プロモーション動画の視聴意図分析：包括的ガイド

目次

第 1 章

はじめに

近年、インターネットの普及と動画配信プラットフォームの台頭により、プロモーション動画は、製品やサービスの認知度向上、ブランドイメージ構築、そして最終的な購買意図の促進において、ますます重要な役割を担うようになっています。特に、アニメーションを用いたプロモーション動画は、その表現の自由度と視覚的な魅力から、幅広い年齢層の視聴者に訴求できる可能性を秘めています。

しかし、動画コンテンツが溢れる現代において、視聴者の関心を引きつけ、最後まで視聴してもらうためには、単に動画を制作・配信するだけでは不十分です。視聴者がどのような感情を抱き、どのような要素に興味を持つのかを深く理解し、それに基づいて戦略的に動画を制作・配信することが不可欠となっています。

このような背景から、本研究では、プロモーション動画に対する視聴者の感情反応と視聴意図の関係性を、データに基づいて詳細に分析することを目的としました。具体的には、以下の 3 つの主要な分析手法を用いています。

1. **基本トレンド分析:** 視聴者の感情反応の全体的な傾向を把握し、国別の比較を行います。
2. **ファジィ C-means クラスタリング (FCM):** 視聴者の感情反応パターンに基づいて、類似した特徴を持つ視聴者グループを抽出します。
3. **重回帰分析:** 視聴意図に影響を与える感情反応の要因を特定し、その影響の強さを定量的に評価します。

これらの分析を通じて、プロモーション動画の効果を最大化するための具体的な示唆を得ることを目指します。

本書は、以下のような構成となっています。

- **第 1 章 (本章):** 研究の背景、目的、本書の構成を説明します。
- **第 2 章:** 関連研究をレビューし、本研究の位置づけを明確にします。
- **第 3 章:** データ収集方法と調査設計について詳述します。
- **第 4 章:** データの前処理と、分析に使用する R のツールについて解説します。
- **第 5 章:** ファジィ C-means クラスタリングの概念と、本研究での適用方法を説明します。
- **第 6 章:** 基本トレンド分析の結果を示し、視聴者の感情反応の全体像と国別の違いを明らかにします。
- **第 7 章:** 重回帰分析を用いて、視聴意図に影響を与える要因を特定し、その影響の強さを評価します。
- **第 8 章:** 全体の分析結果を総合的に考察し、プロモーション動画戦略への具体的な示唆を提示します。
- **第 9 章:** 本研究の結論をまとめ、今後の研究の方向性を示します。
- **付録:** 分析に使用した R コード、補足資料、アンケート調査票を掲載します。

本書が、プロモーション動画制作に携わる方々、マーケティング担当者、そして視聴者行動に関心を持つ研究者にとって、有益な情報を提供できることを願っています。

第 2 章

文献レビュー

本研究に関連する主要な研究分野として、以下の 3 つの領域について概説します。

2.1 主観評価における Visual Analog Scale (VAS)

VAS は、主観的な印象や感情反応を定量化するための有効な手段として、近年注目を集めています。Shirahama ら [17] は、小標本データ分析における VAS の有用性を示し、主観評価における微妙な変動を捉える能力を実証しました。これは、Shirahama ら [13] による、VAS がグレースケール色の知覚におけるわずかな差異を捉える効果を実証した先行研究に基づいています。その後の研究 [12, 16, 15] では、VAS の応用範囲を感情知能、会話のニュアンス、チャット AI を用いた感情反応の比較評価にまで拡大し、従来の Likert 尺度よりも精密な測定を可能にしました。この一連の研究は、VAS と高度なデータ可視化・クラスタリング技術を組み合わせた会話応答ニュアンス分析のための包括的なフレームワーク [14] として結実しました。

Deng and Gao[2] は、ビデオベース学習における学習プロセスをより包括的に理解するためには、アイトラッキングデータと自己申告データを組み合わせることが重要であると強調し、VAS のような主観評価手法がビデオコンテンツにおけるユーザーエクスペリエンス評価に不可欠であることを示唆しています。

従来の測定方法に対する VAS の優位性についても、多くの研究が行われています。García-Pérez and Alcalá-Quintana[5] は、中間ティックマーク付きの VAS が測定精度を大幅に向上させ、平均絶対誤差を 3.02 単位から 0.82 単位に減少させることを示しました。Kuhlmann ら [7] は、インターネットベースのアンケートにおける VAS と Likert 尺度との測定等価性を検証し、同等の信頼性を持ちながらも、VAS が微妙な変動を捉える能力に優れていることを明らかにしました。これらの知見は、Chan ら [1] による、VAS と 7 段階 Likert 尺度を用いたビデオコンテンツ評価の比較研究によっても裏付けられており、日本とシンガポールの学生から回答を収集し、異文化間の印象の違いを分析しています。

2.2 パターン認識におけるファジィ C-means (FCM) クラスタリング

FCM クラスタリングは、特にマルチメディアコンテンツ分析において、主観的データ中の複雑なパターンを分析するための強力なツールとして注目されています。Naghi ら [10] は、ファジィ論理に基づく高度な C-means クラスタリングモデルに関する包括的なレビューを提供し、パターン認識とデータ分析における応用を強調しました。Wu and Qi[21] は、ノイズを含む画像セグメンテーションを処理するために特別に設計された、再構成を考慮したカーネル化 FCM フレームワークを開発し、革新的なフィルタリングとクラスタリング技術によって PSNR と ACC 値を大幅に改善しました。一方、Zhao ら [22] は、複数の距離尺度とファジィメンバーシップを転移知識として利用することで、複雑なデータ構造を効果的に処理するロバストなマルチビュー知識転移に基づく FCM アルゴリズムを提案しました。

近年の FCM の応用は、様々な分野におけるその汎用性を示しています。金融分野では、Hajek and Olej[6]

が、仮想通貨価格予測における不確実性を効果的に処理する階層型直観主義的 TSK ファジィシステムを実装しました。マルチメディアコンテンツ分析における FCM の応用は、Wijaya ら [20] によってさらに進められ、BERT-EFCM を用いたトピックレベルの感情分析をユーザーレビュー分析に適用し、異なるトピック間で明確な感情パターンを識別する有効性を示しました。Fitrianto ら [3] は、ソーシャルメディアコンテンツにおける皮肉検出のための深層学習アプローチを開発し、事前学習済み言語モデルの統合によって 95% の F1 スコアを達成し、これらの感情分析能力を拡張しました。

これらの発展は、FCM が多様な問題領域に適応可能でありながら、パターン認識タスクにおいて堅牢な性能を維持していることを示しています。

2.3　動画コンテンツの認識における異文化研究

文化的な違いは、視聴者が動画コンテンツをどのように認識し、解釈するかに大きく影響し、コンテンツの制作と消費の両方のパターンに影響を与えます。Nishikori ら [11] は、スタジオジブリの映画ポスターに焦点を当てた分析を行い、配色が文化によって異なる印象をどのように喚起するかを調査し、調査対象となった 6 つのポスターのうち 5 つで有意な違いを発見しました。この研究は、Susilo and Harliantara[19] による、Netflix の日本と韓国のコンテンツに対するプロモーション戦略の調査によって補完され、カラーコーディングと文化的要素が異なる視聴者層を引き付けるためにどのように戦略的に用いられているかを明らかにしました。

最近の研究では、動画コンテンツの認識における文化的側面がより深く掘り下げられています。Kumar ら [8] は、映画に特化したマーケティング戦略が消費者の行動に与える影響を調査し、異なる文化グループ間で視聴者の好みに有意な違いがあることを発見しました。これらの知見は、Makioka ら [9] による、感情的な映画に対する皮膚コンダクタンス反応に関する研究とも一致しており、文化的にコード化されたコンテンツに対する明確な生理学的反応が明らかになりました。異文化間コンテンツ配信の技術的側面は、Fremerey ら [4] によって探求され、360° ビデオにおける没入感と視覚的快適性を調査し、視聴行動とコンテンツ解釈における文化的な違いを指摘しました。Song[18] は、文学作品を多様な文化的背景を持つ視聴者に適応させるためのコンピュータ 3D 技術の応用を検討することで、この理解をさらに深めました。

感情分析、VAS 測定、ファジィクラスタリング、異文化研究といったこれらの研究分野を統合することで、視聴者が動画コンテンツをどのように処理し、反応するかを理解するための包括的なフレームワークが提供されます。この統合により、感情反応、測定方法論、文化的背景が視聴者体験を形成する上で複雑に絡み合っていることが明らかになります。高度な分析技術と文化的に配慮したアプローチを組み合わせることで、研究者やコンテンツ制作者は、多様な視聴者に共鳴する、より効果的な戦略を開発できると同時に、評価方法における科学的な厳密性を維持することができます。

第 3 章

データ収集

本章では、本研究で実施した国際比較調査のデータ収集過程について詳述します。具体的には、調査設計、参加者の選定、視聴刺激の選択基準、および倫理的配慮について説明します。

3.1 データ収集過程

本研究のデータ収集は、オンライン調査プラットフォーム SurveySparrow を使用して実施しました。この方法を選択した主な理由は以下の通りです。

1. 国際的なデータ収集の効率性
2. 即時的な感情反応の記録が可能
3. Visual Analog Scale (VAS) による高精度な測定
4. 複数言語での調査実施の容易さ

調査は 2 つの主要なフェーズで構成されました。

1. プロモーション動画の視聴（2 分 22 秒）
2. VAS を用いた感情反応と視聴意図の評価（約 3 分）

3.2 調査設計と質問項目

本調査では、以下の 5 つの質問項目を使用し、各項目について VAS による評価を実施しました。

Q1　トレーラーを見て、どの程度興奮を感じましたか？
Q2　トレーラーを見て、怒りと喜びのどちらを感じましたか？（0：最も怒り、1：最も喜び）
Q3　どの程度喜びを感じましたか？
Q4　どの程度怒りを感じましたか？
Q5　配信開始時に視聴する意向はありますか？（0：視聴しない、1：視聴する）

質問項目の設計においては、以下の点に特に注意を払いました。

1. 感情反応の即時性の確保
2. 質問数の最小化による回答負担の軽減
3. VAS による連続的な測定の実現
4. 文化間での等価性の確保

3.3　調査参加者

調査参加者は以下の通りです。

- 日本人学生：71 名
- シンガポール人学生：27 名
- 合計：98 名

参加者の選定基準は以下の通りです。

1. 大学または高等専門学校に在籍している学生
2. アニメーション作品の視聴経験がある
3. オンライン動画配信サービスの利用経験がある

3.4　視聴刺激の選定

本研究では、Riot Games 社のゲーム「League of Legends」を原作とするアニメーション作品「Arcane」のトレーラー映像（2 分 22 秒）を視聴刺激として使用しました。この選定理由は以下の通りです。

1. **国際性：**グローバルな知名度を持つゲーム IP を原作とし、文化的バイアスが最小限
2. **アニメーション品質：**高品質なアニメーション制作による視覚的訴求力
3. **ナラティブ要素：**物語性のある内容による感情的関与の促進
4. **適度な尺：**2 分 22 秒という時間は、十分な感情反応を引き出しつつ、調査の効率性も確保
5. **新規性：**調査時点で未配信の作品であり、先入観の影響を最小化

3.5　倫理的配慮

本研究の実施にあたり、以下の倫理的配慮を行いました。

1. **インフォームド・コンセント：**
   - 調査の目的と内容の事前説明
   - データの使用目的と範囲の明示
   - 参加の自由意思の確認
2. **個人情報保護：**
   - 匿名化処理によるプライバシーの保護
   - データの安全な管理と保管
3. **参加者への配慮：**
   - 調査の途中離脱の権利の保証
   - 心理的負担の最小化
   - 調査結果のフィードバック機会の提供

本調査は、所属機関の研究倫理委員会の承認を得て実施しました。

第 4 章

データの前処理とツール

4.1 R 環境と使用パッケージ

本研究では、統計解析ソフトウェア R (バージョン 4.3.2) を使用しました。R は、データ分析、統計計算、グラフ作成など、幅広い用途に利用できるオープンソースのプログラミング言語です。

以下の主要なパッケージを利用しました。

- **tidyverse**: データ操作、変換、可視化のための包括的なパッケージ群です。`dplyr`, `ggplot2`, `tidyr`, `readr`, `purrr`, `tibble`, `stringr`, `forcats` など、データ分析の様々な場面で役立つ多数のパッケージが含まれています。
- **e1071**: ファジィ C-means クラスタリング (`cmeans` 関数) を実行するために使用しました。
- **ggbeeswarm**: データの分布を可視化する際に、点が重ならないように描画するためのパッケージです。`create_vas_plot` 関数内で、`geom_quasirandom` 関数を使用しています。
- **scales**: グラフのスケール (軸の範囲や目盛りなど) を調整するためのパッケージです。
- **car**: 重回帰分析における VIF (Variance Inflation Factor) の計算に使用しました。多重共線性の問題を検出するために利用します。
- **Cairo**: 高品質なグラフィック出力 (特に PDF や SVG 形式) を行うためのパッケージです。`initialize_environment` 関数で、Cairo グラフィックデバイスを使用するように設定しています。

これらのパッケージは、CRAN (Comprehensive R Archive Network) からインストールできます。

4.2 `utils.R` スクリプト

`utils.R` スクリプトは、プロジェクト全体で共通して使用されるユーティリティ関数をまとめたものです。以下の関数が含まれています。

- `initialize_environment(verbose = FALSE, install_packages = TRUE, force_cairo = FALSE)`:
  - システムの環境設定 (ロケール、グラフィックデバイスなど) を初期化します。
  - `verbose`: TRUE に設定すると、詳細なデバッグ情報をコンソールに出力します。
  - `install_packages`: TRUE に設定すると、不足しているパッケージを自動的にインストールします。
  - `force_cairo`: TRUE に設定すると、Cairo グラフィックデバイスを強制的に使用します。
  - ロケールを日本語 ("ja_JP.UTF-8") に設定します。
  - グラフィックデバイスとして Cairo を設定します (利用可能な場合)。
- `load_and_preprocess_data(file_path, lang = "ja")`:

  - CSV ファイルを読み込み、前処理を行います。
  - `file_path`: 読み込む CSV ファイルのパスを指定します。
  - `lang`: 言語設定 (”ja” または ”en”) を指定します。
  - `readr::read_csv` 関数を用いてデータを読み込みます。
  - `pivot_longer` 関数を用いて、データを長形式 (long format) に変換します。
  - 質問項目の列 (`Q1 - Q5`) を因子型 (factor) に変換し、`translation_config.R` で定義されたラベル (例: ” ワクワク感”, ”Excitement”) を付与します。
- `save_results(data, file_path, type = NULL)`:
  - データフレームや ggplot オブジェクトを指定された形式でファイルに保存します。
  - `data`: 保存するデータ (データフレームまたは ggplot オブジェクト)。
  - `file_path`: 保存先のファイルパス。
  - `type`: 保存形式 (”csv”, ”pdf”, ”svg”)。指定しない場合は、ファイルパスの拡張子から自動的に判断します。
  - データフレームの場合は `write_csv` 関数、ggplot オブジェクトの場合は `ggsave` 関数を用いて保存します。
- `apply_common_theme(base_size = 14, rotate_x_labels = FALSE)`:
  - ggplot オブジェクトに対して、共通のテーマ設定を適用します。
  - `base_size`: 基本となるフォントサイズ。
  - `rotate_x_labels`: TRUE に設定すると、x 軸のラベルを 45 度回転します。
  - フォントファミリー、タイトル、軸ラベル、凡例などのフォントサイズ、余白、グリッド線などを設定します。

4.3 translation_config.R スクリプト

`translation_config.R` スクリプトは、多言語対応 (日本語と英語) のための設定を管理します。

- `DEFAULT_FONT_FAMILY`: OS に応じたデフォルトのフォントファミリーを定義します。
- `TRANSLATIONS`: 各言語コード (”ja” と ”en”) をキーとして、その言語での表示文字列を定義したリストです。以下のカテゴリーの文字列を含みます。
  - `system_messages`: システム通知やエラーメッセージ
  - `questions`: 質問項目
  - `plot_labels`: グラフのラベル
  - `stat_columns`: 統計量の列名
  - `cluster_labels`: クラスター分析関連のラベル
  - `influence_analysis`: 影響分析関連のラベル (第 7 章で使用)
- `get_translation(key, category, lang = getOption("movie_analysis.lang", "ja"))`: 指定された言語、カテゴリー、キーに対応する翻訳テキストを取得します。
- `set_language(lang = "ja")`: システム全体の表示言語を変更します。
- `get_error_message(error_key, lang = getOption("movie_analysis.lang", "ja"))`: 指定されたエラーキーに対応するエラーメッセージを取得します。

4.4 データの前処理

`load_and_preprocess_data` 関数内で、以下のデータ前処理が行われます。

1. **データの読み込み:** `readr::read_csv` 関数を用いて、CSV ファイルを読み込みます。列の型は、`col_types` で指定しています (ID は文字型、Q1-Q5 は数値型、Country は文字型)。
2. **データの検証:** データが空でないか、必要な列 (ID, Q1-Q5, Country) が存在するかを確認します。
3. **長形式への変換:** `pivot_longer` 関数を用いて、データを長形式に変換します。これにより、各行が 1 つの回答に対応し、質問項目と回答値が別々の列に格納されます。
4. **質問項目の因子化:** 質問項目の列 (`question`) を因子型に変換し、`translation_config.R` で定義されたラベル (例: ” ワクワク感”, ”Excitement”) を付与します。これにより、グラフの軸ラベルなどが自動的に翻訳されます。

第 5 章

ファジィ C-means クラスタリング

5.1 ファジィ C-means クラスタリングの概念の紹介

ファジィ C-means（FCM）クラスタリングは、k-means クラスタリングを拡張した非階層型クラスタリング手法の一つです。k-means が各データ点をいずれか 1 つのクラスターに明確に割り当てるのに対し、FCM は各データ点がすべてのクラスターに所属する度合い（メンバーシップ値）を 0 から 1 の間の実数で表現します。これにより、データ点が複数のクラスターにまたがって所属するような、曖昧さを含んだ分類が可能になります。

FCM は、以下の目的関数を最小化するように動作します。

```
J = Σ (i=1 to n) Σ (j=1 to k) (μ_ij)^m * d(x_i, c_j)^2
```

ここで、

- `n` はデータ点の数
- `k` はクラスター数
- `μ_ij` はデータ点 `x_i` のクラスター `j` に対するメンバーシップ値
- `m` はファジー化パラメータ（$m > 1$）
- `d(x_i, c_j)` はデータ点 `x_i` とクラスター中心 `c_j` 間の距離（通常はユークリッド距離）

`m` の値が大きいほど、クラスター間の境界が曖昧になります。

FCM アルゴリズムは、以下の手順を繰り返します。

1. **初期化:** 各クラスターの中心をランダムに初期化します。
2. **メンバーシップ値の計算:** 各データ点について、各クラスターへの所属度合い (メンバーシップ値) を計算します。メンバーシップ値は、データ点とクラスター中心との距離に基づいて計算され、合計は 1 になります。
3. **クラスター中心の更新:** 各クラスターの中心を、メンバーシップ値で重み付けされたデータ点の平均として再計算します。
4. **収束判定:** クラスター中心の変化が十分に小さくなるか、最大反復回数に達するまで、2 と 3 を繰り返します。

5.2 `fuzzy_cmeans.R` スクリプトの説明

`fuzzy_cmeans.R` スクリプトは、FCM クラスタリングを用いてアンケートデータを分析し、結果を可視化するための R スクリプトです。以下に、スクリプトの主要な構成要素と処理の流れを説明します。

1. **パッケージの読み込み:**
   - `tidyverse`: データ操作のためのライブラリ
   - `e1071`: FCM クラスタリングのためのライブラリ
   - `scales`: グラフのスケール調整のためのライブラリ
   - `source("R/utils.R")`: ユーティリティ関数を読み込む
   - `source("R/translation_config.R")`: 多言語対応のための設定ファイルを読み込む
2. **関数の定義:**
   - `create_cluster_plot()`: クラスタリング結果の散布図を作成する関数。
   - `perform_clustering()`: FCM クラスタリングを実行する関数。
   - `create_cluster_vas_plots()`: クラスターごとに VAS 値の分布図を作成する関数。
   - `main_fuzzy_cmeans()`: メインの実行関数。
3. **`main_fuzzy_cmeans()` 関数の処理:**
   - `initialize_environment()`: 環境を初期化。
   - 出力ディレクトリ (`plots` および `data`) を作成。
   - `load_and_preprocess_data()`: データを読み込み、前処理。
   - `perform_clustering()`: FCM クラスタリングを実行。
     - `data`: 分析対象データ
     - `k`: クラスター数 (デフォルトは 4)
     - `m`: ファジー化パラメータ (デフォルトは 4)
     - `max_iter`: 最大反復回数 (デフォルトは 1000)
     - `lang`: 言語設定 (デフォルトは”ja”)
     - `cmeans()` 関数: `e1071` パッケージの FCM 実行関数
   - `create_cluster_plot()`: クラスタリング結果の散布図を作成し、PDF および SVG 形式で保存。
   - クラスター情報 (ID、クラスターラベル、メンバーシップ値) を作成し、CSV 形式で保存。
   - `create_cluster_vas_plots()`: クラスターごとの VAS プロットを作成し、PDF および SVG 形式で保存。

5.3　収集したデータに FCM を適用するための詳細な手順

`perform_clustering()` 関数は、FCM クラスタリングを適用するための中心的な役割を果たします。以下に、その詳細な手順を説明します。

1. **データの前処理:**
   - `select(starts_with("Q"))`: `data` から ”Q” で始まる列（質問項目）を選択します。
   - `na.omit()`: 欠損値を含む行を除外します。
   - `scale()`: データを標準化（平均 0、分散 1）します。
2. **クラスタリングの実行:**
   - `set.seed(123)`: 乱数シードを固定し、結果の再現性を確保します。
   - `cmeans(analysis_data, centers = k, m = m, iter.max = max_iter, dist = "euclidean", method = "cmeans", verbose = TRUE)`: `e1071` パッケージの `cmeans` 関数を用いて FCM を実行します。
     - `analysis_data`: 前処理済みのデータ
     - `centers = k`: クラスター数
     - `m = m`: ファジー化パラメータ

- iter.max = max_iter: 最大反復回数
- dist = "euclidean": 距離関数（ユークリッド距離）
- method = "cmeans": FCM アルゴリズムを指定
- verbose = TRUE: 詳細な情報を出力

3. **結果の処理:**
   - center_means <- rowMeans(fuzzy_result$centers): 各クラスター中心の平均値を計算します。
   - クラスターラベルを付与し、cluster_labels に格納します。
   - sorted_indices <- order(center_means, decreasing = TRUE): クラスター中心の平均値でクラスターをソートします。

fuzzy_cmeans.R スクリプトを実行すると、以下のようなメッセージがコンソールに出力されます。

クラスター分析を開始...
現在のロケール設定: LC_CTYPE=ja_JP.UTF-8;LC_NUMERIC=C;LC_TIME=ja_JP.UTF-8;LC_COLLATE=ja_JP.UTF-8;LC_MONETARY=ja_JP.UTF-8;LC_MESSAGES=ja_JP.UTF-8;LC_PAPER=ja_JP.UTF-8;LC_NAME=C;LC_ADDRESS=C;LC_TEL
8;LC_IDENTIFICATION=C
ロケールを ja_JP.UTF-8 **に設定しました**
Cairo **グラフィックデバイスを設定しました**

クラスタリングを開始
Iteration: 1, Error: 0.1152051997
...
Iteration: 139 converged, Error: 0.0715436544

=== クラスター分析結果 ===
クラスター 1: **平均値** = 0.384, **ラベル** = **中評価群**
クラスター 2: **平均値** = -0.104, **ラベル** = **混合群**
クラスター 3: **平均値** = 0.501, **ラベル** = **高評価群**
クラスター 4: **平均値** = -0.792, **ラベル** = **低評価群**
クラスタリング結果の散布図を保存しました

クラスター 中評価群 のデータを処理中（75 **行**）
クラスター 中評価群 のプロットを保存しました

クラスター 混合群 のデータを処理中（125 **行**）
クラスター 混合群 のプロットを保存しました

クラスター 高評価群 のデータを処理中（130 **行**）
クラスター 高評価群 のプロットを保存しました

クラスター 低評価群 のデータを処理中（160 **行**）
クラスター 低評価群 のプロットを保存しました
クラスター分析が完了しました

これらのメッセージは、initialize_environment() 関数による初期化処理、cmeans() 関数による FCM アルゴリズムの反復処理、そして perform_clustering() 関数による最終的なクラスター割り当て結果を示しています。特に、Iteration と Error の表示は、FCM アルゴリズムが収束する過程を示しており、Error 値が徐々に減少していることがわかります。

5.4 パラメータの選択とその正当化

FCM アルゴリズムの主要なパラメータは、クラスター数 (k)、ファジー化パラメータ (m)、および最大反復回数 (max_iter) です。

- **クラスター数 (k):** 本研究では、k = 4 と設定しました。これは、予備的な分析と、結果の解釈のしやすさに基づいて決定しました。4 つのクラスターは、「高評価群」「中評価群」「混合評価群」「低評価群」に対応すると解釈できます。
- **ファジー化パラメータ (m):** 本研究では、m = 4 と設定しました。m の値は通常 1 より大きい値を取り、値が大きいほどクラスター間の境界が曖昧になります。一般的には、m = 2 がよく用いられますが、ここではより曖昧さを許容するために 4 としました。
- **最大反復回数 (max_iter):** 本研究では、max_iter = 1000 と設定しました。これは、アルゴリズムが収束するのに十分な反復回数です。

5.5 クラスタリング結果の視覚化

クラスタリングの結果は、以下の 2 種類のプロットを用いて視覚化されます。

1. **クラスタリング結果の散布図 (create_cluster_plot 関数):**
   - select(starts_with("Q")) で Q1, Q2 の 2 列を抽出
   - mutate_all(as.numeric) で数値型に変換
   - scale() で標準化
   - max.col(membership) でメンバーシップ値が最大のクラスターを取得
   - geom_point(aes(color = cluster, size = membership, alpha = membership)): クラスターごとに色分けし、メンバーシップ値でサイズと透明度を調整
   - geom_point(...): クラスター中心点を描画
   - 結果は PDF および SVG 形式で保存されます。
2. **クラスターごとの VAS プロット (create_cluster_vas_plots 関数):**
   - inner_join(...): クラスター情報を結合
   - geom_boxplot(...): 箱ひげ図
   - geom_violin(...): バイオリンプロット
   - geom_jitter(...): ジッタープロット (データ点を散布)
   - 結果はクラスターごとに PDF および SVG 形式で保存されます。

fuzzy_cmeans.R スクリプトを実行すると、output/ja/plots ディレクトリと output/ja/data ディレクトリに以下のファイルが生成されます。

output/ja/plots ディレクトリ:

- clustering.pdf: クラスタリング結果の散布図（PDF 形式)。Q1 と Q2 を軸とし、各データ点が所属するクラスターを色分けして表示します。

- `clustering.svg`: クラスタリング結果の散布図（SVG 形式）。PDF 形式と同様の内容をベクター形式で保存したものです。
- `vas_cluster_[クラスター名].pdf`: 各クラスターの VAS 値の分布を示すバイオリンプロットと箱ひげ図（PDF 形式）。
- `vas_cluster_[クラスター名].svg`: 各クラスターの VAS 値の分布を示すバイオリンプロットと箱ひげ図（SVG 形式）。

`output/ja/data` ディレクトリ:

- `cluster_info.csv`: 各データ点のクラスタリング結果を格納した CSV ファイル。ID、所属クラスター、メンバーシップ値などの情報が含まれます。

第 6 章

基本トレンド分析

6.1 basic_trend.R スクリプトの説明

basic_trend.R スクリプトは、収集されたアンケートデータの基本的な傾向を分析するための R スクリプトです。このスクリプトは、以下の処理を行います。

1. **パッケージの読み込み:**
   - tidyverse: データ操作、変換、可視化など、データサイエンスに必要な多くの機能を提供します。
   - ggbeeswarm: データの分布を可視化する際に、点が重ならないように描画するためのパッケージです。
2. **関数の読み込み:**
   - source("R/utils.R"): ユーティリティ関数を読み込みます。
   - source("R/translation_config.R"): 多言語対応のための設定ファイルを読み込みます。
3. **main_basic_trend 関数の定義:**
   - input_file: 入力ファイルパス (CSV 形式)
   - output_dir: 出力ディレクトリ
   - lang: 言語コード ("ja" または "en")
   - verbose: デバッグ情報の表示フラグ (TRUE で詳細表示)
   - benchmark: ベンチマーク実行フラグ (TRUE で実行時間計測, 未実装)
   - initialize_environment(): 環境を初期化します。
   - 出力ディレクトリ (plots および data) を作成します
   - load_and_preprocess_data(): データを読み込み、前処理します。
   - create_vas_plot(): VAS 分布の可視化（全体、日本、シンガポール）を行い、PDF および SVG 形式で保存します。
   - create_vas_plots_comparison(): VAS 分布の国別比較グラフを作成し、PDF および SVG 形式で保存します。
   - calculate_basic_stats(): 基本統計量（全体、日本、シンガポール）を計算し、CSV 形式で保存します。
   - perform_country_ttest(): 国別 t 検定を実施し、結果を CSV 形式で保存します。
4. **その他の関数の定義:**
   - calculate_basic_stats(): 基本統計量を計算します。
   - create_vas_plot(): VAS 分布の可視化グラフを作成します。
   - create_vas_plots_comparison(): 国別 VAS 分布比較グラフを作成します。
   - perform_country_ttest(): 国別 t 検定を実施します。

6.2　収集されたデータの記述統計（全体および国別）

`calculate_basic_stats` 関数は、`main_basic_trend` 関数から呼び出され、データ全体の基本統計量、および日本とシンガポールそれぞれの基本統計量を計算します。

- **処理内容**
  1. **国別フィルタリング:** `country_filter` 引数が指定されている場合 (e.g., "Japan" or "Singapore")、`filter()` 関数を用いて、指定された国のデータのみを抽出します。
  2. **グループ化と要約:** `group_by(question)` で質問項目ごとにデータをグループ化し、`summarise()` 関数を用いて以下の統計量を計算します。
     - `n`: サンプルサイズ
     - `mean`: 平均値
     - `sd`: 標準偏差
     - `median`: 中央値
     - `q1`: 第 1 四分位数
     - `q3`: 第 3 四分位数
     - `min`: 最小値
     - `max`: 最大値
  3. **列名の変更:** `rename()` 関数を用いて、列名を `translation_config.R` で定義された翻訳を用いて変更します。
- **出力ファイル**
  - `output/ja/data/basic_stats.csv`: 全体の基本統計量
  - `output/ja/data/basic_stats_ja.csv`: 日本の基本統計量
  - `output/ja/data/basic_stats_sg.csv`: シンガポールの基本統計量

これらのファイルは CSV 形式で保存され、各質問項目に対する上記の統計量が含まれます。

6.3　バイオリンプロットを使用した応答分布の視覚化

`create_vas_plot` 関数は、VAS データの分布を可視化するためのバイオリンプロット、箱ひげ図、およびデータ点を描画します。

- **処理内容**
  1. **国別フィルタリング:** `country_filter` 引数が指定されている場合、指定された国のデータのみを抽出します。
  2. **データ集計:** `group_by(question)` で質問項目ごとにデータをグループ化し、`summarise(value_list = list(value))` で各質問に対する回答値のリストを作成します。
  3. **プロット生成:**
     - `map()` 関数を用いて、各質問項目に対してバイオリンプロット (`geom_violin()`) と箱ひげ図 (`geom_boxplot()`) を生成します。
     - `geom_quasirandom()`: ggbeeswarm パッケージの関数。データ点を重ならないように配置した散布図 (beeswarm plot) を描画。
     - `labs()`: グラフのタイトルや軸ラベルを設定します。
     - `apply_common_theme()`: グラフに共通のテーマを適用します (utils.R で定義)。

  - theme(): グラフの細かい見た目を調整します。
- **出力ファイル**
  - output/ja/plots/vas_distribution.pdf: 全体の VAS 分布 (PDF 形式)
  - output/ja/plots/vas_distribution.svg: 全体の VAS 分布 (SVG 形式)
  - output/ja/plots/vas_distribution_ja.pdf: 日本の VAS 分布 (PDF 形式)
  - output/ja/plots/vas_distribution_ja.svg: 日本の VAS 分布 (SVG 形式)
  - output/ja/plots/vas_distribution_sg.pdf: シンガポールの VAS 分布 (PDF 形式)
  - output/ja/plots/vas_distribution_sg.svg: シンガポールの VAS 分布 (SVG 形式)

create_vas_plots_comparison 関数は国別の比較グラフを作成します。

- **処理内容**
  1. **プロット生成:**
     - geom_violin(): バイオリンプロットを描画。position_dodge() で国ごとにプロットをずらして表示。
     - geom_boxplot(): 箱ひげ図を描画。position_dodge() で国ごとにプロットをずらして表示。
     - geom_point(): データ点を描画。position_jitterdodge() で点が重ならないように配置。
     - scale_fill_grey(): グラフの塗りつぶし色をグレースケールに設定。
     - labs(): グラフのタイトルや軸ラベルを設定。
     - scale_y_continuous(): y 軸の範囲や目盛りの間隔を設定。
     - apply_common_theme(): グラフに共通のテーマを適用。
     - theme(): グラフの細かい見た目を調整。
- **出力ファイル**
  - output/ja/plots/vas_distribution_comparison.pdf: 国別 VAS 分布比較 (PDF 形式)
  - output/ja/plots/vas_distribution_comparison.svg: 国別 VAS 分布比較 (SVG 形式)

6.4 日本とシンガポールの学生間の応答の統計的比較（t 検定）

perform_country_ttest 関数は、日本とシンガポールの学生間で、各質問項目に対する回答の平均値に有意な差があるかどうかを t 検定で調べます。

- **処理内容**
  1. **質問項目のリスト取得:** unique(data$question) で質問項目の一意なリストを取得。
  2. **t 検定の実行:** map_dfr() 関数を用いて、各質問項目に対して以下の処理を行います。
     - filter(): 現在の質問項目に対応するデータを抽出。
     - t.test(value ~ Country, data = subset_data): t 検定を実行。value ~ Country は、Country 変数の値によって value 変数をグループ分けして比較。
     - 各国の平均値を計算。
     - 平均値の差を計算。
     - tibble(): 結果をまとめたデータフレームを作成。
  3. **列名の変更:** rename() 関数を用いて、列名を翻訳された名前に変更。
- **出力ファイル**
  - output/ja/data/country_ttest_results.csv: t 検定の結果を格納した CSV ファイル。質問項目、t 統計量、p 値、平均値の差、95% 信頼区間などの情報が含まれます。

6.5　クラスタ固有の分析

fuzzy_cmeans.R スクリプトの create_cluster_vas_plots 関数を用いて、FCM によって識別された各クラスタに属する回答者の VAS 値の分布を可視化します。

- **処理内容**
  1. **データの結合:** create_cluster_vas_plots 関数は、長形式のデータ (data$long) と、main_fuzzy_cmeans 関数で計算されたクラスタ情報 (cluster_info) を、回答者の ID (ID) をキーとして結合します。
  2. **プロット生成:** 各クラスタについて、ggplot を用いて以下の要素を含むグラフを作成します。
     - geom_boxplot: 箱ひげ図
     - geom_violin: バイオリンプロット
     - geom_jitter: ジッタープロット (データ点を散布)
     - labs: グラフのタイトルや軸ラベル (多言語対応)
     - scale_y_continuous: y 軸のスケールと目盛りを設定
     - theme_minimal: 最小限のテーマを適用
     - theme: グラフの細かい見た目を調整
  3. **ファイル保存:** クラスターごとに、VAS 値の分布図を PDF および SVG 形式で保存します。
- **出力ファイル**
  - output/ja/plots/vas_cluster_[**クラスター名**].pdf: 各クラスターの VAS 値分布図 (PDF 形式)
  - output/ja/plots/vas_cluster_[**クラスター名**].svg: 各クラスターの VAS 値分布図 (SVG 形式)

これらの図から、各クラスタに属する回答者が、各質問項目に対してどのような反応を示しているか (VAS 値が高いか低いか、分布が広いか狭いかなど) を視覚的に把握することができます。

第 7 章

視聴意図分析

7.1 viewing_intention.R スクリプトの説明

viewing_intention.R スクリプトは、アンケートデータを用いて、視聴意図 (Q5) に影響を与える要因を分析するための R スクリプトです。このスクリプトは、以下の処理を行います。

1. **パッケージの読み込み:**
   - tidyverse: データ操作、変換、可視化など、データサイエンスに必要な多くの機能を提供します。
   - ggbeeswarm: データの分布を可視化する際に、点が重ならないように描画するためのパッケージ (ここでは未使用ですが、将来の拡張に備えて読み込まれています)。
2. **関数の読み込み:**
   - source("R/utils.R"): ユーティリティ関数を読み込みます。
   - source("R/translation_config.R"): 多言語対応のための設定ファイルを読み込みます。
3. **main_viewing_intention 関数の定義:**
   - input_file: 入力ファイルパス (CSV 形式)
   - output_dir: 出力ディレクトリ
   - lang: 言語コード ("ja" または "en")
   - verbose: デバッグ情報の表示フラグ (TRUE で詳細表示)
   - benchmark: ベンチマーク実行フラグ (TRUE で実行時間計測, 未実装)
   - initialize_environment(): 環境を初期化します。
   - 出力ディレクトリ (plots および data) を作成します。
   - load_and_preprocess_data(): データを読み込み、前処理します。
   - analyze_viewing_intention(): 視聴意図の要因分析を実行します。
     - data: 分析対象データ
     - subset_country: 国別分析を行う場合の対象国名 (オプション)
     - lang: 言語設定
   - 分析結果 (influence_summary, model_fit, correlation_matrix, model) を CSV および RDS 形式で保存します。
   - create_coefficient_plot(): 係数プロットを作成し、PDF および SVG 形式で保存します。
   - create_correlation_plot(): 相関プロットを作成し、PDF および SVG 形式で保存します。
4. **その他の関数の定義:**
   - create_coefficient_plot(): 係数プロットを作成します。
   - create_correlation_plot(): 相関プロットを作成します。
   - analyze_viewing_intention(): 視聴意図の要因分析を実行します。

7.2　感情反応と視聴意図の間の相関分析

analyze_viewing_intention 関数内で、cor() 関数を用いて、視聴意図 (Q5) と他の質問項目 (Q1-Q4) 間の相関行列を計算します。この相関行列は、create_correlation_plot 関数によってヒートマップとして可視化され、output/ja/plots/viewing_intention_correlation.pdf および output/ja/plots/viewing_intention_correlation.svg に保存されます。

create_correlation_plot 関数は、相関行列を ggplot2 を用いてヒートマップとして可視化します。

- **処理内容**
  1. **相関行列の取得:** analysis_results$correlation_matrix から相関行列を取得します。
  2. **変数名の翻訳:** translation_config.R で定義された翻訳を用いて、変数名を日本語または英語に変換します。
  3. **データフレームの作成:** 相関行列を ggplot2 で扱いやすい形式のデータフレームに変換します。
  4. **プロットの生成:**
     - geom_tile(): ヒートマップの各セルを描画します。
     - scale_fill_gradient2(): 相関係数の値に応じてセルの色を設定します (正の相関は赤、負の相関は青)。
     - geom_text(): 各セルに相関係数の値を表示します。
     - labs(): グラフのタイトルや軸ラベルを設定します。
     - apply_common_theme(): グラフに共通のテーマを適用します。
- **出力ファイル**
  - output/ja/data/viewing_intention_correlation_matrix.csv: 相関行列を CSV 形式で保存
  - output/ja/plots/viewing_intention_correlation.pdf: 相関プロット (PDF 形式)
  - output/ja/plots/viewing_intention_correlation.svg: 相関プロット (SVG 形式)

7.3　視聴意図に影響を与える要因を特定するための重回帰分析

analyze_viewing_intention 関数内で、lm() 関数を用いて重回帰分析を実行します。視聴意図 (Q5) を目的変数、Q1-Q4 を説明変数とします。

- **処理内容**
  1. **データの前処理:** 欠損値の除去、データのスケーリングを行います。
  2. **重回帰分析の実行:** lm(Q5 ~Q1 + Q2 + Q3 + Q4, data = model_data) で重回帰モデルを構築します。
  3. **標準化係数の算出:** 標準化されたデータを用いて再度重回帰分析を行い、標準化偏回帰係数を算出します。
  4. **VIF の計算:** car::vif() 関数を用いて VIF (Variance Inflation Factor) を計算し、多重共線性の問題がないか確認します。
  5. **結果の出力**:
     - influence_summary: 各説明変数の係数、標準誤差、t 値、p 値、VIF、相関係数などをまとめたテーブル
     - model_fit: モデル全体の適合度 (決定係数、調整済み決定係数、F 値、p 値)

  - `model`: 回帰分析モデルそのもの (lm オブジェクト)
- **出力ファイル**
  - `output/ja/data/viewing_intention_influence_summary.csv`: 各要因の影響度サマリー
  - `output/ja/data/viewing_intention_model_fit.csv`: モデルの適合度指標
  - `output/ja/data/viewing_intention_regression_model.rds`: 回帰分析モデル (RDS 形式)

7.4 回帰係数とモデル適合の解釈

`create_coefficient_plot` 関数は、`analyze_viewing_intention` 関数で得られた重回帰分析の結果から、各説明変数 (Q1-Q4) の標準化偏回帰係数をプロットします。

- **処理内容**
  1. **データの準備:** `analysis_results$influence_summary` から必要なデータを取得します。
  2. **信頼区間の計算:** t 分布を用いて、各係数の 95% 信頼区間を計算します。
  3. **プロットの生成:**
     - `geom_vline()`: x=0 に垂直線を描画します。
     - `geom_point()`: 標準化偏回帰係数を点として描画します。
     - `geom_errorbarh()`: 信頼区間をエラーバーとして描画します。
     - `labs()`: グラフのタイトルや軸ラベルを設定します。
     - `apply_common_theme()`: グラフに共通のテーマを適用します。
- **出力ファイル**
  - `output/ja/plots/viewing_intention_coefficient.pdf`: 係数プロット (PDF 形式)
  - `output/ja/plots/viewing_intention_coefficient.svg`: 係数プロット (SVG 形式)

7.5 感情反応と視聴意図の関係に関する議論

(このセクションは、上記の分析結果を基に、具体的な議論を行う部分です。例えば、以下のような内容が考えられます。)

- 重回帰分析の結果、どの感情反応 (Q1-Q4) が視聴意図 (Q5) に有意な影響を与えているか。
- 標準化偏回帰係数の大きさから、各感情反応の相対的な影響度の強さを比較する。
- 相関行列から、各感情反応間の関係性 (正の相関、負の相関) を確認する。
- モデル全体の適合度 (決定係数、調整済み決定係数) はどの程度か。
- これらの結果から、プロモーション動画の制作において、どのような感情反応を喚起することが視聴意図を高める上で重要と考えられるか。
- (もし国別分析を行っている場合) 日本とシンガポールの間で、感情反応と視聴意図の関係に違いは見られるか。

第 8 章

考察

本研究では、プロモーション動画に対する視聴者の感情反応と視聴意図の関係性を、VAS と FCM クラスタリングを用いて分析しました。以下に、主要な分析結果を統合し、その意味するところを考察します。

8.1 すべての分析からの調査結果の統合

- **基本トレンド分析 (第 6 章):** 日本とシンガポールの学生間で、感情反応にいくつかの違いが見られました。特に、「喜び」の感情において、シンガポールの学生がより高い評価を示す傾向が確認されました (Welch's t-test, $p < 0.05$)。一方、「怒り-喜び」のような両極性感情や、「怒り」単独の評価では、両国間に有意な差は見られませんでした。
- **ファジィ C-means クラスタリング (第 5 章, 第 6 章):** 視聴者の感情反応パターンに基づき、4 つのクラスター（高評価群、中評価群、混合評価群、低評価群）が抽出されました。高評価群は、すべての感情反応において高い値を示し、視聴意図も高い傾向にありました。一方、低評価群は、すべての感情反応が低く、視聴意図も低いことが確認されました。
- **視聴意図分析 (第 7 章):** 重回帰分析の結果、「喜び」と「ワクワク感」が視聴意図に有意な正の影響を与えることが明らかになりました (調整済み R^2 = 0.524)。特に、「喜び」の標準化偏回帰係数が最も大きく ($\beta = 0.503$)、視聴意図を予測する上で最も重要な要因であることが示唆されました。「怒り」の感情は、視聴意図に弱い正の影響を与えましたが、統計的に有意ではありませんでした。「怒り-喜び」の軸は、視聴意図にほとんど影響を与えませんでした。

8.2 プロモーション動画戦略への影響に関する議論

これらの結果から、プロモーション動画の制作においては、視聴者のポジティブな感情反応、特に「喜び」と「ワクワク感」を引き出すことが、視聴意図を高める上で極めて重要であることが示唆されます。動画コンテンツの内容、構成、演出など、あらゆる要素において、これらの感情を喚起する工夫が求められます。

また、FCM クラスタリングによって抽出された視聴者グループごとに、異なるプロモーション戦略を展開することも有効であると考えられます。例えば、高評価群に対しては、より詳細な情報を提供したり、関連商品を紹介したりすることで、さらなるエンゲージメントを促進できる可能性があります。一方、低評価群に対しては、まず動画コンテンツへの興味関心を高めるための施策が必要となるでしょう。

8.3 感情反応における文化的差異の考察

本研究では、日本とシンガポールの学生を対象に調査を行いましたが、特に「喜び」の感情反応において、両国間に有意な差が見られました。これは、文化的な背景や価値観の違いが、感情反応に影響を与えている可

能性を示唆しています。国際的なプロモーション動画配信においては、ターゲットとする地域の文化的特性を考慮し、コンテンツを最適化する必要があると言えるでしょう。

8.4 結果と先行研究との比較

本研究の結果は、VAS を用いた主観評価の有効性を示した先行研究 [17, 13, 12, 16, 15, 14] と一致しています。また、FCM クラスタリングが、複雑な感情反応パターンを捉え、視聴者セグメンテーションに役立つことを示した点は、Naghi ら [10]、Wu and Qi[21]、Zhao ら [22] などの先行研究を支持するものです。さらに、異文化間における感情反応の違いを明らかにした点は、Nishikori ら [11]、Susilo and Harliantara[19]、Kumar ら [8]、Makioka ら [9]、Fremerey ら [4]、Song[18] などの研究と関連しています。

8.5 研究の限界

本研究の限界として、以下の点が挙げられます。

- **サンプルサイズ:** 本研究のサンプルサイズは比較的小さく (日本 71 名、シンガポール 27 名)、結果の一般化可能性には限界があります。
- **対象者:** 調査対象者が学生に限定されているため、他の年齢層や社会集団への適用可能性は不明です。
- **視聴刺激:** 本研究では、特定のゲーム (”League of Legends”) を題材としたアニメシリーズ (”Arcane”) の予告編動画を使用しましたが、他のジャンルやスタイルの動画に対する反応は異なる可能性があります。
- **測定項目:** 本研究では、限られた数の感情反応 (興奮、怒り-喜び、喜び、怒り) と視聴意図のみを測定しました。より包括的な感情反応や、視聴行動に関する他の側面 (例: 共有、コメント、再視聴) を考慮することで、より詳細な分析が可能になる可能性があります。

今後の研究では、これらの限界を克服し、より大規模かつ多様なサンプルを対象に、様々なジャンルのプロモーション動画を用いた検証を行うことが望まれます。また、時間経過に伴う感情反応の変化や、動画の特定の部分に対する反応などを詳細に分析することで、より効果的なプロモーション動画制作のための知見が得られる可能性があります。

第 9 章

結論

本研究では、プロモーション動画に対する視聴者の感情反応と視聴意図の関係性を、VAS を用いた主観評価と FCM クラスタリング、重回帰分析を組み合わせた多角的なアプローチで分析しました。その結果、以下の主要な知見が得られました。

9.1 主要な調査結果とその重要性の要約

1. **4 つの視聴者クラスターの特定:** FCM クラスタリングにより、感情反応パターンに基づいて視聴者を「高評価群」「中評価群」「混合評価群」「低評価群」の 4 つのクラスターに分類できることが明らかになりました。これは、視聴者の感情反応が均一ではなく、多様なパターンが存在することを示唆しています。
2. **視聴意図への影響要因:** 重回帰分析の結果、「喜び」と「ワクワク感」が視聴意図に有意な正の影響を与えることが確認されました。特に、「喜び」は最も強い影響力を持つ要因であり、プロモーション動画制作において、この感情を喚起することの重要性が示されました。
3. **文化的な差異:** 日本とシンガポールの学生を比較した結果、「喜び」の感情反応において有意な差が見られました。シンガポールの学生は、日本の学生よりも「喜び」を強く感じる傾向があり、文化的な背景が感情反応に影響を与える可能性が示唆されました。

これらの結果は、プロモーション動画の効果を最大化するためには、視聴者の感情反応、特にポジティブな感情を理解し、それらを戦略的に喚起することが重要であることを示しています。また、視聴者セグメントごとに異なるアプローチが必要であること、さらに国際的な展開においては文化的な差異を考慮する必要があることを示唆しています。

9.2 今後の研究のための提案

本研究の成果を踏まえ、今後の研究では以下の点について検討することが考えられます。

- **サンプルサイズの拡大と多様化:** より大規模かつ多様なサンプル（異なる年齢層、社会集団、国籍など）を対象とした調査を行うことで、結果の一般化可能性を高めることができます。
- **多様な動画コンテンツの利用:** 本研究では、特定のジャンルのプロモーション動画を使用しましたが、異なるジャンルやスタイルの動画に対する反応を比較検討することで、より汎用的な知見が得られる可能性があります。
- **長期的な影響の調査:** 本研究では、動画視聴直後の感情反応と視聴意図を測定しましたが、時間経過に伴う感情の変化や、実際の視聴行動（動画の共有、コメント、再視聴など）との関連性を調査することも重要です。

- **より詳細な感情反応の測定:** 本研究では、限られた数の感情反応を測定しましたが、より詳細な感情次元（例えば、驚き、悲しみ、興味など）や、生理指標（心拍数、脳波など）を組み込むことで、より深い洞察が得られる可能性があります。
- **機械学習の応用:** FCM クラスタリング以外の機械学習手法（例えば、深層学習）を応用することで、より高度な視聴者セグメンテーションや、視聴意図の予測モデル構築が可能になるかもしれません。

9.3　視聴者行動理解の重要性に関する結論

動画コンテンツが溢れる現代において、視聴者の心に響くプロモーション動画を制作するためには、データに基づいた客観的な分析が不可欠です。本研究は、VAS と FCM クラスタリング、重回帰分析を組み合わせることで、視聴者の感情反応と視聴意図の関係性を詳細に分析し、プロモーション動画戦略に役立つ具体的な知見を提供しました。

今後、さらに多様なデータと高度な分析手法を活用することで、視聴者行動の理解が深まり、より効果的で魅力的なプロモーション動画の制作に貢献できることを期待します。

付録 A

プログラムコード

A.1 build_sample.R

Listing A.1 build_sample.R

```
# サンプルデータの生成
source("R/create_sample_data.R")

# 言語設定 ("ja" または "en" を指定)
lang <- "ja"
data_file <- "sample_data.csv"
output_dir <- file.path("sample_output", lang)

source("R/basic_trend.R")
# 基本統計量とVASプロットの実行
basic_trend_result <- main_basic_trend(
  input_file = data_file,
  output_dir = output_dir,
  lang = lang,
  verbose = TRUE,
  benchmark = FALSE
)

source("R/fuzzy_cmeans.R")
# Fuzzy cmeansクラスタリングの実行
fuzzy_cmeans_result <- main_fuzzy_cmeans(
  input_file = data_file,
  output_dir = output_dir,
  lang = lang,
  verbose = TRUE,
  benchmark = FALSE
)

source("R/viewing_intention.R")
# 視聴意欲分析の実行
```

```
  viewing_intention_result <- main_viewing_intention(
    input_file = data_file,
    output_dir = output_dir,
    lang = lang,
    verbose = TRUE,
    benchmark = FALSE
  )
```

A.2 basic_trend.R

Listing A.2 basic_trend.R

```
# パッケージの読み込み
# ここでは、Rでデータ分析や可視化を行うためのライブラリ（パッケージ）を読み込んでいます。
# suppressPackageStartupMessages() は、パッケージ読み込み時のメッセージを非表示にする関数
suppressPackageStartupMessages({
    # tidyverse: データ操作、変換、可視化など、データサイエンスに必要な多くの機能を提供
    #            dplyr, ggplot2, tidyr, readr, purrr, tibble, stringr, forcats などのパ
    #            tidyverse は、データ分析のワークフローを効率化するための統一されたイン
    library(tidyverse)
    # ggbeeswarm: データの分布を可視化する際に、点が重ならないように描画するためのパッケ
    #             geom_quasirandom() 関数などで使用され、データの重なりを避けて視覚化で
    library(ggbeeswarm)
  })
# Rの基本的なライブラリ
# stats: 基本的な統計関数（平均、分散、検定など）を提供します。
# graphics: 基本的なグラフ描画関数（散布図、ヒストグラムなど）を提供します。
# grDevices: グラフィックデバイス（画面表示、PDF出力など）を制御します。
# utils: 様々なユーティリティ関数（データの読み込み、ヘルプの表示など）を提供します。
# methods: S4オブジェクトシステムを扱うための関数を提供します。
# これらのライブラリは、Rの基本的な機能を提供し、多くのパッケージで利用されています。

# utils.Rからヘルパー関数をロード
# source() 関数は、指定されたRスクリプトファイル（ここでは utils.R と translation_config
# その中に定義されている関数や変数を現在の環境で使用できるようにします。
# utils.R には、データの前処理や可視化、結果の保存など、このスクリプト全体で使われる便利
source("R/utils.R")
# translation_config.R には、グラフのラベルやメッセージを日本語と英語で切り替えるための設
source("R/translation_config.R")

#' @title メイン実行関数の改善版
#' @description この関数は、指定された入力ファイルを読み込み、データの前処理、可視化、統
#' @param input_file 入力ファイルパス (例: "data/data_all.csv")。CSV形式のデータを想定し
#' @param output_dir 出力ディレクトリ (例: "output/ja")。結果（グラフやデータ）を保存す
```

```
#' @param lang 言語コード ("ja" (日本語) または "en" (英語))。グラフのラベルやメッセージ
#' @param verbose デバッグ情報の表示フラグ (TRUE で詳細な情報を表示)。TRUEにすると、処理
#' @param benchmark ベンチマーク実行フラグ (TRUE で実行時間を計測)。TRUEにすると、各処理
main_basic_trend <- function(input_file, output_dir, lang = "ja", verbose = FALSE, bench
    # 初期化処理の開始時刻を記録
    start_time <- Sys.time()

    # 環境設定の初期化
    # verbose=TRUE: デバッグ情報を表示
    # install_packages=TRUE: 必要なパッケージを自動インストール
    # force_cairo=FALSE: 必要な場合のみCairoを使用
    # initialize_environment() 関数は、Rの実行環境を初期化する関数です (utils.R で定義さ
    # tryCatch() 関数は、エラーが発生した場合の処理を記述するための関数です。
    init_result <- tryCatch(
    {
        initialize_environment(verbose = TRUE)
      },
    error = function(e) {
        # エラーが発生した場合、メッセージを表示してNULLを返します。
        message("環境設定の初期化中にエラーが発生しました: ", e$message)
        return(NULL)
      }
    )

    # 初期化処理の終了時刻を記録
    end_time <- Sys.time()

    # 初期化結果の確認とログ出力
    if (!is.null(init_result)) {
        # 処理時間の計算と表示
        # difftime() 関数は、2つの時刻の差を計算します。
        processing_time <- difftime(end_time, start_time, units = "secs")
        message(sprintf("初期化処理の所要時間: %.2f秒", processing_time))

        # 初期化結果の表示
        if (init_result$success) {
            message("環境設定の初期化が正常に完了しました")
            message(sprintf("- 設定されたロケール: %s", init_result$locale))
            message(sprintf("- グラフィックデバイス: %s", init_result$graphics_device))
          } else {
            warning("環境設定の初期化で一部エラーが発生しました")
            # エラーメッセージの表示
            cat("詳細メッセージ:\n")
```

```
        cat(paste("- ", init_result$messages, collapse = "\n"))
      }
    } else {
      # 初期化に失敗した場合、エラーメッセージを表示してプログラムを停止します。
      stop("環境設定の初期化に失敗しました")
    }

# 出力ディレクトリの作成
if (verbose) message("必要なディレクトリを作成...")
# 出力ディレクトリ構造を設定
# file.path() 関数は、OSに依存しない形式でファイルパスを作成します。
plots_dir <- file.path(output_dir, "plots") # グラフを保存するディレクトリ
data_dir <- file.path(output_dir, "data") # データを保存するディレクトリ
# 作成するディレクトリのリスト
dirs <- c(plots_dir, data_dir)
# lapply() 関数は、リストの各要素に対して指定された関数を適用します。
# ここでは、dir.create() 関数を使ってディレクトリを作成しています。
# recursive = TRUE は、親ディレクトリが存在しない場合にそれも作成することを意味しま
# showWarnings = FALSE は、ディレクトリが既に存在する場合に警告を表示しないようにし
lapply(dirs, dir.create, recursive = TRUE, showWarnings = FALSE)

# データの読み込みと前処理
if (verbose) message("データの読み込みと前処理を開始...")
# load_and_preprocess_data() 関数は、入力ファイルを読み込んで前処理を行う関数です (
data <- load_and_preprocess_data(input_file, lang)

# データの読み込みに失敗した場合、メッセージを表示してNULLを返します。
if (is.null(data)) {
    message("データの読み込みに失敗しました")
    return(NULL)
  }

# VAS分布の可視化（全体）
if (verbose) message("VAS分布(全体)の可視化を開始...")
# create_vas_plot() 関数は、VAS（Visual Analogue Scale）の分布を可視化するグラフを作
vas_plot <- create_vas_plot(data$long, lang)
  # save_results() 関数は、グラフやデータをファイルに保存する関数です (utils.R で定義
  save_results(vas_plot, file.path(plots_dir, "vas_distribution.pdf")) # PDF形式で保
  save_results(vas_plot, file.path(plots_dir, "vas_distribution.svg")) # SVG形式で保

  # VAS分布の可視化（日本）
  if (verbose) message("VAS分布(日本)の可視化を開始...")
  # create_vas_plot() 関数を、国を日本に限定して実行します。
```

```
  vas_plot_japan <- create_vas_plot(data$long, lang, country_filter = "Japan")
save_results(vas_plot_japan, file.path(plots_dir, "vas_distribution_ja.pdf"))
save_results(vas_plot_japan, file.path(plots_dir, "vas_distribution_ja.svg"))

# VAS分布の可視化（シンガポール）
if (verbose) message("VAS分布(シンガポール)の可視化を開始...")
# create_vas_plot() 関数を、国をシンガポールに限定して実行します。
vas_plot_singapore <- create_vas_plot(data$long, lang, country_filter = "Singapore")
  save_results(vas_plot_singapore, file.path(plots_dir, "vas_distribution_sg.pdf"))
  save_results(vas_plot_singapore, file.path(plots_dir, "vas_distribution_sg.svg"))

  # VAS分布の国別比較
  if (verbose) message("VAS分布の国別比較を開始...")
  # create_vas_plots_comparison() 関数は、国別のVAS分布を比較するグラフを作成する関数
  vas_plots_comparison <- create_vas_plots_comparison(data$long, lang)
save_results(vas_plots_comparison, file.path(plots_dir, "vas_distribution_comparison
save_results(vas_plots_comparison, file.path(plots_dir, "vas_distribution_comparison

# 基本統計量の計算（全体）
if (verbose) message("基本統計量(全体)の計算を開始...")
# calculate_basic_stats() 関数は、基本統計量（平均値、標準偏差など）を計算する関数で
stats_all <- calculate_basic_stats(data$long, lang)
  if (verbose) message("全体の基本統計量を保存...(basic_stats.csv)")
  save_results(stats_all, file.path(data_dir, "basic_stats.csv"))

  # 基本統計量の計算（日本）
  if (verbose) message("日本の基本統計量を計算...")
  # calculate_basic_stats() 関数を、国を日本に限定して実行します。
  stats_japan <- calculate_basic_stats(data$long, lang, country_filter = "Japan")
if (verbose) message("日本の基本統計量を保存...(basic_stats_ja.csv)")
save_results(stats_japan, file.path(data_dir, "basic_stats_ja.csv"))

# 基本統計量の計算（シンガポール）
if (verbose) message("シンガポールの基本統計量を計算...")
# calculate_basic_stats() 関数を、国をシンガポールに限定して実行します。
stats_singapore <- calculate_basic_stats(data$long, lang, country_filter = "Singapor
  save_results(stats_singapore, file.path(data_dir, "basic_stats_sg.csv"))

  # 国別t検定
  if (verbose) message("国別t検定を開始...")
  # perform_country_ttest() 関数は、国別のt検定を行う関数です（後述）。
  t_test_results <- perform_country_ttest(data$long, lang)
save_results(t_test_results, file.path(data_dir, "country_ttest_results.csv"))
```

```
  }

#' @title 最適化された基本統計量の計算
#' @description 指定された国のデータに対して基本統計量を計算します
#' @param data 長形式データ
#' @param lang 言語コード ("ja" または "en")
#' @param country_filter 特定の国のデータのみを分析する場合の国名（オプション）
#' @return 基本統計量のデータフレーム
calculate_basic_stats <- function(data, lang = "ja", country_filter = NULL) {
    # 国別フィルタリング
    # country_filter が指定されている場合、指定された国のデータのみを抽出します。
    if (!is.null(country_filter)) {
        # filter() 関数は、dplyr パッケージの関数で、条件に一致する行を抽出します。
        # ここでは、Country 列が country_filter で指定された値と一致する行のみを抽出して
        data <- data %>% filter(Country == country_filter)
      }

    # tidyverseを使用した集計
    # ここでは、tidyverse の dplyr パッケージの関数を使って、データを質問ごとにグループ化
    # %>% はパイプ演算子で、左側のオブジェクトを右側の関数の第一引数として渡します。
    stats <- data %>%
    # group_by() 関数は、dplyr パッケージの関数で、指定された列（ここでは question）の値
    group_by(question) %>%
    # summarise() 関数は、dplyr パッケージの関数で、各グループに対して要約統計量を計算し
    summarise(
    # n() は、各グループの行数（サンプルサイズ）を計算します。
    n = n(),
    # mean() は、各グループの value 列の平均値を計算します。na.rm = TRUE は、欠損値（NA）
    mean = mean(value, na.rm = TRUE),
    # sd() は、各グループの value 列の標準偏差を計算します。
    sd = sd(value, na.rm = TRUE),
    # median() は、各グループの value 列の中央値を計算します。
    median = median(value, na.rm = TRUE),
    # quantile() は、各グループの value 列の四分位数を計算します。0.25 は第1四分位数、0.
    q1 = quantile(value, 0.25, na.rm = TRUE),
    q3 = quantile(value, 0.75, na.rm = TRUE),
    # min() は、各グループの value 列の最小値を計算します。
    min = min(value, na.rm = TRUE),
    # max() は、各グループの value 列の最大値を計算します。
    max = max(value, na.rm = TRUE),
    # .groups = "drop" は、グループ化を解除します。summarise() の後で group_by() の効果を
    .groups = "drop"
    )
```

```
    # 列名の変換
    # rename() 関数は、dplyr パッケージの関数で、列名を変更します。
    # !!get_translation(...) := ... の部分は、列名を動的に変更するための記法です。
    # get_translation() 関数は、指定された言語に対応する翻訳を取得します (translation_c
    stats <- stats %>%
    rename(
    # !!get_translation("x_label", "plot_labels", lang) := question は、
    # question 列の名前を、get_translation("x_label", "plot_labels", lang) の戻り値に変
    !!get_translation("x_label", "plot_labels", lang) := question,
    !!get_translation("sample_size", "stat_columns", lang) := n,
    !!get_translation("mean", "stat_columns", lang) := mean,
    !!get_translation("sd", "stat_columns", lang) := sd,
    !!get_translation("median", "stat_columns", lang) := median,
    !!get_translation("q1", "stat_columns", lang) := q1,
    !!get_translation("q3", "stat_columns", lang) := q3,
    !!get_translation("min", "stat_columns", lang) := min,
    !!get_translation("max", "stat_columns", lang) := max
    )

    return(stats)
  }

#' @title 最適化されたVASプロットの作成
#' @description この関数は、VAS(Visual Analogue Scale) のデータを用いて、質問項目ごとの
#' @param data 長形式データ (各行が1つの回答に対応し、question, value, Country などの列
#' @param lang 言語コード ("ja" または "en")
#' @param country_filter 特定の国のデータのみを表示する場合の国名 (例: "Japan")
#' @return ggplotオブジェクト (このオブジェクトを print() 関数で表示したり、ggsave() 関
create_vas_plot <- function(data, lang = "ja", country_filter = NULL) {
# 国別フィルタリング
# country_filter が指定されている場合、指定された国のデータのみを抽出します。
if (!is.null(country_filter)) {
    # dplyr の filter() 関数を使って、Country 列が指定された国名と一致する行のみを抽出し
    data <- data %>% filter(Country == country_filter)
    # プロットのタイトルを設定します。
    plot_title <- sprintf(
    "%s␣(%s)",
    get_translation("main_title", "plot_labels", lang), # 国名なしのタイトル
    country_filter # 国名
    )
  } else {
    # 国が指定されていない場合は、国名なしのタイトルを使用します。
```

```
    plot_title <- get_translation("main_title", "plot_labels", lang)
  }

# データの事前集計を最適化
# ここでは、プロットを効率的に生成するために、データを質問項目ごとにグループ化し、各質問
plot_data <- data %>%
# dplyr の group_by() 関数を使って、question 列の値でデータをグループ化します。
group_by(question) %>%
# dplyr の summarise() 関数を使って、各グループに対して value 列の値をリストとしてまとめ
summarise(
value_list = list(value), # 各質問に対する回答値をリストとして格納
.groups = "drop" # グループ化を解除
)

# より効率的なプロット生成
# ggplot2 は、Rでグラフを作成するための非常に強力なライブラリです。
# ggplot2 では、データ、マッピング（データと視覚要素の対応関係）、ジオメトリ（グラフの種
# "+" 演算子を使って、グラフの要素を層のように重ねていくことができます。
p <- ggplot() + # 空の ggplotオブジェクトを作成
# map() 関数を使って、各質問項目に対してバイオリンプロットと箱ひげ図を生成します。
map(seq_len(nrow(plot_data)), function(i) {
list(
# バイオリンプロットを直接生成
# geom_violin() は、ggplot2 の関数で、バイオリンプロットを描画します。
# バイオリンプロットは、データの分布を滑らかに表現するグラフです。
geom_violin(
data = tibble( # 各質問項目に対するデータを作成
question = plot_data$question[i],
  value = unlist(plot_data$value_list[i]) # リストを展開してベクトルにする
),
aes(x = question, y = value), # x軸に質問項目、y軸に回答値をマッピング
alpha = 0.4 # プロットの透明度を0.4に設定
),
# 箱ひげ図を生成
# geom_boxplot() は、ggplot2 の関数で、箱ひげ図を描画します。
# 箱ひげ図は、データの中央値、四分位数、外れ値を表示するグラフです。
geom_boxplot(
data = tibble( # 各質問項目に対するデータを作成
question = plot_data$question[i],
  value = unlist(plot_data$value_list[i])
),
aes(x = question, y = value),
width = 0.2, # 箱の幅を0.2に設定
```

```
alpha = 0.4,
outlier.shape = NA # 外れ値を表示しない
)
)
}) +
# より効率的なbeeswarmプロット
# geom_quasirandom() は、beeswarmプロット（点が重ならないように配置された散布図）を描画
# ggbeeswarm パッケージの関数です。
geom_quasirandom(
data = data, # 元のデータを使用
aes(x = question, y = value),
alpha = 0.4, # プロットの透明度を0.4に設定
size = 1.5, # 点のサイズを1.5に設定
width = 0.2, # 点の広がり具合を調整
groupOnX = TRUE # x軸方向にグループ化
) +
# ラベル設定
# labs() 関数は、ggplot2 の関数で、グラフのタイトルや軸ラベルを設定します。
labs(
title = plot_title, # グラフのタイトル
x = get_translation("x_label", "plot_labels", lang), # x軸のラベル
y = get_translation("y_label", "plot_labels", lang) # y軸のラベル
) +
# 共通テーマの適用
# apply_common_theme() 関数は、グラフの見た目を整えるための共通のテーマを適用します (ut
apply_common_theme(rotate_x_labels = TRUE) + # x軸のラベルを回転
# テーマの詳細設定
# theme() 関数は、ggplot2 の関数で、グラフの細かい見た目を調整します。
theme(
text = element_text(size = 14), # 全体のフォントサイズを14に設定
axis.title = element_text(size = 20), # 軸タイトルのフォントサイズを20に設定
axis.text = element_text(size = 20), # 軸ラベルのフォントサイズを20に設定
plot.title = element_text(size = 18), # グラフタイトルのフォントサイズ
legend.title = element_text(size = 14), # 凡例タイトルのフォントサイズ
legend.text = element_text(size = 14) # 凡例のフォントサイズ
)

return(p)
}

#' @title 国別VASプロットの作成
#' @description この関数は、国ごとにVASの値を比較するプロットを作成します。
#' @param data 長形式データ
```

```
#' @param lang 言語コード
#' @return ggplotオブジェクト
create_vas_plots_comparison <- function(data, lang = "ja") {
    # 国別のバイオリンプロット + 箱ひげ図 + データ点
    # ggplot2 を使用してグラフを作成します。
    p <- ggplot(data, aes(x = question, y = value, fill = Country)) +
    # geom_violin() は、バイオリンプロットを描画します。
    # position_dodge() は、複数のグループがある場合に、それぞれのバイオリンプロットを少
    geom_violin(
    position = position_dodge(width = 0.7),
    alpha = 0.5, # 透明度
    scale = "width", # 各バイオリンプロットの幅を、そのグループのデータ数に合わせて調整
    trim = TRUE # バイオリンプロットの両端を、データの範囲に合わせる
    ) +
    # geom_boxplot() は、箱ひげ図を描画します。
    geom_boxplot(
    position = position_dodge(width = 0.7), # 箱ひげ図もずらして表示
    width = 0.2, # 箱の幅
    alpha = 0.7,
    outlier.shape = NA # 外れ値は点プロットで表示するため非表示
    ) +
    # geom_point() は、データ点を描画します。
    # position_jitterdodge() は、データ点を少しずらして、重なりを避けるように配置します。
    geom_point(
    position = position_jitterdodge(
    dodge.width = 0.7,
    jitter.width = 0.1
    ),
    alpha = 0.3,
    size = 1
    ) +
    # scale_fill_grey() は、グラフの塗りつぶし色をグレースケールに設定します。
    scale_fill_grey(start = 0.4, end = 0.8) +
    # labs() は、グラフのタイトルや軸ラベルを設定します。
    labs(
    title = get_translation("country_comparison", "plot_labels", lang),
    x = get_translation("x_label", "plot_labels", lang),
    y = get_translation("y_label", "plot_labels", lang),
    fill = get_translation("country", "plot_labels", lang)
    ) +
    # scale_y_continuous() は、y軸の範囲や目盛りの間隔を設定します。
    scale_y_continuous(
    limits = c(0, 1.00),
```

```
    breaks = seq(0, 1.00, 0.20)
    ) +
    # apply_common_theme() は、グラフに共通のテーマを適用します (utils.R で定義)。
    apply_common_theme(rotate_x_labels = TRUE) +
    # theme() は、グラフの細かい見た目を調整します。
    theme(
    text = element_text(size = 14),
    axis.title = element_text(size = 20),
    axis.text = element_text(size = 20),
    plot.title = element_text(size = 18),
    legend.title = element_text(size = 14),
    legend.text = element_text(size = 14),
    legend.position = "bottom" # 凡例を下に表示
    )
    return(p)
  }

#' @title 国別のt検定実施
#' @description この関数は、2つの国間で、各質問項目に対する回答の平均値に差があるかどう
#' @param data 長形式データ
#' @param lang 言語コード
#' @return t検定結果のデータフレーム
perform_country_ttest <- function(data, lang = "ja") {
    # 質問項目ごとにt検定を実施
    # unique() 関数で、質問項目の一意なリストを取得します。
    questions <- unique(data$question)

    # map_dfr() 関数は、purrr パッケージの関数で、リストの各要素に関数を適用し、結果を1
    t_results <- map_dfr(questions, function(q) {
        # filter() 関数で、現在の質問項目 q に対応するデータのみを抽出します。
        subset_data <- data %>% filter(question == q)

        # t.test() 関数は、Rの組み込み関数で、t検定を行います。
        # value ~ Country は、Country 変数の値によって value 変数をグループ分けして比較
        t_test <- t.test(value ~ Country, data = subset_data)

        # 各国の平均値を計算します。
        means <- subset_data %>%
        group_by(Country) %>%
        summarise(mean = mean(value, na.rm = TRUE), .groups = "drop")
        # 平均値の差を計算します。
        mean_diff <- diff(means$mean)
```

```
      # tibble() 関数は、tibble（tidyverse のデータフレーム）を作成します。
      tibble(
      question = q, # 質問項目
      t_statistic = t_test$statistic , # t統計量
    p_value = t_test$p.value , # p値
      mean_diff = mean_diff , # 平均値の差
      conf_low = t_test$conf.int[1], # 95%信頼区間の下限
    conf_high = t_test$conf.int[2] # 95%信頼区間の上限
    )
  })

  # 列名を、translation_config.R で定義された翻訳を使って変更します。
  t_results <- t_results %>%
  rename(
  !!get_translation("x_label", "plot_labels", lang) := question ,
  !!get_translation("t_stat", "stat_columns", lang) := t_statistic ,
  !!get_translation("p_value", "stat_columns", lang) := p_value ,
  !!get_translation("mean_diff", "stat_columns", lang) := mean_diff ,
  !!get_translation("conf_low", "stat_columns", lang) := conf_low ,
  !!get_translation("conf_high", "stat_columns", lang) := conf_high
  )

  return(t_results)
}
```

A.3 fuzzy_cmeans.R

Listing A.3 fuzzy_cmeans.R

```
  # このスクリプトは、Fuzzy C-means (FCM) クラスタリングを用いて、アンケートデータを分析
# FCMは、各データ点が複数のクラスターに所属する可能性を許容する「ファジィ」なクラスタリ
# 各データ点は、各クラスターに対する「メンバーシップ値」を持ち、この値が高いほど、そのク

# パッケージの読み込み
suppressPackageStartupMessages({
  library(tidyverse) # データ操作のためのライブラリ
  library(e1071) # FCMクラスタリングのためのライブラリ
  library(scales) # グラフのスケール調整のためのライブラリ
})

source("R/utils.R")
source("R/translation_config.R")

#' @title クラスタリング結果の散布図作成
```

```
#' @description 入力データ、メンバーシップ行列、クラスター中心点を用いて、クラスタリング
#' @param data 入力データ (data.frame)。質問項目(Q*)を含む必要があります。
#' @param membership メンバーシップ行列 (matrix)。各データ点が各クラスターに所属する度合
#' @param centers クラスターの中心点 (matrix)。各クラスターの中心座標を示します。
#' @param labels クラスターのラベル (character vector)。各クラスターのラベルです。
#' @param lang 言語コード (character)。グラフのラベルに使用する言語を指定します ("ja" or
#' @return ggplotオブジェクト。作成された散布図です。
create_cluster_plot <- function(data, membership, centers, labels, lang = "ja") {
tryCatch(
{
# データの準備
analysis_data <- data %>%
select(starts_with("Q")) %>%
select(1:2) %>% # Q1とQ2のみを使用
mutate_all(as.numeric) %>% # 数値型に変換
scale() # 標準化

plot_data <- tibble(
x = analysis_data[, 1], # 1列目をx座標とする
y = analysis_data[, 2], # 2列目をy座標とする
cluster = factor(max.col(membership), # メンバーシップ値が最大のクラスターを取得
levels = 1:length(labels), # クラスターのレベルを設定
labels = labels # クラスターのラベルを設定
),
membership = apply(membership, 1, max) # 各データ点の最大メンバーシップ値を取得
)

# 中心点の準備
centers_data <- tibble(
x = centers[, 1], # 1列目をx座標とする
y = centers[, 2], # 2列目をy座標とする
cluster = factor(1:length(labels), # クラスターのレベルを設定
levels = 1:length(labels), # クラスターのレベルを設定
labels = labels # クラスターのラベルを設定
)
)

# プロット生成
p <- ggplot(plot_data, aes(x = x, y = y)) +
geom_point(
aes(color = cluster, size = membership, alpha = membership) # クラスターごとに色分けし、
) +
scale_size_continuous(range = c(2, 5)) + # メンバーシップ値に応じた点のサイズを設定
```

```
scale_alpha_continuous(range = c(0.3, 0.8)) + # メンバーシップ値に応じた点の透明度を設定
geom_point( # クラスター中心点を描画
data = centers_data,
aes(x = x, y = y),
color = "black", # 中心点の色を黒に設定
size = 4, # 中心点のサイズを設定
shape = 8 # 中心点の形を星形に設定
) +
labs( # グラフのラベルを設定
title = get_translation("cluster_title", "plot_labels", lang),
x = get_translation("Q1", "questions", lang),
y = get_translation("Q2", "questions", lang),
color = get_translation("cluster", "cluster_labels", lang),
size = get_translation("membership", "cluster_labels", lang),
alpha = get_translation("membership", "cluster_labels", lang)
) +
scale_color_brewer(palette = "Set2") + # 色分けに"Set2"パレットを使用
theme_minimal(base_size = 14) + # 最小限のテーマを適用し、フォントサイズを設定
theme( # テーマの詳細設定
plot.title = element_text(size = rel(1.6), hjust = 0.5),
axis.title = element_text(size = rel(1.6)),
axis.text = element_text(size = rel(1.6)),
legend.title = element_text(size = rel(1.6)),
legend.text = element_text(size = rel(1.1)),
legend.position = "bottom",
legend.box = "vertical",
panel.grid.minor = element_blank()
)

return(p) # 作成したプロットを返す
},
error = function(e) { # エラーハンドリング
    warning(sprintf("散布図の作成中にエラー:␣%s", e$message)) # エラーメッセージを出力
    return(NULL) # NULLを返す
  }
)
}

#' @title Fuzzy c-means クラスタリング実行と可視化
#' @description Fuzzy c-means (FCM) クラスタリングを実行し、結果をリスト形式で返します。
#' @param data アンケートの生データ (data.frame)。質問項目(Q*)を含む必要があります。
#' @param k クラスター数 (integer)。生成するクラスターの数を指定します。
#' @param m ファジィ化パラメータ (numeric)。メンバーシップ関数の形状を制御するパラメータ
```

```
#' 値が大きいほど、クラスター間の境界が曖昧になります。
#' @param max_iter 最大反復回数 (integer)。FCMアルゴリズムの最大反復回数を指定します。
#' @param lang 言語コード (character)。クラスターラベルに使用する言語を指定します ("ja"
#' @return クラスタリング結果のリスト。以下の要素を含みます:
#' \itemize{
  #'   \item clusters: FCMアルゴリズムの結果オブジェクト (cmeansオブジェクト)。
  #'   \item labels: クラスターラベル (character vector)。
  #'   \item sorted_indices: クラスター中心の平均値でソートされたインデックス (integer
  #'   \item center_means: クラスター中心の平均値 (numeric vector)。
  #' }
perform_clustering <- function(data, k = 4, m = 4, max_iter = 1000, lang = "ja") {
tryCatch(
{
# データの前処理と正規化
analysis_data <- data %>%
select(starts_with("Q")) %>% # Qで始まる列を選択
na.omit() %>% # 欠損値を含む行を除外
scale() # データを標準化 (平均0、分散1)

# データの検証
if (nrow(analysis_data) == 0 || ncol(analysis_data) == 0) {
    stop("前処理後に有効なデータがありません")
  }

# クラスタリングの実行
set.seed(123) # 乱数シードを固定し、結果の再現性を確保
message("\nクラスタリングを開始")

# Fuzzy C-means クラスタリングの実行
# FCMアルゴリズムは、以下の手順を繰り返します。
# 1. 初期化: 各クラスターの中心をランダムに初期化します。
# 2. メンバーシップ値の計算: 各データ点について、各クラスターへの所属度合い(メンバーシッ
#    メンバーシップ値は、データ点とクラスター中心との距離に基づいて計算され、合計は1にな
# 3. クラスター中心の更新: 各クラスターの中心を、メンバーシップ値で重み付けされたデータ
# 4. 収束判定: クラスター中心の変化が十分に小さくなるか、最大反復回数に達するまで、2と3
fuzzy_result <- cmeans(
analysis_data, # クラスタリング対象のデータ
centers = k, # クラスター数
m = m, # ファジィ化パラメータ
iter.max = max_iter, # 最大反復回数
dist = "euclidean", # 距離関数 (ユークリッド距離)
method = "cmeans", # FCMアルゴリズムを指定
verbose = TRUE # 詳細な情報を出力
```

```
)

# メンバーシップ行列の検証
if (is.null(fuzzy_result$membership) || nrow(fuzzy_result$membership) == 0) {
    stop("クラスタリング結果が無効です")
  }

# クラスターの中心点の平均値を計算
center_means <- rowMeans(fuzzy_result$centers)

# クラスターラベルの設定
label_assignments <- c(
"high_eval", "mid_eval", "mixed_eval", "low_eval"
)

# 中心点の平均値でソート
sorted_indices <- order(center_means, decreasing = TRUE)
cluster_labels <- character(k)

for (i in 1:k) {
original_index <- which(sorted_indices == i)
label_key <- if (original_index <= length(label_assignments)) {
label_assignments[original_index]
} else {
    sprintf("cluster_%d", i) # 想定外のクラスター数になった場合のラベル
  }
cluster_labels[i] <- get_translation(label_key, "cluster_labels", lang)
}

# デバッグ情報の出力
message("\n=== クラスター分析結果 ===")
for (i in 1:k) {
message(sprintf(
"クラスター %d: 平均値 = %.3f, ラベル = %s",
i, center_means[i], cluster_labels[i]
))
}

# クラスタリング結果を返す
list(
clusters = fuzzy_result, # FCMアルゴリズムの結果オブジェクト
labels = cluster_labels, # クラスターラベル
sorted_indices = sorted_indices, # クラスター中心の平均値でソートされたインデックス
```

```
center_means = center_means # クラスター中心の平均値
)
},
error = function(e) { # エラーハンドリング
    message("クラスタリング中にエラー:␣", e$message) # エラーメッセージを出力
    return(NULL) # NULLを返す
  }
)
}

#' @title クラスターごとのVASプロット生成
#' @description 各クラスターについて、VAS (Visual Analog Scale) 値の分布を示すプロットを
#' @param data 長形式データ (data.frame)。'ID'、'question'、'value' の各列を含む必要があ
#' @param cluster_info クラスター情報 (data.frame)。'ID' と 'cluster' の各列を含む必要が
#'  'cluster_summary' 属性を持っている必要があります。
#' @param plots_dir プロット保存ディレクトリ (character)。生成されたプロットを保存するテ
#' @param lang 言語コード (character)。グラフのラベルに使用する言語を指定します ("ja" or
#' @param k クラスター数 (integer)。
#' @return プロットのリスト (list)。各クラスターに対応するggplotオブジェクトを含みます。
create_cluster_vas_plots <- function(data, cluster_info, plots_dir, lang = "ja", k = 4)
    tryCatch(
    {
        # クラスター情報の検証
        if (is.null(cluster_info)) {
            stop("クラスター情報がNULLです")
          }

        cluster_summary <- attr(cluster_info, "cluster_summary")
        if (is.null(cluster_summary)) {
            stop("クラスター概要情報が見つかりません")
          }

        # データの結合と検証
        data_with_clusters <- data %>%
        mutate(ID = as.character(ID)) %>% # IDを文字列型に変換
        inner_join( # クラスター情報を結合
        cluster_info %>%
        mutate(
        ID = as.character(ID), # IDを文字列型に変換
        cluster = as.factor(cluster) # clusterを因子型に変換
        ) %>%
        select(ID, cluster), # IDとcluster列を選択
        by = "ID" # IDをキーとして結合
```

```
)

if (nrow(data_with_clusters) == 0) {
    stop("結合後のデータが空です")
  }

# プロット生成
plots <- list()
ordered_clusters <- cluster_summary$cluster # クラスターの順序を取得

for (cluster_label in ordered_clusters) {
    # 現在のクラスターのデータを抽出
    cluster_data <- data_with_clusters %>%
    filter(cluster == cluster_label)

    if (nrow(cluster_data) > 0) {
        message(sprintf(
        "\nクラスター␣%s␣のデータを処理中␣(%d␣行)",
        cluster_label, nrow(cluster_data)
        ))

        # プロット生成
        p <- ggplot(cluster_data, aes(x = question, y = value)) +
        geom_boxplot(width = 0.2, alpha = 0.4, outlier.shape = NA) + # 箱ひげ図
        geom_violin(alpha = 0.4) + # バイオリンプロット
        geom_jitter(alpha = 0.4, width = 0.2, height = 0) + # ジッタープロット
        labs( # グラフのラベルを設定
        title = sprintf(
        "%s:␣%s",
        get_translation("cluster", "cluster_labels", lang),
        cluster_label
        ),
        x = get_translation("x_label", "plot_labels", lang),
        y = get_translation("y_label", "plot_labels", lang)
        ) +
        scale_y_continuous( # y軸のスケールを設定
        limits = c(0, 1.00), # 範囲を0から1に設定
        breaks = seq(0, 1.00, 0.20) # 目盛りを0.2刻みに設定
        ) +
        theme_minimal(base_size = 14) + # 最小限のテーマを適用し、フォントサイズ
        theme( # テーマの詳細設定
        plot.title = element_text(size = rel(1.6), hjust = 0.5),
        axis.title = element_text(size = rel(1.6)),
```

```
                axis.text = element_text(size = rel(1.6)),
                axis.text.x = element_text(angle = 45, hjust = 1),
                panel.grid.minor = element_blank(),
                plot.margin = margin(t = 10, r = 10, b = 10, l = 10)
                )

                plots[[cluster_label]] <- p # プロットをリストに追加

                # プロットの保存
                if (!is.null(plots_dir)) {
                    safe_name <- gsub("␣", "_", tolower(cluster_label)) # クラスターラベ
                    filename_base <- file.path(plots_dir, sprintf("vas_cluster_%s", safe

                    tryCatch(
                    {
                        save_results(p, paste0(filename_base, ".pdf")) # PDF形式で保存
                        save_results(p, paste0(filename_base, ".svg")) # SVG形式で保存
                        message(sprintf("クラスター␣%s␣のプロットを保存しました", cluste
                      },
                    error = function(e) {
                        warning(sprintf(
                        "クラスター␣%s␣のプロット保存中にエラー:␣%s",
                        cluster_label,
                        e$message
                        ))
                      }
                    )
                  }
                } else {
                  warning(sprintf("クラスター␣%s␣のデータが空です", cluster_label))
                }
            }

        return(plots) # プロットのリストを返す
      },
    error = function(e) { # エラーハンドリング
        warning(sprintf("クラスターVASプロットの作成中にエラー:␣%s", e$message)) # エラ
        return(NULL) # NULLを返す
      }
    )
  }

#' @title メイン実行関数
```

```
#' @description Fuzzy C-meansクラスタリングを実行し、結果を可視化・保存するメイン関数で
#' @param input_file 入力ファイルパス (character)。CSV形式のデータファイルへのパス。
#' @param output_dir 出力ディレクトリ (character)。結果を保存するディレクトリへのパス。
#' @param lang 言語コード (character)。グラフのラベル等に使用する言語 ("ja" or "en")。
#' @param verbose デバッグ情報の表示フラグ (logical)。TRUEの場合、詳細な情報を出力します
#' @param benchmark ベンチマーク実行フラグ (logical)。TRUEの場合、実行時間を計測します (
#' @return 成功時は 'invisible(TRUE)'、エラー時は 'invisible(FALSE)' を返します。
main_fuzzy_cmeans <- function(input_file, output_dir, lang = "ja", verbose = FALSE, benc
    tryCatch(
    {
        if (verbose) message("クラスター分析を開始...")

        # 環境設定の初期化
        init_result <- initialize_environment(verbose = TRUE)
        if (is.null(init_result) || !init_result$success) {
            stop("環境設定の初期化に失敗しました")
          }

          # 出力ディレクトリの作成
          plots_dir <- file.path(output_dir, "plots") # プロット保存用ディレクトリ
          data_dir <- file.path(output_dir, "data") # データ保存用ディレクトリ
          dirs <- c(plots_dir, data_dir)
          lapply(dirs, dir.create, recursive = TRUE, showWarnings = FALSE) # ディレクト

          # データの読み込みと前処理
          data <- load_and_preprocess_data(input_file, lang)
          if (is.null(data)) {
              stop("データの読み込みに失敗しました")
            }

          # クラスター分析の実行と散布図の生成
          k <- 4 # クラスター数
          cluster_results <- perform_clustering(data$raw, k = k, lang = lang) # クラスタ
        if (is.null(cluster_results)) {
            stop("クラスター分析に失敗しました")
          }

        # クラスタリング結果の散布図を生成
        cluster_plot <- create_cluster_plot(
        data = data$raw, # 元データ
          membership = cluster_results$clusters$membership, # メンバーシップ行列
          centers = cluster_results$clusters$centers, # クラスター中心
          labels = cluster_results$labels, # クラスターラベル
```

```
    lang = lang # 言語設定
    )

    # 散布図の保存
    if (!is.null(cluster_plot)) {
      save_results(cluster_plot, file.path(plots_dir, "clustering.pdf")) # PDF形式
      save_results(cluster_plot, file.path(plots_dir, "clustering.svg")) # SVG形式
      message("クラスタリング結果の散布図を保存しました")
    } else {
      warning("クラスタリング結果の散布図の生成をスキップしました")
    }

    # クラスター情報の作成
    cluster_info <- tibble(
    ID = data$raw$ID, # ID
    cluster = factor( # クラスター番号 (メンバーシップ値が最大のものを採用)
    max.col(cluster_results$clusters$membership),
    levels = 1:k,
    labels = cluster_results$labels
      ),
      membership = apply(cluster_results$clusters$membership, 1, max) # 最大メンバー
      )

      # クラスター概要情報を属性として追加
      attr(cluster_info, "cluster_summary") <- tibble(
      cluster = factor(1:k, levels = 1:k, labels = cluster_results$labels), # クラス
    center_mean = cluster_results$center_means, # クラスター中心の平均値
      sorted_index = cluster_results$sorted_indices # ソートされたインデックス
    )

    # クラスター情報の保存
    save_results(cluster_info, file.path(data_dir, "cluster_info.csv")) # CSV形式で

    # クラスターごとのVASプロット生成と保存
    create_cluster_vas_plots(data$long, cluster_info, plots_dir, lang, k)

    message("クラスター分析が完了しました")
    return(invisible(TRUE)) # 成功
  },
  error = function(e) { # エラーハンドリング
    message(sprintf("クラスター分析の実行中にエラー: %s", e$message)) # エラーメッセ
    return(invisible(FALSE)) # 失敗
  }
```

```
    )
  }
```

A.4 viewing_intention.R

Listing A.4 viewing_intention.R

```
# パッケージの読み込み
suppressPackageStartupMessages({
  library(tidyverse)
  library(ggbeeswarm)
})

# utils.Rからヘルパー関数をロード
source("R/utils.R")
source("R/translation_config.R")

#' @title 観覧意図分析のメイン実行関数
#' @description
#' CSVファイルから映画の印象評価データを読み込み、観覧意図に影響を与える
#' 要因を分析します。環境設定の初期化、データの前処理、統計分析を
#' 一連の流れとして実行します。
#'
#' @param input_file 入力CSVファイルのパス
#' @param output_dir 分析結果の出力先ディレクトリ
#' @param lang 言語設定（"ja"：日本語、"en"：英語）
#' @param verbose デバッグ情報の詳細表示フラグ
#' @param benchmark 処理時間計測フラグ
#' @return NULL
#' @details
#' 以下の手順で分析を実行します：
#' 1. 環境設定の初期化（フォント、ロケール等）
#' 2. 出力ディレクトリの作成
#' 3. データの読み込みと前処理
#' 4. 観覧意図の要因分析
#'
#' エラー処理：
#' - 環境設定の初期化失敗
#' - データファイルの読み込み失敗
#' - 分析実行時のエラー
main_viewing_intention <- function(input_file, output_dir, lang = "ja", verbose = FALSE,
  # 初期化処理の開始時刻を記録
  start_time <- Sys.time()
```

```
# 環境設定の初期化
# verbose=TRUE: デバッグ情報を表示
# install_packages=TRUE: 必要なパッケージを自動インストール
# force_cairo=FALSE: 必要な場合のみCairoを使用
init_result <- tryCatch(
  {
    initialize_environment(verbose = TRUE)
  },
  error = function(e) {
    message("環境設定の初期化中にエラーが発生しました: ", e$message)
    return(NULL)
  }
)

# 初期化処理の終了時刻を記録
end_time <- Sys.time()

# 初期化結果の確認とログ出力
if (!is.null(init_result)) {
  # 処理時間の計算と表示
  processing_time <- difftime(end_time, start_time, units = "secs")
  message(sprintf("初期化処理の所要時間: %.2f秒", processing_time))

  # 初期化結果の表示
  if (init_result$success) {
    message("環境設定の初期化が正常に完了しました")
    message(sprintf("- 設定されたロケール: %s", init_result$locale))
    message(sprintf("- グラフィックデバイス: %s", init_result$graphics_device))
  } else {
    warning("環境設定の初期化で一部エラーが発生しました")
    # エラーメッセージの表示
    cat("詳細メッセージ:\n")
    cat(paste("- ", init_result$messages, collapse = "\n"))
  }
} else {
  stop("環境設定の初期化に失敗しました")
}

if (verbose) message("必要なディレクトリを作成...")
# 出力ディレクトリ構造を設定
plots_dir <- file.path(output_dir, "plots")
data_dir <- file.path(output_dir, "data")
dirs <- c(plots_dir, data_dir)
```

```
  lapply(dirs, dir.create, recursive = TRUE, showWarnings = FALSE)

  if (verbose) message("データの読み込みと前処理を開始...")
  data <- load_and_preprocess_data(input_file, lang)

  if (is.null(data)) {
    message("データの読み込みに失敗しました")
    return(NULL)
  }

  # ビューティングインテンションの分析
  # データフレームを取り出して分析を実行
  analysis_results <- analyze_viewing_intention(data$raw, lang = lang)
  if (is.null(analysis_results)) {
    stop(get_error_message("analysis_failed", lang))
  }

  # 結果の処理
  influence_summary <- analysis_results$influence_summary
  model_fit <- analysis_results$model_fit
  correlation_matrix <- analysis_results$correlation_matrix
  model <- analysis_results$model

  # 結果の保存
  save_results(influence_summary, file.path(data_dir, "viewing_intention_influence_summ
  save_results(model_fit, file.path(data_dir, "viewing_intention_model_fit.csv"))
  save_results(as.data.frame(correlation_matrix), file.path(data_dir, "viewing_intention
  saveRDS(model, file.path(data_dir, "viewing_intention_regression_model.rds"))

  # プロット生成
  plots <- list(
    coefficient = create_coefficient_plot(analysis_results, lang),
    correlation = create_correlation_plot(analysis_results, lang)
  )

  # プロットの保存
  save_results(plots$coefficient, file.path(plots_dir, "viewing_intention_coefficient.p
  save_results(plots$coefficient, file.path(plots_dir, "viewing_intention_coefficient.sv
  save_results(plots$correlation, file.path(plots_dir, "viewing_intention_correlation.p
  save_results(plots$correlation, file.path(plots_dir, "viewing_intention_correlation.sv
}

#' @title 係数プロットの生成関数
```

```r
#' @description
#' 重回帰分析の結果から、各要因の標準化係数を視覚化するプロットを
#' 生成します。係数の大きさと信頼区間を表示し、影響度の比較を
#' 容易にします。
#'
#' @param analysis_results analyze_viewing_intentionの結果リスト
#' @param lang 言語設定（"ja"：日本語、"en"：英語）
#' @return ggplotオブジェクト（係数プロット）
#' @details
#' プロットの特徴：
#' - 標準化係数のポイントプロット
#' - 95%信頼区間の表示
#' - 要因名の翻訳対応
create_coefficient_plot <- function(analysis_results, lang = "ja") {
  # 影響度データを取得
  influence_data <- analysis_results$influence_summary

  # 信頼区間の計算（t分布の95%信頼区間）
  ci_margin <- influence_data$Std_Error * qt(0.975, df = nrow(influence_data) - 1)

  # プロットの生成
  ggplot(influence_data, aes(x = Std_Estimate, y = Variable)) +
    geom_vline(xintercept = 0, linetype = "dashed", color = "gray50") +
    geom_point(size = 3, color = "steelblue") +
    geom_errorbarh(
      aes(
        xmin = Std_Estimate - ci_margin,
        xmax = Std_Estimate + ci_margin
      ),
      height = 0.2,
      color = "steelblue"
    ) +
    labs(
      title = get_translation("coefficient_plot_title", "influence_analysis", lang),
      x = get_translation("axis_variable", "influence_analysis", lang),
      y = get_translation("axis_std_coefficient", "influence_analysis", lang)
    ) +
    apply_common_theme()
}

#' @title 相関プロットの生成関数
#' @description
#' 変数間の相関関係を視覚化するヒートマッププロットを生成します。
```

```r
#' 各変数ペアの相関係数を色の濃淡で表現し、正負の関係性の強さを
#' 直感的に把握できるようにします。
#'
#' @param analysis_results analyze_viewing_intention()関数から返される分析結果のリスト
#' @param lang 言語設定（"ja"：日本語、"en"：英語）
#' @return ggplotオブジェクト
#'   - 相関係数をヒートマップで表示
#'   - 各セルに相関係数の値を数値で表示
#'   - 軸ラベルは指定言語で表示
#' @details
#' プロットの仕様：
#' - 相関係数の表示範囲：-1から1
#' - 色使い：
#'     - 正の相関：赤色（強いほど濃い）
#'     - 負の相関：青色（強いほど濃い）
#'     - 無相関：白色
#' - 軸の変数名は translation_config.R で定義された翻訳を使用
#'
#' @examples
#' \dontrun{
#' # 日本語での相関プロット生成
#' plot <- create_correlation_plot(analysis_results, lang = "ja")
#' print(plot)
#'
#' # 英語での相関プロット生成
#' plot <- create_correlation_plot(analysis_results, lang = "en")
#' print(plot)
#' }
#'
#' @seealso
#' - \code{\link{analyze_viewing_intention}} - 分析結果の生成
#' - \code{\link{get_translation}} - 翻訳文字列の取得
create_correlation_plot <- function(analysis_results, lang = "ja") {
  # 相関行列を取得
  cor_matrix <- analysis_results$correlation_matrix

  # 変数名を翻訳
  var_names <- c(
    get_translation("excitement", "influence_analysis", lang),
    get_translation("anger_joy", "influence_analysis", lang),
    get_translation("joy", "influence_analysis", lang),
    get_translation("anger", "influence_analysis", lang),
    get_translation("intention", "influence_analysis", lang)
```

```
  )

  # データフレームに変換
  cor_df <- cor_matrix %>%
    as.data.frame() %>%
    rownames_to_column("Var1") %>%
    pivot_longer(-Var1, names_to = "Var2", values_to = "Correlation") %>%
    mutate(
      Var1 = factor(Var1, levels = paste0("Q", 1:5), labels = var_names),
      Var2 = factor(Var2, levels = paste0("Q", 1:5), labels = var_names)
    )

  # プロットの生成
  ggplot(cor_df, aes(x = Var1, y = Var2, fill = Correlation)) +
    geom_tile() +
    scale_fill_gradient2(
      low = "blue",
      mid = "white",
      high = "red",
      midpoint = 0,
      limits = c(-1, 1)
    ) +
    geom_text(aes(label = sprintf("%.2f", Correlation)), size = 6) +
    labs(
      title = get_translation("correlation_plot_title", "influence_analysis", lang),
      x = get_translation("axis_variable", "influence_analysis", lang),
      y = get_translation("axis_variable", "influence_analysis", lang),
      fill = get_translation("correlation", "influence_analysis", lang)
    ) +
    apply_common_theme(rotate_x_labels = TRUE)
}

#' @title 観覧意図への影響要因の統計分析
#' @description
#' 映画の印象評価データを用いて、観覧意図（Q5）に対する
#' 各評価項目（Q1–Q4）の影響度を分析します。
#'
#' @param data 分析用データフレーム（Q1–Q5の列を含む）
#' @param subset_country 国別分析を行う場合の対象国名
#' @param lang 出力言語設定（"ja"：日本語、"en"：英語）
#' @return list 以下の要素を含む分析結果のリスト
#' \describe{
#'   \item{influence_summary}{各要因の影響度サマリー}
```

```
#'   \item{model_fit}{モデルの適合度指標}
#'   \item{model}{回帰分析モデル}
#'   \item{correlation_matrix}{相関行列}
#' }
#' @details
#' 分析手順：
#' 1. データの妥当性検証
#' 2. 前処理（欠損値の除去、スケーリング）
#' 3. 重回帰分析の実行
#' 4. 標準化係数の算出
#' 5. 相関分析
#' 6. VIF（多重共線性）の確認
analyze_viewing_intention <- function(data, subset_country = NULL, lang = "en") {
  message("入力データの検証を開始")
  if (is.null(data)) {
    warning(get_error_message("no_valid_data", lang))
    return(NULL)
  }

  message("データの前処理を開始")
  model_data <- tryCatch(
    {
      if (!is.null(subset_country)) {
        if (!"Country" %in% names(data)) {
          warning(get_error_message("missing_country_column", lang))
          return(NULL)
        }
        data <- data %>% filter(Country == subset_country)
      }

      data %>% select(all_of(paste0("Q", 1:5)))
    },
    error = function(e) {
      warning(sprintf(get_error_message("data_processing_error", lang), e$message))
      return(NULL)
    }
  )

  if (is.null(model_data) || nrow(model_data) == 0) {
    warning(get_error_message("no_valid_data", lang))
    return(NULL)
  }
```

```
# 標準化データの作成
message("標準化分析を開始")
scaled_data <- scale(model_data)
colnames(scaled_data) <- colnames(model_data)

# 重回帰分析の実行
message("重回帰分析を開始")
model <- tryCatch(
  {
    lm(Q5 ~ Q1 + Q2 + Q3 + Q4, data = model_data)
  },
  error = function(e) {
    warning(sprintf(get_error_message("model_error", lang), e$message))
    return(NULL)
  }
)

if (is.null(model)) {
  return(NULL)
}

# 標準化データでの重回帰分析
scaled_model <- lm(scaled_data[, "Q5"] ~ scaled_data[, c("Q1", "Q2", "Q3", "Q4")])

# 相関行列の計算
message("相関行列の計算を開始")
cor_matrix <- cor(model_data)

# VIFの計算
message("VIFの計算を開始")
vif_values <- car::vif(model)

# 影響度の分析結果をまとめる
message("影響度の分析結果を作成")
influence_summary <- tryCatch(
  {
    var_names <- c(
      get_translation("excitement", "influence_analysis", lang),
      get_translation("anger_joy", "influence_analysis", lang),
      get_translation("joy", "influence_analysis", lang),
      get_translation("anger", "influence_analysis", lang)
    )
```

```
    coef_table <- summary(model)$coefficients[-1, ]
    scaled_coef <- coef(scaled_model)[-1]

    result_df <- tibble(
      Variable = var_names,
      Estimate = unname(coef_table[, "Estimate"]),
      Std_Estimate = unname(scaled_coef),
      Std_Error = unname(coef_table[, "Std. Error"]),
      t_value = unname(coef_table[, "t value"]),
      p_value = unname(coef_table[, "Pr(>|t|)"]),
      VIF = unname(vif_values),
      Correlation = unname(cor_matrix[1:4, "Q5"])
    )

    # message("作成された影響度データフレーム:", toString(result_df))
    result_df
  },
  error = function(e) {
    warning(sprintf(get_error_message("influence_summary_error", lang), e$message))
    return(NULL)
  }
)

if (is.null(influence_summary)) {
  return(NULL)
}

# モデル適合度の計算
message("モデル適合度の計算を開始")
model_fit <- tibble(
  R_squared = summary(model)$r.squared,
  Adj_R_squared = summary(model)$adj.r.squared,
  F_statistic = summary(model)$fstatistic[1],
  p_value = pf(
    summary(model)$fstatistic[1],
    summary(model)$fstatistic[2],
    summary(model)$fstatistic[3],
    lower.tail = FALSE
  )
)

message("分析が完了しました")
```

```
  list(
    influence_summary = influence_summary,
    model_fit = model_fit,
    model = model,
    correlation_matrix = cor_matrix
  )
}
```

A.5 utils.R

Listing A.5 utils.R

```
# utils.R

#' @title 環境設定の初期化
#' @description システムの環境設定、ロケール、グラフィックデバイスの初期化を行う
#' @param verbose デバッグ情報の表示フラグ
#' @param install_packages 不足パッケージの自動インストールフラグ
#' @param force_cairo Cairoデバイスの強制使用フラグ
#' @return list 設定結果を含むリスト
#'   \item{success}{bool 全体の成功状態}
#'   \item{locale}{character 設定されたロケール}
#'   \item{graphics_device}{character 設定されたグラフィックデバイス}
#'   \item{messages}{character vector 処理中のメッセージ}
initialize_environment <- function(verbose = FALSE,
install_packages = TRUE,
force_cairo = FALSE) {
    messages <- character(0)
    add_message <- function(msg) {
        messages <<- c(messages, msg)
        if (verbose) message(msg)
      }

    # 結果の初期化
    result <- list(
    success = TRUE,
    locale = NULL,
    graphics_device = NULL,
    messages = NULL
    )

    # 現在のロケールを確認
    current_locale <- Sys.getlocale()
    add_message(sprintf("現在のロケール設定: %s", current_locale))
```

```
# ロケールの設定
tryCatch(
{
    if (.Platform$OS.type == "windows") {
        locale_options <- c("Japanese", "Japanese_Japan.UTF-8")
      } else {
        locale_options <- c("ja_JP.UTF-8", "ja_JP.utf8")
      }

    locale_set <- FALSE
    for (locale in locale_options) {
        tryCatch(
        {
            Sys.setlocale("LC_ALL", locale)
            add_message(sprintf("ロケールを␣%s␣に設定しました", locale))
            locale_set <- TRUE
            result$locale <- locale
            break
          },
        error = function(e) {
            add_message(sprintf("%s␣ロケールの設定に失敗しました", locale))
          }
        )
      }

    if (!locale_set) {
        result$success <- FALSE
        add_message("利用可能な日本語ロケールの設定に失敗しました")
      }
  },
error = function(e) {
    result$success <- FALSE
    add_message(sprintf("ロケール設定中にエラーが発生:␣%s", e$message))
  }
)

# グラフィックデバイスの設定
if (force_cairo || capabilities("cairo")) {
    # Cairoパッケージの確認とインストール
    if (!requireNamespace("Cairo", quietly = TRUE)) {
        if (install_packages) {
            add_message("Cairoパッケージをインストールします")
```

```
                tryCatch(
                {
                    install.packages("Cairo")
                    library(Cairo)
                    add_message("Cairoパッケージのインストールと読み込みに成功しました")
                },
                error = function(e) {
                    result$success <- FALSE
                    add_message(sprintf("Cairoパッケージのインストールに失敗:␣%s", e$me
                    return()
                }
                )
            } else {
                add_message("Cairoパッケージが必要です")
                result$success <- FALSE
                return()
            }
        }

        # グラフィックデバイスの設定
        options(device = "cairo")
        options(bitmapType = "cairo")
        result$graphics_device <- "cairo"
        add_message("Cairoグラフィックデバイスを設定しました")
    } else {
        result$graphics_device <- getOption("device")
        add_message("デフォルトのグラフィックデバイスを使用します")
    }

    result$messages <- messages
    return(result)
}

#' @title データの読み込みと前処理の改善版
#' @param file_path データファイルのパス
#' @param lang 言語コード
#' @return データフレームのリスト（raw と long 形式）
load_and_preprocess_data <- function(file_path, lang = "ja") {
    # データの読み込み
    tryCatch(
    {
        # readr::read_csv を使用してデータを読み込む
        data <- read_csv(file_path,
```

```
col_types = cols(
ID = col_character(),
Q1 = col_double(),
Q2 = col_double(),
Q3 = col_double(),
Q4 = col_double(),
Q5 = col_double(),
Country = col_character()
)
)

# データの検証
if (nrow(data) == 0) {
    stop("Empty dataset")
  }

required_cols <- c("ID", paste0("Q", 1:5), "Country")
missing_cols <- setdiff(required_cols, names(data))
if (length(missing_cols) > 0) {
    stop(sprintf(
    "Missing required columns: %s",
    paste(missing_cols, collapse = ", ")
    ))
  }

# 長形式データの作成
data_long <- data %>%
pivot_longer(
cols = starts_with("Q"),
names_to = "question",
values_to = "value"
) %>%
mutate(
question = factor(
question,
levels = paste0("Q", 1:5),
labels = sapply(paste0("Q", 1:5), function(q) {
    get_translation(q, "questions", lang)
  })
)
)

return(list(
```

```
        raw = data,
        long = data_long
        ))
      },
    error = function(e) {
        message(sprintf("Error in data loading: %s", e$message))
        return(NULL)
      }
    )
  }

#' @title 結果の保存
#' @description データフレームやプロットを各種形式で保存する統合関数
#' @param data 保存するデータ (データフレームまたはggplotオブジェクト)
#' @param file_path ファイルパス
#' @param type オプションで指定する保存形式 ("csv", "pdf", "svg")。指定しない場合は拡張
#' @return invisible(TRUE) if successful, invisible(FALSE) if failed
#' @examples
#' # データフレームをCSVとして保存
#' df <- data.frame(x = 1:3, y = letters[1:3])
#' save_results(df, "output/data.csv")
#'
#' # ggplotをPDFとして保存
#' p <- ggplot(df, aes(x, y)) +
#'   geom_point()
#' save_results(p, "output/plot.pdf")
save_results <- function(data, file_path, type = NULL) {
    # ディレクトリの作成
    dir.create(dirname(file_path), recursive = TRUE, showWarnings = FALSE)

    # ファイル形式の判定
    if (is.null(type)) {
        type <- tolower(tools::file_ext(file_path))
      }

    tryCatch(
    {
        if (type == "csv") {
            # CSVファイルの保存
            if (!is.data.frame(data)) {
                stop("CSVファイルの保存にはデータフレームが必要です")
              }
            write_csv(data, file_path)
```

```
      } else if (type %in% c("pdf", "svg")) {
      # プロットの保存
      if (!inherits(data, "ggplot")) {
        stop("PDF/SVGファイルの保存にはggplotオブジェクトが必要です")
      }

      # デバイスの設定
      device <- if (type == "pdf") cairo_pdf else "svg"

      # プロットの保存
      ggsave(
      filename = file_path,
      plot = data,
      device = device,
      width = 10,
      height = 7,
      units = "in"
      )
    } else {
      stop(sprintf("未対応のファイル形式です: %s", type))
    }

    invisible(TRUE)
  },
error = function(e) {
    warning(sprintf("ファイル保存でエラーが発生: %s", e$message))
    invisible(FALSE)
  }
)
}

#' @title プロットの共通テーマを設定する関数
#' @description ggplotオブジェクトに対して、統一されたテーマ設定を適用します
#' @param p ggplotオブジェクト
#' @param base_size ベースとなるフォントサイズ（デフォルト：14）
#' @param rotate_x_labels X軸ラベルを45度回転するかどうか（デフォルト：FALSE）
#' @return テーマが適用されたggplotオブジェクト
#' @export
apply_common_theme <- function(base_size = 14, rotate_x_labels = FALSE) {
    common_theme <- theme(
    text = element_text(family = "sans"),
    plot.title = element_text(size = rel(1.2), hjust = 0.5),
    axis.title = element_text(size = rel(1.1)),
```

```
    axis.text = element_text(size = rel(1.0)),
    axis.text.x = if (rotate_x_labels) element_text(angle = 45, hjust = 1) else element_
    legend.title = element_text(size = rel(1.0)),
    legend.text = element_text(size = rel(0.9)),
    panel.grid.minor = element_blank(),
    plot.margin = margin(t = 10, r = 10, b = 10, l = 10),
    legend.position = "right"
    )
    theme_minimal(base_size = base_size) + common_theme
  }
```

A.6 translation_config.R

Listing A.6 translation_config.R

```
#' @title 映画印象分析システムの多言語設定
#' @description
#' 映画の印象評価データの分析に使用する多言語設定を管理するモジュール。
#' システムメッセージ、質問項目、プロットラベル、統計量など、
#' 分析に必要な全ての文字列を日本語と英語で管理します。
#'
#' @author データサイエンス学部
#' @version 1.1.0
#' @date 2025-01-15

library(yaml)

# システムのデフォルトフォント設定
# OSに応じて適切なフォントファミリーを選択します
DEFAULT_FONT_FAMILY <- list(
  "ja" = ifelse(
    Sys.info()["sysname"] == "Windows",
    "MS Gothic",
    "IPAGothic"
  ),
  "en" = ifelse(
    Sys.info()["sysname"] == "Windows",
    "Arial",
    "DejaVu Sans"
  )
)

#' @section 翻訳データの構造:
#' 翻訳データは以下のカテゴリーで構成されています：
```

```
#' - system_messages: システム通知やエラーメッセージ
#' - questions: 評価質問項目
#' - plot_labels: グラフやプロットのラベル
#' - stat_columns: 統計情報の列名
#' - cluster_labels: クラスター分析関連のラベル
#' - tabs: UIタブのラベル
#' - emotion_categories: 感情カテゴリーの名称
#' - influence_analysis: 影響分析関連のラベル

# 多言語翻訳設定
# 各言語コードをキーとして、その言語での表示文字列を定義します
TRANSLATIONS <- list(
  "ja" = list(
    # システムメッセージ（エラーや通知など）
    "system_messages" = list(
      "analysis_complete" = "分析が完了しました。出力ファイルは␣'output/ja'␣ディレクトリ
      "error_no_data" = "データファイルが見つかりません。",
      "error_invalid_format" = "データ形式が不正です。",
      "time_series_analysis_start" = "時系列分析を開始します...",
      "time_series_analysis_end" = "時系列分析が完了しました。",
      "data_processing_error" = "データ処理中にエラーが発生しました:␣%s",
      "no_valid_data" = "有効なデータが見つかりませんでした。",
      "analysis_failed" = "分析の実行に失敗しました。",
      "missing_country_column" = "国別分析に必要な'Country'列がデータに存在しません。",
      "model_error" = "モデルの推定中にエラーが発生しました:␣%s",
      "plot_creation_error" = "グラフの作成中にエラーが発生しました:␣%s",
      "visualization_error" = "分析結果の可視化中にエラーが発生しました:␣%s",
      "save_error" = "ファイルの保存中にエラーが発生しました:␣%s",
      "influence_summary_error" = "影響度分析の結果作成中にエラーが発生しました:␣%s"
    ),

    # 質問項目（アンケートの各設問）
    "questions" = list(
      "Q1" = "ワクワク感",
      "Q2" = "怒り−喜び",
      "Q3" = "喜び",
      "Q4" = "怒り",
      "Q5" = "観覧意図"
    ),

    # プロットのラベル（グラフの題名や軸ラベルなど）
    "plot_labels" = list(
      "main_title" = "映画の印象評価（VAS）",
```

```
    "cluster_title" = "クラスター分析結果",
    "x_label" = "質問項目",
    "y_label" = "評価値",
    "pc1_label" = "第1主成分",
    "pc2_label" = "第2主成分",
    "time_series_title" = "時系列変化",
    "distribution" = "分布",
    "boxplot" = "箱ひげ図",
    "overall_prefix" = "全体␣–␣",
    "japan_prefix" = "日本␣–␣",
    "singapore_prefix" = "シンガポール␣–␣",
    "country" = "国",
    "country_comparison" = "国別比較分析"
  ),

  # 統計量の列名（基本統計量や検定結果など）
  "stat_columns" = list(
    "sample_size" = "サンプルサイズ",
    "mean" = "平均値",
    "sd" = "標準偏差",
    "median" = "中央値",
    "q1" = "第1四分位数",
    "q3" = "第3四分位数",
    "min" = "最小値",
    "max" = "最大値",
    "stats" = "統計情報",
    "cluster_stats" = "クラスター統計",
    "t_stat" = "t値",
    "p_value" = "p値",
    "mean_diff" = "平均値の差",
    "conf_low" = "信頼区間下限",
    "conf_high" = "信頼区間上限"
  ),

  # クラスター関連のラベル
  "cluster_labels" = list(
    "cluster" = "クラスター",
    "high_eval" = "高評価群",
    "mid_eval" = "中評価群",
    "low_eval" = "低評価群",
    "mixed_eval" = "混合群",
    "cluster_stats" = "クラスター統計情報",
    "cluster_analysis" = "クラスター分析",
```

```
    "membership" = "メンバーシップ度"
  ),

  # タブ関連のラベル
  "tabs" = list(
    "overview" = "概要",
    "stats" = "統計情報",
    "clusters" = "クラスター分析"
  ),

  # 感情カテゴリーのラベル
  "emotion_categories" = list(
    "excitement" = "ワクワク感",
    "anger_joy" = "喜び−怒り",
    "joy" = "喜び",
    "anger" = "怒り",
    "intention" = "観覧意図"
  ),

  # 影響分析関連のラベル
  "influence_analysis" = list(
    "influence_title" = "観覧意図への影響分析",
    "variable" = "評価項目",
    "estimate" = "推定値",
    "std_estimate" = "標準化係数",
    "std_error" = "標準誤差",
    "t_value" = "t値",
    "p_value" = "p値",
    "vif" = "VIF",
    "correlation" = "相関係数",
    "significant" = "有意",
    "effect_size" = "効果量",
    "effect_large" = "大",
    "effect_medium" = "中",
    "effect_small" = "小",
    "model_fit" = "モデル適合度",
    "r_squared" = "決定係数",
    "adj_r_squared" = "自由度調整済み決定係数",
    "f_statistic" = "F統計量",
    "coefficient_plot_title" = "観覧意図への影響度（標準化係数）",
    "coefficient_plot_subtitle" = "国別分析：%s",
    "correlation_plot_title" = "相関行列ヒートマップ",
    "correlation_plot_subtitle" = "国別分析：%s",
```

```
      "axis_variable" = "評価項目",
      "axis_std_coefficient" = "標準化係数",
      "excitement" = "ワクワク感 (Q1)",
      "anger_joy" = "怒り−喜び (Q2)",
      "joy" = "喜び (Q3)",
      "anger" = "怒り (Q4)",
      "intention" = "観覧意図 (Q5)"
    )
  ),
  "en" = list(
    # System Messages
    "system_messages" = list(
      "analysis_complete" = "Analysis completed. Output files are saved in 'output/en' d
      "error_no_data" = "Data file not found.",
      "error_invalid_format" = "Invalid data format.",
      "time_series_analysis_start" = "Starting time series analysis...",
      "time_series_analysis_end" = "Time series analysis completed.",
      "data_processing_error" = "Error occurred during data processing: %s",
      "no_valid_data" = "No valid data found.",
      "analysis_failed" = "Analysis execution failed.",
      "missing_country_column" = "Country column required for country-specific analysis
      "model_error" = "Error occurred during model estimation: %s",
      "plot_creation_error" = "Error occurred while creating plot: %s",
      "visualization_error" = "Error occurred while visualizing analysis results: %s",
      "save_error" = "Error occurred while saving file: %s",
      "influence_summary_error" = "Error occurred while creating influence analysis summ
    ),

    # Questions
    "questions" = list(
      "Q1" = "Excitement",
      "Q2" = "Anger-Joy",
      "Q3" = "Joy",
      "Q4" = "Anger",
      "Q5" = "Viewing Intention"
    ),

    # Plot Labels
    "plot_labels" = list(
      "main_title" = "Movie Impression Evaluation (VAS)",
      "cluster_title" = "Clustering Results",
      "x_label" = "Questions",
      "y_label" = "Value",
```

```
    "pc1_label" = "First␣Principal␣Component",
    "pc2_label" = "Second␣Principal␣Component",
    "time_series_title" = "Time␣Series␣Change",
    "distribution" = "Distribution",
    "boxplot" = "Box␣Plot",
    "overall_prefix" = "Overall␣-␣",
    "japan_prefix" = "Japan␣-␣",
    "singapore_prefix" = "Singapore␣-␣",
    "country" = "Country",
    "country_comparison" = "Country␣Comparison␣Analysis"
  ),

  # Statistical Columns
  "stat_columns" = list(
    "sample_size" = "Sample␣Size",
    "mean" = "Mean",
    "sd" = "Standard␣Deviation",
    "median" = "Median",
    "q1" = "1st␣Quartile",
    "q3" = "3rd␣Quartile",
    "min" = "Minimum",
    "max" = "Maximum",
    "stats" = "Statistics",
    "cluster_stats" = "Cluster␣Statistics",
    "t_stat" = "t-statistic",
    "p_value" = "p-value",
    "mean_diff" = "Mean␣Difference",
    "conf_low" = "CI␣Lower",
    "conf_high" = "CI␣Upper"
  ),

  # Cluster Labels
  "cluster_labels" = list(
    "cluster" = "Cluster",
    "high_eval" = "High␣Evaluation",
    "mid_eval" = "Medium␣Evaluation",
    "low_eval" = "Low␣Evaluation",
    "mixed_eval" = "Mixed␣Evaluation",
    "cluster_stats" = "Cluster␣Statistics",
    "cluster_analysis" = "Cluster␣Analysis",
    "membership" = "Membership␣Degree"
  ),
```

```
  # Tabs
  "tabs" = list(
    "overview" = "Overview",
    "stats" = "Statistics",
    "clusters" = "Cluster␣Analysis"
  ),

  # Emotion Categories
  "emotion_categories" = list(
    "excitement" = "Excitement",
    "anger_joy" = "Anger_Joy",
    "joy" = "Joy",
    "anger" = "Anger",
    "intention" = "Viewing␣Intention"
  ),

  # Influence Analysis
  "influence_analysis" = list(
    "influence_title" = "Analysis␣of␣Viewing␣Intention",
    "variable" = "Variable",
    "estimate" = "Estimate",
    "std_estimate" = "Standardized␣Coefficient",
    "std_error" = "Std.␣Error",
    "t_value" = "t-value",
    "p_value" = "p-value",
    "vif" = "VIF",
    "correlation" = "Correlation",
    "significant" = "Significant",
    "effect_size" = "Effect␣Size",
    "effect_large" = "Large",
    "effect_medium" = "Medium",
    "effect_small" = "Small",
    "model_fit" = "Model␣Fit",
    "r_squared" = "R-squared",
    "adj_r_squared" = "Adjusted␣R-squared",
    "f_statistic" = "F-statistic",
    "coefficient_plot_title" = "Impact␣on␣Viewing␣Intention␣(Standardized␣Coefficients
    "coefficient_plot_subtitle" = "Country␣Analysis:␣%s",
    "correlation_plot_title" = "Correlation␣Matrix␣Heatmap",
    "correlation_plot_subtitle" = "Country␣Analysis:␣%s",
    "axis_variable" = "Variables",
    "axis_std_coefficient" = "Standardized␣Coefficient",
    "excitement" = "Excitement␣(Q1)",
```

```
        "anger_joy" = "Anger–Joy␣(Q2)",
        "joy" = "Joy␣(Q3)",
        "anger" = "Anger␣(Q4)",
        "intention" = "Viewing␣Intention␣(Q5)"
      )
    )
  )

#' @title 翻訳テキストの取得
#' @description
#' 指定された言語、カテゴリー、キーに対応する翻訳テキストを取得します。
#'
#' @param key 翻訳キー (例: "Q1", "main_title" など)
#' @param category カテゴリー (例: "questions", "plot_labels" など)
#' @param lang 言語コード ("ja" または "en")
#' @return character 翻訳されたテキスト
#' @examples
#' # 日本語の質問1を取得
#' text_ja <- get_translation("Q1", "questions", "ja")
#'
#' # 英語のメインタイトルを取得
#' text_en <- get_translation("main_title", "plot_labels", "en")
#'
#' @export
get_translation <- function(key, category, lang = getOption("movie_analysis.lang", "ja")
  if (!lang %in% names(TRANSLATIONS)) {
    stop(sprintf("Unsupported␣language:␣%s", lang))
  }
  if (!category %in% names(TRANSLATIONS[[lang]])) {
    stop(sprintf("Unknown␣category:␣%s", category))
  }
  if (!key %in% names(TRANSLATIONS[[lang]][[category]])) {
    stop(sprintf("Unknown␣key:␣%s␣in␣category:␣%s", key, category))
  }
  TRANSLATIONS[[lang]][[category]][[key]]
}

#' @title 言語設定の変更
#' @description
#' システム全体の表示言語を変更します。
#'
#' @param lang 言語コード ("ja" または "en")
#' @return NULL
```

```
#' @examples
#' # 日本語に設定
#' set_language("ja")
#'
#' # 英語に設定
#' set_language("en")
#'
#' @export
set_language <- function(lang = "ja") {
  if (!lang %in% names(TRANSLATIONS)) {
    stop(sprintf("Unsupported language: %s", lang))
  }
  options(movie_analysis.lang = lang)
}

#' @title エラーメッセージの取得
#' @param error_key エラーキー
#' @param lang 言語コード
#' @return character エラーメッセージ
get_error_message <- function(error_key, lang = getOption("movie_analysis.lang", "ja")) {
  tryCatch(
    {
      get_translation(error_key, "system_messages", lang)
    },
    error = function(e) {
      # 翻訳が見つからない場合はエラーキーをそのまま返す
      warning(sprintf("Error message not found for key: %s", error_key))
      error_key
    }
  )
}
```

A.7 create_sample.R

Listing A.7 create_sample.R

```
# パッケージの読み込み
suppressPackageStartupMessages({
  library(tidyverse)
})

#' @title サンプルデータの生成
#' @description クラスタリング結果を反映したサンプルデータを生成します
#' @param n_samples_per_country 各国のサンプル数（デフォルト：30）
```

```
#' @return 生成されたサンプルデータのデータフレーム
create_sample_data <- function(n_samples_per_country = 30) {
  # クラスターの特性を定義
  clusters <- list(
    high_eval = list(
      mean = 0.8,
      sd = 0.1,
      weight = 0.3
    ),
    mid_eval = list(
      mean = 0.6,
      sd = 0.1,
      weight = 0.3
    ),
    mixed_eval = list(
      mean = 0.5,
      sd = 0.2,
      weight = 0.2
    ),
    low_eval = list(
      mean = 0.3,
      sd = 0.1,
      weight = 0.2
    )
  )

  # 国ごとのデータ生成
  generate_country_data <- function(country, n_samples) {
    # クラスターの割り当て
    cluster_assignments <- sample(
      names(clusters),
      size = n_samples,
      prob = sapply(clusters, function(x) x$weight),
      replace = TRUE
    )

    # データフレームの作成
    data <- map_dfr(1:n_samples, function(i) {
      cluster <- clusters[[cluster_assignments[i]]]

      # 各質問の値を生成
      values <- pmin(pmax(rnorm(5, mean = cluster$mean, sd = cluster$sd), 0), 1)
```

```
      tibble(
        ID = sprintf("%s_%03d", country, i),
        Q1 = values[1],
        Q2 = values[2],
        Q3 = values[3],
        Q4 = values[4],
        Q5 = values[5],
        Country = country
      )
    })

    return(data)
  }

  # 日本とシンガポールのデータを生成
  data_japan <- generate_country_data("Japan", n_samples_per_country)
  data_singapore <- generate_country_data("Singapore", n_samples_per_country)

  # データを結合
  data_all <- bind_rows(data_japan, data_singapore)

  return(data_all)
}

# メイン処理
main <- function() {
  # 乱数のシードを設定
  set.seed(123)

  # サンプルデータの生成（各国30サンプル）
  sample_data <- create_sample_data(30)

  # CSVファイルとして保存
  write_csv(sample_data, "sample_data.csv")

  # サマリー統計量の表示
  cat("\n=== サンプルデータの基本統計量 ===\n")
  print(summary(sample_data[, c("Q1", "Q2", "Q3", "Q4", "Q5")]))

  # 国別サンプル数の確認
  cat("\n=== 国別サンプル数 ===\n")
  print(table(sample_data$Country))
}
```

```
# スクリプトの実行
main()
```

付録 B

サンプルデータ

B.1　サンプルデータ

ID	Q1	Q2	Q3	Q4	Q5	Country
Japan_001	0.979	0.800	0.603	0.870	0.753	Japan
Japan_002	0.300	0.456	0.295	0.354	0.375	Japan
Japan_003	0.431	0.684	0.615	0.486	0.725	Japan
Japan_004	0.343	0.270	0.390	0.388	0.382	Japan
Japan_005	0.369	0.355	0.294	0.269	0.262	Japan
Japan_006	0.731	0.779	0.673	1.000	0.921	Japan
Japan_007	0.488	0.560	0.553	0.678	0.592	Japan
Japan_008	0.325	0.297	0.296	0.437	0.277	Japan
Japan_009	0.752	0.445	0.658	0.612	0.622	Japan
Japan_010	0.638	0.550	0.567	0.498	0.493	Japan
Japan_011	0.330	0.345	0.305	0.392	0.505	Japan
Japan_012	0.551	0.369	0.701	0.529	0.531	Japan
Japan_013	0.705	0.443	0.256	0.536	0.472	Japan
Japan_014	0.601	0.600	0.563	0.664	0.600	Japan
Japan_015	0.833	0.910	0.844	0.767	0.915	Japan
Japan_016	0.399	0.400	0.324	0.237	0.436	Japan
Japan_017	0.740	1.000	1.000	0.776	0.697	Japan
Japan_018	0.700	0.800	0.775	0.765	0.705	Japan
Japan_019	0.595	0.522	0.433	0.562	0.692	Japan
Japan_020	0.242	0.361	0.138	0.294	0.352	Japan
Japan_021	0.330	0.311	0.236	0.215	0.198	Japan
Japan_022	0.524	0.311	0.402	0.449	0.869	Japan
Japan_023	0.370	0.547	0.516	0.308	0.486	Japan
Japan_024	0.444	0.345	0.304	0.258	0.100	Japan
Japan_025	0.700	0.208	0.648	0.882	0.211	Japan
Japan_026	0.640	0.448	0.186	0.197	0.180	Japan
Japan_027	0.547	0.500	0.669	0.810	0.471	Japan
Japan_028	0.679	0.700	0.633	0.499	0.588	Japan
Japan_029	0.772	0.900	0.763	0.898	0.763	Japan
Japan_030	0.905	0.695	0.674	1.000	0.758	Japan

```
Singapore_001 0.287  0.800  0.430  0.327  0.453  Singapore
Singapore_002 0.780  0.911  0.808  0.875  0.800  Singapore
Singapore_003 0.543  0.435  0.519  0.321  0.238  Singapore
Singapore_004 0.500  0.360  0.175  0.239  0.181  Singapore
Singapore_005 0.820  0.700  0.573  0.654  0.559  Singapore
Singapore_006 0.552  0.521  0.541  0.800  0.600  Singapore
Singapore_007 0.524  0.549  0.746  0.397  0.301  Singapore
Singapore_008 0.800  0.412  0.355  0.253  0.243  Singapore
Singapore_009 0.385  0.624  0.722  0.642  0.427  Singapore
Singapore_010 0.306  0.200  0.228  0.400  0.198  Singapore
Singapore_011 0.796  0.591  0.621  0.526  0.500  Singapore
Singapore_012 0.668  0.782  0.842  0.832  0.722  Singapore
Singapore_013 0.521  0.550  0.750  0.486  0.582  Singapore
Singapore_014 0.990  0.790  0.664  0.734  0.849  Singapore
Singapore_015 0.562  0.544  0.566  0.609  0.760  Singapore
Singapore_016 0.591  0.708  0.663  0.600  0.447  Singapore
Singapore_017 0.248  0.251  0.305  0.430  0.529  Singapore
Singapore_018 0.755  0.587  0.424  0.561  0.609  Singapore
Singapore_019 0.885  0.896  0.868  0.660  0.885  Singapore
Singapore_020 0.411  0.535  0.515  0.586  0.500  Singapore
Singapore_021 0.633  0.874  0.839  0.773  0.812  Singapore
Singapore_022 0.313  0.322  0.464  0.278  0.317  Singapore
Singapore_023 0.717  0.705  0.715  0.542  0.800  Singapore
Singapore_024 0.807  0.987  0.665  0.802  0.925  Singapore
Singapore_025 0.357  0.349  0.312  0.289  0.413  Singapore
Singapore_026 0.333  0.100  0.321  0.424  0.504  Singapore
Singapore_027 0.760  0.651  0.155  0.380  0.430  Singapore
Singapore_028 0.670  0.589  0.474  0.768  0.691  Singapore
Singapore_029 0.624  0.722  0.466  0.700  0.548  Singapore
Singapore_030 0.368  0.294  0.363  0.434  0.301  Singapore
```

付録 C

図表

C.0.1　図表

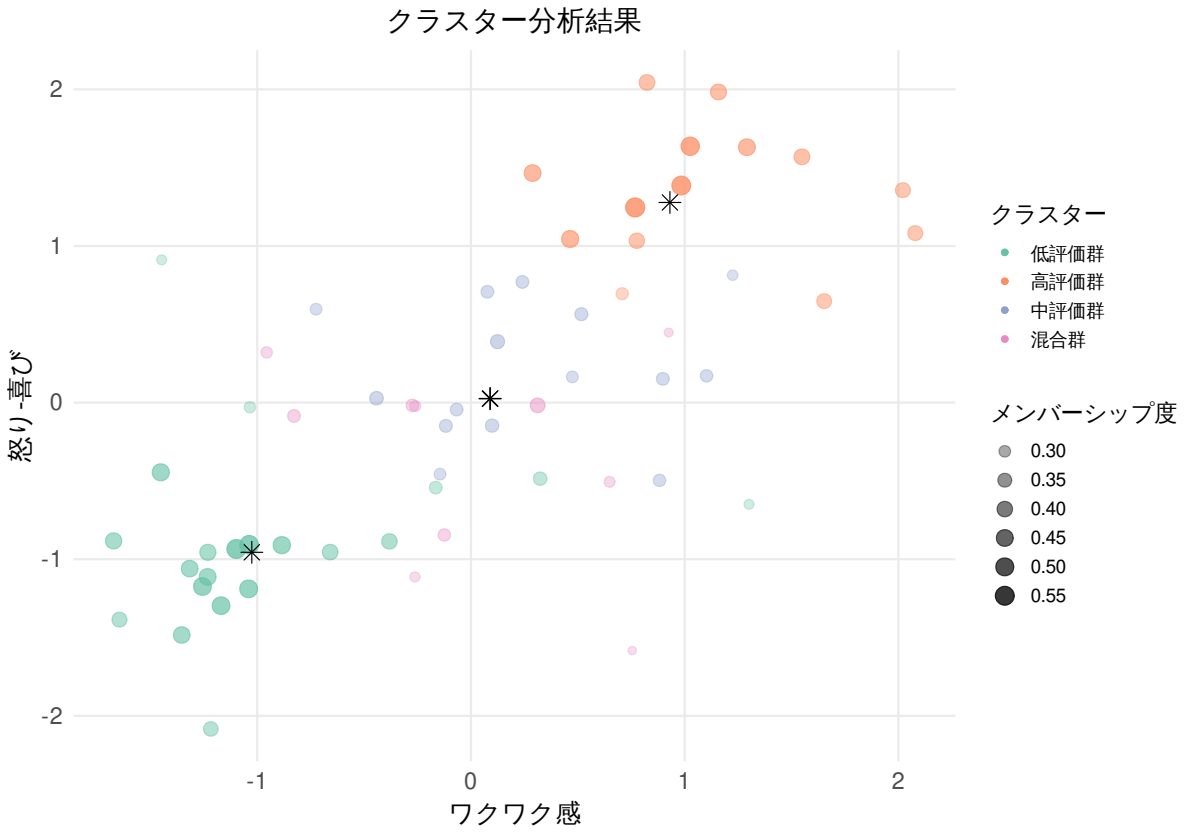

図 C.1　クラスタリング結果

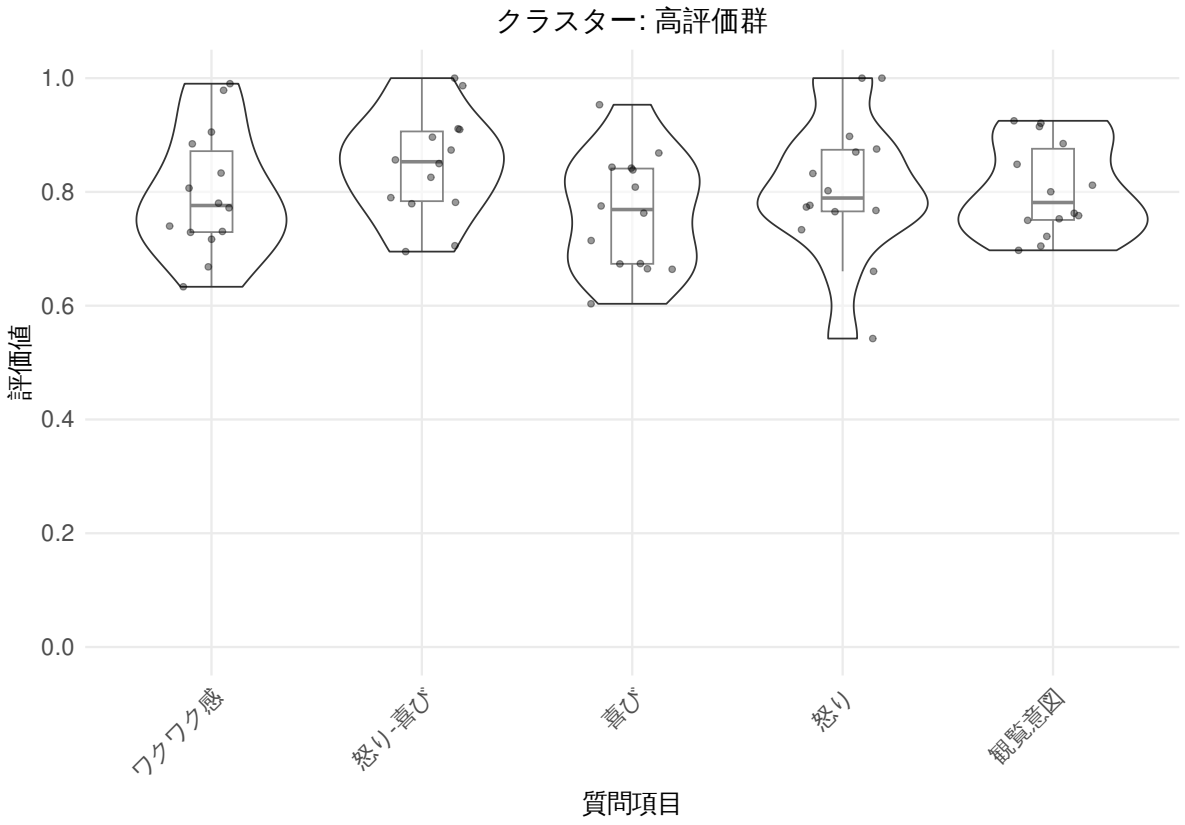

図 C.2　高評価クラスター

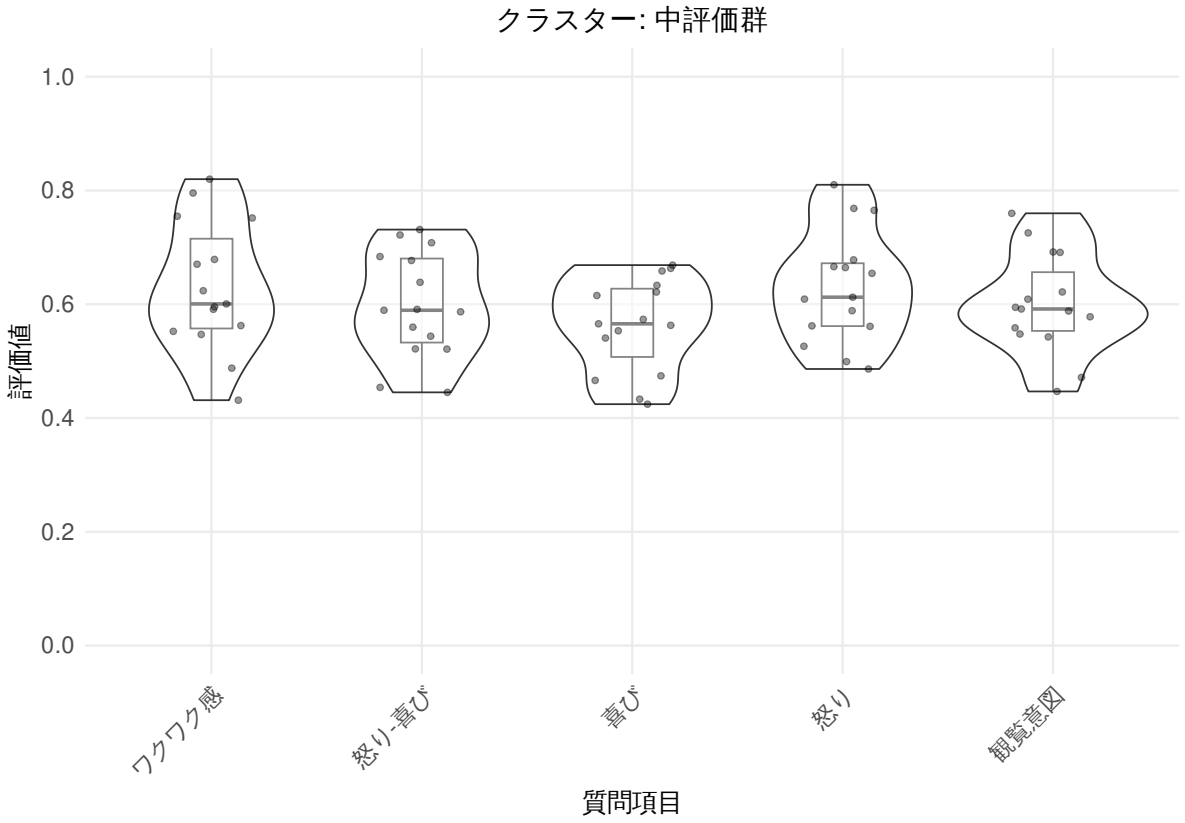

図 C.3　中評価クラスター

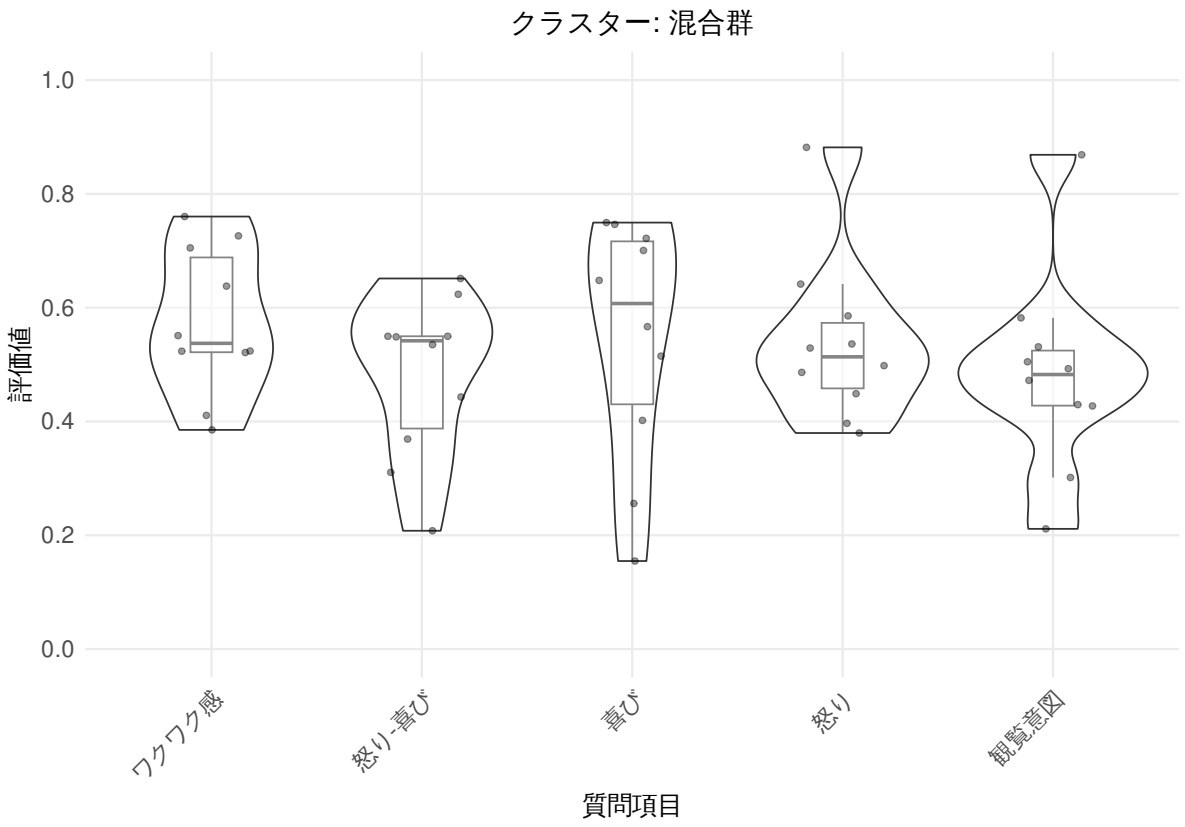

図 C.4　混合群クラスター

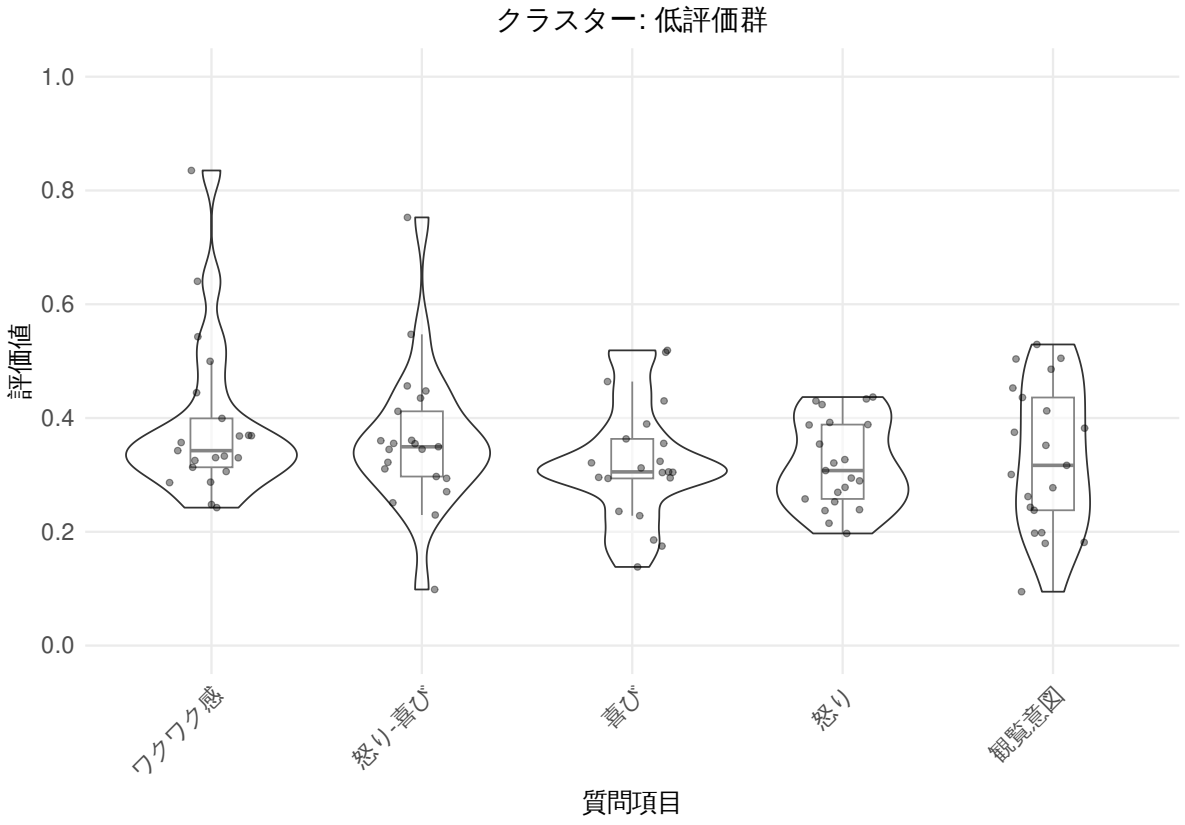

図 C.5　低評価クラスター

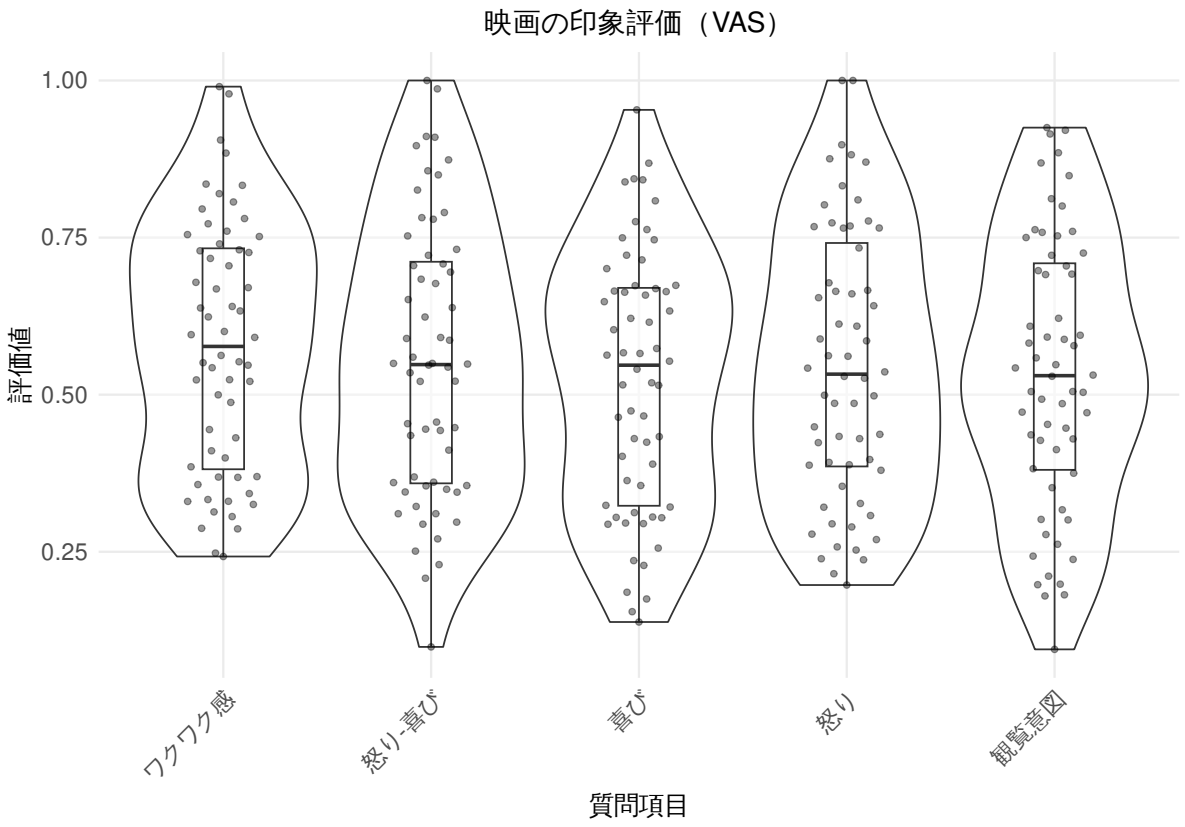

図 C.6 分布 (全体)

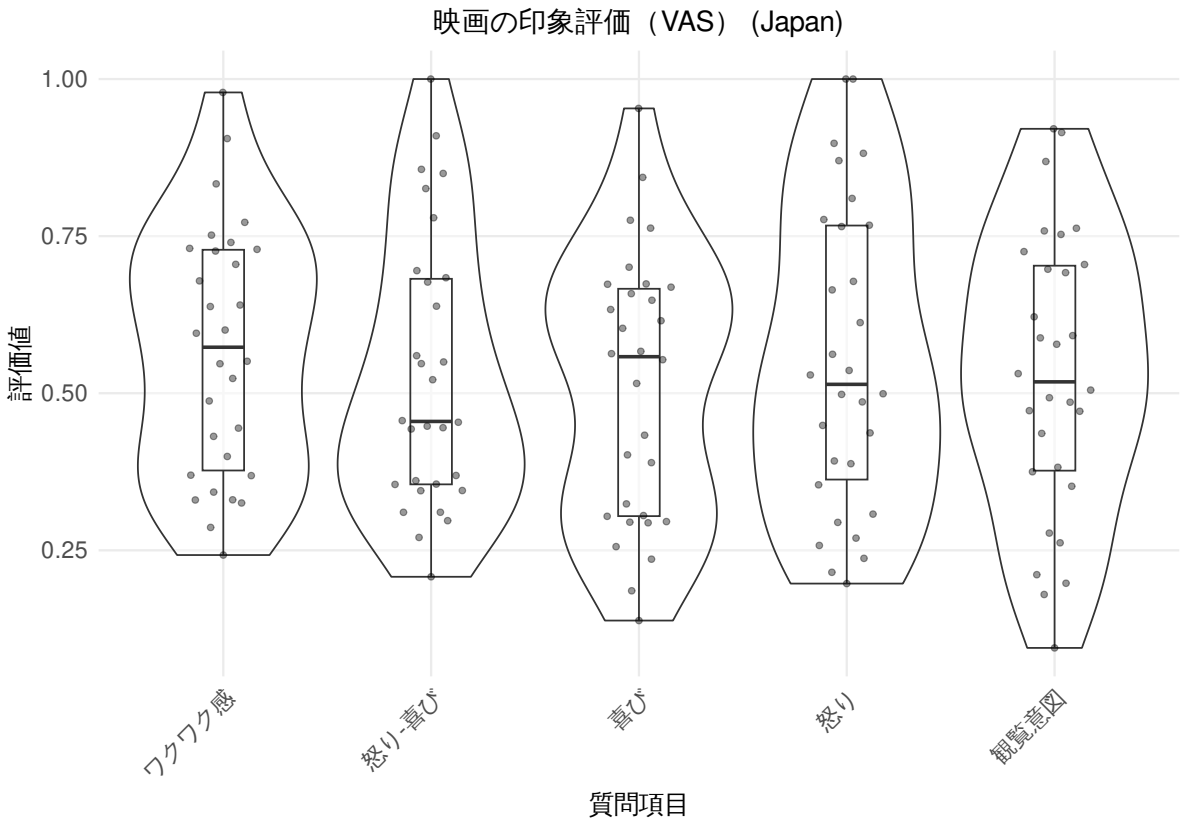

図 C.7 分布 (日本)

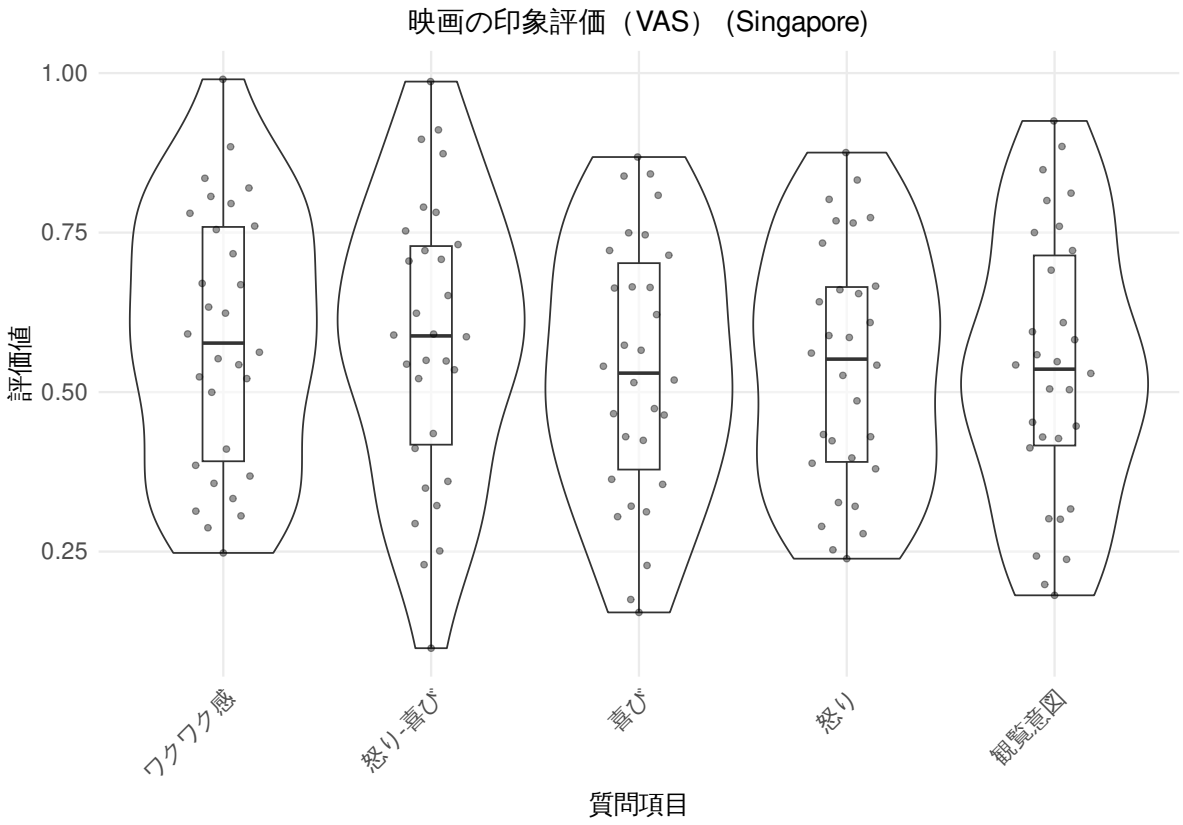

図 C.8　分布 (シンガポール)

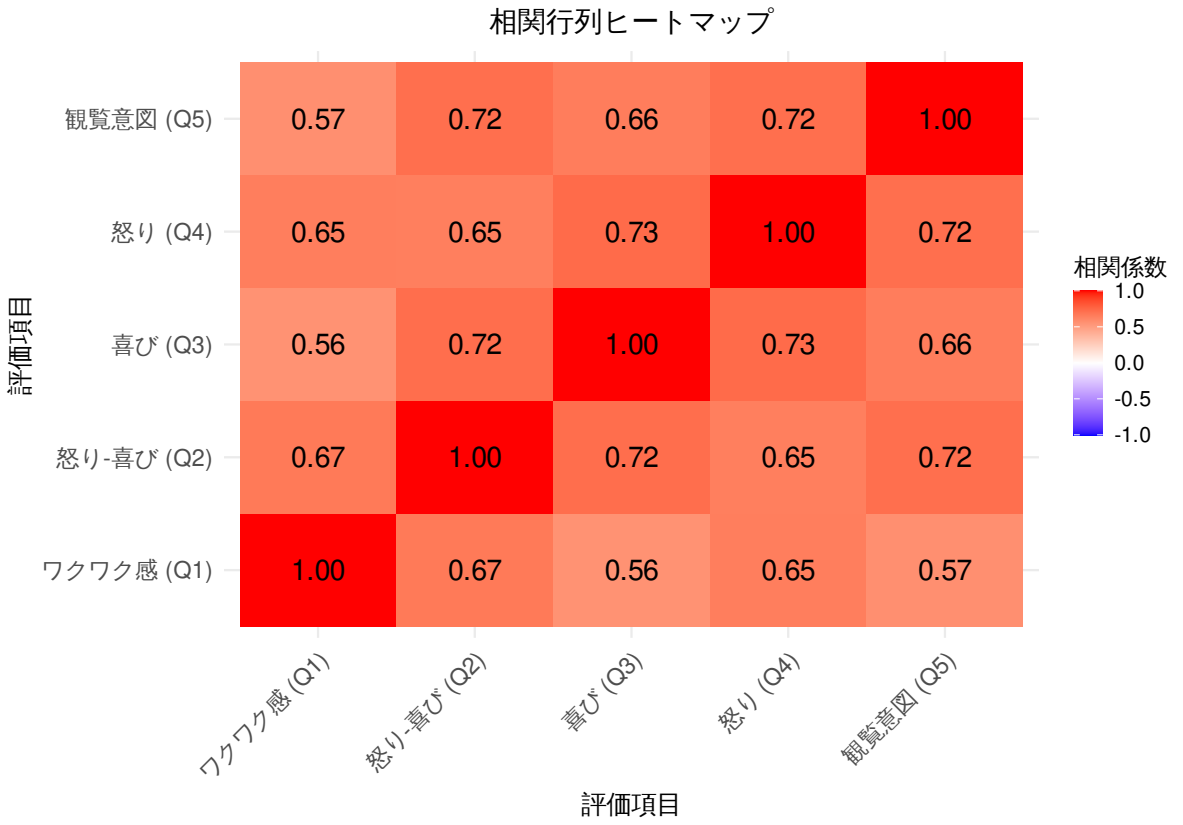

図 C.9　相関行列ヒートマップ

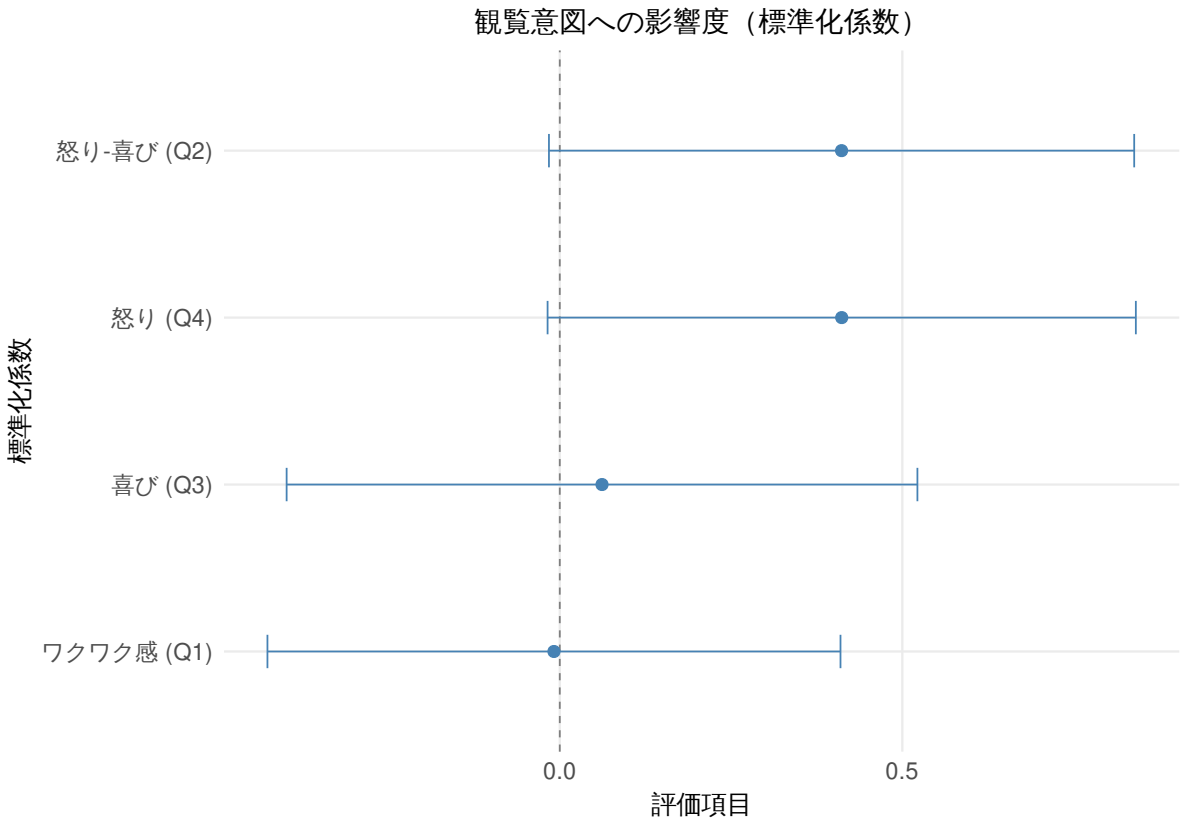

図 C.10　標準化係数

参考文献

[1] Sherman Chan, Nami Imanishi, Satoshi Watanabe, and Naruki Shirahama. A proposal of an experimental method for quantifying impressions using visual analog scale. In *Proceedings of the 9th IIAE International Conference on Intelligent Systems and Image Processing 2022*, pages 46–52. The Institute of Industrial Application Engineers, 1 2022.

[2] Ruiqi Deng and Yifan Gao. A review of eye tracking research on video-based learning. *Education and Information Technologies*, 28:7671–7702, 2023.

[3] Rizal Akbar Fitrianto, Arda Surya Editya, Mochammad Machlul Alamin, Anggay Luri Pramana, and Ahmad Khoir Alhaq. Classification of indonesian sarcasm tweets on x platform using deep learning. In *2024 7th International Conference on Informatics and Computational Sciences (ICICoS)*, pages 388–393, 7 2024.

[4] Stephan Fremerey, Raja Faseeh Uz Zaman, Touseef Ashraf, Rakesh Rao Ramachandra Rao, Steve Göring, and Alexander Raake. Towards evaluation of immersion, visual comfort and exploration behaviour for non-stereoscopic and stereoscopic 360° videos. In *2023 IEEE International Symposium on Multimedia (ISM)*, pages 131–138, 12 2023.

[5] Miguel A García-Pérez and Rocío Alcalá-Quintana. Accuracy and precision of responses to visual analog scales: Inter- and intra-individual variability. *Behavior Research Methods*, 55:4369–4381, 2023.

[6] Petr Hajek and Vladimir Olej. Hierarchical intuitionistic tsk fuzzy system for bitcoin price forecasting. In *2023 IEEE International Conference on Fuzzy Systems (FUZZ)*, pages 1–6, 8 2023.

[7] Tim Kuhlmann, Michael Dantlgraber, and Ulf-Dietrich Reips. Investigating measurement equivalence of visual analogue scales and likert-type scales in internet-based personality questionnaires. *Behavior Research Methods*, 49:2173–2181, 2017.

[8] Surya Kumar and Tanya Sharma. Impact of marketing strategies on consumer buying behaviour with specific reference to movies as a medium. *International Research Journal on Advanced Engineering and Management (IRJAEM)*, 2:248–255, 1 2024.

[9] Tatsuki Makioka, Shogo Okamoto, and Ibuki Tara. Difference in skin conductance dynamics to horror and family bond emotional movies. In *2022 IEEE 11th Global Conference on Consumer Electronics (GCCE)*, pages 444–446, 10 2022.

[10] Mirtill-Boglárka Naghi, Levente Kovács, and László Szilágyi. A review on advanced c-means clustering models based on fuzzy logic. In *2023 IEEE 21st World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 293–298. IEEE, 1 2023.

[11] Hirari Nishikori, Saerom Lee, and Toshihiro Hayashi. Analysis of the relationship between colors and impression on studio ghibli movie posters. In *2024 Joint 13th International Conference on Soft Computing and Intelligent Systems and 25th International Symposium on Advanced Intelligent Systems (SCIS&ISIS)*, pages 1–4, 11 2024.

[12] Naruki Shirahama, Shinichi Kondo, Keiji Matsumoto, Kenji Moriya, Naofumi Nakaya, Kazuhiro Koshi, and Satoshi Watanabe. Quantitative analysis of conversational response nuances using visual analog scale, data visualization, and clustering. In *2024 IEEE/ACIS 22nd International Conference on Software Engineering Research, Management and Applications (SERA)*, pages 362–367, 2024.

[13] Naruki Shirahama, Kaito Murakami, Satoshi Watanabe, Naofumi Nakaya, and Yukio Mori. Subjective evaluation experiment of grayscale color to examine vas measurement method. In *Proceedings of the 7th IIAE International Conference on Intelligent Systems and Image Processing 2019 (ICISIP2019)*, pages 147–153. The Institute of Industrial Application Engineers, 1 2019.

[14] Naruki Shirahama, Naofumi Nakaya, Kazuhiro Koshi, Keiji Matsumoto, Kenji Moriya, and Satoshi Watanabe. Proposal for quantification and analysis method of nuances in conversation responses using visual analog scale. *ICIC Express Letters, Part B: Applications*, 15:71–81, 1 2024.

[15] Naruki Shirahama, Naofumi Nakaya, and Satoshi Watanabe. Comparative study of visual analogue scale and likert scale using chat generation ai. In *The Proceedings of The 11th International Conference on Intelligent Systems and Image Processing 2024*, pages 195–202. The Institute of Industrial Applications Engineers, 1 2024.

[16] Naruki Shirahama, Naofumi Nakaya, Satoshi Watanabe, Kenji Moriya, Keiji Matsumoto, and Kazuhiro Koshi. Assessing emotional intelligence in ai: Visual analogue scale vs. likert scale in short story comprehension. *Journal of the Institute of Industrial Applications Engineers*, 13:21, 1 2025.

[17] Naruki Shirahama, Satoshi Watanabe, Kenji Moriya, Kazuhiro Koshi, and Keiji Matsumoto. A new method of subjective evaluation using visual analog scale for small sample data analysis. *Journal of Information Processing*, 29:424–433, 2021.

[18] Lei Song. Research on the application of computer 3d technology in the creation of films adapted from literary works. In *2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, pages 1461–1465, 4 2022.

[19] Daniel Susilo and Harliantara. The digital promotion of japanese and korean movie in ott platform by netflix. *Indonesian Journal of Business Analytics*, 3:1979–1994, 11 2023.

[20] Darin Wijaya, Hendri Murfi, and Gianinna Ardaneswari. Topic-level sentiment analysis for user reviews in gasoline subsidy application. In *2024 11th IEEE Swiss Conference on Data Science (SDS)*, pages 221–224, 5 2024.

[21] Chengmao Wu and Xiao Qi. Reconstruction-aware kernelized fuzzy clustering framework incorporating local information for image segmentation. *Neural Processing Letters*, 56:123, 2024.

[22] Feng Zhao, Yujie Yang, Hanqiang Liu, and Chaofei Wang. A robust multi-view knowledge transfer-based rough fuzzy c-means clustering algorithm. *Complex & Intelligent Systems*, 10:5331–5358, 2024.