Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin
Parsons Building, Trinity College, Dublin 2, DUBLIN

Engineering Student Report
Second-Year Internship
Field : Mathematical Modeling and Data Science

Comparative Triple Judgement

Presented by :
Romain Perrier

Academic Advisor : Jonas Koko
ISIMA Tutor : Susan Arbon-Leahy
Company Tutor : Kevin Kelly

Defense Date :
March 9, 2024
Internship Duration :
4 months

ISIMA, University Campus des Cézeaux, 1 rue de la Chebarde, CS 60026,
63178 Aubière CEDEX

Acknowledgements

I would like to thank my internship supervisor, Kevin Kelly, for giving me the opportunity to intern at Trinity College in Dublin. Additionally, his guidance and advice throughout the project were invaluable in helping me understand the concepts surrounding his model. I also appreciate him for introducing me to the field of information theory.

I am also grateful to Susan Arbon-Leahy, my English professor at ISIMA and my mentor, whose classes throughout the year, availability during the internship, and introduction to Trinity College were crucial for securing this internship.

I would like to extend my thanks to Anneliese Walsh and Patrick Lynch, who, along with Kevin Kelly, were present at the biweekly meetings to discuss the progress of the projects conducted within RAIL.

Special thanks to Lucas Pichon and Beatrice Wharton, who helped me test the portability of my library on different operating systems.

I would also like to acknowledge my peers and colleagues at ISIMA, as well as the PhD students at Trinity College and the Irish interns, with whom I exchanged ideas about our respective projects. Their collaboration and support created a conducive environment for work and productivity.

Thank you all.

Table des matières

Table des figures

Abstract

When evaluating or judging an item, it is common to assign it a value. However, evaluating objects related to abstract concepts can make this numerical value assignment challenging. To address this issue, **comparative judgments** offer an alternative by focusing not on the intrinsic value of the items, but on their comparison with each other. This method allows for a more accurate overall view and facilitates simpler and fairer tests.

To compare different evaluation methods, I developed a **Python** package implementing one direct judgment method and two comparative judgment methods: adaptive comparative judgment (**ACJ**) and triple comparative judgment (**CTJ**). These methods optimize the selection of items to compare by using **information theory**, enhancing the relevance of the comparisons made by the user.

Keywords: comparative judgment, ACJ, CTJ, information theory, Python

Lexique

absolute judgment The process of rating a series of items based solely on predefined criteria, without considering the items as a whole. 4, 8, 12, 13, 17, 18, 27 . 12.

ACJ Adaptive Comparative Judgement. It is a comparative judgment between two items where users must select the better item. It selects the items to present to users based on the information they can provide.. 4, 5, 12–14, 18, 19, 21, 22, 25, 28, 30, 32, 34, 35, 37, 38, 40, 41

comparative judgment The process of comparing items with each other to establish an order and assign values to the items based on this order and a specified algorithm. 8, 12, 13 . 12, 13.

CTJ Comparative Triple Judgement. It is a comparative judgment for three items. The task is to sort them and then select the distance of the middle item from the others. The model was conceived by Dr. Kelly.. 4, 8, 12, 14, 15, 17–19, 22–27, 30, 33, 35–41

interaction information Corresponds to the information provided by an n-tuple, here a trio. 22

joint entropy Also known as Shannon joint entropy, it corresponds to the amount of information of a pair of elements. 21, 22

library A collection of source code providing users with access to modules. 8, 12, 17, 24, 27–31, 34, 40, 41

license A document that imposes constraints on the use of a project or any other asset, whether material or immaterial. 27, 40, 41

mutual information Corresponds to the probabilistic dependence between the two elements of a pair. 21

n-tuple A tuple of n elements. 6, 13, 19, 22 A pair is a n-tuple of two elements. 13, 19, 21, 22, 27, 32. A trio is a 3-tuple of elements.. 14, 15, 22–24, 27, 35.

open source The practice of developing a project that will be freely accessible, shareable, usable, and improvable by users under certain conditions.. 8, 27, 28, 40

PIP A site for publishing and testing libraries.. 30

Python An object-oriented programming language, one of the most widely used.. 8, 12, 27, 29, 41

RAIL The Robotics and Innovation Laboratory, where I completed my internship.. 4, 11

roulette selection A method of selecting elements that maintains an element of randomness while taking into account the impact of one or more characteristics. 4, 19–24, 28

Shannon entropy Corresponds to the amount of information an element can provide. 4, 20–22

TCD Trinity College Dublin, the main university in Dublin where I completed my internship.. 4, 8–10

web application Code that runs on a server and is accessible via a URL. . 12.

Introduction

This internship was entrusted to me by Dr. Kelly, an engineer in mechanical, manufacturing, and biomedical engineering, as well as an Associate Professor at Trinity College Dublin (TCD).

The goal of evaluation methods is, as their name suggests, to assess a set of elements and assign a value to each one. Generally, this value falls within a range previously known to the evaluator. There are two main categories of evaluation methods : absolute judgment, which involves evaluating an element based on specific criteria and assigning a score independent of the other elements, and comparative judgment, where the evaluator compares the elements of the set to each other, and an algorithm then determines the score of each element based on the evaluator's judgments.

My internship aimed to develop an open source Python library to facilitate research on the Triple Comparative Judgment (CTJ), a model created by my supervisor, Dr. Kelly.

To achieve this, the library must be portable to facilitate its execution across different operating systems. It should implement several evaluation methods to compare with CTJ. Additionally, each method should be usable in multiple contexts : the first being automatic judgment without external intervention, with the possibility of introducing biases ; the second being user-performed judgment to verify the model results with knowledge of the elements' values ; and the last being a real-world test, i.e., without prior knowledge of the elements to be evaluated. A graphical user interface should thus be integrated for the latter two cases.

It was necessary to carefully consider the structure of the library and the functions accessible to users. Furthermore, since the CTJ is a recent model, it was also essential to think about how to select the elements to compare in order to optimize the information provided to the model by the evaluation.

To present this project, I will first discuss the internship and its organization. Next, I will present the evaluation methods implemented, as well as the librarys used for implementation. Finally, I will detail the developed library, its functioning, and the preliminary analysis results on a basic dataset.

1 Internship Presentation

1.1 Introduction to TCD

Trinity College Dublin (TCD) is the oldest university in Ireland ; indeed, it was founded in 1592 by Queen of England, Elizabeth I. Its specific historical status makes it a place where students, doctoral candidates, and professors mingle with tourists and history enthusiasts. This contrast is all the more apparent as TCD houses a major tourist attraction in Dublin, the "Book of Kells Experience". This museum provides access to the old library of TCD, which contains several thousand ancient manuscripts.

TCD is composed of three distinct faculties. The Faculty of Arts, Humanities, and Social Sciences includes courses in literature, film, law, philosophy, and psychology ; this is the literary hub of TCD. The second hub is the scientific one, the Faculty of Engineering, Mathematics, and Science. The main courses taught here are mathematics, engineering sciences, natural sciences, and physics. Finally, the last faculty is that of Health Sciences, including courses in pharmacy, medicine, and dentistry. In addition to these three faculties, an internal business school at TCD was opened in 2019.

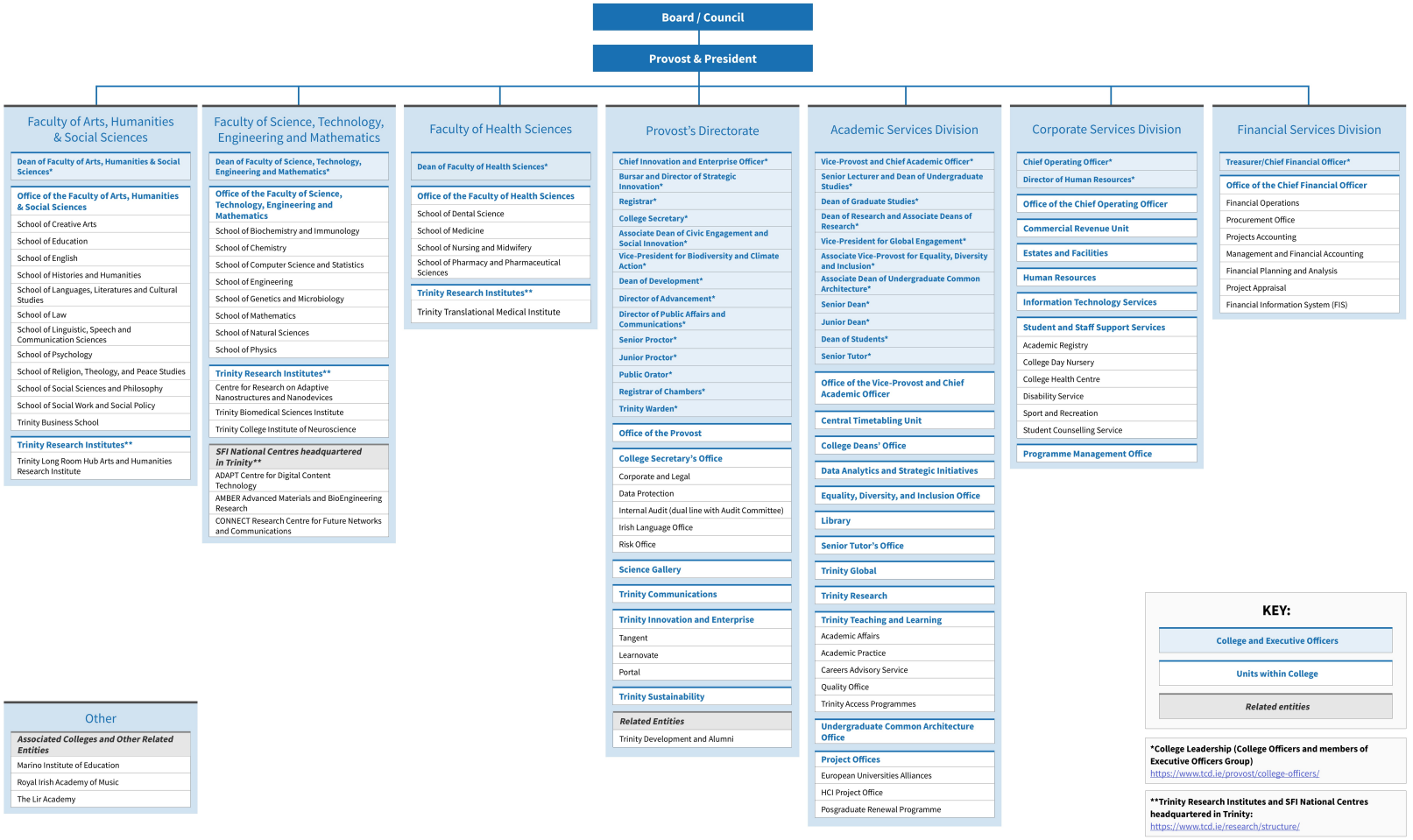
This institution is also renowned for having educated great literary and scientific minds such as William Rowan Hamilton. He is best known for introducing quaternions, the first example of hypercomplex numbers, which are an extension of complex numbers in algebra. Samuel Beckett, Nobel Prize in Literature laureate, is another notable figure, primarily known for his absurd play, "Waiting for Godot".

1.1.1 Organization

TCD is administered by a board of directors, which includes the provost, equivalent to a director, elected for a 10-year term and approved by the Irish government ; the vice-provost ; the chief of studies ; the registrar ; and the steward. This board also includes six fellows, who are staff members with a significant impact in their teaching fields, five other academic staff members, including at least three lecturers, two professors, three non-academic staff members, four students including at least one postgraduate student, an external member chosen by a board committee from nominations by organizations representing commercial or professional interests, and a member appointed by the Minister for Education.

FIGURE 1.1 – Organization chart of TCD

Date: 12 July 2023



1.1.2 Robotics and Innovation Laboratory (RAIL)

RAIL is a laboratory within the Department of Mechanical Engineering, Manufacturing, and Biomedical Engineering, founded in 1980. The main members of this laboratory are Dr. Kelly and Dr. Lynch, who have been supervisors for ISIMA students as well as students in the Engineering and Management cycle. To obtain their end-of-year diploma, students in this cycle must complete a group project addressing a public issue. The laboratory is also frequented by PhD students and other interns who carry out summer work to validate their academic cycle.

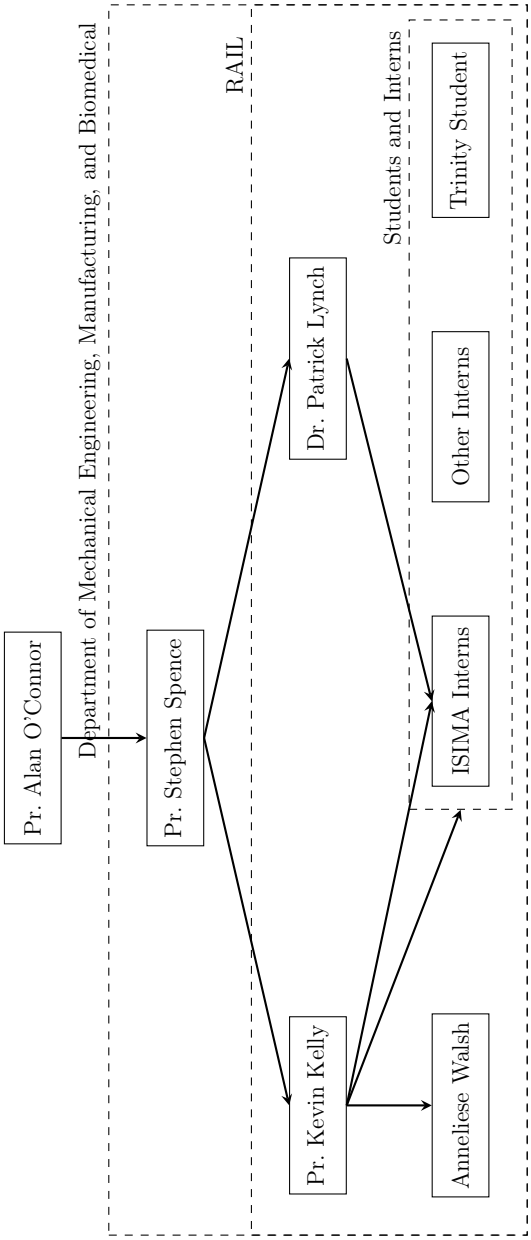


Figure 1.2 – Organization chart of RAIL

1.2 Objective of the Internship

The evaluation of works, individuals, or concepts is currently primarily done using absolute judgmentss. The emergence of comparative judgments seems to indicate that they might be more accurate and effective when aiming to obtain a representative rating of the evaluated group. Dr. Kelly therefore proposes a new method, CTJ, which is based on comparative judgments, but is expected to outperform existing methods.

Following tests, mainly conducted using web applications [1], it was decided to create a Python library to continue testing this method. Ultimately, the goal is to distribute it to allow a larger number of people to contribute to the tests and to verify if it is indeed more effective than existing methods.

To achieve this, the library should provide a means of performing an absolute judgment, as well as a classic comparative judgment, using a method whose effectiveness has been proven, specifically the adaptive comparative judgment (ACJ) [2]. Finally, it should integrate the method we are testing, which is the comparative triple judgment (CTJ) [1].

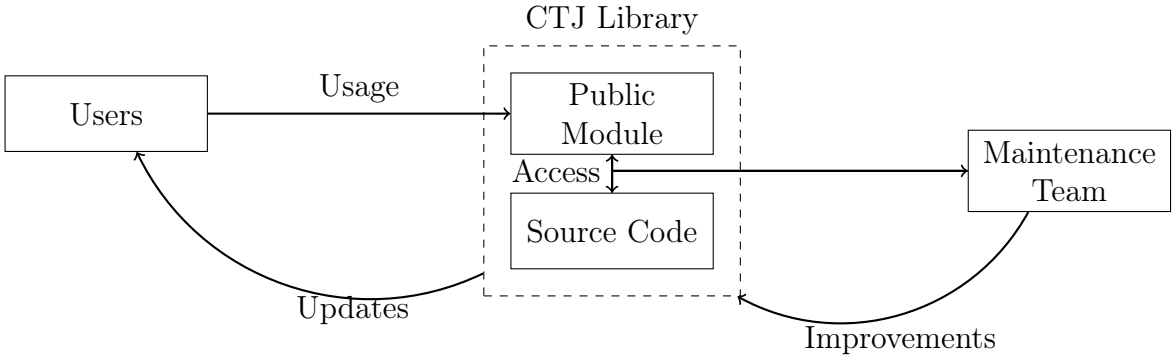


Figure 1.3 – Functional Diagram of the Library

1.2.1 Absolute Method

The absolute method involves performing an absolute judgment for each item in the set. It is the most common when it comes to evaluation and involves assigning a value to an item based on a predefined scale, without having an overall understanding of the entire dataset to be evaluated.

The issue with this method lies in several aspects. First, the resulting ranking is representative of an item taken individually, but it is not necessarily representative of the group as a whole. Additionally, assigning a raw value can become complex if the evaluated concepts themselves are complex. Moreover, evaluators may be biased by the items evaluated just before or after, which can affect the relevance of their judgments.

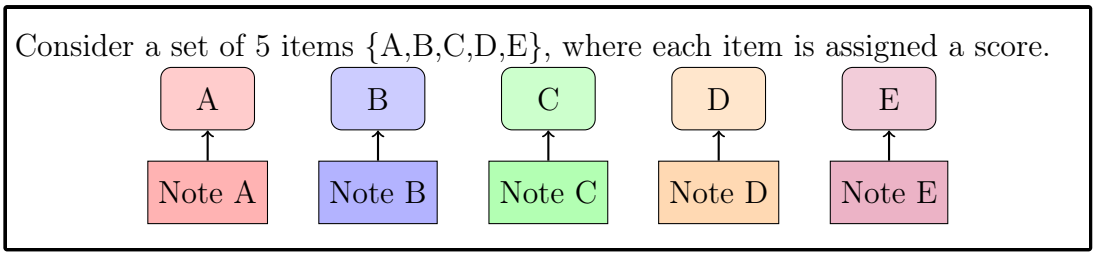


FIGURE 1.4 – Absolute judgment

Thus, the need to use comparative judgmentss seems more appropriate for obtaining a ranking and evaluation that are more representative of the entire set being evaluated, while eliminating some biases and uncertainties.

1.2.2 Comparative Methods

Comparative methods, or comparative judgments, involve considering the elements as integral parts of the whole. Thus, evaluations are no longer focused on a single element at a time, but on comparing the elements with each other. In each evaluation, an n-tuple of elements is assessed, and the evaluator is tasked with ranking the presented elements from best to worst. An algorithm then determines a value associated with each element [3].

The major challenge of comparative methods is to select the elements presented in such a way that they provide the most information possible to the system. The goal is to reduce the number of evaluations to a reasonable number.

ACJ

The ACJ enhances the standard method of comparative judgment for evaluating a pair of elements.

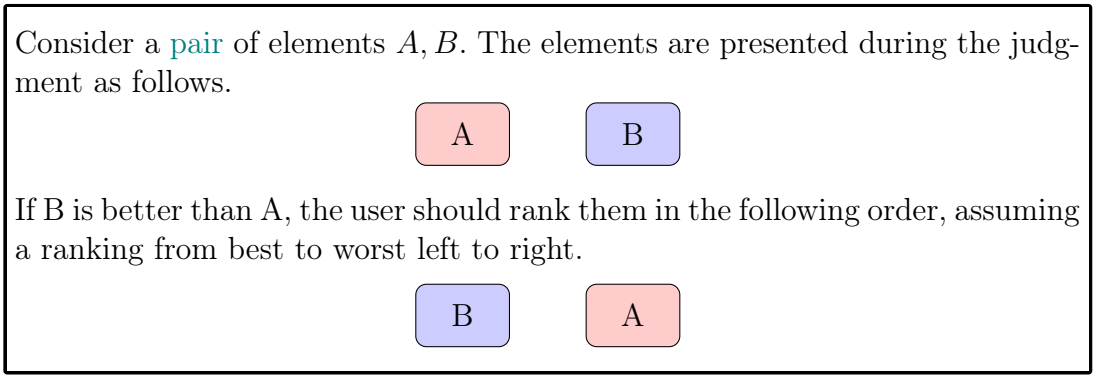


FIGURE 1.5 – Iteration of the ACJ

This improvement consists of presenting a pair in each evaluation to provide more information to the system.

Although ACJ performs better than absolute judgment and the default comparative judgment, a problem remains. The ACJ returns a ranking of the elements, and the assi-

gnment of values is done subsequently, which can result in a loss of information about the exact values of the elements, although the order is preserved.

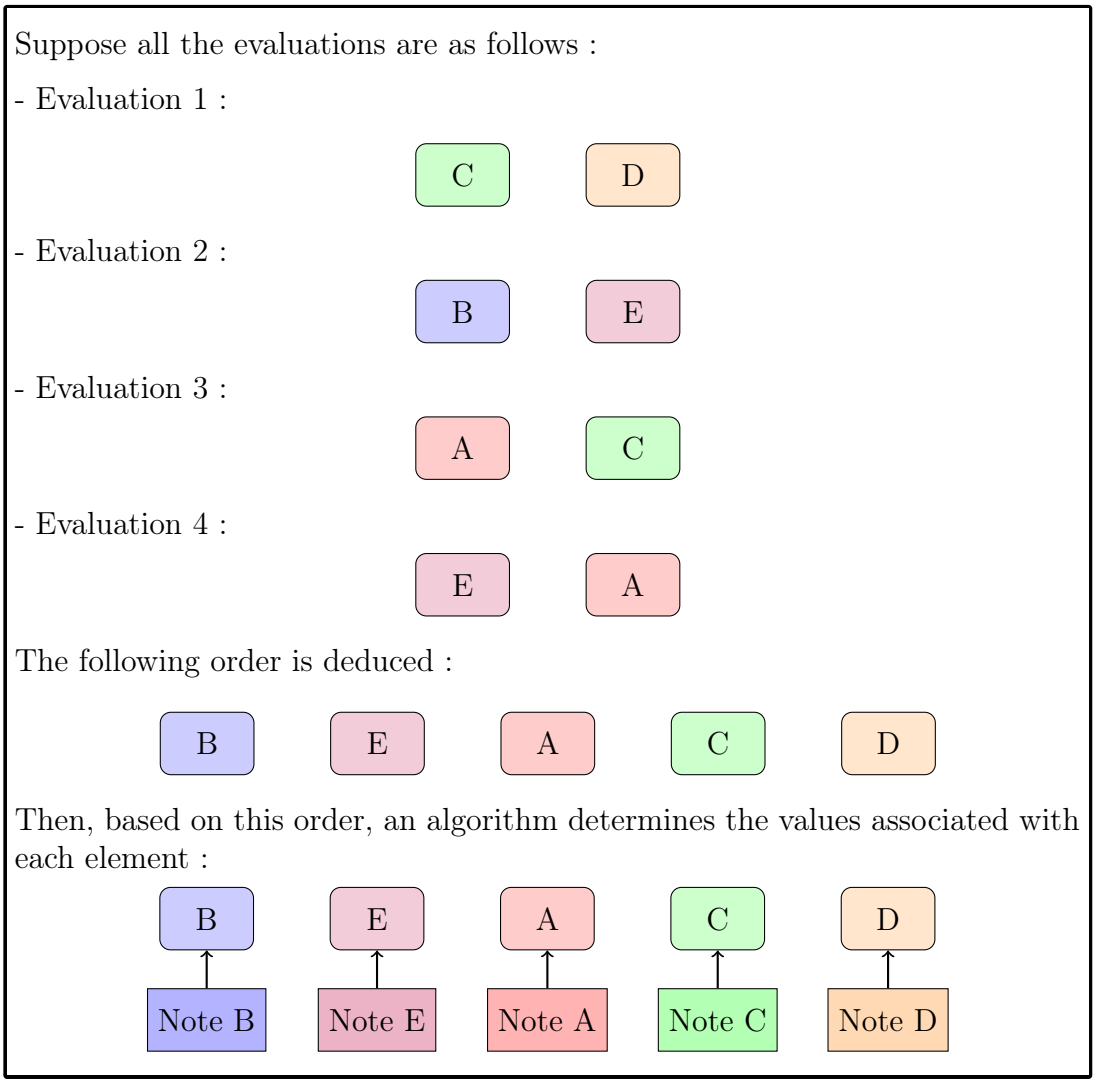


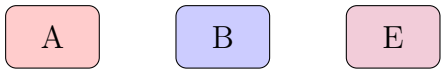
FIGURE 1.6 – Operation of the ACJ

CTJ

The CTJ is a method aimed at creating a ranking while simultaneously determining the values associated with each element in the set. As its name suggests, it involves comparing elements three at a time, so each evaluation involves a trio.

The evaluator must sort the trio from the worst to the best and place the middle element at a distance d_{best} from the best element and d_{worst} from the worst.

Consider a trio of elements A, B, E presented as follows :



If the ordered trio from best to worst is B, E, A, then the user's judgment should correspond to :

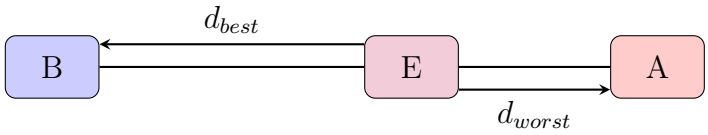
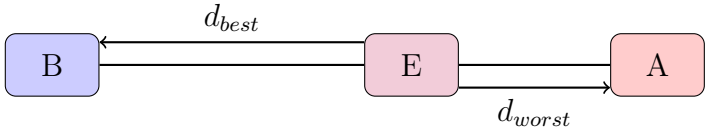


FIGURE 1.7 – Iteration of the CTJ

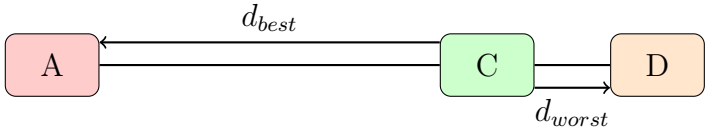
These successive evaluations allow for sorting the elements and estimating their values using a simple linear system.

Suppose the evaluations are as follows :

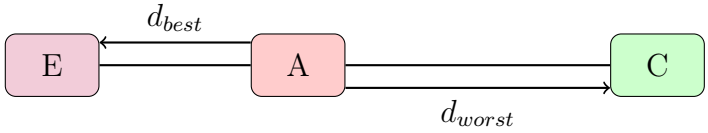
- Evaluation 1 :



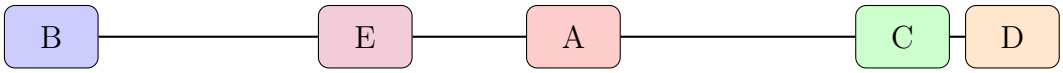
- Evaluation 2 :



- Evaluation 3 :



This leads to the following scale :



Then, knowing the extreme elements allows for deducing the values of the other elements. If the values of the extreme elements are unknown, two theoretical elements are introduced corresponding to the bounds of what is to be evaluated.

FIGURE 1.8 – Operation of the CTJ

The major issue with this model is determining how to select the trios that provides the most information to the system.

2 Work Performed

2.1 Organization

2.1.1 Semi-Agile Method

During the internship, we followed a semi-agile method. Every Wednesday and Friday, we spent an hour with our mentors discussing our progress, forecasts, and any problems encountered, with all the interns, whether from ISIMA or not. We also exchanged ideas on each project to brainstorm solutions for overcoming obstacles.

To facilitate this, we used Trello, an application that allows us to track the progress of multiple projects simultaneously and create schedules. We categorized our information into three sections : To Do, In Progress, and Completed. Trello was dynamically updated at the start or end of each task.

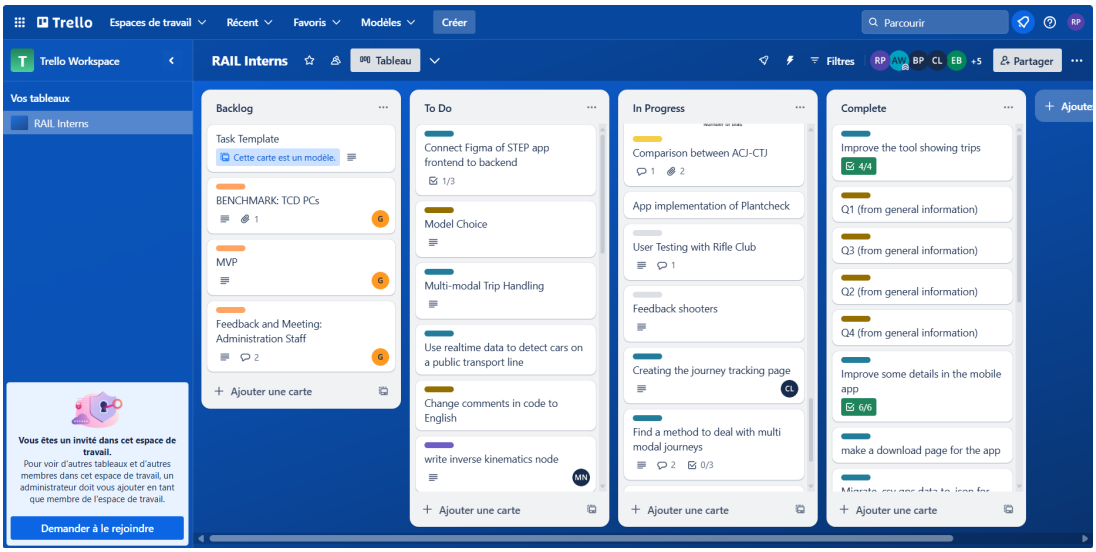


FIGURE 2.1 – Example of Trello

2.1.2 Internship Planning

During the course of the internship, I created two Gantt charts. Initially, I planned to dedicate more time to implementing models and did not anticipate adding graphical interfaces or studying element information. Furthermore, to optimize my time during the internship, I decided to write my report in parallel with the project, rather than allocating a specific week to it.

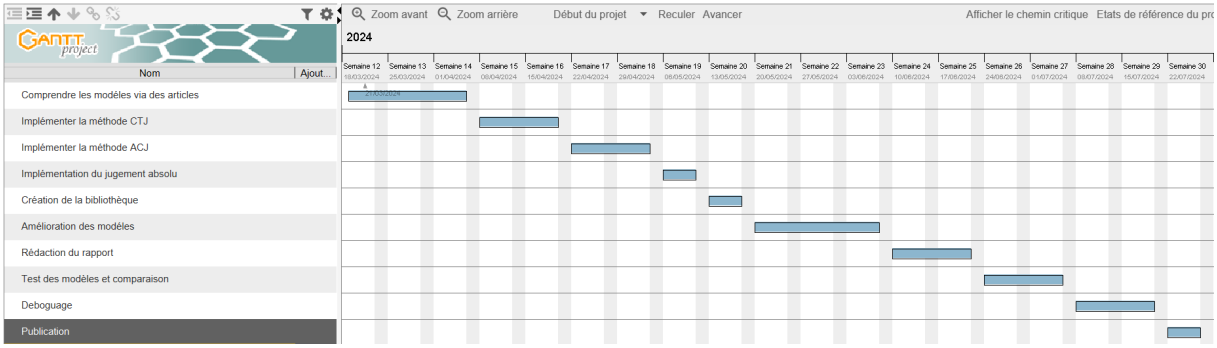


FIGURE 2.2 – Projected Gantt Chart

Originally, I intended to start directly with the implementation of the library and then improve the models coded. However, one of last year's projects had not been fully completed, so my mentor proposed that I finish a website by implementing the CTJ model and then focus on developing the library. Thus, before fully starting my project, I had to acquire skills in web application development, which I had never done before. However, with a background in HTML, I was able to acquire the necessary skills within a week, complete the website, and then proceed to develop the library.

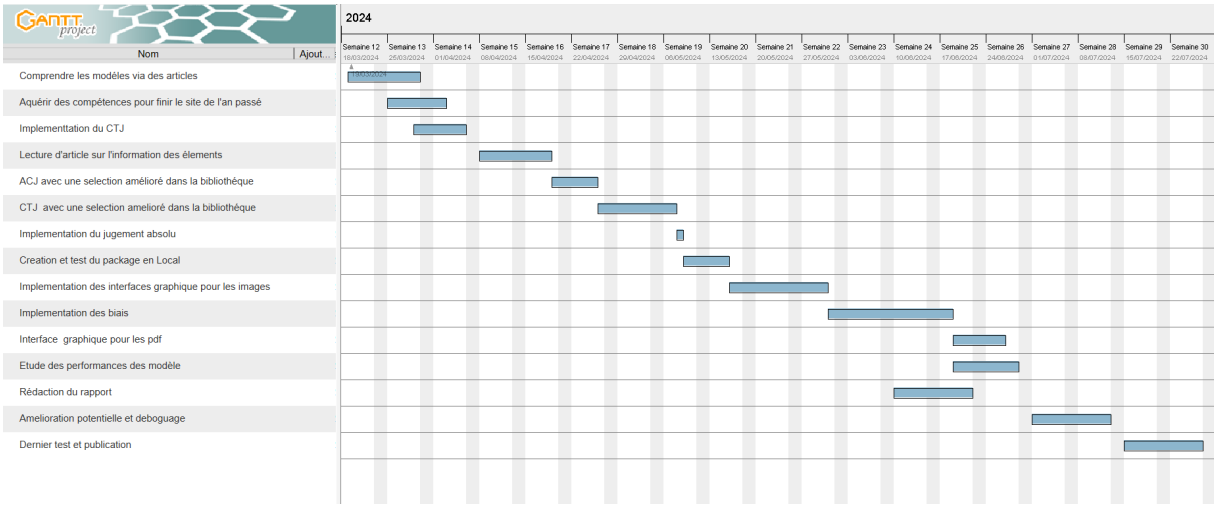


FIGURE 2.3 – Actual Gantt Chart

Some tasks were subdivided into smaller tasks. Some took more time, such as researching methods for comparing elements. Conversely, others were completed more quickly, for example, the implementation of absolute judgment. I also used the remaining time to develop two distinct graphical interfaces, which were not initially planned : one for viewing images and another for viewing PDF files.

2.2 Problem Study

While implementing models that are either present in the literature or directly from the creator of one of the models was straightforward, a question quickly arose : how can we compare two models if their optimizations and biases are not equivalent ? Thus, it was necessary, initially, to consider how to optimize the models that could be optimized.

2.2.1 Selection of Elements

In this case, optimization was performed on the ACJ and CTJ models. Indeed, absolute judgment does not require optimization, as it has a finite number of iterations corresponding to the number of elements to be evaluated. Each iteration involves evaluating a different element. The issue, therefore, arose only for the other two models. This optimization involves selecting the elements to compare to allow evaluators to provide easy and high-quality judgments.

Thus, two versions of ACJ will be implemented : the classic version where the information is calculated using a specific formula and elements are specifically selected, and a variant that uses the same process as CTJ.

For Classic ACJ

To use the classic ACJ method, we will use the probability that an element, e_1, is preferred over another element e_2, given their values, v_1 and v_2, respectively [4].

$$P(e_1 \text{ is better than } e_2 | v_1, v_2) = \frac{\exp(v_1 - v_2)}{1 + \exp(v_1 - v_2)} \tag{2.1}$$
(Probability that one element is better than another)

Politt (2011) [5] defines the information provided by a judgment as the product of the probability defined above 2.1 and that of the inverse event.

$$I = p(1-p) \tag{2.2}$$
(Information of a judgment defined by Politt)

It is noted that the maximum information according to Politt is achieved with a probability of 0.5.

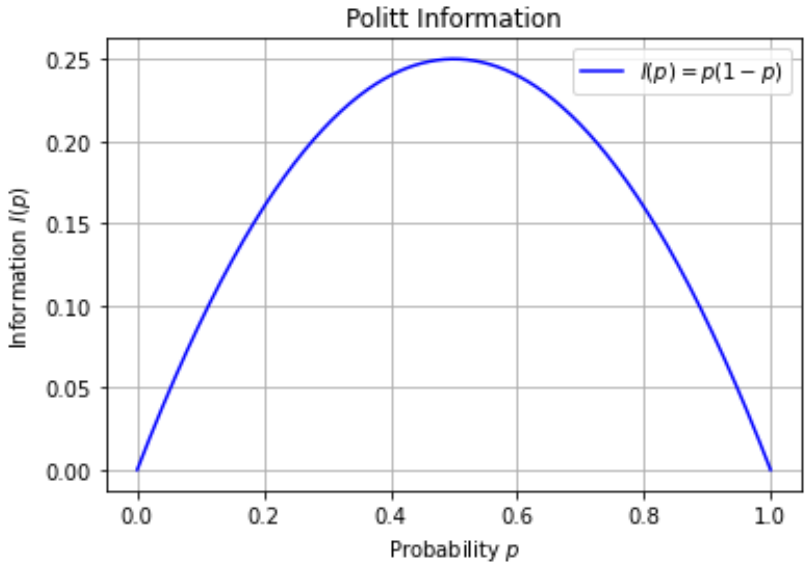


FIGURE 2.4 – Information according to Politt

Thus, for the optimization of ACJ, it is necessary to select the pairs that provide the most information to the system.

It is observed that the probability value that yields the maximum information corresponds to the scenario where the values of the two elements are equal. Indeed, if $v_1 = v_2$, equation 2.1 simplifies to :

$$P(e_1 \text{ is better than } e_2 | v_1, v_2) = \frac{1}{2} \tag{2.3}$$
(Probability that one element is better than another if $v_1 = v_2$)

Thus, the closer two elements are to each other, the more information according to Politt is provided. Therefore, the new pair selected will be the one where the probability 2.1 is closest to 0.5.

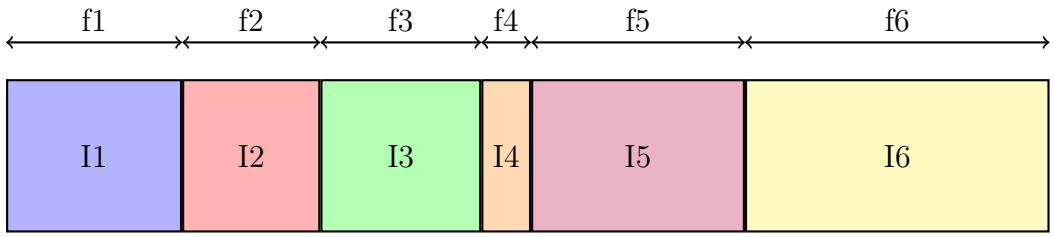
Roulette Selection and Shannon Entropy

This selection of elements, for the variant of ACJ and for CTJ, is done using a roulette selection on the n-tuples to be compared, in order to introduce randomness. Indeed, judgments providing the most information are not always the best to make.

A roulette selection consists, as the name suggests, of "spinning" a virtual roulette with differently sized segments. Therefore, it is necessary to think about the calculation of the lengths of these segments [6].

Let there be 6 individuals $I1, I2, I3, I4, I5, I6$. We define the fitness of each individual with an eponymous function. It corresponds to the "thickness" of each individual.

Individual	Fitness
I1	f1
I2	f2
I3	f3
I4	f4
I5	f5
I6	f6



Next, a virtual roulette is spun to select an individual :

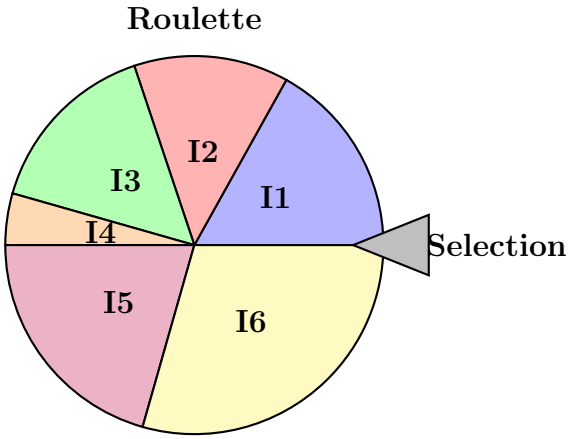


FIGURE 2.5 – Roulette selection

To do this, it is necessary to understand what the quantity of information of an element is for a series of judgments.

We use Shannon entropy to specify this quantity of information provided by an element. We define p_i as the probability of drawing element i among all the elements already compared. We denote X_i as the event associated with this draw.

In our case, we are dealing with a binary event : either this element is drawn, or it is not. Thus, we can write Shannon entropy as follows [7] :

$$H(X_i) = -p_i \log p_i + (1 - p_i) \log(1 - p_i) \quad \text{(Shannon entropy)} \tag{2.4}$$

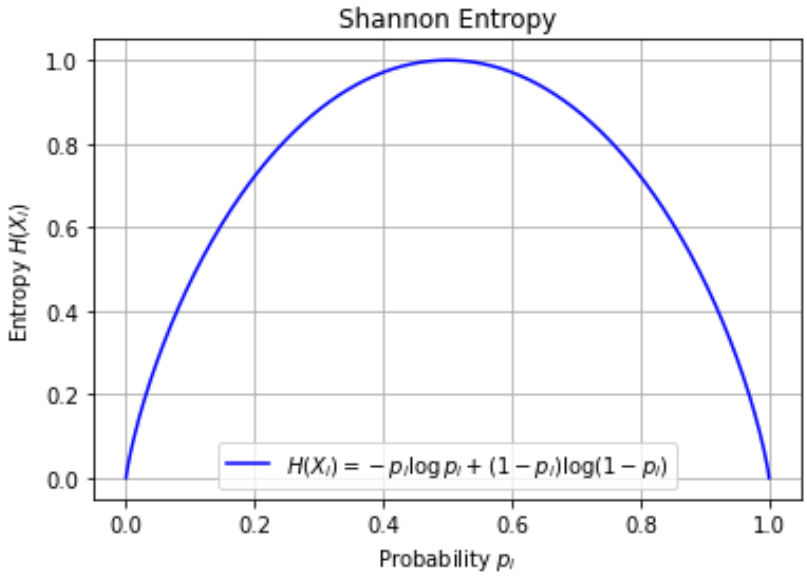


FIGURE 2.6 – Shannon entropy (quantity of information)

For the ACJ Variant

For the ACJ variant, it is important to know the amount of information (in the sense of information theory) provided by the evaluated pair. There is a formula that allows us to determine the mutual information, which is the amount of information a pair of elements brings to the system.

$$MI(X_i; X_j) = H(X_i) + H(X_j) - H((X_i, X_j)) \quad \text{(Mutual information)} \tag{2.5}$$

Where $H(X)$ corresponds to the previously defined Shannon entropy. And where $H((X, Y))$ corresponds to the joint entropy [7]. Here, the event X_i corresponds to drawing element i among the already made judgments, and similarly for X_j with element j.

$$H((X_i, X_j)) = -\sum_{x_i} \sum_{x_j} p(x_i, x_j) \log p(x_i, x_j) \quad \text{(Joint entropy)} \tag{2.6}$$

Given that closer elements provide more additional information (in Politt's sense), I define the length of a case (fitness) of the roulette as follows, with each case corresponding to a pair. For the pair (i, j), we have :

$$\begin{gathered} f(i,j) = \mathrm{MI}(X_i, X_j) \cdot \mathrm{Prox}(i,j)^2 \\ \text{(Fitness function for roulette selection for ACJ)} \end{gathered} \tag{2.7}$$

With $Prox(i, j)$, the proximity of two elements is defined as follows :

$$\begin{gathered} Prox(i,j) = \frac{MAX\_VALUE}{|estimated\_values_i - estimated\_values_j|} \\ \text{(Proximity of elements } i \text{ and } j) \end{gathered} \tag{2.8}$$

With MAX_VALUE being the maximum theoretical value associated with the dataset, and $estimated\_values_i$ corresponding to the estimated value for element i from the last iteration, similarly for $estimated\_values_j$ with element j.

Thus, if the roulette lands on a case, it means the corresponding pair has been selected.

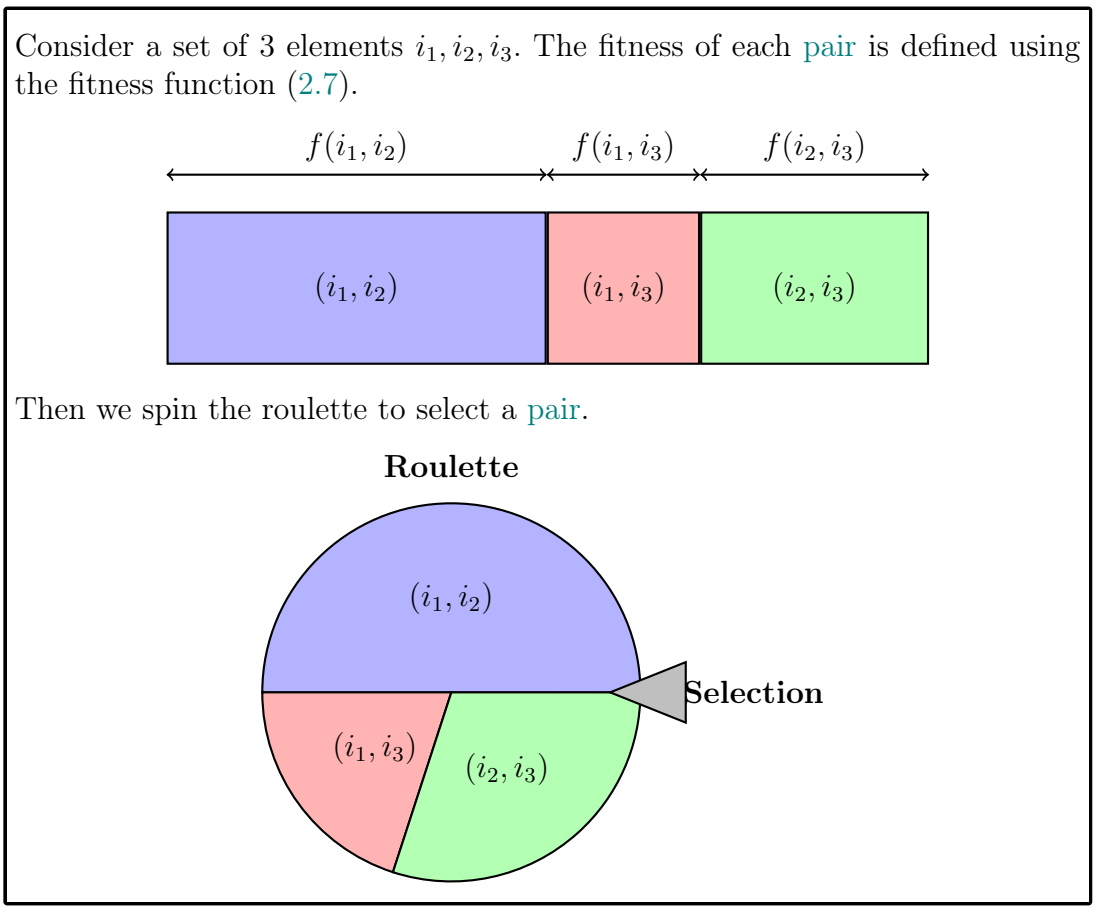


FIGURE 2.7 – Roulette selection for ACJ

For CTJ

Thus, for the CTJ, it is important to know the amount of information brought by the evaluated trio. To do this, we use interaction information. This involves determining the amount of information brought by an n-tuple to the system. In our case, $n = 3$, I use the following formula adapted to trios.

$$\begin{aligned} II(X_i, X_j, X_k) = {} & H(X_i) + H(X_j) + H(X_k) \\ & - (H((X_i, X_j)) + H((X_i, X_k)) + H((X_k, X_j))) \\ & + H((X_i, X_j, X_k)) \\ & \text{(Interaction information)} \end{aligned} \tag{2.9}$$

Where $H(X)$ corresponds to Shannon entropy, $H((X, Y))$ corresponds to joint entropy defined previously. And where $H((X, Y, Z))$ is defined as follows [8]. Here, the event

X_i corresponds to selecting element i among the judgments already made, likewise for X_j with element j, and X_k with element k.

$$H((X_i, X_j, X_k)) = -\sum_{x_i}\sum_{x_j}\sum_{x_k} p(x_i, x_j, x_k) \log p(x_i, x_j, x_k) \tag{2.10}$$

The difficulty in selecting the thickness of the section in roulette selection lies in defining the distance between three elements. I found that the mutual proximity of the three elements does not produce satisfactory results. Through our experiments, I found that maximizing the Euclidean distance for a point in three dimensions seems more appropriate.

$$\begin{aligned} Prox(i,j,k) =& ((estimated\_values_i - estimated\_values_j)^2 \\ &+ (estimated\_values_j - estimated\_values_k)^2 \\ &+ (estimated\_values_i - estimated\_values_k)^2)^{\frac{1}{2}} \\ &\text{(Distance between three elements)} \end{aligned} \tag{2.11}$$

Therefore, I define the thickness of a section as follows :

$$\begin{aligned} f(i,j,k) &= \mathrm{II}(X_i, X_j, X_k)^2 \cdot \mathrm{Prox}(i,j,k) \\ &\text{(Fitness function for roulette selection for CTJ)} \end{aligned} \tag{2.12}$$

Thus, if the roulette lands on a section, it means that the corresponding trio has been selected.

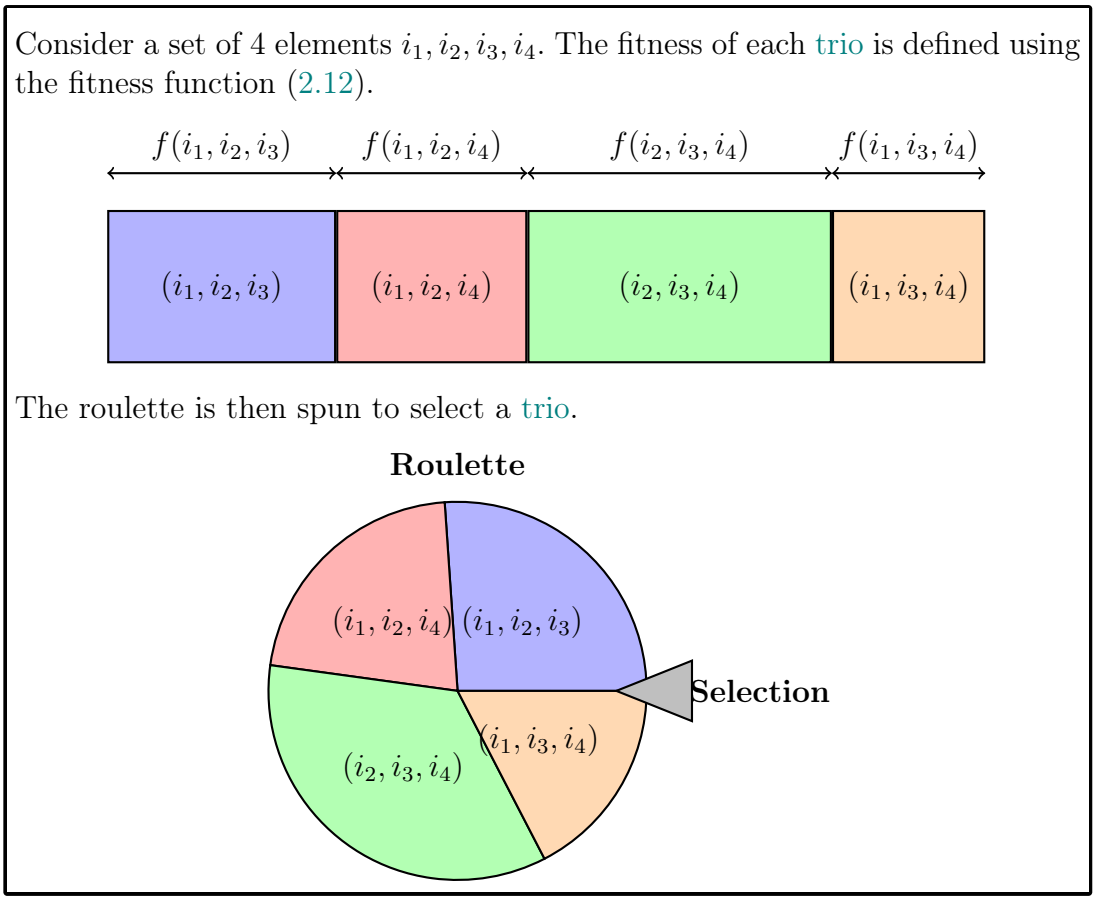


FIGURE 2.8 – Roulette selection for CTJ

2.2.2 Introduction of Bias

Since the library developed is intended for testing, it was important to consider how to introduce biases to compare the performance of algorithms in the presence of user errors.

For element comparisons, it is necessary to consider a perception threshold for users. Below this threshold, errors are allowed. To reflect the fact that the closer elements are, the more likely they are to be confused, I used a sigmoid function (2.13).

$$\sigma_\lambda(x) = \frac{1}{1 + e^{-\lambda x}} \tag{2.13}$$

Thus, when I refer to the perception threshold later, it will correspond to the value of the difference between two values that will have a 0.1 probability of inversion. Below this threshold, I consider that the simulated user does not confuse the elements.

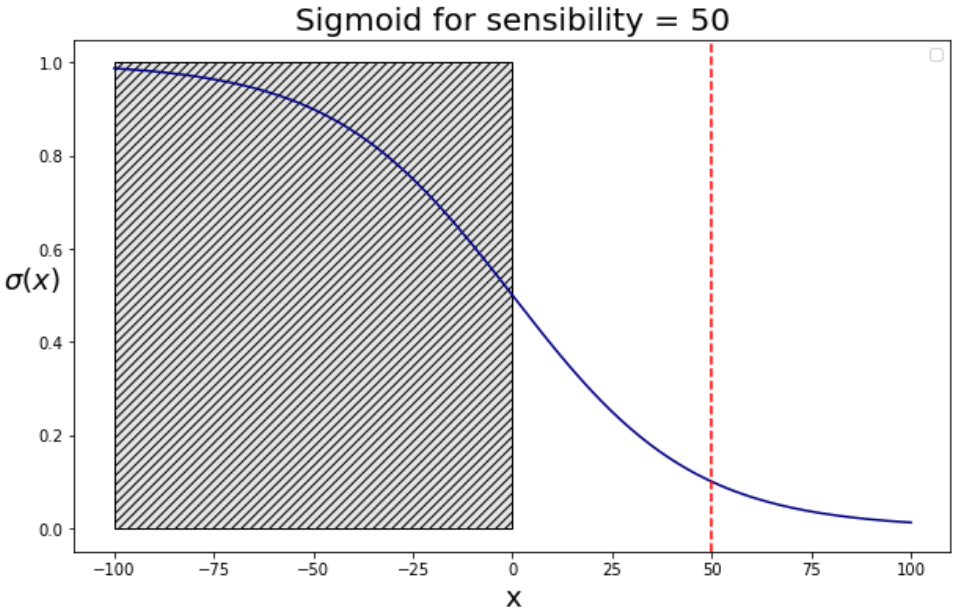
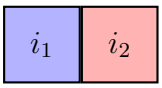


FIGURE 2.9 – Sigmoid function for $\lambda = \frac{\log\left(\frac{1}{9}\right)}{50}$

For the ACJ

In the case of the ACJ, the process is straightforward : bias is introduced solely by the inversion of two elements, which naturally fits the previous definition. Thus, it is enough to provide a perception threshold value to each simulated judge, and then I calculate the probability that the two elements are reversed.

Consider two elements, i_1, i_2 with the correct order being i_1, i_2.



A biased judgment corresponds to :

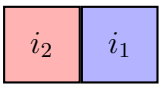


FIGURE 2.10 – Bias for the ACJ

The introduction of bias is the same for both the classic version of the ACJ and its variant.

For the CTJ

However, in the case of the CTJ, a problem arises : since we are evaluating three elements, the pure definition of the perception threshold is difficult to satisfy. However, we want the introduction to be straightforward, so we aim to maintain a single value for the bias of element inversion. Therefore, we will proceed as follows.

Consider three elements, i_1, i_2, i_3 with the correct order being i_1, i_2, i_3. Let $p_{i,j}$ denote the probability of reversing elements i and j.
All possible inversions are as follows ; we will analyze each case.

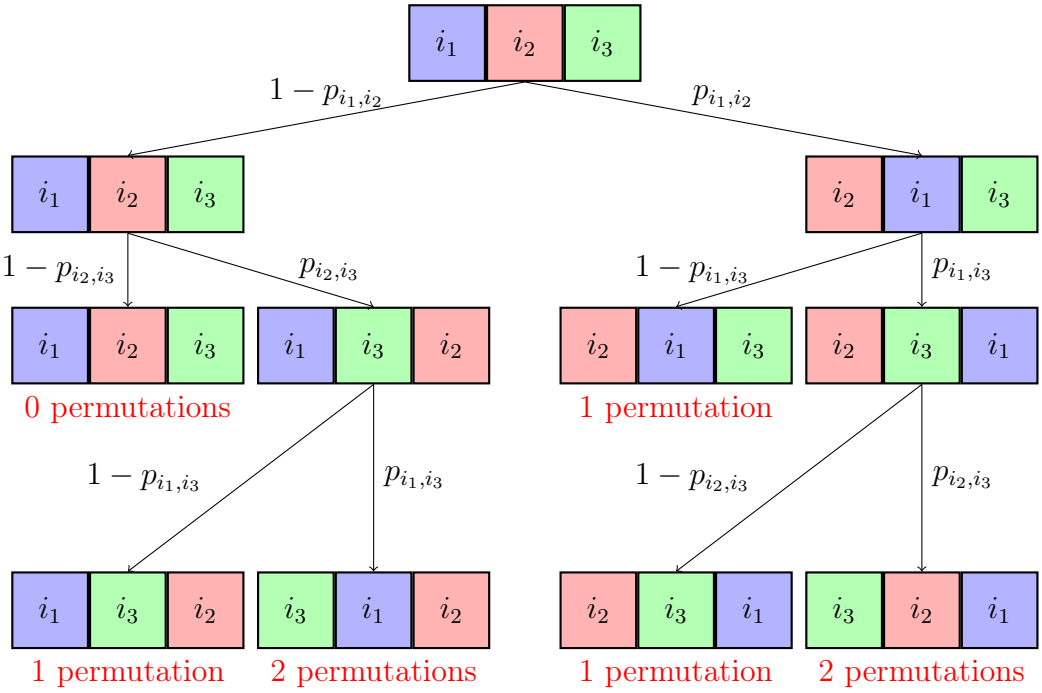
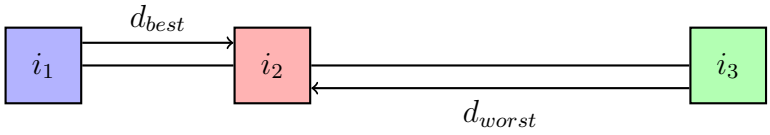


FIGURE 2.11 – Bias Tree

In a single judgment, you can introduce 0, 1, or 2 inversion errors. However, judgment bias is not the only type of error that can be introduced ; it is also possible to make a mistake in the scale.

Consider three elements, i_1, i_2, i_3 with the correct order being i_1, i_2, i_3. Let the distances be d_{worst} and d_{best}.

Thus, the correct evaluation is as follows :



A scale error corresponds to the introduction of a bias b with a probability p_b :

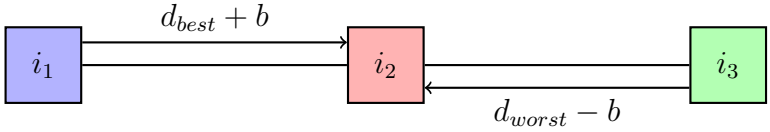


FIGURE 2.12 – Scale Error for the CTJ

This scale bias will be expressed with two factors : the first is the absolute value of the error in the scale, and the second is the probability of making this error. Thus, the bias in CTJ will be defined as a trio, where the first element is the perception threshold, the second is the absolute value of the error in the scale, and the last is the probability of the error.

For Absolute Judgment

For absolute judgment, introducing bias is much simpler : it involves a pair, with the first element being the absolute error introduced on the items, and the second element being the probability of making this error.

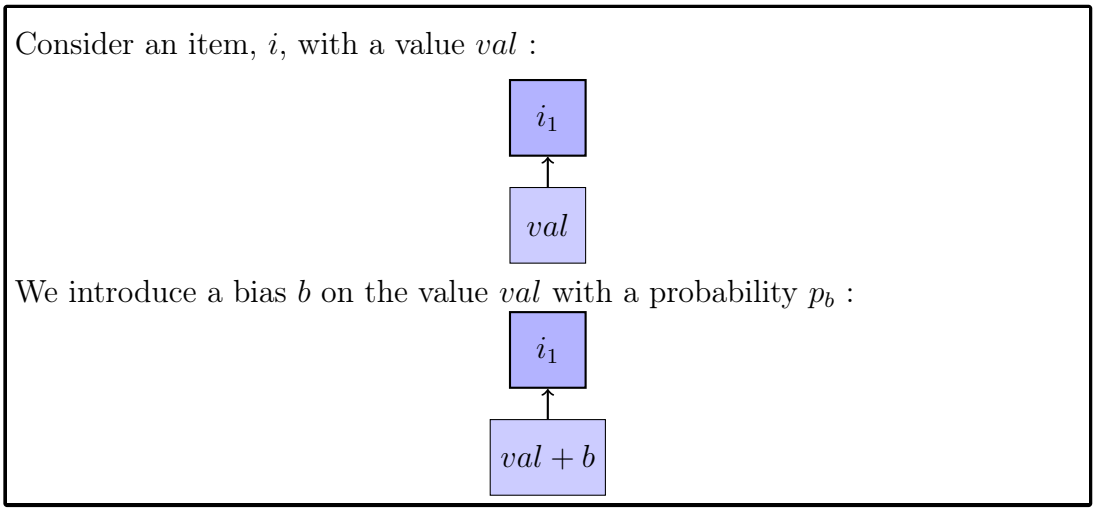


FIGURE 2.13 – Bias for Absolute Judgment

2.3 Tools

To have an open source project and library, it is necessary to use libraries that are themselves open source and allow for proper redistribution.

2.3.1 Libraries Used

A Python library is a collection of modules accessible to users. They can import it to use the functions implemented in it.

The choice of librarys was influenced to ensure that their licenses aligns with an open source project.

- **NumPy**

  NumPy is a library that defines standard mathematical operations, such as exponential, logarithm, and square root. It is also used when handling a large number of values, as it allows these values to be grouped into structures like arrays or matrices and provides operations between these structures. The choice of NumPy is due to performance reasons : operations on these structures are much faster than on native Python lists.

- **SciPy**

  SciPy is an open source library that extends the capabilities of NumPy. It uses the functions predefined by NumPy to define more complex mathematical functions. It will be used mainly for performing standard statistical calculations.

- **Random**

  The Random library is used to generate pseudo-random numbers. It provides functions for random operations, such as selecting a random element from a list, generating pseudo-random numbers following different distributions (uniform, Gaussian, etc.). This library will be used to perform draws and implement roulette selection.

- **Scikit-Learn**

  Scikit-Learn is an open source library dedicated to machine learning. It provides simple and efficient tools for predictive modeling, including classification, regression, clustering, and dimensionality reduction. Scikit-Learn is based on NumPy and SciPy. It will be used to rearrange values between two theoretical extreme values.

- **Itertools**

  The itertools library provides tools for creating and manipulating iterators. It includes functions for permutations, combinations, Cartesian products, and other iterative operations. This library will be used to generate combinations of elements.

- **Choix**

  The Choix library is a tool for implementing choice models, such as the Bradley-Terry model, used for ranking items based on preferences. It will be used to determine the estimated values during ACJ.

- **Time**

  The time library allows for time manipulation. It provides functions to get the current time, measure durations, and create pauses in code execution. It will be used to time the performance of users during judgments.

- **os**

  The os library provides functions to interact with the operating system. It allows operations such as file and directory management, executing system commands, and accessing environment variables. It will be used to access graphical representations (images or PDFs) of the objects to be evaluated.

- **Tkinter**

  Tkinter is a library for creating interactive graphical interfaces. Tkinter will be used to allow users to make judgments with the graphical representations of the elements imported via os.

- **Pillow**

  Pillow is a library for image processing. It allows opening, manipulating, and saving various image formats. It will be used to display images in the interface created with Tkinter.

2.3.2 Library Creation

To create a library, a particular structure must be followed.

Directory Structure

The first step in creating a Python library is to correctly structure the project's directory. Here is the structure I used.

```
CTJ/
|-- CTJ/
|   |-- __init__.py
|   |-- ACJ.py
|   |-- assessment_method.py
|   |-- CTJ.py
|   |-- Rubric.py
|   |-- selection.py
|   |-- util.py
|-- Notebook/
|   |-- ACJ-Tutorial.ipynb
|   |-- black.png
|   |-- CTJ-Tutorial.ipynb
|   |-- g1.png
|   |-- g2.png
|   |-- g3.png
|   |-- g4.png
|   |-- g5.png
|   |-- Rubric-Tutorial.ipynb
|   |-- white.png
|-- fix_import_error_tkinter.md
|-- LICENSE
|-- MANIFEST.in
|-- README.md
|-- setup.py
```

- `CTJ/` is the main directory of the library.
  - `__init__.py` : File that allows this directory to be treated as a Python package.
  - `ACJ.py`, `assessment_method.py`, `CTJ.py`, `Rubric.py`, `selection.py`, `util.py` : The various modules of the library.
- `Notebook/` is the directory containing examples and tests for the library.
  - `ACJ-Tutorial.ipynb`, `CTJ-Tutorial.ipynb`, `Rubric-Tutorial.ipynb` : Unit tests for the different modules.
  - `*.png` : The PNG files correspond to the images that will be displayed, here they are grayscale images.
- `fix_import_error_tkinter.md` : A file explaining how to install `tkinter` if it is not installed natively, shown in an exception generated by the library.
- `LICENSE` : License file for the project.
- `MANIFEST.in` : File providing instructions on which files to include in the library.

- `README.md` : File containing the project description.
- `setup.py` : Configuration script for installing the library.

The functions available to users are ACJ, Rubric, CTJ defined in the respective Python files. And ACJ_assessment_image, CTJ_assessment_image, Rubric_assessment_image, ACJ_assessment_pdf, CTJ_assessment_pdf, Rubric_assessment_pdf defined in `assessment_meth` `py`.

PIP

PyPI allows for simple sharing of libraries, which is why it was considered for use.

To install a library with PIP, you can use the following command in a console environment :

```
pip install 'library_name'
```

Wheel

The Wheel format is a standard for distributing packages on PIP. To do this, you need to create a '.whl' file associated with the library. You can then distribute this file or upload it to PyPI to make it available via PIP.

Source Distribution (sdist)

The source distribution ('sdist') is a format for distributing the source code of your library. Here's how to create a source distribution for the library : This command will create a '.tar.gz' file in the 'dist/' directory. This file contains all the necessary files to install the library from the source code. You can distribute this file or upload it to PyPI to make it available via PIP.

Locally, you can use this tar.gz file to install the library.

3 Results

The library has been fully developed. In this section, I will focus on its functionality and future issues.

There are two main ways to use it : automatic evaluation and manual evaluation.

In the following, to illustrate our results, we will use the following set of items :

```
colors = ['g1', 'g2', 'g3', 'g4', 'g5']
```

The list 'colors' represents the names of the items we want to evaluate, where their names are g_i with i ranging from 1 to 5, representing 5 shades of gray. We also define 'values', the value associated with each shade using the gray level for an 8-bit color, where 0 corresponds to black and 255 to white.

```
values  = [160 , 106 , 209 ,  80 ,  135]
```

We thus also define the two theoretical extreme values, black with the smallest value (the worst item) and white with the largest value (the best).

```
worst     = [0, 'black']
best     = [255, 'white']
```

3.1 Automatic Evaluation

Automatic evaluations allow for rapid testing of models while introducing biases to compare the behavior of models influenced by judges' errors.

However, to perform such evaluations, knowing the values of the items to be compared is essential. Therefore, in each function, you need to define the true_value parameter with the list of values for each item.

For the Classic ACJ Method

By default, it can be used as follows :

```
ACJ(worst, best, colors, true_values = values)
```

In this case, no bias is introduced, and the judgment is performed for a single judge. The maximum number of iterations is 30, and the model's precision is set to 0.95. The algorithm stops when one of these two conditions is met. You can introduce a bias and increase the number of judges as follows :

```
ACJ(worst, best, colors, true_values = values, nb_judge = 3,
sensibility = [20, 50, 90])
```

The 'sensibility' list corresponds to each judge's perception. For reference, the sensitivity threshold corresponds to the difference in value between two items that they might confuse with a probability of 0.1. After execution, the algorithm returns :

- the estimated values of each item
- the number of iterations
- the achieved precision
- the number of errors
- the time taken to perform the judgments

And displays the main results in the console. For example, with the last definition of ACJ :

```
================================================================
| Result of ACJ algorithm
| Items :  ['black', 'g1', 'g2', 'g3', 'g4', 'g5', 'white']
| True values :  [0, 160, 106, 209, 80, 135, 255]
| Estimated values :  [0, 188, 84, 218, 100, 150, 255]
| Accuracy :  0.9571026766588642
| Number of error :  [1. 2. 4.]
| Iteration :  29
================================================================
```

For the ACJ Variant

To use the ACJ variant, simply add the 'entropy' parameter as follows. By default, it is set to false, meaning that the basic version is used. Setting it to true uses the variant employing Shannon entropy.

```
ACJ(worst, best, colors, true_values = values, entropy = True)
```

The only function that changes is the one defining how to select the new pair of items.

```
ACJ(worst, best, colors, true_values = values, nb_judge = 3,
sensibility = [20, 50, 90], entropy = True)
```

```
============================================================
| Result of ACJ algorithm
| Items :  ['black', 'g1', 'g2', 'g3', 'g4', 'g5', 'g6', 'white']
| True values :  [0, 160, 106, 209, 80, 135, 234, 255]
| Estimated values :  [0, 165, 69, 194, 100, 135, 219, 255]
| Accuracy :  0.9607996175572444
| Number of error :  [1. 6. 6.]
| Iteration :  26
============================================================
```

For the CTJ Method

By default, it can be used as follows :

```
CTJ(worst, best, colors, true_values = values)
```

In this case, no bias is introduced, and the judgment is performed with a scale of 10, meaning there are 10 possible positions to place the central item between the two extreme items. The maximum number of iterations is 30, and the model's precision is set to 0.95. The algorithm stops when one of these two conditions is met. You can introduce a bias and change the scale as follows :

```
CTJ(worst, best, colors, true_values = values, scale = 30,
sensibility = (60, 2, 0.3))
```

The 'sensibility' is a tuple of three elements. The first element corresponds to the judge's perception threshold, the second is the absolute error value on the scale, and the third is the probability of making an error on the scale. In each case, the algorithm returns :

- the estimated values of each item
- the number of iterations
- the achieved precision
- the number of errors as a tuple, with the first element corresponding to the number of inversions and the second to the number of scale errors
- the time taken to perform the judgments

And displays the main results in the console. For example, with the last definition of the CTJ function :

```
============================================================
| Result of CTJ algorithm
| Items :  ['black', 'g1', 'g2', 'g3', 'g4', 'g5', 'white']
| True values :  [0, 160, 106, 209, 80, 135, 255]
| Estimated values :  [0, 147, 63, 212, 82, 134, 255]
| Accuracy :  0.956117726908126
| Number of inversion :  1
| Number of scale error :  1
| Iteration :  9
============================================================
```

For Absolute Judgment

By default, it can be used as follows :

```
Rubric(worst, best, colors, true_values = values)
```

In this case, no bias is introduced, and the result will correspond to the association of the true values with each item. You can introduce a bias as follows :

```
Rubric(worst, best, colors, true_values = values,
sensibility = (10, 0.5))
```

The ‘sensibility‘ is a tuple of two elements. The first element corresponds to the absolute error value on the true value, and the second to the probability of making this error. In each case, the algorithm returns :

- the estimated values of each item
- the number of iterations
- the achieved precision
- the number of errors
- the time taken to perform the judgments

And displays the main results in the console. For example, with the last definition of the ‘Rubric‘ function :

```
================================================================
| Result of Rubric algorithm
| Items :  ['black', 'g1', 'g2', 'g3', 'g4', 'g5', 'white']
| True values :  [0, 160, 106, 209, 80, 135, 255]
| Estimated values :  [0, 166, 110, 227, 93, 130, 255]
| Accuracy :  0.9871270390253974
| Number of error :  4
| Iteration :  7
================================================================
```

3.2 Manual Evaluation

To allow users to perform evaluations themselves, I have implemented graphical interfaces in the library. For this to work, there must be PNG files in the current directory with the same names as the list of items passed as arguments. For the ‘colors‘ list, there should be files such as ‘g2.png‘ in the current directory.

For the ACJ Method

In the case of ACJ, the two items will appear side by side, and you just need to click on the item that you think is better. For the colors, the value corresponds to the gray level ; the higher it is, the lighter the gray shade. Therefore, you need to select the lighter item.

FIGURE 3.1 – ACJ Interface for Image Visualization

For the CTJ Method

In the case of CTJ, the trio will appear, with a scale below it to select the distance of the central item from the two other items. To sort the items, simply click on the two items whose positions you want to swap and repeat the process until you believe the items are sorted. Again, based on our choice, we are sorting from lightest to darkest. Then, use the slider to position the central item at a distance from the left item. To proceed to the next judgment, simply click the Next button.

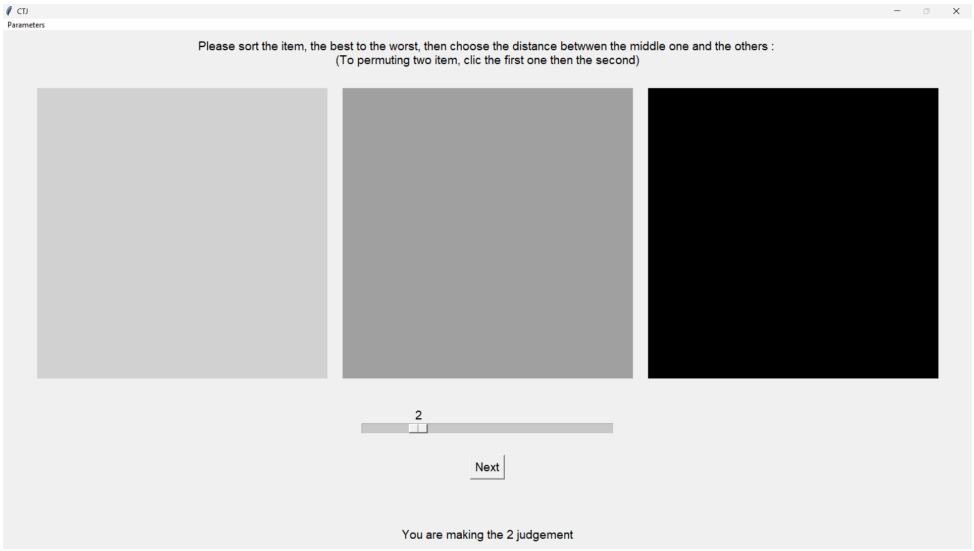


FIGURE 3.2 – CTJ Interface for Image Visualization

For Absolute Judgment

In the case of absolute judgment, the graphical representation of the item appears, with a text box below where the user can enter the presumed value of the item. To move to the next judgment, simply click the Next button.

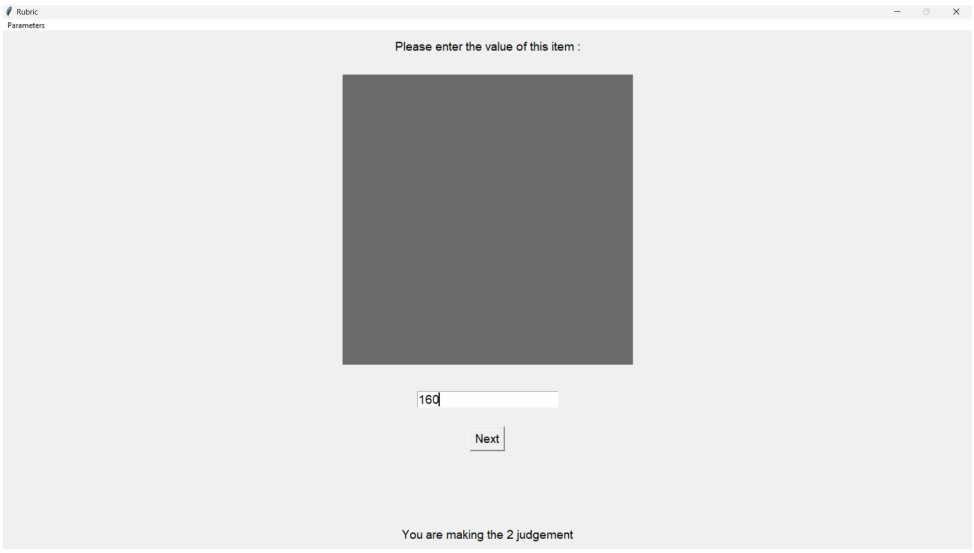


FIGURE 3.3 – Absolute Judgment Interface for Image Visualization

For PDF Files

I have also implemented evaluation methods for visualizing PDFs. The interface is similar, but an additional button is added below each image to open the PDF. The image displayed in the judgment is the first page of the PDF. Here's an example for CTJ.

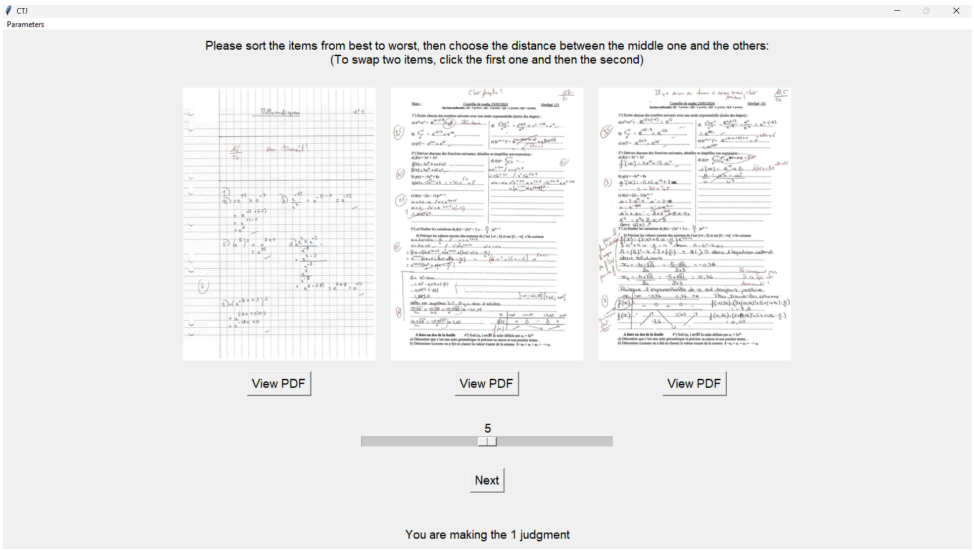


FIGURE 3.4 – CTJ Interface for PDF Visualization

There are two ways to use the methods with graphical interfaces. The first aims to test the effectiveness of the models during a judgment and compare the execution times of the judgments to see which method is the fastest. The second is the use of models under real conditions.

3.2.1 Model Testing

For model testing, in addition to the necessary elements, you need to include the values of the items, and you also need to specify the method used to perform the judgment.

Here is an example for CTJ :

```
CTJ(worst, best, colors, true_values = values,
assessment_method = CTJ_assessment_image)
```

3.2.2 Real Conditions

For real conditions, only the method used for judgment needs to be added; the actual values are no longer necessary, and the stopping condition is calculated based on the convergence of the results. To achieve this, it is recommended to increase the model's precision.

Here is an example for CTJ :

```
CTJ(worst, best, colors, true_values = values,
assessment_method = CTJ_assessment_image, max_precision = 0.99)
```

3.3 Preliminary Analysis of Models

3.3.1 Number of Errors

Initially, to analyze the behaviors of the ACJ and CTJ models, we will set similar errors, specifically, we will set an inversion threshold of 60, without modifying the scale value.

We will observe the precision and the number of iterations as a function of the number of inversion errors. We set the minimum precision to 0.95 and the maximum number of iterations to 100. We take an average over 1000 iterations.

First, we will plot the number of iterations of the models as a function of the number of errors.

Moreover, the models are studied here with the color list items defined in section 2; further tests with various datasets are needed to draw reliable conclusions.

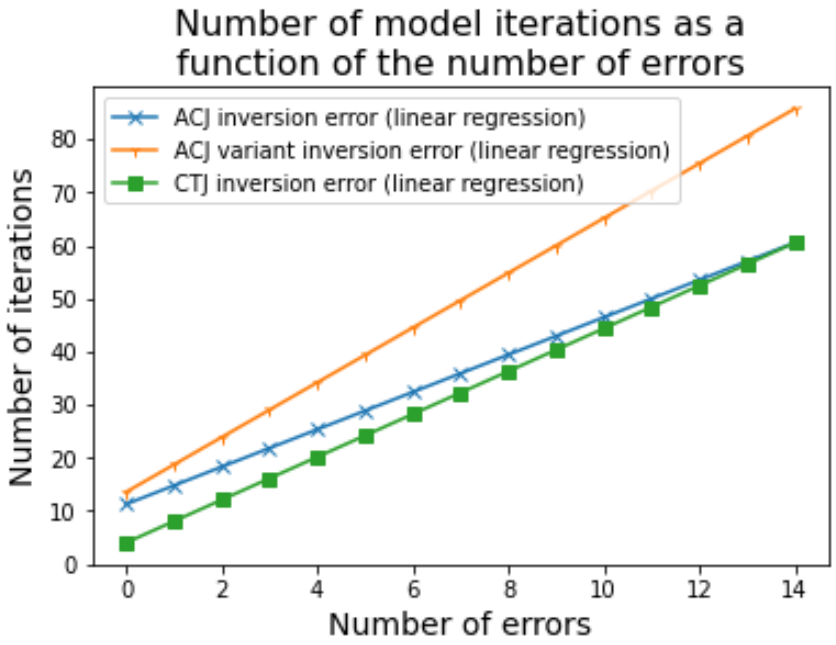


FIGURE 3.5 – Number of iterations of the models as a function of the number of errors

We notice that CTJ would have a lower number of iterations compared to both versions of ACJ for a number of errors below 9. After that, it seems to perform more iterations than the classic ACJ but fewer than its variant, which is optimized similarly to CTJ. Here, the algorithm never reached 100 iterations, leading us to conclude that these results are reliable and correspond to a precision of over 0.95.

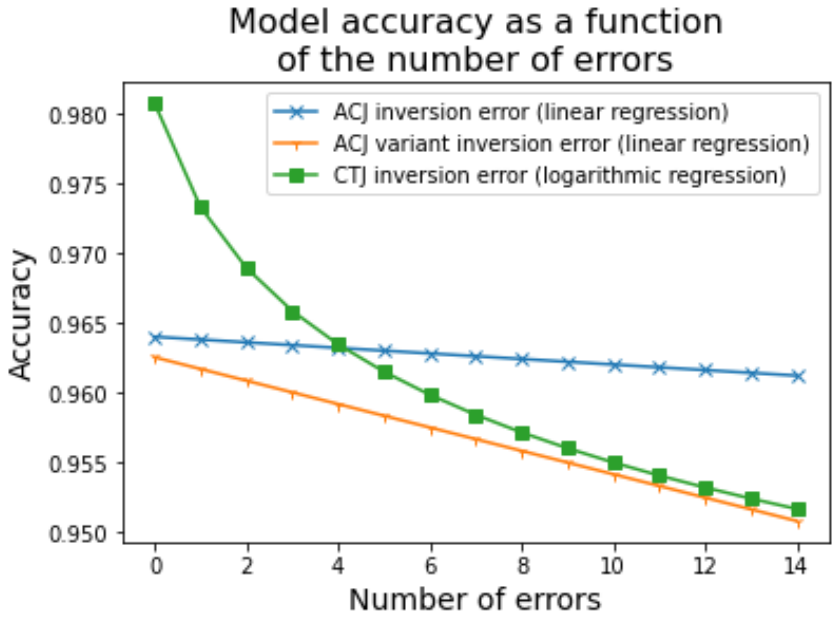


FIGURE 3.6 – Model precision as a function of the number of errors

Similarly, CTJ always has a higher precision than that of the ACJ variant. The precision of CTJ tends towards that of the ACJ variant as the number of errors increases. However, from 4 errors onwards, the precision of the classic ACJ is better than that of CTJ.

3.3.2 Perception Threshold

I have now varied the judges' perception threshold to evaluate the models' effectiveness with more or less competent judges. The maximum number of iterations is 200, and the minimum required precision is 0.95. When either of these two conditions is met, the algorithm stops. Perception levels are selected from 0 to 80 in intervals of ten.

Judgments are still performed with the color list defined in the previous section.

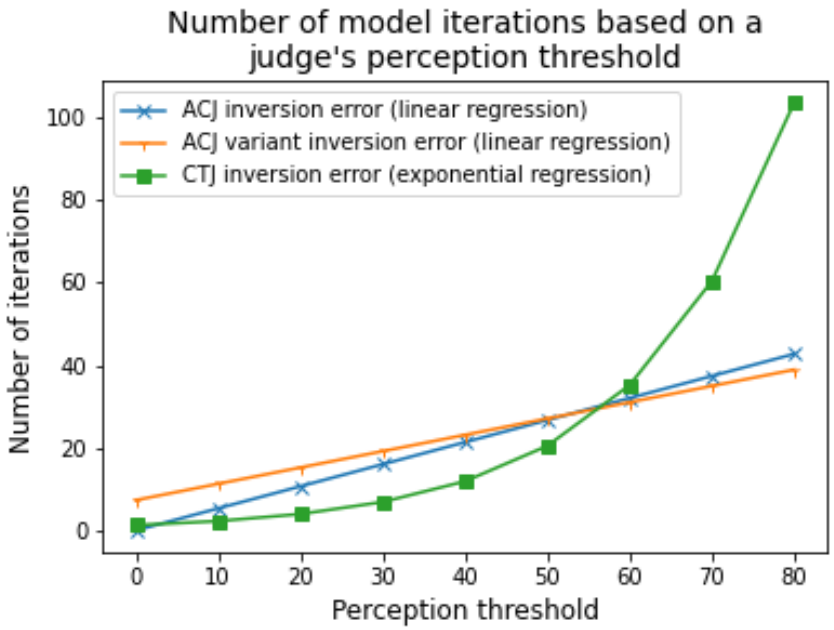


FIGURE 3.7 – Number of iterations of the models as a function of the judges' perception threshold

It can be observed that if the perception threshold is below 60, the number of iterations required by CTJ to achieve at least 0.95 precision is lower than for all other methods. However, beyond this value of 60, the number of iterations appears to increase sharply and exceeds that of the two other methods.

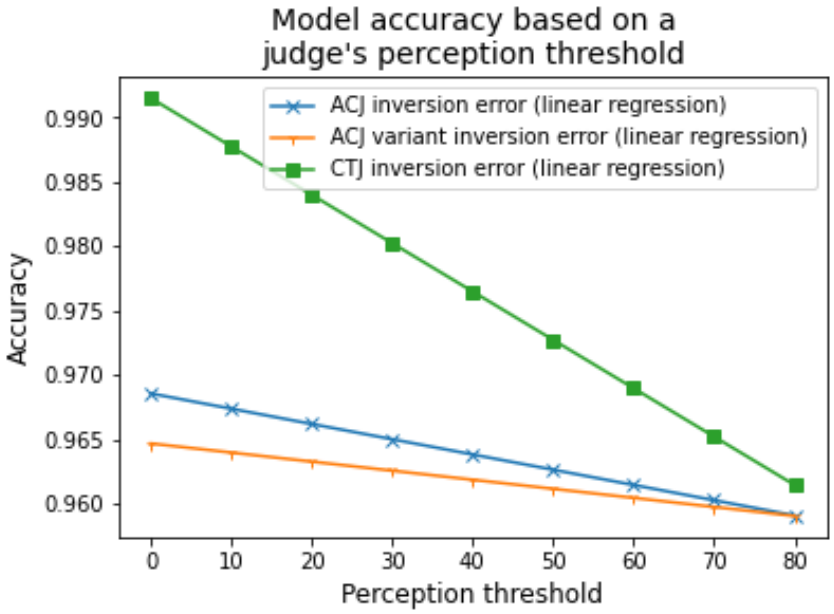


FIGURE 3.8 – Precision of the models as a function of the judges' perception threshold

As for precision, CTJ consistently shows higher precision compared to other methods, even though it tends to approach the precision of ACJ as the threshold increases.

3.4 To Do

The project was initially intended to be developed as an open source library, but as I finish writing this report, the objectives have shifted somewhat, and the project's license is still to be determined, as is the distribution method.

Indeed, without any research paper written on CTJ, the risk of the model being plagiarized or used for commercial purposes cannot be ruled out. Therefore, the library will likely be published later once a research paper on it is released.

You will also need to continue analyzing the models and attempt to find links between the error curves of ACJ and CTJ, while using various datasets.

Conclusion

I was tasked with creating a Python library for using and comparing three judgment methods : CTJ, ACJ, and absolute judgment. The goal was to have a library that could test the effectiveness of CTJ and allow for further detailed study.

I have indeed implemented it and also improved the element selection in the comparison methods to ensure that the optimization is similar, enabling a reasonable comparison between the models. Additionally, I started analyzing the models on datasets and implemented a way to introduce biases into automatic judgments. I also created graphical interfaces for evaluating images and PDFs.

The main difficulty encountered was improving element selection. Initially, the concept of an element's information was quite vague to me, but after researching, I believe I have understood this concept and the related concepts. This challenge was an excellent opportunity for me to delve into a mathematical field previously unknown to me : information theory.

Creating the library has also been enriching. It was my first time exploring how to publish a Python library. An interesting aspect was managing the project's license. Initially, the project was intended to be open source with a permissive license. However, this is unlikely to be the case. As I finish writing this report, the issue of the license is still under discussion.

The project's objective has been largely achieved. The only remaining step is publication and in-depth analysis of the implemented methods. Publication remains a major challenge I currently face. Since no research paper on the CTJ model has been published, releasing a library that uses it poses a problem, as the model might be used without proper credit or, worse, plagiarized. Furthermore, the license is now expected to be a non-commercial one, as per my supervisor's wishes.

Bibliographie

[1] I. B. Mullaney, "Comparative triple judgment as an assessment methodology," bai degree dissertation, Trinity College Dublin, the University of Dublin, Dublin, 2022.

[2] A. Pollitt, "The method of adaptive comparative judgement," *Assessment in Education : Principles, Policy & Practice*, vol. 19, no. 3, pp. 281–300, 2012.

[3] L. L. Thurstone, "A law of comparative judgment.," *Psychological review*, vol. 101, no. 2, p. 266, 1994.

[4] D. Andrich, "Relationships between the thurstone and rasch approaches to item scaling," *Applied Psychological Measurement*, vol. 2, no. 3, pp. 451–462, 1978.

[5] A. Pollitt, "Comparative judgement for assessment," *International Journal of Technology and Design Education*, vol. 22, 05 2011.

[6] A. Lipowski and D. Lipowska, "Roulette-wheel selection via stochastic acceptance," *Physica A : Statistical Mechanics and its Applications*, vol. 391, no. 6, pp. 2193–2196, 2012.

[7] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, New Jersey, USA : John Wiley & Sons Inc., 2nd ed., 2005.

[8] R. W. Yeung, "A new outlook on shannon's information measures," *IEEE Transactions on Information Theory*, vol. 37, no. 03, pp. 466–474, 1991.