Impact of Adaptation Source on Anime Ratings: Statistical Analysis Report

Jacques Jiao Yuanzhe

June 26th, 2025

1 Introduction and Motivation

Anime has evolved into a global cultural phenomenon with adaptations originating from diverse source materials. Understanding how adaptation sources influence audience reception is crucial for production studios making financial decisions, streaming platforms developing content strategies, and audiences seeking quality content. This investigation examines whether source material significantly affects audience ratings across four adaptation categories: original works, manga adaptations, game adaptations, and light novel adaptations. The analysis provides evidence-based guidance for industry professionals while contributing to scholarly understanding of media adaptation dynamics.

2 Data Description and Preparation

2.1 Bangumi.tv Platform Overview

Bangumi.tv (番组计划) is a specialized Chinese-language platform dedicated to cataloging and discussing Japanese media. Founded in 2008, it serves as China's authoritative database for Japanese animation, manga, games, and music. Key platform characteristics include:

- Over 1.5 million registered users (as of 2023)
- 552,820 entries for works (as of June 24, 2025)
- Collaborative wiki-style editing with version control
- Information released under CC BY-SA license, allowing free use under certain conditions

2.2 Data Processing Pipeline

We utilize the subject.jsonlines file in the official data dump released on GitHub at bangumi/Archive. The analysis used 4,136 Japanese TV anime entries meeting the following criteria:

1. **Filtering**: Restricted to animation (type=2) with "日本" and "TV" tags and ≥ 30 ratings
2. **Classification**: Categorized into:
    - Original works (Category 1)
    - Manga adaptations (Category 2)
    - Game adaptations (Category 3)
    - Light novel adaptations (Category 4)
3. **Statistical Transformation**: Calculated:
    - Mean rating (1-10 scale)

- Entry spread (within-anime SD)
- Rating spread (between-anime SD)

4. **Sampling**: Generated representative subset (n=40 per category)

3 Statistical Analysis

3.1 Research Questions

We investigate three primary questions about anime adaptations:

1. Do different adaptation sources have different *mean ratings*?
2. Do different adaptation sources have different *within-anime rating variability*?
3. Do different adaptation sources have different *between-anime rating consistency*?

3.2 Methods

All analyses were conducted at $\alpha = 0.10$ significance level using Python 3.13.4 with the following libraries: pandas for data manipulation, scipy for statistical tests, and numpy for numerical computations. The complete analysis code is provided in Appendix A, and full environment requirements are in Appendix B.

3.2.1 Question 1: Mean Ratings

One-way ANOVA was selected to compare means across multiple independent groups. This test is appropriate for:

- Continuous response variable (mean ratings)
- Categorical predictor with > 2 groups (adaptation sources)
- Independent observations across groups

Table 1: Conditions for one-way ANOVA (Mean Ratings)

Condition	**Fulfilled**	**Justification**
Independence	Yes	Random sampling ensures independence
Normality	Yes	Shapiro-Wilk test ($p > 0.10$) confirmed normality
Homogeneity of Variance	Yes	Levene's test ($p = 0.3999 > 0.10$) confirmed equal variances

3.2.2 Question 2: Within-Anime Variability

One-way ANOVA was used to compare the mean standard deviations of ratings within individual anime. This test is appropriate for:

- Continuous measure of dispersion (within-anime SD)
- Comparison across > 2 independent groups
- Assessing systematic differences in opinion diversity

Table 2: Conditions for one-way ANOVA (Within-Anime Variability)

Condition	**Fulfilled**	**Justification**
Independence	Yes	Stratified random sampling ensures independence
Normality	Yes	Shapiro-Wilk test ($p > 0.10$) confirmed normality
Homogeneity of Variance	Yes	Levene's test ($p = 0.1349 > 0.10$) confirmed equal variances

3.2.3 Question 3: Between-Anime Consistency

Levene's test was chosen to compare the variances of mean ratings across anime titles within categories. This test is appropriate for:

- Assessing homogeneity of variances across groups
- Robust to departures from normality
- Evaluating rating consistency across multiple adaptations

Table 3: Conditions for Levene's Test (Between-Anime Consistency)

Condition	Fulfilled	Justification
Independence	Yes	Random sampling ensures independence
Continuous Data	Yes	Interval scale ratings (1-10)
Minimum Group Size	Yes	$n = 45$ per group exceeds minimum requirements

3.3 Results

3.3.1 Question 1: Mean Ratings

Hypotheses

$$H_0 : \mu_{\text{Original}} = \mu_{\text{Manga}} = \mu_{\text{Game}} = \mu_{\text{Light Novel}}$$
$$H_a : \text{At least one mean differs}$$

Test Statistic
One-way ANOVA: $F(3, 176) = 2.8939$
P-value and Conclusion
$p = 0.0368 < 0.10$
Reject H_0. There is significant evidence that adaptation sources differ in mean ratings at $\alpha = 0.10$.

3.3.2 Question 2: Within-Anime Variability

Hypotheses

$$H_0 : \sigma_{\text{Original}} = \sigma_{\text{Manga}} = \sigma_{\text{Game}} = \sigma_{\text{Light Novel}}$$
$$H_a : \text{At least one within-anime SD differs}$$

Test Statistic
One-way ANOVA: $F(3, 176) = 1.8785$
P-value and Conclusion
$p = 0.1349 > 0.10$
Fail to reject H_0. No significant difference in within-anime rating variability at $\alpha = 0.10$.

3.3.3 Question 3: Between-Anime Consistency

Hypotheses

$$H_0 : \text{Var}(\mu_{\text{Original}}) = \text{Var}(\mu_{\text{Manga}}) = \text{Var}(\mu_{\text{Game}}) = \text{Var}(\mu_{\text{Light Novel}})$$
$$H_a : \text{At least one variance differs}$$

Test Statistic
Levene's test: $W = 0.9877$
P-value and Conclusion
$p = 0.3999 > 0.10$
Fail to reject H_0. No significant difference in between-anime rating consistency at $\alpha = 0.10$.

3.4 Discussion

Our analysis reveals three key findings at $\alpha = 0.10$:

1. **Source material significantly impacts mean ratings** ($F = 2.89$, $p = 0.0368$). Manga adaptations received the highest average ratings (6.70), while game adaptations received the lowest (5.95).
2. **Within-anime rating variability shows no significant differences** ($F = 1.88$, $p = 0.1349$). All sources exhibit similar dispersion of individual opinions about specific titles.
3. **Between-anime consistency is similar across sources** (Levene's $W = 0.99$, $p = 0.3999$). Rating patterns show comparable consistency within each adaptation category.

4 Conclusion

This study demonstrates that at $\alpha = 0.10$, adaptation source significantly influences anime ratings but not rating behavior patterns. Key implications include:

- Production studios should prioritize manga adaptations for the highest probable ratings
- Game adaptations carry a higher risk of lower average ratings
- Original works and light novel adaptations show intermediate performance
- Rating consistency patterns are similar regardless of source material

5 Appendix

The project can be accessed at https://github.com/jiaobenhaimo/Bangumi-Data-Analysis under the BSD 2-Clause License.

5.1 Appendix A: main.py

```
import csv
import json
import pandas as pd
import math
import scipy.stats as stats
import random
import os
from typing import Dict, List

CATEGORY_MAP = [
    (4, ["轻小说改", "小说改", "LN改", "轻改", "小说改编", "轻小说改编"]),
    (3, ["游戏改", "GAL改", "视觉小说改", "游戏改编", "游改", "galgame改", "改", "GAL改编"]),
    (2, ["漫画改", "漫改", "漫画改编"]),
    (1, ["原创", "原创动画"])
]

def has_japan_tag(subject: Dict) -> bool:
    japan_tag = False
    tv_tag = False

    for tag in subject.get("tags", []):
        if tag.get("name") == "日本":
            japan_tag = True
        if tag.get("name") == "TV":
            tv_tag = True

    for mtag in subject.get("meta_tags", []):
        if mtag == "日本":
            japan_tag = True
        if mtag == "TV":
            tv_tag = True

    return japan_tag and tv_tag

def classify_subject(subject: Dict) -> int:
    all_tags = []
    for tag in subject.get("tags", []):
        if "name" in tag:
            all_tags.append(tag["name"])
    all_tags.extend(subject.get("meta_tags", []))

    for cat_id, keywords in CATEGORY_MAP:
        for kw in keywords:
            if any(kw in tag for tag in all_tags):
                return cat_id

    return 0
```

```python
def sort_subject(output_path: str) -> None:
    df = pd.read_csv(output_path)
    sorted_df = df.sort_values(by=["Adapted From", "Date", "ID"], ascending=True)
    sorted_df.to_csv(output_path, index=False)

def process_subject(subject: Dict) -> Dict:
    score_details = subject.get("score_details", {})
    scores = [score_details.get(str(i), 0) for i in range(1, 11)]
    total = sum(scores)

    if total < 30:
        return None

    weighted_sum = sum(i * score for i, score in enumerate(scores, start=1))
    mean = weighted_sum / total
    variance = sum(score * (i - mean)**2 for i, score in enumerate(scores, start=1)) / total
    sd = math.sqrt(variance)

    return {
        "ID": subject["id"],
        "Date": subject.get("date", "")[:4],
        "Adapted From": classify_subject(subject),
        **{str(i): scores[i-1] for i in range(1, 11)},
        "Sum": total,
        "Mean": mean,
        "Standard Deviation": sd
    }

def process_jsonl(input_path: str, output_path: str) -> None:
    data = []

    with open(input_path, "r", encoding="utf-8") as f:
        for line in f:
            try:
                subject = json.loads(line.strip())
                if (subject.get("type") != 2 or
                    not has_japan_tag(subject) or
                    not subject.get("date")):
                    continue

                if row := process_subject(subject):
                    data.append(row)
            except (json.JSONDecodeError, KeyError) as e:
                print(f"Error processing line: {str(e)}")

    df = pd.DataFrame(data)
    if not df.empty:
        df.sort_values(by=["Adapted From", "Date", "ID"], inplace=True)
        df.to_csv(output_path, index=False)

def generate_stats(data_path: str, stats_path: str) -> None:
    df = pd.read_csv(data_path)
    stats = []
```

```
    categories = [0] + [cat_id for cat_id, _ in CATEGORY_MAP]

    overall_stats = calculate_category_stats(df, 0)
    stats.append([0] + overall_stats)

    for cat_id, _ in CATEGORY_MAP:
        cat_df = df[df["Adapted From"] == cat_id]
        cat_stats = calculate_category_stats(cat_df, cat_id)
        stats.append([cat_id] + cat_stats)

    stats_df = pd.DataFrame(stats, columns=[
        "Adapted From", "Entries", "Mean", "Entry Spread", "Rating Spread"
    ])
    stats_df.to_csv(stats_path, index=False)

def calculate_category_stats(df: pd.DataFrame, category: int) -> List[float]:
    """Calculate statistics for a specific category"""
    if df.empty:
        return [0, 0.0, 0.0, 0.0]

    entries = len(df)
    mean = df["Mean"].mean()
    entry_spread = df["Standard Deviation"].mean()
    rating_spread = df["Mean"].std(ddof=0)

    return [entries, mean, entry_spread, rating_spread]

def random_sample(input_path: str, output_path: str, n: int = 45) -> None:
    df = pd.read_csv(input_path)
    samples = []

    for cat_id, _ in CATEGORY_MAP:
        cat_df = df[df["Adapted From"] == cat_id]
        if len(cat_df) > 0:
            samples.append(cat_df.sample(min(n, len(cat_df)), replace=False))

    if samples:
        pd.concat(samples).to_csv(output_path, index=False)

def chi_gof(data_path: str) -> None:
    sumMean=[0,0,0,0,0]
    sumSD=[0,0,0,0,0]
    ratingsd=[0,0,0,0,0]
    with open(data_path, "r", newline="", encoding="utf-8-sig") as statsfile:
        line=list(statsfile)[1:]
        for i in range(5):
            row=line[i].split(',')
            sumMean[i]=float(row[2])
            sumSD[i]=float(row[3])
            ratingsd[i]=float(row[4])

def anova_test(data_path: str) -> None:
    df = pd.read_csv(data_path)
    categories = [1, 2, 3, 4]
```

```
    print("One-way ANOVA results:")

    # Q1
    groups_mean = [df[df['Adapted From'] == cat]['Mean'] for cat in categories]
    f_stat_mean, p_value_mean = stats.f_oneway(*groups_mean)
    print(f"1. Mean ratings: F({len(categories)-1},{len(df)-len(categories)}) = {f_stat_mean:.4f}, p = 

    # Q2
    groups_sd = [df[df['Adapted From'] == cat]['Standard Deviation'] for cat in categories]
    f_stat_sd, p_value_sd = stats.f_oneway(*groups_sd)
    print(f"2. Entry spread: F({len(categories)-1},{len(df)-len(categories)}) = {f_stat_sd:.4f}, p = {p

    # Q3
    rating_spreads = [group.std(ddof=0) for group in groups_mean]

    levene_stat, levene_p = stats.levene(*groups_mean)
    print(f"\n3. Levene's Test for Rating Spread Equality:")
    print(f"   W-statistic = {levene_stat:.4f}, p-value = {levene_p:.4f}")

    print("\nRating Spread by Category:")
    for i, cat in enumerate(categories):
        print(f"   Category {cat} (n={len(groups_mean[i])}): {rating_spreads[i]:.4f}")

if __name__ == "__main__":
    os.system("rm -rf data/*.csv")
    process_jsonl("data/subject.jsonlines", "data/data.csv")
    generate_stats("data/data.csv", "data/data_stats.csv")
    random_sample("data/data.csv", "data/sample.csv")
    generate_stats("data/sample.csv", "data/sample_stats.csv")
    anova_test("data/sample.csv")


if __name__ == "__main__":
    os.system("rm -rf data/*.csv")
    process_jsonl("data/subject.jsonlines", "data/data.csv")
    generate_stats("data/data.csv", "data/data_stats.csv")
    random_sample("data/data.csv", "data/sample.csv")
    generate_stats("data/sample.csv", "data/sample_stats.csv")
    chi_gof("data/sample_stats.csv")
```

5.2 Appendix B: requirements.txt

```
numpy==2.2.6
pandas==2.2.3
python-dateutil==2.9.0.post0
pytz==2025.2
scipy==1.15.3
six==1.17.0
tzdata==2025.2
```

5.3 Appendix C: readme.md

Bangumi-Data-Analysis

This project analyzes how adaptation sources affect anime ratings on Bangumi.tv. It compares mean ratings, within-anime variability, and between-anime consistency across four adaptation categories: original works, manga, game, and light novel adaptations. For detailed methodology and results, see the Statistical Analysis Report[1].

5.3.1 Setup Instructions

1. Clone repo

```
git clone https://github.com/jiaobenhaimo/Bangumi-Data-Analysis.git
cd Bangumi-Data-Analysis
```

2. Create and activate virtual environment

```
python -m venv venv
source venv/bin/activate  # Linux/Mac
venv\Scripts\activate    # Windows
```

3. Install dependencies

```
pip install -r requirements.txt
```

4. Download data

1. Visit bangumi/Archive[2]
2. Download the latest data dump
3. Extract and move `subject.jsonlines` to `data/` directory

5. Run analysis

```
python main.py
```

5.3.2 File Structure

```
├── data/                   # Data directory
│   ├── subject.jsonlines   # Raw data from bangumi/Archive
│   ├── data.csv            # Processed dataset
│   ├── sample.csv          # Sampled dataset (n=45 per category)
│   ├── data_stats.csv      # Full dataset statistics
│   └── sample_stats.csv    # Sample dataset statistics
├── venv/                   # Virtual environment
```

[1] `report.pdf`
[2] `https://github.com/bangumi/Archive`

```
├── main.py                 # Analysis script
├── report.pdf              # Statistical analysis report
├── report.tex              # LaTeX source of the report
├── requirements.txt        # Dependencies
└── README.MD               # This file
```

5.3.3 Output

The script generates:

1. Processed CSV files in `data/` directory
2. Console output with ANOVA and Levene's test results

This repo is licensed under the BSD 2-Clause License. The data generated from this project follows CC BY-SA license, as required by Bangumi.tv.