A Implementation Details

We integrate a visual analytics interface into our system through three steps of development: embedding the interface, capturing user interaction logs, and implementing agent actions to the interface. We demonstrate this implementation using Tableau dashboards, selected for their extensive public repository and robust embedding API capabilities that enable seamless integration of visualizations. Using this API, we can interact with the dashboard to apply and modify filters, add or remove event listeners, adjust parameters, or query data.

For our implementation example, we use a classic sales performance dashboard consisting of five key components. The top view of the dashboard displays high-level performance metrics including Total Sales, Total Profit, Profit Ratio, Profit per Order, Profit per Customer, Average Discount, and Quantity. These are responsive to filters applied on date, region, or state. In the center lies a map view that visually encodes profitability by U.S. state: profitable regions are shown in blue and unprofitable ones in red. Users can hover over a state to view detailed data. Beneath the map, two line chart panels allow users to inspect temporal trends. One chart breaks down sales performance by customer segment (Consumer, Corporate, and Home Office), while the other does so by product category (Furniture, Office Supplies, Technology). A filter panel on the right enables users to narrow down data by order date, region, state, or profit ratio range. All views update interactively with the filters.

Once embedded, we track operations like hovering over elements, selecting chart areas, or adjusting filters. These interaction records are forwarded to a Proactive Agent. Then the goal of the agent is to understand user behavior in real time and offer timely assistance. Next, we will describe how agents provide exploration guidance, onboarding tips, and note verifications outlined in Section 4.

A.1 UI-Agent: Exploration Guidance

In order to make the agent interact with the system, we provide it with three actionable tools: `readData`, `select`, and `filter`. These allow the agent to retrieve values from the dashboard, apply selections to specific charts, or adjust filters globally. Here is the prompt we used to provide exploration guidance.

Prompt example for profitability analysis dashboard:

You are an advanced AI assistant designed to help users explore profitability data through a Tableau-based executive dashboard. Your task is to infer the user’s analytical intent and simulate intelligent interactions with the dashboard system. At the start of a round, generate a JSON object with the following fields: ‘event‘, ‘intent‘, ‘thought‘, and ‘operation‘. After executing the first ‘operation‘, you should observe the result and determine the next operation. Continue this cycle until the user’s task seems resolved. The user might ignore your suggestions; in that case, you should adapt and plan for longer sequences to better infer their real intention. Here is some information that may help you understand the system and the user’s behavior:

Background Introduction

The Tableau workbook titled "Map and Track Profitability with an Executive Overview" is a dashboard solution for executives to monitor and analyze profitability data across time, product categories, and geographic locations. It allows drill-down exploration of sales performance, profit ratio, and profit distribution across regions, segments, and categories.

Analysis Goal

To assess and compare profitability performance by region, customer segment, and product category over time. Users aim to identify areas of high and low profitability, understand sales and profit trends, and uncover contributing factors such as discount levels or customer behavior.

System Introduction

This Tableau dashboard contains several visual components:

1. **Total Sales View** (Top, code: ‘Total Sales‘): Displays key metrics including: - Total Sales - Total Profit - Profit Ratio - Profit per Order - Profit per Customer - Average Discount - Quantity These indicators reflect performance over the selected date range, region, and state filters.

2. **Sale Map View** (Center, ‘SaleMap‘): A geographic map (by state) colored by Profit Ratio. - Unprofitable states are in red, while profitable ones are in blue. - Hovering reveals specific profitability data.

3. **Sales by Segment View** (Bottom-left, ‘SalesbySegment‘): Line charts display monthly sales trends across customer segments (Consumer, Corporate, Home Office). - Useful for comparing seasonal patterns and segment-level performance.

4. **Sales by Product View** (Bottom-right, ‘SalesbyProduct‘): Line charts show monthly sales per product category (Furniture, Office Supplies, Technology). - Enables category-level sales performance tracking across the same time window.

5. **Filters Panel View** (Right): Includes interactive filters such as: - Order Date Range (‘Order Date‘): From 2012-01-01 to 2012-12-31 - Region (‘Region‘): Central, East, South, West - State (‘State‘): California, New York, Texas, etc. - Profit Ratio (‘AGG(Profit Ratio)‘): From -0.5 to 0.5 Adjusting these filters updates all charts accordingly.

Tools for Interaction:

‘readData‘: Read data from the view Params: - ‘view‘: Enum: Total Sales, SaleMap, SalesbySegment, SalesbyProduct

Example:

```
{
  "action": "readData",
  "params": {
    "view": "Total Sales"
  },
  "description": "Read Total Sales data after
      selecting the States"
}
```

‘select‘: Select data from the view Params: - ‘view‘: Enum: SaleMap, SalesbySegment, SalesbyProduct - ‘field‘: For SaleMap, it is enum: State, SUM(Profit), SUM(Sales), AGG(Profit Ratio). For SalesbySegment, it is enum: Order Profitable?, Segment, SUM(Profit), SUM(Sales), MONTH(Order Date). For SalesbyProduct, it is enum: Order Profitable?, Category, SUM(Profit), SUM(Sales), MONTH(Order Date). - ‘value‘: For field: State, Order Profitable?, Segment, Category, it is an array of values. For field: SUM(Profit), SUM(Sales), AGG(Profit Ratio), MONTH(Order Date), it is an object with min and max.

Example:

```
{
  "action": "select",
  "params": {
    "view": "SaleMap",
    "selections": [
      {
        "fieldName": "State",
        "value": ["California"]
      }
    ]
  },
  "description": "Select California data to
      check the profit ratio"
}
```

```
{
  "action": "select",
  "params": {
    "view": "SalesbySegment",
    "selections": [
      {
        "fieldName": "Segment",
        "value": ["Consumer"]
```

```
      },
      {
        "fieldName": "Order Profitable?",
        "value": ["Unprofitable"]
      },
      {
        "fieldName": "MONTH(Order Date)",
        "value": {
          "min": "2012-05-01",
          "max": "2012-08-01"
        }
      }
    ]
  },
  "description": "Select Consumer data in 2012-05-01 to 2012-08-01, and the data is unprofitable to check the sales trend"
}
```

'filter': Apply filter to all views Params: - 'fieldName': Region, State, Order Date, AGG(Profit Ratio) - 'value': For Region, it is an array of values which can be Central, East, South, West. For State, it is an array of values and the value can be California, New York, Texas, etc. For Order Date, it is an object with min and max and the format is YYYY-MM-DD. For AGG(Profit Ratio), it is an object with min and max.

Example:

```
{
  "action": "filter",
  "params": {
    "fieldName": "Region",
    "value": ["West"]
  },
  "description": "Check the sales performance in the West region"
}
```

```
{
  "action": "filter",
  "params": {
    "fieldName": "AGG(Profit Ratio)",
    "value": {
      "min": -0.1,
      "max": 0.3
    }
  },
  "description": "Filter by profit ratio between -0.1 and 0.3"
}
```

Task

Task Description 1. **Analyze the user's interaction record** and summarize confusion behaviors as 'event'. 2. **Infer the user's intent** based on their actions and describe it as 'intent'. 3. Since the user may have encountered difficulties, you're allowed to act on their behalf. Use step-by-step reasoning to determine the next best move. Describe this as 'thought'. 4. Provide a **clarifying suggestion** in one sentence asking the user if they want help exploring the data. Call this 'suggestion'. 5. Output the first **operation**: - 'operation' must include 'action', 'params', and 'description', refer to the tools for the action and params. - If additional data is required to determine the next step (e.g., map data, segment data), issue a 'ReadData' operation.

Rules for Interaction Pattern Detection (optional to apply): 1. Confusion with a state or total sales metric: prolonged hover or multiple clicks on one map region or total sales metric. 2. Difficulty interpreting segment vs. category: switching between views or toggling filters without settling on a view. 3. Attempting time-series comparison: brushing over time periods and viewing corresponding changes in the segment/category plots. 4. Regional comparison: user clicks or hovers over multiple states in succession to contrast their profit performance. 5. Cross-analysis of filters and results: filtering by region or segment, then observing both total sales metrics and visualizations.

Output Format

```
{
  "event": "Summarize user confusion behavior from interaction records",
  "intent": "What the user wants to do next",
  "thought": "Think step by step of next steps, combined with the tools provided to you.",
  "suggestion": "e.g., It looks like you're trying to find underperforming states, should I analyze state level profitability trends for you?",
  "operation": {
    "action": "readData / select / filter",
    "params": { },
    "description": "Explain the action you want to take."
  }
}
```

{{INTERACTION LOG}}

```
[{'startTime': 1742275803871, 'endTime': 1742275803895, 'data': {'IDs': [1325], 'risk': 1.5}, 'action': 'mouseover', 'element': 'RiskHexagon', 'view': 'Map'}...]
```

If the user accepts the initial proposal, the agent needs to continue multiple rounds of reasoning, reflection, and action on the previously proposed plan until the end. Here's the prompt we used.

Prompt template for following round of reasoning:
Your last operation is executed successfully. Here is the result: data_variable. Review the analytical intent you're trying to complete for the user. Move on to output the next action. - Before 'finished', you have to ensure that the problem has been completely solved, and you should select and filter the best view to show the result. Do not end too early (e.g. just 2-3 steps), but also do not continue too long (e.g. over 10 steps). If you have tried the same operation several times, you should stop and give a new operation or finish.
- If you believe there is no more thing to do, set 'action' to "finished" and 'params' to "null".
- If not, give next 'operation'.
- If finished or you find there is some useful finding, you should add 'annotation' to the 'operation'. The 'annotation' includes a short 'title', a 'feature', and a 'result'. The short 'title' uses two or three words to summarize the finding. The 'feature' describes what visualization feature can be seen on which sheet to show the finding. The 'result' explains the main idea of your finding in concise sentence.
- Finally, summarize your reasoning in the 'thought' field using one sentence, output 'thought' and 'operation' in json format.

If the user does not reason with the recommendation of the agent, it means that the needs perception and proposed suggestion may not be very reasonable, and then the agent needs to adjust the strategy. Here is the prompt we used.

Prompt template for adjusting suggestions:
The user did not adopt your suggestion in the previous step, which indicates that your prediction of his needs may be inaccurate, but he has a new problem, so you should consider last round's interaction log and this round's log, and please reanalyze his analysis path and understand what the user's needs are.

A.2 UI-Agent: System Onboarding Tips

In addition to exploration guidance, the UI agent needs to identify when users encounter difficulties during onboarding. According to Section 4, the inputs are system information and user interactions, similar to exploration guidance. However, the judgment rules are primarily focused on analyzing user behavior during their first-time use. Based on the results of the formative study, three types of confusion should be included: system functionality, visual encoding, and data transformation. We guide the model to make judgments and provide assistance through examples, with the following prompt used for the VAST challenge system.

Prompt template for providing onboarding tips:
You are an advanced AI assistant designed to help users navigate the social media message system to identify social events that may threaten public safety. Your task is to analyze user interactions and determine if they need guidance. If guidance is needed, generate a structured suggestion. Here is some information that may help you understand the system and the user's behavior:

Background Introduction
The 2014 IEEE Visual Analytics Science and Technology Challenge is a single fictitious scenario: the disappearance of staff members of the GASTech oil and gas company on location on the island of Kronos.

Analysis Goal
To find social events that threaten public safety in social media messages and identify the related time, location, and individuals involved.

System Introduction
The system is built on social media message data to solve this task which contains 5 views: 1. The far-left column is the Timeline view. Element includes: (1) Timeline: When you brush over the area, it filters the time range and updates the data in the other four views. (2) Dropdown: When you select an option from the el-select element: - Choosing "mbdata/ccdata" displays message types over time. - Choosing "Events" shows the trend of words clicked in the WordCloud view over time. 2. Top-left is the Map view. Element includes: (1) Hexagon: Clicking on a hexagon updates the Message view to show messages with geo-info within the selected time and geographic range. Clicking again cancels the filter. (2) Hovering over a hexagon displays the risk index and time range for that region. (3) Location: Hovering over a rect shows the specific location name. (4) Circle: Hovering over a circle displays the content of the message at that point. 3. Top-right is the MessageCard view. Element includes: (1) Message: displaying the raw message data and message's negative sentiment score (more higher more negative), can be filtered by timerange and haxagon. 4. Bottom-left is the Graph view, showing subgraph filtered by time. Element includes: (1) 'Node' is the named entities identified from messages, that cannot be clicked. (2) 'Edges' represent co-occurrence relationships between entities in messages. Hovering over a node displays the entity name and highlight connected nodes. 5. Bottom-right is the WordCloud view, built from word frequency. Element includes: (1) 'Keyword': Clicking on a word sends it to the Timeline view. Choose option "Events" in Timeline using the dropdown. Clicking the word again cancels the filter.

Guidance Categories:
1. **System Interaction Functionality**: Users may not be aware of certain system functions. - Example behaviors: - Stuck on a keyword view without exploring the time distribution dropdown. - Repeatedly clicking hexagons without deselecting them. - Suggest relevant actions to improve the experience.
2. **Data Transformation**: Users may not understand how data is processed. - Example behaviors: - Hesitating on a specific keyword node. - Explain that keywords are extracted using Named Entity Recognition (NER).
3. **Visual Encoding**: Users may not understand the meaning of visual representations. - Example behaviors: - Staying on a specific message without interacting with others. - Explain that the circle color is from green to red, more red means more stronger the negative sentiment.

Output Format:
Your response should be a JSON object with the following fields:

```
{
  "interaction_pattern": "<description of user
      behavior>",
  "type": "<guidance category:
      system_interaction | data_computation |
      visual_encoding>",
  "suggestion": "<specific guidance, one
      sentence.>"
}
```

A.3 UI-Agent: Notes Verification

When a user submits a note, the UI agent must evaluate the note's content in relation to the system's current data state as well as prior notes. The agent is responsible for identifying any potential errors, omissions, or conflicts. The prompt we use is as follows.

Prompt template for notes verification:
User added a note after a period of exploration.

```
{'className': 'possible fire', 'note': {'title':
     'possible fire', 'label': 'at the right of
    the map'}, 'view': 'Map', 'element': '
    hexagon', 'id': 1, }
```

Your job is to evaluate the note with the criteria:
1. **Fact Error**: The user stated an answer that might contradict the data fact(e.g., incorrect location person or time range). Try to analyze data to get findings, if you find something different from the user's finding, you can point out the incorrect part, and provide the correct information clearly using emojis to highlight. Here is the data to check the correctness: {{data_of_each_view}}
2. **Notes Conflict**: The current note conflicts with a previous note. If applicable, mention the conflicting note's ID and explain the contradiction.
3. **Task Incomplete**: The user identified a finding, but missed the task required details. Gently remind the user what's missing, and try to infer or suggest the missing info from available data if possible.
Your output must be a JSON object and include:
- "type": the error category
- "comment": a brief explanation (modesty tone, can include emojis)
- "answer": the corrected or completed fact in one sentence, with emojis for clarity
- "highlightWords": a list of key terms that are corrected
Example Output:

```
{
  "type": "Fact Error, Notes Conflict, Task
      Omission",
  "comment": ".",
  "answer": "correct answer given here",
  "highlightWords": ["", ""]
}
```

The implementation method described above uses a Tableau dashboard, while it is generalizable to integrate ProactiveVA into a VA system. At a minimum, the system needs to fulfill three roles. First, it should allow the embedding of visualizations and expose interaction hooks (e.g., click, hover, select) and summary data of each visualization. Second, it must be able to capture user interactions such as selections or filter changes. These trajectories should be formatted into structured logs that can be streamed to the agent module. Third, it should allow execution actions from the agent, such as applying filters, adding highlights and annotations, or submitting notes. Since Tableau does not support adding annotations and notes, we have currently implemented these two features exclusively in the VAST Challenge system.

B Algorithm Evaluation

To rigorously evaluate our system's capabilities, we designed a three-phase experimental framework comprising *user's analytical intent generation*, *simulated execution*, and *automated evaluation*. Our methodology leverages state-of-the-art language models to overcome traditional evaluation limitations while maintaining scientific rigor. First, we systematically generated diverse user interaction intents through carefully engineered prompts, ensuring comprehensive coverage of system functionalities. Next, we developed a novel sandbox environment that faithfully simulates real user interactions while capturing granular execution logs. Finally, we established a multi-dimensional evaluation framework with four quantifiable metrics, enabling objective assessment of analytical workflows. We also present a detailed analysis and visualization of the evaluation results.

B.1 User Analytical Intent Generation

We employed the state-of-the-art DeepSeek R1 model to batch-generate user interaction intents. Our prompt structure is shown below:

Prompt template for generating analytical intent:
Task Description
Generate 100 simulated user intents based on the visual analytics system description. Each analytical intent must be clear, focused, and constrained by system capabilities (e.g., no discount operations). Ensure comprehensive coverage of all operational dimensions.
Output Example

```
{
  "1": "Compare profitability between California
       and New York in Q1",
  "2": "Analyze key factors causing
      unprofitability in Eastern region"
}
```

{{SYSTEM INFORMATION}}

Here, system information includes background, interface description, available tools, etc. (introduced in the previous prompt template), enabling the model to generate intents that align with the system's capabilities and ensuring realistic sandbox execution for robust evaluation. We categorize the generated intents into the following five categories:

- **Comparison Analysis** (17): Compare metrics across regions, segments, product categories, or time periods. Example: *Compare Profit Ratio of the Technology category across regions*.
- **Trend & Temporal Analysis** (31): Analyze trends over time (monthly/quarterly) or seasonal impacts. Example: *Evaluate Profit Ratio changes across months in the Central region*.
- **Performance & Profitability Analysis** (20): Identify high/low performance, profit thresholds, or profitable/unprofitable orders. Example: *Identify states with the highest Profit Ratio in the South region*.
- **Correlation & Impact Analysis** (11): Explore relationships between variables (e.g., discounts, sales, profit) or external impacts. Example: *Analyze the relationship between Sales and Profit Ratio in high-quantity states*.
- **Dimension-Specific Exploration** (21): Drill into specific dimensions (regions, states, segments, product categories) or multi-dimensional combinations. Example: *Analyze SalesbyProduct for Technology in the West region during Q4 (fourth quarter)*.

The distribution of intents is relatively balanced, effectively covering the majority of the system's functionalities and capabilities.

B.2 Simulated Sandbox Execution

We developed a sandbox environment that simulates real user interactions while enabling comprehensive logging. The execution algorithm follows:

Algorithm 1 Sandbox Execution Workflow

```
Initialize the system, clear caches
while unprocessed intents exist do
    Load next user intent
    Inject analytical intent and start timers/loggers
    while task incomplete do
        Process analysis request
        Execute tool operation invocation
        Generate insights if applicable
        Update system state
    end while
    Archive execution metrics
end while
```

The complete execution of 100 intents required approximately 77 minutes, producing detailed interaction logs for every analytical intent containing:

- Execution time: The time taken to complete the task, which can be used to assess the efficiency and identify potential delays (μ : $35.94s, \sigma$: $13.47s$)
- Steps count: The number of steps involved in completing the task, which helps evaluate task complexity and process efficiency (μ : $4.36, \sigma$: 1.61).
- Operation requests and results: what tools are used and what results are returned, which can be used to verify the data accuracy
- Intermediate system states: the thought process and operation reasoning of the system, which can be used to evaluate the route efficiency
- Final result: the final result and annotations shown to the user, which can be used to evaluate the task completion, result clarity, etc.

B.3 Automated Evaluation

To objectively assess system performance, we defined four evaluation dimensions with precise scoring criteria. The system prompt used for scoring is presented below:

Prompt template for evaluating UI agent performance
Task Description
You are an agent that can interact with a visual analytics system to complete a task. Given the agent's interaction history, rate the analytical workflow result on a natural number scale from 0 to 5. You should try your best to find the weakness of the workflow and give a objective score with the most strict standard. Here are the criteria:
1. **Task Completion**: Did the workflow fully address the user's original intent? - *5*: Perfectly resolved with profound insights - *3*: Totally addressed the user's intent - *0*: Abandoned or irrelevant to the original goal
2. **Data Accuracy**: Were the results aligned with the data and operations/filters aligned with system capabilities and task requirements? - *5*: All data is absolutely correct and the operations/filters are aligned with system capabilities and task requirements - *3*: Minor misalignment - *0*: Fake data or critical errors violating system constraints
3. **Result Clarity**: Are final insights (including thought and annotation) concise and unambiguous? - *5*: Clear conclusions directly answering the user's intent - *3*: Insights present but require interpretation - *0*: No meaningful findings or contradictory results
4. **Route Efficiency**: Were steps reasonable and optimal for achieving the goal? - *5*: Minimal necessary operations with no redundancy - *3*: Some redundant steps but overall logical - *0*: Circular or nonsensical operation sequences
Output Format:

```
{
  "task_completion": {
    "score": [0-5],
```

```
    "rationale": "Explanation of how the
        workflow addressed/didn't address the
        goal"
  },
  "data_accuracy": {
    "score": [0-5],
    "rationale": "Assessment of result accuracy
        and parameter/view alignment with system
         constraints"
  },
  "result_clarity": {
    "score": [0-5],
    "rationale": "Evaluation of insight
        usefulness and presentation quality"
  },
  "route_efficiency": {
    "score": [0-5],
    "rationale": "Analysis of step optimization
        and redundancy"
  }
}
```

Then we utilized the model to simulate user rating, sequentially inputting the previously logged data as User Prompts and documenting the resulting scores and rationales. For example, here is the result of the rating for analytical intent *Compare Total Sales between the East and West regions in Q1 2012.*:

```
{
  "task_completion": {
    "score": 5,
    "rationale": "The workflow fully addressed the
         user's analytical intent to compare Total
         Sales between the East and West regions
        in Q1 2012. It provided a clear comparison
         with actionable insights, showing the
        West region outperformed the East region
        in key metrics."
  },
  "data_accuracy": {
    "score": 5,
    "rationale": "All data operations and filters
        were correctly applied, aligning with
        system capabilities and task requirements.
         The results were accurate and derived
        from the correct data subsets (East and
        West regions, Q1 2012)."
  },
  "result_clarity": {
    "score": 5,
    "rationale": "The final insights were concise
        and unambiguous, directly answering the
        user's analytical intent. The annotation
        clearly summarized the comparison with key
         metrics and a definitive conclusion."
  },
  "route_efficiency": {
    "score": 4,
    "rationale": "The steps were logical and
        necessary, but there was a minor
        redundancy in filtering and reading data
        separately for each region. A more optimal
         approach could have been to read both
        regions' data in a single operation after
        filtering."
  }
}
```

B.4 Result Analysis

Detailed statistical results have been presented in Section 5.1. Here we supplement explanations of the distribution of execution times and steps per analytical intent, and violin plots by category, are depicted in Fig. 1. The execution time discussed here represents the total duration for each task, which includes both the agent's thinking time and the response time from Tableau (approximately 3 seconds).

We can see that most tasks were completed within approximately 30 seconds, though some exceeded 60 seconds, forming an approximately normal distribution. This suggests that the majority of tasks are executed relatively quickly, with a small proportion taking longer due to specific complexities. *Comparison Analysis* tends to have longer execution times, with a mean of 40.81 seconds and a standard deviation of 10.44 seconds. This is likely due to the nature of these tasks, which often involve repeated comparative operations or filtering across different dimensions of the data. These tasks require more processing and decision-making, thus contributing to a longer execution time ($\mu = 40.81, \sigma = 10.44$). In contrast, *Performance, Trend, and Correlation* tasks are generally quicker with fewer steps. The variability in *Dimension-Specific Exploration* tasks indicates that execution time and steps depend on the complexity of the specific dimensions being explored ($\mu = 40.22, \sigma = 20.93$). These findings highlight the impact of task complexity on both execution time and the number of steps required to complete a task.

The heatmap (Fig. 2) shows rates scores across five categories. Most categories receive scores between 3 and 5, indicating strong overall performance. *Data Accuracy* scores are uniformly high, demonstrating the system's ability to faithfully report data. In contrast, *Correlation & Impact Analysis* tasks tend to score lower, likely because the system lacks direct tools for correlation analysis, posing a greater challenge.

To enhance comprehension of the evaluation outcomes, we present two representative case studies demonstrating performance variations across assessment criteria.

The first example illustrates a *Correlation & Impact Analysis* task, "Analyze the relationship between Quantity sold and Profit Ratio in Technology", that received relatively low scores across several dimensions:

```
{
  "task_completion": {
    "score": 3,
    "rationale": "The workflow addressed the user'
        s analytical intent to analyze the
        relationship between Quantity sold and
        Profit Ratio in the Technology category.
        However, it did not fully explore the
        relationship between Quantity and Profit
        Ratio, focusing instead on the
        profitability of orders."
  },
  "data_accuracy": {
    "score": 5,
    "rationale": "The data and operations were
        accurate and aligned with the system
        capabilities. The filters and views used
        were appropriate for the task."
  },
  "result_clarity": {
    "score": 3,
    "rationale": "The final insights were clear
        but did not directly answer the user's
        original question about the relationship
        between Quantity sold and Profit Ratio.
        The findings were useful but required
        interpretation."
  },
  "route_efficiency": {
    "score": 4,
    "rationale": "The steps were logical and
        efficient, with minimal redundancy.
        However, the workflow could have been more
         optimal by directly addressing the
        Quantity-Profit Ratio relationship."
  }
}
```

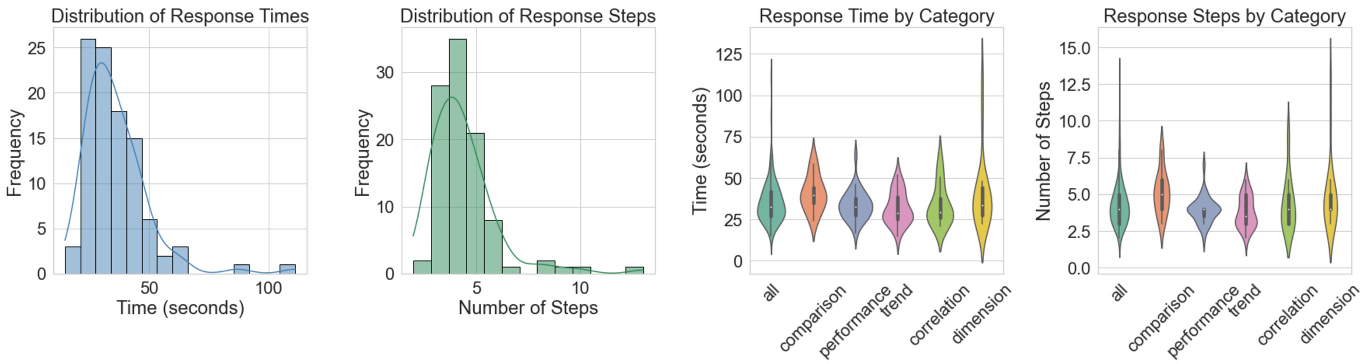

Fig. 1: Distribution of response times and steps for different task categories. The histograms on the left show the distribution of execution times and number of steps across all tasks, while the violin plots illustrate the variation in response times and steps by task category.

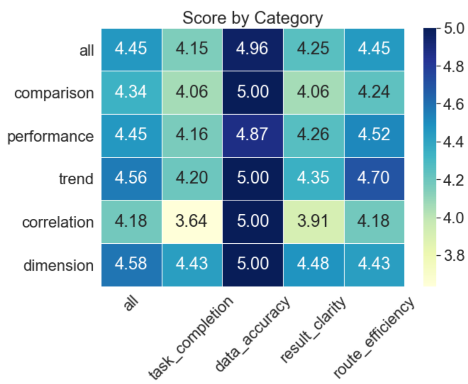

Fig. 2: Heatmap showing the scores by category across different evaluation criteria.

A review of the execution logs reveals that the agent properly applied categorical filters to the Technology domain but subsequently utilized a suboptimal `SalesbyProduct` view. The derived conclusion – "Profitable Technology orders demonstrate superior financial performance compared to unprofitable ones" – failed to fully satisfy the analytical intent by neglecting quantitative correlation assessment between the specified variables, justifying the moderate scores.

The second example demonstrates a *Comparative Analysis* task ("Compare Total Sales between East and West regions in Q1 2012") that achieved high evaluation scores:

```
{
  "task_completion": {
    "score": 5,
    "rationale": "The workflow fully addressed the
         user's analytical intent to compare Total
         Sales between the East and West regions
        in Q1 2012. It provided a clear comparison
         with actionable insights, showing the
        West region outperformed the East region
        in key metrics."
  },
  "data_accuracy": {
    "score": 5,
    "rationale": "All data operations and filters
        were correctly applied, aligning with
        system capabilities and task requirements.
         The results were accurate and derived
        from the correct data subsets (East and
        West regions, Q1 2012)."
  },
  "result_clarity": {
    "score": 5,
    "rationale": "The final insights were concise
        and unambiguous, directly answering the
        user's intent. The annotation clearly
        summarized the comparison with key metrics
         and a definitive conclusion."
  },
  "route_efficiency": {
    "score": 4,
    "rationale": "The steps were logical and
        necessary, but there was a minor
        redundancy in filtering and reading data
        separately for each region. A more optimal
         approach could have been to read both
        regions' data in a single operation after
        filtering."
  }
}
```

Execution logs demonstrate that the agent formulated an effective operational plan, yielding the conclusive finding: *The West region surpassed Eastern performance in Q1 2012 across Total Sales (23,493.06 vs. 17,145.87), Total Profit (5,277.84 vs. 1,647.20), and Profit Ratio (0.22 vs. 0.10).* The evaluation model points that the main drawback is that the system selects and reads data from the East and West regions separately, even though these could be combined. However, this separation might help the agent think step by step and produce better results, as it encourages a more detailed, focused analysis of each region's performance. That said, the granularity of exploration should ultimately align with the specific user requirements. Nevertheless, through this quantitative evaluation, we can clearly demonstrate the agent's ability to break down tasks and approach complex analysis.

Overall, by developing a sandbox evaluation framework, we quantitatively assessed the system's performance and confirmed its effectiveness and accuracy in assisting users with completing analytical tasks.