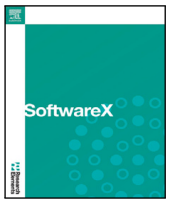
SoftwareX 31 (2025) 102239

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx



Original software publication

An Item Response Theory-based R module for Algorithm Portfolio Analysis

Brodie Oldfield [a], Sevvandi Kandanaarachchi [b,e,*], Ziqi Xu [c], Mario Andrés Muñoz [d,e]

[a] CSIRO's Data61, Eveleigh, NSW 2015, Australia
[b] CSIRO's Data61, Clayton, VIC 3068, Australia
[c] School of Computing Technologies, RMIT University, Melbourne, VIC 3000, Australia
[d] School of Computing and Information Systems, The University of Melbourne Parkville, VIC 3010, Australia
[e] ARC Centre in Optimisation Technologies, Integrated Methodologies, and Applications (OPTIMA), Carlton, VIC 3052, Australia

ARTICLE INFO

Keywords:
Item Response Theory
Algorithm evaluation
Machine learning
R language
R Shiny
Benchmarking portfolios

ABSTRACT

Experimental evaluation is crucial in AI research, especially for assessing algorithms across diverse tasks. Many studies often evaluate a limited set of algorithms, failing to fully understand their strengths and weaknesses within a comprehensive portfolio. This paper introduces an Item Response Theory (IRT) based analysis tool for algorithm portfolio evaluation called AIRT-Module. Traditionally used in educational psychometrics, IRT models test question difficulty and student ability using responses to test questions. Adapting IRT to algorithm evaluation, the AIRT-Module contains a Shiny web application and the R package `airt`. AIRT-Module uses algorithm performance measures to compute anomalousness, consistency, and difficulty limits for an algorithm and the difficulty of test instances. The strengths and weaknesses of algorithms are visualised using the difficulty spectrum of the test instances. AIRT-Module offers a detailed understanding of algorithm capabilities across varied test instances, thus enhancing comprehensive AI method assessment. It is available at https://sevvandi.shinyapps.io/AIRT/.

Code metadata

Current code version	R Package: 0.23 Shiny: 0.0.1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-24-00455
Permanent link to Reproducible Capsule	
Legal Code License	GPL-3.0
Code versioning system used	git
Software code languages, tools, and services used	R
Compilation requirements, operating environments & dependencies	
If available Link to developer documentation/manual	https://sevvandi.github.io/airt/index.html
Support email for questions	

1. Motivation and significance

Experimental evaluation is critical for AI research, especially for problems with elusive theoretical evaluation. AI researchers are interested in the performance of a particular method for a specific problem instance, across multiple instances, and against other methods. Evaluating a diverse set of algorithms across a comprehensive set of test instances contributes to an increased understanding of the interplay between instance characteristics, algorithm mechanisms, and algorithm performance. Such an evaluation helps determine an algorithm's strengths and weaknesses and provides a broad overview of the collective capabilities of an algorithm portfolio. However, many studies that evaluate only a small number of algorithms on a limited set of test instances fail to reveal where any algorithm belongs within a state-of-the-art algorithm portfolio's capabilities or where algorithms' unique strengths and weaknesses lie when considering a diverse range of test problem difficulties and challenges. In this paper, we present AIRT-Module, an Item Response Theory (IRT)-based analysis tool for evaluating a portfolio of algorithms.

* Corresponding author at: CSIRO's Data61, Clayton, VIC 3068, Australia.
E-mail addresses: brodie.oldfield@data61.csiro.au (B. Oldfield), sevvandi.kandanaarachchi@data61.csiro.au (S. Kandanaarachchi).

https://doi.org/10.1016/j.softx.2025.102239
Received 21 August 2024; Received in revised form 3 June 2025; Accepted 16 June 2025

IRT [1,2] is commonly used in educational psychometrics to analyse and model responses to test questions. The premise of IRT is that there is an underlying characteristic, such as verbal/mathematical ability, boredom proneness [3] or misinformation susceptibility [4] that is difficult to measure directly but can be modelled via responses to questions. In an educational setting an $N \times M$ matrix of marks from N students to M questions is the input to the IRT model. The items are the questions, and the response to the items is modelled using IRT. The outputs of an IRT model consist of question characteristics and student ability. For a 2-parameter IRT model the question characteristics are difficulty and discrimination of questions. Different IRT models are appropriate depending on the type of response: dichotomous, polytomous and continuous. Dichotomous models are used for binary responses such as true/false questions. Polytomous models are used for discrete-valued, ordinal responses commonly seen in surveys ranging from "strongly agree" to "strongly disagree" options. Continuous models are used for continuous-valued responses such as extended written responses in exams [1]. While research into educational psychometrics has used IRT since the 1960s, its use in machine learning is more recent [5–7]. The Algorithmic Item Response Theory (AIRT) framework is one such adaptation [8].

When evaluating a portfolio of algorithms we consider the performance of M algorithms to N test instances. The performance of algorithm i to a test instance j is a numerical value, such as predictive accuracy in classification. The AIRT framework adapts traditional IRT to algorithm evaluation by mapping algorithms to items and test instances to participants, resulting in equating algorithm performance values with student marks. With this mapping AIRT fits an IRT model and gains insights on the algorithm strengths and weaknesses.

AIRT-Module comprises an R package called `airt` which can handle continuous and polytomous models, and a Shiny web application called the AIRT Shiny App which can handle only continuous models. Using algorithm performance values as input, AIRT-Module computes an algorithm's anomalousness, consistency and difficulty limit, and the test instance difficulty. For a given problem set, the space of test instance difficulties constitutes the problem difficulty spectrum.

- **Anomalousness** is a boolean value flagged if an algorithm excels with difficult problems but struggles with easy problems.
- **Consistency** is a numeric value that indicates the stability of the performance. A low consistency algorithm gives fluctuating performance for datasets of similar difficulty, whereas a high consistency algorithm gives similar performance irrespective of dataset difficulty.
- **Difficulty Limit** is a numeric value that describes the highest difficulty level an algorithm can handle. A higher difficulty limit score means that the algorithm can handle harder problems.
- **Problem difficulty spectrum** is the one-dimensional space where test instance difficulty values reside, ranging from easy to hard. The strengths and weaknesses of algorithms can be visualised in this space.

In the R package `airt`, these attributes can be computed and the algorithm strengths and weaknesses plotted using continuous data, e.g., an algorithm's accuracy score, or polytomous (discrete) data, e.g. a grading system between A and F. Furthermore, AIRT has a simple set of tools for model validation that uses actual performance values and the performance according to the fitted AIRT model. The Shiny App introduces a dynamic user interface to `airt` with the functionality to upload datasets, transform datasets, change function parameters, and download the resultant plots. We developed the AIRT Shiny App to increase its availability to users unfamiliar with the R language.

2. Software description

The AIRT-Module operates under an Input ⇒ Model ⇒ Output system. The input is a dataset of performance values for a portfolio of algorithms to a diverse set of test instances. An IRT model adapted for algorithm evaluation is fitted to this data [8]. We will call this the AIRT model for the remainder of the paper. The output is the resulting model, its parameters and the created plots. While the R package has the functionality to fit polytomous performance data, our focus here is on continuous data such as classification accuracies (ranging in $[0, 100]$).

The AIRT Shiny App is built using the `airt` R package and has two interfaces: a walkthrough interface and a dashboard (See Fig. 2). The walkthrough interface is oriented in a presentation manner, where users are shown `airt` visualisations and analysis as sections. Each section is only rendered when a user chooses to continue and contains UI elements allowing plots to respond dynamically to user inputs. Each section includes an explanation of the plot and critical methods of analysis. In contrast, the dashboard interface generates all plots simultaneously and renders a plot at a time based on user preference.

2.1. Software architecture

2.1.1. Overview

Fig. 1 illustrates, at a high level, the AIRT-Module architecture, consisting of the R Package `airt`, which handles IRT and related computations, and the AIRT Shiny App made using R Shiny, which runs the R Package and renders the results to users. Users can run the `airt` R Package independently of the AIRT Shiny App.

2.1.2. R package

After suitably mapping algorithms and test problems to the IRT setting [8], the R packages `mirt` [9] and `EstCRM` [10] are used to fit an AIRT Model for polytomous and continuous data respectively. To allow for a broader range of algorithms, such as anomalous algorithms, parts of `EstCRM` code were modified [8]. After fitting the model, AIRT-Module computes algorithm attributes and finds the strengths and weaknesses using `latent_trait_analysis()`. The `airt` attributes are then used within `autoplot` to create plots.

2.1.3. AIRT Shiny App

AIRT Shiny App uses Shiny by posit, allowing a server to run R code and communicate with a user's session. Structurally, the AIRT_Shiny project directory contains a `UI.R` document, which houses the HTML/CSS/JavaScript the web page is scaffolded from, a `server.R` document, which contains the main rendering and logic functions, and utility documents to group related functions. Pre-computed datasets are under the ./Data directory and are loaded into `server.R` on server startup.

AIRT Shiny App also has an Input, Model, and Output pipeline. Inputs are datasets where users can upload their dataset in CSV format or use a pre-generated example dataset. The dataset is validated and modelled within relevant `airt` functions such as `cirtmodel()` and `latent_trait_analysis()`. Where possible and appropriate, the output of functions is cached. Plots are generated from the outputs and rendered by the UI.

2.1.4. Deployment

A release version of `airt` is available from the Comprehensive R Archive Network (CRAN) repository, while a development version is available in GitHub. Users can access the AIRT Shiny App at https://sevvandi.shinyapps.io/AIRT/, deploy it locally via RStudio or host it using a service that handles Shiny-compatible environments.

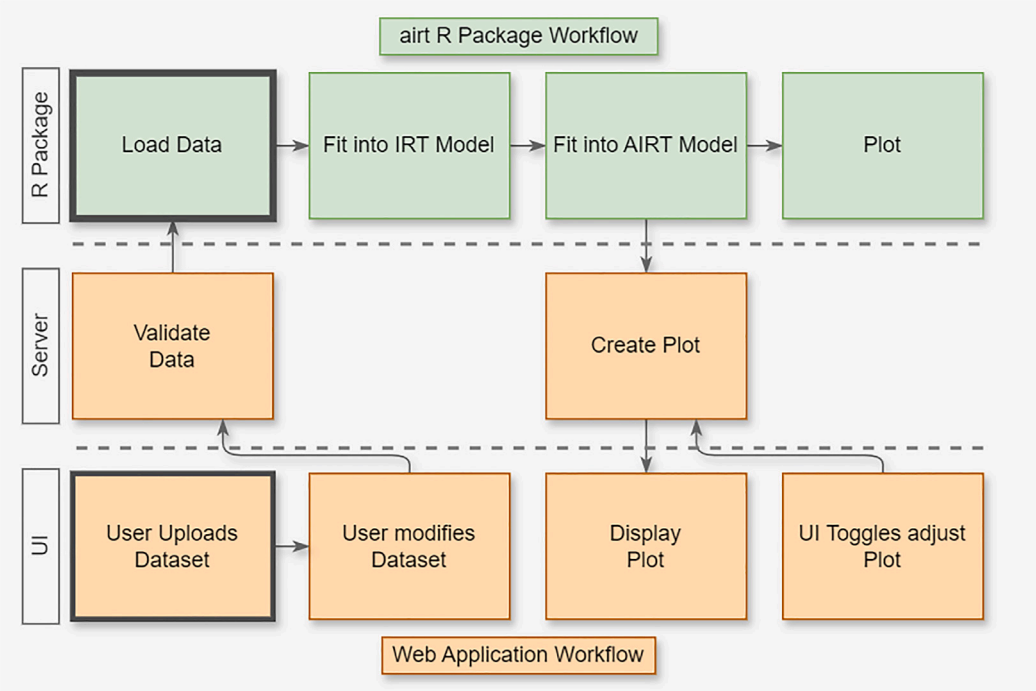


Fig. 1. AIRT-Module workflows: One (Green) showing the workflow when only using the `airt` R Package, the other (Orange) when using the AIRT Shiny App. Thick-bordered cells indicate starting actions in the workflow.

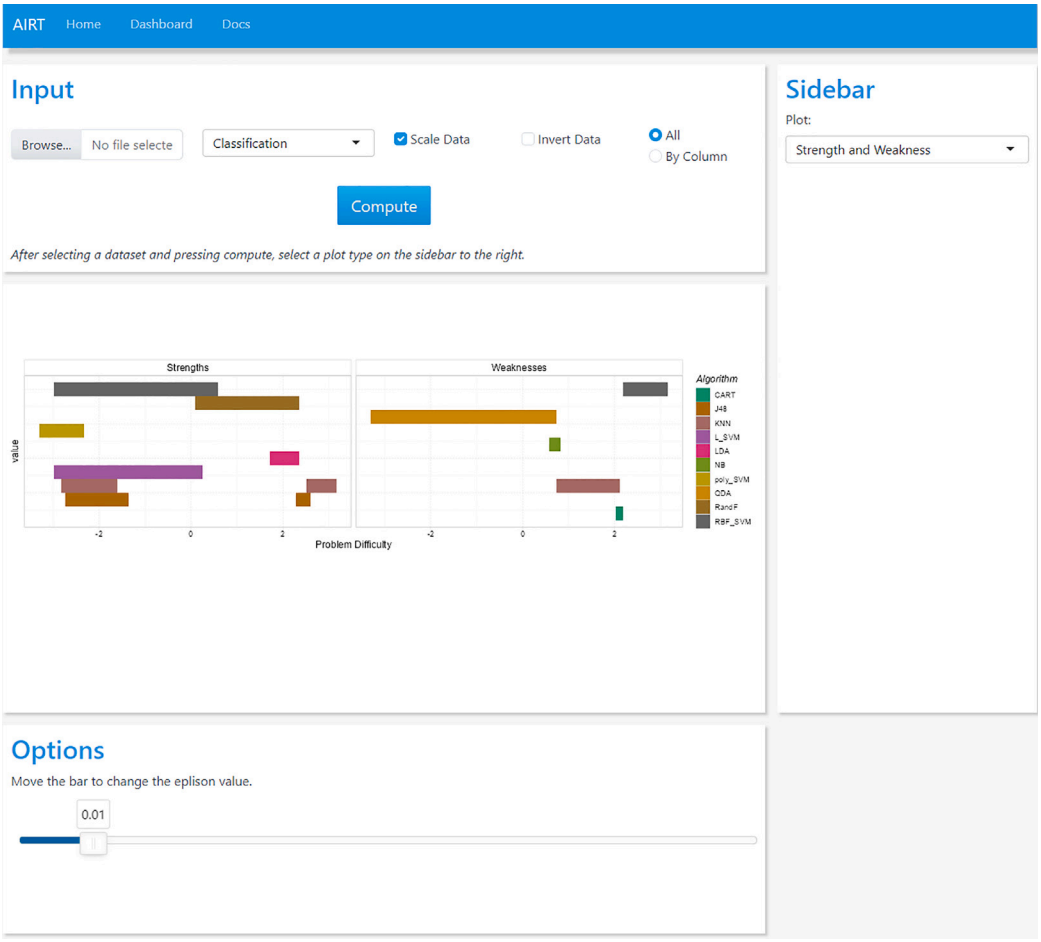


Fig. 2. Dashboard view of the AIRT Shiny App.

```
//CRAN Release Version
install.packages("airt")

//GitHub Development Version
install.packages("devtools")
devtools::install_github("sevvandi/airt")
```

Installation instructions for AIRT Shiny App are available on https://github.com/broldfield/AIRT_Shiny under the `README.md` file.

2.2. Software functionality

2.2.1. AIRT R package functions

Data is loaded into the IRT model using `cirtmodel()` for continuous data or `pirtmodel()` for polytomous data. Typically, functions suffixed or prefixed with 'c' or 'crm' are for continuous data, and functions with 'p' are for polytomous data. These functions accept a dataset as a data frame and output the IRT parameters relevant to `airt`. For `cirtmodel()`, this would be in the element `cirtmodel_output$model$param`.

The original data frame, the param element, and an epsilon value are used as parameters in `latent_trait_analysis()` to create the `airt` attributes. This output, denoted as LTA_output, is used in all `airt` plotting functions for analysis.

The `autoplot(object, plottype)` function is used to generate a plot using `ggplot` [11] based on the input object and the plot type specified. For LTA_output, there are four plot types of relevance. Plot types 1 and 2 show Performance against Problem Difficulty as scatter plots. Plot type 3 shows smoothing splines fitted to the performance values where the x axis denotes the problem difficulty. The smoothing splines are particularly important as the best-performing algorithm for a given problem difficulty will be the one whose spline is at the top. Type 4 generates a bar chart version of plot type 3, where the default setting corresponding to `epsilon` = 0 shows the best algorithm for every value in the problem difficulty spectrum. The `epsilon` value is a goodness threshold. When `epsilon` = 0, only the best algorithm for every problem difficulty value is considered. When `epsilon` = 0.01, algorithms with performance within 0.01 of the best are considered. Modifying the `epsilon` value in `latent_trait_analysis()` allows multiple algorithms to overlap in the same problem difficulty.

Heatmaps can be generated for continuous data using the `heatmaps_crm`, showing positive sloped lines if an algorithm is not anomalous, thinner lines for more discriminating algorithms and blurrier lines for more consistent algorithms.

A user can analyse whether the fitted AIRT model is appropriate by employing `model_goodness_crm()` and `effectiveness_crm()`. When the output of `model_goodness_crm()` is passed to `autoplot`, the distribution of errors is plotted. The output of `effectiveness_crm()` and a plot type are used within `autoplot` to create three different plots. Type 1: Actual Effectiveness against Effectiveness Tolerance, Type 2: Predicted Effectiveness against Effective Tolerance, and Type 3: Predicted Effectiveness against Actual Effectiveness. Type 3 is important as the closer the points are to the dotted line $y = x$, the better the fitted AIRT model.

For polytomous data, the output of `pirtmodel()` is used with `tracelines_ poly()` and `autoplot` to create tracelines showing the probability of reaching a performance band. Performance bands are labelled 1 to 5, with the probability of scoring 5 being higher for easier datasets and lower for challenging datasets. Similar to how model goodness is visualised for continuous data, `model_goodness_poly()` and `effectiveness_poly()` display the same plots and use the same plot types.

2.2.2. AIRT Shiny App functions

Users can use a pre-generated example file or upload their dataset as a CSV document. When a user uploads a dataset to the server, a validation check is committed over the whole CSV to ensure `cirtmodel()` can use it. The validation primarily checks that all fields besides the column names are numeric.

As the AIRT Shiny App aims to assist in data analysis, additional tools exist to modify the dataset. Modifying the dataset occurs before the dataset is processed, with UI elements allowing the user to:

- 'Scale Data' which fits each dataset value to be a proportion between 0 and 1 by flagging `scale` = TRUE in `latent_trait_analysis()`.
- 'Invert Data' transforms the dataset using $\max x - x$ for each column to map low to high values. This functionality is needed when low values indicate better performance, such as when root mean square error is the performance metric.
- 'Scale By' determines whether the proportion of a value received from 'Scale Data' is calculated per Column (Algorithm) or over the whole dataset.

By default, a CSV's minimum and maximum performance values are validated to be between 0 and 1. If performance values are not scaled, e.g. watts, then users can untick 'Scale Data'. This property is set within `cirtmodel()` and `latent_trait_analysis()` as an optional parameters `min.item` and `max.item`.

Furthering the data analysis tools found in `airt` are plots and tables unique to the AIRT Shiny App, which expand upon existing data presented to the user:

- When a user selects an algorithm when viewing the AIRT Attributes table, the Difficulty Limit and Consistency data in `latent_trait_analysis` is used to create a box plot (Fig. 3). This box plot shows all the algorithms as points with the selected algorithm highlighted.
- Extending from the Strengths and Weaknesses bar chart, we can compute the proportion of the latent trait spectrum occupied by each algorithm (see Fig. 4). The table containing these proportions updates alongside the epsilon slider next to the bar chart (Fig. 6).

Users can download their generated plots and tables in PNG format inside a tar file, generated by `downloadHandler()` and create temp directories for that session.

3. Illustrative examples

We follow the workflows shown in Fig. 1 from the AIRT Shiny App and R Package perspectives to complete the task of determining the strengths and weaknesses of the algorithm portfolio.

3.1. R package workflow

Firstly, we would load the `airt` library and load in our data.

```
library("airt")
data("classification_cts")
df <- classification_cts
```

Our data, pre-supplied by the `airt` package, can be replaced by a user's data conforming to the expected format. The pre-supplied data is taken from the MATILDA data repository [12].

```
irt_params <- cirtmodel(df)
airt_params <- latent_trait_analysis(df,
  paras = irt_params$model$param,
  epsilon = 0)
```

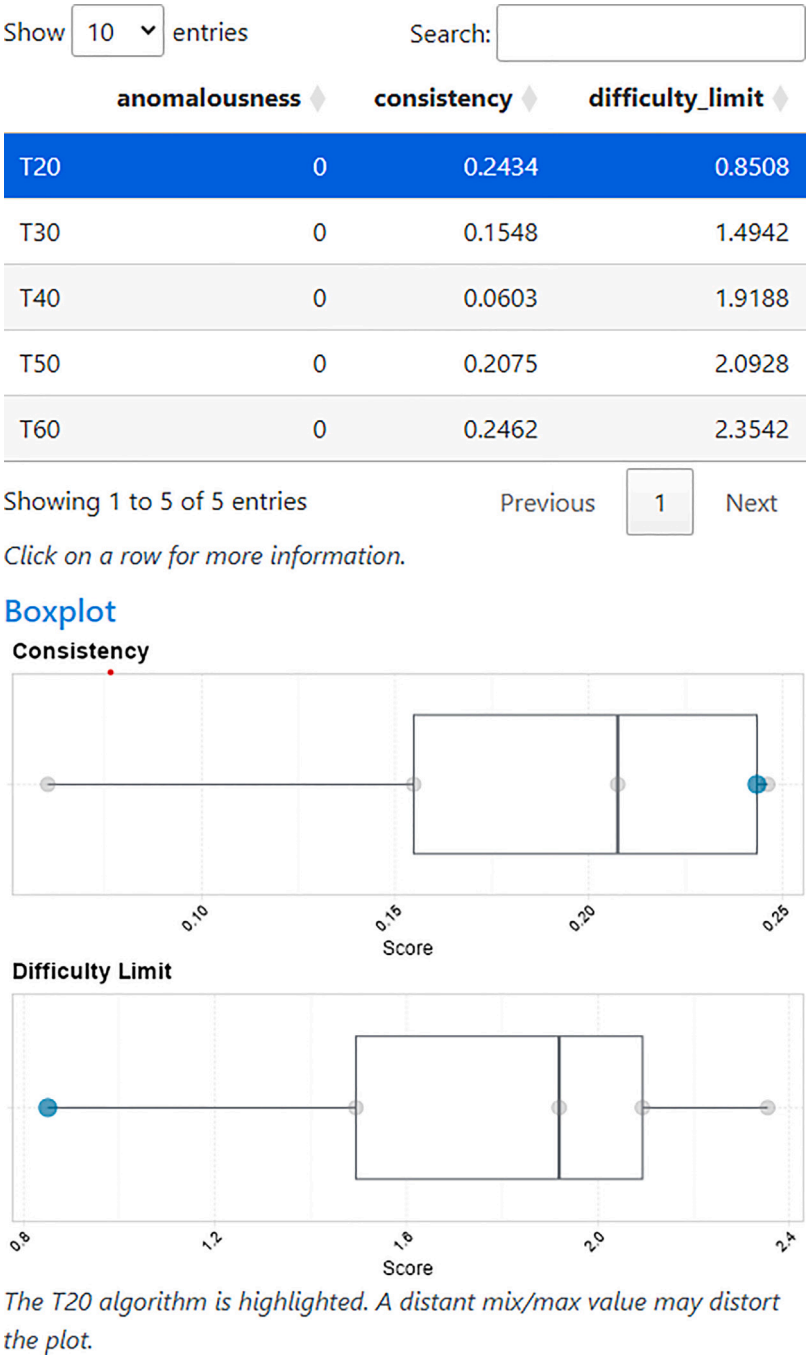


Fig. 3. A table showing the `airt` attributes of an algorithm portfolio. Boxplots show its consistency and difficulty when an algorithm is highlighted on the table.

The data frame is passed to `cirtmodel()` to fit the AIRT model. The IRT parameters stored in `param` are then passed into `latent_trait_analysis()` alongside the original data frame with an epsilon value. The plots are generated with a default `epsilon` value of 0, which shows the strongest and weakest algorithms for every value in the problem difficulty spectrum. Suppose `epsilon` is incremented by 0.1. In that case, algorithms within the strongest and weakest by 0.1 in performance are also displayed, with the range of displayed algorithms increasing with the epsilon value.

```
autoplot(airt_params, plottype = 3)
autoplot(airt_params, plottype = 4)
```

Four plots can be generated from `airt_params` using `autoplot` by setting the value of `plottype`. Options `{1,2}` plot algorithm performance with problem difficulty spectrum on the x axis and algorithm performance on the y axis. Option 3 displays smoothing splines fitted to the performance values as a function of problem difficulty. The splines corresponding to the strongest algorithm for a given problem difficulty

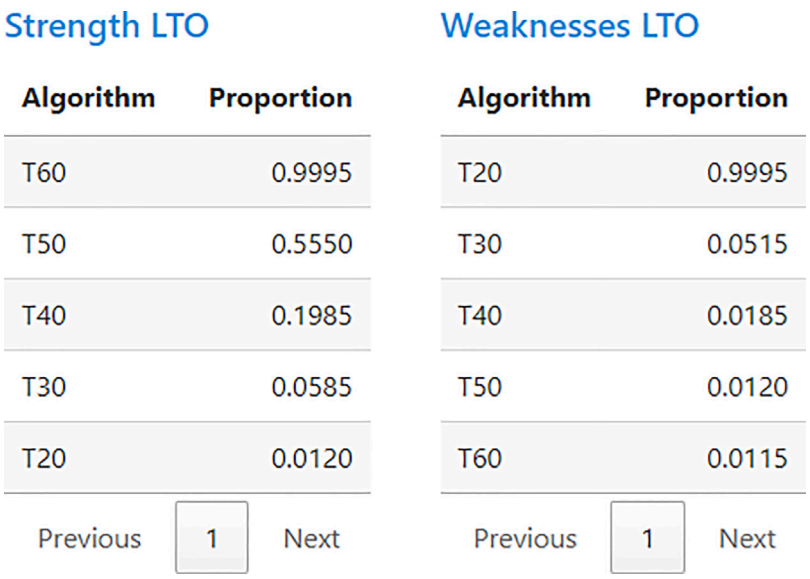


Fig. 4. A table generated under the Strengths and Weaknesses Bar Chart showing the proportion occupied by an algorithm on the problem difficulty spectrum for the selected epsilon value.

come at the top, while the weakest come at the bottom. Option 4 displays the strengths and weaknesses of algorithms across the problem spectrum. Fig. 4 gives an example output of strengths and weaknesses in the form of proportion of latent trait occupied.

3.2. AIRT Shiny App workflow

Here, we need a dataset to analyse, similar to the `airt` package. As shown in Fig. 5, AIRT Shiny App has controls for uploading or selecting datasets. In this workflow example, we will use the 'Classification' example dataset from the 'Select Example File' Dropdown box. Alternatively, the user could use the 'Browse' file upload input to upload their dataset as a CSV to the application. If the user chooses to upload their dataset, it goes through an additional validation check.

After selecting the dataset, the user would press the compute button and navigate to the Splines or Strengths and Weaknesses section to see the rendered plots as in Fig. 6.

Unlike the `airt` package, users can select an algorithm to highlight that algorithm's spline in the Splines section or remove the grey areas corresponding to the standard errors around each smoothing spline. In the Strengths and Weaknesses section, users can move a vertical slider on the left of the plot to change the epsilon value, which re-renders the plot.

3.2.1. AIRT Shiny App internals

Internally, after the user uploads and presses compute, the dataset is passed to the different `airt` functions and cached. In this case, the server loads `classification_cts` from their 'Classification' example file selection, following the same `airt` workflow listed above.

However, to allow for more fine-grained controls of the plot generation, AIRT Shiny App typically does not use `autoplot` when UI controls are added. Because of the number of possible plots with UI controls, only the default plots are cached; for example, a Strengths and Weaknesses plot with `epsilon` set to 0. For other plots, there is a general flow of fetching the cached `cirtmodel()` and `latent_trait_analysis()`, and then using:

```
//in server.R
renderPlot({
  generate_plot(plottype, epsilon_value)
})
```

to send the plot to the UI. This `generate_plot` function is a wrapper around `autoplot` to allow code reuse.

In situations with UI controls, such as the Splines section, custom plotting functions are used instead. In the case of the Splines plot, `generate_splines()` generates the standard Splines plot from cached `airt` functions, but when an algorithm is selected, `generate_spline_plot()` is used instead. This function takes the algorithm chosen from the UI and uses `gghighlight` to highlight that algorithm.

4. Impact

We have outlined the AIRT-Module, a tool that provides unique and accessible insight into evaluating the performance of an algorithm portfolio using Item Response Theory. This module assists users in making empirical-based decisions with easily digestible data visualisations, a streamlined workflow flow, and a choice between using an R Package or a Shiny App.

For a given task such as image classification, as the space of test problems expands, different algorithms are typically proposed to tackle different types of instances. Thus, discovering complementary algorithms is important as they can be part of an algorithm portfolio capable of tackling diverse instances. The AIRT-Module assists in showcasing algorithm diversity by computing IRT-based algorithm metrics and visualising their strengths and weaknesses. Furthermore, it aids reproducibility, an important aspect in AI research.

As of 21st of August 2024, the `airt` has over 23900 downloads on CRAN. Moreover, a tutorial on the AIRT Shiny App was conducted at The Genetic and Evolutionary Computation Conference 2024.

5. Conclusions

We have presented AIRT-Module, a two-component R ecosystem comprising an R package and a Shiny app for algorithm portfolio evaluation. AIRT-Module brings insights from IRT – a suite of methods from educational psychometrics – to algorithm evaluation. Our framework enables a detailed and comprehensive analysis of algorithm performance across diverse problem settings, contributing to a more nuanced understanding of their strengths and weaknesses. This tool enhances the capability to position algorithms within a state-of-the-art portfolio and identify their strengths and weaknesses, ultimately advancing AI research. Future work can explore expanding our framework to incorporate a higher dimensional latent trait and adapting it to handle new data types and evaluation metrics.

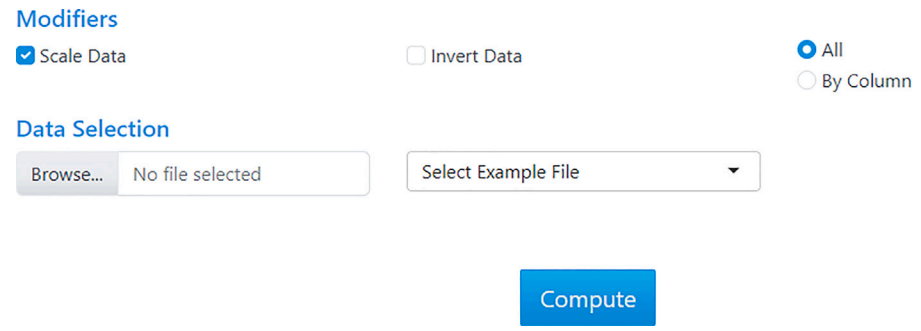


Fig. 5. Inputs shown at the start of AIRT Shiny App. Modifiers transform the dataset, Data Selection allows users to either upload a dataset or select an example dataset from MATILDA.

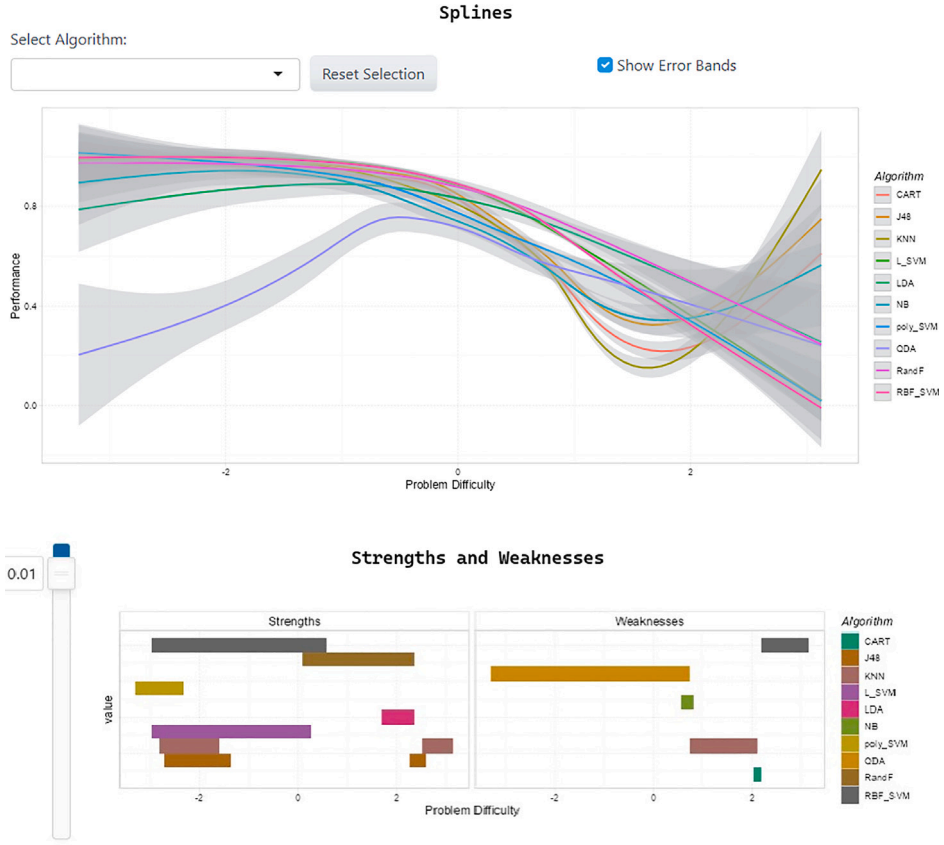


Fig. 6. Splines Plot (Above), Strengths and Weaknesses Plot (Below). Only the AIRT Shiny App has the UI controls.

CRediT authorship contribution statement

Brodie Oldfield: Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation. **Sevvandi Kandanaarachchi:** Writing – review & editing, Writing – original draft, Validation, Supervision, Software, Project administration, Methodology, Investigation, Conceptualization. **Ziqi Xu:** Writing – review & editing, Writing – original draft, Software. **Mario Andrés Muñoz:** Writing – review & editing, Supervision, Project administration, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Embretson SE, Reise SP. Item Response Theory. Psychology Press; 2013.

[2] Hambleton RK, Swaminathan H. Item Response Theory: Principles and Applications. Boston: Kluwer Nijhoff Publishing, Springer Science; 1985.

[3] Struk AA, Carriere JS, Cheyne JA, Danckert J. A short boredom proneness scale: Development and psychometric properties. Assessment 2017;24(3):346–59. http://dx.doi.org/10.1177/1073191115609996.

[4] Maertens R, Götz FM, Golino HF, Roozenbeek J, Schneider CR, Kyrychenko Y, et al. The Misinformation Susceptibility Test (MIST): A psychometrically validated measure of news veracity discernment. Behav Res Methods 2024;56(3):1863–99. http://dx.doi.org/10.3758/s13428-023-02124-2.

[5] Martínez-Plumed F, Prudêncio RBC, Usó AM, Hernández-Orallo J. Item response theory in AI: analysing machine learning classifiers at the instance level. Artificial Intelligence 2019;271:18–42. http://dx.doi.org/10.1016/j.artint.2018.09.004.

[6] Xu Z, Ma C, Ren Y, Chan J, Shao W, Xia F. Towards Better Evaluation of Recommendation Algorithms with Bi-directional Item Response Theory. In: Companion proceedings of the ACM on web conference 2025. 2025, p. 1455–9. http://dx.doi.org/10.1145/3701716.3715540.

[7] Xu Z, Kandanaarachchi S, Ong CS, Ntoutsi E. Fairness Evaluation with Item Response Theory. In: Proceedings of the ACM on web conference 2025. 2025, p. 2276–88. http://dx.doi.org/10.1145/3696410.3714883.

[8] Kandanaarachchi S, Smith-Miles K. Comprehensive Algorithm Portfolio Evaluation using Item Response Theory. J Mach Learn Res 2023;24:177:1–52, URL https://jmlr.org/papers/v24/20-1318.html.

[9] Chalmers RP. mirt: A Multidimensional Item Response Theory package for the R Environment. J Stat Softw 2012;48(6):1–29. http://dx.doi.org/10.18637/jss.v048.i06.

[10] Zopluoglu C. EstCRM: Calibrating parameters for the Samejima's Continuous IRT model. 2022, R package version 1.6.

[11] Wickham H. ggplot2: Elegant graphics for data analysis. Springer-Verlag New York; 2016.

[12] Smith-Miles K, Muñoz MA, Neelofar N. Melbourne Algorithm Test Instance Library with Data Analytics (MATILDA). 2020, URL https://matilda.unimelb.edu.au/matilda/. [Accessed 5 August 2024].