User Manual for Consensus Rankings User Interface

Contents

1 Main interface

1.1 Initialization

Before downloading the package `RankingTool`, please make sure all dependencies were installed. The following dependencies are required:

- `PIL, json, os, tkinter, matplotlib`
- `pyautogui, pandas, toml, sys`
- `ry2, subprocess`

Download the package using

```
pip install rankingTool==1.0
```

To initialize the GUI, the user needs a configuration file. An example of the configuration file is provided in `config.toml`. Specifically, it contains the following arguments:

1. **[default]**
   - name (str): name of the rankings UI.
   - num_str (int): length of strings allowed for the "short" version of the item names.
   - rat_min (int): minimum of the ratings.
   - rat_max (int): maximum of all the ratings.
   - overall_merit_max (int): minimum of the overall ratings.
   - overall_merit_min (int): maximum of the overall ratings.
   - topk (int): number of top items included.
2. **[default_graphic_attributes]**
   - color (str): default background color of item boxes.
   - outline (str): default color of the outline of item boxes.
   - width (int): default width of the outline of item boxes.
   - dash (int): default dash type of the outline of item boxes.
3. **[graphic_to_rating]**
   - Bands (str): Seems currently useless in codes?
   - Box_Background_Color (str): feature represented by the background colors of item boxes.
   - Width (str): feature represented by the width of the outline of item boxes.
   - Dash (str): feature represented by dash types of the outline of item boxes.
   - Outline (str): feature represented by colors of the outline of item boxes.

4. **[review]**
    - titles (str): feature name of the text review.
5. **[filter]**
    - yesno_list (lst): Seems currently useless in codes?
6. **[box_size]**
    - box_width (numeric): width of each item box.
    - box_height (numeric): height of each item box.
    - box_distance_x (numeric): distance along x axis between adjacent item boxes.
    - box_distance_y (numeric): distance along y axis between adjacent item boxes.
    - ranking_area_x (numeric): total length of the ranking area along x axis.
7. **[box_graph_attributes]**: this part specifies the values of ratings represented by the graphical attributes. Currently using integers.
    - [box_graph_attributes.color]: rating (numeric) = color (str).
    - [box_graph_attributes.width]: rating (numeric) = width (numeric).
    - [box_graph_attributes.dash]: rating (numeric) = dash type (tuple).

After installation and preparation of the configuration file, the GUI can be opened by running the `main.py` file with the following codes:

```
instance = GUI(``config.toml'')
instance.show()
```

1.2 Loading data

We start by loading data into the GUI in the window displayed in Figure 1.1.

The option menu with default type `.xlsx/.xls` specifies the format of data. If `.xlsx/.xls` is selected, the excel file can have multiple sheets depending on the type of the data and the file whould be loaded. If `.csv` is selected, the directory containing the .csv files should be selected.

The button "Select data file(s)" will open a window for selecting directory/file. After selecting, the line "Data Selected!" will appear below the button.

The option menu with default type `Rankings only` specifies the type of data. There are three options: "Rankings only", "Ratings only" and "Ratings and Rankings".

Table 1.1 concludes the input document needed under each combination:

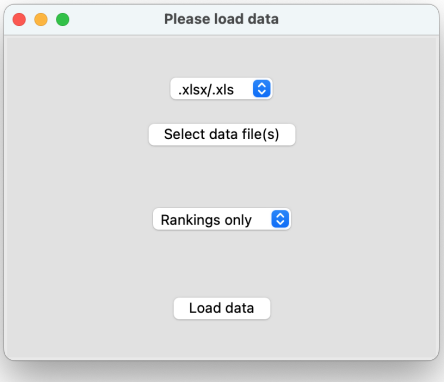


Figure 1.1 Window for loading data.

Datatype	File format	Document required	Models
`Rankings only`	`.xlsx/.xls`	Sheet names: "Rankings", "Attributes"	Mallows, GMM, RIM, Borda
	`.csv`	"rankings.csv"	
`Ratings only`	`.xlsx/.xls`	Sheet names: "Scores", "Attributes"	Borda
	`.csv`	"ratings.csv"	
`Ratings and Rankings`	`.xlsx/.xls`	Sheet names: "Scores","Ratings", "Attributes"	Mallows, GMM, RIM, Landmarks, Borda
	`.csv`	"ratings.csv", "rankings.csv"	

Table 1.1 Table of input types and document(s) required.

After inputting the document(s), click on "Load data" to load in the dataset. The excel file should be arranged as follows:

- Sheet "Scores": The first column contains the names of the candidates/proposals; the second column contains the names of the reviewers; the rest of the columns represent different features.
- Sheet "Rankings": The columns represent rankings of items for all reviewers (using reviewer names). The names of the columns should match the names of the reviewers in "Rankings", if "`Ratings and Rankings`" is selected.
- Sheet "Attributes": Has two columns "Categories" and "Type". "Categories" contain all features in the "Scores" sheet and "Type" specifies whether it is a sub-rating ("Ratings"), text commment ("Text") or overall rating ("Main").

Note that here "overall rating" means the ratings we use to generate the consensus rankings.

1.3 Main canvas

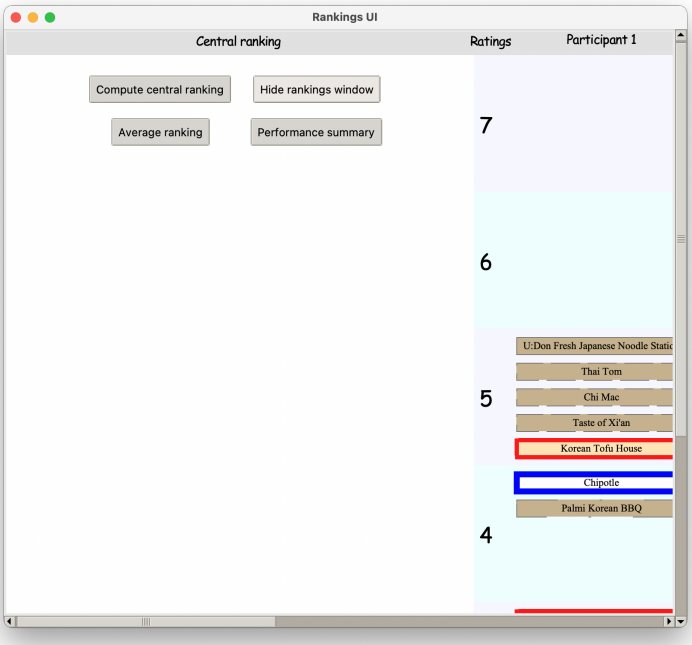


Figure 1.2 Main canvas of the GUI.

On the left-hand side, the four buttons have the following functions:

- Button "Compute consensus ranking": open the window for computation under models in Section 2.1.
- Button "Hide rankings window": hide the consensus rankings window temporarily while still saving the results.
- Button "Average ranking": provide basic statistics based on the consensus rankings computed using the current model.
- Button "Performance summary": open the performance summary window in Section 2.3. Should be used after computing the results.

On the right-hand side, the columns represent all the reviewers and the rows represent the overall ratings. The item boxes are displayed according to the ratings.

- Single clicking an item box will color the same item for all reviewers so that the user can see the variation among reviewers.
- Right clicking the item boxes will show the option menu in Figure 1.3. The options are:

  - Filter: open a window that can filter certain entries.
  - Rating Details: open a window that contains all detailed sub-ratings.
  - Review Text: open a window that contains the reviews in text.
  - Proposal Details: open a window that contains the proposal details. Not currently used.
  - Exit: exit the option menu.
- Double clicking an item will shift the entire column with the column on its left if exists. So the order of the reviewers can be changed.

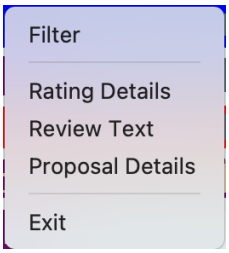


Figure 1.3 Option menu after right clicking the item.

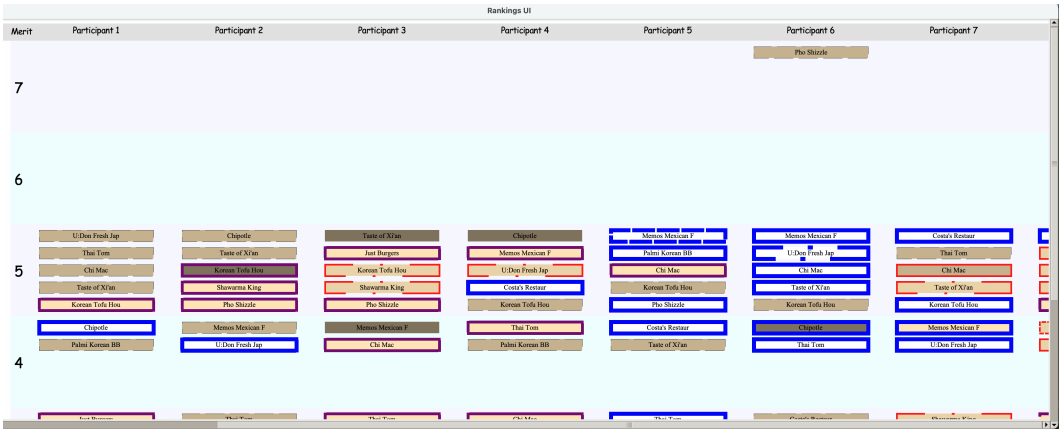


Figure 1.4 Item boxes on the canvas.

1.4 Legends

The legend window displays a table of the graphical attributes and the corresponding features they represent, see Figure 1.5.

Double clicking any row in this table will open a window that shows the details of that feature, see Figure 1.6.

On top of the window, there is a button called “Change Graphical Attributes” for changing their assignments. The new window is in Figure 1.7.

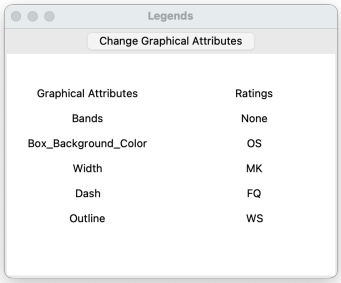


Figure 1.5 The legend window.

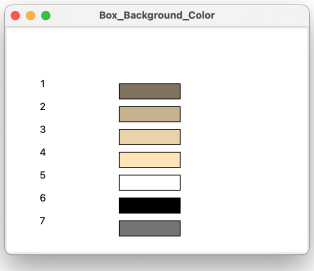


Figure 1.6 A sub window for the box color in the legend.

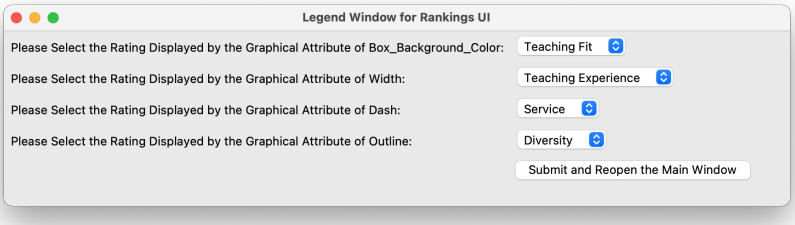


Figure 1.7 Window for changing graphical attributes.

2 Concensus rankings

2.1 Consensus rankings window

The window has an area for computing the rankings under different models.

- It starts with an option menu with the model options under the current data type.
- Button "Set working directory": choose a directory for saving model outputs. The selected path will be printed next to the button.
- Button "Load Model Results": choose a `.json` file with saved model results. After loading, the saved results would re-display on the window.
- Button "Compute Consensus Ranking": compute consensus rankings for the chosen model.
- Button "Reset parameter(s)": if there are parameter(s) for the model, the default values will be reset.
- Button "Save models": save the computed model results in a `.json` file.

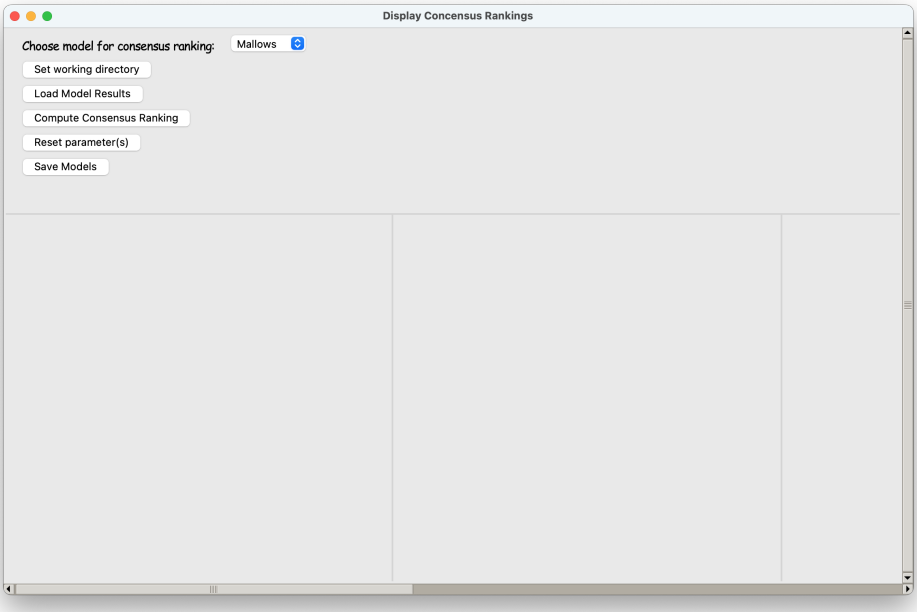


Figure 2.1 Window for consensus rankings.

2.2 Model details

The user may select among the following models for computation. The default choice is the Mallows model, which is very fast and has a single paramter θ. Revisit Table 1.1 for the models available under different data types.

Model	Parameter(s)	Results displayed	Reference(s)
Mallows		cost; consensus rankings; $\exp(-\theta)$	Mallows ((1957))
RIM	iterations: no. of iterations temp: tuning parameter for convergence	cost; tree plot; $\exp(-\theta)$	Meek and Meila ((2014))
GMM		cost; consensus rankings; $\exp(-\theta_i)$	Fligner and Verducci ((1986))
Landmarks		cost; consensus rankings; $\exp(-\theta_i)$	
Borda		consensus rankings; column sums of Q matrix (rankings and/or ratings)	

Table 2.1 Table of parameters, results displayed and references for the concensus ranking models used.

The models used are summarized in Table 2.2.

For Borda, if both rankings and ratings are provided, the rankings based on rankings (increasing) and ratings (decreasing) will be computed. They should agree with each other.

For users not familiar with the details of the model, we also highlight important properties of the models:

Model	Computation Speed	Cost	Interpretability
Mallows	Very fast	Higher because of its simplicity	
GMM	Fast	Lower than Mallows	
RIM	Moderate, depending on number of iterations	Lower, complex model	Tree plot available
Landmarks	Moderate, depending on number of iterations		Can utilize both rankings and ratings information
Borda	Very fast	—	Intuitive

Table 2.2 Table of comparisons between different models.

Faster models might be preferred if the number of items or the number of bootstrap samples is large.

2.3 Model summary window

Upon opening the model comparison window, on the left-hand side we can first select the window(s) we wish to compare. Only models computed in the model comparison window will be included. See Figure 3.6.

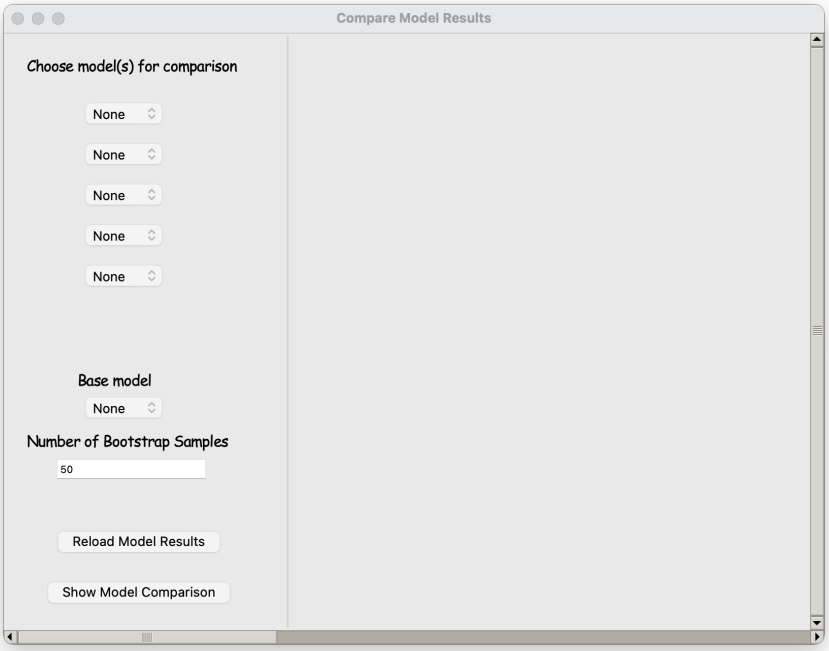


Figure 2.2 Selecting model(s) for comparison.

We then need to select a "base" model used as a benchmark for comparisons across rankings from different models.

The number of bootstrap samples (integer) should be entered for comparing the variation of results among bootstrap samples.

The button "Reload Model Results" is used when results from new models are computed and they need to be updated into the summary window.

The button "Show Model Comparison" computes the summary of the models which will appear on the right-hand side of the window.

The following results will be displayed, from the first row to the third:

- Q matrices computed using the dataset with columns ranked by consensus ranking computed from each model. We expect the color pallette to be smooth for the rankings to be sensible.

- Q matrices computed using the set of consensus rankings computed from the bootstrap samples. Items are ranked by the consensus ranking computed from each model. Lighter colors indicate smaller differences between results from bootstrap samples.

- Comparison matrix across different models. Lighter color indicates smaller difference between results from different models.

The formula for the Q matrix is as follows: Let $\mathcal{E} = \{e_1, \ldots, e_n\}$ be the set of items. We use $e \prec_\pi e'$ to denote the case that e percedes e' in the permutation π. For a dataset containing N sets of rankings $\{\pi_1, \ldots, \pi_N\}$ on n items, the $n \times n$ matrix is computed as

$$Q_{e,e'} = \sum_{i=1}^{N} \mathbb{1}_{e \prec_{\pi_i} e'} / N. \tag{2.1}$$

It is the empirical estimate of the probability of item e perceding e'.

The values of the matrices can be checked by placing the mouse on the blocks. See Figure 2.3.

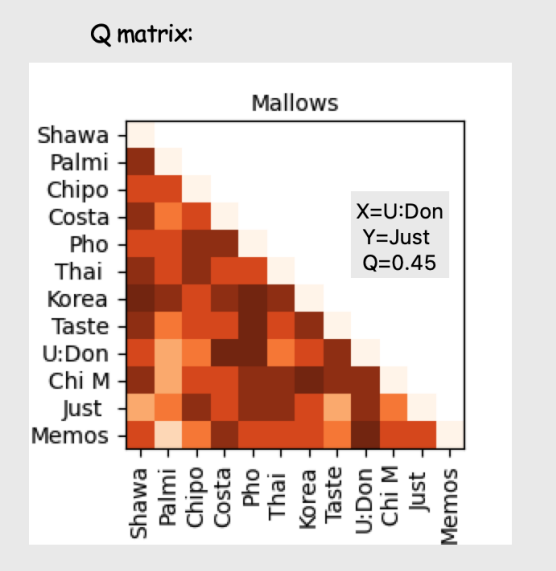
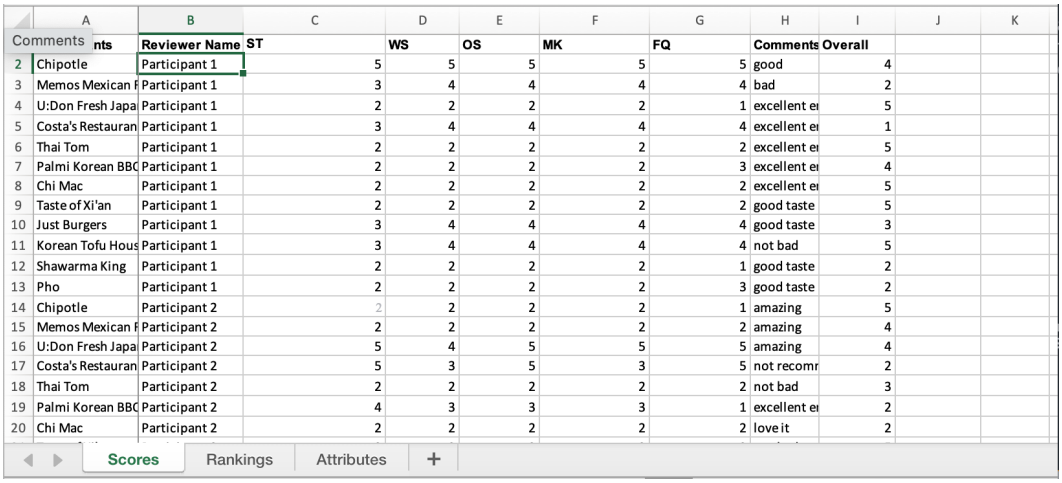


Figure 2.3 Moving the mouse on matrices can check the exact values.

3 A simple example

In this section we present using our GUI to analyse a dataset containing reviews from 12 participants on restaurants in University District of Seattle. The dataset has both rankings and ratings and is in the format of an Excel file, see Figure 3.1.



	A	B	C	D	E	F	G	H	I	J	K
	Comments										
1	nts	Reviewer Name	ST	WS	OS	MK	FQ	Comments	Overall		
2	Chipotle	Participant 1	5	5	5	5	5	good	4		
3	Memos Mexican F	Participant 1	3	4	4	4	4	bad	2		
4	U:Don Fresh Japa	Participant 1	2	2	2	2	1	excellent e	5		
5	Costa's Restauran	Participant 1	3	4	4	4	4	excellent e	1		
6	Thai Tom	Participant 1	2	2	2	2	2	excellent e	5		
7	Palmi Korean BBC	Participant 1	2	2	2	2	3	excellent e	4		
8	Chi Mac	Participant 1	2	2	2	2	2	excellent e	5		
9	Taste of Xi'an	Participant 1	2	2	2	2	2	good taste	5		
10	Just Burgers	Participant 1	3	4	4	4	4	good taste	3		
11	Korean Tofu Hous	Participant 1	3	4	4	4	4	not bad	5		
12	Shawarma King	Participant 1	2	2	2	2	1	good taste	2		
13	Pho	Participant 1	2	2	2	2	3	good taste	2		
14	Chipotle	Participant 2	2	2	2	2	1	amazing	5		
15	Memos Mexican F	Participant 2	2	2	2	2	2	amazing	4		
16	U:Don Fresh Japa	Participant 2	5	4	5	5	5	amazing	4		
17	Costa's Restauran	Participant 2	5	3	5	3	5	not recomr	2		
18	Thai Tom	Participant 2	2	2	2	2	2	not bad	3		
19	Palmi Korean BBC	Participant 2	4	3	3	3	1	excellent e	2		
20	Chi Mac	Participant 2	2	2	2	2	2	love it	2		

Scores Rankings Attributes +

Figure 3.1 Dataset used in the example.

The column "overall" contains the scores for ranking. There are five sub-ratings denoted as ST, WS, OS, MK and FQ. A text review is in the column "Comments". Then we prepare the configuration file `config.toml`. Running the `main.py` file, we load in the data from the following window:

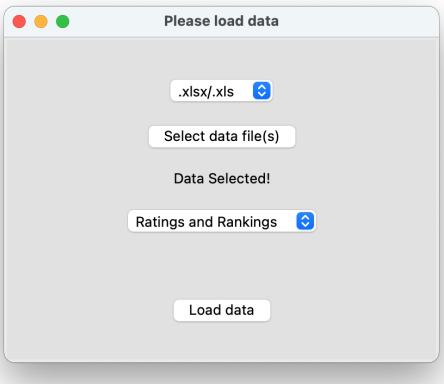


Figure 3.2 Loading data into the GUI.

Since we have both rankings and ratings from the data, we select the option ``Ratings and Rankings''. After pressing "Load data", the main canvas would pop up as in

Figure 3.3.

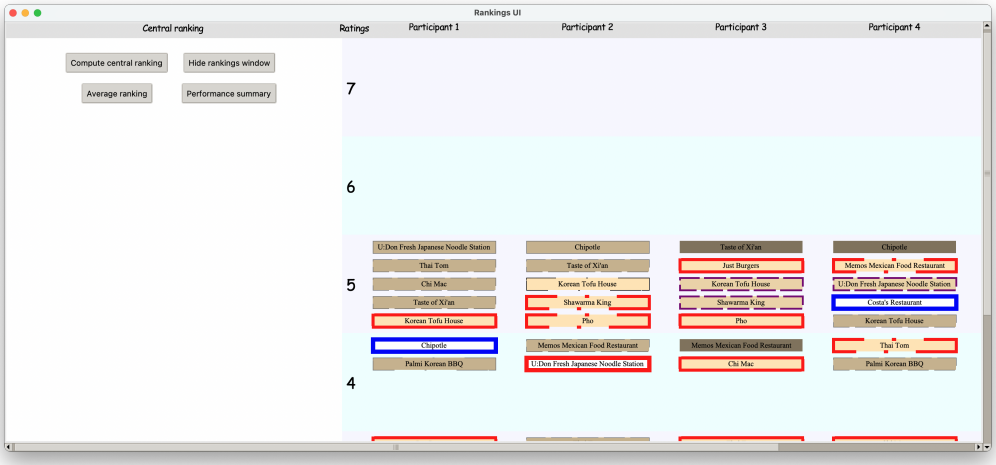


Figure 3.3 Main canvas for the example.

Then we can check the details of our items: if we press an item, for example, Korean Tofu House, the same candidate will darken in all columns, see Figure 3.4a. If we right-click we can use the option menu to:

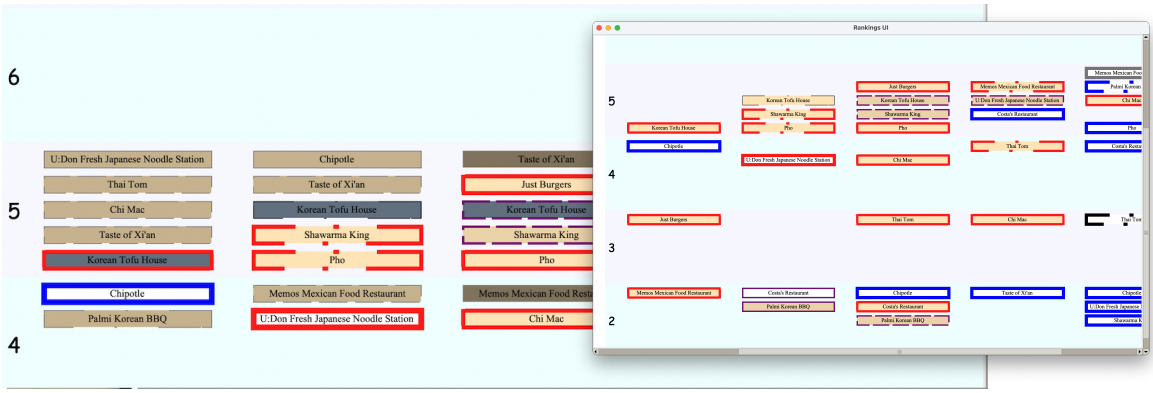
- Filter the items. By selecting items with minimum scores 2 and maximum 5 for all sub-ratings, the items filtered can be found as in Figure 3.4b.
- Check the sub-ratings. The "Rating details" option allows us to check the 4 sub-ratings of an item; see Figure 3.4c.
- Check the review texts with the "Review Texts" option; see Figure 3.4d.
- This example does not include any proposal details; so we skip this function.

Now we move on to compute the consensus rankings by clicking the "Compute consensus rankings" button. Since we have both rankings and ratings in the dataset, we can use all models. We start by selecting the output directory by using "Set working directory". Then we compute using the default Mallows model, see the result in Figure 3.5a.

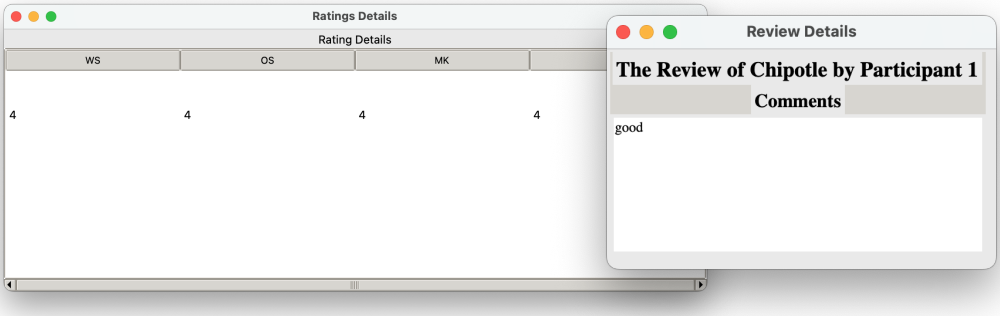
Similarly for the rest of the models we can compute them one-by-one, see the results in Figure 3.5b. In the working directory selected, there will be severall folders named by the models, and each of them will contain the intermediate file(s) from their packages. We can save the results to a `.json` file by clicking on the "Save Models" button.

Now we move on to compare the models we have by clicking the "Performance summary" button on the main canvas. We then compare the models with base model Mallows and use 50 bootstrap samples, see Figure 3.6 for the settings.

The results are in Figure 3.7.

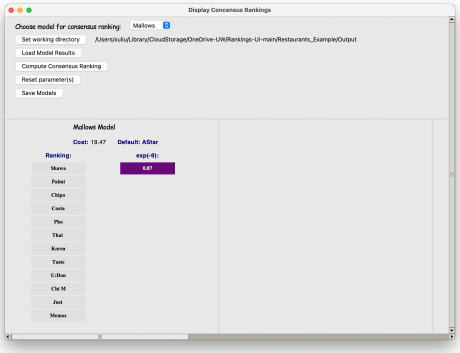


(a) (b)

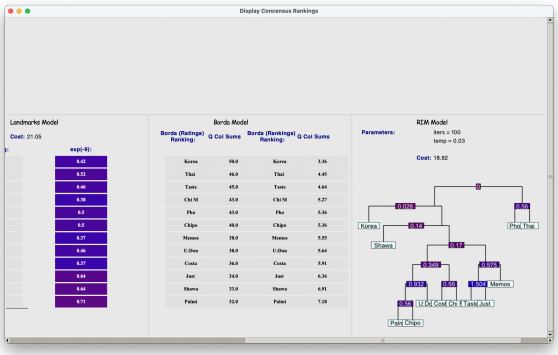


(c) (d)

Figure 3.4 Figures for functions on the main canvas.



(a) Results from Mallows model.



(b) Results from other models.

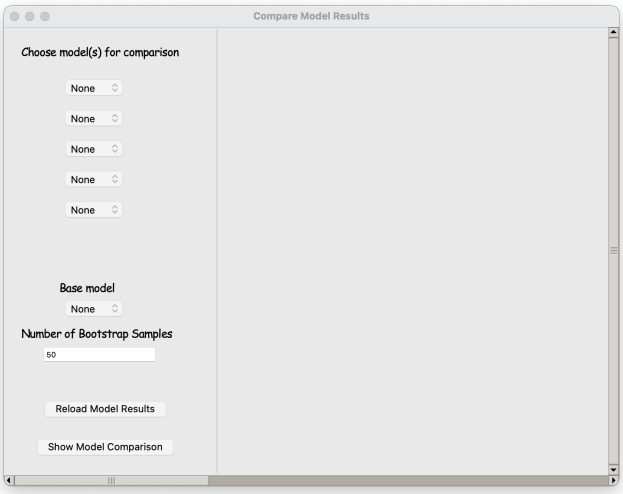


Figure 3.6 Loading data into the GUI.

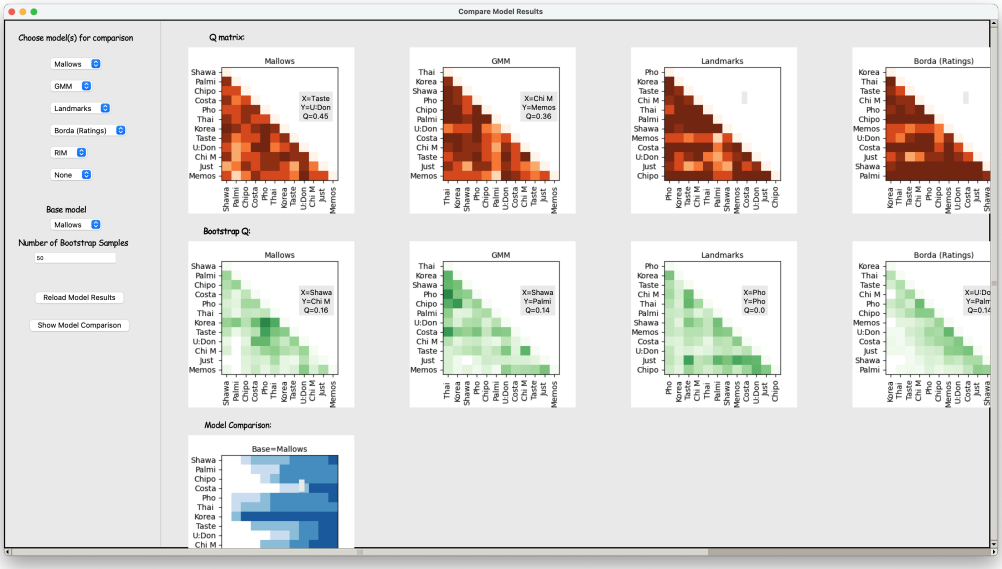


Figure 3.7 Loading data into the GUI.

Bibliography

M. A. Fligner and J. S. Verducci. Distance based ranking models. *Journal of the Royal Statistical Society: Series B*, 48(3):359–369, 1986.

C. L. Mallows. Non-null ranking models. I. *Biometrika*, 44(1/2):114–130, 1957.

C. Meek and M. Meila. Recursive inversion models for permutations. *Advances in Neural Information Processing Systems*, 27, 2014.