OptimalCAT 2.5 Manual

OptimalCAT is a ***production-grade*** Computer Adaptive Testing (CAT) engine designed for both psychometricians running simulations and IT engineers deploying to production.

It builds on the shadow-test approach and leverages state-of-the-art Mathematical Programming solvers to deliver tailored, optimal tests to every student. It ensures that test blueprints and constraints are always met, and measurement accuracy is maximized.

The engine is built as a microservice with a REST API interface. It can integrate seamlessly into 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 2.5 Manual 1

OptimalCAT License 3

Prerequisites 4

Getting Started Guide (Windows version) 5

1. Test configuration 5
2. Start CAT engine 9
3. Run simulation (for Psychometricians) 11
4. Run load test (for IT engineers) 17

Appendix I. OptimalCAT engine REST API 20

Appendix II. OptimalCAT engine IRT models and parameterization 25

1PL model 25

2PL model 25

3PL model 25

GPCM model 26

GRM model 26

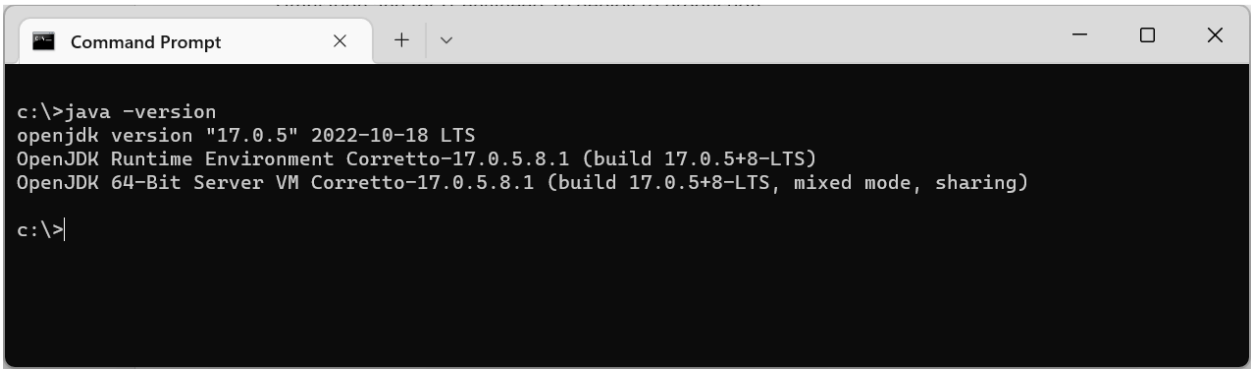
OptimalCAT License

Use	Pricing	Support and Services
Research use	Free	
Production use	Contact us for pricing	• Psychometrics Consulting • Technical Support • Customization • Solver OEM Licensing

Prerequisites

To Use OptimalCAT, you need the following:

- **Java 17 or above:** If you are not sure which Java distribution to use, try Amazon Corretto https://aws.amazon.com/corretto. Once installed, make sure to verify your Java version.



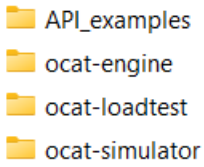
- **Microsoft Excel** (or Apache OpenOffice): editing tool for CAT test configurations.

- **Commercial Solvers (Optional):** OptimalCAT embeds the open-source SCIP solver as default and is ready to run out of the box. However, if you want to use commercial solvers* (FICO Xpress or Gurobi), you need to install the solver software and obtain licenses yourself (or contact us for an OEM solver-embedded version of OptimalCAT).

  * Commercial solvers are generally faster than open-source ones.

Getting Started Guide (Windows version)

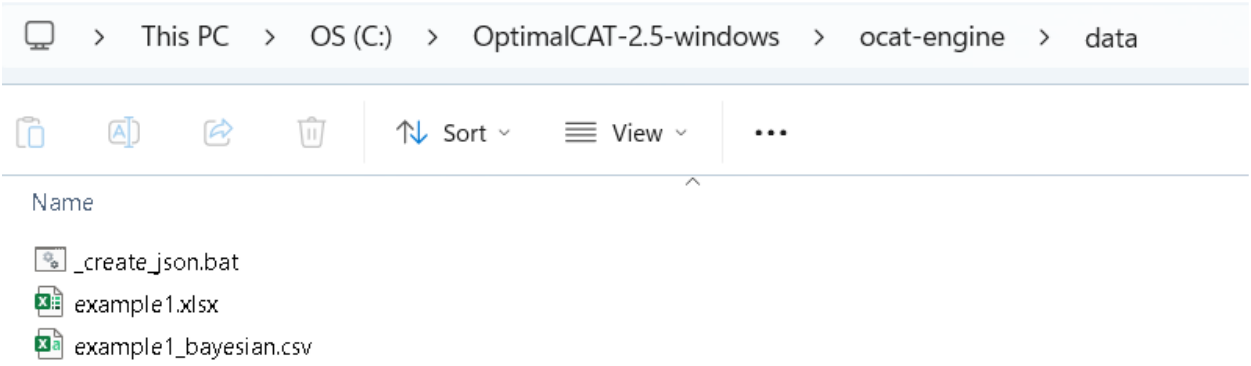
Download and unzip OptimalCAT from www.Optimal-CAT.com. In the unzipped folder, you will find the following subfolders:



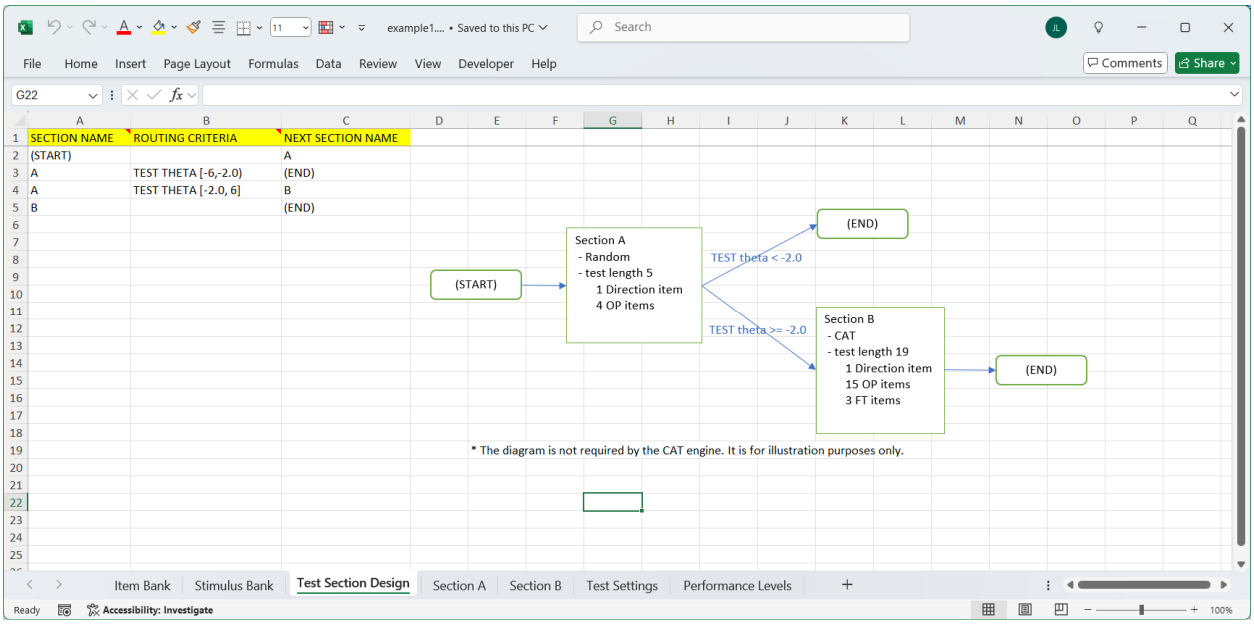
- API_examples: API examples in Java, Python and R[1].
- ocat-engine: the core OptimalCAT engine.
- ocat-loadtest: load-testing tool designed for IT engineers.
- ocat-simulator: simulator designed for psychometricians.

1. Test configuration

Navigate to the **ocat-engine\data** folder. Inside, you will find a sample test configuration file named **example1.xlsx**.



[1] No programming is required for psychometricians to set up CAT tests and run simulations. These API examples are provided to help IT engineers integrate the OptimalCAT engine into test delivery systems. You can also use these examples to build your own custom simulation clients.



Overview of the sheets in test configuration files:

- **Item Bank** contains item parameters and metadata. Columns highlighted in yellow are required by the system. You can add custom columns as needed and reference them later in your constraint definitions.
    - supported IRT Models: 1PL, 2PL, 3PL, GPCM and GRM.

- **Stimulus Bank** contains stimulus and passage information.

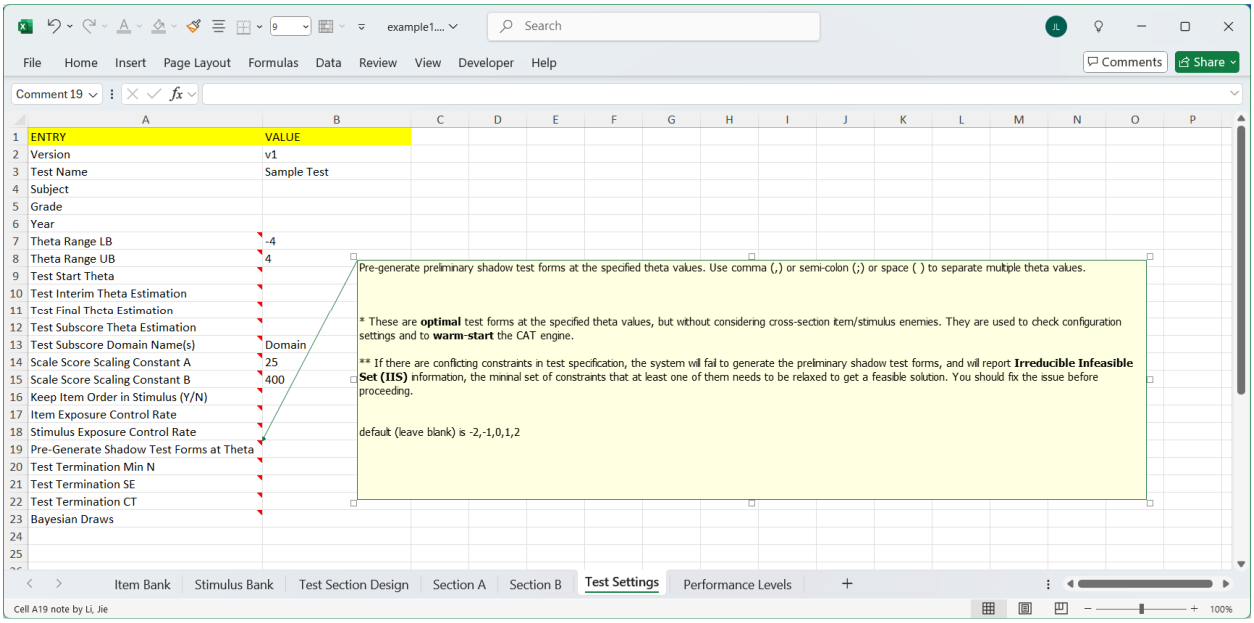
- **Test Section Design** defines test section routing rules. You can create multiple test sections and specify routing criteria between them, enabling configurations such as Multi-Stage Testing (MST). Two dummy sections are **(START)** and **(END)** to designate the beginning and the end of a test.

- **Section A**, **Section B** … (or simply **A**, **B**…): Each section defined in the **Test Section Design** sheet must have a corresponding sheet specifying its delivery format and configuration parameters. Supported test format:
    - CAT - item/unit level CAT using Maximum Fisher Information.
    - CAT_BM - item/unit level CAT using b-param matching.
    - Random - linear test with randomized item presentation.
    - Linear - linear test with a user-specified item order.
    - LOFT - linear on-the-fly test (a special case of CAT with no test form refresh)

- **Test Settings** contains test level parameters, such as theta ranges and scale score constants.

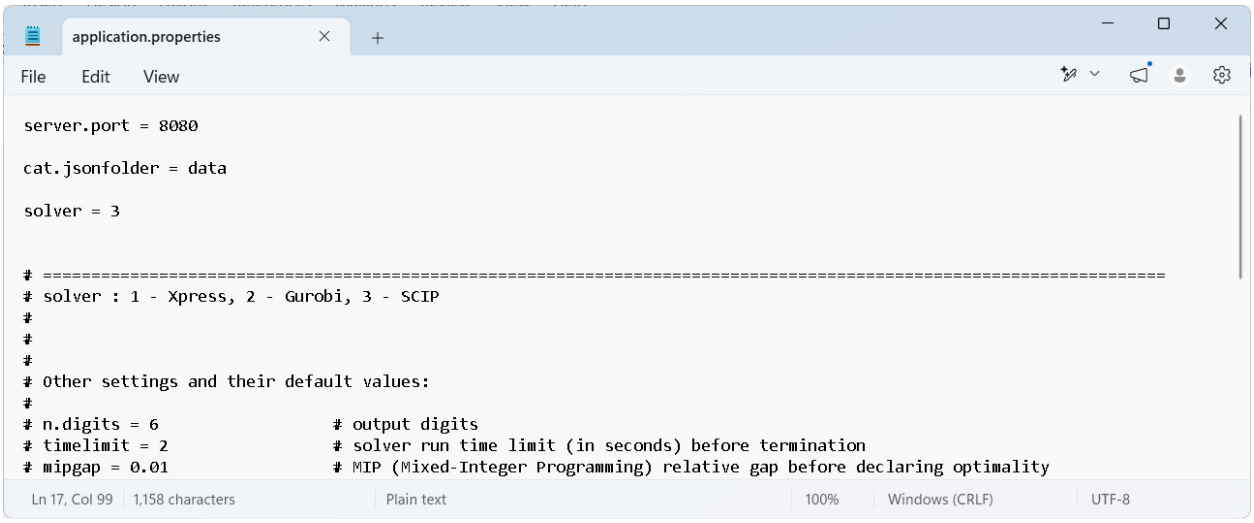
- **Performance Levels** contains cut score definitions and performance level classifications.

Cells with red triangles contain **pop-up comments** that offer additional guidance on specific fields.



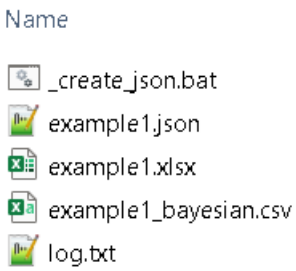
Once you have defined your test specifications in the Excel configuration files, the next step is to run **_create_json.bat**.

* **Note for Commercial Solvers:** If you installed FICO Xpress or Gurobi, navigate to the **ocat-engine** folder before running **_create_json.bat**. Open the **application.properties** file and set the **solver** property to your installed solver: 1 – Xpress, 2 – Gurobi, 3 – SCIP (default).



After running **_create_json.bat**, verify that a log file (**log.txt**) and a JSON configuration file (**example1.json**) have been generated.

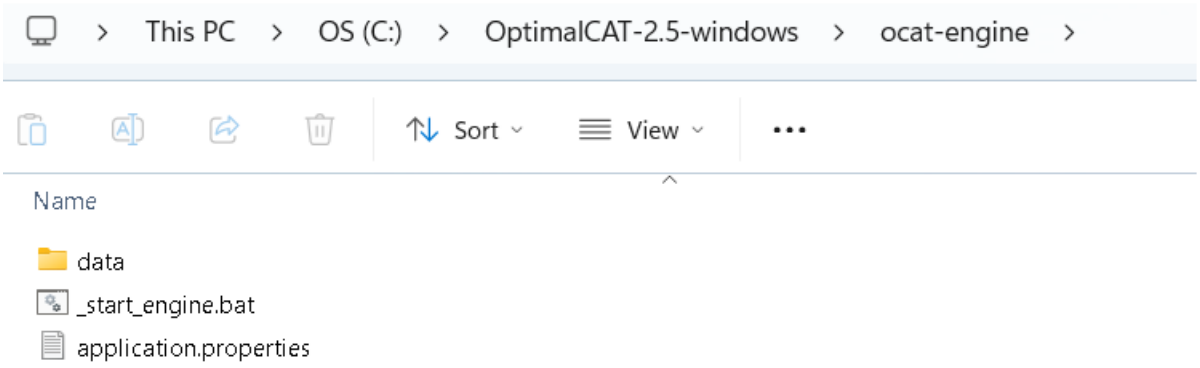
Carefully review both the screen output and the log file to ensure no errors were reported. If any errors appear, correct the issues in your Excel configuration file and re-run the script.



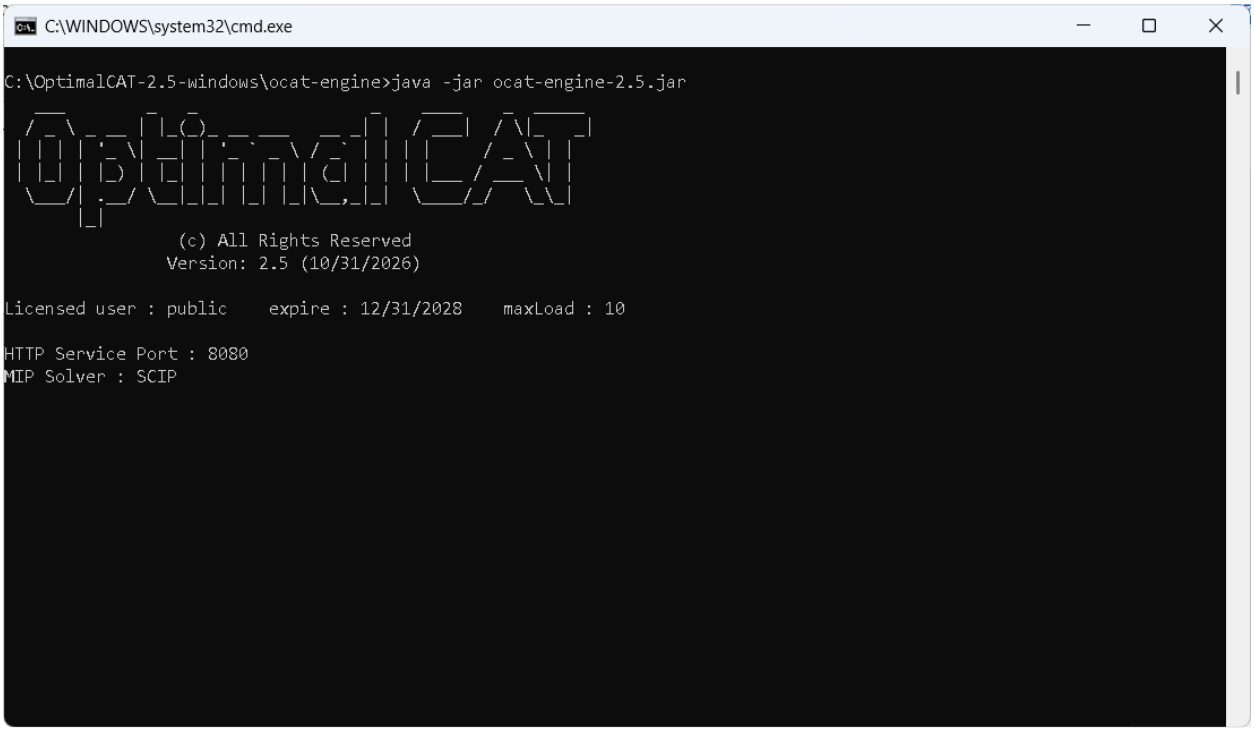
Notice: the CAT engine reads only JSON files, not test configuration Excel files.

2. Start CAT engine

To start the OptimalCAT engine, navigate to the **ocat-engine** folder and run **_start_engine.bat**.



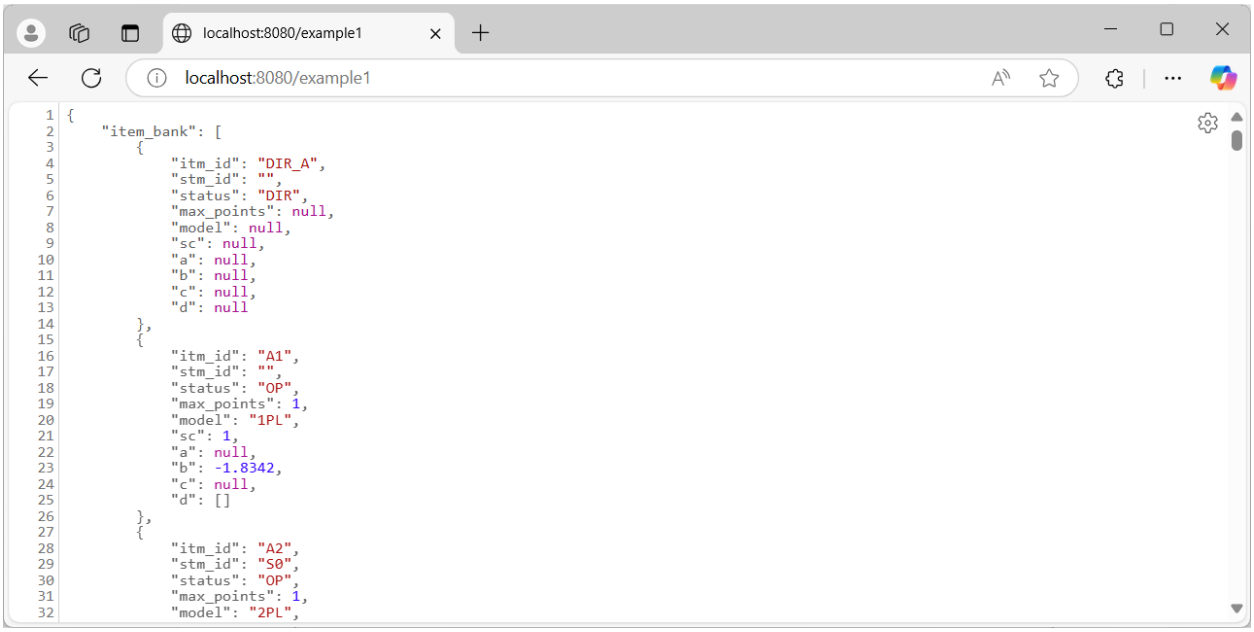
A command window will open, confirming that the engine has started. It shows that the engine is started and listening on port 8080, MIP solver is Xpress, Gurobi or SCIP (depending on the settings in **application.properties** file) and test example1 is loaded. **Keep this command window open.** The engine is now active and ready to process incoming API requests.



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.

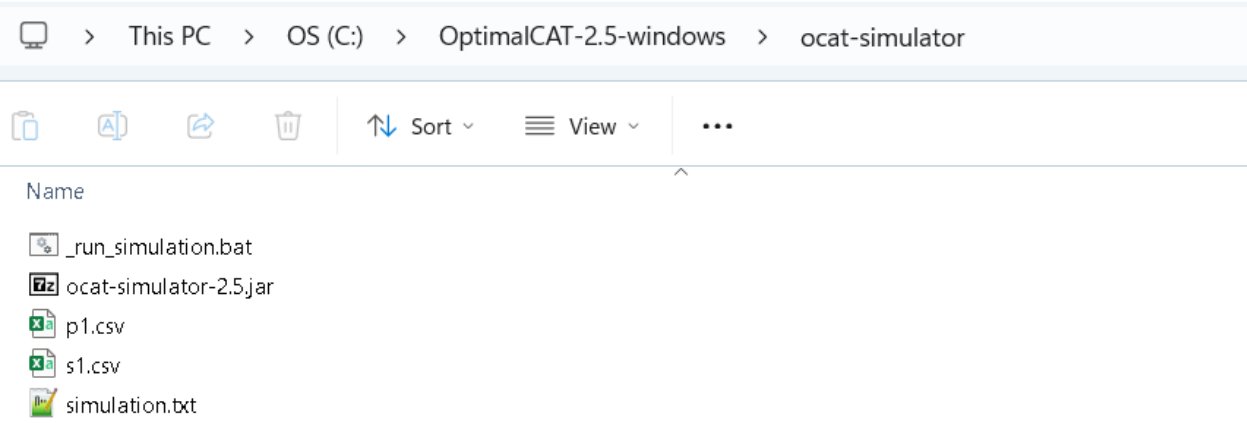
For GET request endpoints, you could use a browser to access them, for example,



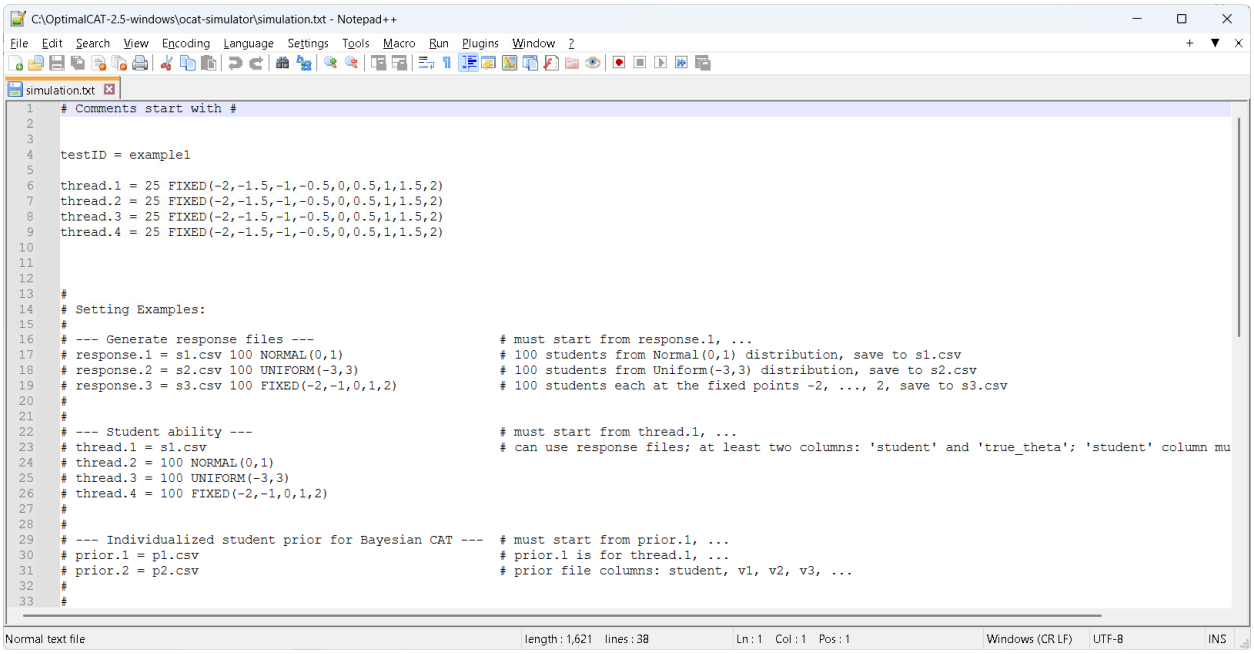
For POST request endpoints (/cat and /rescore), they are for IT engineers to integrate with test delivery systems for test administration and scoring. See “Appendix I. OptimalCAT engine REST API” for details.

3. Run simulation (for Psychometricians)

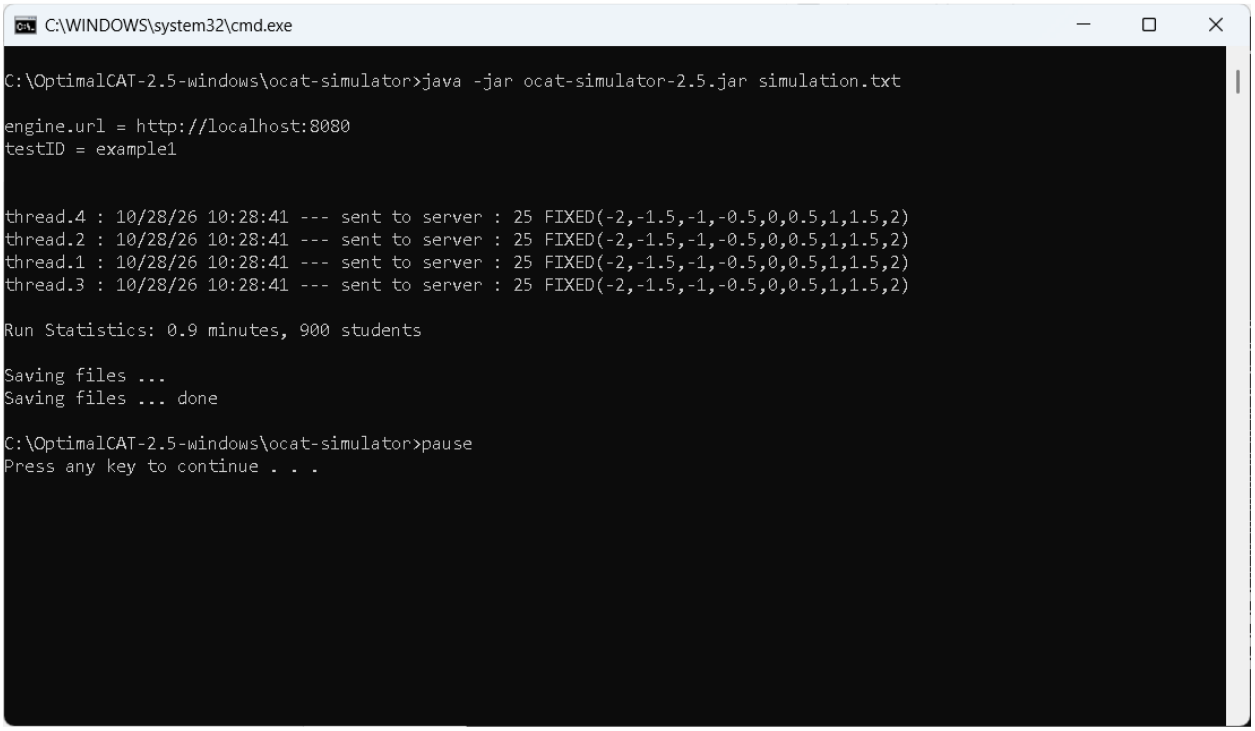
With the CAT engine up and running, you can now start the simulation. Navigate to the **ocat-simulator** folder.



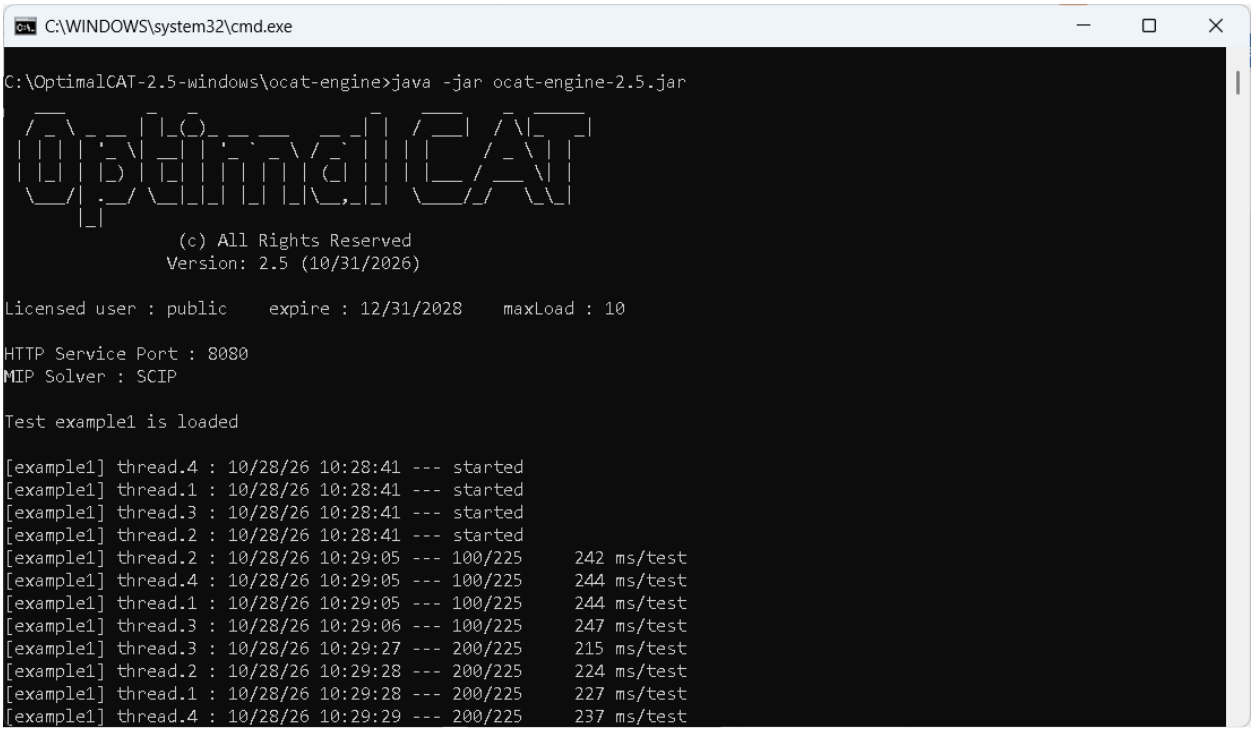
Use the **simulation.txt** file to configure the parameters for running your simulation.



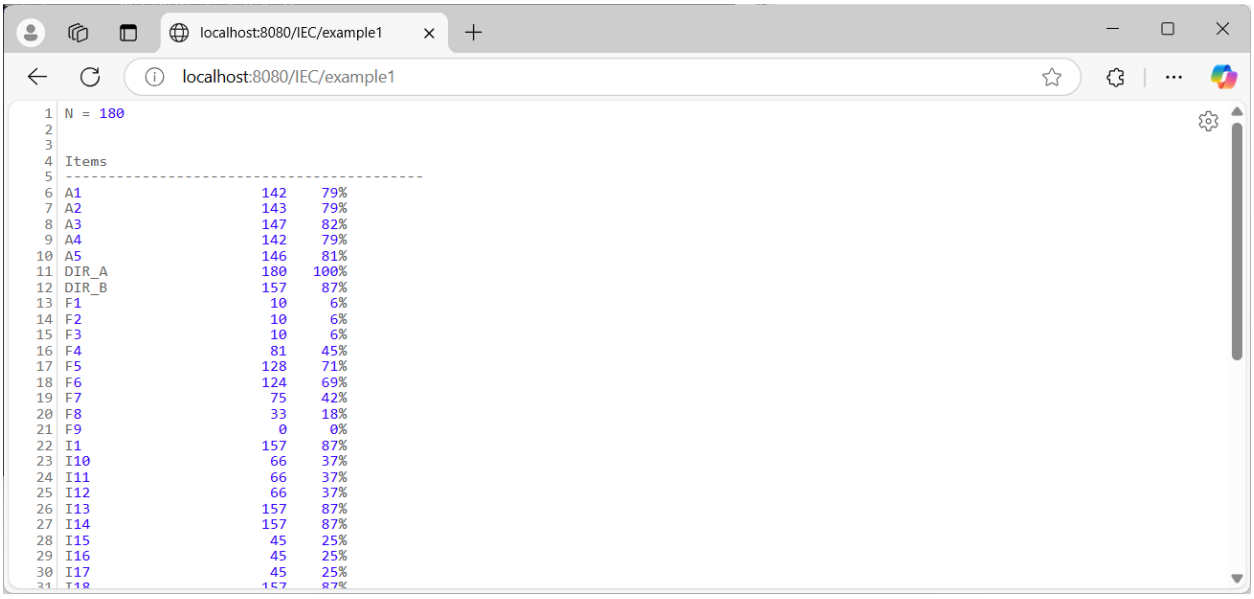
Run **_run_simulation.bat** to start the simulation. Once the process finishes, the console will display the output like below.



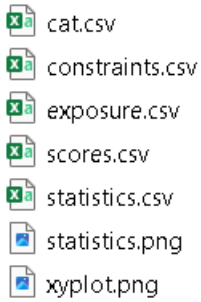
The simulation runs on the server side to minimize API traffic between the client and server. You can monitor its progress directly in the server window.



While the simulation is running, you could monitor real-time item and stimulus exposure status through the /IEC/{testID} endpoint.



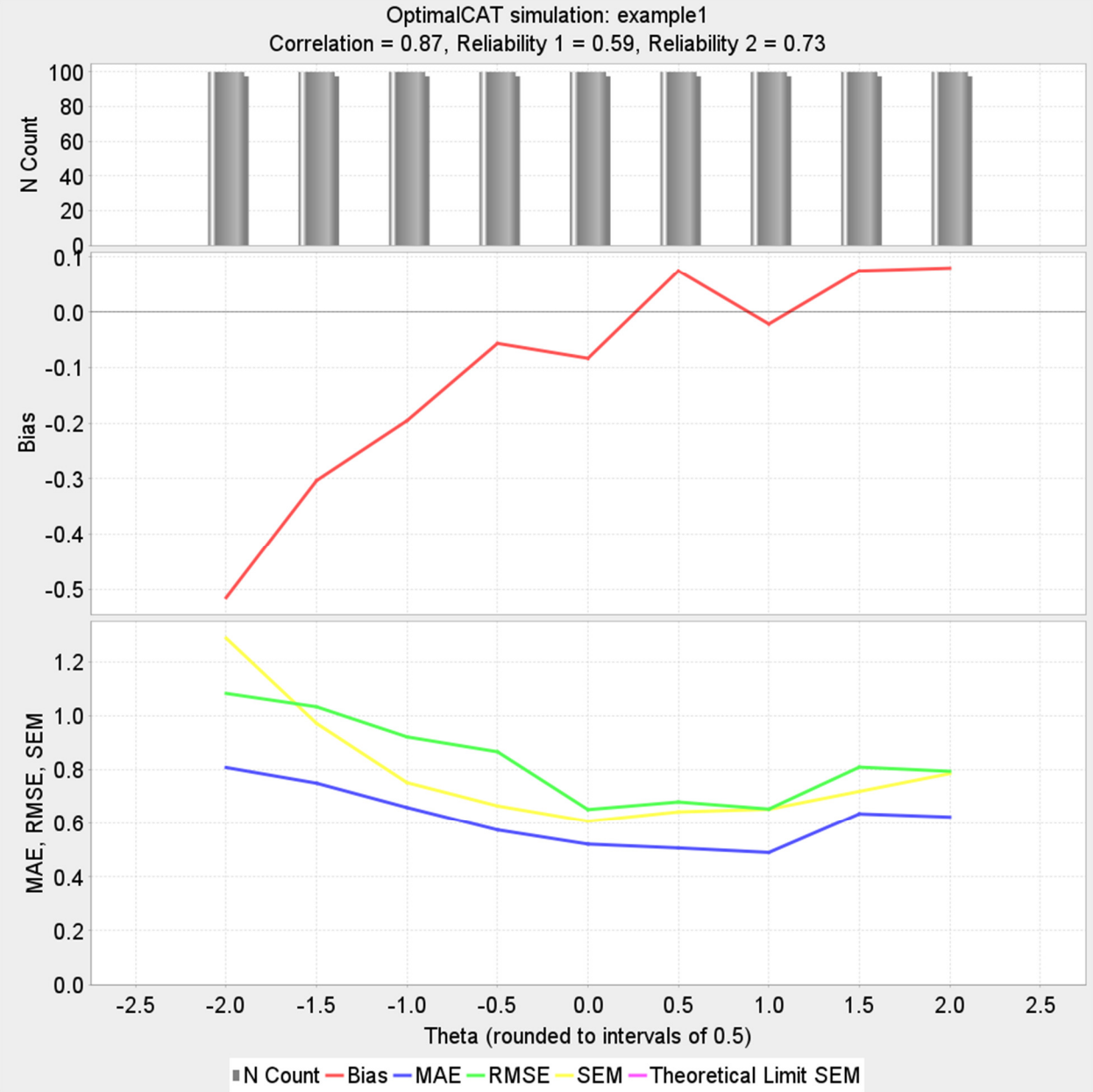
After the simulation completes, results are saved to a newly created timestamped directory named {testID}_{date}_{time} inside the **ocat-simulator** folder, which contains seven output files:



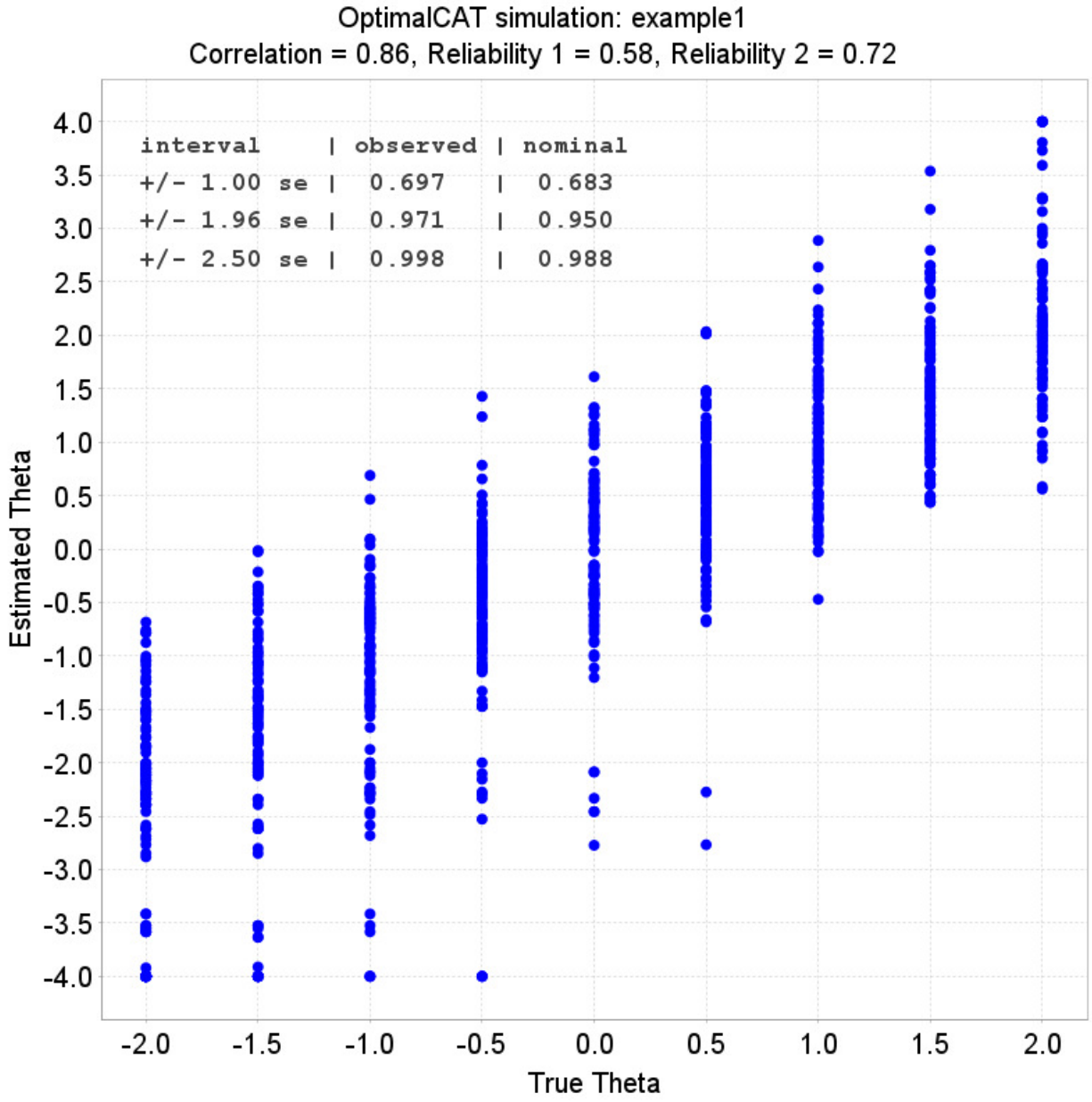
- cat.csv – this file contains detailed test session information, item by item.
- constraints.csv – this file contains constraints details.
- exposure.csv – this file contains exposure counts for both items and stimuli.
- scores.csv – this file contains final scores and sub-scores.
- statistics.csv – this file contains the following measurement statistics:
    - N Count
    - Bias - bias is $\hat{\theta}_j - \theta_j$
    - MAE – mean of absolute error $\left|\hat{\theta}_j - \theta_j\right|$
    - RMSE - Root Mean Square Error
    - SEM - Standard Error (SE) of Measurement (or standard deviation of posterior theta draws in Bayesian CAT)

- Theoretical Limit SEM - calculated using the pre-generated shadow test forms at each specified theta point (see "**Pre-Generate Shadow Test Forms at Theta**" in **Testing Settings** sheet in the Excel test configuration files).
  - **Notice: Theoretical Limit SEM is displayed (1) only for traditional CAT (not for Bayesian CAT), and (2) only when all students see all test sections defined in the test configuration.**
- 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 at the estimated ability $\hat{\theta}_j$
- Observed coverage of true theta θ_j falls within intervals of $\hat{\theta}_j \pm 1.0\ SE_j$, $\hat{\theta}_j \pm 1.96\ SE_j$, and $\hat{\theta}_j \pm 2.50\ SE_j$ (nominal coverage is 0.683, 0.95, and 0.988 respectively).

- statistics.png – chart using the numbers in statistics.csv.
- xyplot.png – chart of estimated theta ($\hat{\theta}_j$) vs. true theta (θ_j), and coverage.



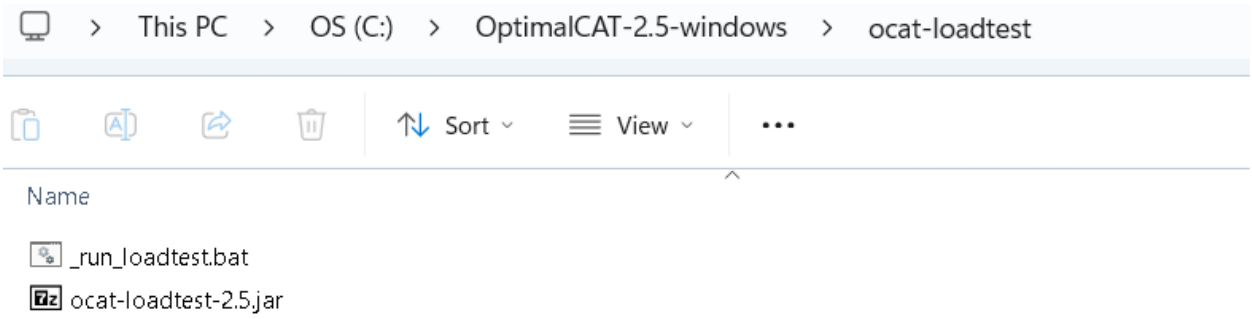
statistics.png


OptimalCAT simulation: example1
Correlation = 0.86, Reliability 1 = 0.58, Reliability 2 = 0.72
interval | observed | nominal
+/- 1.00 se | 0.697 | 0.683
+/- 1.96 se | 0.971 | 0.950
+/- 2.50 se | 0.998 | 0.988
Estimated Theta
True Theta
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
-2.0
-2.5
-3.0
-3.5
-4.0
-2.0
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0

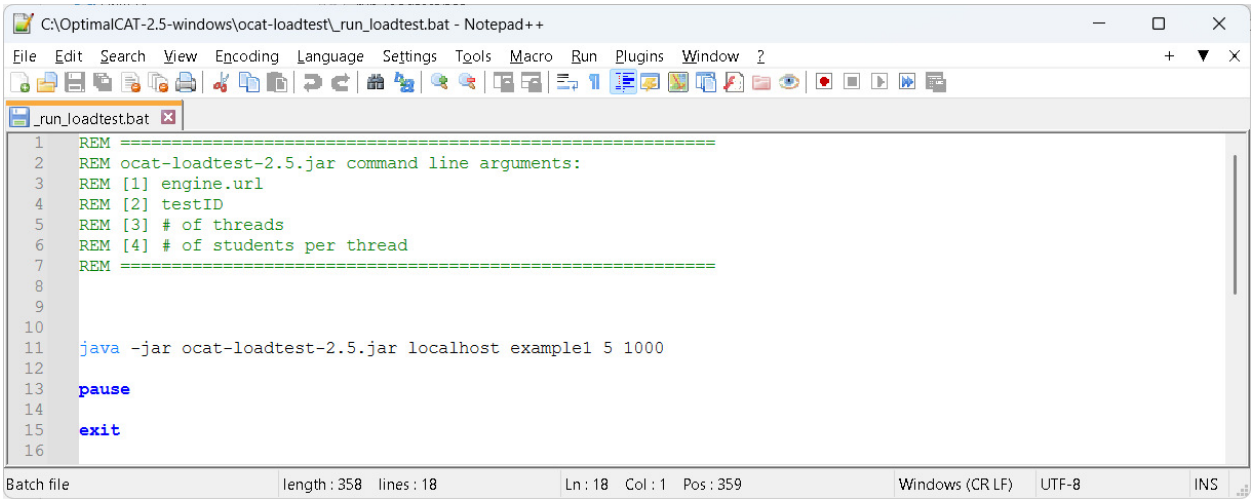
xyplot.png

4. Run load test (for IT engineers)

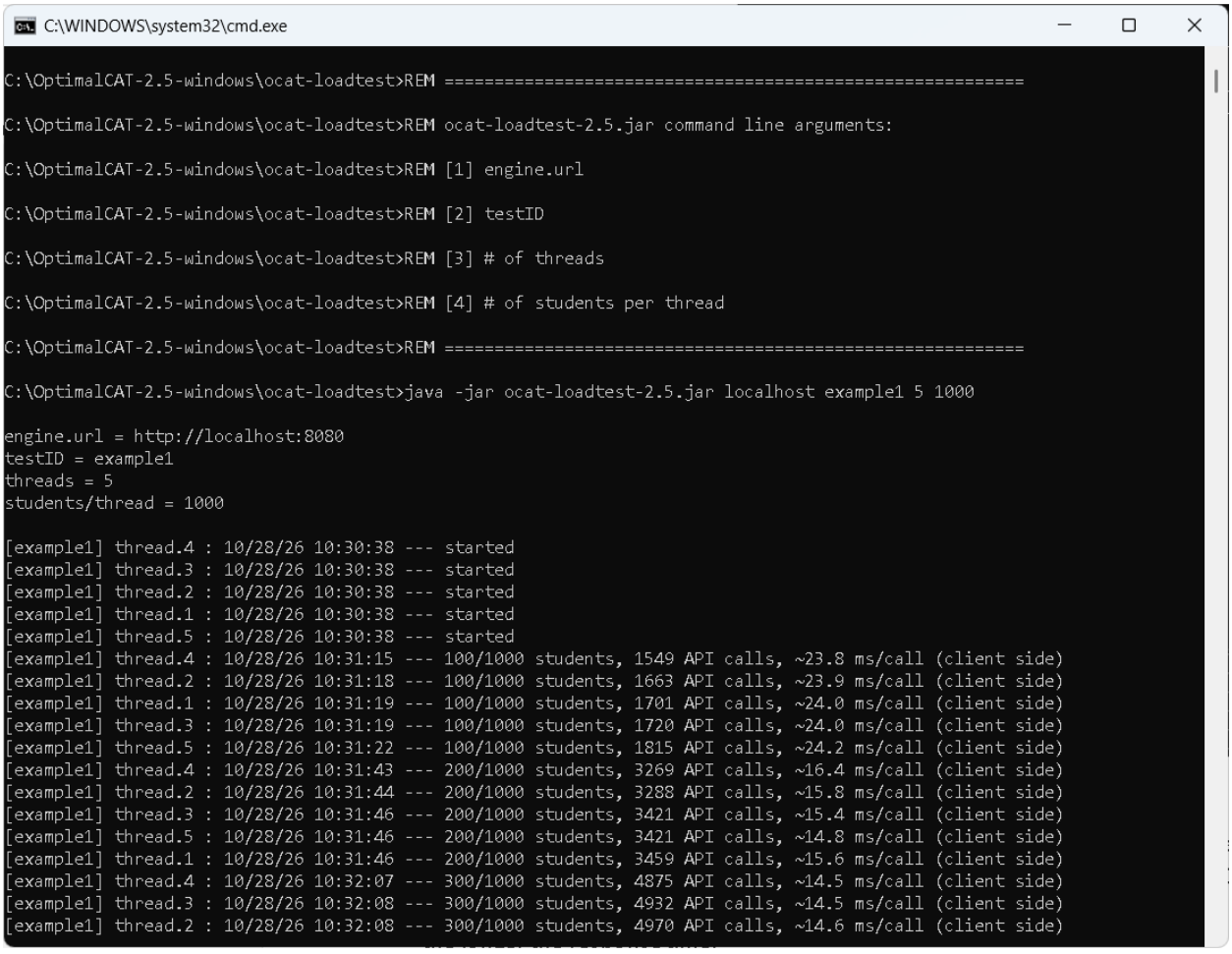
To evaluate the actual response time and performance of engine API calls under realistic operational conditions, a load testing utility is provided in the **ocat-loadtest** folder.



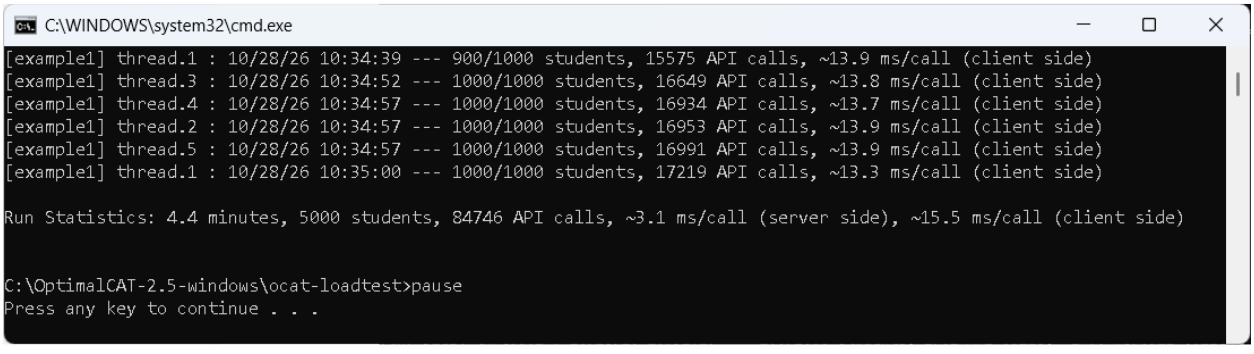
You can modify the command-line arguments to suit your specific load testing requirements.



Run **_run_loadtest.bat** to start the load test. A console window will open 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 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.

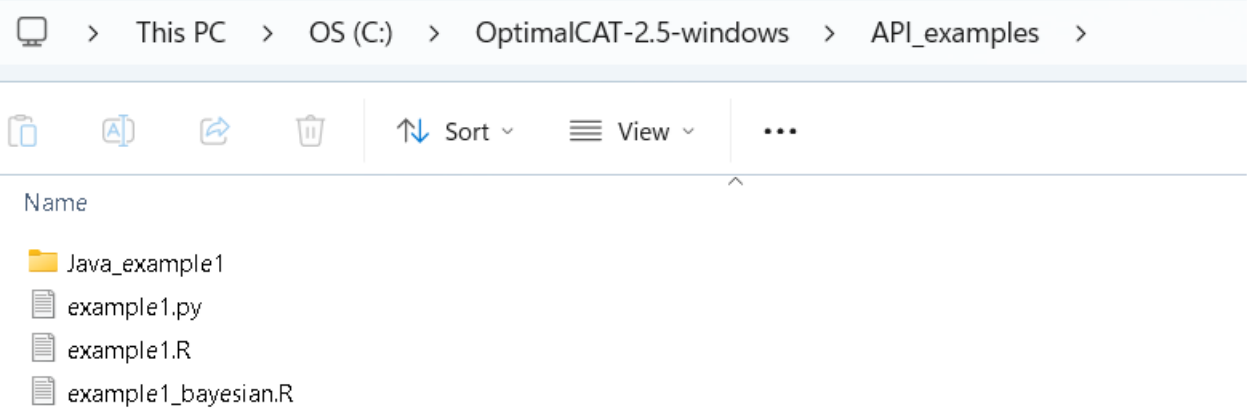
For most assessment programs, we believe one or two powerful servers should be enough. For very large testing programs with high concurrency requirements, we suggest creating a server pool and putting a load balancer in front, e.g., using Load Balancer - Amazon Elastic Load Balancer (ELB) - AWS.

Appendix I. OptimalCAT engine REST API

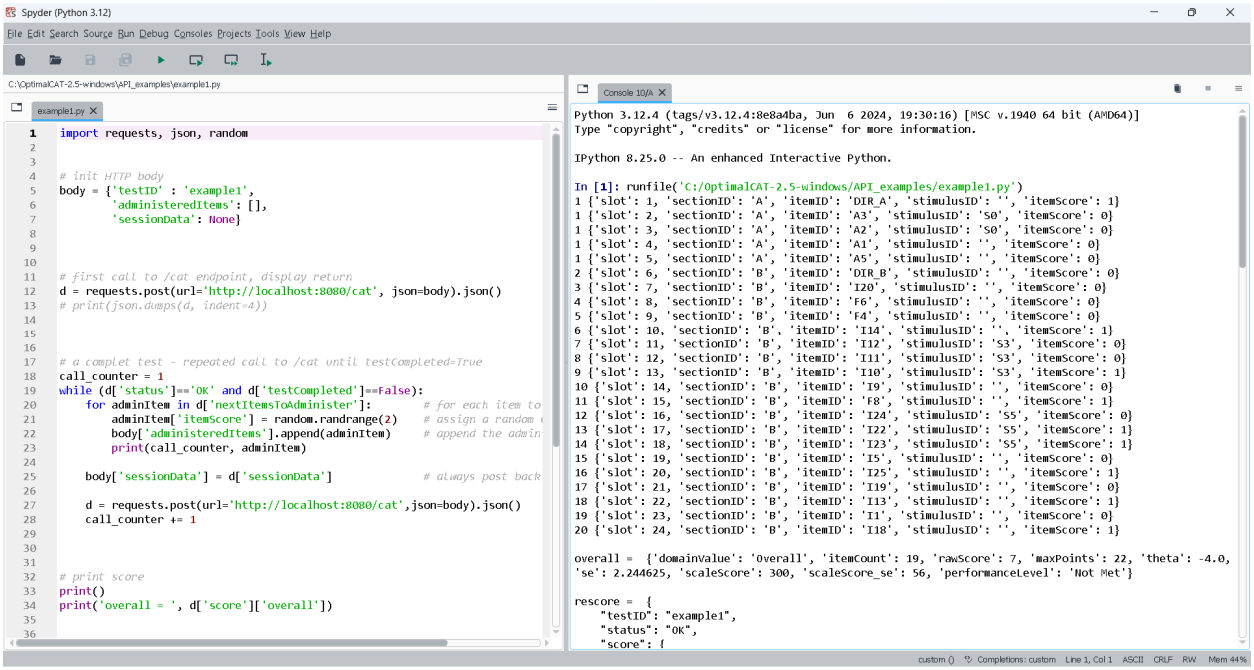
The CAT engine has the following public REST API endpoints:

- / - 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.



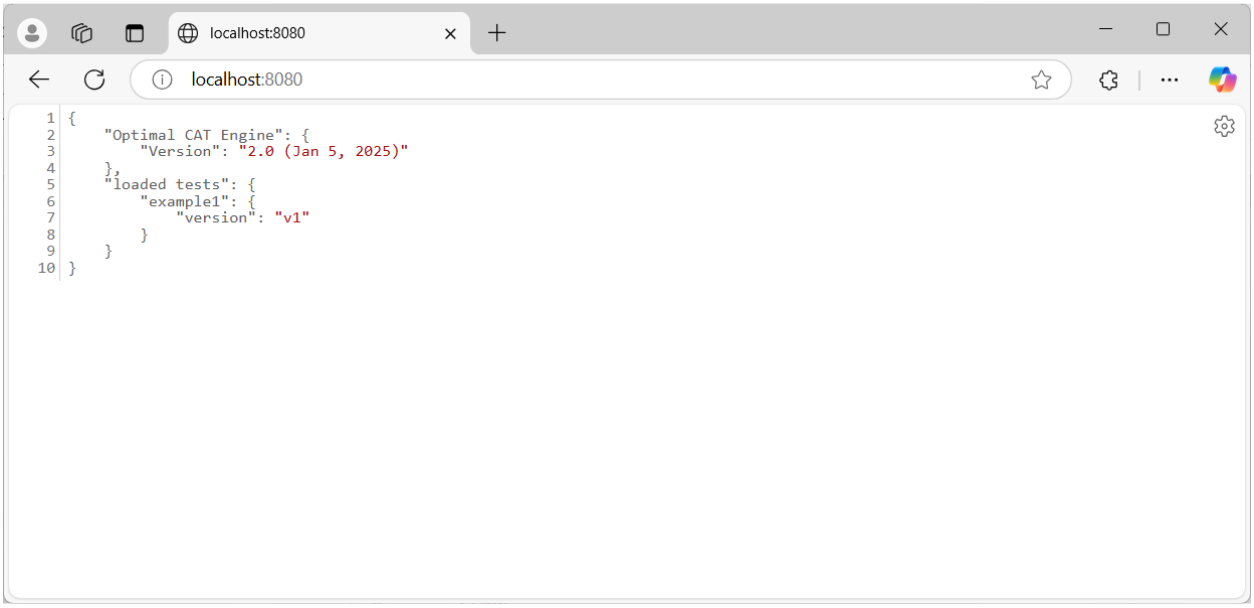
For example, this is the Python API example (example1.py).



GET /

Parameters: (none)

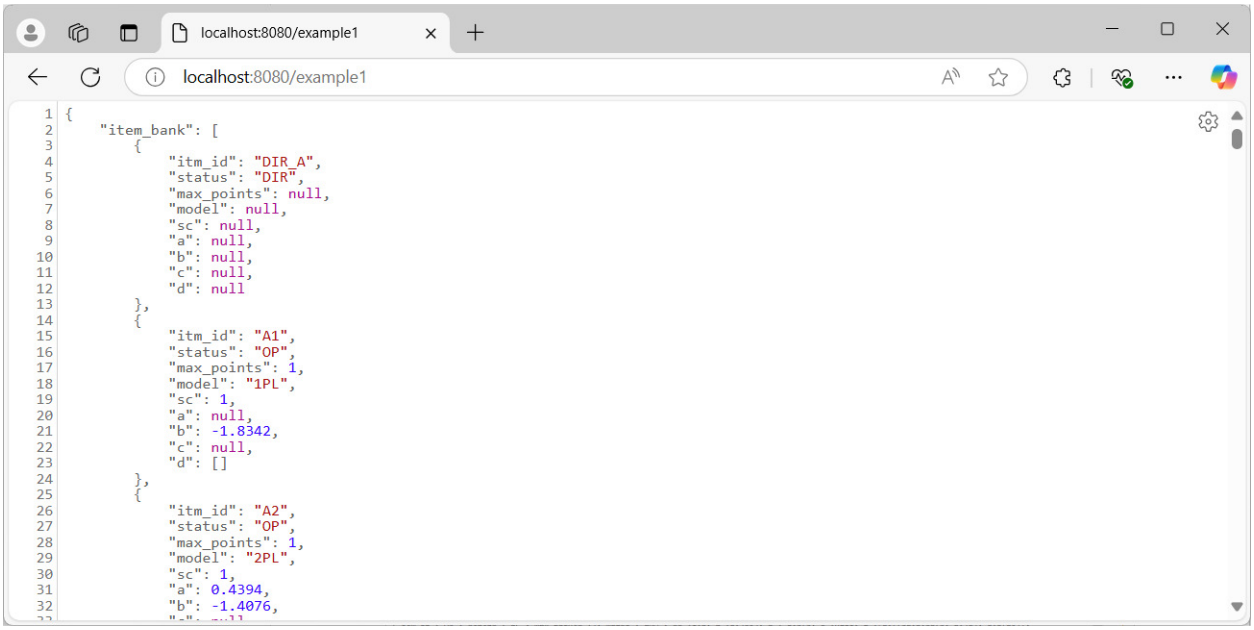
Returns: basic info of OptimalCAT engine and loaded test



GET /{testID}

Parameters: testID String

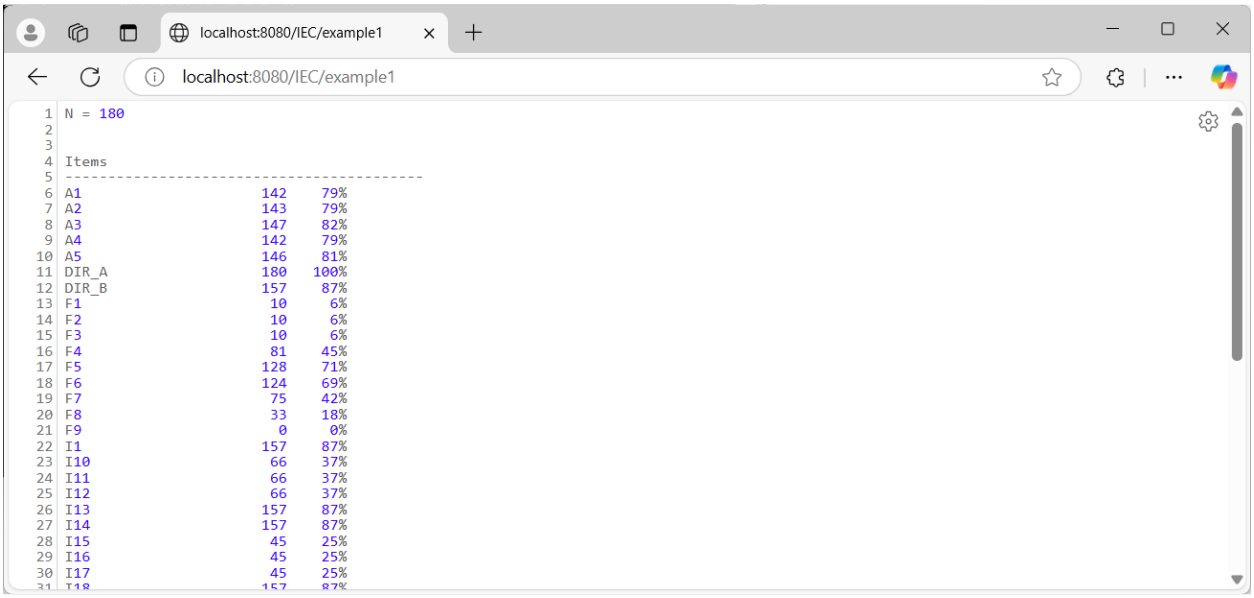
Returns: test JSON file, including item bank, test sections, test settings, etc.



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" sheet in test configuration Excel files.

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
- s – 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

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

2PL model

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

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

3PL model

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

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

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

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

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

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

$$p_cum(\theta, x = 0) = 1$$

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

The probability of getting a score point of x is then

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

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