Foundational Understanding of Item Response Theory (IRT)

High-level Concept

What problem does IRT solve, and why was it invented? Item Response Theory (IRT) is a psychometric framework developed to overcome limitations of earlier **Classical Test Theory (CTT)** in measuring abilities and skills. Under CTT, a test-taker's score is simply the total of correct answers, which makes comparisons difficult across different test forms or populations. IRT, by contrast, was invented to focus on the **item level**: it models each question's characteristics and the examinee's ability on a common scale [1]. This allows for *invariant measurement* – item difficulty can be assessed independently of who took the test, and an examinee's ability can be estimated independently of the specific set of items they answered. IRT thus solves critical problems like **test form equating** (comparing scores across different test versions), designing **adaptive tests**, and maintaining fair, consistent scoring over time [1] [2].

Historically, IRT emerged in the 1950s–60s from the work of psychometricians like Frederic Lord and Georg Rasch to address these needs [3]. It provided a more powerful alternative to CTT by recognizing that *not all items are equal* – some are harder, some discriminate better, etc. IRT was adopted for high-stakes exams (e.g. GRE, GMAT) because it yields more accurate and fair ability estimates [4]. In summary, IRT was invented to create **adaptive, fair, and precise assessments** by modeling how individual test items function across different levels of student ability [5] [6]. (This makes possible innovations like computer-adaptive testing and unbiased item analysis that were impractical under classical methods.)

<small>*Side Note:* **Classical Test Theory (CTT)** assumes each test-taker's observed score = true score + error, and treats all items equally in contributing to the score. It lacks a way to model item difficulty or discrimination. **Item Response Theory (IRT)** introduced item-specific parameters to overcome this, enabling more nuanced measurement and adaptive testing.</small>

Core Idea

How does IRT model the relationship between a person's ability and an item's difficulty? The core idea of IRT is that each person has a latent ability level (often denoted θ), and each item has certain parameters (like difficulty), and **the probability of a correct response is a mathematical function of both** [7]. Intuitively, think of item responses as follows: If an item is very easy *relative* to your ability, you're very likely to get it right; if it's far above your ability, you'll likely get it wrong. IRT captures this intuition with an **Item Characteristic Curve (ICC)** – an S-shaped curve that plots the probability of a correct answer (y-axis) against the person's ability θ (x-axis) [8].

In the simplest case (1-parameter model), an item's **difficulty (b)** is the point on the ability scale where a test-taker has a 50% chance of answering correctly [9]. For example, an item with b = 0 (average difficulty) gives a 50% success chance to someone with average ability (θ=0 in a standardized scale). A person with

ability higher than the item's b will have >50% chance of getting it right, and below b <50%. By modeling each item's curve, IRT places **people and items on the same latent scale** [10]. This means we can talk about a person's ability in absolute terms (e.g. θ = +1 might mean above average ability) and an item's difficulty in the same terms (e.g. b = +1 is a relatively hard question).

Crucially, IRT's person and item parameters are **invariant** under certain conditions: an item's difficulty does not change depending on who takes it, and a person's estimated ability would be the same regardless of which specific set of calibrated items they took [5]. This makes scores more interpretable and comparable. In summary, IRT's core concept is modeling *the probability of success as a function of ability and item properties*. The higher the ability relative to an item's difficulty, the higher the success probability – akin to *"ability – difficulty = odds of correct"* in a logistic sense [9].

<small>*Side Note:* **Latent Trait (θ)** – an unobservable personal attribute or ability that IRT seeks to measure (e.g. math proficiency, English fluency). **Item Difficulty (b)** – a parameter indicating how challenging an item is; it's the ability level at which the item has about 50% chance of being answered correctly [9]. If θ and b are on the same scale, an item with b = 2.0 requires a high ability (2 std dev above mean) for 50% success.</small>

Mathematical Essence (light)

What are the basic logistic functions used in 1PL, 2PL, 3PL models (conceptually, not heavy on formulas)? IRT models typically use a logistic (S-shaped) function to model the probability that a person with ability θ answers an item *i* correctly. The general form is:

$$P(\text{correct}|\theta) = f(\theta, a_i, b_i, c_i)$$

where *a*, *b*, *c* are item parameters that define the curve's shape. Conceptually: - **1PL Model (Rasch Model):** Only **1 parameter** per item – *difficulty* (b). All items are assumed to have equal discrimination, and guessing is not modeled. The probability of a correct answer is a logistic function of the difference between person ability and item difficulty:

$$P_i(\text{correct}|\theta) = \frac{1}{1 + \exp[-(\theta - b_i)]}$$

This yields the classic S-curve: at θ = b_i, the probability is 50%. If θ is one unit above b_i, the odds of correct vs. incorrect increase by e (the curve's steepness is fixed in Rasch). Difficulty *b* thus shifts the curve along the θ axis [9]. An item with b = -1.0 is easier (curve shifted left: even a low-ability person has ~50% chance), whereas b = +2.0 is hard (only a high-ability person reaches 50% chance). - **2PL Model:** Adds a second parameter **discrimination (a)** per item. Discrimination *a* reflects how sharply the probability increases as ability exceeds the difficulty. A higher *a* means the item **better differentiates** between examinees of slightly different ability around the difficulty point – the curve is steeper. Lower *a* yields a more gradual slope (people with abilities above *b* still have only modestly higher probability than those below *b*) [11]. In the 2PL formula, *a_i* multiplies (θ – b_i) inside the logistic function: a large *a* makes the 50%→90% probability transition occur in a narrow θ range. Thus, 2PL accounts for items varying in how informative they are. (For instance, an ambiguous question might have low discrimination *a* because even high-ability students sometimes miss it.) - **3PL Model:** Adds a third parameter **guessing (c)**. This represents the lower asymptote of the ICC – essentially the probability that even a very low-ability examinee could get the item

right by guessing. In multiple-choice questions with 4 options, *c* might be around 0.25 (25%) [12] [13]. The 3PL equation is often written as:

$$P_i(\text{correct}|\theta) = c_i + (1 - c_i)\frac{1}{1 + \exp[-a_i(\theta - b_i)]}$$

Here, when θ is extremely low, probability approaches *c* (not 0). As θ increases, the curve rises from the *c* baseline toward 1. The *c* parameter captures **guessing or pseudo-chance** performance [12] [14]. This is important for exams where a completely unskilled person could still get some items right by luck. The 3PL model is widely used in high-stakes tests (like SAT/GRE multiple-choice) to account for guessing behavior.

In all these logistic models, the **ability θ is a latent trait** usually scaled with mean 0 and SD 1 (like a z-score). Items parameters are on the same scale. The logistic function form gives the characteristic S-curve: at low θ, $P(\text{correct})$ is near the guessing floor (or 0 in 1PL/2PL); at high θ, $P(\text{correct})$ approaches 1 (assuming no ceiling effect). The point where $P=0.5$ is the difficulty b. The slope at that point is governed by *a*. And the lower bound is *c*. These parameters (a, b, c) **shape the ICC** [15] [16]. By fitting such models to real test data (using specialized software), we can estimate each item's parameters and each person's ability.

To summarize without heavy math: **1PL (Rasch)** uses difficulty to shift the curve left/right; **2PL** adds discrimination to stretch or squeeze the curve's steepness; **3PL** adds a guessing floor raising the curve's lower tail [17]. Each extension makes the model more flexible in describing real item behaviors at the cost of additional complexity (and data needed to estimate reliably).

<small>*Side Note:* In practice, there are also **4PL models** (adding an upper asymptote <1 if extremely hard items might never be answered correctly even by highest ability, or if there's a chance of careless errors causing <100% max probability). But 4PL is rare in educational testing, mostly of theoretical interest [18]. Beyond binary scored items, there are **polytomous IRT models** for partial credit, rating scales, etc., which use analogous ideas.</small>

Adaptive Testing Flow

How is IRT used in computer-based adaptive testing? In **Computerized Adaptive Testing (CAT)**, IRT is the engine that allows tests to *adapt in real-time* to an examinee's ability. The typical flow of an IRT-based adaptive test is:

- **Initial Ability Estimate:** Since the test doesn't know the examinee's ability at the start, it begins with an initial guess. Often this is set to an average level (θ = 0 on the IRT scale) or based on any prior information (if available). Some systems start everyone at medium difficulty to avoid extremes [19].

- **First Item Selection:** The test system selects an opening question, usually of medium difficulty, so as not to discourage low-ability examinees or bore high-ability ones [20]. The goal is a question that maximizes information given the initial ability estimate – roughly, an item with difficulty near θ=0 in this case. After the examinee answers, the test scores the response (correct/incorrect).

- **Ability Estimation:** Using the IRT model and the examinee's responses so far, the computer updates its estimate of the examinee's ability θ. This can be done via **maximum likelihood estimation (MLE)**

or Bayesian methods (e.g. MAP or EAP estimators that use a prior) [21]. For example, if the first item was answered correctly, the ability estimate goes up; if wrong, it goes down. The algorithm essentially finds the θ that would most likely produce the observed pattern of answers to the administered items [21]. Initially, after one question, this estimate is rough, but it guides the next step.

- **Item Selection (Iterative):** Given the updated ability estimate, the CAT algorithm selects the next item that is most informative at that estimate. Informally, this often means choosing an item with difficulty close to the current θ (and with high discrimination) to maximize the test's ability to measure the examinee [19] [22]. In IRT terms, the item with the highest Fisher information at that θ is chosen – this tends to be where θ ≈ b for a high-*a* item. Sometimes randomness or content constraints are added, but fundamentally the item bank is scanned for a question that will optimally hone in on the examinee's true ability. The examinee answers the selected item, and then ability is re-estimated. This answer-by-answer adaptation continues in a loop [22].

- **Termination (Stopping Rule):** The test continues selecting items and updating θ until a preset stopping criterion is met [23]. Common stopping rules include:

- A fixed number of questions administered (e.g. the test will always end after 20 items).
- **Precision threshold:** stop when the ability estimate's standard error falls below a certain level, meaning the score is precise enough [23]. For instance, the test might stop once it is 95% confident in the ability estimate within ±0.3 θ.
- Content coverage: ensure certain topics or skill areas have been tested, then stop.
- Or a combination of these (many operational CATs use a max item count or time limit alongside a precision check).

- **Scoring:** Once stopped, the final ability estimate (θ_final) is taken as the test score (often converted to a scaled score or percentile for reporting). Thanks to IRT, this θ_final is based on both the difficulty of questions answered and the pattern of right/wrong responses, not just the raw number correct. In an adaptive test two examinees might both get, say, 50% of items correct, but if one was answering much harder questions, their ability estimate will be higher. CAT scoring reflects that by using the item parameters in the estimation [24].

Throughout this process, IRT is crucial because it provides the **mathematical model to estimate ability from responses and to predict how the person might do on other items**. The adaptivity ensures that each examinee gets items tailored to their level – harder questions if they're doing well, easier if they're struggling [19]. This yields several benefits: tests can be shorter (since we're zeroing in on ability efficiently) yet maintain accuracy, and each examinee is challenged appropriately (reducing boredom and frustration) [25]. For example, high-achievers aren't stuck answering a dozen trivial questions – the test quickly moves to questions that challenge them and pinpoint their skill, whereas a novice isn't overwhelmed by a string of impossibly hard items.

In practice, an adaptive test might look like this: Everyone starts with a medium item around b=0. If you get it right, the system might jump to an item of b ≈ +0.5 or +1.0 difficulty; if you get it wrong, next item might be b ≈ -0.5. As the test progresses, the jumps get smaller and cluster around an ability value that seems to fit your performance. When the test ends, that value is your score. The underlying IRT algorithm has been

continuously refining its estimate of your θ after each item, much like a doctor narrowing down a diagnosis with each new symptom or test result.

To sum up, **IRT-powered adaptive testing** involves an iterative loop of: estimate ability → select optimal item → get response → update ability → repeat, until a stopping rule is met [22]. This produces a personalized test for each examinee and an efficient, precise measurement of their ability.

<small>*Side Note:* A variation of item-level CAT is **multi-stage testing (MST)**, used by some exams like the GRE. In MST, instead of adapting every question, the test adapts in stages (groups of questions). For instance, all examinees get an initial module of items; then the next module's difficulty is chosen (easy, medium, or hard) based on performance in the first module [26]. This is a hybrid between fully adaptive and fixed forms, allowing some review within a module. MST still relies on IRT for calibrating item difficulties and scoring, but offers practical advantages like allowing examinees to review answers within a module, which pure CAT typically doesn't allow.</small>

Modern Extensions of IRT and AI Approaches

How can IRT be extended or replaced with modern AI methods (Neural IRT, Deep IRT, Bayesian IRT, RL, LLMs)? In recent years, researchers have been blending IRT with machine learning and AI techniques to address some limitations of traditional IRT models and to leverage new data sources. Here are some key modern extensions:

- **Bayesian IRT:** Traditional IRT parameter estimation often used maximum likelihood, but **Bayesian methods** have become common, using approaches like Markov Chain Monte Carlo or variational inference. Bayesian IRT allows one to put *priors* on parameters and obtain full posterior distributions for abilities and item parameters. This is helpful when data is sparse or to incorporate expert knowledge. For example, the open-source R package *brms* can fit IRT models in a Bayesian framework using Stan, providing uncertainty estimates for each person's ability [27]. Similarly, the Python library **py-irt** uses variational Bayes to fit 1PL, 2PL, 4PL models, even allowing hierarchical priors (multilevel IRT) [28]. The benefit of Bayesian IRT is more robust inference with credible intervals for abilities, and the flexibility to extend IRT models (e.g. latent regression, multidimensional IRT) naturally in a probabilistic programming setting.

- **Neural IRT & Deep Learning Models:** One emerging direction is using neural networks to enhance or replace the logistic function in IRT. **Neural IRT** can refer to methods where we use neural network architectures to learn item-person interaction patterns. For example, a **Variational IRT (VIRT)** model might treat item difficulties and abilities as latent variables to be learned via neural networks. Researchers have demonstrated neural networks that jointly learn an IRT model and even generate new items. One study showed a **neural IRT-3PL model** outperforming other cognitive models in an adaptive testing simulation [29] [30]. Another approach, **Deep-IRT**, combines IRT with deep knowledge tracing: Yeung (2019) proposed a model where a deep network (DKVMN – Dynamic Key-Value Memory Network) tracks a student's learning over time and periodically yields estimates of the student's ability and items' difficulties, which are then fed into an IRT equation to predict responses [31]. Essentially, the deep model handles temporal dynamics (learning, forgetting) while IRT provides an interpretable measurement layer. Deep-IRT was shown to retain the predictive performance of pure deep learning while *adding* interpretability – for instance, the network could output a student's evolving θ and an item's inferred b over time [32]. In general, neural network extensions of IRT

(sometimes called “Neural Cognitive Diagnosis” or similar in education research) allow modeling more complex interactions or incorporating additional features (e.g. time spent, problem content) that a simple logistic IRT can’t. They aim to overcome IRT’s strong assumptions (like unidimensionality) by using the function-approximating power of deep learning, while still maintaining an IRT-like interpretation of ability.

- **Reinforcement Learning (RL) for Adaptive Testing:** IRT traditionally selects items based on a myopic criterion (maximizing current information). But one can frame the entire test administration as a **sequential decision-making problem**, and that’s where reinforcement learning comes in. Researchers have proposed treating the adaptive test as an RL environment: the test algorithm is an agent that selects items (actions) to maximize a cumulative reward (e.g. measurement accuracy or information) over the test. A recent framework called **Neural Computerized Adaptive Testing (NCAT)** explicitly reformulates CAT as a reinforcement learning problem and trains a neural network to choose items optimally [33]. The RL agent can learn policies that, for instance, consider long-term gains (maybe sacrificing a tiny information gain now to avoid item exposure or to satisfy content balance later). Early studies (Zhang et al., 2023) show that deep RL item selection can perform comparably or better than static IRT information-based rules, especially when combined with constraints or multidimensional traits. Another use of RL is deciding **when to stop** the test – e.g. a deep RL model that learns to decide test length dynamically for each examinee, based on when additional questions no longer significantly improve precision [34]. These approaches are still emerging, but they represent a way to bring AI planning into testing, on top of the IRT measurement model.

- **LLM-based and Generative Approaches:** Large Language Models (LLMs) like GPT-4 are being explored in assessment in several ways:

- *Synthetic Data and Item Generation:* Collecting large response datasets for calibrating IRT models can be expensive. Researchers have begun using LLMs as **simulated examinees** – e.g. prompting an LLM to answer test items as if it were a student – to generate response data. Remarkably, studies found that item parameters (difficulty, etc.) estimated from **LLM-generated responses** can correlate highly with those from real student data [35]. This suggests LLM “respondents” might assist in pre-testing items or calibrating difficulties when real pilot data is scarce. On the flip side, LLMs can be used to *generate new test questions*. Given a target difficulty or skill, an LLM can potentially draft an item, which is then calibrated (perhaps initially by the model’s own prediction of solution or by a smaller pilot). One innovative approach jointly trained a **generator model alongside an IRT model**: the IRT part evaluates item quality (difficulty, discrimination) and the generator creates items to match desired parameters [30]. This kind of **generative IRT** pipeline could produce a supply of calibrated items on the fly, greatly accelerating test development.
- *Content Embeddings and Difficulty Prediction:* Instead of treating item parameters as purely latent, we can use AI to predict them from item content. For example, one can use NLP techniques to encode a question (and even its correct answer and distractors) into a vector embedding, and use a regression model (or an LLM) to predict the item’s difficulty **b** and perhaps discrimination **a** [36]. If accurate, this would allow test designers to get difficulty estimates before piloting, or to continuously calibrate items as content changes. Research in 2023 has combined manual linguistic features with LLM-derived features (like how an LLM reasons about the question) to estimate IRT difficulty parameters, often achieving reasonably high correlations with empirical difficulties [37].

- *Adaptive Learning and Tutoring:* Beyond formal tests, AI (especially LLMs) can enable *conversational, adaptive learning experiences*. For instance, an LLM-based tutor could, in real-time, adjust the difficulty of questions it asks a student by gauging the student's performance (implicitly implementing an IRT-like adaptive strategy). It might not explicitly use logistic models, but the principle is similar: estimate student skill from their responses and adapt accordingly. Moreover, AI can provide **adaptive hints, explanations, or rephrase questions** if it detects a student is struggling [38] – something traditional IRT CAT doesn't do (it just selects items). This merges measurement with instruction, an exciting direction in EdTech.

In summary, **AI extensions of IRT** aim to either improve the measurement model or the test delivery: - *Neural/Deep IRT* improves how we model responses and learn parameters (handling complex patterns, temporal change, multi-skills). - *Bayesian IRT* gives a principled probabilistic framework, often underlying the above (e.g. variational neural IRT is essentially Bayesian). - *RL-based CAT* improves the strategy of item selection and test control. - *LLM and generative methods* enhance item development, calibration, and even real-time testing in richer ways (e.g. generating new items, simulating examinees, providing interactive feedback).

All these still rely on the fundamental idea of a latent trait being measured by item responses, but they leverage modern computing to either relax assumptions or add capabilities. In practice, an AI engineer might use these extensions to build an "IRT 2.0" where, for example, an examinee's ability estimate is continuously updated not just from right/wrong answers, but also from response time, text responses scored by an NLP model, and other signals – a fusion of IRT with multimodal ML. This is an active research frontier.

Practical Examples in Large-Scale Exams

How is IRT applied in exams like GRE, PTE, GMAT? Virtually all major standardized tests today use IRT at some stage, especially those with adaptive formats. Let's look at a few:

- **Graduate Record Examination (GRE):** The GRE General Test (by ETS) is a high-stakes exam for graduate admissions. After 2011, the GRE adopted a **multi-stage adaptive design**. This means the GRE isn't item-by-item adaptive but section adaptive. Here's how it works: Each test-taker gets a first section of (say) Quantitative questions that is of average difficulty. After completing it, the computer uses an IRT-based scoring algorithm to estimate ability. The second section's difficulty (easy, medium, or hard) is then assigned based on that estimate. So a strong performer gets a harder second section (yielding a higher possible score ceiling), whereas a weaker performer gets an easier second section (focused on measuring basics). IRT comes in by calibrating all the questions beforehand and scoring the performance across sections on a common scale. This approach balances adaptivity with test fairness (all candidates face two sections, and within a section they can skip/review items which pure CAT wouldn't allow). The result is a shorter test than a non-adaptive approach, with score precision maintained. ETS reported that using IRT and adaptive design for the GRE improved the fairness and efficiency of the exam. In fact, IRT is crucial for **equating** different sections and forms – a GRE in October vs one in November can have different questions, but because of IRT calibration, scores remain comparable over time.

- **Pearson Test of English Academic (PTE-A):** PTE Academic is a computer-based English language proficiency test. It is fully adaptive within certain question types. Pearson uses IRT to calibrate each

item's difficulty and other parameters during test development. During an actual PTE exam, the computer continuously adjusts which question to present next based on the test-taker's performance (very much like a CAT). This ensures that each test-taker, whether lower proficiency or highly fluent, is presented with items appropriate to their level. The payoff is a more accurate measure of English ability in a shorter test time – the algorithm can zero in on your level by giving harder items if you sail through the easy ones, and vice versa. **PTE exemplifies IRT in action**: the scoring algorithm uses IRT to account for item difficulties, so a person who answers a series of very difficult items correctly will get a higher score than someone who answered the same number of easier items correctly [39]. Pearson publishes technical reports indicating that IRT-based adaptive scoring increases both accuracy and test security (everyone sees a different mix of items). This tailored approach "ensures each candidate faces questions appropriate to their skill level, resulting in a more precise measurement of their proficiency" [39]. In practice, when you take PTE, you might notice that if you do well on early questions, later ones seem quite challenging – that's the IRT-based engine challenging you to find your true level.

- **Graduate Management Admission Test (GMAT):** The GMAT (by GMAC) is a prime example of a **fully adaptive, item-level CAT**. From the start, GMAT has used IRT as the cornerstone of its scoring algorithm [24]. In the GMAT, each question you get influences the difficulty of the next one. The test starts with a medium difficulty question; if you answer correctly, the next question is harder, if wrong, the next is easier. Behind the scenes, after each question, the computer updates an ability estimate (θ). GMAT's algorithm uses a version of the 3PL IRT model to do this, taking into account the difficulty (b) of each question you got right or wrong [24]. By the end of the section, the algorithm has honed in on your ability such that roughly half of the questions you saw were below your ability and half above – meaning you got about half of them right (this is typical in well-targeted CAT). It's not the percentage correct that matters, but the difficulty of what you got right. For instance, GMAT explicitly notes that *which* questions you get right is more important than *how many* [40] [24]. A high ability examinee might only answer 60% of questions correctly, but because they were all very hard questions, they can still score in the top percentile. Conversely, an examinee who gets 90% correct but on easy items will score lower. The IRT scoring makes this possible: it predicts your *true ability* by considering item difficulties. As a concrete example, GMAT psychometricians illustrate that if two test-takers both miss 4 questions, but one of them missed relatively easy ones while the other missed only the very hardest ones, the second person will have a higher θ estimate [24] [41]. The GMAT uses this paradigm in Quant and Verbal sections. The end result is an efficiently administered exam (~31 Quant questions, ~36 Verbal) that achieves measurement precision equivalent to a much longer fixed test, thanks to IRT-driven adaptivity.

These examples show IRT's role: **calibrating item banks and driving adaptive algorithms**. In all cases, a large pool of items is pre-calibrated with IRT (often through extensive field testing with sample takers). When you sit for the exam: - The computer selects items (in real-time for PTE/GMAT, or module-wise for GRE) based on your running estimated θ. - Scoring at the end is typically reported not as θ itself, but converted to a scale (e.g. GMAT 200-800). Under the hood, that conversion is a monotonically increasing function of θ. - These tests also use IRT for **equating**: ensuring this year's scores are equivalent to last year's, even if some items changed, by anchoring item parameters on a common scale [42].

It's worth noting that beyond admissions tests, IRT is used in exams like the **TOEFL** (ETS), **ASVAB** (military entrance, adaptive), certification exams, and large-scale surveys like **PISA** and **NAEP** (though those aren't adaptive, they rely on IRT for analysis). Even when tests are not adaptive, IRT is often used to analyze item

quality and to score examinees via **ability estimates** rather than raw scores. For instance, NAEP (the U.S. national assessment) uses IRT to place students on proficiency scales and to link assessments across years.

Common Software & Libraries for IRT

What tools can an engineer use to model and simulate IRT? Working with IRT typically involves specialized software for calibrating items and scoring examinees. Fortunately, many options exist, including open-source libraries in Python and R, as well as dedicated commercial programs. Here are some of the common ones:

- **R Packages:** R has a rich ecosystem for IRT, which is well-established in the psychometrics community. Notable packages:
- `mirt` (Multidimensional Item Response Theory) – a very popular R package by Phil Chalmers. It can fit a wide range of IRT models: 1PL, 2PL, 3PL, graded response (for Likert scales), multidimensional IRT, bifactor models, etc., using maximum-likelihood EM algorithms. It's known for its flexibility and has good documentation, including a tutorial paper [43]. *mirt* even includes a GUI shiny app for interactive use.
- `ltm` – an older but accessible package for unidimensional IRT (logistic models and graded response). It's good for getting started with 1PL/2PL/3PL modeling and provides functions to factor-analyze item response data.
- `TAM` (Package for the **Test Analysis Modules**) – developed by Margareth Wu et al., TAM is powerful for large-scale assessment data (like PISA). It can do IRT with latent regression (i.e., incorporate demographic predictors of ability), multi-group IRT, and more [44]. ACER ConQuest is a commercial software in the same vein, and TAM essentially provides similar capabilities in R.

- **Other R tools:** `irtoys` (simpler toolkit), `eRm` (for Rasch models), `psych` (general psychometric functions, including IRT ability estimation), `catR` (simulating CAT administrations using IRT), and a CRAN Task View "Psychometrics" listing many such packages [45]. For Bayesian IRT in R, packages like `brms` (Bayesian multilevel modeling) or `rstanarm` can fit IRT models with MCMC under the hood [27].

- **Python Libraries:** Python historically had fewer IRT libraries, but this is changing:

- `py-irt` – A newer Python library that implements Bayesian IRT models via variational inference [28]. It currently supports 1PL, 2PL, and even 4PL, with the 3PL in development. Under the hood it uses PyTorch and Pyro (probabilistic programming) to fit models, which means it's quite scalable and can handle large datasets. One can use `py-irt` to estimate item parameters and abilities either from Python code or via a command-line interface. It's a great tool if you want to integrate IRT calibration into a modern ML pipeline (for example, calibrating model-generated responses as in some NLP tasks [46]).
- `irt` **(via 17zuoye/pyirt)** – There's a GitHub project "pyirt" (perhaps with overlapping name) that was designed by an online education company to handle sparse student response data. This might be similar or an earlier variant focused on big data from ed-tech platforms.
- **PyMC / PyStan / TensorFlow Probability** – Since IRT is fundamentally a statistical model, one can also leverage general Bayesian modeling libraries. For example, one can write a simple 2PL model in PyMC3 or PyMC4 and sample from the posterior. Or use TensorFlow Probability to set up an IRT

model as a probabilistic neural network. These require more work but are very flexible (useful for custom IRT variants).

- **scikit-learn?** – While scikit-learn doesn't have IRT built-in (since it's more of a psychometric than general ML model), some people have framed IRT as a one-dimensional logistic PCA or used scikit's logistic regression in creative ways. But generally, one would use specialized libraries or write custom code.

- **Commercial Software:** In testing organizations, you'll encounter tools like:

- **BILOG-MG, PARSCALE** – classic programs for binary and polytomous IRT (from Scientific Software / SSI). They are command-line tools popular in the 90s/00s.
- **IRTPRO** – a Windows software for IRT analysis (successor to BILOG, from SSI and later Scientific Software International).
- **Xcalibre** – a user-friendly commercial program for IRT from Assessment Systems (Nathan Thompson's company). It provides nice reports and graphics [47].
- **Winsteps** – for Rasch modeling (very popular among Rasch purists).
- **CatSim** – a tool to simulate computerized adaptive tests (mentioned by Assessment Systems).

While commercial software often has nice interfaces and is optimized, as an AI engineer you might stick to open-source unless dealing with very large secure item banks.

For learning and prototyping, R's packages are extremely handy due to their maturity. For instance, using `mirt` you can: - Calibrate a set of items (get estimates of a, b, c). - Plot ICCs (item characteristic curves) and IICs (information curves). - Score new respondents (estimate θ given their item responses). - Even simulate an adaptive test: `catR` package can select items given an item bank and simulate test-takers.

In Python, if you prefer that ecosystem, `py-irt` now gives similar capabilities with a Bayesian twist, and it's actively maintained (as of 2025, last updated August 2025) [48] [28].

Lastly, if your work will involve custom modeling, libraries like **PyTorch** or **JAX** can be used to build IRT models from scratch since an IRT model is basically a logistic regression on latent features (θ and item parameters). In fact, one can view 2PL IRT as equivalent to a one-layer neural network with person and item embeddings dot-product, if you fix one embedding as the ability and the other as discrimination. Some practitioners have leveraged this view to fit IRT using gradient descent (treating θ_i and b_j as free parameters to optimize a log-likelihood).

TL;DR: R has more out-of-the-box IRT packages (mirt, ltm, TAM, etc.), whereas Python now has *py-irt* for Bayesian IRT and one can always resort to probabilistic programming or deep learning libraries for custom implementations. Being familiar with at least one of these (say, running a simple analysis in `mirt` or `py-irt`) will help solidify your understanding of IRT practically.

Public Datasets for Practice

Are there open datasets to play with IRT or adaptive testing? Yes – while testing data can be sensitive, several datasets in education and psychometrics are publicly available. Here are a few notable ones that an AI engineer can use for experimentation:

- **ASSISTments (2009-2010 Skill Builder Dataset):** ASSISTments is an online learning platform that released an anonymized dataset of student question responses. The 2009-2010 dataset is a classic in education research, containing thousands of students solving math problems with right/wrong answers recorded (often across multiple attempts) [49]. This dataset has been widely used for knowledge tracing and IRT research. You can obtain it from archived websites or repositories (originally on a Google Site, now often on data hubs like Figshare). With this data, you could, for example, fit an IRT model to a set of algebra questions to estimate their difficulties and see how student ability estimates correlate with, say, their class performance. **Why it's useful:** it's real student interaction data and relatively large, allowing testing of IRT vs. deep models. Keep in mind this data might violate some IRT assumptions (because students learn as they go), but it's great for experiments in dynamic testing and for testing modern extensions like Deep-IRT.

- **EdNet (KAIST 2020):** A very large-scale dataset from a Korean online learning platform, containing millions of interactions (students answering questions, with timestamps, etc.). EdNet was released to foster research in knowledge tracing. It includes item IDs, whether the student answered correctly, and sometimes metadata about the items. With EdNet, one can try fitting IRT to subsets (it's huge, so you might use a portion) or evaluate how an IRT model performs versus more complex neural knowledge tracing on predicting responses. EdNet brings in the challenge of *big data* and potential multidimensionality (since it spans many skills), which is a good testbed for advanced IRT or AI models.

- **PISA Dataset:** The OECD's **Program for International Student Assessment (PISA)** releases datasets for each cycle (e.g. PISA 2018, PISA 2022) [50]. These include responses of tens of thousands of students from around the world to exam items in reading, math, and science. PISA uses IRT extensively to scale student proficiency. While not all actual test items are released (some are kept confidential for reuse), the datasets contain item-level response data for a subset of items and overall IRT-based proficiency estimates. You can use the released items to practice IRT modeling – for example, fit a 2PL to the math items to recover difficulty estimates and compare to published values. PISA data also let you explore differential item functioning (DIF) and other analyses (e.g., are certain items easier in some countries than others?). Do note, PISA uses a multiple matrix sampling design (each student only answers some items), so you might need to deal with missing data or use their conditioning approach. But it's a rich, real-world dataset for psychometrics.

- **Item Response Theory Examples (Educational Testing data):** Some textbooks or papers provide small example datasets, like sets of dichotomous item responses for a classroom test or a set of **LSAT questions** (Law School Admission Test) that are often used in IRT teaching. For instance, the R package *ltm* comes with a dataset of 5 binary-scored LSAT questions taken by 1000 examinees. This can be great for a quick hands-on: you can fit a 2PL and see the estimated difficulties and discriminations, or plot the ICCs. Similarly, there's a famous dataset of **25 GMAT questions** with known parameters used in some IRT literature. While small, these are useful for validating your IRT implementation.

- **Psychological Questionnaires Data:** The **Open Psychometrics** project [51] provides datasets from online personality quizzes (e.g., Big Five inventory, IQ test items, etc.). These are not educational tests, but they are amenable to IRT analysis (often using graded response models if Likert scale). For example, you could take a Big Five personality test dataset (100k responses to 50 items on a 1-5 agreement scale) and fit a graded IRT model to see how well the items discriminate introversion vs. extraversion, etc. This could be an interesting way to apply IRT outside of education, and the data is readily available.

- **Kaggle Datasets:** Occasionally, competitions or research yield datasets – e.g., the **DuoLingo Shared Task** released data on language learners answering exercises, including whether they got them right, time taken, etc. Another Kaggle dataset contains **2012-2013 ASSISTments** data (with some affect labels) [52]. Searching Kaggle for "education" or "adaptive testing" might yield new open datasets.

When using these datasets, be mindful of their structure: - Some are **wide format** (one row per student, columns as item scores), others **long format** (one row per student-item interaction). You'll often need to wrangle into the format needed by your IRT software. - Large datasets (EdNet, ASSISTments) may require subsampling or using batched optimization (which is where something like `py-irt` with variational inference shines, versus classic EM which might be slow on millions of records).

To get started, the ASSISTments 2009-10 data is a good choice (moderate size, well-known). PISA is good for seeing a polished application of IRT (with many demographics, etc.). And if you are venturing into knowledge tracing, ASSISTments or EdNet are essentially the go-to benchmarks – you can try to incorporate IRT by, say, periodically calibrating item difficulties and see if that improves prediction.

Next-Step Guidance for an AI Engineer

Where to go from here to integrate IRT with AI in adaptive systems? Now that you have a foundational grasp of IRT, you can start thinking about how to **apply and extend these concepts** in your AI-driven adaptive testing project. Here are some recommended next steps and considerations:

1. **Hands-on with IRT tools:** Implement a simple IRT analysis on a dataset. For instance, take a small set of Q&A data (maybe from an internal quiz or one of the open datasets) and use a library like `mirt` or `py-irt` to estimate item parameters and abilities. This will solidify your understanding. Try scoring an individual with IRT vs. classical percent-correct and see the difference. Also, if possible, simulate a short adaptive test with that data: write a script that picks an item, updates an ability estimate, etc., to mimic the CAT flow. This doesn't have to be complex (you could even do a few iterations manually to see the logic). This experience will make the adaptive algorithm more concrete.

2. **Learn the domain specifics:** Since you're at Pearson, find out specifics of the testing product you'll be working on. Is it a language test (like PTE) or an educational assessment? The domain can dictate extensions: e.g. language tests might involve sub-skills (listening, speaking) – consider if a **multidimensional IRT** model is relevant (ability vector instead of single θ). Or if it's a test with automated scoring of essays, how might IRT integrate with that (perhaps treating the essay score as

an “item” response with some measurement error). Understanding the test content and goals will guide whether you need, say, polytomous IRT models or if you can assume unidimensionality, etc.

3. **Combine IRT with ML for** item banking**:** One near-term application of AI is improving the item pool. IRT needs calibrated items – AI can help generate or analyze items quickly:

4. You could use NLP (maybe a transformer model) to predict item difficulty from its text, as mentioned. Perhaps experiment with an LLM by asking it to take some items and see if its success probability correlates with known difficulty (this could inspire a tool to pre-calibrate items before pilot testing).
5. Investigate **automated item generation**. For example, for a vocabulary test, could an LLM generate new fill-in-the-blank items at specific difficulty levels? If yes, you’d still use IRT to validate and fine-tune those items, but AI could drastically speed up item development.

6. Look into **content balancing algorithms**; Pearson’s adaptive tests often require content constraints (e.g., certain number of algebra vs geometry items). AI (like integer programming or even heuristic search with ML guidance) can assist in assembly of test forms or constrain item selection. This is adjacent to IRT – IRT gives the measurement model, but assembling a *blueprint* is an optimization task.

7. **Adaptive algorithm enhancement:** With your AI/ML background, consider where a smarter algorithm could improve over traditional methods:

8. Possibly implement an **adaptive strategy using reinforcement learning**. As discussed, an RL agent could learn an optimal policy that might juggle the trade-off between maximizing information and meeting content specs or minimizing test length. You could simulate examinees (perhaps using an IRT generative model) and train an RL algorithm to select items. Even if you don’t deploy an RL policy directly, the insights could inform tweaks to the item selection heuristic.
9. Another angle: **Bayesian adaptive testing** – using the posterior distribution of θ (instead of point estimate) to select items. This is already in psychometric literature (e.g., using Kullback-Leibler information or expected information). With AI, one could approximate these calculations faster or use particle filters for ability. Explore if a Bayesian approach is suitable (especially if tests are very short or if you want to incorporate prior info like previous test scores).

10. **Stopping rules via ML:** Perhaps use supervised learning to predict if an examinee’s score would change materially by asking more questions. Or train a model on simulations to decide an optimal stopping point for different profiles of response patterns.

11. **Explore** Neural IRT / Knowledge Tracing **further:** If your project involves not just one-off tests but continuous learning scenarios (e.g., a practice system that adapts to students over weeks), bridging IRT with **knowledge tracing** is key. You might want to read that 2019 Deep-IRT paper [31] or related works, which try to get the best of both worlds: interpretability of IRT and temporal dynamics of recurrent models. Implementing a simple knowledge tracing model (like Deep Knowledge Tracing, which is an RNN) and then seeing if you can overlay an IRT layer (maybe periodically extracting a θ) could be a cool R&D experiment. It might lead to an adaptive tutor that not only predicts performance but also diagnoses skill levels explicitly.

12. Additionally, look into **Cognitive Diagnostic Models (CDMs)** – these are alternatives to IRT that model multiple fine-grained skills per item (like Q-matrix models). Modern neural versions (Neural

CDM) exist [53]. Depending on your project, if the goal is more formative (tell students which specific skills to work on), CDMs or multitrait IRT might be relevant. Your AI skills can help scale CDMs with neural nets.

13. **Leverage LLMs carefully:** Large Language Models could become part of your adaptive system's backend. For example:

14. Use an LLM to **analyze student responses** to open-ended questions (scoring essays or short answers). This is already happening (automated scoring engines using ML). If doing so, consider using IRT to calibrate those scores (each prompt can be treated like an item with some rater error). There's research on using IRT to adjust for bias in automated scoring [54].
15. Perhaps incorporate an LLM to provide **feedback or hints**. IRT traditionally doesn't do that (since it's focused on measurement), but an AI-augmented system could. You'd need to ensure the measurement aspect (IRT θ) remains separate from any teaching intervention (to avoid skewing the estimate). One approach is the **Shadow Test** approach, where a separate set of items could be used for measurement while LLM-driven help is given on practice items.

16. As a creative experiment, you might use an LLM to simulate a range of **ability-specific personas** (e.g., prompt the LLM to behave like a student who is moderately skilled vs very skilled) and run your adaptive algorithm on them. This can reveal if the algorithm differentiates them properly. Some papers did similar by having GPT-3.5 and GPT-4 answer test items and treating them as examinees of varying "ability" [35]. If GPT-4 gets 90% right and GPT-3.5 gets 80%, we can treat them as high vs slightly lower ability and see if the adaptive test yields different θ estimates accordingly.

17. **Keep fairness and validity in mind:** As you integrate AI, remember the core principles of psychometrics:

18. **Validity:** Does your adaptive test actually measure what it should? If you use AI to generate items or provide hints, ensure the construct isn't compromised. You might have to run IRT analyses for bias/DIF if, say, an LLM's hints inadvertently help some groups more.
19. **Fairness and Bias:** AI can introduce new biases. For instance, an LLM might generate items that inadvertently are easier for native speakers due to phrasing. IRT can help detect biased items via techniques like Differential Item Functioning analysis. As you innovate, plan to use such analyses to vet the AI contributions.

20. **Security:** Adaptive tests face item exposure issues (people could memorize the pool). AI might help by generating items on the fly (making each test unique), but you'd need to verify those on-the-fly items are psychometrically calibrated. Perhaps use IRT to quickly calibrate new items using a **shadow-testing** method or seeding them in alongside known items to estimate their parameters.

21. **Further Reading and Networking:** To deepen your expertise:

22. Consider reading *"Item Response Theory for Psychologists" by Embretson & Reise (2000)* for a conceptual understanding (less math-heavy) [55]. Since you're already technical, also look at *de Ayala (2009)* for more math. For the bleeding edge, there's a 3-volume *Handbook of IRT* (van der Linden) [55] – you might not need all that now, but good to know of.

23. Look at ETS or Pearson's research papers on adaptive testing. ETS has research reports on how the GRE and TOEFL use IRT and AI. Pearson's technical reports on PTE Academic could be insightful (they likely discuss the AI scoring of speaking items combined with IRT scaling).
24. If possible, talk to a psychometrician in Pearson. They can provide insight into current practices and pain points (which might be exactly where AI can help). For example, they might say "calibrating new items takes months" – which could spark an AI solution to predict item parameters from limited data.

25. Follow conferences like **EDM (Educational Data Mining)**, **LAK (Learning Analytics)**, and **ICAI (Int. Conf. on Artificial Intelligence in Education)**. These often have papers on "deep knowledge tracing", "neural IRT", "RL in tutoring systems", etc. For instance, a 2023 paper in EDM might talk about an RL-based CAT algorithm, or an AAAI paper about NCAT (we saw references to an AAAI paper 2022 on Neural CAT [56]). Staying abreast of these will inspire approaches for your project.

26. **Prototype AI-enhanced adaptive system:** Finally, when you start designing your system, consider a modular approach:

27. The **IRT module**: handles ability estimation and item parameter storage.
28. The **AI module**: perhaps handles content generation or analysis (like an LLM generating items or scoring responses).
29. The **Adaptive engine**: the part that decides what to do next (could initially use simple IRT info rules, but you can swap in an RL policy later).
30. The **User interface and data collection**: ensure it captures responses with time stamps, etc., as this data will feed back into improving your models (for example, using response times with IRT to detect if items are speeded).

By keeping these separate, you can iteratively replace or improve each. For example, start with a static item bank and IRT-based CAT algorithm. Then perhaps pilot an AI-generated item as an experimental item (not scored) to gather data and see if its IRT parameters match expectation. Use that feedback to trust AI-generated items more. Or start with a simple IRT scoring, then later incorporate a **Neural Network to fine-tune ability estimates** using additional data (like pattern of option choices, not just correct/incorrect).

In essence, your path is to **marry the robustness of IRT with the flexibility of AI**: - IRT gives you a strong measurement foundation (ground truth of ability, item quality). - AI gives you tools to enhance item creation, selection, and overall user experience.

Your immediate next step could be as simple as: *take a small dataset, fit an IRT model, and simulate an adaptive test* to build intuition. From there, layer in complexity with AI components. By tomorrow, you should be able to explain to your manager how IRT works and discuss ideas like "we could use an LLM to generate items which we then calibrate with IRT" or "we might train a model to pick items in an adaptive way – possibly an RL agent – instead of the static method". Backed with the foundational knowledge from above, you'll be equipped to design the next generation of Pearson's adaptive testing system that is both **psychometrically sound** and **AI-powered** [57].

<small>*Side Note:* Don't forget the human element – AI can modernize IRT, but expert review of items and continuous validation are key in high-stakes testing. An AI engineer with psychometric awareness (like you're becoming) plus collaboration with assessment experts is a powerful combination. As you proceed,

keep communicating in both “languages”: the statistical rigor of IRT and the data-driven adaptability of AI. Good luck with your project!</small>

Sources:

1. Nathan Thompson (2024). *Item Response Theory (IRT): Better Assessment with Machine Learning* [1] [2] – Overview of IRT benefits (equating, adaptive testing) and why it’s needed over CTT.
2. Wikipedia – *Item Response Theory* [7] [17] – Definition of IRT as probability function of ability and item parameters; explanation of difficulty, discrimination, guessing.
3. Julie Wood (2017). *Logistic IRT Models Tutorial* [9] – Explanation of 1PL (Rasch) model and meaning of item difficulty (ability for 50% correct).
4. HackMD Tutorial – *Computerized Adaptive Testing with IRT* [21] [23] – Outline of CAT process (ability initialization, item selection, ability update, stopping rule) and item information.
5. Ranganath S. (2023). *Evolution and Role of Adaptive Tests* (e-assessment blog) [19] [22] – Describes how an IRT-based CAT works (starting item of average difficulty, iterative selection, stopping criteria).
6. Yeung (2019). *Deep-IRT: Deep Learning Knowledge Tracing with Item Response Theory* [31] – Abstract describing a model that uses deep networks to estimate ability and difficulty over time, then IRT to predict responses.
7. Zhang et al. (2022). *Neural Computerized Adaptive Testing (NCAT)* [33] – Redefining CAT as a reinforcement learning problem, using neural networks to learn item selection policies.
8. OpenAI/DeepMind (2023). *Generative IRT for LLM evaluation* [30] – Example of training a variational IRT model with an item generator, to create new test items dynamically.
9. PsicoSmart Blog (2024). *IRT in Modern Psychometric Tests* [39] – Mentions Pearson’s PTE using IRT for adaptive difficulty and ETS using IRT for GRE adaptive testing (improved fairness and precision).
10. Dominate Test Prep (2024). *GMAT Scoring Algorithm Explained* [24] [41] – Explains GMAT’s use of IRT (θ as a function of item difficulties and responses) and why which questions you get right matters more than how many.
11. Py-IRT Documentation (2025). *Bayesian IRT in Python* [28] – Notes on the `py-irt` library capabilities (1PL, 2PL, 4PL via variational inference).
12. Chalmers (2012). *mirt Package in R (JSS, v48(6))* [43] – Introduction to an R package for multidimensional IRT analysis.
13. OECD PISA Database (2018). *PISA dataset download page* [50] – Availability of item-level response data for international assessment (for practice with large-scale IRT).
14. ASSISTments 2009 Data (original site) [58] – Description of the open ASSISTments 2009-2010 skill builder dataset (student-item response records) for education data mining.

---

[1] [2] [5] [6] [8] [10] [15] [18] [42] [47] [55] Item Response Theory (IRT): Better assessment with ML | Assessment Systems (ASC)
https://assess.com/what-is-item-response-theory/

[3] [4] [7] [12] [17] [27] [44] Item response theory - Wikipedia
https://en.wikipedia.org/wiki/Item_response_theory

[9] [11] Logistic IRT Models
https://quantdev.ssri.psu.edu/sites/qdev/files/IRT_tutorial_FA17_2.html

13 14 16 20 21 23 45 Computerized Adaptive Testing (CAT) with Item Response Theory(IRT) models. - HackMD
https://hackmd.io/@VS6EsdvbThi2Y4QwMwcQUg/HJIBItnMu

19 22 25 26 The Evolution and Role of Adaptive Tests - The e-Assessment Association
https://www.e-assessment.com/news/the-evolution-and-role-of-adaptive-tests/

24 40 41 GMAT 705 Scoring Analysis | Understanding Adaptive Scoring on the GMAT
https://www.dominatetestprep.com/blog/gmat-705-how-many-questions-do-you-need-to-get-right

28 46 48 py-irt · PyPI
https://pypi.org/project/py-irt/

29 Investigating the Values of Large Language Models via Generative ...
https://arxiv.org/html/2406.14230

30 arXiv:2406.14230v2 [cs.CL] 12 Jul 2024 - OpenReview
https://openreview.net/pdf?id=aWTvEcCDqj

31 32 [1904.11738] Deep-IRT: Make Deep Learning Based Knowledge Tracing Explainable Using Item Response Theory
https://arxiv.org/abs/1904.11738

33 56 A hybrid model based on learning automata and cuckoo search for ...
https://pmc.ncbi.nlm.nih.gov/articles/PMC12104449/

34 Using Deep Reinforcement Learning to Decide Test Length - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC12049363/

35 Leveraging LLM respondents for item evaluation: A psychometric ...
https://bera-journals.onlinelibrary.wiley.com/doi/10.1111/bjet.13570?af=R

36 38 57 Assessing the Landscape of Adaptive Testing Modalities with AI | by Sam Bobo | Medium
https://medium.com/@sam.r.bobo/assessing-the-landscape-of-adaptive-learning-modalities-with-ai-04c470e9c4ba

37 LLM-Extracted Features for IRT Difficulty Parameter Estimation
https://purl.stanford.edu/hh553ky5049

39 What role does item response theory play in the development of modern psychometric tests?
https://blogs.psico-smart.com/blog-what-role-does-item-response-theory-play-in-the-development-of-modern-psychometric-tests-101089

43 [PDF] mirt: Multidimensional Item Response Theory - CRAN
https://cran.r-project.org/web/packages/mirt/mirt.pdf

49 [PDF] Item Response Ranking for Cognitive Diagnosis - IJCAI
https://www.ijcai.org/proceedings/2021/0241.pdf

50 PISA 2022 Database - OECD
https://www.oecd.org/en/data/datasets/pisa-2022-database.html

51 Open psychology data: Raw data from online personality tests
http://openpsychometrics.org/_rawdata/

52 ASSISTments Data Set 2012-2013 - Kaggle
https://www.kaggle.com/datasets/nicolaswattiez/skillbuilder-data-2009-2010

53 bigdata-ustc/EduCAT: Computerized Adaptive Testing - GitHub
https://github.com/bigdata-ustc/EduCAT

54 Robust Neural Automated Essay Scoring Using Item Response Theory
https://pmc.ncbi.nlm.nih.gov/articles/PMC7334153/

58 ASSISTmentsData - 2009-2010 ASSISTment Data - Google Sites
https://sites.google.com/site/assistmentsdata/home/2009-2010-assistment-data