notebook_council

February 1, 2025

1 Troop Booth Signups Analysis

This notebook loads data from (which contains columns such as **Troop**, **Slot Start Time**, **Slot End Time**) 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.

[6]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Define the path to the data file
data_file = "data/2025_booth_signups_council.csv"

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

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

# 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')

# Display the first few rows
print("Data preview:")
df.head()
```

```
Data preview:
```

[6]:

```
   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 09:00:00 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
```

[7]:

```python
# 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)
```

```
Booth signup counts per troop:
      Troop  Num_Booths
262    3396         271
13      203         155
336    3829          85
36      558          84
119    2121          73
..      …           …
144    2313           1
143    2265           1
240    3241           1
444    4508           1
626  704674           1

[627 rows x 2 columns]
```

[8]:

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

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}")
```

```python
# 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)
```

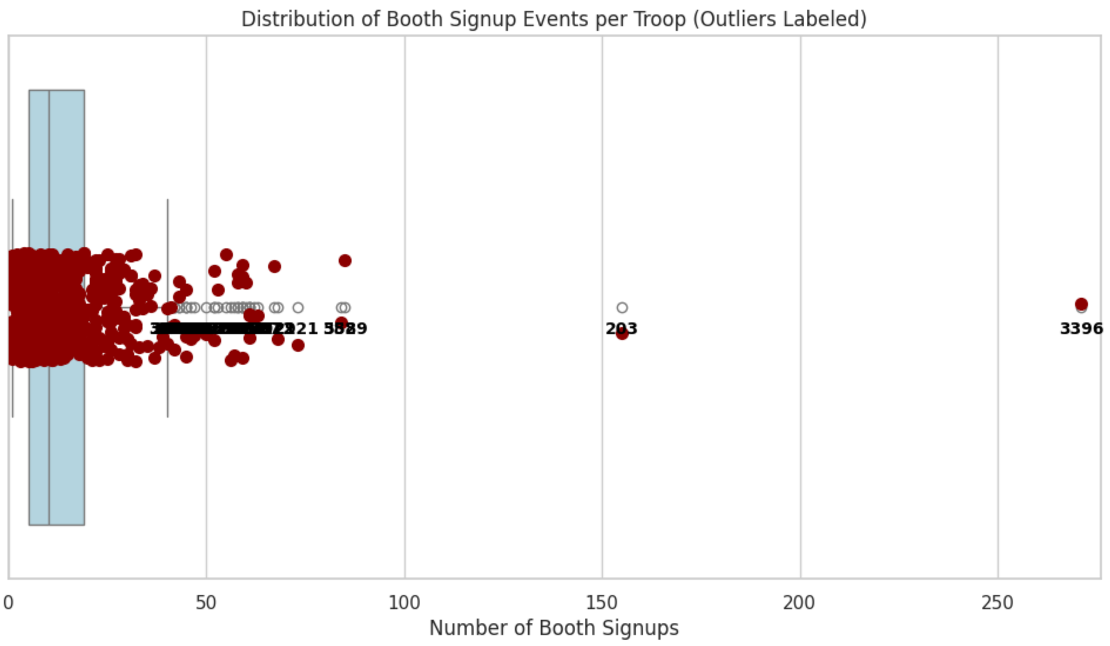
```
Descriptive statistics for booth signups per troop:
count    627.000000
mean      14.827751
std       17.839872
min        1.000000
25%        5.000000
50%       10.000000
75%       19.000000
max      271.000000
Name: Num_Booths, dtype: float64
Q1: 5.0, Q3: 19.0, IQR: 14.0
Lower bound: -16.0, Upper 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
370   3983          59
332   3822          59
30     436          59
88    1682          58
568   7073          58
286   3561          57
6      123          56
448   4520          55
518   6310          53
587   7425          52
498   5381          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
170   2589           42
28     417           42
203   3092           41
299   3605           41
```

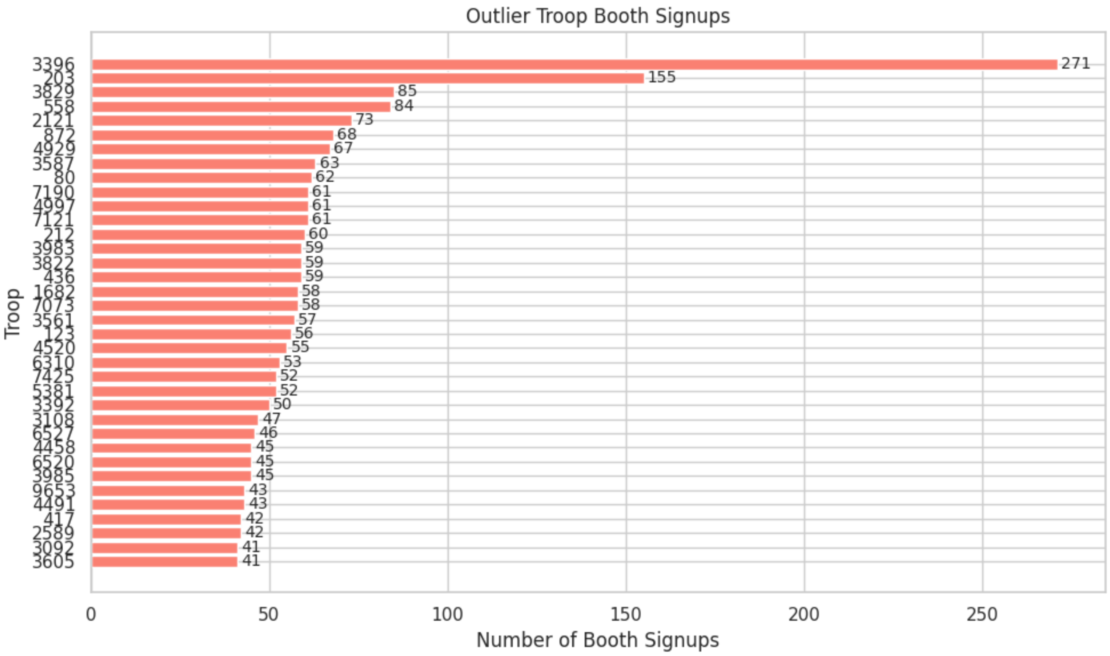
1.1 5. Visualization with Outlier Labels

Below is a boxplot of the booth signup counts per troop. In addition, we overlay a stripplot (with jitter) for each troop and label the outlier points with the corresponding troop number. We also rescale the x-axis so that the non-outlier data is not bunched up.

```
[9]: plt.figure(figsize=(12,6))

     # Create a horizontal boxplot (vertical orientation) with the signup counts.
     ax = sns.boxplot(x='Num_Booths', data=troop_counts, color='lightblue')
     sns.stripplot(x='Num_Booths', data=troop_counts, color='darkred', size=8,␣
      ↪jitter=True, ax=ax)

     # Extend the x-axis: set the limit from 0 to (the maximum number of signups␣
      ↪among troops + a margin).
     max_val = troop_counts['Num_Booths'].max()
     plt.xlim(0, max_val + 5)

     # Annotate outlier points with their Troop number
     for index, row in troop_counts.iterrows():
         if row['Num_Booths'] < lower_bound or row['Num_Booths'] > upper_bound:
             ax.text(row['Num_Booths'], 0.05, 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()
```



[10]:
```python
# Additional Visualization: Bar Chart for Outlier Troops Only

# Filter the aggregated data to include only outlier troops
outlier_troops = troop_counts[(troop_counts['Num_Booths'] < lower_bound) |␣
 ↪(troop_counts['Num_Booths'] > upper_bound)]

# If there are outliers, create a horizontal bar chart; otherwise, print a␣
 ↪message.
if outlier_troops.empty:
    print("No outlier troops detected based on the current IQR thresholds.")
else:
    # Sort the outlier data by number of booth signups in ascending order
    outlier_troops_sorted = outlier_troops.sort_values('Num_Booths',␣
 ↪ascending=True)

    plt.figure(figsize=(10, 6))
    # Convert the Troop column to string for better labeling on the y-axis.
    plt.barh(outlier_troops_sorted['Troop'].astype(str),␣
 ↪outlier_troops_sorted['Num_Booths'], color='salmon')
    plt.xlabel("Number of Booth Signups")
    plt.ylabel("Troop")
    plt.title("Outlier Troop Booth Signups")

    # Annotate each bar with its signup count, with a small horizontal offset
    for i, (troop, count) in enumerate(zip(outlier_troops_sorted['Troop'], outlier_troops_sorted['Num_Booths'])):
        plt.text(count + 1, i, str(count), va='center', fontsize=10)

    plt.tight_layout()
    plt.show()
```



1.2 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 reporting or analysis.

[11]:
```
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
```