OptimalCAT 2.4 Manual

OptimalCAT is an easy-to-use ***production-grade*** Computer Adaptive Testing (CAT) engine. It's both for psychometricians to run simulations and for IT engineers to deploy to production.

It builds upon the shadow-test approach. It leverages state-of-the-art Mathematical Programming solvers (FICO Xpress or Gurobi) to deliver ***optimal*** tests to each student. It guarantees that test blueprints and constraints are always met, and measurement accuracy is maximized.

The engine is built as a microservice (REST API interface) and can be easily integrated with test delivery systems.

OptimalCAT Core Team:

Jie Li, PhD
Wim J. van der Linden, PhD
Richard J. Patz, PhD

Support: OptimalCAT@outlook.com

Contents

OptimalCAT License

	Use	Limitation and Support
OptimalCAT Free Version available on www.Optimal-CAT.com	Personal use and research	Users need to install solver and obtain solver license* (FICO Xpress or Gurobi) before using OptimalCAT.
OptimalCAT Commercial Version shared with commercial users through private links	Commercial use	The solver and commercial solver license are embedded as OEM components, optimized for deployment. Supports both Windows and Linux platforms. Please contact us for pricing.

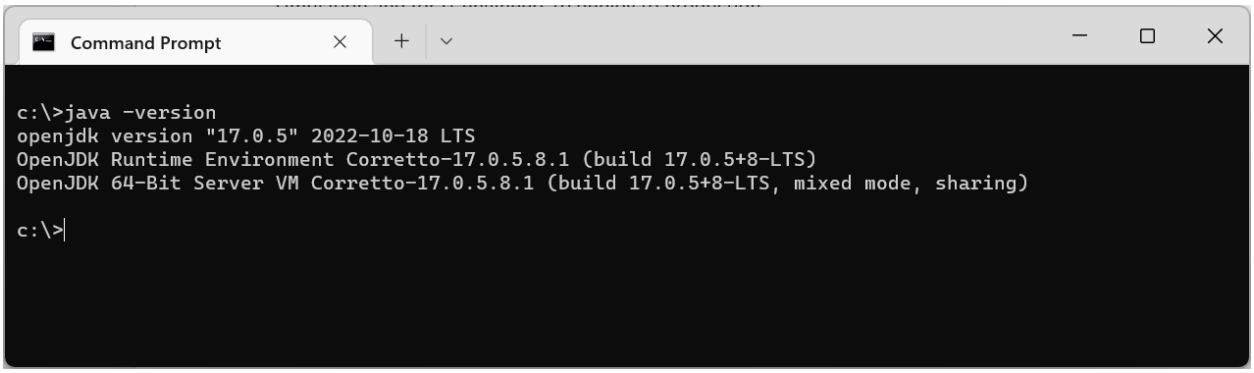
*Solver and free-but-limited license:

FICO Xpress Community License
https://www.fico.com/en/fico-xpress-community-license

Gurobi Online Courses License
https://www.gurobi.com/product/download-center/online-course-license-request

Prerequisites

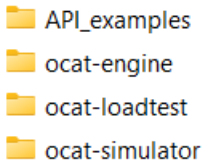
To use the OptimalCAT engine, you need to have **Java 17 or above** installed on your computer. If you are not sure which Java to use, you could try Amazon Corretto https://aws.amazon.com/corretto. Once you have installed Java, make sure you check the version.



The OptimalCAT engine also uses **Microsoft Excel** (or Apache OpenOffice) as the editing tool for CAT test configurations.

Getting Started Guide (Windows version)

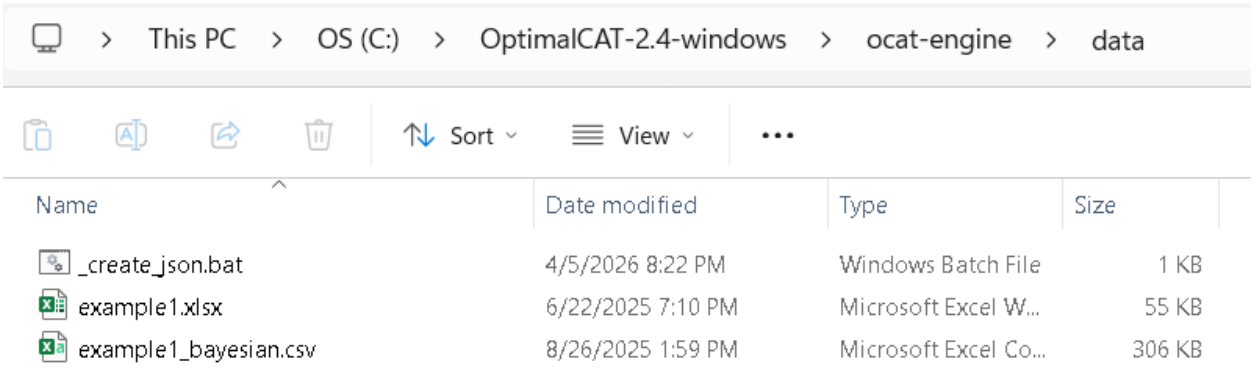
Download and unzip OptimalCAT from www.Optimal-CAT.com. In the unzipped folder, you should see the following sub-folders.



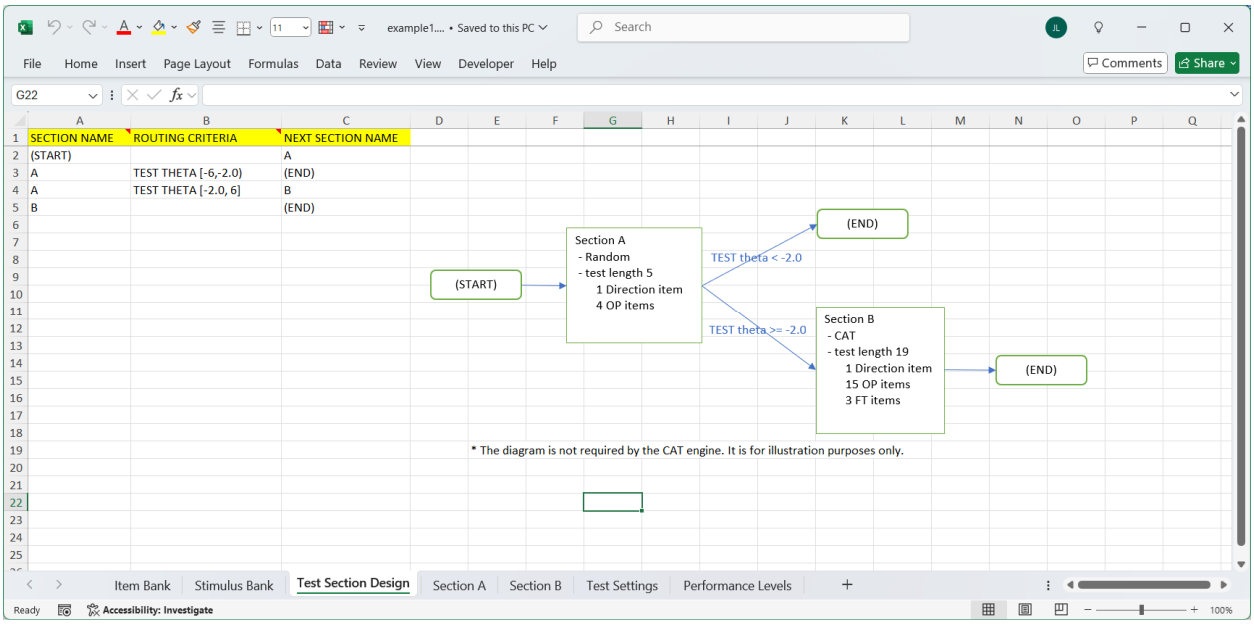
- API_examples: API examples in Java, Python and R [1]
- ocat-engine: OptimalCAT engine
- ocat-loadtest: OptimalCAT load test tool for IT engineers
- ocat-simulator: OptimalCAT simulator for psychometricians

1. Test configuration

First, go to the "ocat-engine" folder, and then the "data" folder. There is one test configuration Excel file (example1.xlsx) in the folder. You could have multiple test configuration Excel files, each for one test.



[1] No programming is needed for psychometricians to set up CAT tests and run simulations. These API examples are provided to assist IT engineers to integrate the OptimalCAT engine into the testing delivery systems. You may also use these examples to build your own simulation client.



A quick overview of the sheets in test configuration files:

- **Item Bank** sheet contains item info. Yellow highlighted columns are required by the system. You could add any other columns you need, and you could use them later in Constraint definition.
    - The engine supports the following IRT models: 1PL, 2PL, 3PL, GPCM and GRM.

- **Stimulus Bank** sheet contains stimulus info.

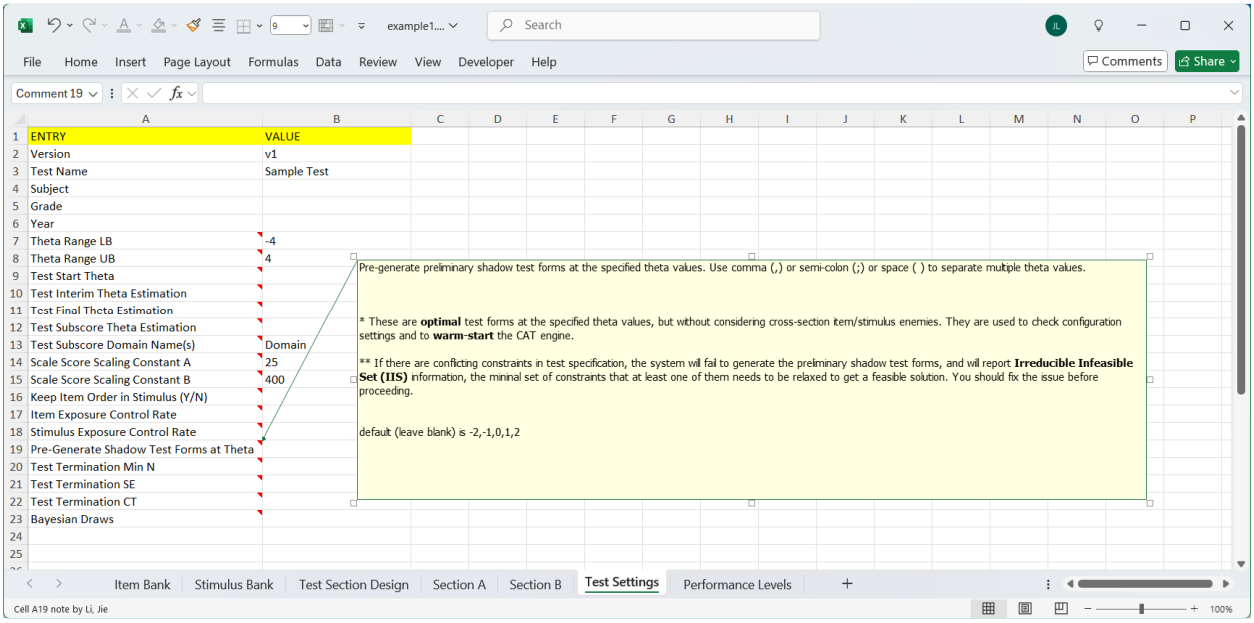
- **Test Section Design** sheet contains test section routing info. You can have many test sections and define routing criteria from one section to another. This allows building many test configurations like Multi-Stage Testing (MST). There are two default dummy sections - (START) and (END) - to designate the beginning and the end of a test.

- **Section A**, **Section B** … (or simply **A**, **B**…) for each section defined in Test Section Design sheet, there should be a corresponding sheet for the section, in which you define things like test format for the section. The engine supports the following test formats:
    - CAT - item/unit level CAT using Maximum Fisher Information
    - CAT_BM - item/unit level CAT using b-param matching
    - Random - linear test, item order is random
    - Linear - linear test, item order specified by user
    - LOFT - linear on-the-fly test, this is a special case of CAT with no test form refresh

- **Test Settings** sheet contains test level parameters, e.g., theta range, scale score constants, etc.

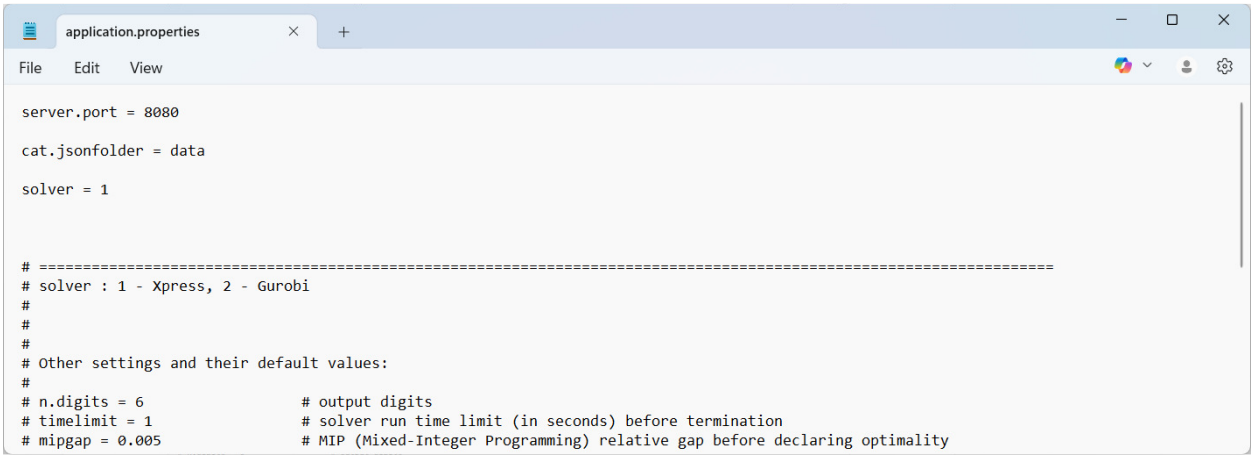
- **Performance Levels** sheet contains cut scores and performance levels definition.

The **cell comments** (a pop-up message when moving over the cell with a red triangle) provide useful tips about the fields.

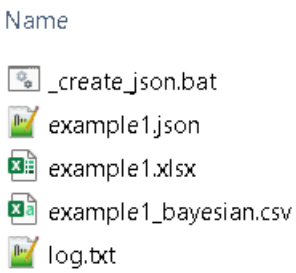


Once you have test specifications defined in Excel configuration files, the next step is to run ***_create_json.bat***.

*** OptimalCAT Free Version Users**: Before you run ***_create_json.bat***, you need to go to "ocat-engine" folder, make sure the "solver" setting in the ***application.properties*** file (1 – Xpress, 2 – Gurobi) matches the solver installed on your computer.



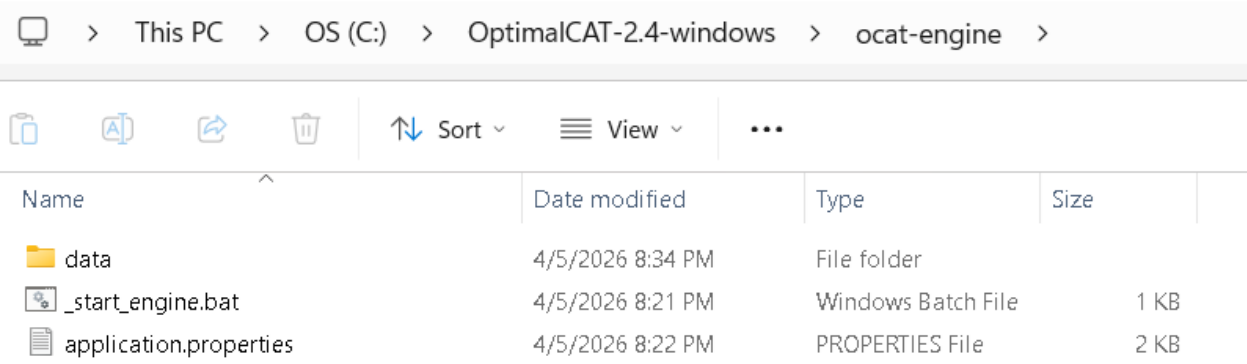
After running ***_create_json.bat***, you should see a log file (log.txt) and a JSON file (example1.json) for each test. Check the screen output and the log file (log.txt) carefully to make sure no error is reported. Otherwise, you should correct errors and then re-run.



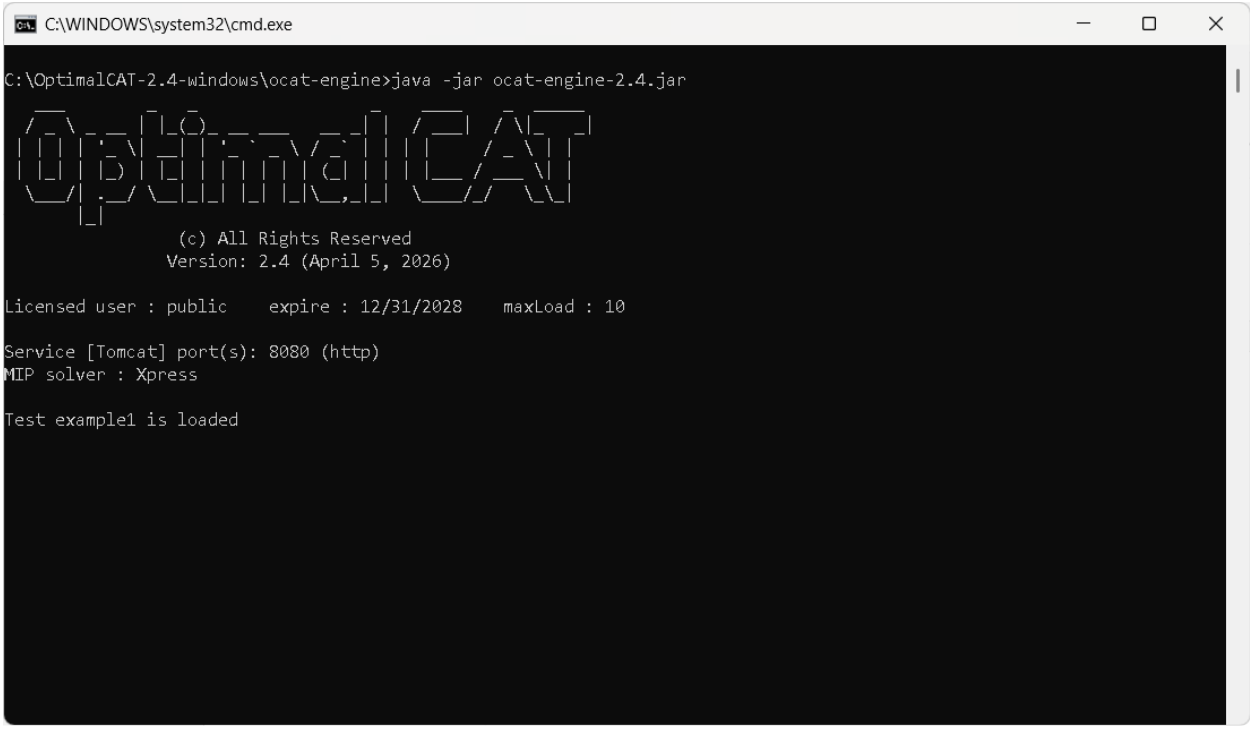
Notice: the CAT engine reads only JSON files, not test configuration Excel files.

2. Start CAT engine

To start the CAT engine, go to “ocat-engine” folder and run ***_start_engine.bat***.



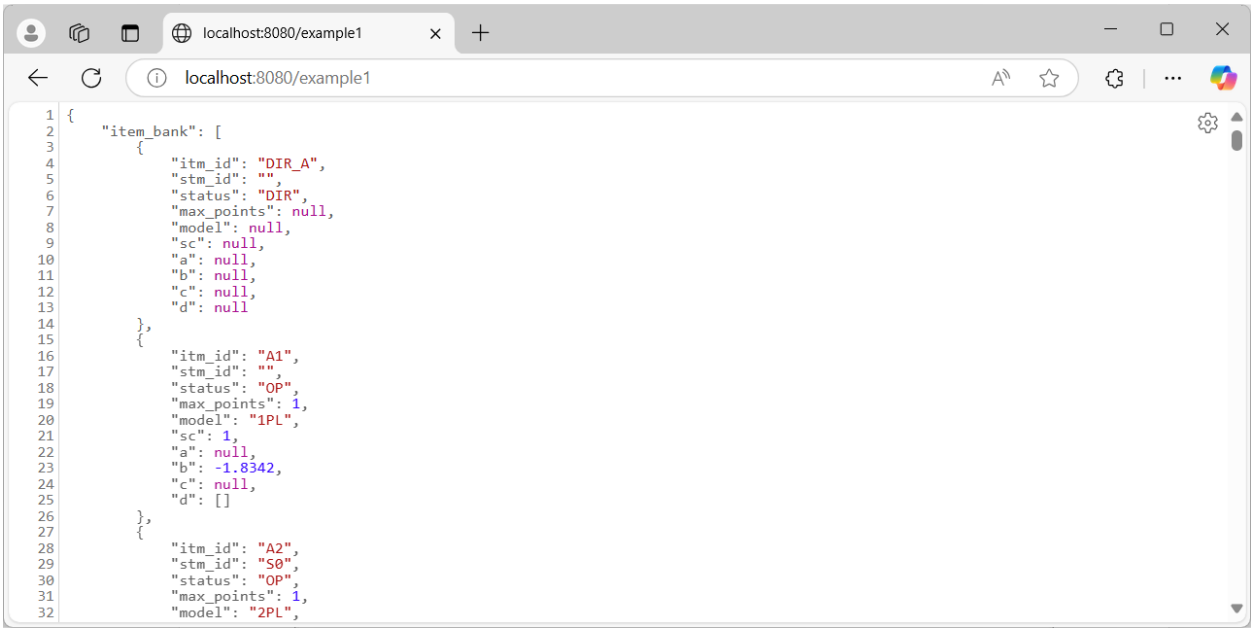
You should then see a command window like below. It shows that the engine is started and listening on port 8080, MIP solver is Xpress or Gurobi (depending on the settings in ***application.properties*** file) and test example1 is loaded. Leave the command window there. It’s now ready to process incoming API calls.



The CAT engine has the following public REST API endpoints.

- / - HTTP GET request. It returns basic info of the engine and the loaded tests.
- /{testID} - HTTP GET request. It returns the test JSON file.
- /IEC/{testID} - HTTP GET request. It returns item/stimulus exposure counts.
- /cat - HTTP POST request. It returns the next items to administer and final scores.
- /rescore - HTTP POST request. It is for rescoring.

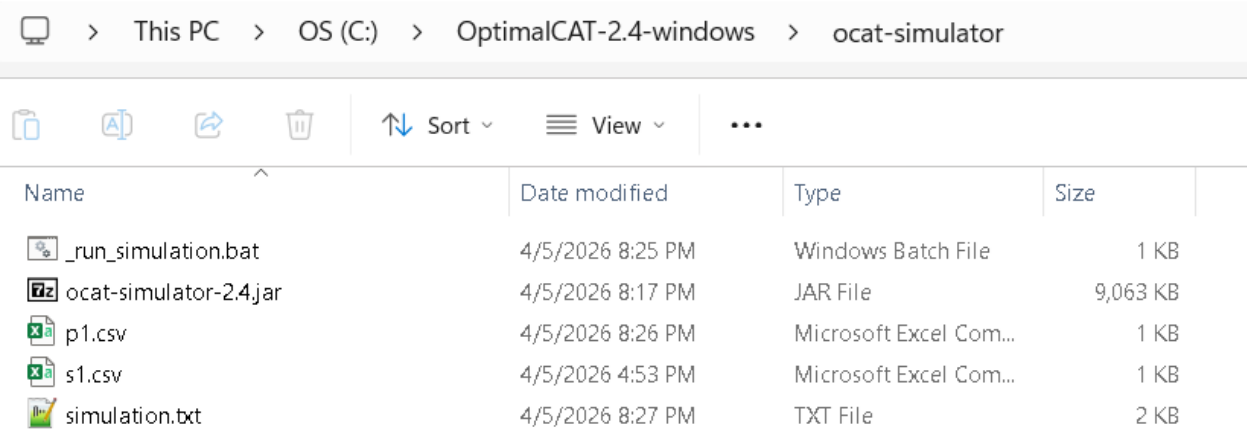
For GET request endpoints, you could use a browser to access them, for example,



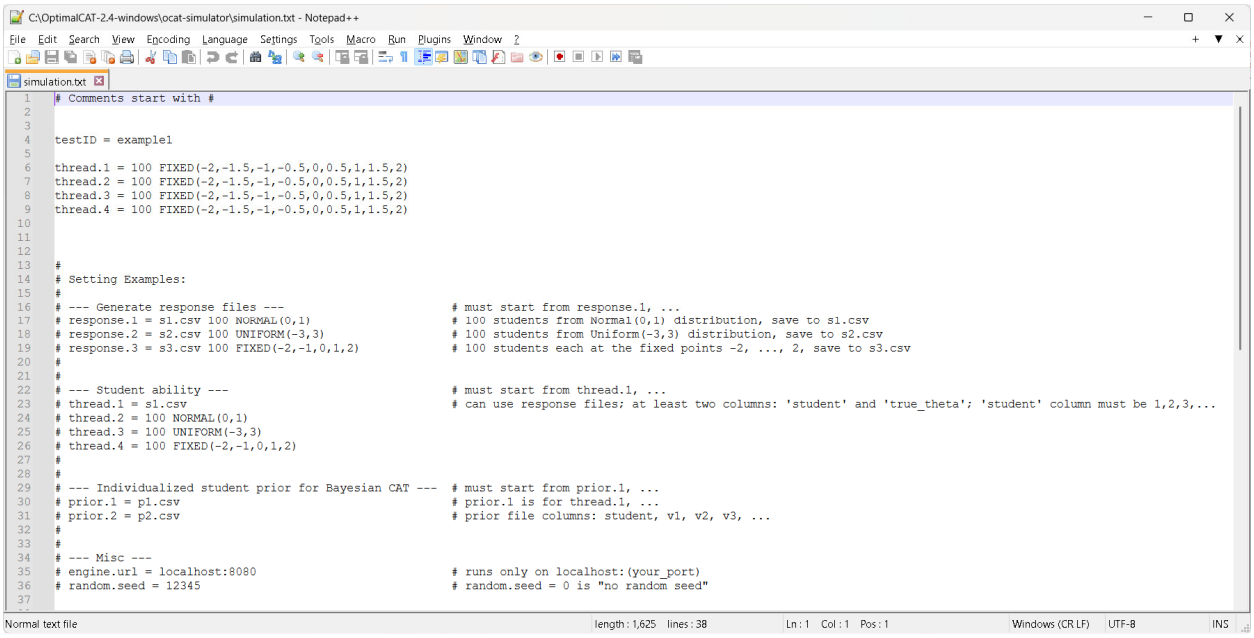
For POST request endpoints (/cat and /rescore), they are for IT engineers to integrate with test delivery systems for test administration and scoring. See “Appendix I. OptimalCAT engine REST API” for details.

3. Run simulation (for Psychometricians)

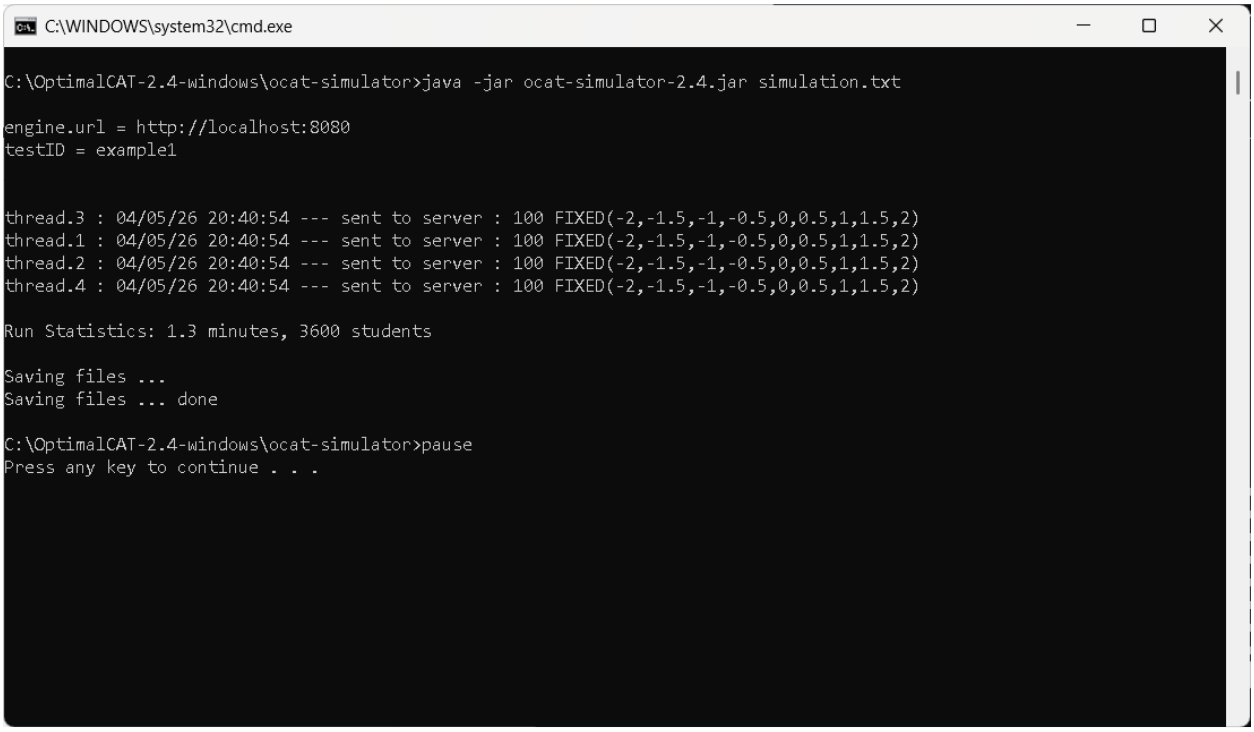
Now the CAT engine is up-running, you could start simulation. Go to “ocat-simulator” folder



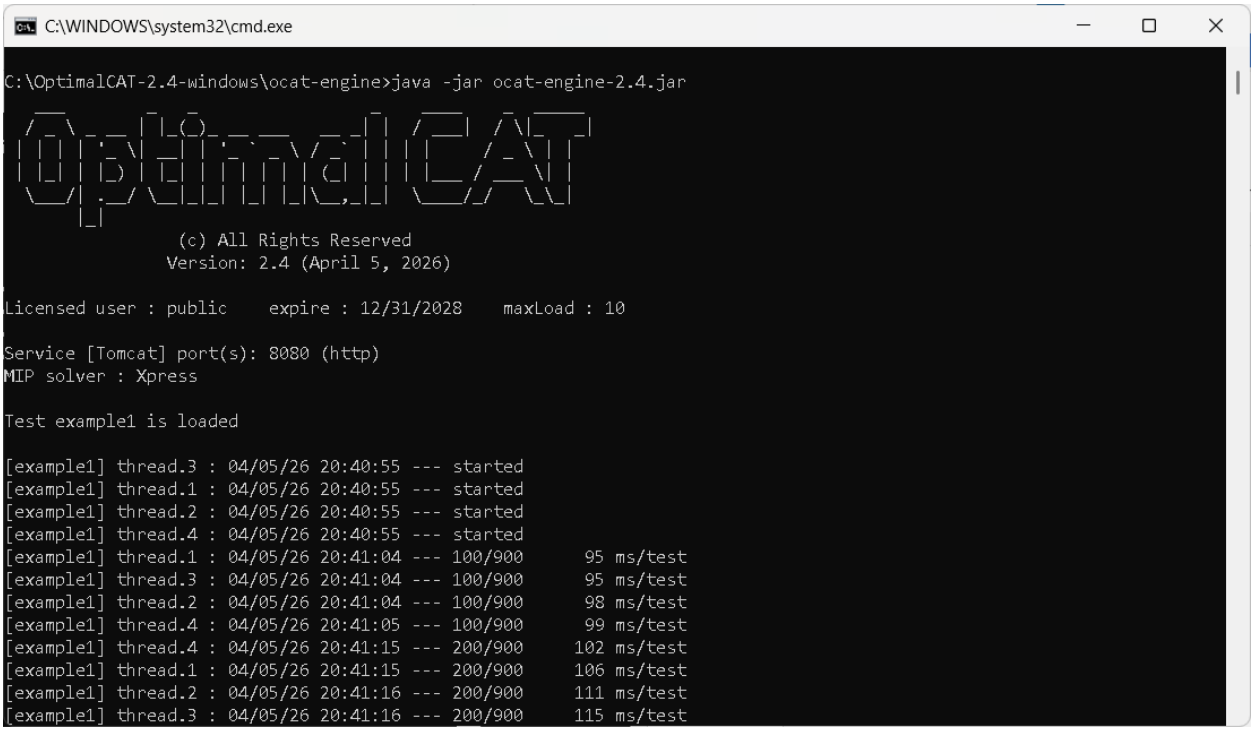
The ***simulation.txt*** file defines the parameters to run the simulation.



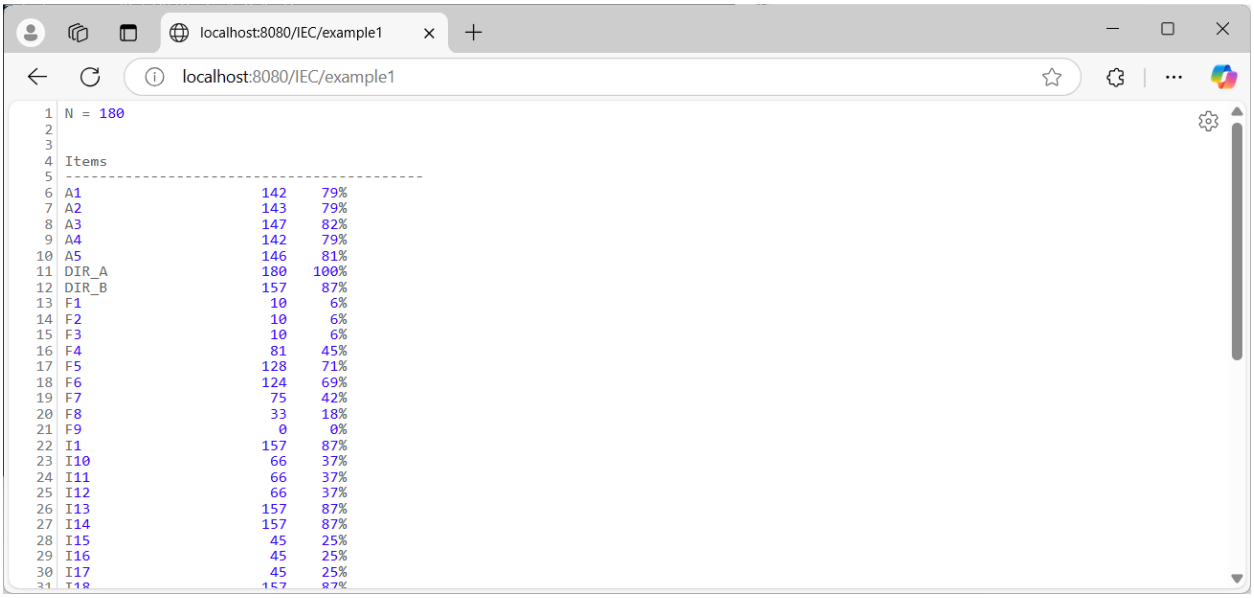
Run ***_run_simulation.bat*** to start the simulation. Once it's completed, you should see the output below.



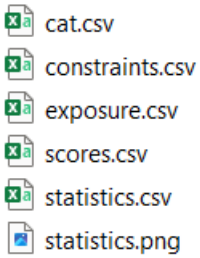
The simulation is run on the server side (to save the API traffic between client and server), you could see its progress in the Server window.



While the simulation is running, you could also check the item/stimulus exposure status through the /IEC/{testID} endpoint.



After the simulation run is completed, you should see results in a newly created folder named as {testid}_{date}_{time}. There are six files in the folder:

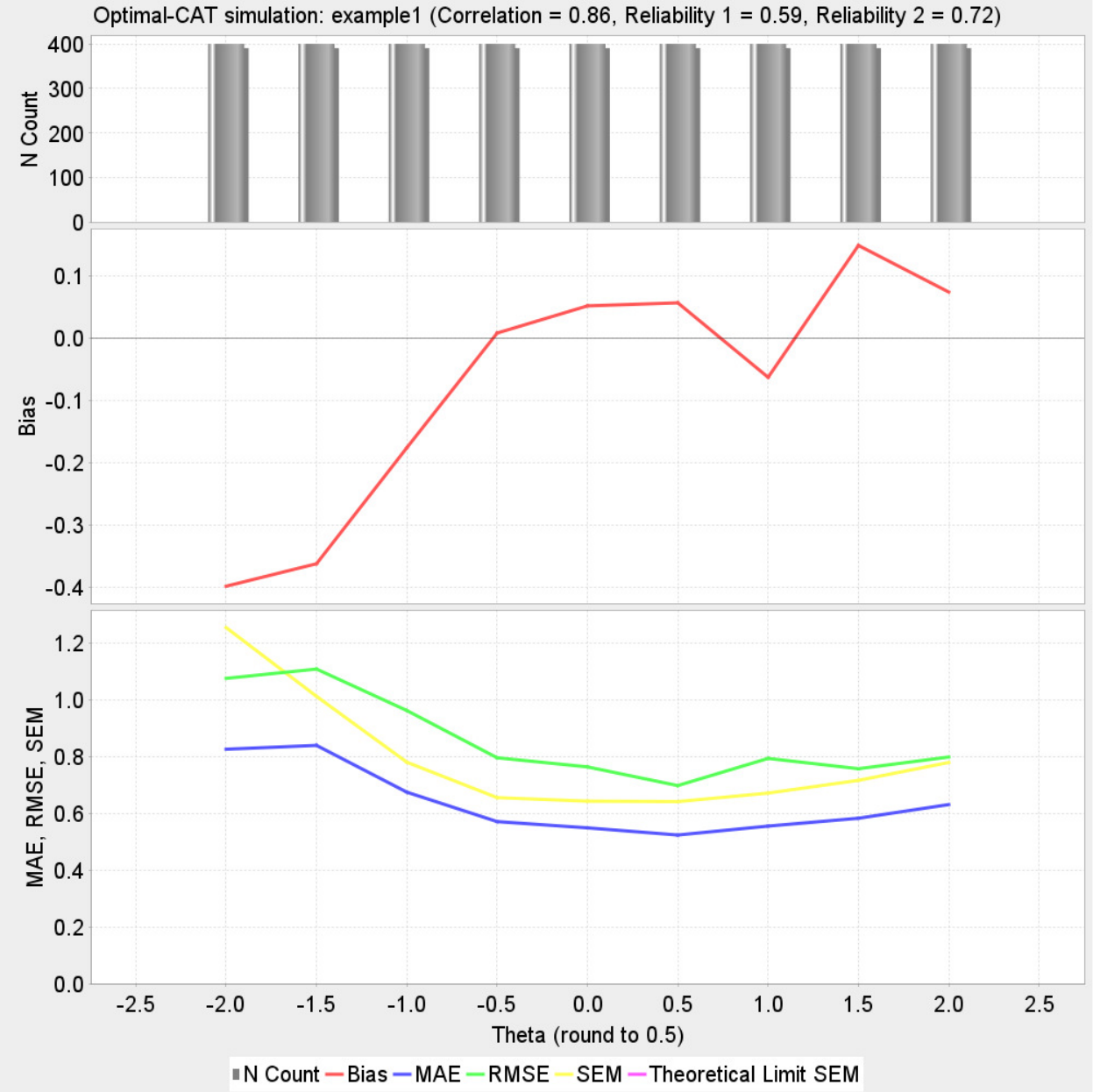


- cat.csv – this file contains detailed test session information, item by item.
- constraints.csv – this file contains constraints details.
- exposure.csv – this file contains exposure counts for both items and stimuli.
- scores.csv – this file contains final scores and sub-scores.
- statistics.csv – this file contains the following measurement statistics
    - N Count
    - Bias
    - MAE - Mean Absolute Error
    - RMSE - Root Mean Square Error
    - SEM (or Bayesian_SD) - Standard Error of Measurement (or Standard Deviation in Bayesian CAT)
    - Theoretical Limit SEM – it is calculated by aggregating all the pre-generated shadow test forms. **It is shown only when the CAT engine detects that all students see all the test**

sections defined in the configuration Excel file; otherwise, the Theoretical Limit SEM is not shown (you must manually aggregate only the test sections that the student routes through to find out).

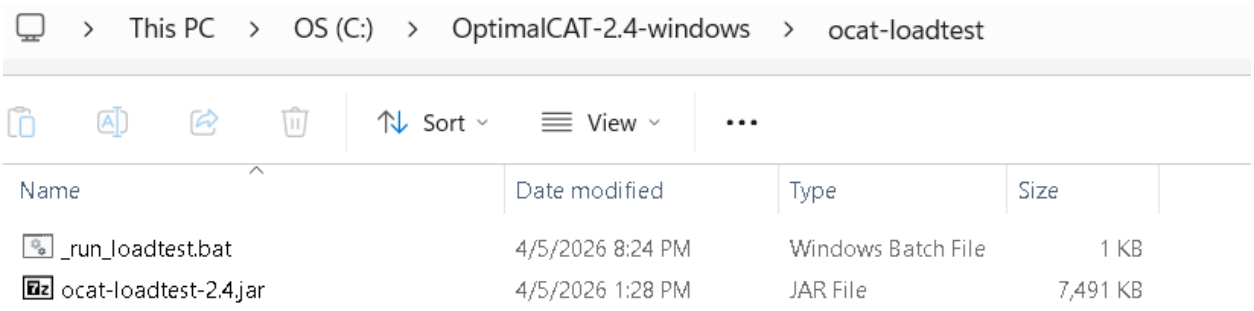
- Correlation of true theta (θ_j) and estimated theta ($\hat{\theta}_j$)
- Reliability 1 - defined as $\rho^2 = \frac{Var(\theta_j)}{Var(\hat{\theta}_j)}$
- Reliability 2 - defined as $\widehat{\rho^2} = 1 - \frac{E(CSEM_{\hat{\theta}_j}{}^2)}{Var(\hat{\theta}_j)}$, where $CSEM_{\hat{\theta}_j}$ is the Conditional Standard Error of Measurement (CSEM) at the estimated ability $\hat{\theta}_j$

- Statistics.png – chart using the numbers in statistics.csv

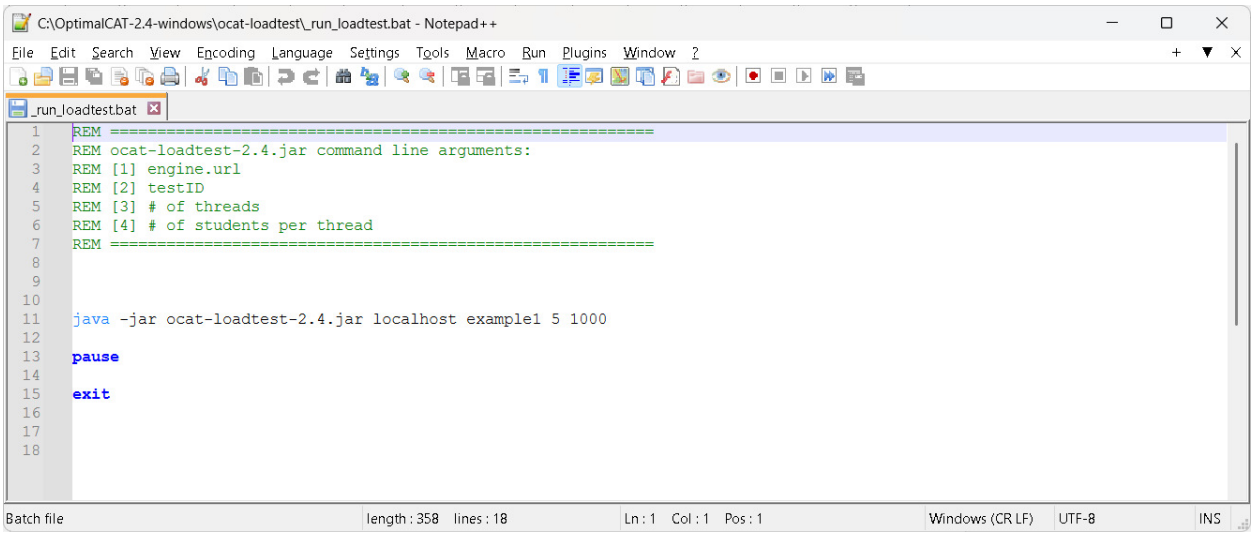

Optimal-CAT simulation: example1 (Correlation = 0.86, Reliability 1 = 0.59, Reliability 2 = 0.72)
N Count
400
300
200
100
0
Bias
0.1
0.0
-0.1
-0.2
-0.3
-0.4
MAE, RMSE, SEM
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-2.5
-2.0
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
2.5
Theta (round to 0.5)
N Count
Bias
MAE
RMSE
SEM
Theoretical Limit SEM

4. Run load test (for IT engineers)

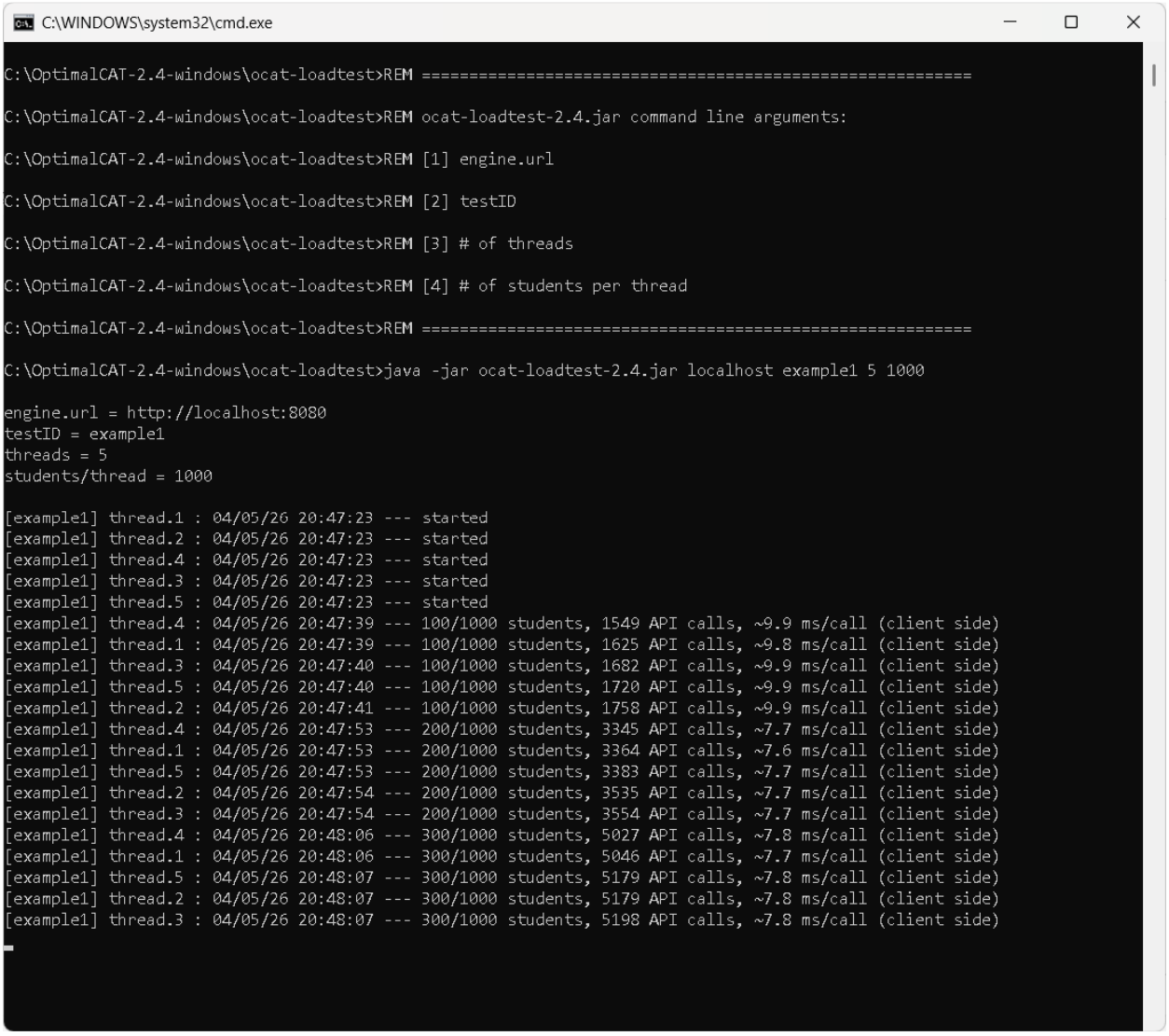
To get an idea of the actual response time of engine API calls in real operations, a load test tool is provided in “ocat-loadtest” folder.



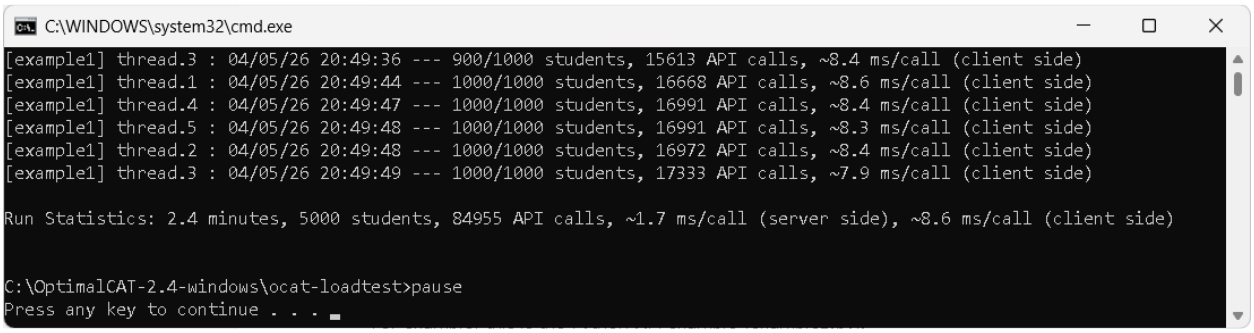
You could change the command line arguments to fit your load test needs.



Run ***_run_loadtest.bat*** to start a load test. You should see a window like below.



At the end of the load test run, it reports the engine response time (average milliseconds per API call) from both server side and client side. Based on our experience, most of the real CAT tests have response time below 100 milliseconds. The larger the item bank and the more complicated the test specifications, the longer the response time.



Notice: The load test results heavily depend on the hardware and OS software. We recommend using Linux servers for production.

A typical load test scenario: set up OptimalCAT engine on a Linux server, and then set up 10 or 20 computers as clients each running 10 or 20 threads to stress test the CAT engine and see how many servers would need for operational tests.

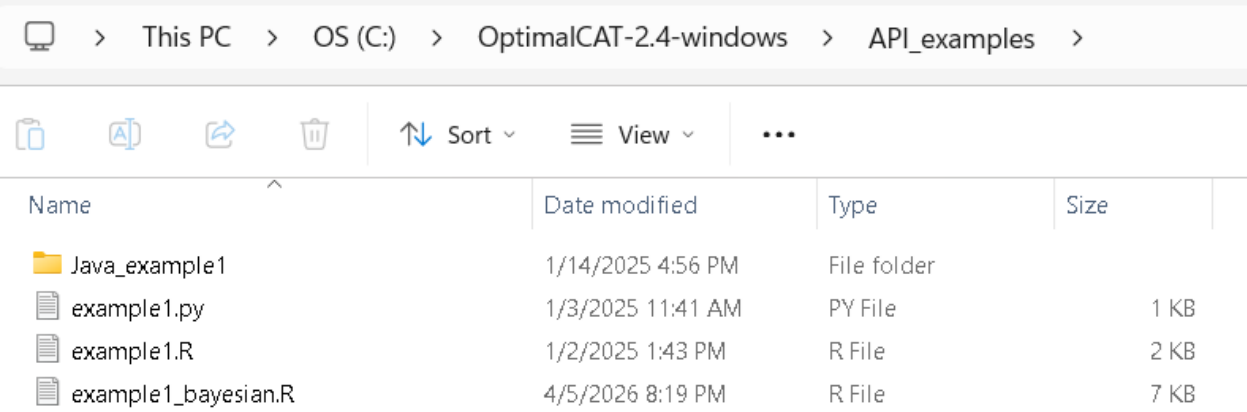
For most assessment programs, we believe one or two powerful servers should be enough. For very large testing programs with high concurrency requirements, we suggest creating a server pool and putting a load balancer in front, e.g., using Load Balancer - Amazon Elastic Load Balancer (ELB) - AWS.

Appendix I. OptimalCAT engine REST API

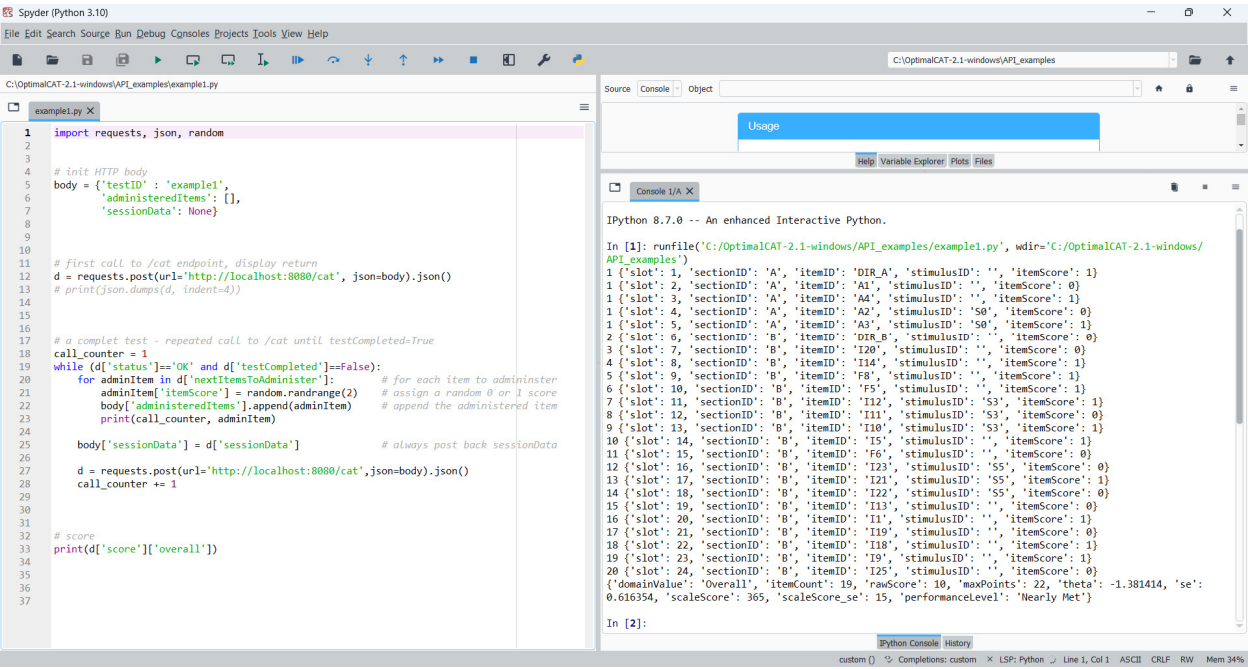
The CAT engine has the following public REST API endpoints:

- / - HTTP GET request. It returns basic info of the engine and the loaded tests.
- /{testID} - HTTP GET request. It returns the test JSON file.
- /IEC/{testID} - HTTP GET request. It returns item/stimulus exposure counts.
- /cat - HTTP POST request. It returns the next items to administer and final scores.
- /rescore - HTTP POST request. It is for rescoring.

There are examples (written in Java, Python and R) in “API_examples” folder for your reference.



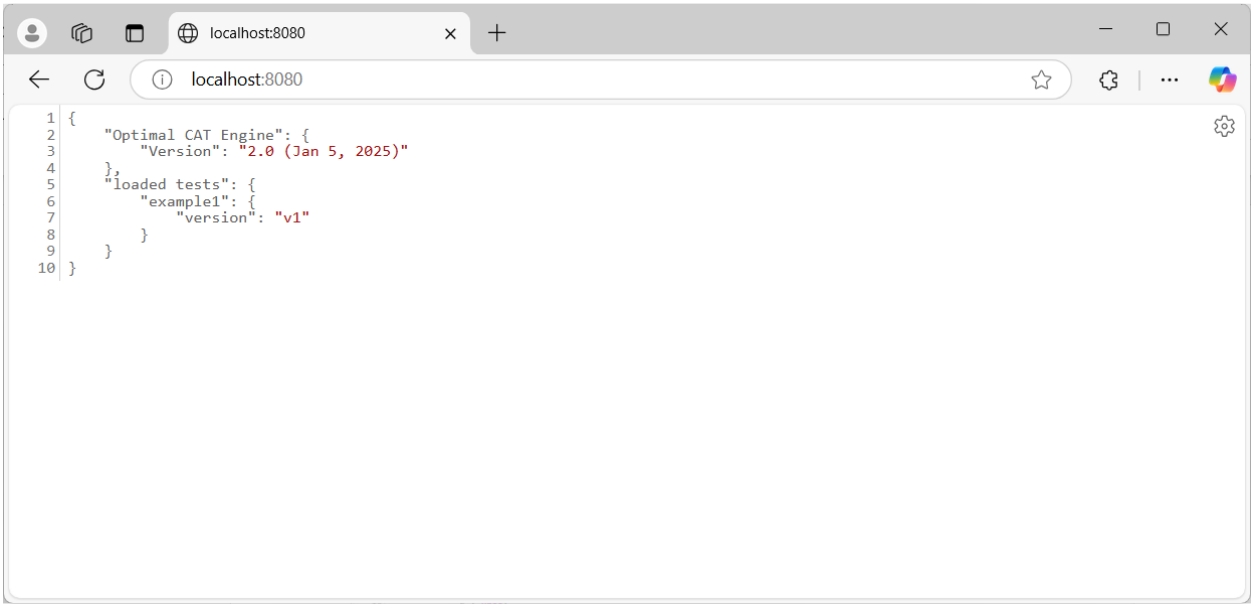
For example, this is the Python API example (example1.py).



GET /

Parameters: (none)

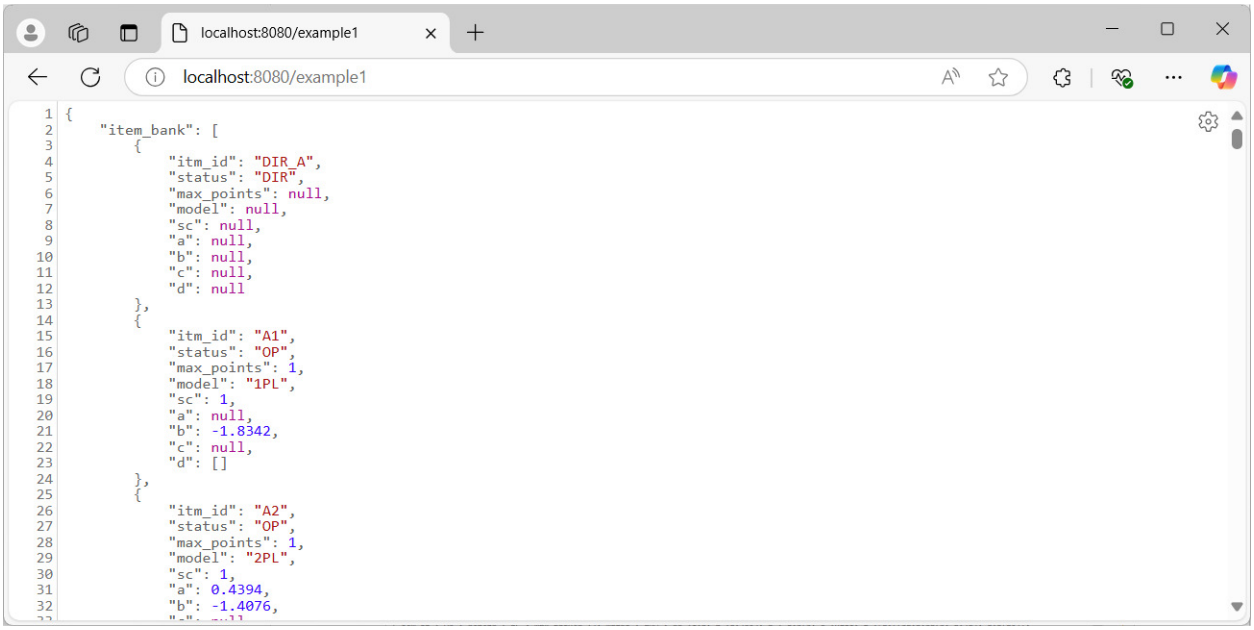
Returns: basic info of OptimalCAT engine and loaded test



GET /{testID}

Parameters: testID String

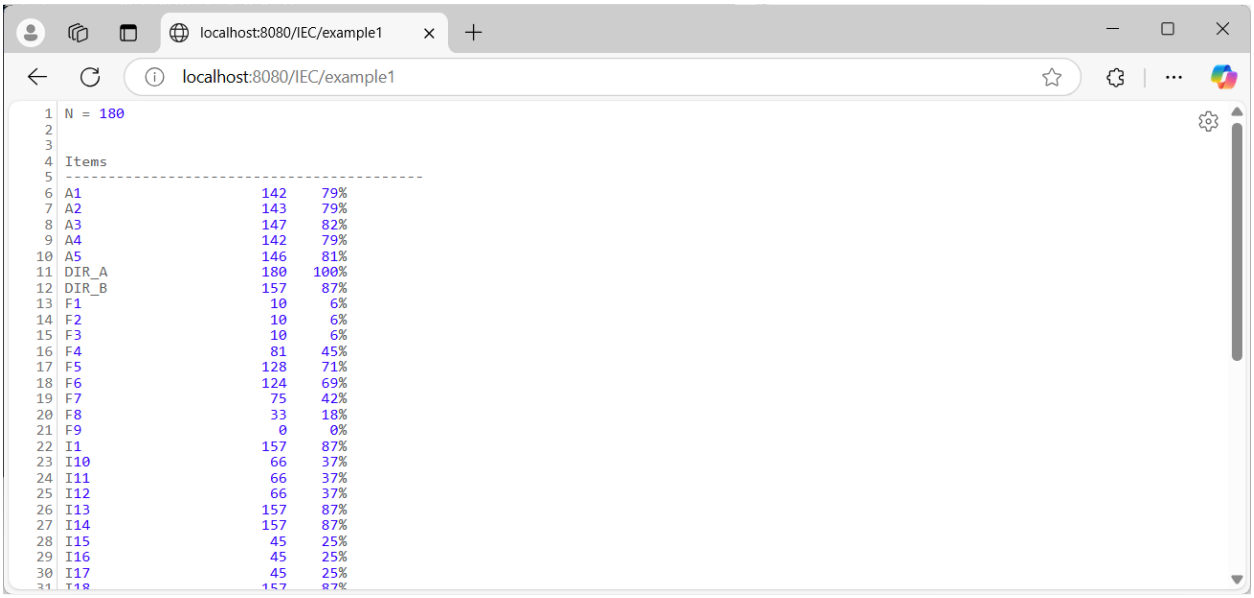
Returns: test JSON file, including item bank, test sections, test settings, etc.



GET IEC/{testID}

Parameters: testID String

Returns: item and stimulus exposure counts



POST /CAT - for test administration (what is the next item?)

Header: Content-Type = application/json

Parameters: (none)

Body: JSON format of the following fields:

- **testID** String valid testID

- **administeredItems** list of administered item information, as follows
  - slot Integer the slot in the test
  - sectionID String the section in the test
  - itemID String the item ID
  - stimulusID String the item's stimulus ID
  - itemScore Integer item score. If the item cannot be scored in real time, use **-1**, and the item will be ignored in theta calculation. If the item is a FT or a non-scorable item, itemScore is ignored by the engine.

- **sessionData***

The is the sesstionData returned by /cat call. The CAT engine requires the client to post back sessinData. See below Returns for details of the sessionData.

- testStartTheta (optional) for overriding the default test start theta
  - distribution: String NORMAL, UNIFORM, FIXED
  - param1 Double
  - param2 Double

  for NORMAL distribution, param1 is mean and param2 is standard deviation
  for UNIFORM distribution, param1 is min value and param2 is max value
  for FIXED distribution, param1 is the fixed value, param2 is ignored

- avoidItems (optional) list of String avoid item IDs. If the item ID is not in the item pool, it will be ignored.

Returns: JSON format of the following fields:

- **testID** String testID
- **status** String the status of the request. "**OK**" if successful, otherwise see the detailed error message
- **testCompleted** Boolean whether the test has completed
- **nextItemsToAdminster** a list of the next items to administer
  - slot Integer the slot in the test
  - sectionID String the section in the test
  - itemID String the item ID
  - stimulusID String the item's stimulus ID
- **score** same structure as "score" in /RESCORE (see below) score is only filled when testCompleted = True
- **sessionData*** the session data that the CAT engine requires the client to always post back
  - shadowTest the current shadow test. A list of the following fields
    - slot Integer the slot in the test
    - sectionID String the section in the test
    - itemID String the item ID
    - stimulusID String the item's stimulus ID
  - lastSlotSeen int the last slot seen
  - test_theta Double interim test theta
  - test_se Double interim test theta S.E.
  - session_theta Double interim section theta
  - session_se Double interim session theta S.E.

- bayesian_theta
  - mean Double mean of theta draws
  - sd Double S.D. of theta draws
  - vector Array of Double theta draws

*__sessionData__: for Bayesian CAT with individual prior for the student, you could initialize a sessionData with a vector that contains the individual prior in the first call to /CAT (see example1_bayesian.R)

POST /RESCORE - for test rescoring

Header: Content-Type = application/json

Parameters: (none)

Body: JSON format

the same structure as in /CAT request body (see above). sessionData should be set to **null** usually, unless for Bayesian CAT with individual prior (see example1_bayesian.R)

Returns: JSON format of the following fields:

- **testID** String testID
- **status** String the status of the request. "**OK**" if successful, otherwise see the detailed error message
- **score**
  - overall overall score details with the following fields
    - domainValue String "Overall"
    - itemcount Integer number of items
    - rawScore Integer raw score
    - maxPoints Integer max possible points
    - theta Double overall theta
    - se Double overall theta S.E.
    - scaleScore Integer overall scale score
    - scaleScore_se Integer overall scale score S.E.
    - performanceLevel String performance level
  - subscore list of subcores with the following fields
    - domainName String the subscore domain name
    - domainScore same structure as overall score
  - bayesian_theta
    - mean Double mean of theta draws
    - sd Double S.D. of theta draws
    - vector Array of Double theta draws
  - scoredItems list of String the list of item IDs used in scoring
  - skippedItems list of String the list of item IDs skipped in scoring

Appendix II. OptimalCAT engine IRT models and parameterization

The OptimalCAT engine supports the following IRT models: 1PL, 2PL, 3PL, GPCM and GRM.

The IRT model parameters are defined in the "Item bank" tab sheet in test configuration Excel file.

ITM_ID	STM_ID	ENEMIES	STATUS	MAX POINTS	MODEL	SCALING CONSTANT	A	B	C	D1	D2	D3	D4
DIR_A			DIR										
A1			OP	1	1PL	1		-1.8342					
A2			OP	1	2PL	1	0.4394	-1.4076					
A3			OP	1	3PL	1	0.9085	-1.4063	0.2452				
A4			OP	4	GPCM	1	1.25	-0.8342		0.6032	0.239	-0.2841	-0.5581
A5			OP	4	GRM	1	0.4394	-0.5750		0.7756	0.3673	-0.464	-0.6789

The following notations are used in the formulas below:

- m – MAX POINTS
- sc – SCALING CONSTANT
- a – A parameter (discrimination parameter)
- b – B parameter (difficulty parameter)
- c – C parameter (guessing parameter)
- d $[d_1, d_2, \ldots]$ – D parameter vector (threshold parameters for GPCM and GRM)
- θ – student ability
- x – student score

1PL model

The item response function (the probability of a correct response to a dichotomous item) is

$$p(\theta, x = 1) = \frac{1}{1 + e^{-sc*(\theta - b)}}$$

2PL model

The item response function (the probability of a correct response to a dichotomous item) is

$$p(\theta, x = 1) = \frac{1}{1 + e^{-sc*a*(\theta - b)}}$$

3PL model

The item response function (the probability of a correct response to a dichotomous item) is

$$p(\theta, x = 1) = c + \frac{1 - c}{1 + e^{-sc*a*(\theta - b)}}$$

GPCM model

For the Generalized Partial Credit Model (GPCM, Muraki 1992), there are m+1 possible score points (0, 1, 2, …, m). The probability of getting a score point of x is

$$p(\theta, x = 0) = \frac{1}{1 + \sum_{r=1}^{m} \exp\left[\sum_{j=1}^{r} sc * a * (\theta - b + d_j)\right]}$$

$$p(\theta, x = 1,2, \dots, m) = \frac{\exp\left[\sum_{j=1}^{x} sc * a * (\theta - b + d_j)\right]}{1 + \sum_{r=1}^{m} \exp\left[\sum_{j=1}^{r} sc * a * (\theta - b + d_j)\right]}$$

GRM model

For the Graded Response Model (GRM, Samejima's 1969), there are m+1 possible score points (0, 1, 2, …, m). The cumulative probability of getting a score of x or above is

$$p\_cum(\theta, x = 0) = 1$$

$$p\_cum(\theta, x = 1,2, \dots, m) = \frac{1}{1 + e^{-sc*a*(\theta - b + d_x)}}$$

The probability of getting a score point of x is then

$$p(\theta, x = 0,1, \dots, m - 1) = p\_cum(\theta, x) - p\_cum(\theta, x + 1)$$

$$p(\theta, x = m) = p\_cum(\theta, m)$$

Appendix III. OptimalCAT software libraries and licenses

OptimalCAT software includes the following software/libraries:

- FICO Xpress, which is proprietary software, needs **a valid license for commercial use**
- Gurobi, which is proprietary software, needs **a valid license for commercial use**
- Springframework library, which is licensed under Apache License 2.0.
- Fasterxml jackson library, which is licensed under Apache License 2.0.
- Apache Commons Math library, which is licensed under Apache License 2.0.
- License3j library, which is licensed under Apache License 2.0.
- jfreechart library, which is licensed under GNU Lesser General Public License.
- Pandas library, which is licensed under BSD 3-Clause License.
- Pandasql library, which is licensed under MIT License.
- Openpyxl library, which is licensed under MIT License.