CSDS 313 Assignment 1: Data & Distributions

Camden Larson

September 2025

1 Introduction

Briefly introduce the goals of the assignment:

- Understanding how visualizations influence data interpretation.
- Practicing distribution fitting with real-world data.
- Comparing real and synthetic datasets.

2 Task 1: Comparing Visualizations

2.1 Dataset 1: Best-Selling Albums

2.1.1 Data Cleaning & Preprocessing

- Dropped unnecessary columns including Artist, Ranking, Tracks, and Album Length.
- Dropped entries prior to 2016.
- Stripped whitespace and put values in title case.

2.1.2 Visualizations

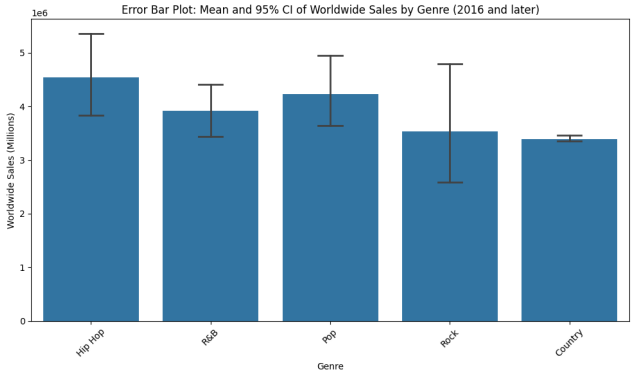

Figure 1: Error Bar Plot for Album Sales by Genre

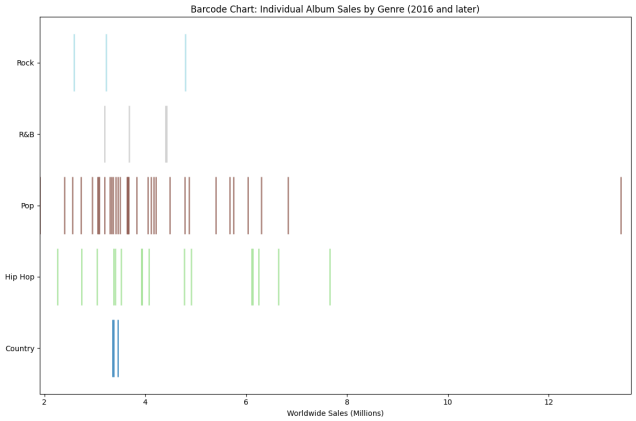

Figure 2: Barcode Plot for Album Sales by Genre

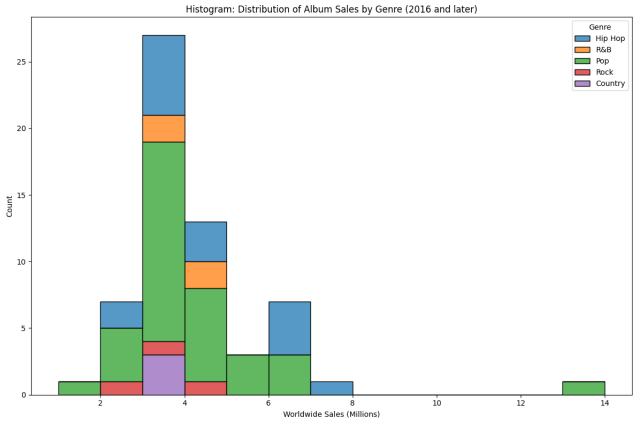

Figure 3: Histogram of Album Sales by Genre

2.1.3 Evaluation

- The error bar plot was a really good comparison of the means between each genre making it easy to see which genre produced the highest average sales. However, it doesn't tell the story behind the volume of sales. For example, if one genre had only 5 songs in the top 10 and averaged 4.5 million per song, and another had 50 songs in the top 10 in the last 10 years, but averaged 4.3 million per song, the viewer would not be able to differentiate between the two very easily on this graph.

- The barcode chart reveals clustering, gaps, and outliers which shows the distribution well. It isn't that useful for comparing central tendency as the mean and confidence intervals are hard to judge from the plot.

- The histogram shows the distribution of sales really well and doesn't hide the outliers. It also has genre info still with the color scheme. One con is that the bin size can affect the interpretation.

- I think the best visualization for the research question is the barcode chart. It easily displays the distribution of album sales across genres because you get to see every data point and it is a small enough dataset that it is not super cluttered. The second part of the research question about high-sales outliers is also easily seen in the graph. While the histogram can also show both of these, the varying bin size and inability to see individual data points makes it less useful. The error bar plot just shows the mean and can give us some insight on outliers if we look at the error bars, however, it doesn't give us a good idea of the volume of albums for each genre which could lead to misleading conclusions.

2.2 Dataset 2: Anime

2.2.1 Data Cleaning & Preprocessing

- Stripped whitespace for all string columns.
- Kept only TV series and movies released after 2015.
- Converted rating column to numeric.
- Kept rating and type for analysis.

2.2.2 Visualizations

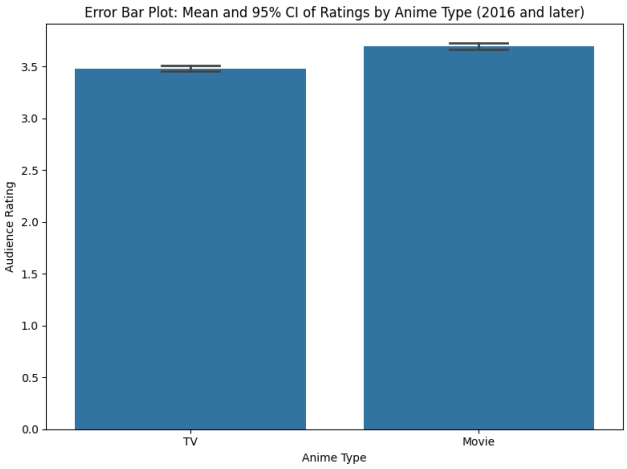

Figure 4: Error Bar Plot for Ratings by Anime Type

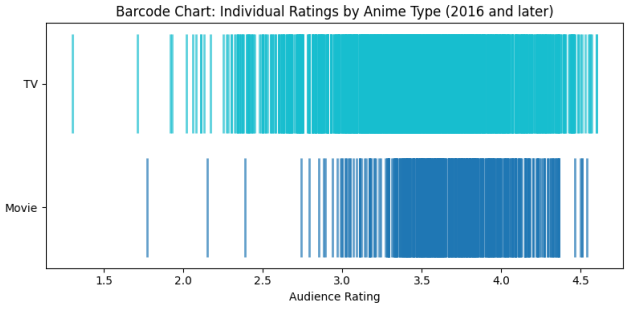

Figure 5: Barcode Plot for Ratings by Anime Type

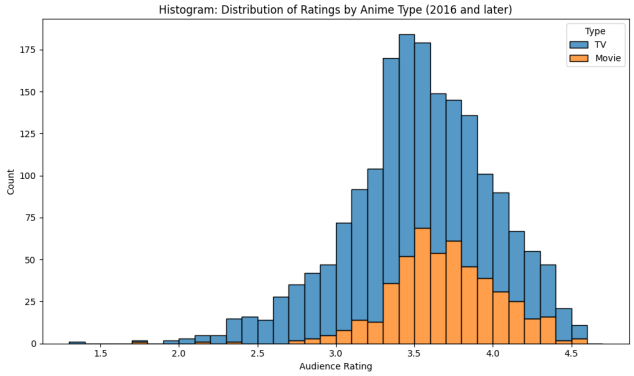

Figure 6: Histogram of Ratings by Anime Type

2.2.3 Evaluation

- The error bar plot summarizes mean ratings and displays variability of TV series for movies making direct comparison easy. However, it does not show the full spread or outliers, so it only tells us about the average ratings, not the distribution.
- The barcode plot shows the spread and outliers really well and gives us a good idea of where the clusters/groupings are. It is tough to tell which has a higher average though because of the volume of data points.
- The histogram gives us a great idea of the distribution for both TV series and movies. Can be misleading with bins that have low counts if the bins are chosen poorly.
- The clear best chart for the research question is the error bar plot. The main thing we are looking for is which format generally receives higher ratings, and the error bar plot is the simplest to see the mean and confidence interval, which give us a good idea of which media type gets better audience ratings. The histogram is not bad, but is tough to see the mean and can be potentially misleading if values are clustered in certain bins. The bar code chart has too many data points it is difficult to compare the means.

2.3 Dataset 3: Algorithm Performance

2.3.1 Data Cleaning & Preprocessing

- Strip whitespace and make title case for algorithm names.
- Drop rows with missing or invalid (¿1) accuracy.
- Drop the index and other unnecessary columns.

2.3.2 Visualizations

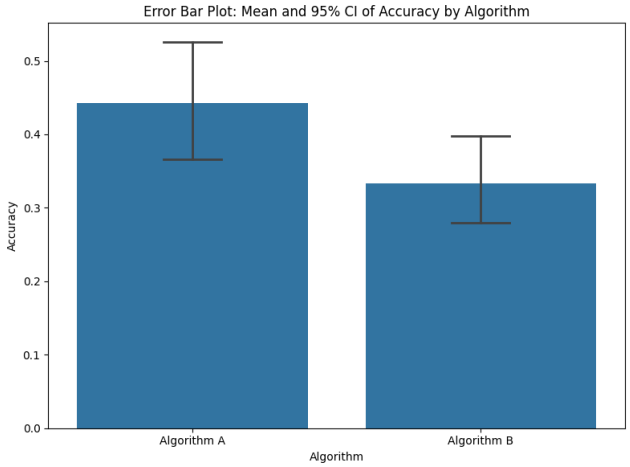

Figure 7: Error Bar Plot for Accuracy by Algorithm

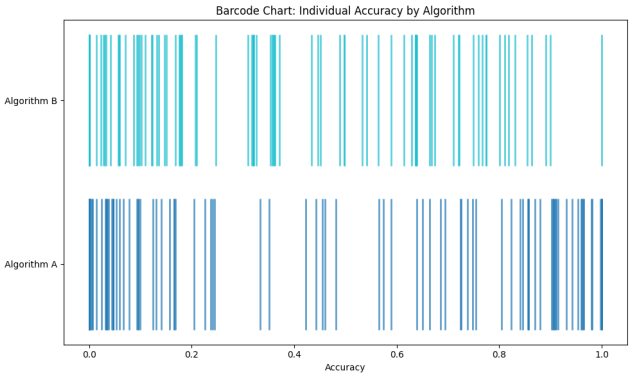

Figure 8: Barcode Plot for Accuracy by Algorithm

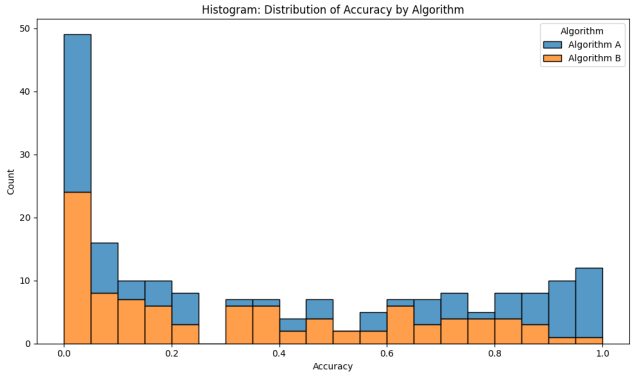

Figure 9: Histogram of Accuracy by Algorithm

2.3.3 Evaluation

- The error bar plot compares the mean accuracy and variability in a useful way. It doesn't show the distribution or outliers so could be potentially misleading.
- The barcode chart shows every result so it is a good representation of the distribution and outliers. It gets a little cluttered with this many trials and is less effective for summarizing performance.
- The histogram does a good job of showing the distribution of accuracy as well as the consistency and spread. It is difficult to directly compare means.
- The best visualization for the research question is the error bar plot because it gives us the average across all epochs. The width of the error bar helps us assess the variation of each algorithm.

3 Task 2: Fitting and Comparing Distributions

3.1 Part A: Developing Hypotheses

- Normal - Men's college 10k times. Found on Kaggle.
- Uniform - Random numbers between 1-100. Generated using a python script. This should be uniform because every number has an equal likelihood of being selected. If we run the experiment a sufficient amount of times, the distribution should be uniform.
- Power law - City populations of Brazil. Found on Kaggle. Similar to the example discussed in class about the populations of us cities starting with 1. Each digit should have a decreased likelihood from 1-9.

- Exponential - Call lengths at a call center. Found on Kaggle. Call lengths should cluster around some length and exponentially decrease in frequency as the length of calls increases.

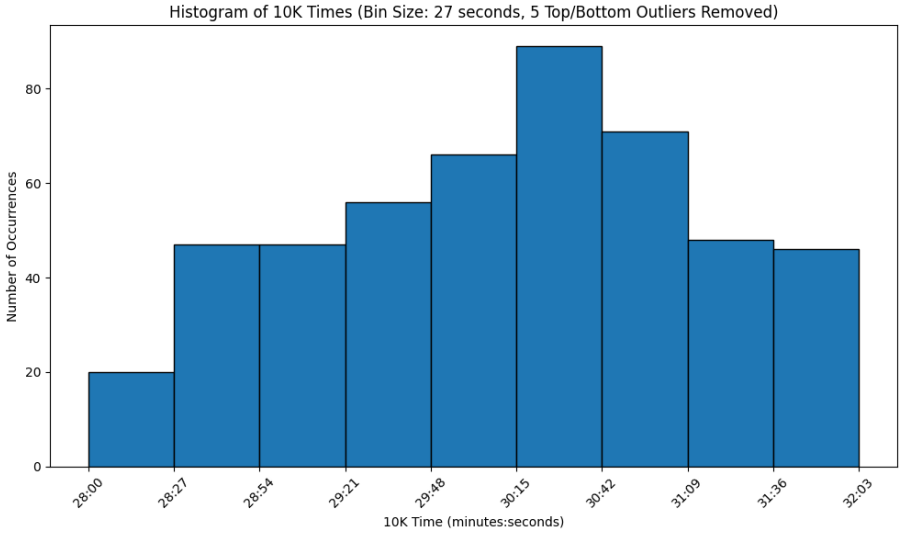

Figure 10: Normal Distribution Hypothesis

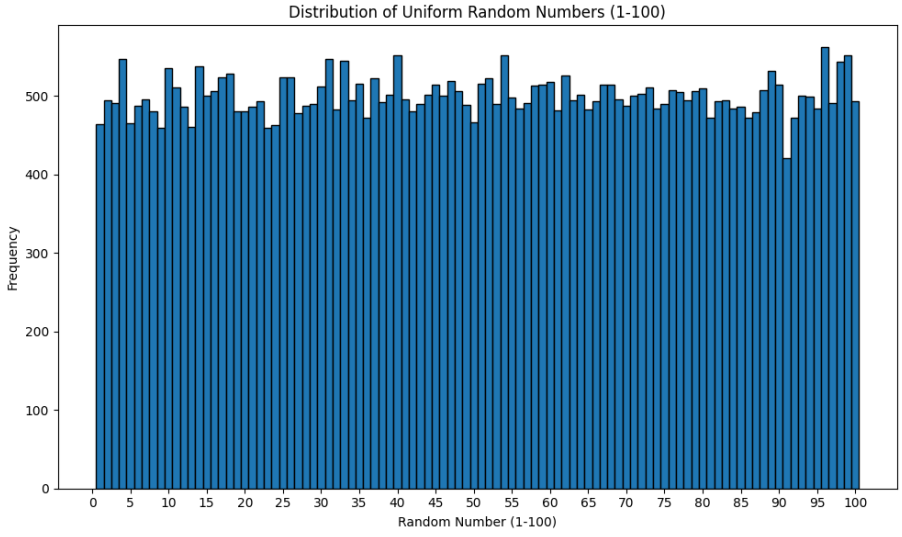

Figure 11: Uniform Random Distribution Hypothesis

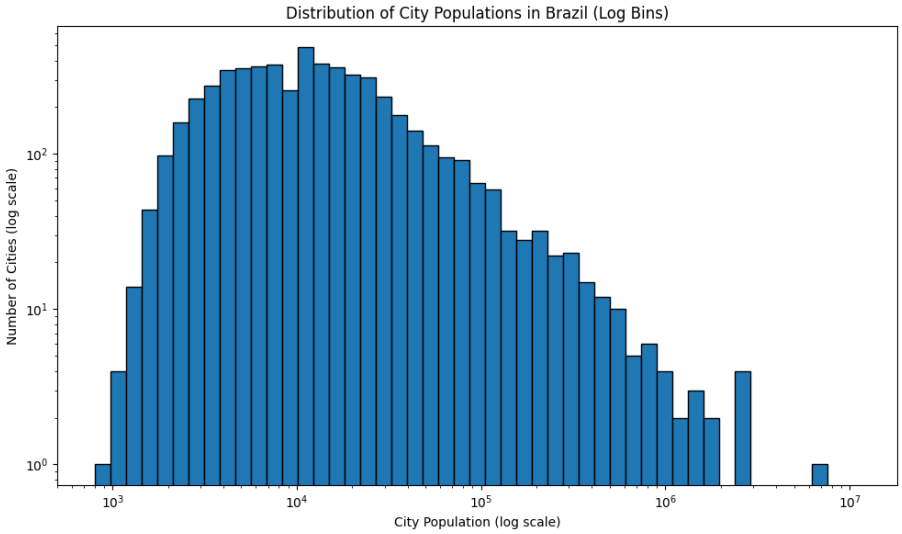

Figure 12: Power Law Distribution Hypothesis

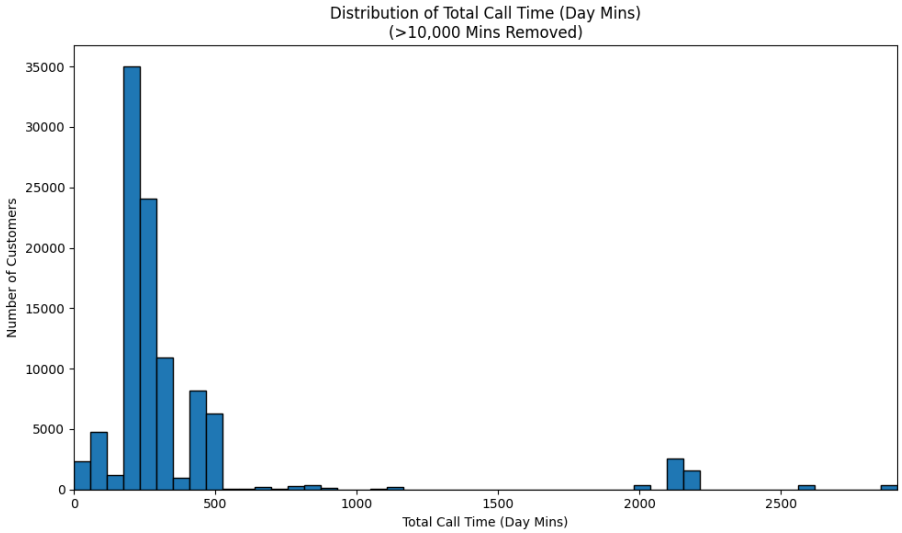

Figure 13: Exponential Distribution Hypothesis

3.2 Part B: Fitting Distributions

3.2.1 Parameter Estimates

Dataset	Normal (μ, σ)	Uniform (a, b)	Power law $(\alpha, x_{\min})$	Exponential (λ)
Dataset 1 (#=500)	1810.71, 61.47	1677.47, 1912.02	42.17, 1815.43	0.00055
Dataset 2 (#=50,000)	50.58, 28.82	1.00, 100.00	6.51, 71.00	0.01977
Dataset 3 (#=100,142)	378.38, 449.51	0.00, 2911.90	3.12, 198.00	0.00264
Dataset 4 (#=5,565)	34277.77, 203112.62	805.00, 11253503.00	2.14, 22151.00	0.00003

Table 1: Estimated parameters for each dataset and distribution.

3.2.2 Most Similar Synthetic (KS Test)

Dataset	Best Model by KS	KS Statistic
Dataset 1	Normal	0.0700
Dataset 2	Uniform	0.0148
Dataset 3	Power law	0.1195
Dataset 4	Exponential	0.2828

Table 2: Best-matching synthetic distribution per dataset based on the Kolmogorov–Smirnov test.

3.2.3 Code for Fitting Distributions

Listing 1: Python script used to fit and compare distributions.

```
import pandas as pd
import numpy as np
from scipy import stats
import powerlaw
import matplotlib.pyplot as plt
import os

# --- Dataset 1: 10k_times.csv (Time) ---
df1 = pd.read_csv('../data/10k_times.csv')
data1 = pd.to_numeric(df1['Time'], errors='coerce').dropna().values
n1 = len(data1)

# --- Dataset 2: uniform_random_results.csv (RandomNumber) ---
df2 = pd.read_csv('../data/uniform_random_results.csv')
data2 = pd.to_numeric(df2['RandomNumber'], errors='coerce').dropna
    ().values
n2 = len(data2)

# --- Dataset 3: call_details.csv (Day Mins, filtered to <= 10000)
    ---
df3 = pd.read_csv('../data/call_details.csv')
data3 = pd.to_numeric(df3['Day Mins'], errors='coerce').dropna()
data3 = data3[data3 <= 10000].values
n3 = len(data3)
```

```
23
24  # --- Dataset 4: BRAZIL_CITIES.csv (IBGE_RES_POP) ---
25  df4 = pd.read_csv('../data/BRAZIL_CITIES.csv', sep=';')
26  data4 = pd.to_numeric(df4['IBGE_RES_POP'], errors='coerce').dropna
        ().values
27  n4 = len(data4)
28
29  def fit_all_models(data):
30      # Normal
31      mu, sigma = np.mean(data), np.std(data, ddof=1)
32      # Uniform
33      a, b = np.min(data), np.max(data)
34      # Power law (using powerlaw package)
35      fit = powerlaw.Fit(data, verbose=False)
36      alpha = fit.power_law.alpha
37      xmin = fit.power_law.xmin
38      # Exponential (MLE: lambda = 1/mean)
39      lambd = 1 / np.mean(data)
40      return (mu, sigma), (a, b), (alpha, xmin), lambd
41
42  results = []
43  for i, (name, data, n) in enumerate([
44      ("Dataset 1", data1, n1),
45      ("Dataset 2", data2, n2),
46      ("Dataset 3", data3, n3),
47      ("Dataset 4", data4, n4),
48  ]):
49      (mu, sigma), (a, b), (alpha, xmin), lambd = fit_all_models(data
            )
50      results.append({
51          'Dataset': name,
52          '# Observations': n,
53          'Normal': f"{mu:.2f}, {sigma:.2f}",
54          'Uniform': f"{a:.2f}, {b:.2f}",
55          'Power law': f"{alpha:.2f}, {xmin:.2f}",
56          'Exponential': f"{lambd:.5f}"
57      })
58
59  # Print results
60  print(f"{'Model':<12}{'# Obs':<10}{'Normal':<20}{'Uniform':<20}{'
        Power law':<20}{'Exponential':<15}")
61  for r in results:
62      print(f"{r['Dataset']:<12}{r['# Observations']:<10}{r['Normal
            ']:<20}{r['Uniform']:<20}{r['Power law']:<20}{r['
            Exponential']:<15}")
63
64  # Generate synthetic data
65  synthetic = {}
66  for idx, (name, data, n) in enumerate([
67      ("Dataset 1", data1, n1),
68      ("Dataset 2", data2, n2),
69      ("Dataset 3", data3, n3),
70      ("Dataset 4", data4, n4),
71  ]):
72      (mu, sigma), (a, b), (alpha, xmin), lambd = fit_all_models(data
            )
73      np.random.seed(42)
```

```
    synth = {
        'Normal': np.random.normal(mu, sigma, n),
        'Uniform': np.random.uniform(a, b, n),
        'Power law': (np.random.pareto(alpha, n) + 1) * xmin,
        'Exponential': np.random.exponential(1/lambd, n)
    }
    synthetic[name] = synth

def plot_real_vs_synth(real, synth_dict, dataset_name, outdir):
    plt.figure(figsize=(12,8))
    for model, synth in synth_dict.items():
        plt.hist(synth, bins=50, alpha=0.5, label=f'{model} (
            synthetic)', density=True)
    plt.hist(real, bins=50, alpha=0.7, label='Real Data', color='
        black', density=True, histtype='step')
    plt.title(f'Real vs. Synthetic Distributions for {dataset_name}
        ')
    plt.xlabel('Value')
    plt.ylabel('Density')
    plt.legend()
    plt.tight_layout()
    os.makedirs(outdir, exist_ok=True)
    plt.savefig(os.path.join(outdir, f'{dataset_name.lower().
        replace(" ", "_")}_real_vs_synth.png'))
    plt.close()

def compare_ks(real, synth_dict):
    scores = {}
    for model, synth in synth_dict.items():
        stat, _ = stats.ks_2samp(real, synth)
        scores[model] = stat
    return scores

# Run for each dataset
outdir = '../graphs/model_fits'
most_similar = {}
for name, data, n in [
    ("Dataset 1", data1, n1),
    ("Dataset 2", data2, n2),
    ("Dataset 3", data3, n3),
    ("Dataset 4", data4, n4),
]:
    plot_real_vs_synth(data, synthetic[name], name, outdir)
    ks_scores = compare_ks(data, synthetic[name])
    best_model = min(ks_scores, key=ks_scores.get)
    most_similar[name] = best_model
    print(f"{name}: Most similar synthetic model by KS test: {
        best_model} (KS={ks_scores[best_model]:.4f})")
```

3.3 Part C: Comparing Real and Synthetic Data

3.3.1 Visual Comparisons

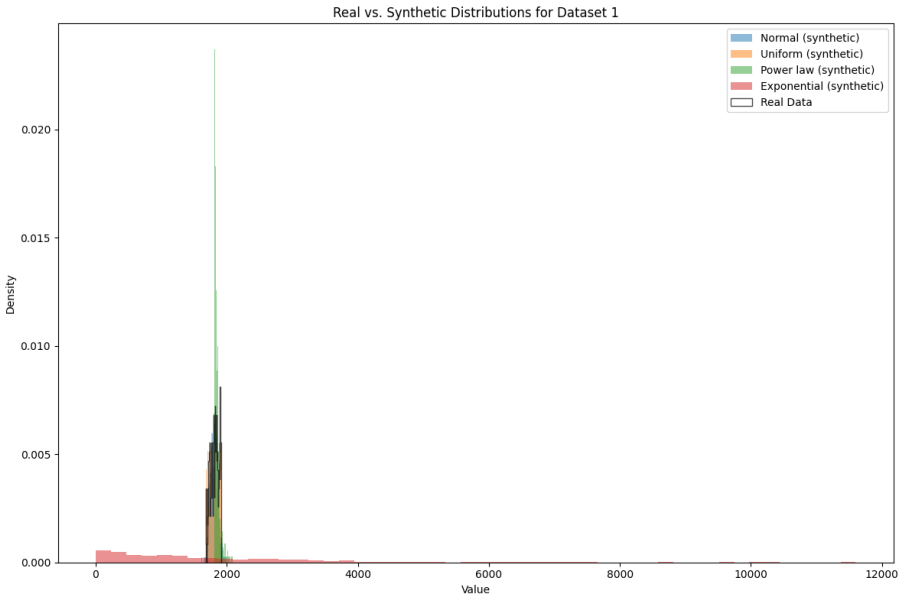

Figure 14: Real vs Synthetic Distributions for Dataset 1

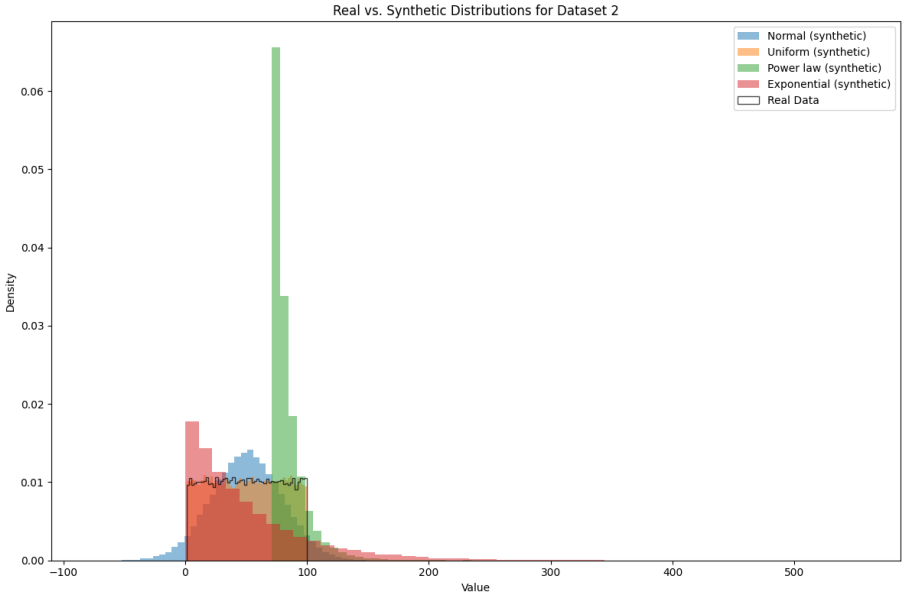

Figure 15: Real vs Synthetic Distributions for Dataset 2

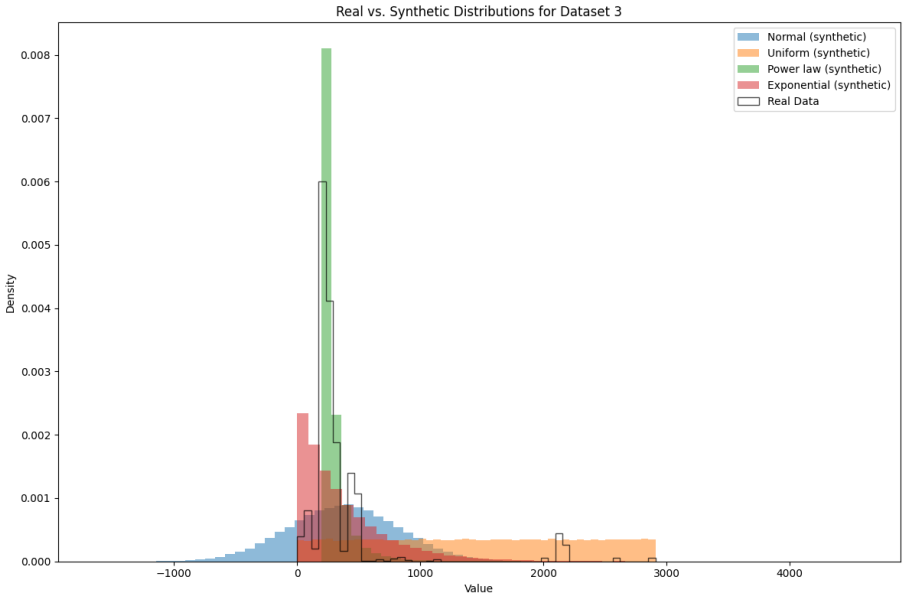

Figure 16: Real vs Synthetic Distributions for Dataset 3

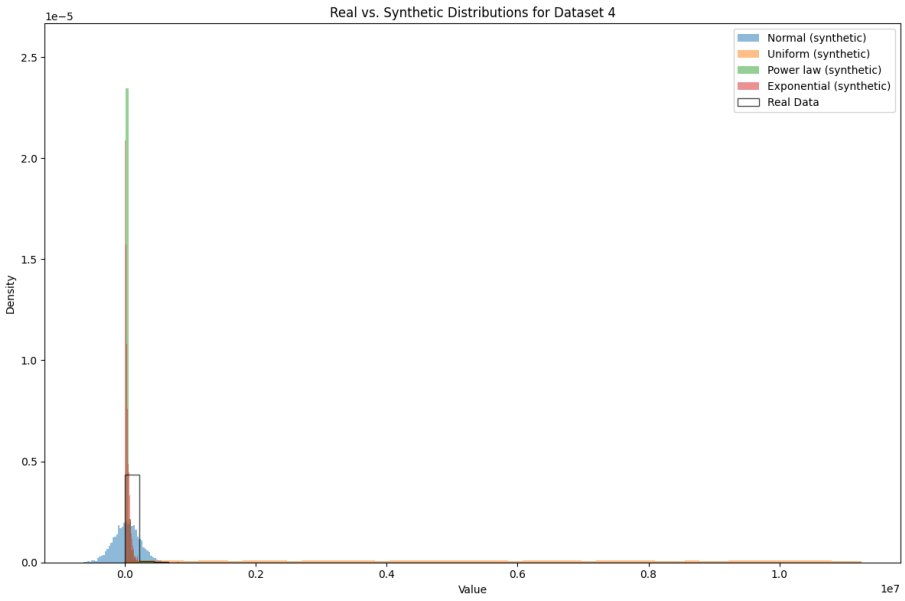

Figure 17: Real vs Synthetic Distributions for Dataset 4

(Repeat for each dataset)

3.3.2 Discussion

- **Normal:** Yes, the dataset was most similar to the normal distribution model. Looking at the histogram it appears almost normal but skewed a little left due to the dataset containing the top 300 rather than all 10k runners (which would fill in more on the right side and make it look more like a normal distribution). However, when compared with all of the models, it was most similar to the normal distribution.
- **Uniform:** Yes, the dataset was most similar to the uniform distribution model. The histogram appeared very similar already, and was verified by overlaying the four synthetic models. I used 50000 random 1-100 numbers and if I used more it would get even closer to a perfect uniform distribution.
- **Power Law:** No, the dataset was not most similar to power law distribution. After viewing the histogram it appeared somewhat similar to a power law distribution after the mean, but did not follow power law before the mean. The exponential distribution was more similar during synthetic model testing and by ks test. The observed results likely differ because I used the entire dataset rather than focusing on just part of the dataset that could have demonstrated power law. For example, If I bound the range from 10,000 and up, it would have been pretty close to power law distribution displayed by the linearity after $10^4 on the histogram$.

- **Exponential:** No, the dataset was not most similar to exponential distribution. I initially thought call length would cluster around the mean and exponentially decrease as the length got longer, but there were a significant amount of calls much longer than the mean that may have skewed the data. It also followed more of a power law distribution from the mean to the 75th percentile which is likely why it fit most closely with the power law distribution.