FINCH: Benchmarking Finance & Accounting across Spreadsheet-Centric Enterprise Workflows

Haoyu Dong[1,2*] **Pengkun Zhang**[3] **Yan Gao**[5] **Xuanyu Dong**[6] **Yilin Cheng**[4,7]
Mingzhe Lu[1] **Adina Yakefu**[8] **Shuxin Zheng**[4,5]

[1]University of Chinese Academy of Sciences [2]Microsoft [3]South China University of Technology
[4]Zhongguancun Academy [5]Zhongguancun Institute of Artificial Intelligence [6]Harvest Fund
[7]Fudan University [8]Hugging Face

Abstract

We introduce FinWorkBench (a.k.a. FINCH) for evaluating AI agents on real-world, enterprise-grade finance and accounting workflows that interleave data entry, structuring, formatting, web search, cross-file retrieval, calculation, modeling, validation, translation, visualization, and reporting. FINCH is sourced from authentic enterprise workspaces from Enron (15,000 files and 500,000 emails) and other financial institutions, covering the period 2000–2025, preserving in-the-wild messiness of multimodal artifacts such as tables and charts, across diverse domains such as budgeting, trading, asset management, and operational management.

We propose a workflow construction process that combines LLM-assisted mining of workflows from authentic enterprise environments with expert annotation: (1) LLM-assisted, expert-verified derivation of workflows from real-world email threads and spreadsheet version histories, and (2) meticulous annotation requiring over 700 hours of expert effort. This yields 172 composite workflows with 384 tasks, involving 1,710 spreadsheets with 27 million cells, along with PDFs and other artifacts, capturing the intrinsically messy, long-horizon, knowledge-intensive, and collaborative nature of real-world enterprise work.

We conduct both human and automated evaluations of frontier AI systems, including GPT 5.1, Claude Sonnet/Opus 4.5, Gemini 3 Pro, Grok 4, and Qwen 3 Max. Under human evaluation, GPT 5.1 Pro spends an average of 16.8 minutes per workflow yet passes only 38.4% of workflows. Comprehensive case studies further surface the challenges that real-world enterprise workflows pose for AI agents.

* Corresponding to donghaoyu82@gmail.com. Author list: Appendix A.

1 Introduction

Frontier AI systems are increasingly transforming professional workspaces. AI-assisted tools like Claude (Anthropic, 2025a), OpenClaw (OpenClaw, 2025), ChatGPT (OpenAI, 2025), Gemini (Google, 2024), and Copilot (Microsoft, 2024) are now embedded in daily enterprise workflows—helping professionals draft documents, explore data, manipulate spreadsheets, and generate reports. They are particularly impactful in finance and accounting (F&A), a high-stakes, knowledge- and labor-intensive domain critical to every organization.

However, real-world F&A work is inherently **messy** with substantial contextual complexity: artifacts are interconnected across heterogeneous spreadsheets, PDFs, and more, evolving through multiple versions with collaborative edits (Klimt and Yang, 2004); spreadsheets contain large, complex structures (Dong et al., 2024) with cross-sheet references, intricate layouts, inconsistent formatting, cryptic terms, erroneous formulas, and multimodal artifacts such as charts, images, and code. It is also **long-horizon** (Patwardhan et al., 2025): workflows demand multi-step reasoning spanning data entry, editing, retrieval, calculation, modeling, validation, reporting, and more.

This raises a key question: *Can today's frontier AI agents actually handle the messy, long-horizon, and knowledge-intensive workflows that professionals face daily?*

To answer this, we introduce FINCH, a F&A benchmark sourced from authentic enterprise environments. FINCH captures the intrinsic complexity of professional work through:

- **In-the-wild enterprise sourcing**: FINCH is built around authentic enterprise spreadsheets, emails, and PDFs from real-world enterprise workspaces—primarily Enron (En-

Findings of the Association for Computational Linguistics: ACL 2026, pages 10768–10794

ronData.org) (about 15,000 spreadsheet files and 500,000 emails from 150 employees) and EUSES (Fisher and Rothermel, 2005) (about 450 financial spreadsheet files from various sources), along with securities and asset management firms, global organizations such as World Bank (Bank, 2024), and Canadian and British governments (Department of Finance Canada, 2025; HM Treasury, 2023). Documents are large, cross-referenced, and messy—containing rich multimodal artifacts such as tables, formulas, charts, pivots, and images.

- **Rigorous construction process**: We propose a novel workflow construction pipeline grounded in the real collaborative context of emails and versioned artifacts. We induce workflows from enterprise email threads and attachments, where collaborators naturally describe, discuss, and track workflows as part of their daily work. We further use an LLM-assisted, expert-verified method to derive workflows by analyzing changes across versioned spreadsheets, surfacing the underlying goals that drive professionals' work. Annotators must reason over large multi-sheet workbooks and subtle version deltas to infer underlying workflows, making the annotation process substantially more difficult than curating QA pairs over isolated tables.

We compile 172 meticulously annotated, enterprise-grade workflows built on 1,710 spreadsheets, along with PDFs and other artifacts, collectively capturing the compositional, messy, knowledge-intensive, and collaborative nature of real work. Each workflow spans one or more interdependent tasks—data entry, editing, retrieval, calculation, modeling, validation, translation, visualization, and reporting—mirroring how professionals actually work on artifact manipulation and creation. These workflows cover a broad set of enterprise domains such as trading and risk management, planning and budgeting, pricing and valuation, and asset management.

Evaluating such workflows poses nontrivial challenges, because FINCH tasks usually involve complex and large spreadsheets and require assessing both structural correctness and semantic fidelity beyond exact matching. To address this, we provide both expert human evaluation and a scalable LLM-as-judge pipeline. Human evaluation serves as the gold standard, judging whether a workflow has been satisfactorily completed end-to-end. In parallel, we introduce an efficient multimodal LLM-as-judge evaluation method that compares inputs, model outputs, and reference artifacts using structured diffs, compact snapshots, and multimodal renderings, enabling reliable assessment at scale.

We evaluate frontier AI systems including Claude Sonnet/Opus 4.5, GPT 5.1, Gemini 3, Grok 4, and Qwen 3 using both expert evaluation and automated pipeline. Even the strongest agents pass fewer than 50% of workflows: GPT 5.1 Pro spends 16.8 minutes per workflow on average yet achieves 38.4% under human evaluation.

Case analyses show that failure is driven by the combination of real-world complexity rather than any single bottleneck. GPT 5.1 Pro drops from 44.3% on workflows with $\leq$2 tasks to 23.5% on those with >2 tasks. Agents also struggle with irregular spreadsheet structure, latent business logic in formulas, and heterogeneous artifacts; on the 20 workflows involving PDFs, images, or Word documents, GPT 5.1 Pro achieves only 35.0%.

2 FINCH: A Real-world Finance & Accounting Workflow Benchmark

2.1 Dataset Construction

Echoing the existentialist dictum that existence precedes essence (Sartre, 1946), we argue that professional work should be observed in authentic enterprise environments before it is formally defined. Motivated by this, we propose a novel workflow construction pipeline grounded in authentic collaborative context of emails, versioned spreadsheets, and final deliverable spreadsheets and reports.

All workflows derived from these sources are consolidated into a unified schema with consistent fields (NL instruction, input files, reference outputs), and each workflow is tagged with task types (e.g., data entry/import, structuring, validation) and business types (e.g., planning and budgeting, pricing and valuation, operations, asset management). Note that the reference outputs may include both file-based reference answers (for most generation/editing cases) and textual reference answers (for a small number of QA and summary/visualization cases).

2.1.1 Workflow Derivation from Enterprise Email Threads

We first mine real-world enterprise email threads to surface workflows. Starting from the Enron Email

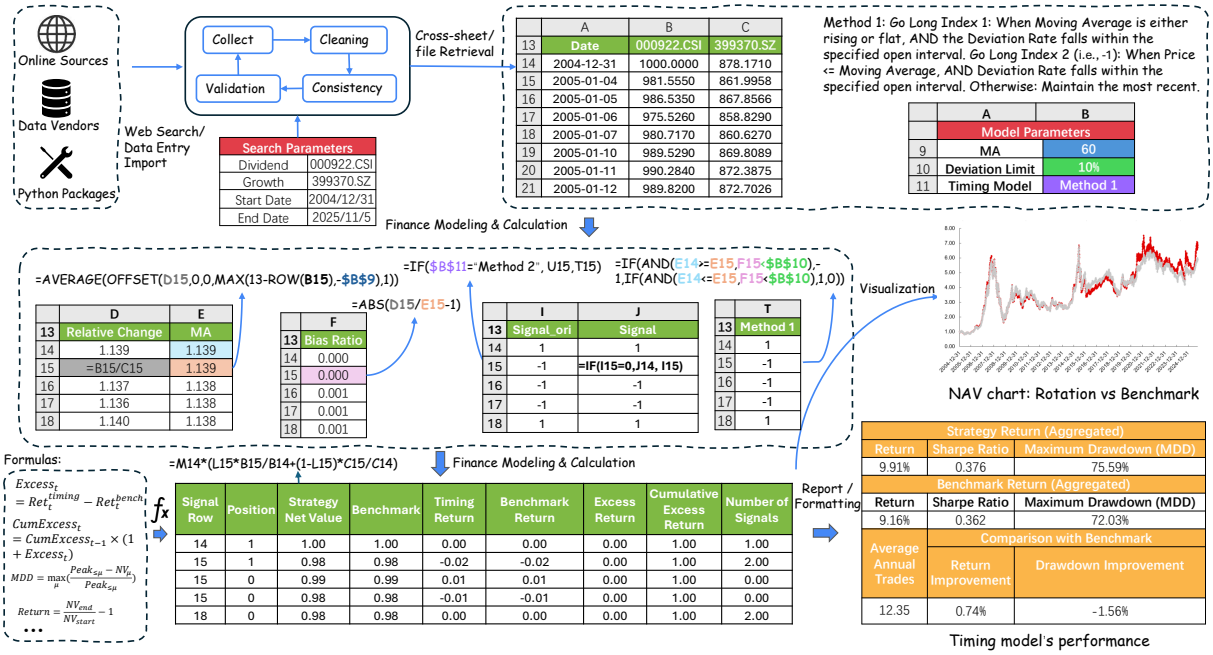


Figure 1: Illustration of an end-to-end predictive modeling workflow involving web search, data import, cross-sheet retrieval, calculation, and visualization. More illustrative examples in FINCH are presented in Appendix G.

Corpus, we prompt GPT-5 to identify collaborative messages that (i) explicitly state a business goal (e.g., "update the RAC rankings" or "revise the 2002 allocations") and (ii) reference one or more attached spreadsheets. For each selected thread, the model summarizes the communicative intent and articulates a workflow description.

As illustrated in Figure 2, in the *strongly grounded* case, both the input and the reference artifacts for a workflow are already present as attachments in the email thread (e.g., an initial ranking file and its corrected version). While strongly grounded cases are highly motivating for this benchmark, they are relatively rare. In the *partially grounded* case, the email specifies a clear goal, but only the input artifacts or the reference outputs are attached.

Across both cases, human experts normalize conversational email text and LLM-drafted descriptions into workflow instructions and abstract away idiosyncratic details while preserving the business intent. Annotators need to carefully verify and revise attachments to align with the instructions.

2.1.2 Workflow Derivation from Versioned Spreadsheets

Beyond explicit messages in email threads, we propose to discover workflows that are implicitly captured in spreadsheet version histories, as illustrated in Figure 2(c). We collect families of versioned workbooks from the Enron and EUSES repositories and apply an LLM-based differencing procedure

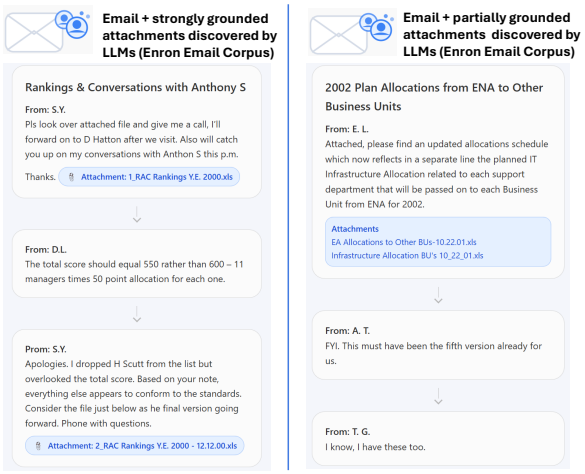


Figure 2: Illustration of real-world enterprise emails with attached artifacts.

that recognizes consecutive versions and infers the underlying workflow.

For each aligned pair of versions, we prompt GPT-5 to propose (i) one or more workflow types (e.g., "date-stamped versioning, assumption updates, and error correction", "data entry, structuring, and visualization") and (ii) a detailed NL description of all changes. Human experts then validate and refine these LLM-induced workflow candidates. They first determine whether the proposed diffs constitute a coherent and meaningful workflow rather than incidental churn. For accepted cases, they (i) rewrite the draft description into a precise task instruction that describes the transfor-

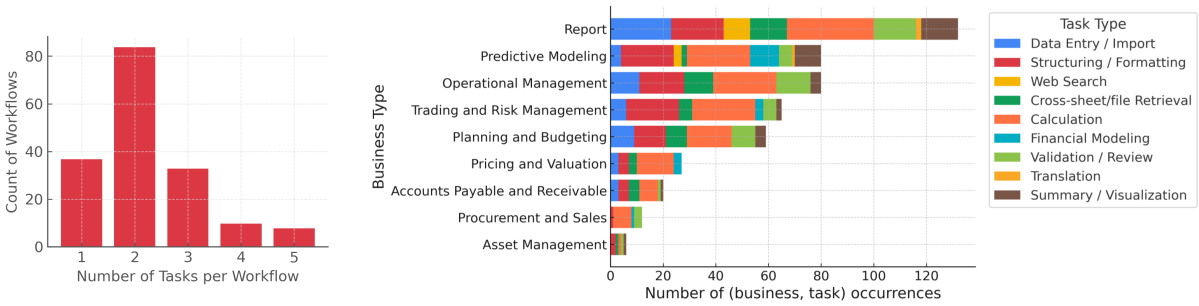


Figure 3: Distribution of number of tasks per workflow and task types across business types.

mation, and (ii) edit the corresponding workbook versions so that the input and reference files cleanly realize the described workflow without introducing unexpected out-of-scope changes.

2.1.3 Workflow Sourced from Final Deliverable Spreadsheets and Reports

Third, we curate workflows from high-quality artifacts drawn from the Enron and EUSES corpora, as well as from various investment and securities companies (e.g., Figure 1), international organizations, and national governments. Domain experts author realistic workflow instructions and construct input and reference files based on final deliverable artifacts. For example, a valuation model from an investment firm can be turned into a financial modeling task; a World Bank report can be used to define a summarization and visualization task; and bilingual reports from the Canadian government can be used to construct translation tasks. To broaden coverage of web search and multi-source QA, we adapt 10 financial cases from WideSearch (Wong et al., 2025) into web-search-centric workflows and extend them into multi-step calculation and visualization pipelines. We further leverage 3 examples from DABStep (Egg et al., 2025) to construct multi-source question answering workflows. We rely on WideSearch's annotations without further revision, as they are supported by extensive web-crawled information available for validation.

2.1.4 Quality Control

Given the high complexity of each workflow, we adopt a rigorous multi-stage quality control process. All workflows are annotated and reviewed by a team of five experts: two annotators with interdisciplinary backgrounds in finance and computer science, and three annotators with computer science backgrounds. Among them, three annotators (including one female) have over nine years of industry experience, and two are outstanding Ph.D. or master's students.

Annotators are instructed to skip workflows that are similar to existing ones to maximize diversity. Every workflow is validated by at least two independent annotators: about 40% of workflows undergo at least one round of revision, and particularly complex cases require multiple iterations. More details can be found in Appendix C. In addition, ChatGPT and Claude are used as secondary checkers to flag potential issues, which are always verified by human experts. Together, these procedures required over 700 hours of expert annotation.

2.2 Dataset Characteristics

FINCH comprises 172 meticulously annotated, enterprise-grade workflows that collectively capture the compositional, messy, multimodal, and collaborative nature of F&A work. All workflows are labeled exclusively for evaluation and are not intended for training. Across these workflows, the corpus contains 1,710 spreadsheets (956 distinct ones in 301 Excel files) together with 17 PDFs, 12 images, 3 Word documents, as well as JSON, CSV, and Markdown. 20 workflows involve file types beyond spreadsheets. This mixture reflects how real analysts coordinate over heterogeneous artifacts rather than clean, single-table inputs.

Figure 3 provides an overview of the task and business coverage in FINCH. This distribution highlights that FINCH targets realistic, core enterprise workflows rather than curated toy tasks. More details can be found in Appendix B.

2.2.1 Task Compositionality

FINCH is explicitly designed around composite workflows rather than isolated tasks. As shown in Figure 3, only 37 workflows (21.5%) are single-task; the remaining 135 (78.5%) involve multiple tasks. These tasks are not independent subtasks, but are interleaved around shared spreadsheets and files. A typical workflow may begin with structuring raw data, proceed to cross-sheet or cross-file retrieval, and then culminate in calculations, mod-

eling, or reporting. The distribution in Figure 3 shows that most workflows weave multiple tasks.

Importantly, each "task" itself typically requires substantial multi-turn reasoning: for example, web search often entails many rounds of LLM calls to discover, filter, and verify evidence; cross-sheet retrieval requires iterative calls to read and locate key information across multiple sheets; and calculation usually spans many formulas distributed over different rows and columns.

To further characterize the practical complexity of FINCH workflows, we conduct a case study on tool-call overhead using Claude Coworker (Opus 4.6) on 20 representative tasks (Appendix F). Excluding two extremely tool-intensive web-search workflows that require 71 and 107 tool calls respectively, the remaining tasks range from 6 to 25 tool calls (median 14). Notably, tool-call overhead does not scale linearly with the number of tasks; intrinsic task complexity—such as the depth of business-logic reasoning and cross-sheet dependencies—is a stronger driver. These findings suggest that even single- or two-task workflows can demand substantial multi-step agent interaction, reinforcing that FINCH captures meaningful operational complexity beyond what task counts alone convey.

2.2.2 Messiness

The source files in FINCH are large and interconnected. At the file level, 86.6% of workflows involve more than one file when counting both input and reference artifacts, and a workflow may touch up to 14 distinct files. At the spreadsheet level, 92.4% of workflows involve multiple input and reference sheets, with an average of 8 sheets and a long tail reaching up to 91 sheets. The median workflow covers 15K cells (157K on average), with the largest one scaling to 3.7 million cells.

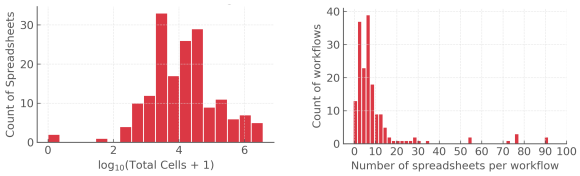


Figure 4: Distribution of the number of cells and sheets.

Moreover, most spreadsheets exhibit multimodal artifacts and intricate structures such as hierarchical headers, merged cells, and other irregularities. On average, a workflow involves 21.5K formulas (with a median of 212), reflecting deeply nested calculations and long dependency chains. In addition, 20.3% of workflows include charts, while 7.6% explicitly require reasoning over PDFs or images.

2.3 Evaluation Method

2.3.1 Human Evaluation

We conduct human evaluation on all workflows to directly assess model performance. For each workflow, annotators read the NL instruction and inspect the input, reference, and model output files side by side (typically by aligning spreadsheets or documents in adjacent tabs) to determine whether the model has faithfully completed the requested task. A workflow is marked as successful only if the model generates or revises content and structure in accordance with the instruction, without introducing critical errors, omissions, or unintended changes; otherwise, it is labeled as a failure.

Importantly, evaluation is based on whether the instruction has been satisfactorily fulfilled rather than on a purely mechanical comparison between the model and reference outputs, since multiple acceptable solutions may exist for summarization, visualization, formatting, structuring, formulas, and related aspects. To reduce subjectivity and ambiguity, annotators ultimately assign a binary pass/fail label to each workflow.

2.3.2 LLM-as-Judge Evaluation

To scale evaluation, we employ an LLM-as-judge framework that supports three file processing types: *modification* (editing input artifacts), *generation* (creating new workbooks or documents), and *QA* (answering questions based on one or more artifacts). The framework accepts heterogeneous and large inputs—including `.xlsx`, `.txt`, `.docx`, `.md`, `.pdf`, and images—and normalizes them efficiently into multimodal context for the judge model.

For spreadsheet modification tasks, instead of encoding full inputs and outputs separately—which would consume a large number of tokens when working with sizable artifacts—the framework computes *structured diffs* between the input and the reference output (`diff_ref`), as well as between the input and the model output (`diff_model`). In addition, it constructs a *compact input snapshot* (`snapshot`) that, for each modified sheet, retains only the first and last ten rows and the first five columns—typically capturing table headers and overall layout—along with all rows and columns containing modified cells. This design preserves the critical context required to interpret `diff_ref` and `diff_model` while substantially reducing to-

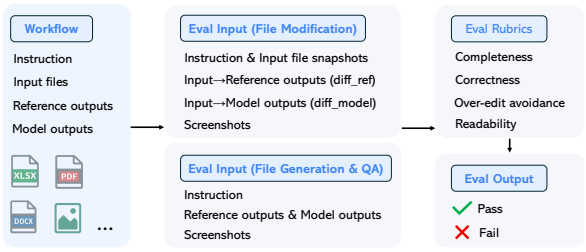


Figure 5: Illustration of automated evaluation pipeline.

ken usage. In parallel, the framework renders *screenshots* of sheets with changes from the input, reference, and model output, enabling the judge to perceive merged cells, conditional formatting, charts, and other layout-sensitive properties.

For generation tasks, the method extracts all text, cell values, and formulas from both the reference and the model output, and captures screenshots of every sheet/doc, since all contents should be verified rather than just local edits. For QA tasks, it feeds the reference answer and the model's response, augmented with relevant input artifacts when the question requires grounding.

Although different file processing strategies are used, they share common evaluation rubrics on (i) *completeness* with respect to the natural language instruction, (ii) *numerical and logical correctness* of derived values and formulas, (iii) *over-edit avoidance*, penalizing unnecessary or unexpected changes beyond the instruction, and (iv) readability of the formatting and structure. Exact cell-by-cell equality with the reference is not required when multiple solutions are acceptable (e.g., alternative layouts, equivalent formulas, or different but semantically equivalent summaries); instead, the judge determines whether the model has satisfactorily fulfilled the instruction. To reduce subjectivity, the judge outputs a binary score (pass/fail) along with a short natural language rationale for expert review and validation. For web search tasks, the rubric permits a 1% tolerance band, allowing for precision discrepancies across different data sources.

This LLM-as-judge framework not only automates large-scale evaluation but also surfaces subtle errors (such as formulas silently replaced with static values) that are difficult to catch with GUI-based human inspection alone. In Section 3.2, we report the consistency between human and automated evaluations and show that the LLM-as-judge scores closely align with human judgments. Looking forward, **agentic evaluation** is quite promising for future work for flexible workflow evaluation.

3 Experiments

3.1 Baselines

3.1.1 Product-side Agents

We evaluate two frontier product-side agents: (i) *ChatGPT* using GPT 5.1 in Pro mode, and (ii) *Claude* using Opus 4.5 or Sonnet 4.5 in Thinking mode. We focus on these two systems (rather than Gemini or Grok) because they natively return downloadable artifacts (e.g., spreadsheets) for human inspection, instead of emitting executable code or intermediate markdown tables. For both agents, we enable external web browsing but disable access to historical chats, so that each workflow is evaluated independently without cross-run leakage.

3.1.2 API-based Models

We evaluate five frontier LLMs via APIs (Table 7). We adopt SpreadsheetBench (Ma et al., 2024) as our baseline framework because it provides a principled code-generation paradigm for spreadsheets, enabling complex spreadsheet operations that cannot be reliably captured by text-only outputs.

Since SpreadsheetBench was originally designed for relatively small and clean spreadsheets, we extend it with richer encodings and multimodal input support to scale to the large and messy enterprise workflows in FINCH. We preserve its single-call setting to enable fair comparison across different LLMs without substantial framework changes. This setting may underestimate model performance, however, because it precludes iterative interaction, execution feedback, and self-correction mechanisms that product-side agents can leverage. Additional details are deferred to Appendix D.

Spreadsheet Encoding. SpreadsheetBench produces text tables without preserving cell addresses, data types, or formulas. We leverage SpreadsheetLLM (Dong et al., 2024)'s address-based encoding and introduce a *semantic-rich markdown encoding*. Each sheet begins with its name and the corresponding data range (e.g. `## Sheet: [name] (A1:Z100)`), using a Markdown-based format for serialization. Each cell is encoded as a tuple `(Address, Value, Type, Formula)`, where `Address` denotes the cell reference (e.g. `A3`), `Type` indicates the data type (`T` = Text, `I` = Integer, `F` = Float, `D` = Date, `B` = Boolean), and `Formula` records the cell formula (e.g., `=SUM(A1:A10)->100`).

Multimodal Input Handling. We extend the framework to support multimodal inputs involving images and PDFs, detailed in Appendix D.

Model	Pass Rate (%)
GPT-5.1 Pro (Product)	41.9
Claude Sonnet 4.5 (Product)	29.1
Claude Opus 4.5 (Product)	43.0
GPT-5.1 (API)	32.0
Claude Sonnet 4.5 (API)	20.3
Gemini 3 Pro Preview (API)	27.3
Grok 4 (API)	23.8
Qwen 3 Max (API)	14.5

Table 1: Automated LLM-as-judge evaluation results on FINCH workflows.

3.2 Experimental Results

Product-side agents. As shown in Table 1 (automated evaluation using GPT-5-mini as the judge) and Table 6 (human evaluation), ChatGPT 5.1 Pro and Claude Opus 4.5 achieve the strongest overall pass rates on FINCH. Their advantage largely comes from interactive affordances: they can iteratively inspect spreadsheets, revise intermediate states, and recover from partial errors through multiple tool calls. However, even these frontier agents solve fewer than 50% of workflows under human evaluation, suggesting that real-world finance and accounting work remains far from "solved."

The human evaluation results in Table 2 highlight **long-horizon composition** as a key bottleneck: when a workflow contains more than two tasks, the pass rate drops sharply—GPT 5.1 Pro falls from 44.3% for workflows with ≤ 2 tasks to 23.5% for workflows with > 2 tasks. We also examine how GPT 5.1 Pro's completion time scales with workflow length (Table 2). On average, single-task workflows take 13.1 minutes, while workflows with two and three tasks require 17.4 and 18.7 minutes, respectively, reflecting the increased compositional complexity of multi-task settings. Interestingly, workflows with 4 or more tasks show a lower average time (17.4 minutes) than workflows with 3 tasks, most of which involve web search and almost all of which result in failure. The longest individual run takes roughly 60 minutes yet still fails, underscoring the difficulty of highly compositional workflows for current agents.

Pass rates also vary substantially by **task type** (Table 3). Data Entry / Import and Structuring / Formatting are among the most challenging categories, consistent with the messy layouts, irregular tables, and multi-sheet structures in FINCH. Data Entry / Import is also frequently coupled with web search, a known difficult setting (Wong et al., 2025). Notably, Translation—where modern LLMs typically excel in standard NLP benchmarks—performs poorly in FINCH, as finance-heavy tables make it easy to distort or omit critical structural cues (e.g., header hierarchies and row/column alignment). Detailed error analysis is provided in Section 3.3.

Tasks per workflow	# Workflows	Pass (%)	Time (min)
1	37	48.6	13.1
2	84	42.4	17.4
3	33	33.3	18.7
$\geq$4	18	5.6	17.4

Table 2: GPT-5.1 Pro's pass rates and completion time by workflow composition (human evaluation).

Pass Rate by Task Type(%)	GPT-5.1 Pro	Sonnet 4.5
Data Entry / Import	25.0	13.6
Structuring / Formatting	29.1	25.6
Web Search	9.1	0.0
Cross-sheet/file Retrieval	35.1	13.5
Calculation	39.2	24.2
Validation / Review	37.8	21.6
Financial Modeling	33.3	20.0
Translation	0.0	0.0
Summary / Visualization	36.4	27.3

Table 3: Pass rates by task type for GPT-5.1 Pro and Claude Sonnet 4.5 under human evaluation. For composite workflows, we use workflow-level correctness: a task is counted as correct only if the entire workflow succeeds; if a workflow fails, all tasks it contains are counted as incorrect.

In addition, we examine performance on a subset of 20 workflows that involve **non-spreadsheet artifacts** (e.g., PDFs, Word documents, and images). As shown in Table 4, GPT 5.1 Pro achieves a 35.0% pass rate on these workflows (7/20), lower than its overall rate of 38.4%, while Claude Sonnet 4.5 achieves 25.0% (5/20), matching its overall rate. This suggests that handling heterogeneous document types introduces additional challenges.

Pass Rate	Overall	Other file types
GPT-5.1 Pro (Product)	38.4%	35.0%
Claude Sonnet 4.5 (Product)	25.0%	25.0%

Table 4: Pass rates on workflows involving non-spreadsheet artifacts (e.g., PDFs, Word documents, and images) under human evaluation.

Beyond binary pass/fail, our LLM-as-judge framework provides **rubric-level scores** along four dimensions: *completeness*, *correctness*, *over-edit avoidance*, and *readability*. Table 5 reports these scores for the two Claude product-side agents. Correctness is the primary bottleneck: Sonnet 4.5 achieves 55.8% completeness but only 36.0% correctness. These scores are produced by GPT-5-mini using structured output, where a single call assigns all four scores jointly; this may introduce inter-rubric confusion. More robust agentic evaluation methods are a worthwhile direction for future work.

Rubric (%)	**Sonnet 4.5**	**Opus 4.5**
Completeness	55.8	69.8
Correctness	36.0	45.3
Over-edit Avoidance	55.8	64.0
Readability	65.1	74.4

Table 5: Rubric-level scores in automated evaluation.

API-based models. Table 1 shows that API-based baselines are generally weaker than official product-side agents. Under automated evaluation, GPT 5.1 Pro reaches 41.9% pass rate, whereas GPT 5.1 with our API-based agent design reaches 32.0%. A key limitation of the API baselines is the single-call setting, which precludes iterative interaction, execution feedback, and self-correction—an important direction for future work on enterprise-grade agents. Despite this constraint, our design narrows the gap by using more efficient spreadsheet encodings and task-appropriate tool outputs within the single-call budget.

Consistency Between Human and Automated Evaluation As shown in Table 6, the automated evaluation largely aligns with human judgments: for GPT 5.1 Pro and Claude Sonnet 4.5, the GPT-5-mini judge agrees with human labels on 82.1% and 90.2% of workflows, respectively. The judge also achieves high recall (83.3% and 88.4%), meaning it recovers most human-labeled passes. Replacing GPT-5-mini with the stronger GPT-5.1 judge yields lower automated pass rates (37.2% and 23.2%, respectively), bringing them closer to human scores. We adopt GPT-5-mini as the judge to avoid using the same model for both evaluation and the API-based baselines.

On the model side, the LLM judge can occasionally miss nuances in the rubric—either failing to catch subtle visual or numerical errors in large spreadsheets or, conversely, being overly literal about certain instructions (e.g., penalizing benign formula-to-constant conversions). However, we also observe cases where the LLM-based judge is correct but human raters are wrong—for example, when formulas are silently replaced with static values, which are difficult to detect through GUI-based inspection alone. On the system side, limitations of our spreadsheet tooling and data pipeline (e.g., incomplete support for corrupted but human-readable workbooks or uncommon file formats) can cause valid outputs to be marked as failures. Taken together, these factors mean that our automated scores should be interpreted as approximate rather than exact, and that human review remains important for borderline or high-impact workflows.

Product	Automated (%)		Human (%)
	GPT-5-mini	GPT-5.1	
GPT 5.1 Pro	41.9	37.2	38.4
Sonnet 4.5	29.1	23.2	25.0

Table 6: Comparison of automated LLM-as-judge pass rates using different judges and human evaluation on product-side agents. Human evaluation serves as the gold standard. Since Opus 4.5 has been newly released, we did not perform human evaluation for it.

3.3 Error Analysis

To understand the sources of failure on FINCH, we conducted a qualitative error analysis of GPT 5.1 and Claude Sonnet 4.5, considering both product-side agents and API-based models. For all failed workflows in our evaluation, we manually inspected the trajectories and annotated the primary cause of failure.

From a workflow-centric perspective, we identify five dominant categories of error. Take the Claude Sonnet 4.5 product-side agent as an example. Across all examined failures, 10% stem from *task misunderstanding*: enterprise tasks often rely on implicit context in artifacts (e.g., spreadsheets), which models frequently overlook, leading them to misinterpret what is being asked and the required deliverable. 25% are *data retrieval errors*, including selecting the wrong cross-sheet, cross-table, or intra-table row/column ranges. 35% arise from *formula reasoning errors*, such as failing to reconstruct the latent business logic encoded in formulas or deriving incorrect new formulas. 25% are due to *code generation errors*, where generated scripts (e.g., Python with spreadsheet APIs) are

syntactically invalid or misaligned with the spreadsheet layout. The remaining 5% correspond to *data rendering errors*, including incorrect formatting, misconfigured charts, or flawed final reports that deviate from the requested layout or narrative—for example, creating a brand-new spreadsheet instead of modifying the original one as requested. We also compare error patterns between web-based agents and API-based setups, with details provided in Appendix E.1.

Notably, these errors largely arise from *generic capabilities* that modern LLMs already appear to master in isolation on many existing benchmarks. The sharp degradation on FINCH therefore stems not from the absence of these abilities, but from their *composition* within realistic enterprise Finance & Accounting workflows. In FINCH, individual workflows simultaneously involve large and fragmented spreadsheet ecosystems, dense and semantically homogeneous financial content, irregular table structures, latent business logic encoded in formulas, and multimodal artifacts spanning spreadsheets, PDFs, charts, and other documents. When these challenges co-occur, small local errors—such as minor retrieval mistakes or misinterpreted structures—are easily amplified and propagate across long-horizon execution, ultimately leading to workflow-level failure.

Taken together, these observations suggest that FINCH does not require fundamentally new model abilities; rather, it probes existing capabilities under an enterprise "extreme" regime characterized by high complexity, noise, and long-horizon dependencies that closely mirror real F&A work. A detailed breakdown and concrete examples of these error factors are provided in Appendix E.2.

4 Related Work

The integration of LLMs into enterprise productivity tools has accelerated dramatically in recent years. The recently launched ChatGPT Agent (OpenAI, 2025) extends these capabilities to financial and accounting spreadsheet automation. Major productivity suites such as Microsoft 365 and Google Workspace have also integrated LLMs (e.g., Copilot and Gemini) to support document drafting, spreadsheet analysis, and workflow automation through natural language interaction (Microsoft, 2024; Google, 2024). Anthropic's Claude Excel has similarly entered the enterprise space with spreadsheet automation capabilities (Anthropic, 2025b). In parallel, a growing body of research has emerged on agentic spreadsheet processing (Li et al., 2023a; Chen et al., 2025; Zhu et al., 2025; Ma et al., 2024; Wang et al., 2025c).

In the past few years, there has also been significant progress in benchmarks for financial reasoning (Chen et al., 2021; Islam et al., 2023; Bigeard et al., 2025; Xie et al., 2024a; Liu et al., 2025a; Choi et al., 2025; Wang et al., 2025b; Li et al., 2025a; Liu et al., 2025b; Hu et al., 2025; Zhang et al., 2025; Xie et al., 2024b), office workspace automation (Wang et al., 2024, 2025a), and spreadsheet and tabular understanding and reasoning (Wang et al., 2021; Cheng et al., 2022; Ma et al., 2024; Indika and Molybog, 2025; Li et al., 2023b; Dong et al., 2024; Wu et al., 2025; Li et al., 2025b; Dong et al., 2025; Li et al., 2024; Zhao et al., 2024), as well as multimodal document understanding (Mathew et al., 2021), driving advances in LLM-based agents for enterprise tasks. In particular, GDPval (Patwardhan et al., 2025) evaluates economically valuable and professional tasks by comparing the quality of full work deliverables. FINCH complements these efforts by emphasizing collaborative and long-horizon workflows grounded in pre-existing enterprise artifacts (emails, versioned workbooks), where agents must coordinate and revise inherited materials rather than produce artifacts entirely from scratch.

5 Conclusion

We introduced FINCH, a new benchmark for real-world F&A enterprise workflows. FINCH derives workflows induced from enterprise email threads, version histories of spreadsheets, and high-quality financial artifacts with rigorous expert annotation, enabling systematic evaluation of agents on diverse workflows that operate over large, messy, and multimodal enterprise artifacts and require long-horizon reasoning. Experiments show that even the strongest frontier systems pass fewer than 50% of workflows, revealing a gap between current AI capabilities and the demands of real enterprise practice. We hope FINCH will benchmark agents on real, messy, and long-horizon F&A work.

6 Acknowledgments

We thank Jieqiong Li, Zikun Zhu, Chi Zhang for reviewing the dataset and providing valuable advice. This work was supported by the Zhongguancun Academy (Grant Nos. C20250210 and XTS0044).

7 Limitations

Access to contemporary enterprise data. A central challenge in building realistic workflow benchmarks is that true enterprise work records are extremely difficult to obtain due to privacy, compliance, and contractual restrictions. FINCH is therefore anchored in publicly available real-world sources, most prominently the Enron corpus, which contains uniquely rich email threads and attached artifacts that reflect authentic collaborative work. However, Enron primarily reflects workplace practices in the early 2000s, and enterprise tooling and collaboration patterns have evolved over the past two decades. Fortunately, the core productivity substrates for finance and accounting—email as the communication backbone and spreadsheets as the primary computational artifact—were relatively mature by that time and remain central in modern enterprise operations.

To mitigate this limitation, we complement Enron with a large collection of more recent, publicly available enterprise artifacts from financial institutions, global organizations, and government agencies, including the International Debt Report 2025, World Bank 2024 reports, Adidas 2024 financial statement, and Public Accounts of Canada 2024. FINCH also includes modern workflow patterns such as importing/processing data from contemporary finance software and web-search tasks requiring up-to-date information (up to 2025/04). These materials help FINCH better reflect modern finance and accounting work while respecting privacy and compliance constraints.

From measurement to closing the gap. Our benchmark surfaces both promising capabilities and substantial shortcomings of frontier AI systems on realistic workflows. However, FINCH is designed primarily as an evaluation and diagnostic resource rather than as a proposal of new modeling or training techniques. How to develop methodologically novel approaches that close the observed gaps—e.g., robust schema understanding under messy layouts, reliable formula-centric reasoning, and long-horizon execution with error recovery—is an important direction for future work.

8 Ethical considerations

The FINCH benchmark is constructed entirely from existing, publicly available data sources. Concretely, our workflows are derived from (1) the Enron email corpora, including the parsed Enron email dataset on Kaggle (released under the CC0 Public Domain dedication) and the Enron Email Dataset from EnronData.org (licensed under CC BY 3.0 US); (2) the EUSES spreadsheet corpus and its modified variants (CC BY 4.0); and (3) a diverse collection of enterprise-like artifacts, including documents from investment and securities companies, the World Bank (CC BY 3.0), Canadian and British government websites (Open Government License), and public corpora such as WideSearch (MIT license) and DABStep (CC BY 4.0). We respect the original licenses of all upstream resources and only redistribute content within the terms they allow.

On top of these sources, we apply additional filtering, normalization, and expert annotation to organize spreadsheets and related documents into coherent workflows with task instructions, input files, and reference outputs. We do not introduce any new personally identifiable information. During curation, we remove obviously sensitive fields when they are not necessary for the task (e.g., personal contact information or signatures) and avoid annotating workflows whose successful completion would depend on sensitive personal attributes. The resulting FINCH dataset is released under the Creative Commons Attribution 3.0 United States license (CC BY 3.0 US), which permits broad reuse while requiring appropriate attribution.

AI-assisted tools were used to summarize workflows from emails and versioned spreadsheets, identify labeling issues, and improve language clarity during manuscript preparation. All technical content, experimental design, and conclusions were carefully determined by the authors.

The language in FINCH is primarily English, reflecting the dominant language of the underlying Enron and EUSES corpora and many of the public institutional sources. Because some artifacts originate from funds and securities institutions and from Canadian government materials, a small fraction of workflows include Chinese or French content.

We used AI to discover human annotation issues and polish the grammar of this paper.

Potential risks include over-reliance on agentic systems evaluated on enterprise-like benchmarks, where automation without sufficient human-in-the-loop validation could miss unexpected errors in professional workflows.

References

Anthropic. 2025a. Claude cowork. https://www.anthropic.com/product/claude-cowork.

Anthropic. 2025b. Claude for excel. https://claude.com/claude-for-excel.

Anthropic. 2025. Introducing claude sonnet 4.5. https://www.anthropic.com/news/claude-sonnet-4-5. Accessed: 2025-12-14.

World Bank. 2024. *International Debt Report 2024*. World Bank, Washington, DC. World Bank's annual publication on external debt statistics.

Antoine Bigeard, Langston Nashold, Rayan Krishnan, and Shirley Wu. 2025. Finance agent benchmark: Benchmarking llms on real-world financial research tasks. *arXiv preprint arXiv:2508.00828*.

Yibin Chen, Yifu Yuan, Zeyu Zhang, Yan Zheng, Jinyi Liu, Fei Ni, Jianye Hao, Hangyu Mao, and Fuzheng Zhang. 2025. Sheetagent: towards a generalist agent for spreadsheet reasoning and manipulation via large language models. In *Proceedings of the ACM on Web Conference 2025*, pages 158–177.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R Routledge, and 1 others. 2021. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2022. Hitab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1110.

Chanyeol Choi, Jihoon Kwon, Alejandro Lopez-Lira, Chaewoon Kim, Minjae Kim, Juneha Hwang, Jaeseon Ha, Hojun Choi, Suyeol Yun, Yongjin Kim, and 1 others. 2025. Finagentbench: A benchmark dataset for agentic retrieval in financial question answering. In *Proceedings of the 6th ACM International Conference on AI in Finance*, pages 632–637.

Department of Finance Canada. 2025. Fiscal reference tables, november 2025. Technical report, Government of Canada, Ottawa, Canada. Provides annual data on the financial position of the federal, provincial-territorial and local governments.

Haoyu Dong, Yue Hu, and Yanan Cao. 2025. Reasoning and retrieval for complex semi-structured tables via reinforced relational data transformation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1382–1391.

Haoyu Dong, Jianbo Zhao, Yuzhang Tian, Junyu Xiong, Shiyu Xia, Mengyu Zhou, Yun Lin, José Cambronero, Yeye He, Shi Han, and 1 others. 2024. Spreadsheetllm: encoding spreadsheets for large language models. *arXiv preprint arXiv:2407.09025*.

Alex Egg, Martin Iglesias Goyanes, Friso Kingma, Andreu Mora, Leandro von Werra, and Thomas Wolf. 2025. Dabstep: Data agent benchmark for multi-step reasoning. *arXiv preprint arXiv:2506.23719*.

EnronData.org. Edo enron email pst dataset. https://enrondata.readthedocs.io/en/latest/data/edo-enron-email-pst-dataset/. Creative Commons Attribution 3.0 United States License. To provide attribution, please cite to "EnronData.org.".

Marc Fisher and Gregg Rothermel. 2005. The euses spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms. In *Proceedings of the first workshop on End-user software engineering*, pages 1–5.

Google. 2024. Gemini for google workspace. https://workspace.google.com/solutions/ai/.

Google DeepMind. 2025. Gemini 3 pro. https://deepmind.google/models/gemini/pro/. Accessed: 2025-12-14.

HM Treasury. 2023. Public expenditure statistical analyses 2023. Technical report, HM Treasury, London, United Kingdom. UK public expenditure statistical release (PESA).

Liang Hu, Jianpeng Jiao, Jiashuo Liu, Yanle Ren, Zhoufutu Wen, Kaiyuan Zhang, Xuanliang Zhang, Xiang Gao, Tianci He, Fei Hu, and 1 others. 2025. Finsearchcomp: Towards a realistic, expert-level evaluation of financial search and reasoning. *arXiv preprint arXiv:2509.13160*.

Amila Indika and Igor Molybog. 2025. Sodbench: A large language model approach to documenting spreadsheet operations. *arXiv preprint arXiv:2510.19864*.

Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. 2023. Financebench: A new benchmark for financial question answering. *arXiv preprint arXiv:2311.11944*.

Bryan Klimt and Yiming Yang. 2004. The enron corpus: A new dataset for email classification research. In *European conference on machine learning*, pages 217–226. Springer.

Haohang Li, Yupeng Cao, Yangyang Yu, Shashidhar Reddy Javaji, Zhiyang Deng, Yueru He, Yuechen Jiang, Zining Zhu, Kp Subbalakshmi, Jimin Huang, and 1 others. 2025a. Investorbench: A benchmark for financial decision-making tasks with llm-based agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2509–2525.

Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and Zhao-Xiang Zhang. 2023a. Sheetcopilot: Bringing software productivity to the next level through large language models. *Advances in Neural Information Processing Systems*, 36:4952–4984.

Jinyang Li, Nan Huo, Yan Gao, Jiayi Shi, Yingxiu Zhao, Ge Qu, Yurong Wu, Chenhao Ma, Jian-Guang Lou, and Reynold Cheng. 2024. Tapilot-crossing: Benchmarking and evolving llms towards interactive data analysis agents. *arXiv preprint arXiv:2403.05307*.

Peng Li, Yeye He, Cong Yan, Yue Wang, and Surajit Chaudhuri. 2023b. Auto-tables: Synthesizing multi-step transformations to relationalize tables without using examples. *Proceedings of the VLDB Endowment*, 16(11):3391–3403.

Zheng Li, Yang Du, Mao Zheng, and Mingyang Song. 2025b. Mimotable: A multi-scale spreadsheet benchmark with meta operations for table reasoning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 2548–2560.

Shu Liu, Shangqing Zhao, Chenghao Jia, Xinlin Zhuang, Zhaoguang Long, Jie Zhou, Aimin Zhou, Man Lan, and Yang Chong. 2025a. Findabench: Benchmarking financial data analysis ability of large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 710–725.

Zhaowei Liu, Xin Guo, Haotian Xia, Lingfeng Zeng, Fangqi Lou, Jinyi Niu, Mengping Li, Qi Qi, Jiahuan Li, Wei Zhang, and 1 others. 2025b. Visfineval: A scenario-driven chinese multimodal benchmark for holistic financial understanding. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 24099–24157.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. 2024. Spreadsheetbench: Towards challenging real world spreadsheet manipulation. *Advances in Neural Information Processing Systems*, 37:94871–94908.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209.

Microsoft. 2024. Microsoft 365 copilot. https://www.microsoft.com/en-us/microsoft-365/copilot.

OpenAI. 2025. Gpt-5. https://openai.com/index/introducing-gpt-5/. Accessed: 2025-12-14.

OpenAI. 2025. Introducing chatgpt agent. https://openai.com/index/introducing-chatgpt-agent/.

OpenClaw. 2025. Openclaw: Personal ai assistant. https://github.com/openclaw/openclaw.

Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, and 1 others. 2025. Gdpval: Evaluating ai model performance on real-world economically valuable tasks. *arXiv preprint arXiv:2510.04374*.

Jean-Paul Sartre. 1946. *Existentialism Is a Humanism*.

Weixuan Wang, Dongge Han, Daniel Madrigal Diaz, Jin Xu, Victor Rühle, and Saravan Rajmohan. 2025a. Odysseybench: Evaluating llm agents on long-horizon complex office application workflows. *arXiv preprint arXiv:2508.09124*.

Yan Wang, Keyi Wang, Shanshan Yang, Jaisal Patel, Jeff Zhao, Fengran Mo, Xueqing Peng, Lingfei Qian, Jimin Huang, Guojun Xiong, and 1 others. 2025b. Finauditing: A financial taxonomy-structured multi-document benchmark for evaluating llms. *arXiv preprint arXiv:2510.08886*.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1780–1790.

Zilong Wang, Yuedong Cui, Li Zhong, Zimin Zhang, Da Yin, Bill Yuchen Lin, and Jingbo Shang. 2024. Officebench: Benchmarking language agents across multiple applications for office automation. *arXiv preprint arXiv:2407.19056*.

Ziwei Wang, Jiayuan Su, Mengyu Zhou, Huaxing Zeng, Mengni Jia, Xiao Lv, Haoyu Dong, Xiaojun Ma, Shi Han, and Dongmei Zhang. 2025c. Sheetbrain: A neuro-symbolic agent for accurate reasoning over complex and large spreadsheets. *arXiv preprint arXiv:2510.19247*.

Ryan Wong, Jiawei Wang, Junjie Zhao, Li Chen, Yan Gao, Long Zhang, Xuan Zhou, Zuo Wang, Kai Xiang, Ge Zhang, and 1 others. 2025. Widesearch: Benchmarking agentic broad info-seeking. *arXiv preprint arXiv:2508.07999*.

Pengzuo Wu, Yuhang Yang, Guangcheng Zhu, Chao Ye, Hong Gu, Xu Lu, Ruixuan Xiao, Bowen Bao, Yijing He, Liangyu Zha, and 1 others. 2025. Realhitbench: A comprehensive realistic hierarchical table benchmark for evaluating llm-based table analysis. *arXiv preprint arXiv:2506.13405*.

xAI. 2025. Grok 4. https://x.ai/news/grok-4.

Qianqian Xie, Weiguang Han, Zhengyu Chen, Ruoyu Xiang, Xiao Zhang, Yueru He, Mengxi Xiao, Dong Li, Yongfu Dai, Duanyu Feng, and 1 others. 2024a. Finben: A holistic financial benchmark for large language models. *Advances in Neural Information Processing Systems*, 37:95716–95743.

Qianqian Xie, Weiguang Han, Zhengyu Chen, Ruoyu Xiang, Xiao Zhang, Yueru He, Mengxi Xiao, Dong Li, Yongfu Dai, Duanyu Feng, and 1 others. 2024b. Finben: A holistic financial benchmark for large language models. *Advances in Neural Information Processing Systems*, 37:95716–95743.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Zhihan Zhang, Yixin Cao, and Lizi Liao. 2025. Xfinbench: Benchmarking llms in complex financial problem solving and reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8715–8758.

Wei Zhao, Zhitao Hou, Siyuan Wu, Yan Gao, Haoyu Dong, Yao Wan, Hongyu Zhang, Yulei Sui, and Haidong Zhang. 2024. Nl2formula: Generating spreadsheet formulas from natural language queries. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 2377–2388.

Ruiyan Zhu, Xi Cheng, Ke Liu, Brian Zhu, Daniel Jin, Neeraj Parihar, Zhoutian Xu, and Oliver Gao. 2025. Sheetmind: An end-to-end llm-powered multi-agent framework for spreadsheet automation. *arXiv preprint arXiv:2506.12339*.

Figure 6: Real-world F&A work is messy, spanning heterogeneous and large-scale artifacts such as spreadsheets and PDFs. It's also long-horizon and knowledge-intensive: workflows interleave multiple tasks and span diverse domains such as budgeting, trading, asset management, and operations.

Author	Affiliation	Email
Haoyu Dong*	UCAS; Microsoft	donghaoyu82@gmail.com
Pengkun Zhang	South China University of Technology	sezhangpengkun@mail.scut.edu.cn
Yan Gao	Zhongguancun Institute of Artificial Intelligence	gaoyan@zgci.ac.cn
Xuanyu Dong	Harvest Fund	qianmuxuanyu@126.com
Yilin Cheng	Zhongguancun Academy; Fudan University	ylcheng23@m.fudan.edu.cn
Mingzhe Lu	UCAS	lumingzhe23@mails.ucas.edu.cn
Zikun Zhu	Duke University	zz295@duke.edu
Adina Yakefu	Hugging Face	adina@huggingface.com
Shuxin Zheng	Zhongguancun Academy; Zhongguancun Institute of Artificial Intelligence	sz@bza.edu.cn

*Corresponding author. UCAS stands for the University of Chinese Academy of Sciences

A Author List

B Detailed Task and Business Type Distribution

Figure 3 summarizes the coverage of task and business types in FINCH. On the task side, the benchmark includes calculation tasks (119 workflows) such as filling in formulas or computing figures; structuring and formatting tasks (86) involving table reorganization, layout adjustment, and row/column insertion or deletion; data entry and import tasks (44) that transcribe or ingest data from spreadsheets, PDFs, images, or external sources; validation and review tasks (37) that check consistency and reconcile calculations within or across sheets and files; cross-sheet or cross-file retrieval tasks (36) that aggregate values from multiple sources into a target workbook; summary and visualization tasks (33) that produce financial summaries or charts; financial modeling tasks (15) that extend or calibrate valuation and timing models; web search tasks (11) that collect and integrate financial data from online sources; and a small number of translation tasks (3) that translate spreadsheets or reports while preserving structure, formatting, and layout.

On the business side, workflows span reporting (48 workflows), trading and risk management (35), predictive modeling (33), operational management (36), planning and budgeting (26), pricing and valuation (15), accounts payable and receivable (10), as well as procurement and sales (7) and asset management (3). Some workflows are tagged with multiple business types, reflecting the cross-functional nature of real-world enterprise finance and accounting work.

C Quality Control

20% of our annotators are full-time reviewers who do not create any original annotations, and 80% act

as full-time workflow creators (producing the initial annotations) and also serve as part-time reviewers. Each workflow is proposed by one person and typically reviewed by two others, including a dedicated full-time reviewer who reviews all queries and also inspects LLM-generated review comments to surface potential annotation issues. Importantly, Quality Control (QC) is iterative rather than one-pass: approximately 40% of workflows undergo at least one round of revision; 20+ workflows went through three or more QC rounds, and we held regular group discussions to resolve challenging cases and align standards. We also maintain version-controlled documents and review notes that record key parts of this process.

D Experiment Details

D.1 API-based Models

Execution Paradigm. We frame the API evaluation as a **code generation task**. Models are instructed to solve spreadsheet manipulation and generation workflows by generating executable Python scripts, which are then executed in a sandboxed environment to produce output artifacts. This paradigm aligns with SpreadsheetBench's philosophy of treating model-written code as the primary action space, but is adapted here to accommodate long-horizon, multimodal enterprise tasks.

- **Action Space**: Models generate Python code using standard libraries including `openpyxl` (for Excel manipulation), `pandas` (for data processing), `matplotlib` (for visualization), and `scikit-learn` (for statistical analysis).
- **Output Format**: Models must produce complete, self-contained Python scripts wrapped in markdown code blocks (“‘`python ...` “‘).
- **Sandboxed Execution**: Generated code is extracted via regex parsing and executed in isolated Docker containers running Jupyter Kernel Gateway. Each container mounts the dataset volume at `/mnt/data/` with a 10-minute session timeout.
- **Single-shot Protocol**: We employ a strict one-shot generation protocol without iterative refinement—each model produces exactly one solution per workflow. If the generated code fails to execute (e.g., due to syntax errors or runtime exceptions), the workflow is marked as failed without retry. This strict setting is designed to evaluate the model's raw code generation capability under realistic deployment constraints.

This unified code-as-action setting ensures that the measured performance reflects the model's inherent competence on complex workflows rather than benefits derived from interactive debugging.

Prompting Strategy. We employ a **zero-shot** setting with a structured system prompt comprising:

1. A role definition: “You are an expert who can manipulate spreadsheets through Python code.”
2. A detailed description of the compact spreadsheet encoding format with illustrative examples.
3. The task instruction and explicit input/output file paths.
4. Library-specific best practices (e.g., `openpyxl` chart creation patterns) to mitigate common code errors.
5. An explicit directive to generate Python code as the final output.

This structured design explicitly guides models toward generating valid, context-aligned Python code, minimizing ambiguity in task interpretation. However, for models that support reasoning traces (GPT 5.1, Gemini 3 Pro), we request explicit reasoning via the `include_reasoning` API parameter, enabling us to capture the model's internal deliberation process for subsequent qualitative error analysis. Temperature is set to 0.7 across all models.

Model License All models evaluated in this work are accessed through their official APIs or product interfaces and are subject to the corresponding providers' terms of service and licensing agreements. We do not redistribute any model weights, training data, or proprietary outputs.

GPT 5.1 is provided by OpenAI and used in accordance with OpenAI's API and product usage policies. Claude Sonnet 4.5 is provided by Anthropic and accessed via its official API under Anthropic's commercial license. Grok 4 is provided by xAI and evaluated through its public API under xAI's terms of service. Qwen 3 Max is provided by Alibaba Cloud and accessed subject to Alibaba's licensing terms. Gemini 3 Pro Preview is provided by Google and evaluated via Google's official API under the applicable Google AI service terms.

All evaluations are conducted for research purposes. We do not claim ownership over any model outputs, and the results reported in this paper reflect

Model	Provider	Context	Max Output	Vision	Native PDF
GPT 5.1 (OpenAI, 2025)	OpenAI	400K	128K	✓	✓
Claude Sonnet 4.5 (Anthropic, 2025)	Anthropic	1M†	64K	✓	✓
Grok 4 (xAI, 2025)	xAI	256K	256K	✓	—
Qwen 3 Max (Yang et al., 2025)	Alibaba	256K	32.8K	—	—
Gemini 3 Pro Preview (Google DeepMind, 2025)	Google	1.05M	65.5K	✓	✓

Table 7: API-based model configurations. Context and output limits are measured in tokens. Vision indicates native image input support, while Native PDF refers to direct PDF file ingestion via the provider's API without explicit text extraction. †Available via long-context beta API mode.

observed model behavior under controlled experimental settings rather than any guarantees of model performance. Any use of the evaluated models beyond this study must comply with the respective providers' licenses and usage policies.

Multimodal Input Handling. We extend the framework to support multimodal inputs involving images and PDFs. For vision-capable models (GPT 5.1, Claude Sonnet 4.5, Grok 4, and Gemini 3 Pro), we use each provider's official multimodal API to transmit visual inputs alongside text prompts. For PDF documents, we adopt a tiered strategy. Models with native PDF support—GPT 5.1, Claude Sonnet 4.5, and Gemini 3 Pro Preview—directly ingest PDF files via their file upload interfaces, enabling analysis of both textual and visual elements without pre-extraction. For Grok 4, which lacks native PDF support, we extract text using PyMuPDF and include it in the `pdf_content` field. For Qwen 3 Max, which lacks multimodal support entirely, both image and PDF content are converted to textual descriptions. While this fallback retains semantic cues, it loses layout and visual context.

Context Management. To handle large spreadsheets that may exceed model context limits, we implement automatic truncation. We reserve 32K tokens for model output—sufficient for comprehensive code generation and analysis while remaining within the output limits of all evaluated models. Truncation is triggered when input exceeds the remaining capacity, removing content from the end of the spreadsheet data with an explicit notice appended to inform the model of data loss.

E Detailed Analysis

E.1 Agent Behavior Analysis

Product-side agents. ChatGPT 5.1 Pro tends to decompose workflows into more, smaller steps, with explicit reasoning, tool calls, execution, and self-checking at each step. This leads to longer traces and noticeably higher latency, but also more opportunities for intermediate validation (e.g., sanity-checking partial results). However, the code it generates is often hidden behind tool abstractions, so our error attribution is limited to observed behavior and natural language reasoning rather than the exact implementation details. Claude Sonnet 4.5 typically uses fewer steps and produces more direct solutions. In visualization-heavy workflows, its generated charts are often both more accurate and more aesthetically polished than those produced by ChatGPT 5.1 Pro, leading to relatively fewer failures in the data visualization sub-tasks.

ChatGPT 5.1 Pro and Claude Sonnet 4.5 agents can explore Excel files through many API calls within a single workflow, but their encoding methods are not well-suited to spreadsheets with complex layouts and structures. Thanks to efficient encoding and appropriate tool use, the following single-call API-based method achieves a pass rate that is much closer to that of product-side agents.

API-based models Our API-based runs are single-call: they leverage the models' underlying reasoning capabilities but lack two crucial affordances that web agents exploit. (i) interleaved code execution with feedback, and (ii) explicit reflection based on intermediate tool outputs. As a result, the API agents must generate the entire plan, code, and outputs within a single LLM call. When their initial structural assumptions about a spreadsheet are slightly off, they have no mechanism to detect or correct the mistake, leading to a significantly higher error rate, particularly in categories related to schema understanding and table manipulation. It's desirable for future work to explore agentic methods with multiple rounds of API calls.

E.2 Detailed Error Factors in Enterprise F&A Workflows

First, FINCH workflows routinely involve *large, fragmented spreadsheet ecosystems*: dozens of in-

terlinked workbooks and thousands of rows distributed across many sheets. Executing these workflows accurately requires long-range cross-sheet navigation and precise referencing, which substantially increases the likelihood of small retrieval errors. Second, the *content is dense and semantically homogeneous*: many cells contain domain-specific financial concepts that are subtly different yet lexically similar (e.g., variants of revenue/expense items, adjusted vs. unadjusted metrics), making entity disambiguation and cell grounding unusually difficult. Third, the *table layouts and structures are complex and often irregular*, including multi-level headers, merged cells, nested subtotals, and bespoke layouts that force the model to infer structure from noisy contents and ad hoc formatting. For example, at the code level, even tiny misinterpretations of these layouts (e.g., off-by-one errors when specifying ranges) can then propagate into globally incorrect outputs, especially when logic is applied in batch across many such sheets. Fourth, *formulas encode latent structure and logic*. In the FINCH dataset, each sheet contains a large number of formulas that encode latent business logic, temporal assumptions, and fine-grained dependencies that are not visible from displayed values alone; yet models typically prioritize cell values and underuse formulas, leading to systematic misinterpretations. For example, in a pricing sheet with the column header `IF NGPL MidContinent index (@ Baker)`, the apparent semantics from the header alone suggest a daily exposure metric. However, inspecting the associated formula (`25 * V21 + C41 * C22`) reveals that the column in fact encodes a 55-day payment timing. Models that ignore or under-utilize formulas systematically misinterpret such columns' roles in downstream calculations, and this misinterpretation then propagates through subsequent steps. Finally, many workflows involve *multimodal artifacts and chat-centric tasks* such as combining spreadsheets with PDFs, charts, and screenshots, requiring the agent to jointly reason over heterogeneous formats. For example, tables embedded in PDFs are often only partially referenced, with key entries missing or truncated.

F Tool-Call Overhead Case Study

To better understand the practical complexity of completing FINCH tasks, we conduct a small-scale case study on tool-call overhead. From the full set of 172 workflows, we select 20 representative examples covering different task types and varying levels of complexity, and analyze how many tool calls are required for an LLM-based agent to complete them.

F.1 Methodology

For each selected task, we use the up-to-date (March 2026) Claude Coworker (Opus 4.6) in an isolated session and ask it to solve the task according to the provided instruction. The agent is also required to write its tool-use trajectory into a log file, recording each tool call step by step. Since the number of tool calls in a failed attempt may differ from that in a successful completion, we use a second session to compare the produced output against the reference answer file and determine whether the result is correct.

When the output is incorrect, we observe two distinct cases from the perspective of tool usage. In the first case, the overall tool-use procedure is already complete, but the execution within one or more steps is wrong; for example, the agent invokes a Python script as needed, but the script contains an incorrect formula. In the second case, the trajectory is missing one or more critical steps that are necessary for producing the correct output. This second case is relatively uncommon, appearing only twice among the 20 examples we analyze. For such cases, we ask the model to identify the missing indispensable steps and merge them into the original trajectory. Therefore, our final count is intended to approximate the tool calls required to correctly complete the task, rather than merely reporting the raw tool calls from a failed run.

F.2 Results and Analysis

Table 8 summarizes the 20 selected tasks and their corresponding numbers of tool calls.

Web-grounded tasks incur substantially higher tool-call overhead. Among the 20 cases, the two tasks involving web search (ID 19 and ID 88) require dramatically more tool calls than the others. In Task 19, `websearch` and `webfetch` together account for 60 out of 107 tool calls (56%). In Task 88, these two tools account for 34 out of 71 tool calls (48%). During inspection, we find that web-grounded tasks often require repeated retrieval from multiple external sources, followed by cross-source comparison, integration, and verification. As a result, the dominant cost in such tasks comes not from spreadsheet manipulation itself, but from

Workflow ID	Task Type(s)	#Tasks	#Tool Calls
46	Calculation	1	7
31	Summary / Visualization	1	8
26	Summary / Visualization	1	10
47	Calculation	1	13
17	Cross-sheet/file Retrieval	1	14
137	Structuring / Formatting	1	16
91	Calculation, Financial Modeling	2	6
106	Structuring / Formatting, Calculation	2	6
49	Validation / Review, Calculation	2	14
16	Calculation, Summary / Visualization	2	17
28	Validation / Review, Structuring / Formatting	2	17
158	Validation / Review, Calculation	2	18
15	Structuring, Translation, Summary / Vis.	3	9
150	Calculation, Data Entry, Cross-sheet Retrieval	3	9
66	Calculation, Structuring, Financial Modeling	3	17
139	Financial Modeling, Calculation, Summary / Vis.	3	25
159	Structuring, Calc., Fin. Modeling, Validation	4	14
81	Data Entry, Structuring, Validation, Calc.	4	17
19	Web Search, Data Entry, Structuring, Calc.	4	107
88	Web Search, Data Entry, Structuring, Calc., Vis.	5	71

Table 8: Case study of tool-call overhead on 20 representative tasks.

external information acquisition and validation.

Distribution of tool calls for non-web tasks. After excluding the two web-search tasks, the number of tool calls ranges from 6 to 25, with a mean of 13.2 and a median of 14. The first and third quartiles are 9 and 17, respectively, giving an interquartile range of 8.

Task-type count alone does not explain tool-call overhead. After removing the two web-search outliers, we do not observe a clear positive relationship between the number of task-type labels and the number of tool calls. For example, both Task 46 and Task 47 are labeled as *Calculation*, but Task 47 requires nearly twice as many tool calls due to additional business-logic reasoning (sign conventions, Purchase vs. Pooling Method tax-base logic, and cross-checking of business consistency).

Limitations. This analysis should be viewed as a case study rather than a full-scale statistical evaluation. The 20 tasks are selected as representative examples rather than a random sample, and the measured tool-call counts depend on the behavior of a specific agent configuration (Claude Coworker, Opus 4.6). In addition, our corrected trajectories for failed runs involve a post hoc reconstruction of missing indispensable steps, which introduces some degree of judgment into the counting process.

F.3 Representative Tool-Call Logs

We present two representative examples. Task 26 represents the general case, while Task 139 illustrates the case where the original trajectory omits critical steps required for correct completion.

F.3.1 Task 26

Task description. Add two charts to each of the three worksheets—`Scenario1`, `Scenario2`, and `Scenario3`:

1. A **Hedge Coverage** stacked column chart (first year only; X-axis = months; stacked series = contract MW values; Short = reference line; Y-axis = MW)
2. A **Portfolio Cost** line chart (all years; X-axis = months; series = Portfolio Weighted Avg and Monthly Price, with an optional fixed-price line; Y-axis = $/MWh)

Tool-call log.

1. **Bash** — List files in the working directory (`ls mnt/26/`)
   Purpose: Confirm that the source file exists.
2. **Read** — Read `metadata.json`
   Purpose: Understand the task instructions, source file, and output requirements.
3. **Read** — Read `SKILL.md`
   Purpose: Review best practices for Excel file operations.
4. **Bash** — Run a Python script to inspect the Excel file structure
   Purpose: Use `openpyxl` to read

26_src_0.xlsx and print all sheet names plus the first 20 rows of data, in order to understand the layout, column mapping, and row structure of each Scenario worksheet.

5. **Bash** — Read the full output file
   Purpose: Retrieve the complete structure information because the previous step's output was truncated.

6. **Bash** — Run a Python script to inspect the first 20 rows of each Scenario sheet in detail
   Purpose: Use get_column_letter to parse column names and values, confirm the meaning of each column, and verify data ranges and types.

7. **Bash** — Run a Python script to inspect all date rows in column Z of Scenario1 (rows 4–63)
   Purpose: Confirm the date range (January 2001 to December 2005), determine the first-year row range (4–15), and identify the full multi-year range (4–63).

8. **Bash** — Copy the source file to the output file (cp mnt/26/26_src_0.xlsx mnt/26/26.xlsx)
   Purpose: Preserve the original file and operate on a copy.

9. **Bash** — Create and run the Python script add_charts.py
   Purpose: Use openpyxl to add two charts to each of the three Scenario sheets, creating all six charts in total.

10. **Bash** — Run a Python script to verify that the charts were added successfully
    Purpose: Check the number of charts, titles, chart types, and number of series in each sheet to confirm that all six charts were correctly embedded.

F.3.2 Task 139

Note. This is a corrected tool-call log. Based on the original trajectory, several missing but essential steps were added (marked as **[Added]**). These mainly involve constructing monthly cashflow data for XNPV Variables, building multi-year analysis data for Cedar Brakes, and performing precise calculations using native Excel formulas.

Tool-call log.

1. **Bash** — ls mnt/139/
   Description: List the files in the workspace folder and confirm that the source file exists.

2. **Read** — metadata.json
   Description: Read the metadata file to understand the task instructions, source file list, and output requirements.

3. **Read** — xlsx SKILL.md
   Description: Read the Excel skill file to understand best practices and workflow procedures.

4. **Bash** — Python script to inspect the source Excel workbook structure (data_only=True)
   Description: Use openpyxl in data_only mode to open the source workbook and inspect all worksheet names, dimensions, and data values.

5. **Bash** — Python script to fully read the data in the Damage Calculations sheet
   Description: Read all rows in the Damage Calculations worksheet in detail to understand the annual capacity payment amounts for Q1–Q4, the hard-coded XNPV values, and the sensitivity-table region.

6. **Bash** — Python script to fully read the Summary sheet
   Description: Read all regions of the Summary worksheet to understand the layout of each Q1–Q4 section and the structure of the Delta region.

7. **Bash** — Python script to read formulas in the Damage Calculations sheet (data_only=False)
   Description: Open the workbook in formula mode and retrieve all cell formulas in the Damage Calculations sheet to understand the calculation logic and the data-table formula objects.

8. **Read** — Read the full text output from Tool Call 7
   Description: Read the complete output to obtain all formula details.

9. **Bash** — Python script to read formulas in the Summary sheet
   Description: Read the formulas in the Summary worksheet in formula mode to understand the formula reference relationships.

10. **Bash** — Python script to read formulas and variables in the Tenaska's i Adjustment sheet

Description: Understand the calculation formulas and key variables used in the Tenaska interest-rate adjustment.

11. **Bash** — Python script to read variable values in the `Tenaska's i Adjustment` sheet
Description: Retrieve the variable values used in the calculations.

12. **Bash** — Python script to read formulas and variables in the `Brazos' i Adjustment` sheet
Description: Understand how the Brazos interest-rate adjustment differs from the Tenaska adjustment.

13. **Bash** — Python script to validate the XNPV calculation method
Description: Use known data to verify the XNPV calculation method and determine the correct date convention and discounting method.

14. **Bash** — Python script to inspect `DataTableFormula` parameters
Description: Inspect the data-table formula parameters in the `Damage Calculations` sheet and confirm that the table uses `F8` (plant capacity) as the input variable.

15. **[Added] Bash** — Python script to read the full data and formulas in the `Cedar Brakes` sheet
Description: Read all row and column data as well as the formula structure in the `Cedar Brakes` worksheet to understand:
    - the layout of annual data rows (`Year`, `Volume`, `Old PPA`, `Market`, `SI`, `New PPA`, `Savings`);
    - the calculation relationships among columns (e.g., `SI = Old PPA total - Market total`);
    - the range of years covered by the data (2000–2016+);
    - which rows and columns need to be filled under different capacity assumptions.

16. **[Added] Bash** — Python script to analyze annual capacity payment amounts under different capacities
Description: Based on the rate structure ($/kW-mo) in the `Damage Calculations` sheet, compute the annual capacity payment amounts for Q1–Q4 under four capacity assumptions: 240 / 245 / 258 / 263 MW:
    - **Q1 (Status Quo):** compute annual payments using the original rates;
    - **Q2 (Modified i):** use rates adjusted by Tenaska's i adjustment;
    - **Q3 (Modified i + Brazos):** use rates adjusted by both Tenaska and Brazos;
    - **Q4 (No i):** remove the interest-rate adjustment and recompute the rates.

17. **[Added] Bash** — Python script to construct monthly cash-flow data
Description: Convert annual capacity payment amounts into monthly cash-flow sequences in preparation for XNPV calculation:
    - generate a date sequence (from January 1997 to the contract end year, using month-end dates);
    - generate the corresponding monthly cash flows for Q1–Q4 under each scenario (= annual payment / 12);
    - verify that the sum of monthly cash flows equals the annual total.

18. **TodoWrite** — Create a task list
Description: Create and track task progress.

19. **Bash** — Main Python script for calculation and file modification
Description: Execute the main calculation and file-modification script:
    - copy the source file to the output file `139.xlsx`.

    [Added section — fill in `XNPV Variables`]
    - Write the title `"XNPV Variables:"` starting from row 88 of the `Damage Calculations` sheet;
    - write the column headers: `Date`, `Question 1`, `Question 2`, `Question 3`, `Question 4`;
    - write the date sequence and monthly cash-flow data for Q1–Q4 month by month;
    - for the 263 MW baseline capacity, directly use the annual rates already present in the `Damage Calculations` sheet;
    - for 240 / 245 / 258 MW capacities, scale the capacity payments proportionally.

    [Added section — fill in multi-year `Cedar Brakes` data]
    - Fill in the complete financial data for each year in the data region of the `Cedar Brakes` sheet (rows 24+);

- include `Volume`, `Old PPA` ($/MWh and total), `Market` ($/MWh and total), `SI`, `New PPA` ($/MWh and total), and `Savings`;
- cover all years from 2000 through the contract end year.

[Added section — use Excel `XNPV` formulas]

- Write XNPV formulas into the sensitivity-table region of the `Damage Calculations` sheet, referencing the monthly cash flows and dates in `XNPV Variables`;
- formula format: `=XNPV(discount_rate, cash_flow_range, date_range)`;
- ensure the formulas can be recalculated correctly in Excel.

[Original section]

- Fill in the values for each capacity in the Q1–Q4 sections of the `Summary` sheet;
- fill in the Delta section of the `Summary` sheet (differences for each capacity relative to the 263 MW baseline);
- save the file.

20. **TodoWrite** — Update task progress

21. **Glob** — Search for the path of `recalc.py`

22. **Bash** — `python recalc.py`
Description: Run the `recalc.py` script to recalculate all formulas and ensure that formulas such as XNPV are properly evaluated.

23. **[Added] Bash** — Python script to verify the integrity of `XNPV Variables` data
Description: Open the output file in `data_only` mode and verify:
    - the integrity of the `XNPV Variables` section (row count, date continuity, cash-flow values);
    - the correctness of the `Cedar Brakes` data (whether the values in each yearly column are reasonable);
    - whether the XNPV formula results are consistent with expectations.

24. **Bash** — Python script to verify values in the `Summary` sheet and the DC sensitivity table
Description: Verify the correctness of all populated values, including:
    - `PV5`, debt, `SWAP`, and implied equity value for each capacity in Summary Q1–Q4;
    - differences for each capacity relative to the 263 MW baseline in the Summary Delta section;
    - all XNPV values in the DC sensitivity table.

25. **TodoWrite** — Final update of task progress

G Examples

This section presents representative examples from FINCH to illustrate the diversity and complexity of spreadsheet-centric enterprise workflows covered by the benchmark. The examples span a wide range of realistic professional tasks, including cross-sheet verification, formula auditing, document-grounded extraction, schema transformation, financial modeling, and reporting.

Note that the figures shown in this section are illustrative excerpts rather than complete workflow inputs or full task specifications. Many FINCH workflows involve large, multi-sheet or multi-file inputs and extensive supporting documents, which cannot be fully visualized in static screenshots. Accordingly, the figures are intended to highlight representative data characteristics, structural complexity, and reasoning challenges that arise in the underlying workflows, rather than to fully specify the tasks themselves.

Across these workflows, AI agents are required to jointly reason over heterogeneous artifacts, while preserving structural consistency and semantic correctness. Many tasks demand multi-step reasoning, precise interpretation of formulas, and careful cross-referencing across sheets or external documents. Collectively, these examples highlight the core challenges targeted by FINCH: messy and multimodal document processing, long-horizon reasoning, robust code generation under schema variation, and faithful execution of complex financial and accounting logic.

Figure 7: For this task, the model must verify the department headcount summary by cross-checking each of the **39** departments against its detailed roster sheet. It should correct discrepancies such as miscounts and missing or duplicate entries. The summary must be updated by fixing incorrect totals, removing departments that no longer exist, and adding any omitted departments. Furthermore, the underlying schema varies slightly across departments, which challenges reliable code generation.

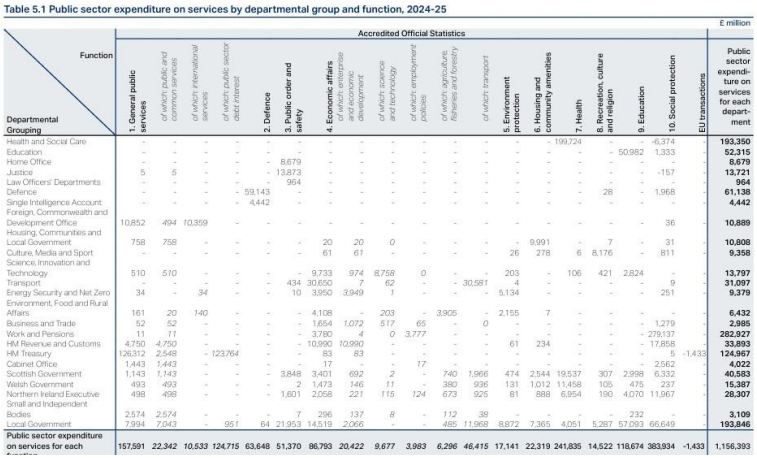


Table 5.1 Public sector expenditure on services by departmental group and function, 2024-25

£ million

Departmental Grouping	1. General public services	of which: public and common services	of which: international services	of which: public sector debt interest	2. Defence	3. Public order and safety	4. Economic affairs	of which: general economic and commercial affairs	of which: science and technology	of which: employment policies	of which: agriculture, fisheries and forestry	of which: transport	5. Environment protection	6. Housing and community amenities	7. Health	8. Recreation, culture and religion	9. Education	10. Social protection	EU transactions	Public sector expenditure on services for each department
Health and Social Care	-	-	-	-	-	-	-	-	-	-	-	-	-	199,724	-	-	-	-6,374	-	193,350
Education	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	50,982	1,333	-	52,315
Home Office	-	-	-	-	-	8,679	-	-	-	-	-	-	-	-	-	-	-	-	-	8,679
Justice	5	5	-	-	-	13,873	-	-	-	-	-	-	-	-	-	-	-	-157	-	13,721
Law Officers' Departments	-	-	-	-	-	964	-	-	-	-	-	-	-	-	-	-	-	-	-	964
Defence	-	-	-	-	59,143	-	-	-	-	-	-	-	-	-	-	28	-	1,968	-	61,138
Single Intelligence Account	-	-	-	-	4,442	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4,442
Foreign, Commonwealth and Development Office	10,852	494	10,359	-	-	-	-	-	-	-	-	-	-	-	-	-	-	36	-	10,889
Housing, Communities and Local Government	758	758	-	-	-	-	20	20	0	-	-	-	-	9,991	-	7	-	31	-	10,808
Culture, Media and Sport	-	-	-	-	-	-	61	61	-	-	-	-	26	278	6	8,176	-	811	-	9,358
Science, Innovation and Technology	510	510	-	-	-	-	9,733	974	8,758	0	-	-	203	-	106	421	2,824	-	-	13,797
Transport	-	-	-	-	-	434	30,650	7	62	-	-	30,581	4	-	-	-	-	9	-	31,097
Energy Security and Net Zero	34	-	34	-	-	10	3,950	3,949	1	-	-	-	5,134	-	-	-	-	251	-	9,379
Environment, Food and Rural Affairs	161	20	140	-	-	-	4,108	-	203	-	3,905	-	2,155	7	-	-	-	-	-	6,432
Business and Trade	52	52	-	-	-	-	1,654	1,072	517	65	-	0	-	-	-	-	-	1,279	-	2,985
Work and Pensions	11	11	-	-	-	-	3,780	4	0	3,777	-	-	-	-	-	-	-	279,137	-	282,927
HM Revenue and Customs	4,750	4,750	-	-	-	-	10,990	10,990	-	-	-	-	61	234	-	-	-	17,858	-	33,893
HM Treasury	126,312	2,548	-	123,764	-	-	83	83	-	-	-	-	-	-	-	-	-	5	-1,433	124,967
Cabinet Office	1,443	1,443	-	-	-	-	17	-	-	17	-	-	-	-	-	-	-	2,562	-	4,022
Scottish Government	1,143	1,143	-	-	-	3,848	3,401	692	2	-	740	1,966	474	2,544	19,537	307	2,998	6,332	-	40,583
Welsh Government	493	493	-	-	-	2	1,473	146	11	-	380	936	131	1,012	11,458	105	475	237	-	15,387
Northern Ireland Executive	498	498	-	-	-	1,601	2,058	221	115	124	673	925	81	888	6,954	190	4,070	11,967	-	28,307
Small and Independent Bodies	2,574	2,574	-	-	-	7	296	137	8	-	112	38	-	-	-	-	232	-	-	3,109
Local Government	7,994	7,043	-	951	64	21,953	14,519	2,066	-	-	485	11,968	8,872	7,365	4,051	5,287	57,093	66,649	-	193,846
Public sector expenditure on services for each	**157,591**	**22,342**	**10,533**	**124,715**	**63,648**	**51,370**	**86,793**	**20,422**	**9,677**	**3,983**	**6,296**	**46,415**	**17,141**	**22,319**	**241,835**	**14,522**	**118,674**	**383,934**	**-1,433**	**1,156,393**

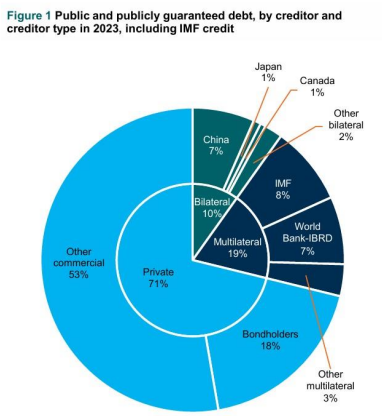
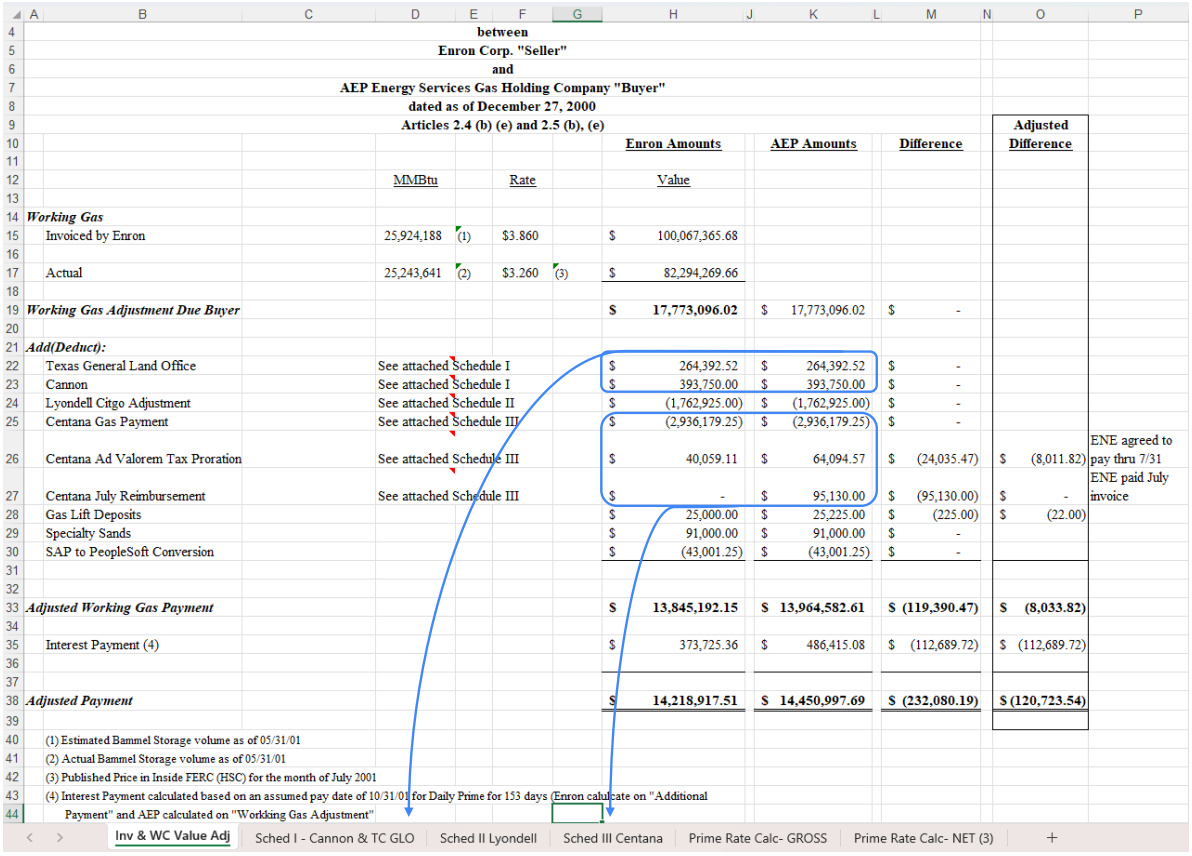


Figure 8: An example of extracting data from tables and charts in PDFs and saving it to a spreadsheet. AI agents must understand the layout, parse hierarchical structures, interpret how values map to specific cells, and reconstruct formulas from aggregated values.



					Enron Amounts	AEP Amounts	Difference	Adjusted Difference	
		MMBtu		Rate	Value				
Working Gas									
	Invoiced by Enron	25,924,188	(1)	$3.860	$ 100,067,365.68				
	Actual	25,243,641	(2)	$3.260 (3)	$ 82,294,269.66				
Working Gas Adjustment Due Buyer					**$ 17,773,096.02**	$ 17,773,096.02	$ -		
Add(Deduct):									
	Texas General Land Office	See attached Schedule I			$ 264,392.52	$ 264,392.52	$ -		
	Cannon	See attached Schedule I			$ 393,750.00	$ 393,750.00	$ -		
	Lyondell Citgo Adjustment	See attached Schedule II			$ (1,762,925.00)	$ (1,762,925.00)	$ -		
	Centana Gas Payment	See attached Schedule III			$ (2,936,179.25)	$ (2,936,179.25)	$ -		
	Centana Ad Valorem Tax Proration	See attached Schedule III			$ 40,059.11	$ 64,094.57	$ (24,035.47)	$ (8,011.82)	ENE agreed to pay thru 7/31
	Centana July Reimbursement	See attached Schedule III			$ -	$ 95,130.00	$ (95,130.00)	$ -	ENE paid July invoice
	Gas Lift Deposits				$ 25,000.00	$ 25,225.00	$ (225.00)	$ (22.00)	
	Specialty Sands				$ 91,000.00	$ 91,000.00	$ -		
	SAP to PeopleSoft Conversion				$ (43,001.25)	$ (43,001.25)	$ -		
Adjusted Working Gas Payment					**$ 13,845,192.15**	**$ 13,964,582.61**	**$ (119,390.47)**	**$ (8,033.82)**	
	Interest Payment (4)				$ 373,725.36	$ 486,415.08	$ (112,689.72)	$ (112,689.72)	
Adjusted Payment					**$ 14,218,917.51**	**$ 14,450,997.69**	**$ (232,080.19)**	**$ (120,723.54)**	

(1) Estimated Bammel Storage volume as of 05/31/01
(2) Actual Bammel Storage volume as of 05/31/01
(3) Published Price in Inside FERC (HSC) for the month of July 2001
(4) Interest Payment calculated based on an assumed pay date of 10/31/01 for Daily Prime for 153 days (Enron calculate on "Additional Payment" and AEP calculated on "Workking Gas Adjustment"

Inv & WC Value Adj | Sched I - Cannon & TC GLO | Sched II Lyondell | Sched III Centana | Prime Rate Calc- GROSS | Prime Rate Calc- NET (3)

Figure 9: Cross-sheet reference validation. This example is relatively easy for frontier AI agents.

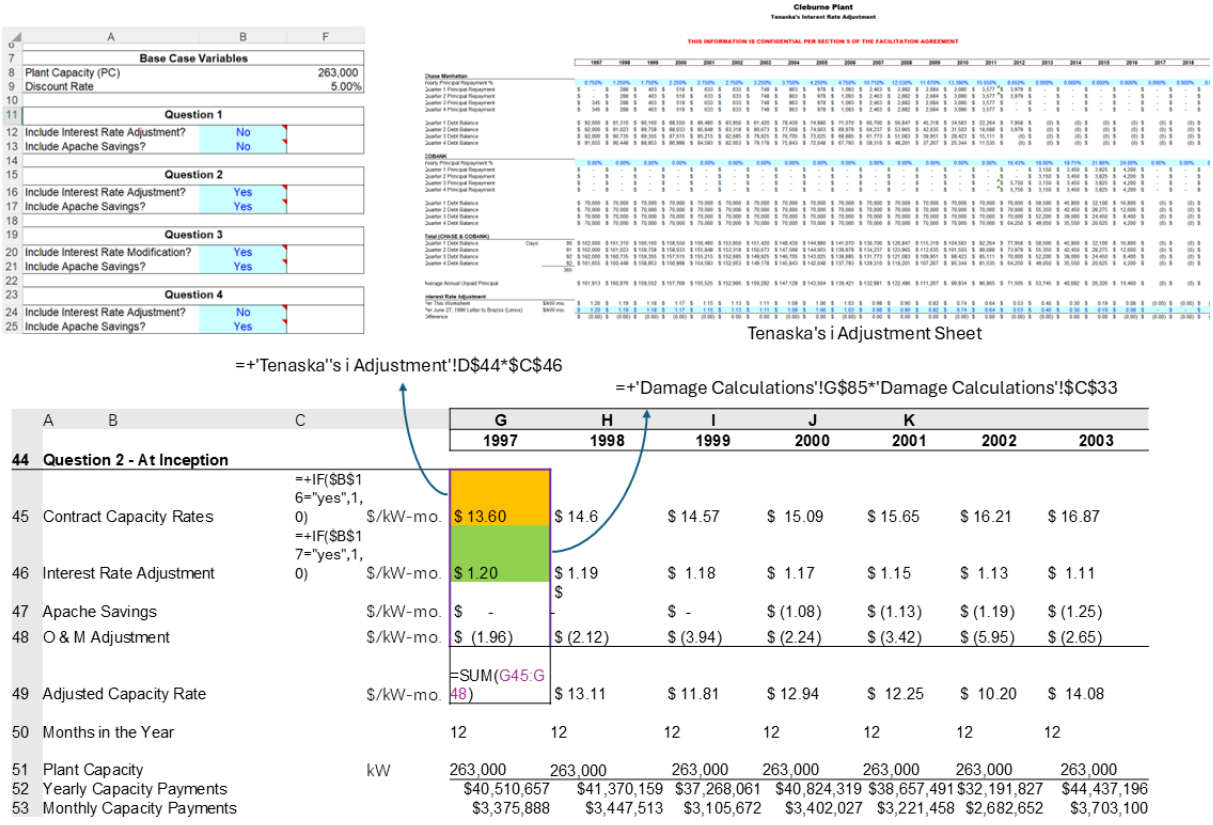


Figure 10: This task requires deriving the XNPV5 of the contract under different combinations of assumptions. The analysis uses contract capacity rates, plant capacity, and the specified discount rate provided in the table. While key adjustment components—namely the Interest Rate Adjustment, Apache Savings, and O&M Adjustment—must be retrieved from supporting documents and applied according to each scenario (included, excluded). For each assumption set, the analyst must then construct the annual capacity payment cash flows by deriving the adjusted capacity rate, converting it into monthly and annual capacity payments, and assembling the full month-by-month cash-flow schedule. Only after these intermediate steps are completed can the cash flows be discounted to the valuation date (e.g., December 31, 2000) to compute XNPV5.

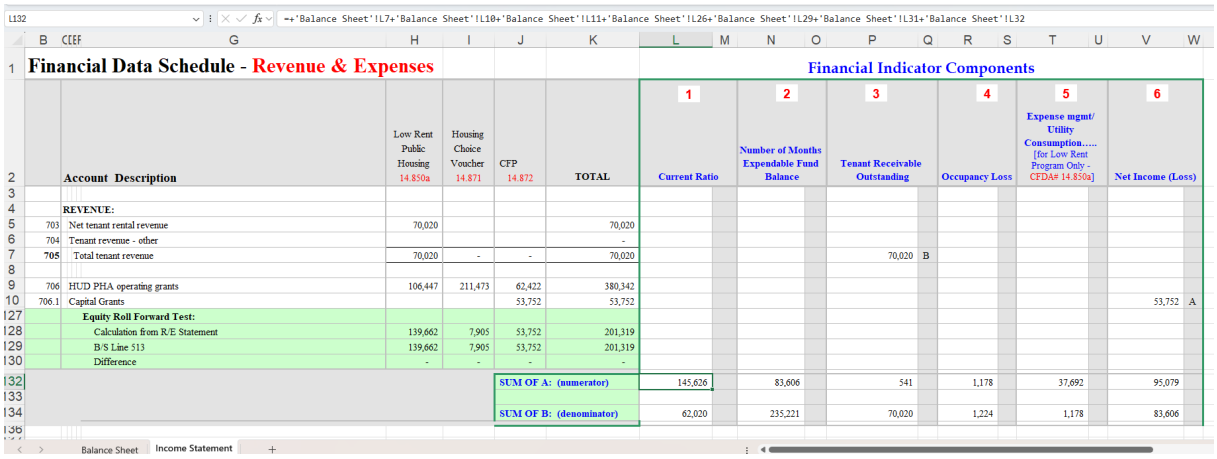


Figure 11: The sum of A&B and the equity roll-forward test requires cross-sheet retrieval and calculation.

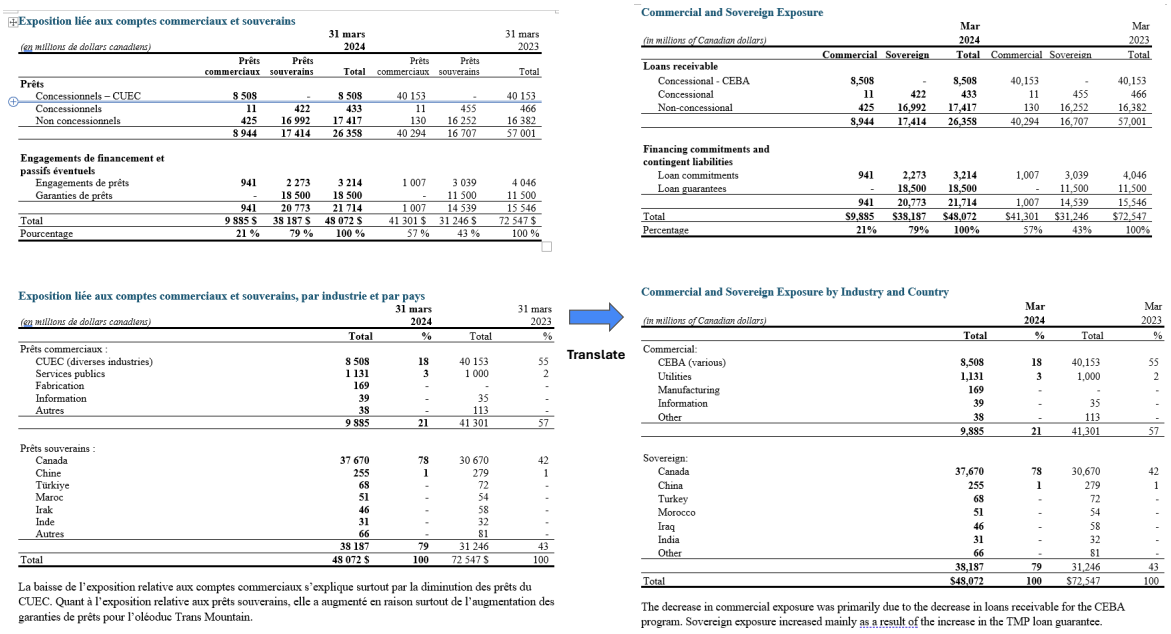


Figure 12: A workflow that translates a French report into English while preserving its format and structure. The report contains many tables to translate, along with text, notes, and even charts.

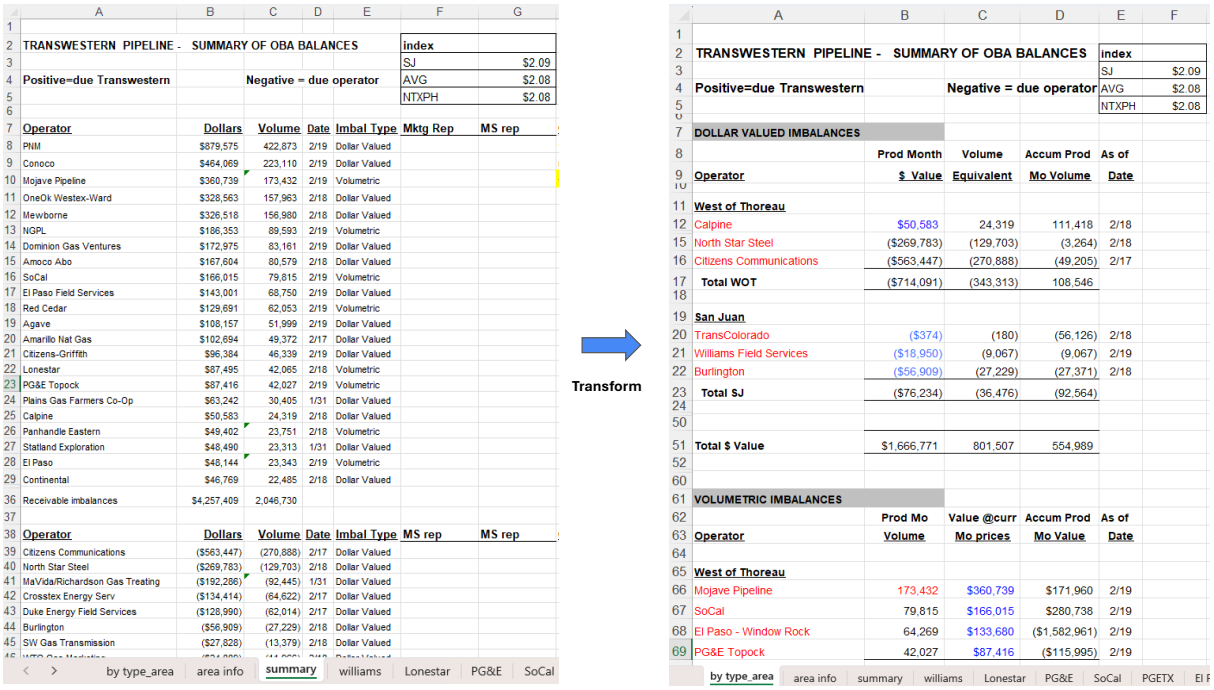
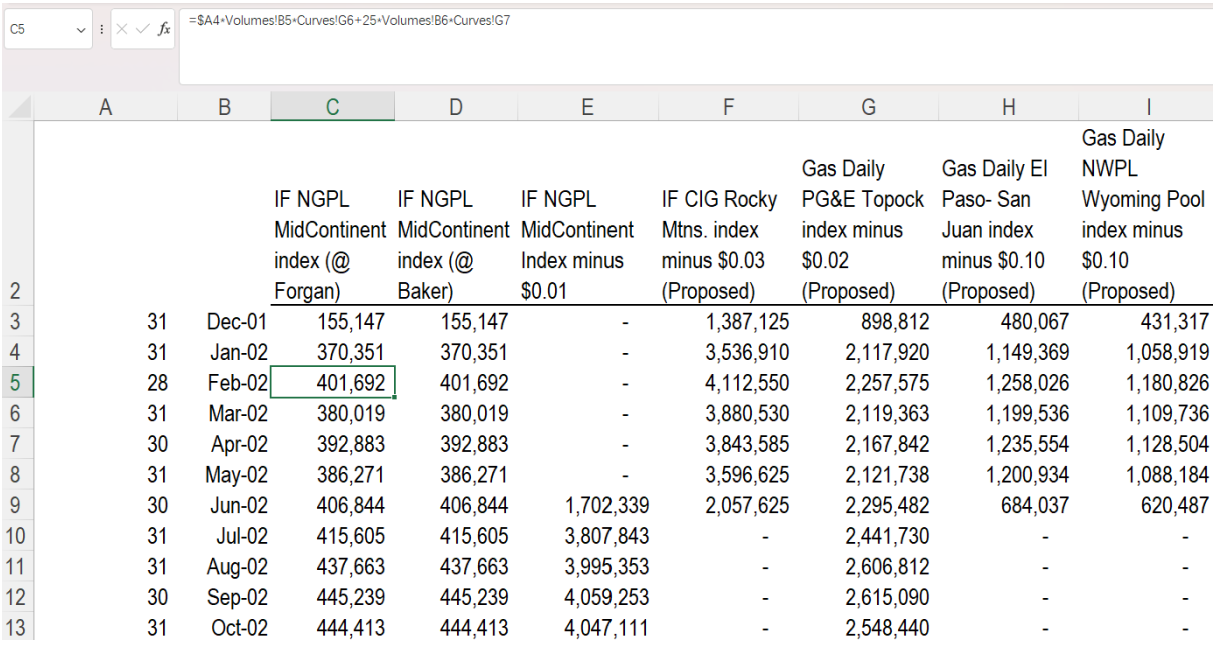


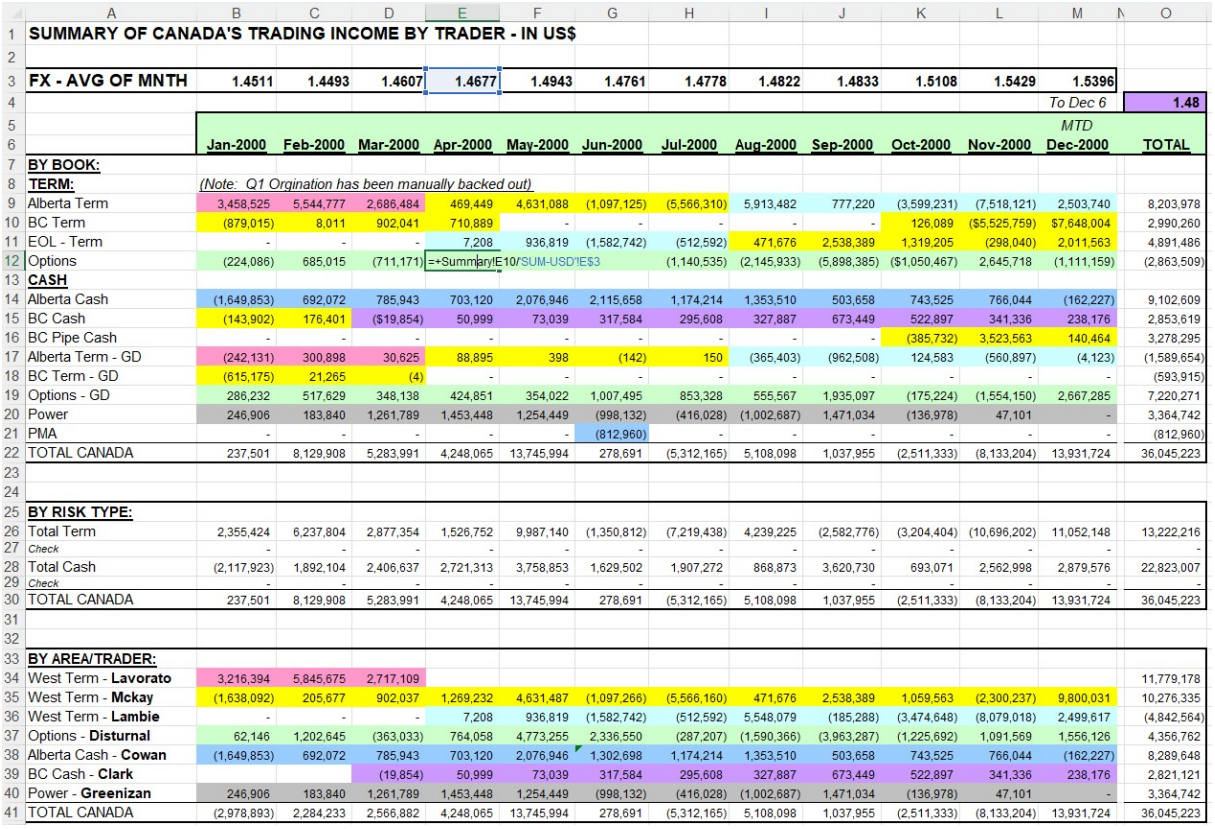
Figure 13: Transforming a table from one structure to another requires reorganizing data and retrieving information across sheets (e.g., `area_info` and `summary`). This example poses additional challenges: (1) distinguishing value-driven operators from volume-driven operators, and (2) performing aggregation and validation over the reorganized data.



C5: =$A4*Volumes!B5*Curves!G6+25*Volumes!B6*Curves!G7

	A	B	C	D	E	F	G	H	I
2			IF NGPL MidContinent index (@ Forgan)	IF NGPL MidContinent index (@ Baker)	IF NGPL MidContinent Index minus $0.01	IF CIG Rocky Mtns. index minus $0.03 (Proposed)	Gas Daily PG&E Topock index minus $0.02 (Proposed)	Gas Daily El Paso- San Juan index minus $0.10 (Proposed)	Gas Daily NWPL Wyoming Pool index minus $0.10 (Proposed)
3	31	Dec-01	155,147	155,147	-	1,387,125	898,812	480,067	431,317
4	31	Jan-02	370,351	370,351	-	3,536,910	2,117,920	1,149,369	1,058,919
5	28	Feb-02	401,692	401,692	-	4,112,550	2,257,575	1,258,026	1,180,826
6	31	Mar-02	380,019	380,019	-	3,880,530	2,119,363	1,199,536	1,109,736
7	30	Apr-02	392,883	392,883	-	3,843,585	2,167,842	1,235,554	1,128,504
8	31	May-02	386,271	386,271	-	3,596,625	2,121,738	1,200,934	1,088,184
9	30	Jun-02	406,844	406,844	1,702,339	2,057,625	2,295,482	684,037	620,487
10	31	Jul-02	415,605	415,605	3,807,843	-	2,441,730	-	-
11	31	Aug-02	437,663	437,663	3,995,353	-	2,606,812	-	-
12	30	Sep-02	445,239	445,239	4,059,253	-	2,615,090	-	-
13	31	Oct-02	444,413	444,413	4,047,111	-	2,548,440	-	-

Figure 14: The apparent semantics from the headers suggest a monthly/daily exposure metric. However, inspecting the underlying formula (e.g., `C5=$A4*Volumes!B6*Curves!G7+25*Volumes!B7*Curves!G8`) reveals that it actually encodes a 55-day payment timing schedule. Models that ignore or underutilize formula information, therefore systematically misattribute the column's role in downstream computations, and this misinterpretation then propagates through subsequent steps.

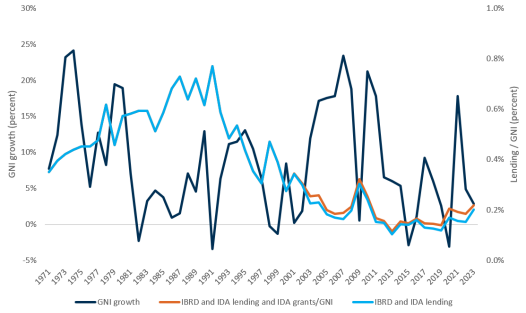


	A	B	C	D	E	F	G	H	I	J	K	L	M	O
1	**SUMMARY OF CANADA'S TRADING INCOME BY TRADER - IN US$**													
2														
3	**FX - AVG OF MNTH**	1.4511	1.4493	1.4607	1.4677	1.4943	1.4761	1.4778	1.4822	1.4833	1.5108	1.5429	1.5396	
4													*To Dec 6*	1.48
5													*MTD*	
6		Jan-2000	Feb-2000	Mar-2000	Apr-2000	May-2000	Jun-2000	Jul-2000	Aug-2000	Sep-2000	Oct-2000	Nov-2000	Dec-2000	TOTAL
7	**BY BOOK:**													
8	**TERM:**	*(Note: Q1 Orgination has been manually backed out)*												
9	Alberta Term	3,458,525	5,544,777	2,686,484	469,449	4,631,088	(1,097,125)	(5,566,310)	5,913,482	777,220	(3,599,231)	(7,518,121)	2,503,740	8,203,978
10	BC Term	(879,015)	8,011	902,041	710,889	-	-	-	-	-	126,089	($5,525,759)	$7,648,004	2,990,260
11	EOL - Term	-	-	-	7,208	936,819	(1,582,742)	(512,592)	471,676	2,538,389	1,319,205	(298,040)	2,011,563	4,891,486
12	Options	(224,086)	685,015	(711,171)	=+Summary!E10/SUM-USD!E$3			(1,140,535)	(2,145,933)	(5,898,385)	($1,050,467)	2,645,718	(1,111,159)	(2,863,509)
13	**CASH**													
14	Alberta Cash	(1,649,853)	692,072	785,943	703,120	2,076,946	2,115,658	1,174,214	1,353,510	503,658	743,525	766,044	(162,227)	9,102,609
15	BC Cash	(143,902)	176,401	($19,854)	50,999	73,039	317,584	295,608	327,887	673,449	522,897	341,336	238,176	2,853,619
16	BC Pipe Cash	-	-	-	-	-	-	-	-	-	(385,732)	3,523,563	140,464	3,278,295
17	Alberta Term - GD	(242,131)	300,898	30,625	88,895	398	(142)	150	(365,403)	(962,508)	124,583	(560,897)	(4,123)	(1,589,654)
18	BC Term - GD	(615,175)	21,265	(4)	-	-	-	-	-	-	-	-	-	(593,915)
19	Options - GD	286,232	517,629	348,138	424,851	354,022	1,007,495	853,328	555,567	1,935,097	(175,224)	(1,554,150)	2,667,285	7,220,271
20	Power	246,906	183,840	1,261,789	1,453,448	1,254,449	(998,132)	(416,028)	(1,002,687)	1,471,034	(136,978)	47,101	-	3,364,742
21	PMA	-	-	-	-	-	(812,960)	-	-	-	-	-	-	(812,960)
22	TOTAL CANADA	237,501	8,129,908	5,283,991	4,248,065	13,745,994	278,691	(5,312,165)	5,108,098	1,037,955	(2,511,333)	(8,133,204)	13,931,724	36,045,223
23														
24														
25	**BY RISK TYPE:**													
26	Total Term	2,355,424	6,237,804	2,877,354	1,526,752	9,987,140	(1,350,812)	(7,219,438)	4,239,225	(2,582,776)	(3,204,404)	(10,696,202)	11,052,148	13,222,216
27	*Check*	-	-	-	-	-	-	-	-	-	-	-	-	-
28	Total Cash	(2,117,923)	1,892,104	2,406,637	2,721,313	3,758,853	1,629,502	1,907,272	868,873	3,620,730	693,071	2,562,998	2,879,576	22,823,007
29	*Check*	-	-	-	-	-	-	-	-	-	-	-	-	-
30	TOTAL CANADA	237,501	8,129,908	5,283,991	4,248,065	13,745,994	278,691	(5,312,165)	5,108,098	1,037,955	(2,511,333)	(8,133,204)	13,931,724	36,045,223
31														
32														
33	**BY AREA/TRADER:**													
34	West Term - **Lavorato**	3,216,394	5,845,675	2,717,109										11,779,178
35	West Term - **Mckay**	(1,638,092)	205,677	902,037	1,269,232	4,631,487	(1,097,266)	(5,566,160)	471,676	2,538,389	1,059,563	(2,300,237)	9,800,031	10,276,335
36	West Term - **Lambie**	-	-	-	7,208	936,819	(1,582,742)	(512,592)	5,548,079	(185,288)	(3,474,648)	(8,079,018)	2,499,617	(4,842,564)
37	Options - **Disturnal**	62,146	1,202,645	(363,033)	764,058	4,773,255	2,336,550	(287,207)	(1,590,366)	(3,963,287)	(1,225,692)	1,091,569	1,556,126	4,356,762
38	Alberta Cash - **Cowan**	(1,649,853)	692,072	785,943	703,120	2,076,946	1,302,698	1,174,214	1,353,510	503,658	743,525	766,044	(162,227)	8,289,648
39	BC Cash - **Clark**			(19,854)	50,999	73,039	317,584	295,608	327,887	673,449	522,897	341,336	238,176	2,821,121
40	Power - **Greenizan**	246,906	183,840	1,261,789	1,453,448	1,254,449	(998,132)	(416,028)	(1,002,687)	1,471,034	(136,978)	47,101	-	3,364,742
41	TOTAL CANADA	(2,978,893)	2,284,233	2,566,882	4,248,065	13,745,994	278,691	(5,312,165)	5,108,098	1,037,955	(2,511,333)	(8,133,204)	13,931,724	36,045,223

Figure 15: This workflow requires creating a new spreadsheet with all values converted to USD. It also requires correct in-sheet and cross-sheet formula references while preserving the original spreadsheet layout.

Figure 1.4 GNI Growth Versus Ratio of New World Bank Lending to Gross National Income, Low- and Middle-Income Countries, 1971-2023

Percent	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987
GNI growth	8%	12%	23%	24%	14%	5%	13%	8%	19%	19%	7%	-2%	3%	5%	4%	1%	2%
IBRD and IDA lending and IDA grants/GNI	0.35%	0.40%	0.42%	0.44%	0.45%	0.45%	0.48%	0.62%	0.46%	0.57%	0.58%	0.59%	0.59%	0.51%	0.59%	0.68%	0.73%
IBRD and IDA lending	0.35%	0.40%	0.42%	0.44%	0.45%	0.45%	0.48%	0.62%	0.46%	0.57%	0.58%	0.59%	0.59%	0.51%	0.59%	0.68%	0.73%



but also implicit ex ante debt relief and financial support. Most IDA credits carry a zero or very low interest rate, and repayments typically extend over 30–50 years; however, more than one-third of IDA-eligible countries receive all or part of their IDA resources in the form of grants that carry no repayments in the future. Whereas IDA focuses on the most impoverished nations, the World Bank's other lending arm, IBRD, has played a crucial role in coordinating responses to regional and global challenges by providing loans and financial services to middle-income and creditworthy low-income countries (figure 1.4). IBRD was created to support countries rebuilding after World War II and has continued its crisis and emergency support through increased lending to countries affected by other crises since then, including the 2008–09 financial crisis, the 2014 Ebola outbreak, and the COVID-19 pandemic. Since inception, IBRD and IDA lending has responded positively to adverse external shocks affecting the economies of countries eligible for such financing, and this countercyclical lending has been a recurring and stabilizing response to dramatic drops in economic growth in these economies over the years.

Figure 16: Generating reports from tabular data requires financial knowledge of data analysis, financial events, and visualization. For example, one may plot two series with different units on a single chart (e.g., using a secondary y-axis) to reveal their correlation.

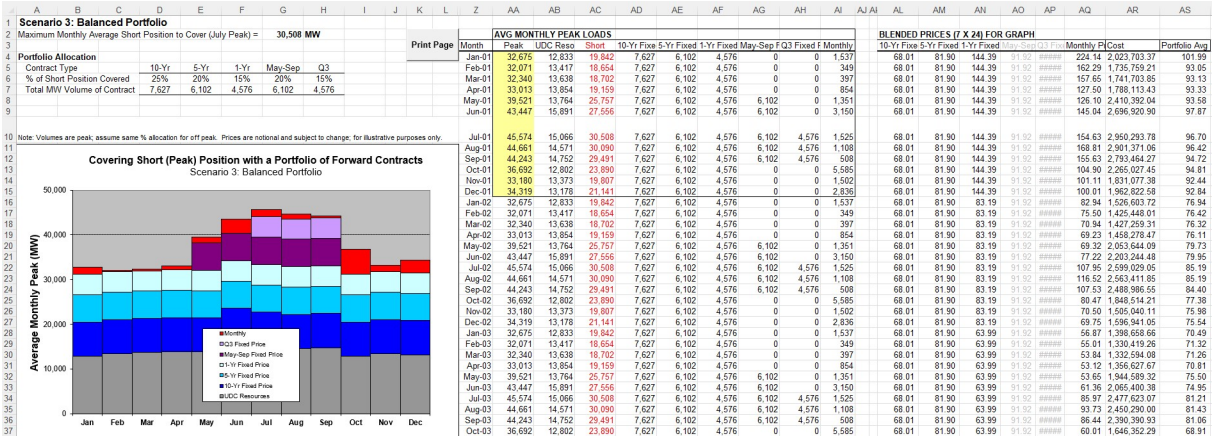


Figure 17: This Excel sheet shows an assumption-update workflow, where a mix of forward contracts is used to cover monthly peak-load short positions. It lists the contract allocations and MW volumes, along with monthly peak loads and the resulting short MW. A table on the right computes blended prices and portfolio costs, and the stacked chart visualizes coverage by contract type over the year.