad \text{for } x \in \{a, a + 1, \ldots, b\}$$

where a is 1 and b is 20, so each face of the die ought to have a probability of around:

$$P(X = x) = \frac{1}{20} = 0.05$$

Because the d20 has a finite outcome space and possesses a notion of fairness, we expect the Uniform distribution to be the most appropriate model for representing this sample. It specifies exactly the parameters necessary to characterize this environment; nothing more imposed. The die doesn't not possess notions of general behavior or extreme values nor does it possess a unique spread of mass within different ranges of outcomes; it simply traverses the outcome space freely, bounded only by the values it can take on.

2.2 Part B: Fitting Distributions

For this exercise, we will call each of the four different theoretical distributions (normal, uniform, power law, exponential) a “model". Fit the dataset (i.e., estimate the model parameters) against each model (not just the one you hypothesized) using maximum likelihood estimation (or using any technique you think is appropriate; make sure to comment on the validity of your approach). This should result in a total of **4 parameter sets**. Report the estimated parameters in the following tabular format:

		Model			
Dataset	***# Observations***	**Normal**	**Uniform**	**Power law**	**Exponential**
Dataset 2	n_2	μ_2, σ_2	a_2, b_2	$\alpha_2, x_{\min_2}$	λ_2

Be sure to show the code you used to arrive at your final estimates clearly.

—

Below are the tabulated parameter estimates for this dataset (for the full tabulation described in the original assignment TeX, see Appendix). The code in Fig. 1 was also used to arrive at our final parameter estimates for this dataset, here is the same implementation below for convenience:

```
def dists_fit(input_csv: str) -> tuple:
    """
    Fits the obs dataset to each model using MLE.

    :param input_csv: Path to input data to fit paramater(s) to
    :type input_csv: str
    """
    obs = pd.read_csv(input_csv).iloc[:, 0].to_numpy()

    mu = np.mean(obs)
    std = np.sqrt(np.sum((obs - mu) ** 2) / len(obs))

    a, b = obs.min(), obs.max()

    alpha = 1 + len(obs) / np.sum(np.log(obs / a))

    lamb = 1 / np.mean(obs)

    return (mu, std, a, b, alpha, lamb)
```

		Model			
Dataset	***# Observations***	**Normal**	**Uniform**	**Power law**	**Exponential**
D20 Rolls	145	11.117, 6.055	1, 20	1.459, 1	0.090

Figure 7: Parameter estimates for each model on D20 Rolls dataset.

2.3 Part C: Comparing Real and Synthetic Data

For each fitted distribution (there will be 4 of them for this dataset, each corresponding to a different model), generate a synthetic sample of data points equal to the sample size of the real dataset using the respective model parameters you inferred from the real dataset.

Compare the real vs. synthetic data distributions using methods you think are the most appropriate, including visualizations. So, for this dataset, we compare the original dataset to four synthetic datasets, all with equal number of observations, but each synthetic dataset is generated using a different model.

For this dataset, identify the synthetic dataset (which corresponds to a model) that is most similar to the original data in terms of its distribution.

Now revisit your initial hypothesis. For this dataset: Did the dataset behave as expected, or was another model (assumed distribution) a better fit to the dataset? Reflect on why the observed results may differ from your expectations.

—

To assess the synthetic data versus the real sample given our Uniform hypothesis, we need an understanding of how evenly distributed the probability mass is of the real sample and each of the model samples. That makes a histogram plot sound very appealing, however some of the other models, namely the exponential model and especially the power law model, fit toward very infrequent extreme values, which make it rather hard to visualize bins of die roll values in this way. Additionally, the Normal model possesses the unique quality of fitting toward a symmetric distribution, allowing for negative roles at the extremes of its distribution. That being said, we first take a look at their K-S statistics tabulated below:

Distribution	K-S Statistic	p-value	Significant
Normal	0.11	0.341	No
Uniform	0.12	0.273	No
Power Law	0.30	5.00×10^{-6}	Yes
Exponential	0.21	0.004	Yes

Figure 8: Kolmogorov-Smirnov test results between D20 Die Rolls dataset and synthetic samples.

Interestingly enough, the K-S statistic accompanied with p-values shows that the Power Law model and the Exponential model are unsuitable for representing the real data die rolls, showing relatively higher disagreement that is almost certainly significant in nature, however the representative ability of the Normal and Uniform distribution appear quite similar from this test despite the fact that the Gaussian can and will sometimes sample values outside of the outcome space.

We now plot for our two most representative models the aforementioned histogram, without any worries of single extremes obfuscating the entire visual:

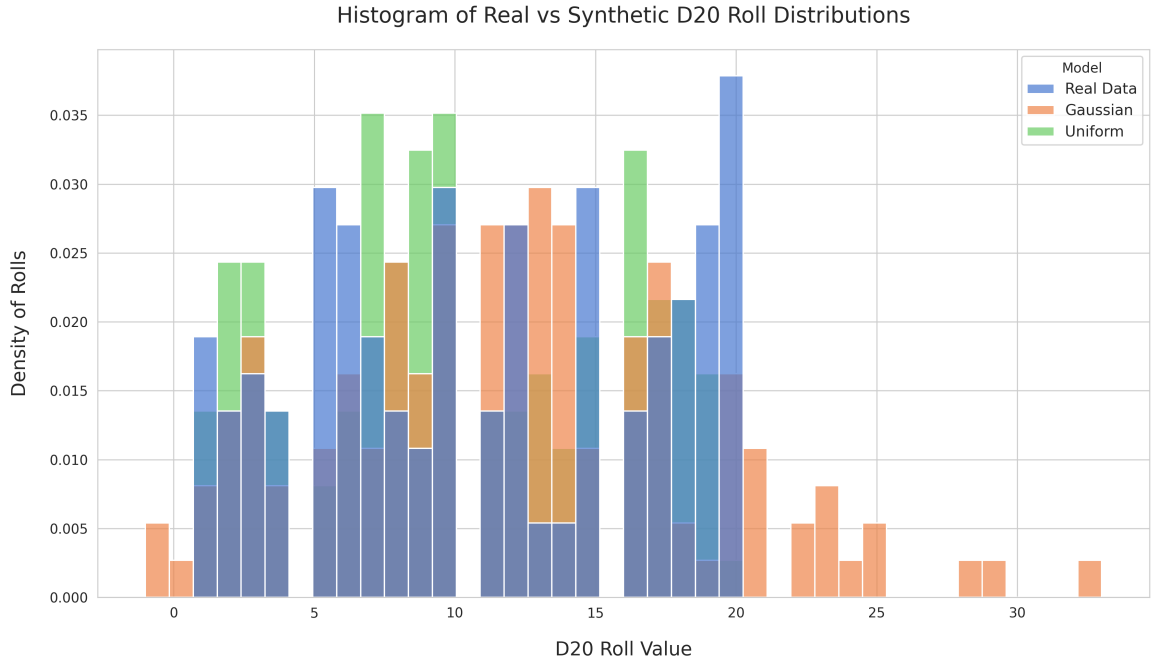

Figure 9: Histogram of Real vs Synthetic Distributions on D20 Die Rolls dataset.

It is immediately obvious from this plot that the Normal distribution is not as well-representative of the real sample toward extreme values, where the underlying assumptions of the uniform model regarding outcome space become relevant. The Uniform distribution mirrors the sporadic distribution of mass as the real sample set within the realms of possibilities for the die.

We can make a confident claim that the data sample produced by the Uniform model best represents the original data from its distribution. This confirms our hypothesis and the dataset behaved as we had expected; unweighted dice ought to be uniform in outcome.

3 Power Law Distribution Dataset

3.1 Part A: Developing Hypotheses

Identify and collect a real-world dataset that you hypothesize follows a Power Law distribution. Please be clear about the reasoning behind your hypothesis and be specific about the source of the dataset.
For the Power Law distribution, we decided to select a well-known example of power lar in the real-world: a dataset of US city populations, sourced from this Kaggle dataset. The Power Law distribution captures a very unique proclivity of certain phenomenon to have extreme values you can count on being present; a notion predictable frequency of extreme values in the distribution. It still decays and extreme values do become more and more scarce further from the mean, however the Power Law distribution maintains some amount of mass it relies on to capture a predicatable extreme behavior in the data. In this context, we hypothesize that the population of cities will follow this distribution because although it is true that very generally larger cities become less commonplace as that population grows, however there always will be a small yet reliable amount of cities that are extremely large in size.

This behavior of city population is reflected in the PDF for Power Law:

$$f(x) = \frac{\alpha - 1}{x_{\min}} \left(\frac{x}{x_{\min}} \right)^{-\alpha}$$

the chance of observing a city with some input size is proportional to x^-a, which dictates the natural inequality in the distribution between the average case behavior represented in the data and the extreme cases. This would allow the model to represent the proclivity of US cities to become mega-cities.

Regarding the data itself, it is a 2022 census of 19,268 US cities. For the purpose of fitting the Power Law model, it was necessary to remove 49 cities from the dataset that were made up of less than 10 people. This choice was made primarily because Power Law is designed to describe scaling behaviors about some minimum size, and in calculating logs of populations this small, we faced programmatic errors. Not to mention the census of cities this small could reflect some level of data collection error surely if the cities being characterized is the size of one large family. For this reason, and to preserve the underlying mechanism of the Power Law distribution, these "cities" were excluded.

3.2 Part B: Fitting Distributions

For this exercise, we will call each of the four different theoretical distributions (normal, uniform, power law, exponential) a “model". Fit the dataset (i.e., estimate the model parameters) against each model (not just the one you hypothesized) using maximum likelihood estimation (or using any technique you think is appropriate; make sure to comment on the validity of your approach). This should result in a total of **4 parameter sets**. Report the estimated parameters in the following tabular format:

		Model			
Dataset	***# Observations***	**Normal**	**Uniform**	**Power law**	**Exponential**
Dataset 3	n_3	μ_3, σ_3	a_3, b_3	$\alpha_3, x_{\min_3}$	λ_3

Be sure to show the code you used to arrive at your final estimates clearly.

—

Below are the tabulated parameter estimates for this dataset (for the full tabulation described in the original assignment TeX, see Appendix). The code in Fig. 1 was also used to arrive at our final parameter estimates for this dataset, here is the same implementation below for convenience:

```
def dists_fit(input_csv: str) -> tuple:
    """
    Fits the obs dataset to each model using MLE.

    :param input_csv: Path to input data to fit paramater(s) to
    :type input_csv: str
    """
    obs = pd.read_csv(input_csv).iloc[:, 0].to_numpy()

    mu = np.mean(obs)
    std = np.sqrt(np.sum((obs - mu) ** 2) / len(obs))

    a, b = obs.min(), obs.max()

    alpha = 1 + len(obs) / np.sum(np.log(obs / a))

    lamb = 1 / np.mean(obs)

    return (mu, std, a, b, alpha, lamb)
```

		Model			
Dataset	***# Observations***	**Normal**	**Uniform**	**Power law**	**Exponential**
US City Populations	19268	10800, 83590	11, 8335897	1.206, 11	0.0000926

Figure 10: Parameter estimates for each model on US City Populations dataset.

3.3 Part C: Comparing Real and Synthetic Data

For each fitted distribution (there will be 4 of them for this dataset, each corresponding to a different model), generate a synthetic sample of data points equal to the sample size of the real dataset using the respective model parameters you inferred from the real dataset.

Compare the real vs. synthetic data distributions using methods you think are the most appropriate, including visualizations. So, for this dataset, we compare the original dataset to four synthetic datasets, all with equal number of observations, but each synthetic dataset is generated using a different model.

For this dataset, identify the synthetic dataset (which corresponds to a model) that is most similar to the original data in terms of its distribution.

Now revisit your initial hypothesis. For this dataset: Did the dataset behave as expected, or was another model (assumed distribution) a better fit to the dataset? Reflect on why the observed results may differ from your expectations.

—

We plotted a log-log Q-Q plot for the sample because of the magnitude of the sample ranges, which should in turn tell us which samples better resemble a Power Law distribution based on which are best aligned with the diagonal:

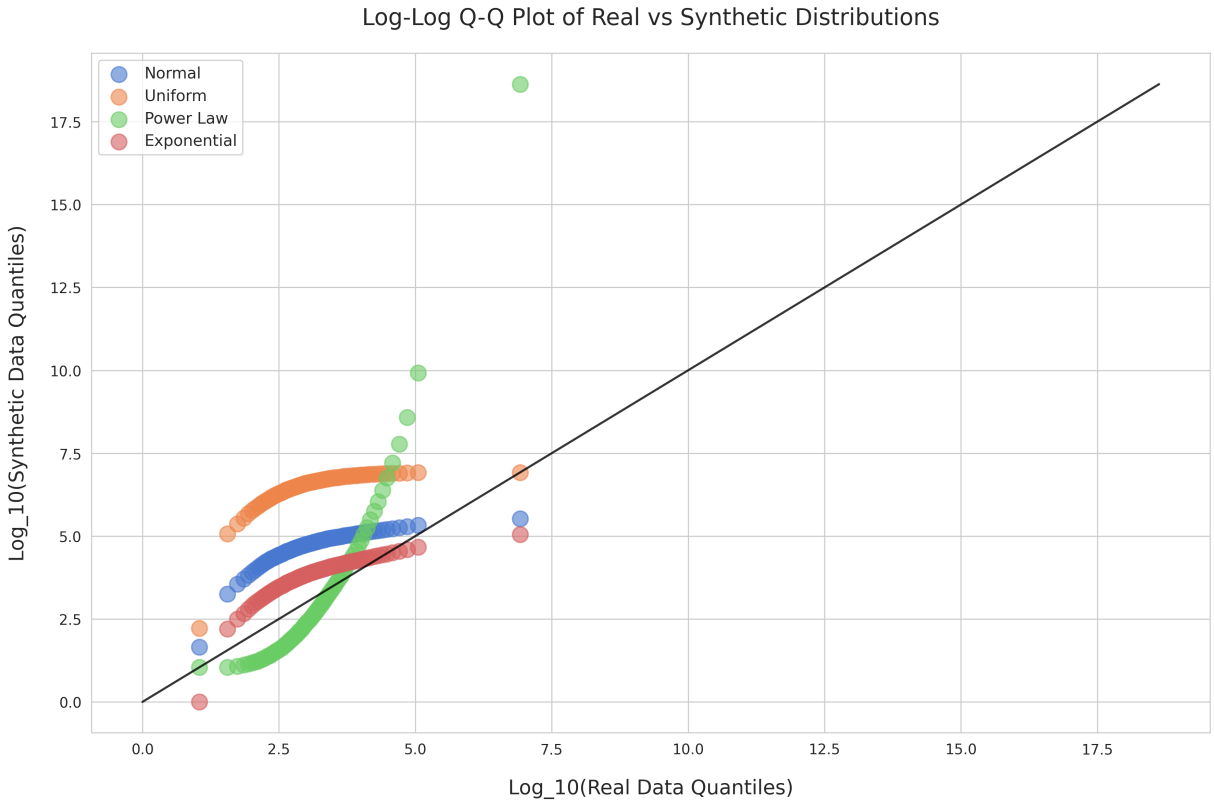

Figure 11: Q-Q Log-Log Plot comparing Real vs Synthetic quantiles on US City Populations dataset.

This visualization shows the models all struggled to produce a data sample resembling a Power Law distribution. We were surprised by this result until we doubled back and saw that the estimates of the Power Law parameters produce by the MLE step were quite poor, with an α of 1.206. Our data sample for our most promising distribution does not have a definable mean nor variance, and data values so extreme that plotting on visuals like a histogram would be infeasible for the entire range.

Distribution	K-S Statistic	p-value	Significant
Normal	0.46	0.000	Yes
Uniform	0.97	0.000	Yes
Power Law	0.31	0.000	Yes
Exponential	0.45	0.000	Yes

Figure 12: Kolmogorov-Smirnov test results between the US City Populations dataset and synthetic samples.

To corroborate this result for the data, we produce a K-S statistics table, showing incredibly low levels of uncertainty in the structural disagreements between the real data and each of these models. We can at least say though that the agreement of the Power Law model is still the best, but is a far cry from anything that we can speak definitely on.

That being said, we can only say with moderate levels of uncertainty that the Power Law model did perform best to mirror the real data from is distribution. The dataset certainly did not behave as expected, and perhaps we would benefit from pursuing other techniques with which to fit the data. Potential reasons why the reality was so different from our expectations could be attributed to the fact that there were many data points that perhaps were not filtered out that were too small to be reasonably subjected to Power Law scaling, making the challenge of estimation much more difficult. Not to mention the exponential distribution, the only distribution somewhat resembling the shape of the Power Law distribution struggled greatly as well, so there is an argument to be made that perhaps that the data may need to be dissected and cleaned much more heavily before attempting to fit a model to it using MLE.

4 Exponential Distribution Dataset

4.1 Part A: Developing Hypotheses

Identify and collect a real-world dataset that you hypothesize follows an Exponential distribution. Please be clear about the reasoning behind your hypothesis and be specific about the source of the dataset.
We sourced a dataset from Allen Downey, a principal data scientist at PyMC and a former professor at Olin college, from his Data Exploration repository on GitHub. This dataset possesses maternity hospital data, characterizing the intervals of births in minutes. It is a small set of timestamps through a single day, which we transformed into uni-variate observations of the inter-arrival times of consecutive births.

We hypothesize that birth intervals follow an Exponential distribution as this measure possess a notion of a central tendency, an average interval between births, and can at times extends to extreme values for a myriad of underlying reasons. What we see are a bunch of independent events naturally occurring a some rate throughout this day, and what is important to understand about these observations is that despite a central tendency, they are completely unrelated to the intervals between previous consecutive births. For this reason, our density function for the Exponential distribution:

$$f(x) = \lambda e^{-\lambda x}$$

makes a natural representation of independent events spaced out somewhat evenly. Additionally, this distribution has a decay property at further deviates from its mean in the positive direction, meaning that it can readily characterize perhaps the increasing rarity or longer intervals between births occurring, and it turn characterizes the increasing proclivity of births to occur at intervals surrounding the mean.

4.2 Part B: Fitting Distributions

For this exercise, we will call each of the four different theoretical distributions (normal, uniform, power law, exponential) a “model". Fit the dataset (i.e., estimate the model parameters) against each model (not just the one you hypothesized) using maximum likelihood estimation (or using any technique you think is appropriate; make sure to comment on the validity of your approach). This should result in a total of **4 parameter sets**. Report the estimated parameters in the following tabular format:

		Model			
Dataset	***# Observations***	**Normal**	**Uniform**	**Power law**	**Exponential**
Dataset 4	n_4	μ_4, σ_4	a_4, b_4	$\alpha_4, x_{\min_4}$	λ_4

Be sure to show the code you used to arrive at your final estimates clearly.

—

Below are the tabulated parameter estimates for this dataset (for the full tabulation described in the original assignment TeX, see Appendix). The code in Fig. 1 was also used to arrive at our final parameter estimates for this dataset, here is the same implementation below for convenience:

```
def dists_fit(input_csv: str) -> tuple:
    """
    Fits the obs dataset to each model using MLE.

    :param input_csv: Path to input data to fit paramater(s) to
    :type input_csv: str
    """
    obs = pd.read_csv(input_csv).iloc[:, 0].to_numpy()

    mu = np.mean(obs)
    std = np.sqrt(np.sum((obs - mu) ** 2) / len(obs))

    a, b = obs.min(), obs.max()

    alpha = 1 + len(obs) / np.sum(np.log(obs / a))

    lamb = 1 / np.mean(obs)

    return (mu, std, a, b, alpha, lamb)
```

		Model			
Dataset	***# Observations***	**Normal**	**Uniform**	**Power law**	**Exponential**
Brisbane Birth Intervals	43	33.3, 29.2	1, 157	1.32, 1	0.0301

Figure 13: Parameter estimates for each model on Brisbane Birth Intervals dataset.

4.3 Part C: Comparing Real and Synthetic Data

For each fitted distribution (there will be 4 of them for this dataset, each corresponding to a different model), generate a synthetic sample of data points equal to the sample size of the real dataset using the respective model parameters you inferred from the real dataset.

Compare the real vs. synthetic data distributions using methods you think are the most appropriate, including visualizations. So, for this dataset, we compare the original dataset to four synthetic datasets, all with equal number of observations, but each synthetic dataset is generated using a different model.

For this dataset, identify the synthetic dataset (which corresponds to a model) that is most similar to the original data in terms of its distribution.

Now revisit your initial hypothesis. For this dataset: Did the dataset behave as expected, or was another model (assumed distribution) a better fit to the dataset? Reflect on why the observed results may differ from your expectations.

We first ought to compare the samples by how well-they reflect the decaying nature of an Exponential at greater interval ranges, for which we can use a histogram:

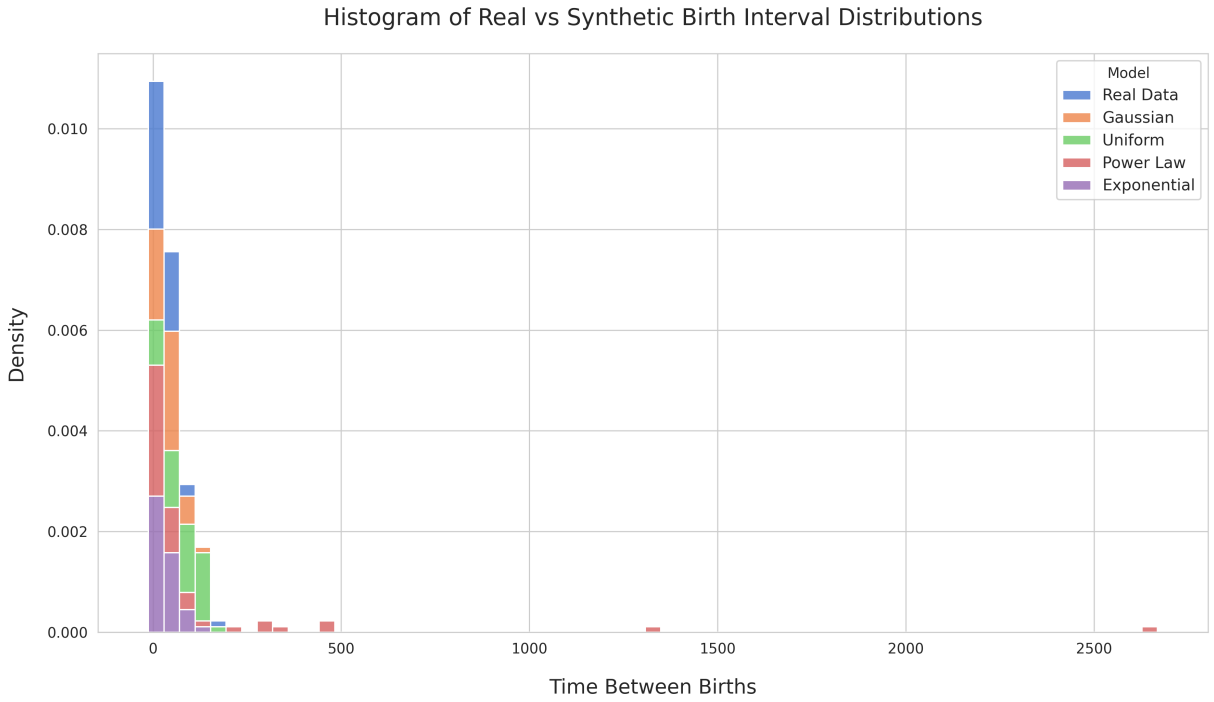

Figure 14: Histogram of Real vs Synthetic Distributions on Birth Intervals dataset.

Viewing how the mass is distributed was challenging for overlapping distributions, so we chose to stack them, and it is clear visually that for where the mass is concentrated in the real dataset, we see more mass distributed in the same spot for the Power Law Distribution and the Exponential distribution. It is also apparent that the Power Law distribution is still trying to fit to extreme values in the data, suggesting that while a strong choice, it may not be the best option for representing a decay pattern at extremes. The other two datasets are quite unsuitable in this case and are showcase their own idiosyncrasies while somewhat adhere to the structure of the Birth Intervals dataset.

We produced a K-S test below to gain a more concrete understanding of how the Power Law distribution and the Exponential distribution compare in this context:

Distribution	K-S Statistic	p-value	Significant
Gaussian	0.28	0.070	No
Uniform	0.51	1.73×10^{-5}	Yes
Power Law	0.30	0.039	Yes
Exponential	0.12	0.938	No

Figure 15: Kolmogorov-Smirnov test results between the Birth Intervals dataset and synthetic samples.

Surprisingly, the K-S test told us that there is actually much higher disagreement than we could discern from the histogram alone between the real data and the Power Law model with a significant p-value of 0.039, and even more surprisingly, the Gaussian, while not as in agreement with the data as the Exponential, agrees more with the real data than the Power Law distribution, which perhaps makes sense when you reduce them both to distributions that prioritize decay away from the center.

From these visualizations, we can make a confident claim that the dataset produce by the Exponential distribution is most similar to the real dataset. This was the expected behavior for this data, and in fact the fit and visualization showed with much greater clarity how important the general notions of a distribution are to how it is characterized, and specifically how it is differentiated from others. Two distributions like Power Law and Exponential can look very similar, but there is nuance that distances them greater than may be readily apparent.