CUBClustR: Clustering Rating data within the CUB framework

Matteo Ventura

April 5, 2026

Contents

---

1 Introduction

This document provides the comprehensive technical documentation for the implementation of the **MLC-CUB** (Multivariate Latent Class CUB) model in R. The framework allows segmenting ordinal data (ratings) by accounting for both the *feeling* process (reasoned choice) and the *uncertainty* of the respondent, grouping subjects into K latent clusters.

The estimation relies on the Expectation-Maximization (EM) algorithm, deeply optimized via a **C++** backend (`Rcpp`) to ensure high computational performance.

To use the library, keep the file CUBClustR_functions.R and the file engine.cpp in the same folder. Upload the function with the R command `source`.

2 Function Reference

2.1 `fit_MLCCUB`

Description: Estimates the parameters of an MLC-CUB model using the C++ accelerated EM algorithm.

Usage:

```
fit_MLCCUB(data, m, K, tol = 1e-5, EM_iter = 10, max_iter = 1500)
```

Arguments (Inputs):

- `data`: A numeric matrix or dataframe containing the ordinal responses. Rows represent subjects (N) and columns represent items (J).
- `m`: A numeric vector specifying the maximum number of response categories for each item.
- `K`: Integer. The number of latent classes (clusters) to estimate.
- `tol`: Numeric. Tolerance for the log-likelihood convergence. *(Default: 1e-5).*
- `EM_iter`: Integer. Number of random initializations to avoid local maxima. *(Default: 10).*
- `max_iter`: Integer. Maximum number of EM iterations. *(Default: 1500).*

Value (Outputs): A list containing:

- `pi`, `xi`, `omega`: Estimated uncertainty, feeling, and weight parameters.
- `LL`, `class`: Final Log-Likelihood and Maximum A Posteriori (MAP) cluster assignments.

2.2 `compare_MLCCUB_models`

Description: Fits multiple MLC-CUB models for a sequence of latent classes (K) and compares them to facilitate model selection using the Bayesian Information Criterion (BIC).

Usage:

```
compare_MLCCUB_models(data, m, K_vector, ...)
```

Arguments (Inputs):

- `data`: A numeric matrix or dataframe containing the observed ordinal ratings.
- `m`: A numeric vector specifying the maximum categories for each item.
- `K_vector`: An integer vector containing the range of clusters to be tested (e.g., `1:6`).
- `...`: Additional arguments passed directly to `fit_MLCCUB` (e.g., `EM_iter`, `tol`).

Value (Outputs): A list containing:

- `all_models`: A named list containing all the successfully fitted model objects.

- `summary`: A `data.frame` summarizing the results for each K, including Log-Likelihood, number of parameters, and BIC values.
- `best_k`: Integer. The number of clusters that yielded the lowest BIC.
- `best_model`: The complete model object corresponding to the optimal BIC.

2.3 `check_identifiability`

Description: Performs a non-parametric Bootstrap analysis to assess the stability and identifiability of the latent clusters. It calculates the Adjusted Rand Index (ARI) between assignments across different bootstrap samples.

Usage:

```
check_identifiability(data, m, K, n_boot = 100, n_cores = 1,
                      plot_hist = TRUE, seed = 250535,
                      cpp_file = .engine.cpp, ...)
```

Arguments (Inputs):

- `data`: The original dataset used for the analysis.
- `m`: Numeric vector of maximum categories per item.
- `K`: Integer. The number of clusters to test for stability.
- `n_boot`: Integer. The number of bootstrap samples to extract. *(Default: 100)*.
- `n_cores`: Integer. Number of CPU cores for parallel computation. *(Default: 1)*.
- `plot_hist`: Logical. If `TRUE`, generates the ARI distribution histogram.
- `cpp_file`: Path to the C++ engine required for parallel workers. This is needed to link the functions to the Rcpp code. Don't modify it.

Value (Outputs): A list containing:

- `ARI_matrix`: A symmetric matrix containing pairwise ARI scores between bootstrap iterations.
- `ARI_values`: A numeric vector of the pairwise ARI scores (flattened).
- `plot`: A `ggplot2` object showing the ARI distribution.
- `raw_clusters`: A list containing subject assignments for every bootstrap run.

2.4 `calculate_WAD`

Description: Computes the Weighted Average Dissimilarity (WAD) index and provides granular diagnostic metrics to evaluate the absolute goodness of fit.

Usage:

```
calculate_WAD(model_obj, data, m)
```

Arguments (Inputs):

- `model_obj`: A fitted model object.

- `data`: The original dataset. The function internally ensures compatibility via matrix conversion and index-matching checks.
- `m`: The vector of maximum categories.

Value (Outputs): A diagnostic list containing:

- `WAD`: Global fit index in $[0, 1]$. Values closer to 0 indicate excellent fit.
- `Cluster_Dissimilarity`: Unweighted average dissimilarity for each cluster.
- `Item_Dissimilarity`: Average dissimilarity per item, weighted by cluster sizes (ω).
- `Matrix_Dissimilarity`: The full $K \times J$ dissimilarity matrix.

2.5 Plotting Functions

2.5.1 `plot_MLCCUB_clusters`

Description: Generates a 2D parameter space plot (Feeling vs Uncertainty) for every item, grouped by cluster.

- `cluster_order`: Optional vector to manually reorder clusters in the legend.
- `custom_colors`: Character vector for custom HEX color palettes.

2.5.2 `plot_CUB_fit`

Description: Creates a multi-panel facet grid (*Clusters* $\times$ *Items*) comparing observed relative frequencies (solid black segments) with theoretical CUB distributions (dashed lines).

Usage:

```
plot_CUB_fit(model_obj, data, m, cluster_order = NULL, line_color = "
    darkblue")
```

Customization:

- `line_color`: Change the HEX string to customize the theoretical probability line (e.g., `"#E74C3C"` for red). *Default: Dark Blue.*
- **Robustness:** The function automatically handles missing values and prevents "subscript out of bounds" errors by verifying row consistency between the model and the data.