review_analysis_nms

August 6, 2024

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
from wordcloud import WordCloud
import numpy as np
from IPython.display import display, Markdown, Latex
```

[2]:
```
# Define the game name variable
GAME_NAME = "No Man's Sky"  # Change this to the name of the game you're␣
↪analyzing
```

[3]:
```
# Ensure plots are displayed inline
%matplotlib inline
```

[4]:
```
# Set default figure size for better PDF output
plt.rcParams['figure.figsize'] = [10, 6]
```

[5]:
```
# Load the data
df = pd.read_csv('../Review CSVs/reviews_275850.csv')
```

[6]:
```
# Convert timestamp_created to datetime
df['timestamp_created'] = pd.to_datetime(df['timestamp_created'])
```

[7]:
```
# Display basic information about the loaded data
print(f"Data loaded successfully for {GAME_NAME}")
print(f"Number of reviews: {len(df)}")
print(f"Date range: from {df['timestamp_created'].min()} to␣
↪{df['timestamp_created'].max()}")
```

```
Data loaded successfully for No Man's Sky
Number of reviews: 183349
Date range: from 2016-08-12 17:10:35 to 2024-07-29 18:39:45
```

[8]:
```
# Filter data from January 2019 to June 2024
start_date = '2019-01-01'
end_date = '2024-06-30'
```

```python
df_filtered = df[(df['timestamp_created'] >= start_date) & (df['timestamp_created'] <= end_date)]

print(f"\nNumber of reviews after filtering (Jan 2019 - Jun 2024): {len(df_filtered)}")
```

```
Number of reviews after filtering (Jan 2019 - Jun 2024): 107777
```

[9]:
```python
# Create a table of contents
display(Markdown(f"# Steam Review Analysis: {GAME_NAME}"))
display(Markdown("## Table of Contents"))
display(Markdown("""
1. [Monthly Review Trends](#monthly-review-trends)
2. [Sentiment Analysis Over Time](#sentiment-analysis-over-time)
3. [Playtime vs Review Sentiment](#playtime-vs-review-sentiment)
4. [Review Length Analysis](#review-length-analysis)
5. [Early Access Impact](#early-access-impact)
6. [Language Distribution](#language-distribution)
7. [Player Experience Level](#player-experience-level)
8. [Review Helpfulness Over Time](#review-helpfulness-over-time)
9. [Seasonal Trends](#seasonal-trends)
10. [Word Frequency Analysis](#word-frequency-analysis)
"""))
```

1 Steam Review Analysis: No Man's Sky

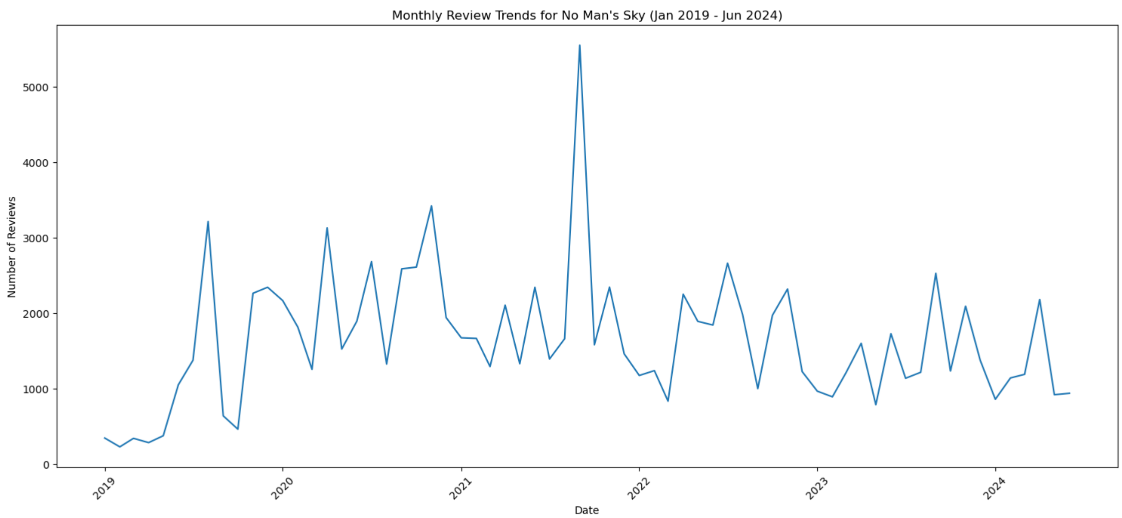
1.1 Table of Contents

[10]:
```python
# Function to create section headers with anchors for the table of contents
def section_header(title, anchor):
    display(Markdown(f"<a id='{anchor}'></a>"))
    display(Markdown(f"## {title}"))
```

[11]:
```python
# 1. Monthly Review Trends
section_header("Monthly Review Trends", "monthly-review-trends")
```

```
"""
This visualization shows the ebb and flow of review activity over time.

- Spikes might indicate major updates, sales, or viral moments.
- Troughs could suggest periods of lower player interest or game issues.
- The overall trend can reveal if the game is gaining or losing momentum.

"""

monthly_reviews = df_filtered.groupby(df_filtered['timestamp_created'].dt.
 ↪to_period("M")).size().reset_index(name='count')
monthly_reviews['timestamp_created'] = monthly_reviews['timestamp_created'].dt.
 ↪to_timestamp()

plt.figure(figsize=(15, 7))
plt.plot(monthly_reviews['timestamp_created'], monthly_reviews['count'])
plt.title(f"Monthly Review Trends for {GAME_NAME} (Jan 2019 - Jun 2024)")
plt.xlabel('Date')
plt.ylabel('Number of Reviews')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

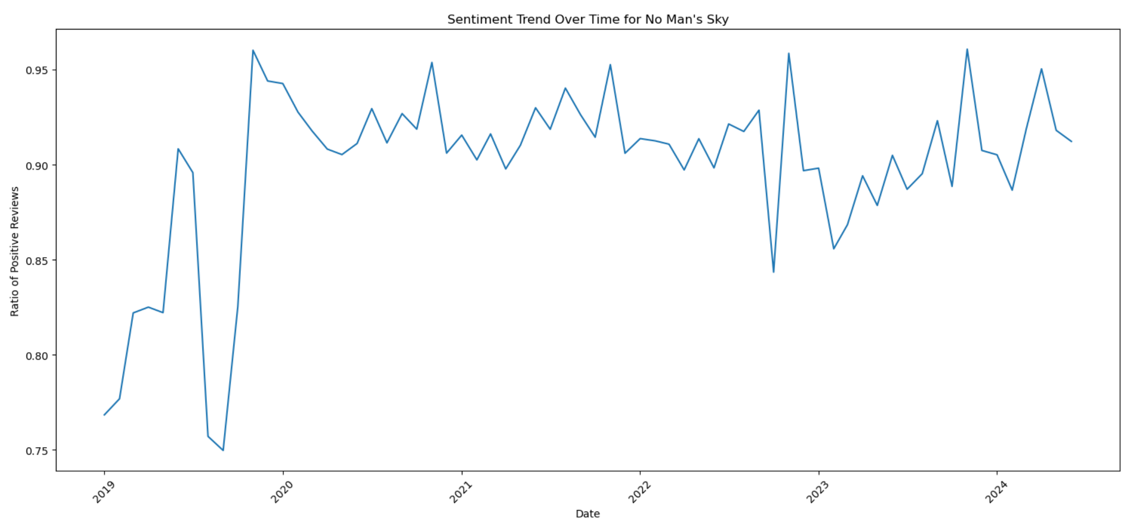
1.2 Monthly Review Trends



[12]:
```
# 2. Sentiment Analysis Over Time
section_header("Sentiment Analysis Over Time", "sentiment-analysis-over-time")

"""
```

```
Here we're tracking the game's approval rating over its lifetime.

- Upward trends suggest improvements or positive updates.
- Downward trends might indicate issues or unpopular changes.
- Stability could mean consistent quality or stagnation.

"""

df_filtered = df_filtered.copy()  # Create an explicit copy
df_filtered.loc[:, 'month'] = df_filtered['timestamp_created'].dt.to_period('M')
sentiment_over_time = df_filtered.groupby('month').agg({'voted_up': 'mean'}).
 ↪reset_index()
sentiment_over_time['month'] = sentiment_over_time['month'].dt.to_timestamp()

plt.figure(figsize=(15, 7))
plt.plot(sentiment_over_time['month'], sentiment_over_time['voted_up'])
plt.title(f"Sentiment Trend Over Time for {GAME_NAME}")
plt.xlabel('Date')
plt.ylabel('Ratio of Positive Reviews')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

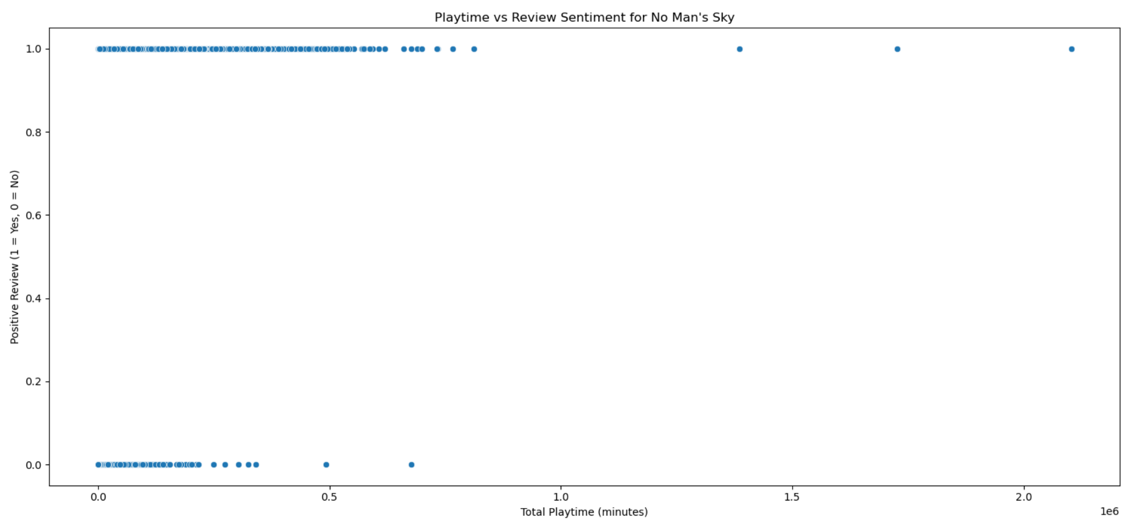
1.3 Sentiment Analysis Over Time



[13]:
```
# 3. Playtime vs Review Sentiment
section_header("Playtime vs Review Sentiment", "playtime-vs-review-sentiment")

"""
```

```
This shows the relationship between how long people play and how they feel␣
↪about it.

- Clusters in the top-right are die-hard fans: long playtime and positive␣
↪reviews.
- Bottom-left clusters might be disappointed players or those who couldn't get␣
↪into the game.
- Outliers tell interesting stories: loved it but barely played, or played␣
↪forever but didn't like it?

"""

plt.figure(figsize=(15, 7))
sns.scatterplot(data=df_filtered, x='author_playtime_forever', y='voted_up')
plt.title(f"Playtime vs Review Sentiment for {GAME_NAME}")
plt.xlabel('Total Playtime (minutes)')
plt.ylabel('Positive Review (1 = Yes, 0 = No)')
plt.tight_layout()
plt.show()
```

1.4 Playtime vs Review Sentiment



[14]:
```
# 4. Review Length Analysis
section_header("Review Length Analysis", "review-length-analysis")


"""
This histogram shows how verbose (or concise) the reviewers tend to be.
```

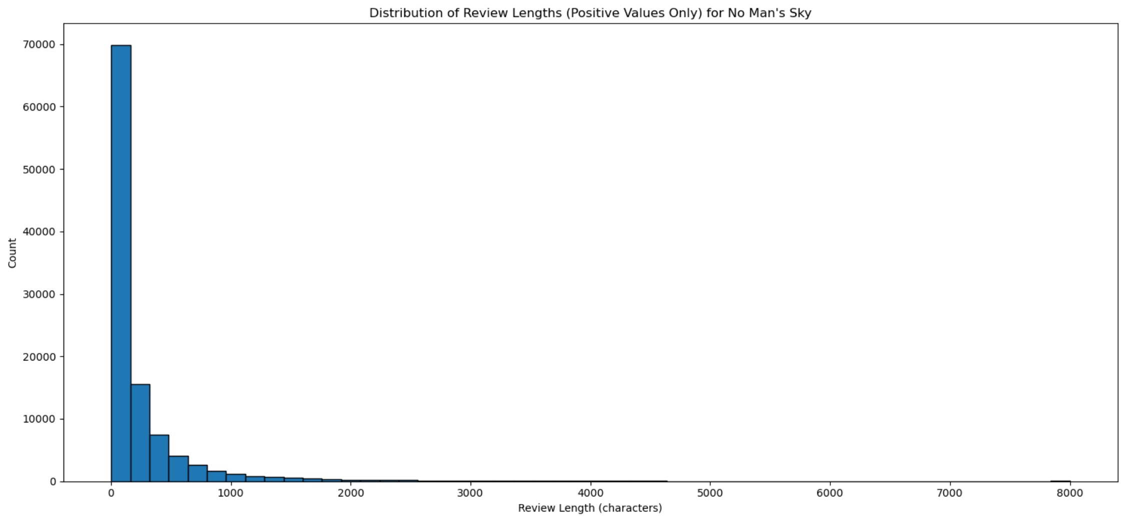
```
- A peak at lower lengths might indicate quick, emotional responses.
- Longer reviews could suggest more thoughtful, detailed feedback.
- The shape of the distribution can tell you about the reviewers' habits.

"""

df_filtered.loc[:, 'review_length'] = df_filtered['review'].str.len()

# Data validation
print("Review Length Statistics:")
print(df_filtered['review_length'].describe())

print("\nUnique review lengths:")
print(df_filtered['review_length'].value_counts().sort_index())

# Check for non-positive values
non_positive = df_filtered[df_filtered['review_length'] <= 0]
print(f"\nNumber of non-positive review lengths: {len(non_positive)}")

if len(non_positive) > 0:
    print("Sample of reviews with non-positive lengths:")
    print(non_positive[['review', 'review_length']].head())

# Proceed with visualization only if we have positive values
positive_lengths = df_filtered[df_filtered['review_length'] >␣
↪0]['review_length']

if len(positive_lengths) > 0:
    # Histogram
    plt.figure(figsize=(15, 7))
    plt.hist(positive_lengths, bins=50, edgecolor='black')
    plt.title(f"Distribution of Review Lengths (Positive Values Only) for␣
↪{GAME_NAME}")
    plt.xlabel('Review Length (characters)')
    plt.ylabel('Count')
    plt.tight_layout()
    plt.show()

    # Box Plot
    plt.figure(figsize=(10, 6))
    bp = plt.boxplot(positive_lengths, vert=False, whis=[5, 95])
    plt.title(f"Box Plot of Review Lengths (5th to 95th percentile, Positive␣
↪Values Only) for {GAME_NAME}")
    plt.xlabel('Review Length (characters)')
    plt.xscale('log')  # Use log scale for x-axis

    # Add labels for quartiles
```

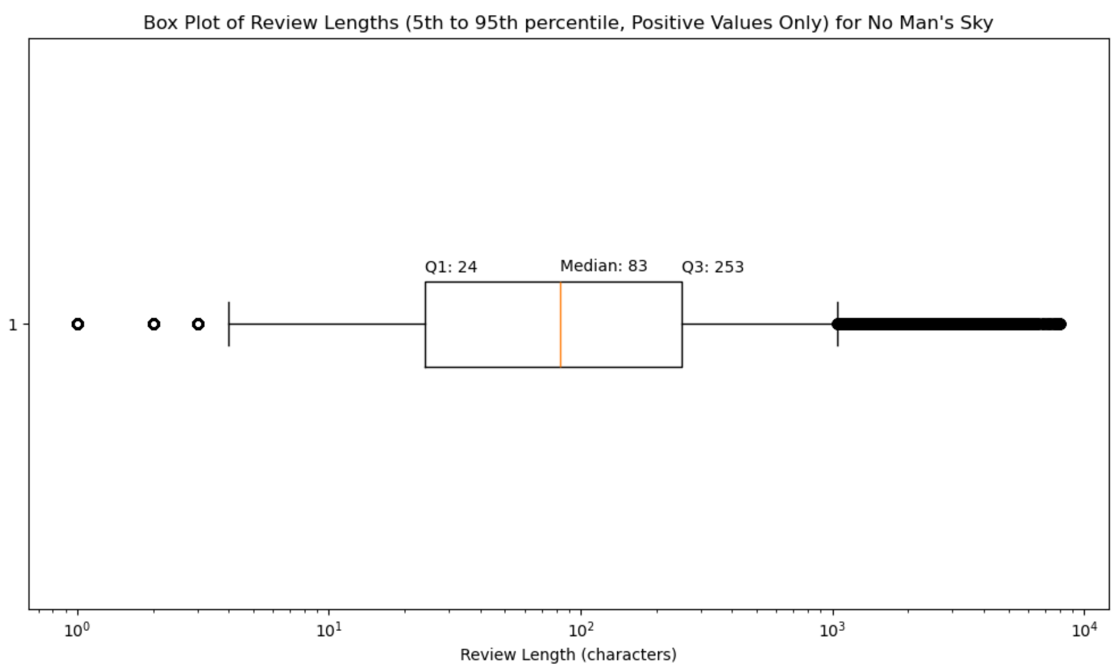
```
    quartiles = positive_lengths.quantile([0.25, 0.5, 0.75])
    for i, q in enumerate(['Q1', 'Median', 'Q3']):
        plt.text(quartiles.iloc[i], 1.1, f'{q}: {quartiles.iloc[i]:.0f}',
                 verticalalignment='center')

    plt.tight_layout()
    plt.show()

    # Print additional percentiles for context
    percentiles = positive_lengths.quantile([0.05, 0.25, 0.5, 0.75, 0.95])
    print("\nReview Length Percentiles (Positive Values Only):")
    for p, v in percentiles.items():
        print(f"{p*100:.0f}th percentile: {v:.0f} characters")

    # Correlation between review length and sentiment
    correlation = positive_lengths.corr(df_filtered.loc[positive_lengths.index,␣
↪'voted_up'])
    print(f"\nCorrelation between review length and positive sentiment:␣
↪{correlation:.2f}")
else:
    print("No positive review lengths found. Cannot create visualizations.")

# If we have non-positive values, let's investigate further
if len(non_positive) > 0:
    print("\nAnalysis of non-positive review lengths:")
    print(non_positive['review_length'].value_counts().sort_index())
    print("\nSample of reviews with non-positive lengths:")
    print(non_positive[['review', 'review_length']].head())
```

1.5 Review Length Analysis

```
Review Length Statistics:
count    107362.000000
mean        256.116848
std         547.300654
min           1.000000
25%          24.000000
50%          83.000000
75%         253.000000
max        8000.000000
Name: review_length, dtype: float64

Unique review lengths:
review_length
1.0         889
2.0         699
3.0        2040
```

```
4.0        2143
5.0        1368
           …
7994.0        2
7997.0        2
7998.0        2
7999.0        4
8000.0       11
Name: count, Length: 3236, dtype: int64

Number of non-positive review lengths: 0
```



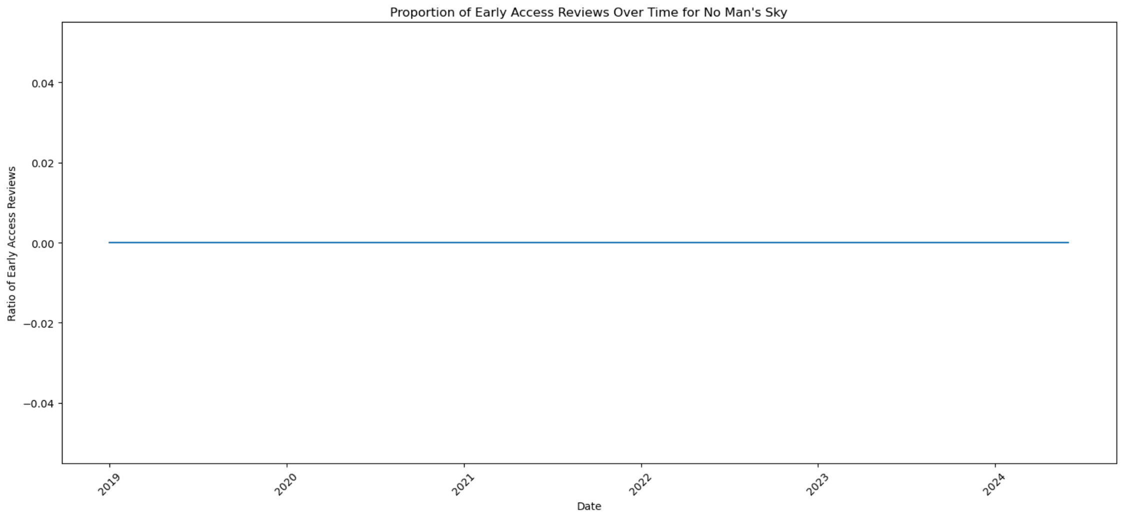


```
Review Length Percentiles (Positive Values Only):
5th percentile: 4 characters
25th percentile: 24 characters
50th percentile: 83 characters
75th percentile: 253 characters
95th percentile: 1044 characters

Correlation between review length and positive sentiment: -0.18
```

[15]:

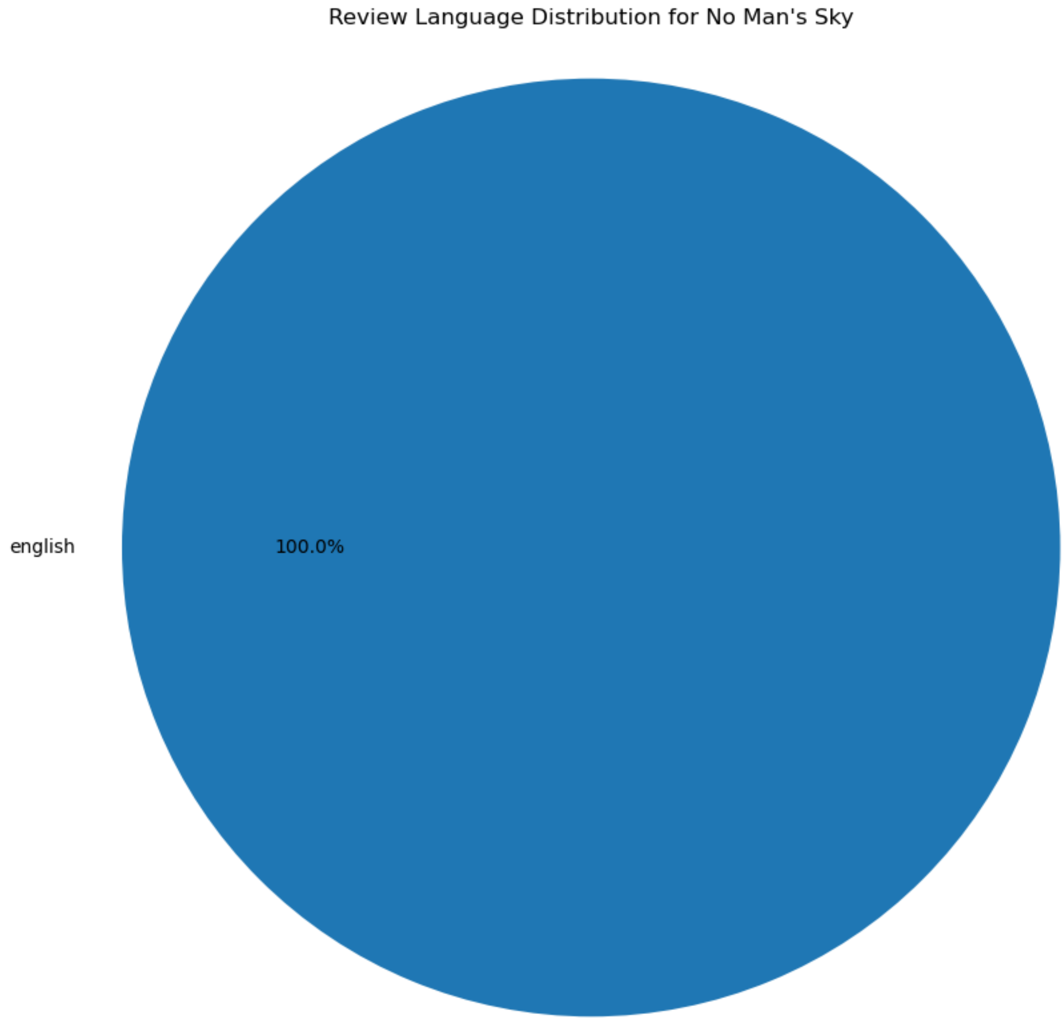
```
# 5. Early Access Impact
section_header("Early Access Impact", "early-access-impact")


"""
This chart shows how the Early Access phase influenced player opinions over␣
↪time.

- High early proportions show initial early adopter enthusiasm.
- The trend downwards indicates the transition to full release.
- Post-release blips might suggest nostalgia or comparisons to the early days.

"""

early_access_trend = df_filtered.groupby('month').
↪agg({'written_during_early_access': 'mean'}).reset_index()
early_access_trend['month'] = early_access_trend['month'].dt.to_timestamp()

plt.figure(figsize=(15, 7))
plt.plot(early_access_trend['month'],␣
↪early_access_trend['written_during_early_access'])
plt.title(f"Proportion of Early Access Reviews Over Time for {GAME_NAME}")
plt.xlabel('Date')
plt.ylabel('Ratio of Early Access Reviews')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

1.6 Early Access Impact



[16]:
```
# 6. Language Distribution
section_header("Language Distribution", "language-distribution")


"""
This pie chart shows the global linguistic reach and cultural impact of the␣
↪game.

- Dominant languages indicate the primary markets.
- Diverse small slices suggest a global cult following.
- Unexpected languages might reveal surprising new markets.

"""

language_dist = df_filtered['language'].value_counts()
plt.figure(figsize=(10, 10))
plt.pie(language_dist.values, labels=language_dist.index, autopct='%1.1f%%')
plt.title(f"Review Language Distribution for {GAME_NAME}")
plt.axis('equal')
plt.show()
```

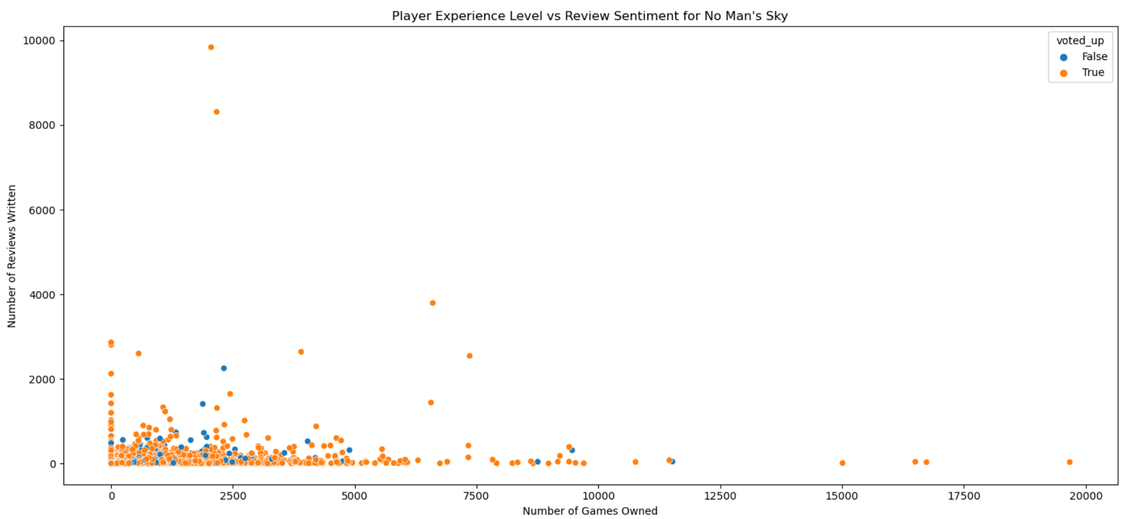
1.7 Language Distribution



[17]:
```
# 7. Player Experience Level
section_header("Player Experience Level", "player-experience-level")


"""
This scatter plot is like a gamer demographic survey for the reviewers.
It shows how experienced Steam users react to the game.

- Clusters can reveal the core audience: newbies, veterans, or a mix?
- The color distribution shows how different player types rate the game.
- Outliers might be influential reviewers or unique player experiences.
```

```
It's a way to see if the game is a hit with the Steam elite or a gateway game␣
↪for newcomers!

"""

plt.figure(figsize=(15, 7))
sns.scatterplot(data=df_filtered, x='author_num_games_owned',␣
↪y='author_num_reviews', hue='voted_up')
plt.title(f"Player Experience Level vs Review Sentiment for {GAME_NAME}")
plt.xlabel('Number of Games Owned')
plt.ylabel('Number of Reviews Written')
plt.tight_layout()
plt.show()
```

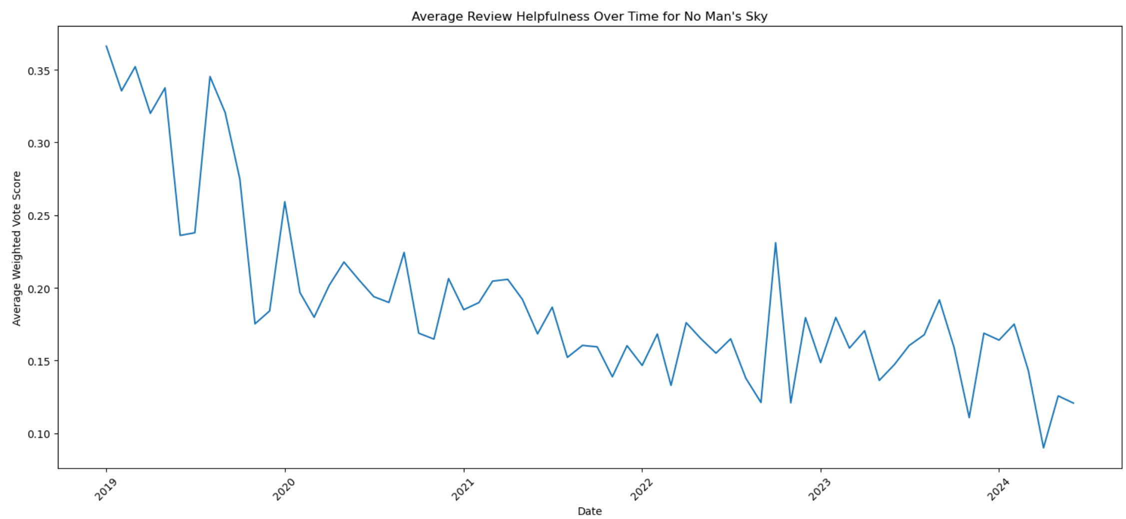
1.8 Player Experience Level



[18]:
```
# 8. Review Helpfulness Over Time
section_header("Review Helpfulness Over Time", "review-helpfulness-over-time")



"""
This trend line shows how useful other players find the reviews.

- Upward trends suggest more insightful or balanced reviews over time.
- Downward trends might indicate controversial periods or review bombing.
- Stability could mean consistent community engagement.

"""
```

```
helpfulness_over_time = df_filtered.groupby('month').agg({'weighted_vote_score':
 ↪ 'mean'}).reset_index()
helpfulness_over_time['month'] = helpfulness_over_time['month'].dt.
 ↪to_timestamp()

plt.figure(figsize=(15, 7))
plt.plot(helpfulness_over_time['month'],␣
 ↪helpfulness_over_time['weighted_vote_score'])
plt.title(f"Average Review Helpfulness Over Time for {GAME_NAME}")
plt.xlabel('Date')
plt.ylabel('Average Weighted Vote Score')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

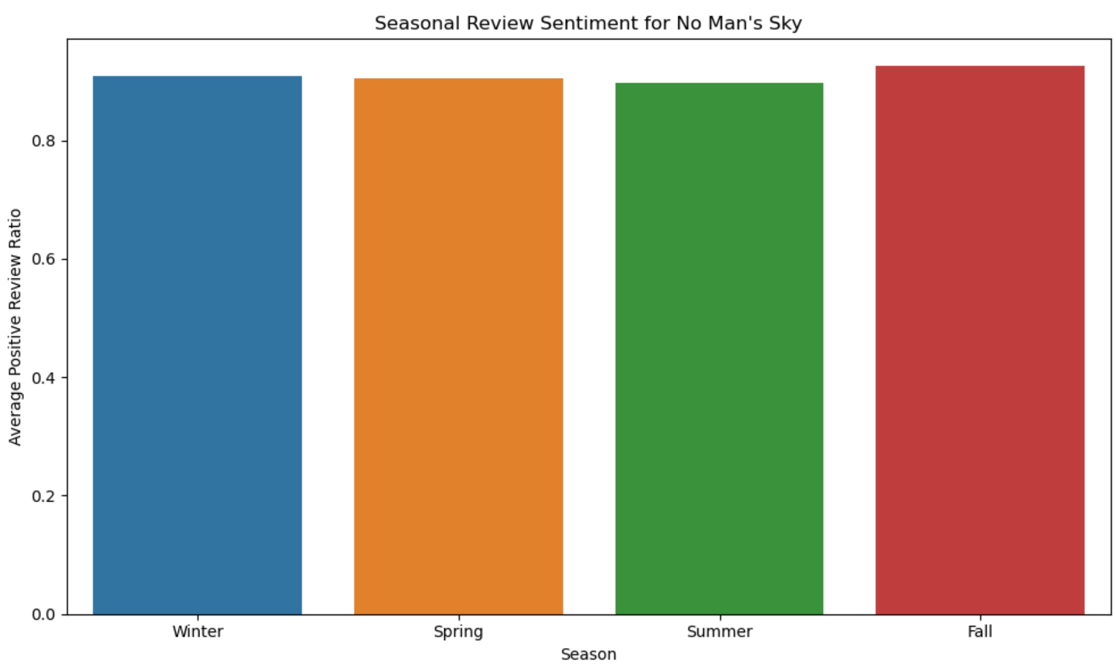
1.9 Review Helpfulness Over Time



[19]:
```
# 9. Seasonal Trends
section_header("Seasonal Trends", "seasonal-trends")


"""
This bar chart shows how the time of year affects player sentiment.

- Higher bars indicate seasons when players tend to enjoy the game more.
- Lower bars might suggest seasonal challenges or competition.
- Patterns could reveal optimal times for updates or promotions.

"""
```

```
df_filtered.loc[:, 'season'] = df_filtered['timestamp_created'].dt.month.map({1:
 ↪'Winter', 2:'Winter', 3:'Spring',
                                                                             4:
 ↪'Spring', 5:'Spring', 6:'Summer',
                                                                             7:
 ↪'Summer', 8:'Summer', 9:'Fall',
                                                                            ␣
 ↪10:'Fall', 11:'Fall', 12:'Winter'})
seasonal_sentiment = df_filtered.groupby('season')['voted_up'].mean().
 ↪reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='season', y='voted_up', data=seasonal_sentiment, order=['Winter',␣
 ↪'Spring', 'Summer', 'Fall'])
plt.title(f"Seasonal Review Sentiment for {GAME_NAME}")
plt.xlabel('Season')
plt.ylabel('Average Positive Review Ratio')
plt.tight_layout()
plt.show()
```

1.10 Seasonal Trends



[20]:
```
# 10. Word Frequency Analysis (Word Cloud)
section_header("Word Frequency Analysis", "word-frequency-analysis")
```

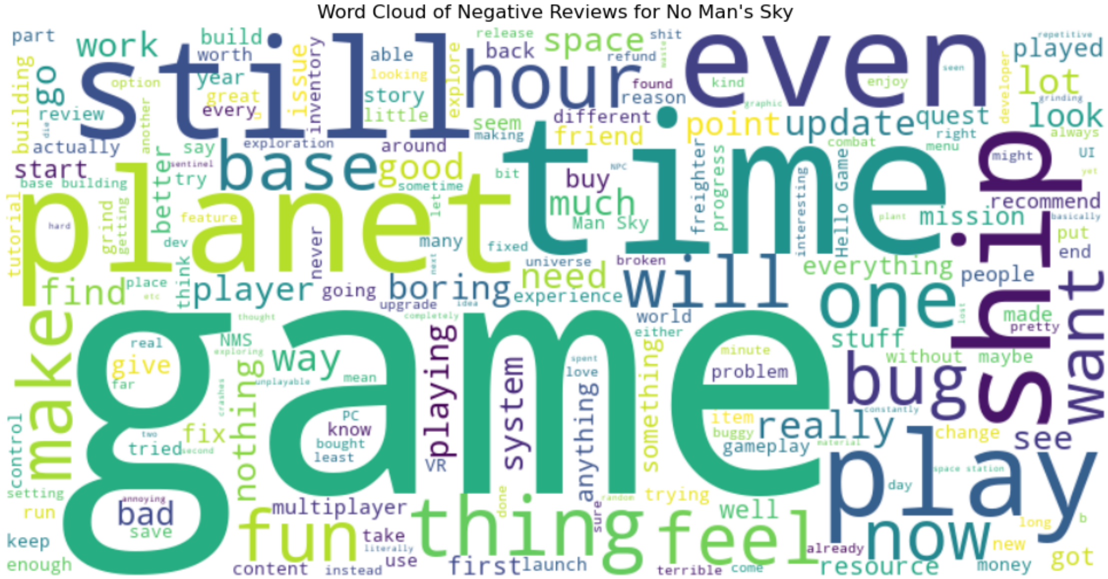
```python
"""
These word clouds are like thought bubbles floating above the positive and␣
↪negative reviewers.
They visually represent the most common terms used in reviews.

- Larger words are more frequently mentioned and potentially more important.
- Positive cloud might reveal beloved features or emotions.
- Negative cloud could highlight pain points or areas for improvement.

"""


def clean_reviews(reviews):
    return ' '.join([str(review) for review in reviews if isinstance(review,␣
↪str)])

positive_reviews = clean_reviews(df_filtered[df_filtered['voted_up'] ==␣
↪True]['review'])
negative_reviews = clean_reviews(df_filtered[df_filtered['voted_up'] ==␣
↪False]['review'])

def generate_wordcloud(text, title):
    if not text:
        print(f"No valid text for {title}. Skipping word cloud generation.")
        return

    wordcloud = WordCloud(width=800, height=400, background_color='white').
↪generate(text)
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(title)
    plt.tight_layout(pad=0)
    plt.show()

# Print some statistics about the reviews
print(f"Number of positive reviews: {len(df_filtered[df_filtered['voted_up'] ==␣
↪True])}")
print(f"Number of negative reviews: {len(df_filtered[df_filtered['voted_up'] ==␣
↪False])}")
print(f"Number of words in positive reviews: {len(positive_reviews.split())}")
print(f"Number of words in negative reviews: {len(negative_reviews.split())}")
```

```
generate_wordcloud(positive_reviews, f'Word Cloud of Positive Reviews for␣
↪{GAME_NAME}')
generate_wordcloud(negative_reviews, f'Word Cloud of Negative Reviews for␣
↪{GAME_NAME}')

# Optional: Display some sample reviews
print("\nSample positive review:")
print(df_filtered[df_filtered['voted_up'] == True]['review'].iloc[0])

print("\nSample negative review:")
print(df_filtered[df_filtered['voted_up'] == False]['review'].iloc[0])
```

1.11 Word Frequency Analysis

```
Number of positive reviews: 98061
Number of negative reviews: 9716
Number of words in positive reviews: 4013703
Number of words in negative reviews: 1008752
```

Word Cloud of Negative Reviews for No Man's Sky




```
Sample positive review:
GOOD

Sample negative review:
This game would be absolutely amazing. However, The game seems to crash every 20
mins, and decreases the time in between crashes, the more one discovers or
crafts. I cant walk 5 feet without the game crashing now in my current save.
```

[21]:
```
# Conclusion section
display(Markdown("## Conclusion"))
display(Markdown(f"""
This analysis provides comprehensive insights into the player reception and␣
↪trends of {GAME_NAME} on Steam. """))
```

1.12 Conclusion

This analysis provides comprehensive insights into the player reception and trends of No Man's Sky on Steam.