A Unified Diagnostic Framework for GMAT Quantitative, Data Insights, and Verbal Sections

Yuchen Teng

June 3, 2025

Abstract

This paper presents a unified diagnostic framework for analyzing student performance in the quantitative (Q), data insights (DI), and verbal (V) sections of the GMAT. The framework employs a standardized chapter-based methodology to move beyond simple accuracy metrics, aiming to identify root causes of errors and inefficiencies. Core inputs include per-question data (time, correctness, type, difficulty, skill/domain), and overall test metrics. The analysis progresses through evaluating time strategy and data validity, conducting multidimensional performance analysis specific to each sectionś constructs, diagnosing error patterns (considering time, difficulty, and potential carelessness), applying coverage rules to detect widespread weaknesses, and generating personalized practice recommendations. The final output is a comprehensive diagnostic summary delivered in natural language, providing actionable insights and guidance for targeted student improvement. This unified structure ensures consistency in the analysis while accommodating the unique characteristics of each GMAT section.

1 Introduction

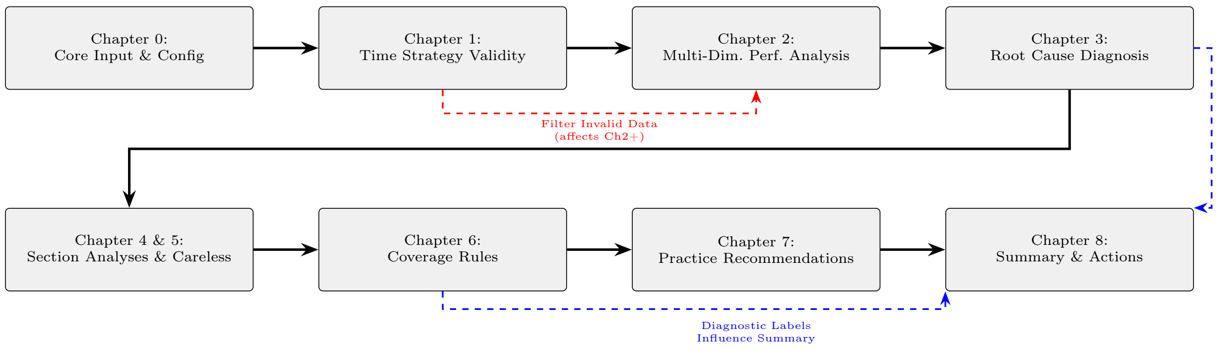
The Graduate Management Admission Test (GMAT) assesses critical reasoning, quantitative, data analysis, and verbal skills essential for graduate business programs. Effective preparation requires not only content mastery, but also strategic test-taking skills and an understanding of individual strengths and weaknesses. While numerous resources exist for practice, a systematic and standardized approach to diagnosing performance across all scored sections (Quantitative, Data Insights, Verbal) can significantly enhance study efficiency.

This paper introduces a unified diagnostic framework designed to provide in-depth analysis of student performance on the GMAT. Unlike simple score reports, this framework delves into the underlying reasons for errors and time inefficiencies, considering factors such as time pressure, question type, difficulty level, specific skills or content domains, and behavioral patterns.

The framework follows a consistent nine-chapter structure for analyzing each section (Q, DI, V), ensuring a comparable depth of insight while adapting the specific metrics and logic to the nuances of each section. This structure facilitates the following.

1. **Standardized Input:** Defining core data requirements (Chapter 0).
2. **Time & Validity Assessment:** Evaluating Pacing and Filtering Unreliable Data (Chapter 1).
3. **Multi-Dimensional Performance Analysis:** Examining performance across relevant section-specific dimensions (Chapter 2).
4. **Root Cause Diagnosis:** Classifying errors and exploring the underlying causes (Chapter 3).
5. **Section-Specific Analyses:** Investigating efficiency and patterns (Chapters 4 & 5, adapted per section).
6. **Coverage Assessment:** Identifying pervasive skill/type weaknesses (Chapter 6).

7. **Personalized Recommendations:** Generating actionable practice plans (Chapter 7).
8. **Synthesized Reporting:** Delivering a comprehensive, natural language summary (Chapter 8).



Chapter 0: Core Input & Config
Chapter 1: Time Strategy Validity
Chapter 2: Multi-Dim. Perf. Analysis
Chapter 3: Root Cause Diagnosis
Filter Invalid Data (affects Ch2+)
Chapter 4 & 5: Section Analyses & Careless
Chapter 6: Coverage Rules
Chapter 7: Practice Recommendations
Chapter 8: Summary & Actions
Diagnostic Labels Influence Summary

Figure 1: Overall Flow of the Unified Diagnostic Framework

By applying this unified framework, students and instructors can gain a holistic understanding of performance, identify specific areas requiring intervention, and develop targeted strategies for improvement throughout the GMAT exam. The subsequent sections detail the methodology applied within each chapter of this framework.

Note on Document Purpose: This document serves primarily as a technical report detailing the design, logic, and implementation rationale behind the unified GMAT diagnostic framework. It is intended as comprehensive documentation for users of the associated analysis tools (e.g., on GitHub) and as a methodological reference outlining the systematic approach developed. While initial parameters are informed by practical experience and preliminary testing, the core contribution presented here is the formalization of the diagnostic process itself.

Although experienced GMAT instructors often provide valuable information based on score reports, traditional analysis can suffer from subjectivity and inconsistency. Relying solely on intuition or anecdotal patterns introduces variability and hinders iterative refinement. Recognizing these limitations, the main motivation behind developing this unified diagnostic framework was to establish a more scientific, systematic, and objective approach. The goal is to enhance the efficiency, accuracy, and reliability of GMAT performance diagnosis by replacing purely experiential interpretation with a structured, parameterized, and verifiable methodology.

2 Methodology: The Diagnostic Framework

2.1 Framework Development Methodology

The development of this unified diagnostic framework originated from the practical need to structure insights gained through analyzing student GMAT performance, combining quantitative score report data with qualitative follow-up. Initial quantitative analysis relied on custom logic applied to performance metrics, while qualitative assessment, such as comparing student approaches to specific problems, aimed to uncover cognitive obstacles not evident in the raw data.

To establish a replicable process, consultations were transcribed and analyzed, utilizing AI assistance to consolidate the underlying analytical logic and structure. A key phase involved the explicit formalization of the diagnostic workflow, transforming subjective or ambiguous rules (e.g., "significant time pressure") into precise, parameterized definitions and functions. The outputs of this logic were designed as computable diagnostic tags, facilitating the generation of recommendations or guiding further qualitative inquiry.

Validation proceeded iteratively by comparing the framework's automated analysis against independent expert evaluations on real data. Discrepancies informed refinements to the logic, parameters, and functions, ensuring closer alignment with established diagnostic practices. The core framework was subsequently implemented as an automated tool using Python, a process involving learning logical control flow, modular design, and debugging techniques.

Throughout development, considerations for end-users—both students seeking intuitive interaction and instructors requiring detailed, traceable diagnostics—influenced design choices, leading to concepts like AI-assisted interfaces and standardized output tagging. This process highlighted that while AI can assist, the core analytical logic must be soundly defined by the expert. It also emphasized the need for maintainable code structure when dealing with AI-generated or complex components.

This narrative provides context for the detailed methodology presented in the subsequent chapters.

2.2 Chapter 0: Core Input Data and Configuration

Objective: Define the foundational data structures, parameters, and pre-processing steps required for analysis.

Required Input CSV Structure: For the analysis scripts to function correctly, the input CSV file (e.g., `testset-q.csv`) must contain specific columns with data in the expected format. While column names can sometimes be adapted during implementation, the following represents the ideal structure and data types:

- `question_position`: Sequence number of the question in the test (Integer, 1-indexed, serves as unique identifier).
- `question_time`: Response time for the question in minutes (Numeric, Float).
- `is_correct`: Indicator of correctness (Boolean: `True`/`False`).
- `content_domain`: Classification as 'Math Related' or 'Non-Math Related' (String, DI only).
- `question_type`: Category of the question (String). Specific values depend on the section:
  - Q: 'Real' or 'Pure'.
  - DI: 'DS', 'TPA', 'MSR', or 'GT'.
  - V: 'CR' or 'RC'.
- `msr_group_id`: MSR group identifier (Required only for MSR questions in DI section).
- `question_difficulty`: Numeric difficulty value (Numeric). The source might differ (e.g., `DI_b`, `V_b`), but it's mapped internally.
- `question_fundamental_skill`: Core skill or domain tested (String, Required for Q and V, e.g., 'Rates/Ratio/Percent', 'Identify Stated Idea'). **Note: Fundamental Skills are NOT tracked for DI.**

Overall Metrics (Derived or Input): While not always direct columns, the analysis requires:

- `total_test_time`: Total minutes spent (45 minutes standard).
- `max_allowed_time`: Standard 45 minutes per section.
- `total_number_of_questions`: 20 questions for DI, varies for Q/V.

Section-Specific Inputs & Pre-processing:

- **Quant (Q):**
  - Inputs: `question_type` categorized as 'Real' (word problems) or 'Pure' (computation/concept); `question_fundamental_skill` representing core mathematical areas, e.g.:
    - `Rates/Ratio/Percent`
    - `Value/Order/Factor`

    - `Equal/Unequal/ALG`
    - `Counting/Sets/Series/Prob/Stats`
  - Pre-processing: Calculate `average_time_per_type`, `max_correct_difficulty_per_skill`. Numerical conversions and handling of missing values are performed.

- **Data Insights (DI):**
  - **Basic Settings:** `Max Allowed Time`: 45 minutes; `Total Number of Questions`: 20.
  - **Question Type Abbreviations:** `DS`: Data Sufficiency; `TPA`: Two-Part Analysis; `MSR`: Multi-Source Reasoning; `GT`: Graph & Table.
  - Inputs: `question_type` ('DS', 'TPA', 'MSR', 'GT'); `content_domain` ('Math Related'/'Non-Math Related'); `msr_group_id` (required if `question_type` is 'MSR'). **Note: Fundamental Skills are typically NOT tracked for DI.**
  - **DI Derived Data Pre-processing:**
    - **MSR Reading Time (`msr_reading_time`):** For each MSR group, calculate reading time as: first question's `question_time` minus average `question_time` of all other questions in the group. This calculation is only valid when the group contains at least two questions, and the result is attached to the first question of the group. (This data will be used in Chapter 1 for MSR single question overtime judgment.)
    - **MSR Group Data Pre-calculation:**
      - `group_total_time`: Sum of `question_time` for all questions in each MSR group.
      - `num_q_in_group`: Number of questions in each MSR group.
    - **GT Time Allocation:** No special distinction between graph viewing and answering time.
    - **Average Time per Type (`average_time_per_type`):** Calculate average `question_time` for each `question_type` ('DS', 'TPA', 'MSR', 'GT') based on filtered valid data. (This data will be used in Chapter 3 for defining `is_relatively_fast` and Chapter 4 for carelessness rate calculation.)
    - **First Third Average Time per Type (`first_third_average_time_per_type`):** For each `question_type` ('DS', 'TPA', 'MSR', 'GT'), calculate average `question_time` for all questions where `question_position` $\leq$ (`Total Number of Questions` / 3). (This data will be used in Chapter 1 for invalid data judgment.)
    - **Max Mastered Difficulty per Combination (`max_correct_difficulty_per_combination`):** For each `question_type` and `content_domain` combination, record the highest `question_difficulty` value among all questions where `is_correct` == `True`.

- **Verbal (V):**
  - Inputs: `question_type` ('CR'/'RC'); `question_fundamental_skill`, e.g.:
    - `Plan/Construct`
    - `Identify Stated Idea`
    - `Identify Inferred Idea`
    - `Analysis/Critique`
  - Pre-processing: Identify `RC` passage groups (consecutive RC questions, usually 3-4). Calculate the following:
    - `rc_group_id`
    - `questions_in_group`
    - `group_total_time`
    - `average_time_per_type`
    - `first_third_average_time_per_type`

    Estimate `rc_reading_time` using a similar logic to MSR: `reading_time = time_q1 - average_time_of_other_qs_in_group`.

Implementation Context: Data ingestion typically involves reading CSV files using libraries like `pandas`. Pre-processing includes data type conversion (e.g., `pd.to_numeric`), handling missing data (`dropna`), mapping raw inputs (e.g., full question type names to abbreviations like 'DS'), renaming columns for consistency, and computing the derived metrics mentioned above. Configuration parameters (thresholds, factors) are defined constants within the implementation.

Rationale: Establishes a consistent, clean, and enriched data foundation crucial for reliable and comparable diagnostics across sections, with particular attention to DI-specific requirements such as MSR group processing and content domain classification.

2.3 Chapter 1: Overall Time Strategy and Data Validity Assessment

Objective: Evaluate overall pacing, assess time pressure, establish section-appropriate overtime criteria, and identify potentially invalid data points resulting from rushed end-section performance.

Operational Logic:

1. **Calculate Time Difference (`time_diff`):** Computed as `max_allowed_time - total_test_time`.
2. **Determine Time Pressure Status (`time_pressure`):** This Boolean flag indicates potential time pressure during the test section.
    - Default: `time_pressure` = `False`.
    - Check end-section conditions: Find questions in the last 1/3 of the test (`last_third_questions`) where `question_time` < 1.0 minute (`fast_end_questions`).
    - Judgment Logic: If `time_diff` $\leq$ 3 minutes **AND** `fast_end_questions` is not empty, then set `time_pressure` = `True`.
    - **User Override:** If user explicitly indicates low pressure, force `time_pressure` = `False`.
3. **Establish Overtime Thresholds/Rules (Based on `time_pressure`):** These criteria are dynamically set based on the `time_pressure` status and vary by section and question type.
    - **Q:** A single `overtime_threshold` (e.g., 2.5 min if `time_pressure`, 3.0 min otherwise).
    - **DI:** Type-specific thresholds and MSR group rules:
        - **TPA Overtime Threshold (`overtime_threshold_tpa`):** 3.0 min if `time_pressure` == `True`; 3.5 min if `time_pressure` == `False`.
        - **GT Overtime Threshold (`overtime_threshold_gt`):** 3.0 min if `time_pressure` == `True`; 3.5 min if `time_pressure` == `False`.
        - **DS Overtime Threshold (`overtime_threshold_ds`):** 2.0 min if `time_pressure` == `True`; 2.5 min if `time_pressure` == `False`.
        - **MSR Group Target Time Rules (`msr_group_target_time`):** 6.0 min if `time_pressure` == `True`; 7.0 min if `time_pressure` == `False`.
        - **MSR Single Question Analysis Thresholds:**
            * `msr_reading_threshold`: 1.5 minutes (for judging if reading time is excessive).
            * `msr_single_q_threshold`: 1.5 minutes (for judging if single question answering time is excessive).
    - **V:** Separate thresholds/rules for `CR` (`overtime_threshold_cr`: 2.0 min if `time_pressure`, 2.5 otherwise) and `RC` (e.g., `rc_group_target_time` for 3Q group: 6.0 min if `time_pressure`, 7.0 otherwise; for 4Q group: 8.0 min if `time_pressure`, 9.0 otherwise; `rc_individual_q_threshold`=2.0 min). V also includes a preliminary check for `reading_comprehension_barrier_inquiry` based on estimated `rc_reading_time` exceeding thresholds (e.g., >2.0 min for 3Q group, >2.5 min for 4Q group).
4. **Identify Invalid Data (`is_invalid`):** This flag identifies questions likely answered without genuine effort due to time constraints.

- **Trigger Condition:** Only executed if `time_pressure` == `True`. If `time_pressure` == `False`, no questions are marked as `is_invalid`.
- **Scope:** Only checks questions in the final third (`last_third_questions`).
- **Definition of "Abnormally Fast Response" (`abnormally_fast_response`) Standards (meeting any one triggers):**
  - *Standard 1 (Suspected abandonment):* `question_time` < 0.5 minutes.
  - *Standard 2 (Absolute rush):* `question_time` < 1.0 minute.
  - *Standard 3 (Relative single question rush - DS):* `question_time` < (`first_third_average_time_per_type`['DS'] * 0.5).
  - *Standard 4 (Relative single question rush - TPA):* `question_time` < (`first_third_average_time_per_type`['TPA'] * 0.5).
  - *Standard 5 (Relative single question rush - GT):* `question_time` < (`first_third_average_time_per_type`['GT'] * 0.5).
  - *Standard 6 (Relative group rush - MSR):* The MSR group containing the question has `group_total_time` < (`first_third_average_time_per_type`['MSR'] * `num_q_in_group` * 0.5).
- **Marking Logic:** Within the `last_third_questions` scope, if a question (or its MSR group for Standard 6) meets **at least one** `abnormally_fast_response` standard, **and** `time_pressure` == `True`, then mark that question as `is_invalid` = `True`.

5. **Global Rule - Data Filtering and Overtime Flagging:**

- **Mark Overtime (`overtime`):** Based on the thresholds/rules defined in this chapter, mark `overtime` = `True` for all questions **not marked as `is_invalid`**.
- **DI-Specific MSR Overtime Marking (Four-tier Standards):**
  - **a. Group Overtime Marking (`msr_group_over`):** Calculate total time (`msr_group_total_time`) for each MSR group. If `msr_group_total_time` > `msr_group_target_time`, mark **all** questions in that MSR group as `msr_group_over` = `True`.
  - **b. Reading Time Overtime Marking (`msr_reading_over`):** For MSR questions that are **not marked by `msr_group_over`** and are **first questions in their group**: if `msr_reading_time` > `msr_reading_threshold` (1.5 minutes), mark `msr_reading_over` = `True`.
  - **c. Adjusted First Question Overtime Marking (`msr_adj_first_over`):** For MSR questions that are **not marked by the above two conditions** and are **first questions in their group**: calculate `adjusted_time` = `question_time` - `msr_reading_time`. If `adjusted_time` > `msr_single_q_threshold` (1.5 minutes), mark `msr_adj_first_over` = `True`.
  - **d. Non-First Question Overtime Marking (`msr_non_first_over`):** For MSR questions that are **not marked by `msr_group_over`** and are **not first questions in their group**: if `question_time` > `msr_single_q_threshold` (1.5 minutes), mark `msr_non_first_over` = `True`.
  - **e. Final `overtime` Marking:** An MSR question is finally marked as `overtime` = `True` if and only if it meets any of these conditions: `msr_group_over` OR `msr_reading_over` OR `msr_adj_first_over` OR `msr_non_first_over`.
- **Create Filtered Dataset:** Remove all questions marked as `is_invalid` = `True` from the original question data.
- **Subsequent Analysis Scope:** All analysis in Chapters 2-6 is based exclusively on this filtered dataset.

Implementation Context: Boolean flags (`time_pressure`, `is_invalid`, `overtime`) are added as columns to the `pandas` DataFrame. Thresholds are assigned to variables based on conditional logic applied to the `time_pressure` status. Filtering is achieved via DataFrame slicing (e.g., `df_filtered =`

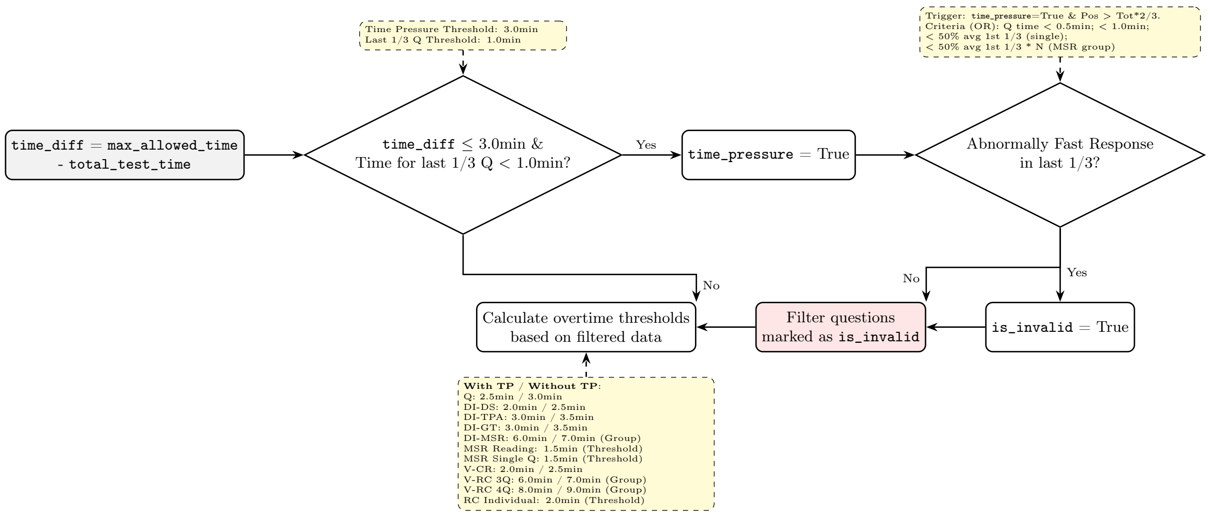

Time Pressure Threshold: 3.0min
Last 1/3 Q Threshold: 1.0min
Trigger: time_pressure=True & Pos > Tot*2/3.
Criteria (OR): Q time < 0.5min; < 1.0min;
< 50% avg 1st 1/3 (single);
< 50% avg 1st 1/3 * N (MSR group)
time_diff = max_allowed_time - total_test_time
time_diff ≤ 3.0min & Time for last 1/3 Q < 1.0min?
Yes
time_pressure = True
Abnormally Fast Response in last 1/3?
Yes
No
No
Calculate overtime thresholds based on filtered data
Filter questions marked as is_invalid
is_invalid = True
With TP / Without TP:
Q: 2.5min / 3.0min
DI-DS: 2.0min / 2.5min
DI-TPA: 3.0min / 3.5min
DI-GT: 3.0min / 3.5min
DI-MSR: 6.0min / 7.0min (Group)
MSR Reading: 1.5min (Threshold)
MSR Single Q: 1.5min (Threshold)
V-CR: 2.0min / 2.5min
V-RC 3Q: 6.0min / 7.0min (Group)
V-RC 4Q: 8.0min / 9.0min (Group)
RC Individual: 2.0min (Threshold)

Figure 2: Flowchart for Chapter 1 Logic: Time Pressure, Data Validity, and Overtime Thresholds

`df[~df['is_invalid']]`). For DI section, special handling is required for MSR questions with the four-tier overtime marking system.

Rationale: Ensures analyses are based on reliable data reflecting genuine effort and capability, while setting context-aware standards for time efficiency. Explicitly filtering invalid data before overtime flagging prevents misinterpreting rushed guesses as slow performance. The sophisticated MSR overtime detection system accounts for the unique structure and timing requirements of Multi-Source Reasoning questions.

2.4 Chapter 2: Multi-Dimensional Performance Analysis

Objective: Analyze performance accuracy and efficiency across key dimensions relevant to each section, using the **filtered dataset**.

Operational Logic:

- **Difficulty Level Standardization:** Raw numeric `question_difficulty` values are mapped to standardized categorical bands (e.g., "Low / 505+", "Medium / 605+", "High / 705+") using a consistent mapping function across sections. This facilitates comparison and interpretation.
- **Performance Metric Calculation:** For each relevant dimension or combination, calculate key metrics using the filtered data: total count, number of errors (`is_correct==False`), number of overtime instances (`overtime==True`), error rate, and overtime rate. Division by zero is handled gracefully (e.g., returning 0.0 or NaN).

Section-Specific Analysis Dimensions & Metrics:

- **Quant (Q):**
  - The primary dimensions analyzed for Quant section are: `question_type` ('Real'/'Pure'), `question_fundamental_skill`, and `difficulty_label`.
  - Analysis: Compares error rates and overtime rates between 'Real' and 'Pure'. Identifies significant differences based on absolute count difference (≥ 2) in errors or overtime instances, setting flags (`poor_real`, `slow_pure`, etc.).
- **Data Insights (DI):**
  - Dimensions: `content_domain` ('Math Related'/'Non-Math Related'), `question_type` ('DS', 'TPA', 'MSR', 'GT'), `difficulty_label`.

  - Analysis: Calculates error and overtime rates per dimension/combination. Identifies significant differences between `content_domains` based on absolute count difference (≥ 2), setting flags (`poor_math_related`, `slow_non_math_related`, etc.).

- **Verbal (V):**
  - Dimensions: `question_fundamental_skill`, `difficulty_label`, `question_type` ('CR'/'RC').
  - Analysis: Calculates error rates per dimension/combination. Identifies difficulty ranges and skills with the highest error concentration.

Implementation Context: `pandas groupby()` operations combined with `agg()` or `size().unstack()` are used extensively to compute metrics across dimensions. Error and overtime rates are calculated element-wise, often requiring intermediate steps to handle potential zero denominators (e.g., replacing 0 with `np.nan` before division). Significance flags are set using conditional logic based on the calculated metric differences.

Rationale: Pinpoint specific areas (types, skills, domains, difficulties) where accuracy or efficiency challenges exist within the valid performance data, providing focus for root cause diagnosis and practice planning.

2.5 Chapter 3: Root Cause Diagnosis and Analysis

Objective: Investigate the underlying reasons (ẅhy") for errors and inefficiencies identified in previous chapters, utilizing the **filtered dataset**.

Core Concepts & Operational Framework:

1. **Time Performance Classification:** Categorizes each valid questionś time relative to the average for its type. Required metrics (`average_time_per_type`) are pre-calculated from the filtered data.
   - `is_relatively_fast`: `question_time` < (`average_time_per_type` * 0.75). The 0.75 factor is a configurable parameter.
   - `is_slow`: Determined by the `overtime` flag (or its variants) applied in Chapter 1 based on the filtered data.
   - `is_normal_time`: Default state if neither `is_relatively_fast` nor `is_slow`.
2. **Special Focus Error (`special_focus_error`, SFE):** This critical flag identifies instability in foundational knowledge.
   - Definition: An error is classified as SFE under the following condition: `is_correct` == False AND `question_difficulty` < `max_mastered_difficulty`.
   - `max_mastered_difficulty` is calculated per `question_fundamental_skill` (for Q/V) or per `question_type/content_domain` combination (for DI section) by finding the maximum `question_difficulty` among correctly answered questions within that category in the filtered data.
   - Handling: SFE-flagged questions receive priority in subsequent reporting and recommendation generation.
3. **Diagnostic Label Assignment:** Based on the time performance classification and SFE status, each question is assigned diagnostic labels. These labels encapsulate the primary findings regarding potential root causes (e.g., S̈low & Wrong,̈ F̈ast & Wrong,̈ S̈FE)̈. These labels, rather than complex scenarios, serve as the core input for generating targeted recommendations in Chapter 7 and the narrative summary in Chapter 8. The detailed follow-up actions (student recall, evidence review, qualitative analysis) are presented as guidance in Chapter 8.

Implementation Context: The logic is implemented within the main analysis loop iterating through the filtered DataFrame. Time classification and SFE flags (serving as primary diagnostic labels) are computed for each question. These labels are stored, often alongside brief descriptive notes derived from the

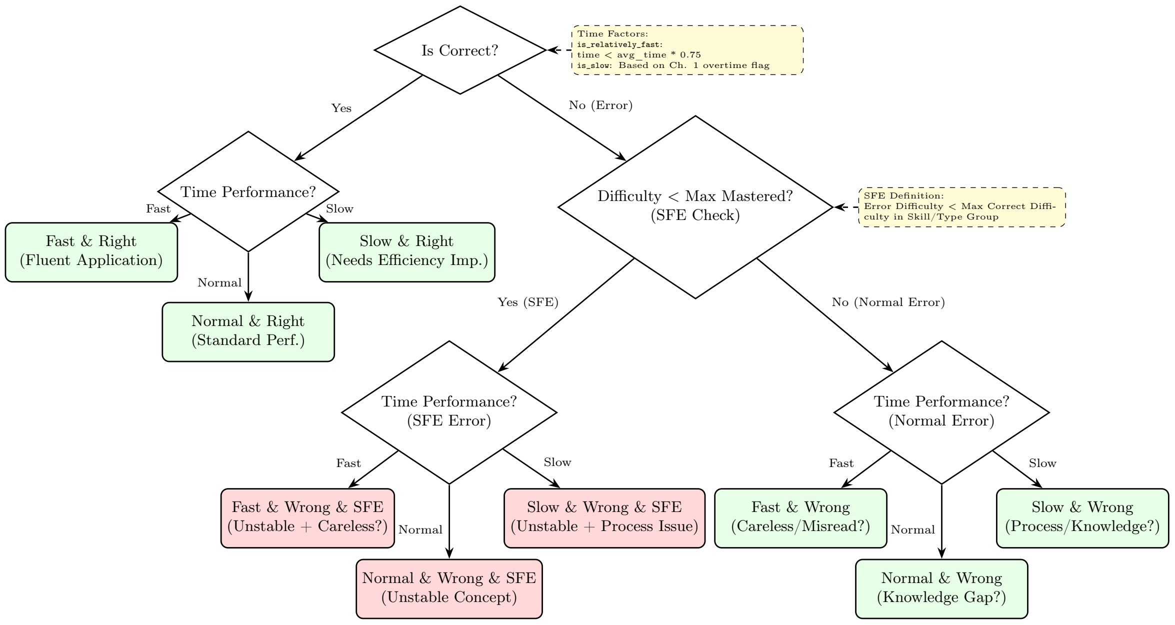

Is Correct?
Time Factors:
is_relatively_fast:
time < avg_time * 0.75
is_slow: Based on Ch. 1 overtime flag
Yes
No (Error)
Time Performance?
Fast
Slow
Normal
Fast & Right (Fluent Application)
Slow & Right (Needs Efficiency Imp.)
Normal & Right (Standard Perf.)
Difficulty < Max Mastered? (SFE Check)
SFE Definition:
Error Difficulty < Max Correct Difficulty in Skill/Type Group
Yes (SFE)
No (Normal Error)
Time Performance? (SFE Error)
Fast
Slow
Normal
Fast & Wrong & SFE (Unstable + Careless?)
Slow & Wrong & SFE (Unstable + Process Issue)
Normal & Wrong & SFE (Unstable Concept)
Time Performance? (Normal Error)
Fast
Slow
Normal
Fast & Wrong (Careless/Misread?)
Slow & Wrong (Process/Knowledge?)
Normal & Wrong (Knowledge Gap?)

Figure 3: Flowchart for Chapter 3 Logic: Root Cause Diagnosis Labels (Diagonal Lines)

combination (e.g., potential cause hypotheses like "calculation barrier" for DI Math Slow & Wrong), for use in generating detailed diagnostic text and recommendations in Chapters 7 and 8.

Rationale: Moves beyond identifying *what* is wrong to systematically classifying *why* using standardized diagnostic labels, enabling the development of targeted and effective improvement strategies by providing structured inputs for subsequent planning and reporting.

2.6 Chapters 4 & 5: Section-Specific Analyses, Patterns, and Carelessness

Objective: Conduct further analyses tailored to section characteristics, examine efficiency in correct answers, and assess behavioral patterns like carelessness, using the **filtered dataset**.

Synthesized Analyses & Operationalization:

- **Analysis of Correct but Overtime Questions:**
  - Identification: Filter questions where `is_correct` == True AND `is_slow` (or equivalent overtime flag) == True.
  - Data Recorded: `question_id`, `question_type`, `question_fundamental_skill` (if applicable), `question_time`.
  - Purpose: These instances trigger recommendations in Chapter 7 aimed at improving fluency and efficiency.
- **Special Pattern Observation (Early-Stage Rapid Responses):**
  - Identification: We check for questions meeting these criteria: Located in the first third (`question_position` $\leq$ `total_number_of_questions` / 3) AND `question_time` $<$ 1.0 minute (absolute threshold).
  - Reporting: If found, an alert regarding potential pacing issues or `flag for review` risks is included in the Chapter 8 summary.
- **Carelessness Assessment (`carelessness_issue`):**

  - Calculation: Determine `fast_wrong_rate` = (Count of `is_relatively_fast` AND `is_correct` == False) / (Total count of `is_relatively_fast`). Requires prior calculation of the `is_relatively_fast` flag from Chapter 3. Handle division by zero if no questions were relatively fast.
  - Flagging: If `fast_wrong_rate` > 0.25 (configurable threshold), set `carelessness_issue` = `True`.
  - Reporting: If flagged, the Chapter 8 summary includes a note about potential carelessness.

- **DI-Specific Pattern Observation:** For the Data Insights section, specific patterns like Multi-Source Reasoning (MSR) time distribution (e.g., comparing reading time vs. question-solving time) are also examined to identify potential inefficiencies within complex item types.
- **Core Skill/Type Reference (Context for Verbal):** The detailed breakdown of `CR` and `RC` subtypes, originally presented as a separate chapter in the Verbal source document, serves as reference material for interpreting Chapter 3 diagnoses and formulating Chapter 7 recommendations for the Verbal section. It is not an active analytical step itself but provides necessary classification context.

Implementation Context: These analyses are typically performed after the main Chapter 3 loop. Identification involves conditional filtering of the DataFrame. Rate calculations use simple aggregation and division. Flags are stored for use in Chapter 8 report generation.

Rationale: These chapters refine the diagnosis by examining efficiency patterns even in correct responses, identifying potential test-taking habits (pacing, carelessness), and providing necessary contextual classification (for Verbal), thereby adding further layers to the performance understanding.

2.7 Chapter 6: Fundamental Ability / Skill / Type Coverage Rules

Objective: Determine if pervasive weakness exists across an entire `fundamental_skill` (Q/V) or `question_type` (DI), warranting foundational reinforcement rather than solely addressing individual errors, using the **filtered dataset**.

Operational Logic:

1. **Calculate Performance Rates:** For each `fundamental_skill` (Q/V) or `question_type` (DI), compute the overall `error_rate` and `overtime_rate` using the filtered data.
2. **Trigger Override:** A coverage rule is triggered if, for a given skill/type, `error_rate` $>$ 0.5 **OR** `overtime_rate` $>$ 0.5. The 0.5 (50%) threshold is a configurable parameter. Set `skill_override_triggered`[*Skill*] (Q/V) or `override_triggered`[*Type*] (DI) = `True`.
3. **Determine Macroscopic Parameters (If Triggered):** These parameters guide the foundational practice recommendations.
   - **Macroscopic Difficulty (`Y_agg`):** Identify the minimum `question_difficulty` among all error or overtime questions within the triggered skill/type. Map this minimum difficulty to the standardized 6-level label (e.g., "Low / 505+", "Medium / 605+").
   - **Macroscopic Time Limit (`Z_agg`):**
     - Q: Fixed at 2.5 minutes.
     - DI: Based on the maximum `question_time` observed within the triggered type, rounded down to the nearest 0.5 minutes (`floor(max_time_triggering * 2) / 2`).
     - V: Standard target times for the respective skill/type are used (e.g., CR 2.0 min, RC 1.5 min).

Implementation Context: Performance rates are calculated using `groupby()` on the filtered DataFrame. The override trigger logic involves conditional checks on these rates. If triggered, `Y_agg` is determined by finding the minimum difficulty in the relevant subset and applying the standard mapping function; `Z_agg` is calculated based on the rules above. Results are typically stored in dictionaries mapping skills/types to their override status and associated `Y_agg`/`Z_agg` values.

Rationale: Acts as a crucial gating mechanism for practice planning. If a fundamental area shows systemic weakness (high error/overtime rate), the framework prioritizes broad, foundational practice (macroscopic recommendation) over potentially numerous, less effective fixes for individual symptoms (microscopic recommendations).

2.8 Chapter 7: Practice Planning and Recommendations

Objective: Translate all diagnostic findings from the filtered data analysis into a specific, actionable, and personalized practice plan.

Operational Logic:

1. **Identify Recommendation Triggers:** Collate all instances requiring recommendations:
   - Individual questions flagged as incorrect (`is_correct`==False) or correct but overtime (`is_correct`==True AND `is_slow`==True) based on Chapters 3 & 4 analyses.
   - Skills (Q/V) or Types (DI) flagged by the override rule (`skill_override_triggered` / `override_triggered`) in Chapter 6.
   - **DI-Specific:** MSR groups with overtime or reading inefficiency issues identified in Chapter 1.
2. **Generate Recommendations (Iterative Process):**
   - **Exemption Check:** Before generating a case-specific recommendation for a skill (Q/V) or type-/domain combination (DI), check if it exhibits stable performance. This is determined by verifying if **all** valid questions within that category are **both correct AND not overtime**. If this condition is met (100% accuracy and 100% efficiency within valid data), the category is considered exempt, and case-specific recommendations for it are skipped. Exemption notes are recorded.
   - **Override Check:** If a skill/type has its override flag set (`True` from Chapter 6), generate **only one Macroscopic Recommendation** for that entire skill/type, using the pre-calculated `Y_agg` and `Z_agg`. This recommendation emphasizes foundational practice. Mark the skill/type as processed to prevent adding further case-specific suggestions for it.
   - **Generate Case-Specific or Aggregated Recommendation (If not overridden or exempt):** The process varies by section:
     - **Quant (Q):** Generates an independent recommendation for *each individual triggering incorrect/overtime question (trigger point).* No aggregation is performed.
     - **Verbal (V) & Data Insights (DI):** Generally aggregates findings. For each `fundamental_skill` (V) or `question_type` + `content_domain` combination (DI) with triggers, it generates *one single aggregated recommendation* covering all trigger points within that category.
     - **DI MSR Groups:** For MSR groups with reading or timing issues, generate MSR-specific recommendations focusing on reading strategy and time allocation.

   The following steps detail how the parameters (Y, Z, Annotations) are determined, considering these section differences:

   (a) Determine Practice Difficulty (`Y`):
     - **For Q (Per Trigger Point):** Map the original question's difficulty (`D`) to the standardized 6-level label.
     - **For V/DI (Aggregated Group):** Find the *minimum* `question_difficulty` among all triggering questions within the skill (V) or type/domain (DI) group. Map this minimum difficulty to the standardized 6-level label.
     - **For DI MSR Groups:** Use the minimum difficulty across all MSR questions in the problematic group.

   (b) Determine Starting Practice Time Limit (`Z`): Employs a unified calculation rule for individual triggers, but the final Z depends on aggregation:

- *Individual Z Calculation (Base Rule for all sections):* For any given triggering question:
  - Define `target_time` based on question type:
    - Q: 2.0 min
    - DI: DS=2.0, TPA=3.0, GT=3.0, MSR_reading=1.5 min
    - V: CR=2.0, RC=1.5 min
  - Calculate `base_time` = `question_time` - 0.5 (if `is_slow`) else `question_time`.
  - Calculate `Z_raw` = `floor(base_time * 2) / 2`.
  - Set Individual `Z_indiv` = `max(Z_raw, target_time)`.
- *Final Z Determination:*
  - **For Q (Per Trigger Point):** The final `Z` is simply the `Z_indiv` calculated for that specific trigger.
  - **For V/DI (Aggregated Group):** Calculate `Z_indiv` for *all* triggering questions within the group. The final aggregated `Z` for the group recommendation is the *maximum* of all these calculated `Z_indiv` values.
  - **For DI MSR Groups:** Use group-level time allocation recommendations (6.0-7.0 min per group depending on pressure status).

(c) Construct Suggestion Text: Include skill/type/domain, brief issue description (from Ch 3/4), difficulty `Y`, time `Z`, and target time. For DI MSR groups, include reading strategy recommendations.

(d) Apply Annotations:
- Prepend "*Fundamental mastery potentially unstable:*" or similar if triggered by `special_focus_error` (from Ch 3). For V/DI, if any question in the group triggered SFE, the aggregated recommendation gets the prefix.
- Append volume alert like "**(Requires increased practice volume...)**" based on section-specific rules:
  - **Quant (Q):** If final `Z` > 4.0 min.
  - **Verbal (V) & DI:** If final `Z` - `target_time` > 2.0 min.
  - **DI MSR Groups:** If group time allocation exceeds 7.0 min or reading time exceeds 2.0 min.

3. **Organize and Finalize Output:**

- Group all generated recommendations (macroscopic, case-specific, exemption notes, MSR-specific) by `fundamental_skill` (Q/V) or `question_type` (DI).
- Apply Focus Rules: Adjust recommendation text based on Chapter 2 flags (e.g., if `poor_real` in Q, suggest higher proportion of 'Real' questions for relevant skills; if `poor_math_related` in DI, emphasize math-related practice).
- Prioritize/Highlight SFE-related recommendations within the grouped list.
- Sort the final list of recommendations (e.g., by priority derived from SFE status, override status, or MSR group criticality).

Implementation Context: This logic typically resides in the latter part of the main analysis function. Exemption checks require pre-calculating relevant metrics per category. Recommendation generation involves iterating through triggers or groups, applying conditional logic for exemption/override, calculating Y/Z using helper functions or inline logic based on section rules (aggregation, min/max), formatting text strings with annotations, and storing them (e.g., in a list of dictionaries). Focus rules are applied during the final organization phase. For DI sections, special handling is required for MSR groups including reading time analysis and group-level time allocation. The flowchart in Figure 4 illustrates the core calculation steps for determining Y and Z for a single trigger point, which forms the basis for Q recommendations and is a pre-aggregation step for V/DI. To enhance the fluency and clarity of the generated report, AI-powered tools may optionally be utilized to assist in synthesizing the findings and recommendations into user-friendly prose.

Rationale: Provides a holistic, actionable summary translating complex quantitative diagnostics and parameterized recommendations into clear, user-friendly insights and next steps, maximizing the practical value of the analysis for the student. For DI sections, this includes specialized MSR reading strategy guidance critical for performance improvement.

2.9 Chapter 8: Diagnostic Summary and Subsequent Actions

Objective: Synthesize all analysis findings and recommendations into a comprehensive, easily understandable report for the student, **using exclusively natural language**.

Report Structure and Content Synthesis:

1. **Opening Summary:** This initial section synthesizes key findings from Chapter 1, including: the assessed `time_pressure` status (described naturally, e.g., "significant time pressure was likely experienced"), total time usage relative to the limit, and whether any data was deemed invalid due to end-section rushing (e.g., "analysis excludes X questions answered hastily at the end under pressure"). **For DI sections**, include MSR group timing assessment and reading efficiency evaluation.

2. **Performance Overview:** Summarizes Chapter 2 results descriptively: Relative performance across key dimensions (Q: 'Real' vs. 'Pure'; DI: 'Math Related' vs. 'Non-Math Related', types 'DS', 'TPA', 'GT'; V: Skill performance, 'CR' vs. 'RC') and difficulty levels (e.g., "Errors were concentrated in the High difficulty range," "Performance on 'Math Related' DI questions was significantly weaker than 'Non-Math Related'"). Mention reading time assessment for V if flagged. **For DI**, specifically address MSR group performance and reading efficiency patterns.

3. **Core Problem Diagnosis:** Translates Chapter 3 findings into narrative form: Describes primary error patterns identified (e.g., "A tendency towards calculation errors in Pure Quant questions was observed," "Difficulties in interpreting complex graph relationships under time pressure were noted in GT questions," "Reading comprehension challenges in MSR passages affected subsequent question accuracy," "Logical flaws in evaluating assumptions were apparent in CR questions"). **Crucially highlights SFE findings** using descriptive language (e.g., "Particular attention is needed for errors occurring on questions below your typical mastery level in [Skill/Type], suggesting instability in applying fundamental concepts."). **Includes MSR-specific time issues for DI** if detected, with detailed reading strategy analysis.

4. **Pattern Observation:** Integrates Chapter 5 alerts: Notes risks associated with early-stage rapid responses or potential carelessness if the `carelessness_issue` flag was triggered (e.g., "A pattern of rapid, incorrect answers suggests that focusing on accuracy over speed may be beneficial"). **For DI**, includes MSR reading pattern observations and time allocation efficiency notes.

5. **Foundational Consolidation Advisory:** Explicitly states which skills/types require systematic foundational work based on Chapter 6 override triggers (e.g., "Systematic review and practice of the fundamentals for [Skill/Type] is recommended due to overall performance patterns"). **For DI**, includes MSR reading strategy consolidation recommendations when applicable.

6. **Practice Plan Presentation:** Presents the full, organized list of recommendations generated in Chapter 7, ensuring clarity and including all annotations (priority, volume alerts, focus rules, exemptions, MSR-specific guidance) in natural language descriptions within the recommendation text itself.

7. **Guidance for Subsequent Actions:** Provides actionable next steps:

   - **Guiding Reflection Questions:** Poses targeted, open-ended questions based on the specific diagnoses to prompt student self-assessment. **For DI**, incorporates MSR-specific questions about reading approach, note-taking strategy, and time allocation between reading and questions.
   - **Secondary Evidence Review Suggestion:** Explains *when* (e.g., uncertain recall, need pattern confirmation, MSR strategy unclear) and *how* (e.g., review recent logs, focus on specific error types, analyze MSR reading habits) to use past practice data.

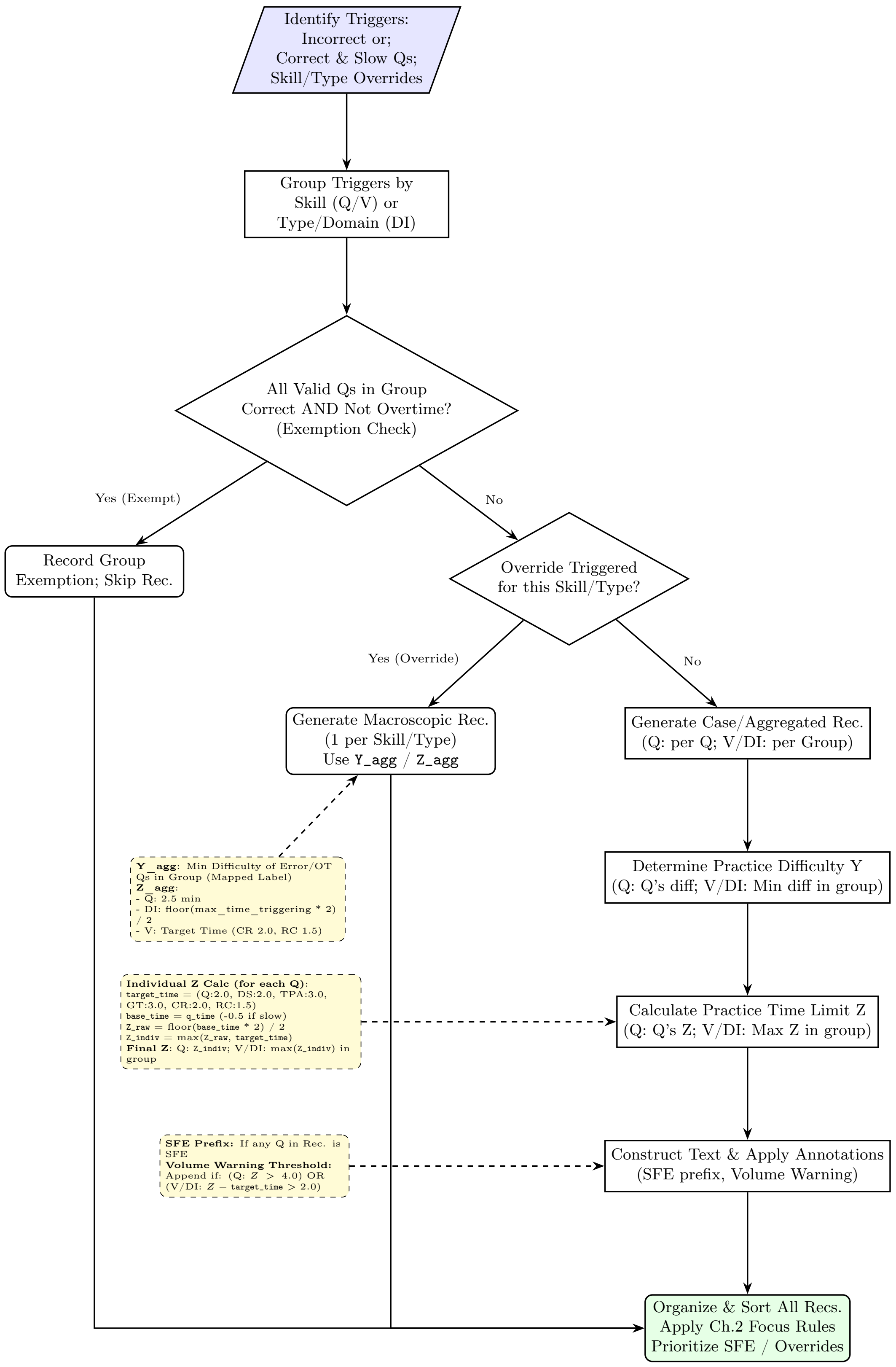

Identify Triggers: Incorrect or; Correct & Slow Qs; Skill/Type Overrides
Group Triggers by Skill (Q/V) or Type/Domain (DI)
All Valid Qs in Group Correct AND Not Overtime? (Exemption Check)
Yes (Exempt)
No
Record Group Exemption; Skip Rec.
Override Triggered for this Skill/Type?
Yes (Override)
No
Generate Macroscopic Rec. (1 per Skill/Type) Use Y_agg / Z_agg
Generate Case/Aggregated Rec. (Q: per Q; V/DI: per Group)
Y_agg: Min Difficulty of Error/OT Qs in Group (Mapped Label)
Z_agg:
- Q: 2.5 min
- DI: floor(max_time_triggering * 2) / 2
- V: Target Time (CR 2.0, RC 1.5)
Determine Practice Difficulty Y (Q: Q's diff; V/DI: Min diff in group)
Individual Z Calc (for each Q):
target_time = (Q:2.0, DS:2.0, TPA:3.0, GT:3.0, CR:2.0, RC:1.5)
base_time = q_time (-0.5 if slow)
Z_raw = floor(base_time * 2) / 2
Z_indiv = max(Z_raw, target_time)
Final Z: Q: Z_indiv; V/DI: max(Z_indiv) in group
Calculate Practice Time Limit Z (Q: Q's Z; V/DI: Max Z in group)
SFE Prefix: If any Q in Rec. is SFE
Volume Warning Threshold:
Append if: (Q: Z > 4.0) OR (V/DI: Z − target_time > 2.0)
Construct Text & Apply Annotations (SFE prefix, Volume Warning)
Organize & Sort All Recs. Apply Ch.2 Focus Rules Prioritize SFE / Overrides

Figure 4: Flowchart for Chapter 7 Logic: Practice Recommendation Generation

- **Qualitative Analysis Suggestion:** Explains *when* (e.g., root cause remains unclear after other steps, MSR reading strategy needs refinement) and *how* (e.g., provide detailed walkthroughs for specific problem types, MSR reading technique demonstration) to engage in deeper analysis, potentially with an advisor.

Core Constraint Adherence: The implementation generating this chapterś output must rigorously avoid exposing internal variable names (e.g., `time_pressure`, `is_invalid`, `special_focus_error`, `msr_group_id`), specific performance flags, numerical thresholds, or calculation details. All findings must be translated into clear, easily understandable, descriptive prose suitable for the end-user (student). **DI-specific technical details** such as MSR group algorithms, reading time calculations, and statistical thresholds should be abstracted into user-friendly reading strategy guidance.

Implementation Context: This is typically handled by a dedicated `generate_report_*` function that receives all computed metrics, flags, lists, and generated recommendation texts as arguments. It uses conditional statements (`if/else`) and formatted strings (`f-strings`) to assemble the report sections based on the presence and values of these inputs, ensuring adherence to the natural language constraint. **For DI sections**, special handling functions process MSR group data and convert technical timing analysis into readable reading strategy recommendations.

Rationale: Provides a holistic, actionable summary translating complex quantitative diagnostics and parameterized recommendations into clear, user-friendly insights and next steps, maximizing the practical value of the analysis for the student. The natural language presentation ensures accessibility while maintaining the analytical rigor of the underlying diagnostic framework. **For DI sections**, this includes making MSR reading strategy analysis accessible and actionable for students.

2.10 Illustrative Example: Simplified Question Journey

To illustrate how the framework processes a single question, consider this simplified hypothetical example for a Quant ('Pure') question:

Input Data:

- `question_id`: Q5
- `question_time`: 3.2 min
- `is_correct`: False
- `question_difficulty`: 650
- `question_position`: 10 (out of 21)
- `question_type`: 'Pure'
- `question_fundamental_skill`: 'Algebra'
- (Assume `total_test_time` = 43 min, `max_allowed_time` = 45 min, implies `time_pressure` = False based on Q/DI rule)
- (Assume `average_time_per_type` for 'Pure' = 2.1 min)
- (Assume `max_mastered_difficulty` for 'Algebra' = 700)

Framework Steps Applied:

1. **Chapter 1 (Validity/Overtime):**
   - `time_pressure` is False.
   - `question_position` is not in the last third, and `question_time` $>$ 1.0 min $\rightarrow$ `is_invalid` = False. (Data is valid).
   - The `overtime_threshold` for Q (no pressure) is 3.0 min. Since 3.2 min $>$ 3.0 min $\rightarrow$ `overtime` = True.
2. **Chapter 2 (Performance - Context):** This question contributes to the overall metrics for 'Pure' type, 'Algebra' skill, and the corresponding difficulty band (e.g., "Medium / 605+").
3. **Chapter 3 (Root Cause):**
   - Time Classification: 3.2 min is not $<$ (2.1 * 0.75) $\rightarrow$ `is_relatively_fast` = False. Since `overtime` is True $\rightarrow$ `is_slow` = True.
   - SFE Check: `is_correct` is False, and `question_difficulty` (650) $<$ `max_mastered_difficulty` (700) $\rightarrow$ `special_focus_error` = True.
   - Scenario: Slow & Wrong & SFE.
4. **Chapter 6 (Coverage - Check):** Check if 'Algebra' skill has `error_rate` or `overtime_rate` $>$ 50%. (Assume for this example it does not $\rightarrow$ `skill_override_triggered`['Algebra'] = False).
5. **Chapter 7 (Recommendation):**
   (a) Trigger: Incorrect question (Q5).
   (b) Exemption/Override: Not exempt (based on other Algebra questions) and not overridden.
   (c) Y (Difficulty): Map 650 to label (e.g., "Medium / 605+").
   (d) Z (Time): `is_slow` is True. `base_time` = 3.2 - 0.5 = 2.7. `Z_raw` = floor(2.7 * 2) / 2 = floor(5.4) / 2 = 5.0 / 2 = 2.5. `target_time` for Q is 2.0. `Z` = max(2.5, 2.0) = 2.5 min.
   (e) Annotation: Prepend "*Fundamental mastery potentially unstable:* Practice Algebra ('Pure') problems at Medium / 605+ difficulty, starting with a 2.5 min time limit (target 2.0 min)."

This example shows the flow from raw data through flagging (overtime, SFE) to a specific, annotated recommendation based on the framework's rules.

2.11 Implementation Details

The unified diagnostic framework described herein has been implemented as a set of Python scripts (`gmat_q_analyzer.py`, `gmat_di_analyzer.py`, `gmat_v_analyzer.py`), one for each GMAT section. The development followed an iterative process involving requirements analysis based on the methodological documents (`en-gmat-.md`), coding, testing with sample datasets (`testset-.csv`), debugging, and refinement based on analysis outcomes and evolving reporting requirements.

Core Technologies:

- **Python:** The primary programming language.
- **Pandas:** Extensively used for data manipulation, including reading CSV files, data cleaning (handling missing values, type conversion), filtering, grouping, aggregation, and time-series operations where applicable. DataFrame structures are central to storing and processing the per-question data and analytical results.
- **NumPy:** Utilized for numerical operations, handling potential `NaN` values, and creating placeholder data structures (e.g., spacer rows using `np.nan`).

- **Argparse:** Employed for command-line argument parsing, allowing users to specify input/output file paths and optional flags (e.g., overriding `time_pressure` status).

Development Challenges & Solutions:

- **Environment & Dependencies:** Initial challenges included resolving Python interpreter path issues and managing dependencies (`pandas`, `numpy`). The use of Python virtual environments (`venv`) was adopted to ensure consistent and isolated execution environments, addressing `externally-managed-environment` errors related to PEP 668.
- **Data Inconsistencies:** Handling variations in input CSV formats, such as extra commas in headers or differing column names (e.g., `V_b`, `DI_b` vs. a consistent `question_difficulty`), required robust parsing logic (e.g., using `usecols` in `pd.read_csv`, dynamic column renaming). Encoding issues were addressed by attempting multiple common encodings (`utf-8`, `gbk`, `cp950`).
- **Logic Implementation:** Translating the sometimes complex, multi-conditional logic from the markdown documents into precise code required careful structuring, particularly for overtime calculations, SFE detection, and recommendation generation involving multiple interacting rules (override, exemption, focus). Helper functions were created to encapsulate reusable logic (e.g., difficulty mapping, safe division, Z-time calculation, MSR/RC group processing).
- **Pandas Operations:** Specific `pandas` operations occasionally led to errors (e.g., ambiguity in boolean operations on DataFrames, errors during DataFrame initialization with specific structures like `[[]] * n`), necessitating refactoring to use element-wise operations or more robust initialization methods (`pd.DataFrame(np.nan, ...)`).
- **Evolving Requirements:** The most significant evolution was the increasing demand for report granularity. Initial implementations produced summary-level reports, but later iterations required significant refactoring to generate detailed, per-question diagnostics, actions, reflections, and evidence prompts, necessitating the creation of `get_detailed_diagnosis_*` functions and modifications to the main analysis loop and report generation logic across all three scripts.
- **Debugging:** Identifying the root cause of errors often involved tracing data flow through the DataFrame, checking intermediate values, and verifying that conditional logic correctly handled edge cases (e.g., division by zero, empty data subsets after filtering). `NameError` and `KeyError` issues were common during refactoring, requiring careful checking of variable scope and dictionary key handling.

Output Data Format (Annotated CSV): In addition to the natural language summary report, the Python scripts generate an output CSV file (e.g., `testset-q-analyzed.csv`). This file contains all the original input data plus several new columns representing the diagnostic flags calculated during the analysis. Key added columns include:

- `difficulty_label`: The standardized difficulty category (e.g., "Medium / 605+").
- `time_pressure`: Boolean flag indicating if overall time pressure was detected (Chapter 1).
- `is_invalid`: Boolean flag indicating if the question data was excluded due to extreme rushing under time pressure (Chapter 1).
- `overtime` (or variants like `group_overtime`): Boolean flag indicating if the question (or group) exceeded the calculated overtime threshold based on the *filtered* data (Chapter 1).
- `is_relatively_fast`: Boolean flag indicating if the question was answered significantly faster than the average for its type (Chapter 3).
- `is_slow`: Boolean flag, essentially mirroring the `overtime` flag for consistency in time classification (Chapter 3).
- `max_mastered_difficulty`: The highest difficulty level mastered within the relevant skill/type category based on correct answers (Chapter 3).

- `special_focus_error`: A Boolean indicator signifying an error made on a question possessing a difficulty level inferior to the user's attained `max_mastered_difficulty` within that specific subject category (as specified in Chapter 3).
- `skill_override_triggered` / `override_triggered`: Boolean flag indicating if a coverage rule was triggered for the question's skill/type (Chapter 6).
- `Recommendation_Y`: The suggested practice difficulty level (String label).
- `Recommendation_Z`: The suggested starting practice time limit (Numeric, minutes).
- `Diagnostic_Notes`: Text field containing generated diagnostic comments or recommendation details.

This annotated CSV allows for detailed review and further analysis of the framework's per-question conclusions.

Current State: The resulting Python scripts represent functional implementations of the diagnostic framework. They successfully ingest section-specific GMAT data, apply the complex analytical logic outlined in Chapters 0-7, handle various data edge cases, and produce detailed output CSV files containing both per-question diagnostic flags and a comprehensive, natural-language summary report adhering to the structure of Chapter 8. The scripts are designed to be run from the command line, providing a practical tool for automated GMAT performance analysis.

3 Conclusion

The primary value of this framework, as detailed in this technical report, lies in its systematic and unified approach to formalizing the complex process of GMAT performance diagnosis across multiple sections. It provides a transparent, structured, and potentially automatable methodology moving beyond simple metrics.

As documentation focused on the framework's design, readers should note its current limitations. The specific parameter values used in the implementation represent informed heuristics derived from initial analysis and expert consultation, rather than statistically optimized values derived from large-scale empirical validation. While preliminary comparisons to expert analysis are encouraging, rigorous validation of the framework's diagnostic accuracy and effectiveness remains a crucial next step for future development.

The unified GMAT diagnostic framework presented offers a systematic, multi-faceted approach to analyzing student performance across the Quantitative, Data Insights, and Verbal sections. By adhering to a consistent chapter-based structure while accommodating section-specific nuances through parameterized logic and tailored analyses, the framework moves beyond superficial score reporting to identify root causes of errors and inefficiencies. Key strengths include:

- **Comprehensive Scope:** Addresses time management, data validity, accuracy, efficiency, behavioral patterns, and foundational knowledge across all scored sections.
- **Depth of Analysis:** Employs concepts like relative time performance, `special_focus_error` detection, and coverage rules to provide nuanced diagnoses based on operationalized parameters.
- **Actionable Output:** Generates personalized practice plans with specific difficulty (`Y`/`Y_agg`) and time (`Z`/`Z_agg`) parameters, alongside structured guidance for self-reflection and further analysis.
- **Standardization & Adaptability:** Provides a consistent analytical process applicable to Q, DI, and V, ensuring comparable insights while respecting the unique demands of each section, as evidenced by the successful implementation across three distinct Python modules.

This framework, validated through empirical testing and refinement, empowers students and instructors with detailed, data-driven insights, facilitating more targeted preparation and ultimately aiming for improved GMAT performance.

4 Future Directions

Building upon the established framework and its current implementation, several avenues for future development and application are envisioned:

1. **Web-Based Implementation:** To enhance accessibility and utility, the diagnostic framework is planned for implementation as an automated, user-facing tool integrated into a web platform. This would allow students to upload their score data (e.g., via CSV or potentially through direct API integration if available) and receive instant, standardized diagnostic reports through a user-friendly interface.

2. **Rigorous Parameter Optimization and Validation:** Beyond initial heuristics, future work involves rigorously validating and optimizing framework parameters (e.g., time thresholds, speed factors, SFE sensitivity, coverage rules) and predictive accuracy. Hyperparameter optimization methods like Grid Search will be used on larger student datasets to find parameter sets maximizing concordance between the framework's diagnoses and validated metrics or expert assessments, enhancing diagnostic and predictive effectiveness.

3. **Integration of Qualitative Feedback:** Explore methods to integrate qualitative student feedback (e.g., self-reported reasons for errors, confidence levels) directly into the diagnostic process, potentially refining the root cause analysis beyond purely quantitative data.

4. **Longitudinal Analysis:** Extend the framework to analyze performance trends over multiple test administrations or practice sessions for a single student, identifying patterns of improvement or persistent weaknesses.

Appendix A: Diagnostic Parameter Tags and Descriptions

Table 1: Diagnostic Parameter Tags and Descriptions

Diagnostic Parameter Tags and Descriptions
Reading / Comprehension / Interpretation
`Q_READING_COMPREHENSION_ERROR` Q Reading: Difficulty understanding word problem text (Real Context)
`Q_PROBLEM_UNDERSTANDING_ERROR` Q Problem Understanding: Misinterpretation of question requirements/logic
`DI_READING_COMPREHENSION_ERROR` DI Reading Comprehension: Error/Difficulty understanding text (Math/Non-Math)
`DI_GRAPH_TABLE_INTERPRETATION_ERROR` DI Graph/Table Interpretation: Error/Difficulty interpreting visual data
`DI_MSR_READING_COMPREHENSION_BARRIER` DI MSR Reading Barrier: Excessive overall reading time affecting group performance
Concept / Logic Application
`Q_CONCEPT_APPLICATION_ERROR` Q Concept Application: Error applying mathematical concepts/formulas
`DI_CONCEPT_APPLICATION_ERROR` DI Concept Application (Math): Error applying mathematical concepts/formulas
`DI_LOGICAL_REASONING_ERROR` DI Logical Reasoning (Non-Math): Error in inherent logical reasoning/judgment
Data Handling / Calculation / Location
`Q_CALCULATION_ERROR` Q Calculation: Error in mathematical computation
`DI_DATA_EXTRACTION_ERROR` DI Data Extraction (GT): Error extracting data from graphs/tables
`DI_INFORMATION_EXTRACTION_INFERENCE_ERROR` DI Info Extraction/Inference (GT/MSR Non-Math): Error locating/inferring information
`DI_CALCULATION_ERROR` DI Calculation: Error in mathematical computation
`DI_MSR_TIME_ALLOCATION_ISSUE` DI MSR Time Allocation: Inefficient time distribution between reading and questions
Section/Type Specific (MSR - DI Only)
`DI_MULTI_SOURCE_INTEGRATION_ERROR` DI Multi-Source Integration (MSR): Error integrating information across multiple sources
`DI_MSR_READING_STRATEGY_INEFFICIENCY` DI MSR Reading Strategy: Inefficient reading approach for multi-source materials
`DI_MSR_GROUP_TIMING_MANAGEMENT_ERROR` DI MSR Group Timing Management: Poor time management across MSR question groups
`DI_QUESTION_TYPE_SPECIFIC_ERROR` DI Question Type Specific Error (e.g., DS, TPA, GT specific weaknesses)
`DI_MSR_READING_DEPTH_VS_SPEED_IMBALANCE` DI MSR Reading: Imbalance between reading depth and speed requirements
Critical Reasoning (CR) - Verbal

Continued on next page

Table 1 – *Continued from previous page*
CR Stem Understanding Errors
`CR_STEM_UNDERSTANDING_ERROR_QUESTION_REQUIREMENT_GRASP` CR Stem Understanding Error: Misinterpretation of question requirements
`CR_STEM_UNDERSTANDING_ERROR_VOCAB` CR Stem Understanding Error: Vocabulary issues in stem
`CR_STEM_UNDERSTANDING_ERROR_SYNTAX` CR Stem Understanding Error: Syntactical complexity in stem
`CR_STEM_UNDERSTANDING_ERROR_LOGIC` CR Stem Understanding Error: Logical structure of stem misunderstood
`CR_STEM_UNDERSTANDING_ERROR_DOMAIN` CR Stem Understanding Error: Domain-specific content in stem misunderstood
CR Reasoning Errors
`CR_REASONING_ERROR_LOGIC_CHAIN_ANALYSIS_PREMISE_CONCLUSION_RELATIONSHIP` CR Reasoning Error: Flawed analysis of premise-conclusion relationship
`CR_REASONING_ERROR_ABSTRACT_LOGIC_TERMINOLOGY_UNDERSTANDING` CR Reasoning Error: Misunderstanding of abstract logic or terminology
`CR_REASONING_ERROR_PREDICTION_DIRECTION` CR Reasoning Error: Incorrect prediction of answer direction
`CR_REASONING_ERROR_CORE_ISSUE_IDENTIFICATION` CR Reasoning Error: Failure to identify the core issue of the argument
`CR_REASONING_ERROR_CHOICE_RELEVANCE_JUDGEMENT` CR Reasoning Error: Incorrect judgment of answer choice relevance
`CR_REASONING_ERROR_STRONG_DISTRACTOR_CHOICE_CONFUSION` CR Reasoning Error: Confusion caused by strong distractor choices
CR Choice Understanding Errors
`CR_CHOICE_UNDERSTANDING_ERROR_VOCAB` CR Choice Understanding Error: Vocabulary issues in answer choices
`CR_CHOICE_UNDERSTANDING_ERROR_SYNTAX` CR Choice Understanding Error: Syntactical complexity in answer choices
`CR_CHOICE_UNDERSTANDING_ERROR_LOGIC` CR Choice Understanding Error: Logical structure of answer choices misunderstood
`CR_CHOICE_UNDERSTANDING_ERROR_DOMAIN` CR Choice Understanding Error: Domain-specific content in choices misunderstood
CR Specific Question Type Weakness
`CR_SPECIFIC_QUESTION_TYPE_WEAKNESS_NOTE_TYPE` CR Specific Question Type Weakness: (Type to be noted)
CR Stem Understanding Difficulties (Efficiency Issue / Slow Performance)
`CR_STEM_UNDERSTANDING_DIFFICULTY_VOCAB` CR Stem Understanding Difficulty: Vocabulary challenges in stem
`CR_STEM_UNDERSTANDING_DIFFICULTY_SYNTAX` CR Stem Understanding Difficulty: Syntactical complexity in stem causing slowdown
`CR_STEM_UNDERSTANDING_DIFFICULTY_LOGIC` CR Stem Understanding Difficulty: Logical structure of stem hard to grasp
`CR_STEM_UNDERSTANDING_DIFFICULTY_DOMAIN` CR Stem Understanding Difficulty: Domain-specific content in stem hard to grasp
CR Reasoning Difficulties (Efficiency Issue / Slow Performance)
Continued on next page

Table 1 – *Continued from previous page*
`CR_REASONING_DIFFICULTY_ABSTRACT_LOGIC_TERMINOLOGY_UNDERSTANDING` CR Reasoning Difficulty: Difficulty with abstract logic or terminology
`CR_REASONING_DIFFICULTY_PREDICTION_DIRECTION_MISSING` CR Reasoning Difficulty: Missing or slow prediction of answer direction
`CR_REASONING_DIFFICULTY_CORE_ISSUE_IDENTIFICATION` CR Reasoning Difficulty: Difficulty identifying the core issue of the argument
`CR_REASONING_DIFFICULTY_CHOICE_RELEVANCE_JUDGEMENT` CR Reasoning Difficulty: Difficulty judging answer choice relevance
`CR_REASONING_DIFFICULTY_STRONG_DISTRACTOR_CHOICE_ANALYSIS` CR Reasoning Difficulty: Difficulty analyzing strong distractor choices
CR Choice Understanding Difficulties (Efficiency Issue / Slow Performance)
`CR_CHOICE_UNDERSTANDING_DIFFICULTY_VOCAB` CR Choice Understanding Difficulty: Vocabulary challenges in answer choices
`CR_CHOICE_UNDERSTANDING_DIFFICULTY_SYNTAX` CR Choice Understanding Difficulty: Syntactical complexity in choices causing slowdown
`CR_CHOICE_UNDERSTANDING_DIFFICULTY_LOGIC` CR Choice Understanding Difficulty: Logical structure of choices hard to grasp
`CR_CHOICE_UNDERSTANDING_DIFFICULTY_DOMAIN` CR Choice Understanding Difficulty: Domain-specific content in choices hard to grasp
Reading Comprehension (RC) - Verbal
RC Reading Comprehension Errors
`RC_READING_COMPREHENSION_ERROR_VOCAB` RC Reading Comprehension Error: Vocabulary misunderstanding
`RC_READING_COMPREHENSION_ERROR_LONG_DIFFICULT_SENTENCE_ANALYSIS` RC Reading Comprehension Error: Incorrect analysis of long/difficult sentences
`RC_READING_COMPREHENSION_ERROR_PASSAGE_STRUCTURE` RC Reading Comprehension Error: Misunderstanding of passage structure
`RC_READING_COMPREHENSION_ERROR_KEY_INFO_LOCATION_UNDERSTANDING` RC Reading Comprehension Error: Incorrect location or understanding of key information
RC Question Understanding Errors
`RC_QUESTION_UNDERSTANDING_ERROR_FOCUS_POINT` RC Question Understanding Error: Misinterpretation of the question's focus point
RC Location Skill Errors
`RC_LOCATION_SKILL_ERROR_LOCATION` RC Location Skill Error: Incorrect location of information in the passage
RC Reasoning Errors
`RC_REASONING_ERROR_INFERENCE` RC Reasoning Error: Flawed inference from the passage text
RC Choice Analysis Errors
`RC_CHOICE_ANALYSIS_ERROR_VOCAB` RC Choice Analysis Error: Vocabulary issues in answer choices
`RC_CHOICE_ANALYSIS_ERROR_SYNTAX` RC Choice Analysis Error: Syntactical complexity in answer choices
`RC_CHOICE_ANALYSIS_ERROR_LOGIC` RC Choice Analysis Error: Logical structure of answer choices misunderstood
`RC_CHOICE_ANALYSIS_ERROR_DOMAIN`
Continued on next page

Table 1 – *Continued from previous page*
RC Choice Analysis Error: Domain-specific content in choices misunderstood
`RC_CHOICE_ANALYSIS_ERROR_RELEVANCE_JUDGEMENT` RC Choice Analysis Error: Incorrect judgment of answer choice relevance
`RC_CHOICE_ANALYSIS_ERROR_STRONG_DISTRACTOR_CONFUSION` RC Choice Analysis Error: Confusion caused by strong distractor choices
RC Method Errors
`RC_METHOD_ERROR_SPECIFIC_QUESTION_TYPE_HANDLING` RC Method Error: Incorrect handling of specific question types
RC Reading Comprehension Difficulties (Efficiency Issue / Slow Performance)
`RC_READING_COMPREHENSION_DIFFICULTY_VOCAB_BOTTLENECK` RC Reading Comprehension Difficulty: Vocabulary bottleneck
`RC_READING_COMPREHENSION_DIFFICULTY_LONG_DIFFICULT_SENTENCE_ANALYSIS` RC Reading Comprehension Difficulty: Difficulty analyzing long/difficult sentences
`RC_READING_COMPREHENSION_DIFFICULTY_PASSAGE_STRUCTURE_GRASP_UNCLEAR` RC Reading Comprehension Difficulty: Unclear grasp of passage structure
`RC_READING_COMPREHENSION_DIFFICULTY_SPECIFIC_DOMAIN_BACKGROUND_KNOWLEDGE_LACK` RC Reading Comprehension Difficulty: Lack of specific domain background knowledge
RC Question Understanding Difficulties (Efficiency Issue / Slow Performance)
`RC_QUESTION_UNDERSTANDING_DIFFICULTY_FOCUS_POINT_GRASP` RC Question Understanding Difficulty: Difficulty grasping the question's focus point
RC Location Skill Difficulties (Efficiency Issue / Slow Performance)
`RC_LOCATION_SKILL_DIFFICULTY_INEFFICIENCY` RC Location Skill Difficulty: Inefficient location of information
RC Reasoning Difficulties (Efficiency Issue / Slow Performance)
`RC_REASONING_DIFFICULTY_INFERENCE_SPEED_SLOW` RC Reasoning Difficulty: Slow inference speed
RC Choice Analysis Difficulties (Efficiency Issue / Slow Performance)
`RC_CHOICE_ANALYSIS_DIFFICULTY_VOCAB` RC Choice Analysis Difficulty: Vocabulary challenges in answer choices
`RC_CHOICE_ANALYSIS_DIFFICULTY_SYNTAX` RC Choice Analysis Difficulty: Syntactical complexity in choices causing slowdown
`RC_CHOICE_ANALYSIS_DIFFICULTY_LOGIC` RC Choice Analysis Difficulty: Logical structure of choices hard to grasp
`RC_CHOICE_ANALYSIS_DIFFICULTY_DOMAIN` RC Choice Analysis Difficulty: Domain-specific content in choices hard to grasp
`RC_CHOICE_ANALYSIS_DIFFICULTY_RELEVANCE_JUDGEMENT` RC Choice Analysis Difficulty: Difficulty judging answer choice relevance
`RC_CHOICE_ANALYSIS_DIFFICULTY_STRONG_DISTRACTOR_ANALYSIS` RC Choice Analysis Difficulty: Difficulty analyzing strong distractor choices
RC Method Difficulties (Efficiency Issue / Slow Performance)
`RC_METHOD_DIFFICULTY_SPECIFIC_QUESTION_TYPE_HANDLING` RC Method Difficulty: Difficulty handling specific question types
Foundational Mastery Instability
`Q_FOUNDATIONAL_MASTERY_APPLICATION_INSTABILITY_SFE` Q Foundational Mastery: Application instability (Special Focus Error)
`DI_FOUNDATIONAL_MASTERY_APPLICATION_INSTABILITY_SFE`
Continued on next page

Table 1 – *Continued from previous page*
DI Foundational Mastery: Application instability (Special Focus Error)
`V_FOUNDATIONAL_MASTERY_APPLICATION_INSTABILITY_SFE` V Foundational Mastery: Application instability (Special Focus Error)
`FOUNDATIONAL_MASTERY_APPLICATION_INSTABILITY_SFE` General Foundational Mastery: Application instability (Special Focus Error)
Efficiency Bottlenecks (Time-Based Performance Issues)
`Q_EFFICIENCY_BOTTLENECK_READING` Q Efficiency Bottleneck: Reading time issues (Real Context problems)
`Q_EFFICIENCY_BOTTLENECK_CONCEPT` Q Efficiency Bottleneck: Concept recall/application time
`Q_EFFICIENCY_BOTTLENECK_CALCULATION` Q Efficiency Bottleneck: Calculation time
`DI_EFFICIENCY_BOTTLENECK_READING` DI Efficiency Bottleneck: Reading/Comprehension time (Math/Non-Math)
`DI_EFFICIENCY_BOTTLENECK_CONCEPT` DI Efficiency Bottleneck: Concept/Formula application time (Math)
`DI_EFFICIENCY_BOTTLENECK_CALCULATION` DI Efficiency Bottleneck: Mathematical computation time
`DI_EFFICIENCY_BOTTLENECK_LOGIC` DI Efficiency Bottleneck: Logical reasoning time (Non-Math)
`DI_EFFICIENCY_BOTTLENECK_GRAPH_TABLE` DI Efficiency Bottleneck: Graph/Table interpretation time
`DI_EFFICIENCY_BOTTLENECK_INTEGRATION` DI Efficiency Bottleneck: Multi-source integration time (MSR)
`DI_MSR_EFFICIENCY_BOTTLENECK_READING_DISTRIBUTION` DI MSR Efficiency Bottleneck: Poor reading time distribution within groups
Behavioral Patterns and Data Quality Issues
`DATA_INVALID_SHORT_TIME_PRESSURE_AFFECTED` Data Invalid: Short time due to time pressure affecting validity
`Q_CARELESSNESS_DETAIL_OMISSION` Q Behavior: Carelessness - Detail omission/misread (Fast & Wrong pattern)
`DI_CARELESSNESS_DETAIL_OMISSION` DI Behavior: Carelessness - Detail omission/misread (Fast & Wrong pattern)
`V_CARELESSNESS_DETAIL_OMISSION` V Behavior: Carelessness - Detail omission/misread (Fast & Wrong pattern)
`BEHAVIOR_CARELESSNESS_ISSUE` Behavior Pattern: Carelessness issue (overall high fast-wrong rate $>$ 25%)
`BEHAVIOR_EARLY_RUSHING_FLAG_RISK` Behavior Pattern: Rushing early in section ($<$ 1.0 min, flagging risk)
`BEHAVIOR_PATTERN_FAST_GUESSING_HASTY` Behavior Pattern: Hasty response/fast guessing pattern
`DI_MSR_GROUP_TIME_PRESSURE_BEHAVIOR` DI MSR Behavior: Time pressure affecting group performance