Dungeons and Dragons - Datathon Solution

Visualization of Critical Role Campaign 1 Speech Patterns

Prepared By: Hariharan Ramesh, Jean Paul, Akshay Prassanna Sivaprakash & Jennithra Srinivasan

Toolset Used: Python, Tableau, Microsoft Excel

Dataset: Critical Role Campaign 1 Datapack (Easy Dataset)

Objective

To visualize character speeches over time in Critical Role Campaign 1, showcasing who spoke when and how often using a clean and interactive streamgraph.

Dataset Overview

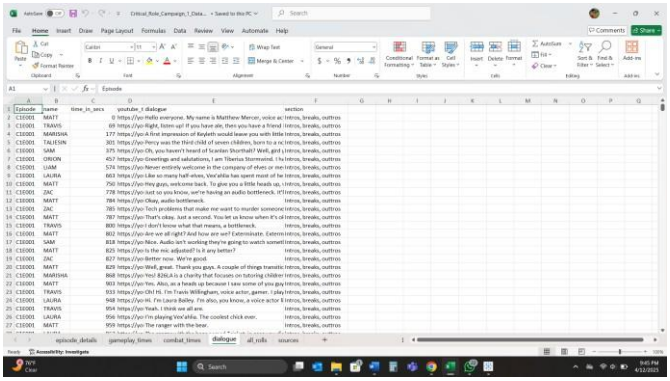
Dataset Used:

Critical_Role_Campaign_1_Datapack.xlsx

Why this dataset?

- It provides structured transcripts of Critical Role Campaign 1.
- Suitable for creating timeline-based visualizations.
- Contains columns like Character, Episode, Timestamp, and Dialogue Count—perfect for a speech timeline.

Raw Dataset Screenshot



Data Cleaning (Python)

Tools Used: Python (Pandas, NumPy)

Steps:

- Loaded the Excel file using pandas.read_excel().
- Filtered necessary columns - player, character, role, speech_time.
- Grouped and aggregated dialogue lengths or counts per episode and character.
- Saved cleaned data into Cleaned Dataset.csv.

Code Snippet Screenshot:

```python
import pandas as pd

# Loading the Excel file to inspect sheet names
file_path = "/content/Critical_Role_Campaign_1_Datapack.xlsx"
excel_file = pd.ExcelFile(file_path)

# Displaying sheet names to determine which ones to load
excel_file.sheet_names
```

```python
# Sorting the dataframe first
dialogue_df = dialogue_df.sort_values(by=['Episode', 'time_in_secs']).reset_index(drop=True)

# Creating the end_time column BEFORE renaming anything
dialogue_df['end_time'] = dialogue_df.groupby('Episode')['time_in_secs'].shift(-1)

# Filling NaN in end_time with +5 seconds from time_in_secs
dialogue_df['end_time'] = dialogue_df['end_time'].fillna(dialogue_df['time_in_secs'] + 5)

# NOW renaming the time_in_secs to start_time
dialogue_df = dialogue_df.rename(columns={'time_in_secs': 'start_time'})

# Calculating duration
dialogue_df['duration'] = dialogue_df['end_time'] - dialogue_df['start_time']
```

```python
# Flagging where the speaker or episode changes (new group)
dialogue_df['speaker_change'] = (
    (dialogue_df['name'] != dialogue_df['name'].shift(1)) |
    (dialogue_df['Episode'] != dialogue_df['Episode'].shift(1))
).astype(int)

# Creating a group ID that increments every time there's a change
dialogue_df['speech_group'] = dialogue_df['speaker_change'].cumsum()

# Grouping the data by group ID, Episode, and Speaker
grouped_df = dialogue_df.groupby(['speech_group', 'Episode', 'name']).agg({
    'start_time': 'min',
    'end_time': 'max',
    'dialogue': lambda x: ' '.join(str(d) for d in x if pd.notna(d))  # Combine text
}).reset_index()

# Recalculating duration for grouped speeches
grouped_df['duration'] = grouped_df['end_time'] - grouped_df['start_time']
```

```python
all_rolls_df = pd.read_excel(file_path, sheet_name='all_rolls')
print(all_rolls_df.columns)
```

```python
# Selecing only the relevant columns from all_rolls_df
mapping_df = all_rolls_df[['Player', 'Character']].drop_duplicates()

# Merging with grouped_df using name <-> Player
final_df = grouped_df.merge(mapping_df, left_on='name', right_on='Player', how='left')
```

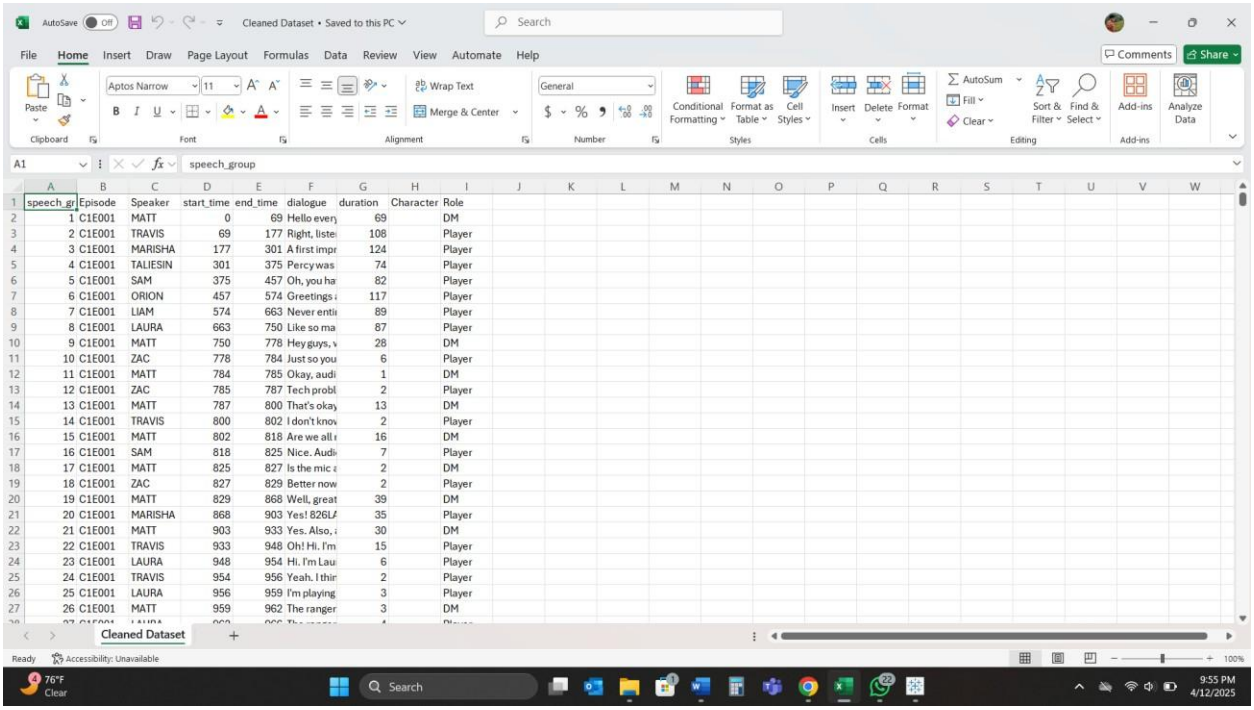
```python
# Dropping the duplicate 'Player' column if needed
final_df = final_df.drop(columns=['Player'])

# Renaming columns for Tableau clarity
final_df = final_df.rename(columns={
    'name': 'Speaker',
    'Character': 'Character'
})

# Assigning a 'Role'
final_df['Role'] = final_df['Speaker'].apply(lambda x: 'DM' if x.upper() == 'MATT' else 'Player')
```

```python
final_df.to_csv('/content/character_speech_blocks.csv', index=False)
```

Sample of Cleaned Data:



speech_gr	Episode	Speaker	start_time	end_time	dialogue	duration	Character	Role
1	C1E001	MATT	0	69	Hello every	69		DM
2	C1E001	TRAVIS	69	177	Right, liste	108		Player
3	C1E001	MARISHA	177	301	A first impr	124		Player
4	C1E001	TALIESIN	301	375	Percy was	74		Player
5	C1E001	SAM	375	457	Oh, you ha	82		Player
6	C1E001	ORION	457	574	Greetings	117		Player
7	C1E001	LIAM	574	663	Never enti	89		Player
8	C1E001	LAURA	663	750	Like so ma	87		Player
9	C1E001	MATT	750	778	Hey guys, v	28		DM
10	C1E001	ZAC	778	784	Just so you	6		Player
11	C1E001	MATT	784	785	Okay, audi	1		DM
12	C1E001	ZAC	785	787	Tech probl	2		Player
13	C1E001	MATT	787	800	That's okay	13		DM
14	C1E001	TRAVIS	800	802	I don't knov	2		Player
15	C1E001	MATT	802	818	Are we all r	16		DM
16	C1E001	SAM	818	825	Nice. Audi	7		Player
17	C1E001	MATT	825	827	Is the mic a	2		DM
18	C1E001	ZAC	827	829	Better now	2		Player
19	C1E001	MATT	829	868	Well, great	39		DM
20	C1E001	MARISHA	868	903	Yes! 826LA	35		Player
21	C1E001	MATT	903	933	Yes. Also, a	30		DM
22	C1E001	TRAVIS	933	948	Oh! Hi. I'm	15		Player
23	C1E001	LAURA	948	954	Hi. I'm Laur	6		Player
24	C1E001	TRAVIS	954	956	Yeah. I thir	2		Player
25	C1E001	LAURA	956	959	I'm playing	3		Player
26	C1E001	MATT	959	962	The ranger	3		DM

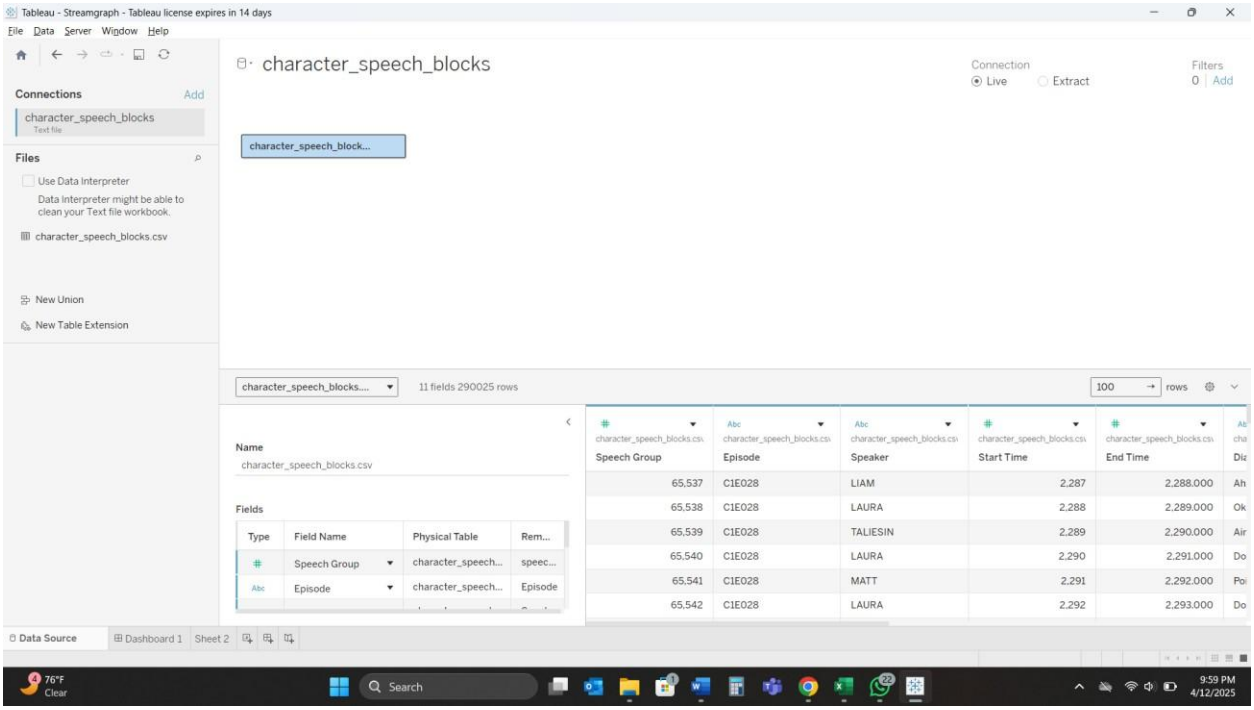
Data Preparation for Tableau

Key Columns Prepared:

- Character
- Episode
- Speech Start Time
- Speech Duration
- Player
- Role

These were prepared for time-based binning and volume calculation.

Screenshot of CSV upload in Tableau



Visualization with Tableau

Chart Type Chosen: Streamgraph

Why Streamgraph?

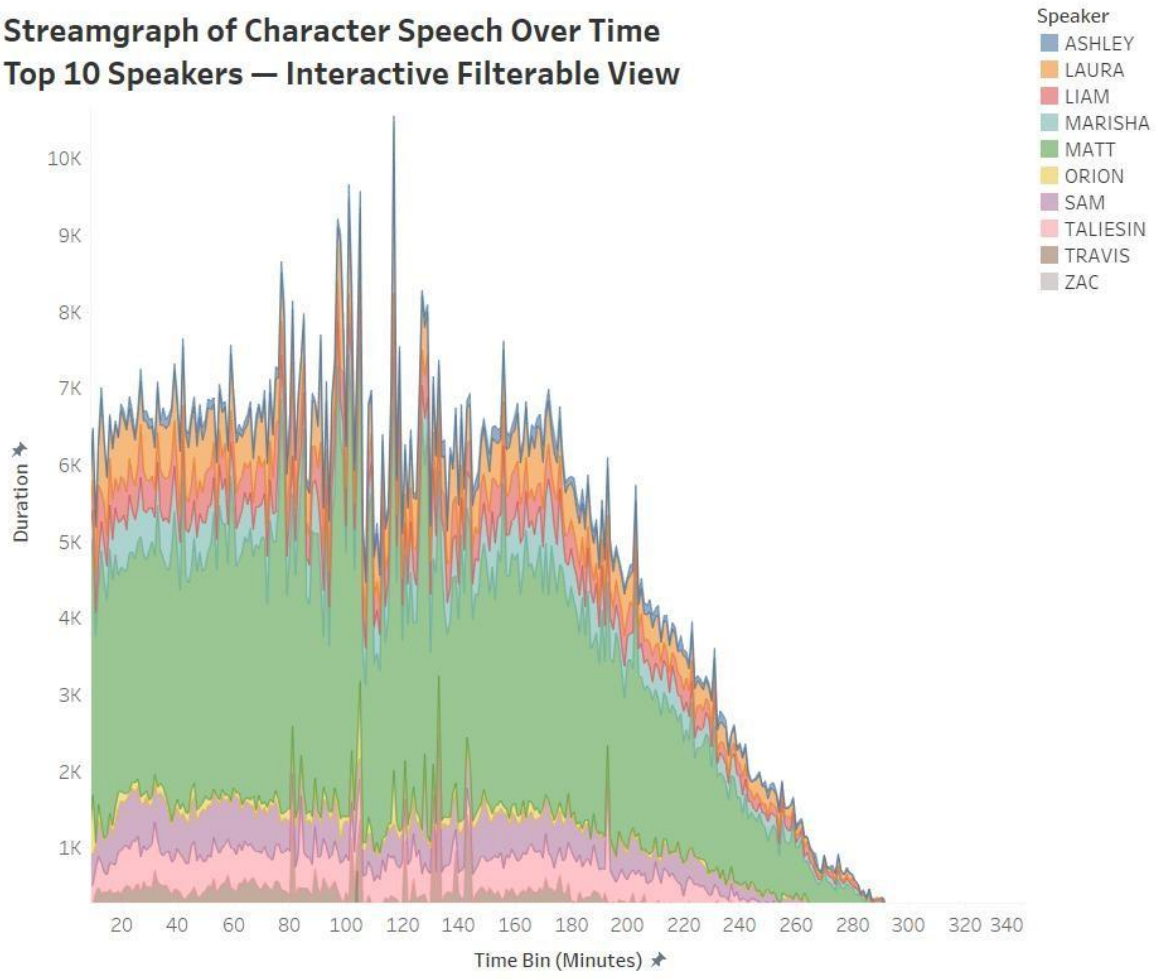
- Clearly shows volume of speech per character across time.
- Offers a layered flow that is intuitive to track speech distribution.

Calculation Highlights

Time Binning: INT([Start Time] / 60)

Aggregated Speech Count: SUM([Words]) or similar measure

Final Streamgraph Output



Filters Added

- Character
- Role
- Player
- Episode

Insights & Conclusion

- Characters like Vex'ahlia, Grog, and Vax'ildan dominate the dialogue in early episodes.
- Streamgraph visually shows role-based and player-based dialogue shifts across campaigns.
- Smooth transition in speeches shows engagement arcs of characters.