analyze_trimmed_data_council_02

February 1, 2025

1 Troop Booth Signups Analysis

This notebook loads data from `data/2025_booth_signups_trimmed.csv` (which contains columns such as **Troop**, **Troop Email**, **Slot Start Time**, **Slot End Time**, **When Selected Date**, **When Selected Time**, **User Selecting**) and performs statistical analysis focused on the number of booth signup events per troop. In addition, the notebook identifies and labels potential outliers in the distribution of booth signup counts.

[21]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Set up plotting style
sns.set(style="whitegrid")
plt.rcParams['figure.figsize'] = (10, 6)
```

1.1 1. Load the Data

The data file is located at `data/2025_booth_signups_trimmed.csv`. It is assumed that the file has a header with the following columns:

- Troop
- Troop Email
- Slot Start Time
- Slot End Time
- When Selected Date
- When Selected Time
- User Selecting

Let's load the data into a pandas DataFrame.

[22]:
```
# Define the path to the data file
data_file = 'data/2025_booth_signups_council_trimmed.csv'

# Load the CSV file (with header)
df = pd.read_csv(data_file)

# Display the first few rows
```

```
print("Data preview:")
df.head()
```

```
Data preview:
```

[22]:

	Troop	Troop Email	Slot Start Time	Slot End Time	\
0	13	yflores7@cox.net	2000/01/01 13:00:00	2000/01/01 15:00:00	
1	13	yflores7@cox.net	2000/01/01 9:00:00	2000/01/01 12:00:00	
2	13	yflores7@cox.net	2000/01/01 12:00:00	2000/01/01 15:00:00	
3	71	nikolec@gmail.com	2000/01/01 15:00:00	2000/01/01 17:00:00	
4	71	nikolec@gmail.com	2000/01/01 17:00:00	2000/01/01 19:00:00	

	When Selected Date	When Selected Time
0	2024/12/04 20:01:40	2024/12/04 20:01:40
1	2024/12/03 20:00:01	2024/12/03 20:00:01
2	2024/12/02 20:00:00	2024/12/02 20:00:00
3	2024/12/03 20:08:00	2024/12/03 20:08:00
4	2024/12/04 20:00:03	2024/12/04 20:00:03

1.2 2. Data Preparation

We will convert the time columns to datetime objects. (Note: The sample times use a format like `2000/01/01 14:00:00`, which we assume is consistent for the time fields.)

In our analysis we focus on counting the number of booth signup events per troop (each row is one event).

[23]:
```
# Convert time columns to datetime objects
df['Slot Start Time'] = pd.to_datetime(df['Slot Start Time'], format='%Y/%m/%d␣
 ↪%H:%M:%S', errors='coerce')
df['Slot End Time']   = pd.to_datetime(df['Slot End Time'], format='%Y/%m/%d %H:
 ↪%M:%S', errors='coerce')

# If desired, you can also convert the When Selected Date and When Selected Time
df['When Selected Date'] = pd.to_datetime(df['When Selected Date'], format='%Y/
 ↪%m/%d %H:%M:%S', errors='coerce')
df['When Selected Time'] = pd.to_datetime(df['When Selected Time'], format='%Y/
 ↪%m/%d %H:%M:%S', errors='coerce')

# Check the data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9297 entries, 0 to 9296
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Troop               9297 non-null   int64
 1   Troop Email         9271 non-null   object
```

```
 2   Slot Start Time     9297 non-null   datetime64[ns]
 3   Slot End Time       9297 non-null   datetime64[ns]
 4   When Selected Date  9240 non-null   datetime64[ns]
 5   When Selected Time  9240 non-null   datetime64[ns]
dtypes: datetime64[ns](4), int64(1), object(1)
memory usage: 435.9+ KB
```

1.3 3. Aggregating Booth Signups per Troop

Since each row represents one booth signup event, we can count the number of events per troop by grouping on the **Troop** column.

[24]:
```
# Group by Troop and count the number of signup events per troop
troop_counts = df.groupby('Troop').size().reset_index(name='Num_Booths')

# Sort by number of booths (signup events)
troop_counts.sort_values('Num_Booths', ascending=False, inplace=True)
print("Booth signup counts per troop:")
print(troop_counts)

# Compute descriptive statistics
desc_stats = troop_counts['Num_Booths'].describe()
print("\nDescriptive statistics for booth signups per troop:")
print(desc_stats)
```

```
Booth signup counts per troop:
      Troop  Num_Booths
262    3396         271
13      203         155
336    3829          85
36      558          84
119    2121          73
..      …           …
190    2999           1
214    3140           1
144    2313           1
589    7427           1
626  704674           1

[627 rows x 2 columns]

Descriptive statistics for booth signups per troop:
count    627.000000
mean      14.827751
std       17.839872
min        1.000000
25%        5.000000
50%       10.000000
```

```
75%       19.000000
max      271.000000
Name: Num_Booths, dtype: float64
```

1.4 4. Outlier Detection

We use the Interquartile Range (IQR) method to identify potential outliers in the number of booth signup events per troop.

An outlier is defined as a troop whose count is below Q1 - 1.5×IQR or above Q3 + 1.5×IQR.

[25]:
```
Q1 = troop_counts['Num_Booths'].quantile(0.25)
Q3 = troop_counts['Num_Booths'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print(f"Q1: {Q1}, Q3: {Q3}, IQR: {IQR}")
print(f"Lower bound: {lower_bound}, Upper bound: {upper_bound}")

# Identify outlier troops
outliers = troop_counts[(troop_counts['Num_Booths'] < lower_bound) |␣
 ↪(troop_counts['Num_Booths'] > upper_bound)]
print("\nOutlier troops (by number of booth signups):")
print(outliers)
```

```
Q1: 5.0, Q3: 19.0, IQR: 14.0
Lower bound: -16.0, Upper bound: 40.0

Outlier troops (by number of booth signups):
     Troop  Num_Booths
262   3396         271
13     203         155
336   3829          85
36     558          84
119   2121          73
49     872          68
489   4929          67
293   3587          63
3       80          62
572   7190          61
493   4997          61
570   7121          61
15     212          60
332   3822          59
30     436          59
370   3983          59
568   7073          58
```

```
88    1682          58
286   3561          57
6      123          56
448   4520          55
518   6310          53
498   5381          52
587   7425          52
260   3392          50
206   3108          47
539   6527          46
434   4458          45
537   6520          45
371   3985          45
616   9653          43
438   4491          43
28     417          42
170   2589          42
299   3605          41
203   3092          41
```

1.5 5. Visualization with Outlier Labels

Below is a boxplot of the booth signup counts per troop. In addition, we overlay a stripplot (jittered points) for each troop and label the outlier points with the corresponding troop number.

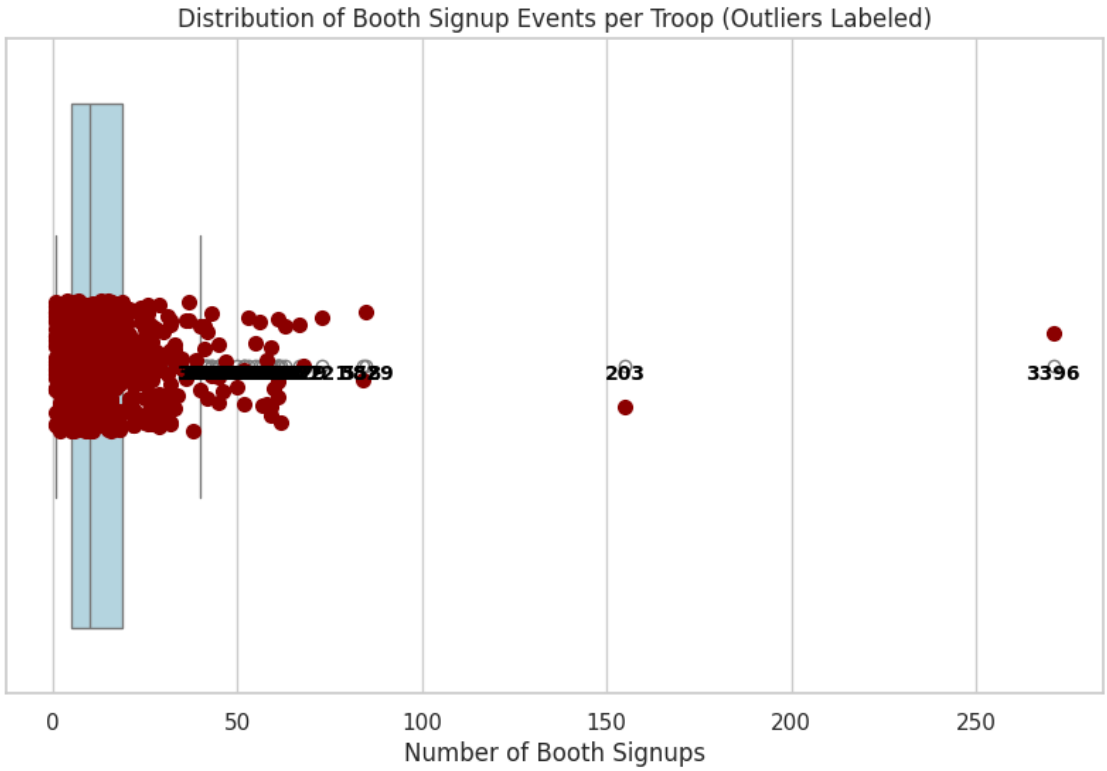
[26]:
```
plt.figure(figsize=(10,6))

# Create a boxplot of the booth signup counts
ax = sns.boxplot(x='Num_Booths', data=troop_counts, color='lightblue')

# Overlay a stripplot of individual troop counts
sns.stripplot(x='Num_Booths', data=troop_counts, color='darkred', size=8,␣
 ↪jitter=True, ax=ax)

# Annotate outliers with their Troop number
for index, row in troop_counts.iterrows():
    if row['Num_Booths'] < lower_bound or row['Num_Booths'] > upper_bound:
        # The y-value: use a slight vertical offset (here 0) because stripplot␣
 ↪adds jitter.
        ax.text(row['Num_Booths'], 0.02, str(row['Troop']),␣
 ↪horizontalalignment='center',
                color='black', weight='bold', fontsize=10)

plt.title('Distribution of Booth Signup Events per Troop (Outliers Labeled)')
plt.xlabel('Number of Booth Signups')
plt.show()
```



1.6 6. Save the Summary Data

Finally, we save the aggregated summary (the number of booth signup events per troop) to a CSV file for further use.

[27]:
```
output_file = 'troop_booth_summary.csv'
troop_counts.to_csv(output_file, index=False)
print(f"Summary data saved to {output_file}")
```

```
Summary data saved to troop_booth_summary.csv
```

1.7 Conclusion

In this notebook we have:

- Loaded the troop booth signup data from `data/2025_booth_signups_trimmed.csv`
- Prepared the data by converting time fields to datetime
- Aggregated the data to compute the number of signup events per troop
- Computed descriptive statistics and used the IQR method to identify potential outliers
- Visualized the distribution with a boxplot and overlayed a stripplot, labeling the outlier troops with their troop numbers
- Saved the aggregated summary to a CSV file

Feel free to extend or modify this analysis as needed.