# New Research: The Best AI Model Passes Just 38.3% of Real Spreadsheet Tasks on the GDP.xlsx Benchmark

Hand a frontier model a real workbook and ask it the question a colleague would ask. Across 20 configurations, the best one answered acceptably in 38.3% of attempts. The median managed 19.3%.

THE SHORT VERSION

GDP.xlsx, posted to arXiv on 13 November 2026 by Suhaas Garre, Sushant Mehta and Edwin Chen, gives agents 70 workplace requests written by professionals across 13 domains, each with the complete workbook or CSV file and an expert rubric of binary criteria (791 in total, across files holding 2.18 million populated cells). Under strict pass@1 the top configuration, Gemini 4 Argon, passed 38.3% of attempts and the median configuration passed 19.3%.

## Seventy questions with no single reference string to match against

The paper is GDP.xlsx: Benchmarking Frontier Models on Professional Workbooks, arXiv:2611.17766v1, 14 pages, 4 figures, 6 tables, accepted to the Agent Evaluation Science Symposium. It belongs to a family of benchmarks from the same group built on the materials professionals work from: GDP.pdf for documents, Chartography for charts, HANDBOOK.md for policies, DAYJOB for long-horizon workspaces.

The construction detail matters more than the headline number. Each task was written by a domain professional starting from a workflow in which frontier models had already failed them. The author wrote the request the way they would ask a colleague, prepared the input file (public data, or modeled on the files they work with), and the task then passed three

rounds of review covering the file, the request, the expected reasoning and the criteria. The inputs are 58 Excel workbooks and 12 CSV files. Sizes run from a few dozen populated cells to more than 300,000, with a median of about 3,600, and all but six of the 58 workbooks have more than one worksheet.

Requests are short and underspecified the way real requests are: 58 of the 70 never use the word “tab” or “sheet”. The public equipment-register task reads, in full, “What is the location and the cabinet number of the DC auxiliary power cabinet. it should start +MDY…” and points into a workbook of roughly 16,000 populated cells. The cabinet is labeled only in a diagram pasted onto a worksheet.

That is the part I find genuinely interesting. The benchmark treats formatting as data. Figure 3 counts 13 structural and formatting features across the 58 workbooks, and the median workbook carries four of them. In 21 tasks the answer depends on layout or on those features. In the rent-forecast task, comments record concessions and an AutoFilter hides a tenancy the answer must include. In held-out tasks, strikethrough, fills, data validation and shapes drawn over cells mark records that must be treated differently. Load that workbook into a pandas DataFrame and you keep the values and lose the evidence.

## Why the pass rate looks so low next to other agent benchmarks

The grading is unforgiving on purpose, and the unit of success is the whole answer.

An agentic judge scores each attempt: the OpenCode agent, with the same judge model and instructions for every attempt of every configuration. In one session it receives the request, the input filename and all criteria. It may open the file and run code to verify claims, but it may not complete the task itself or credit content the response does not contain, and it must write its reasoning before each binary verdict. Only the agent’s final text response is graded, never any file it produced, which separates comprehension from file production.

Rubrics run from 3 to 30 criteria, about 11 on average. Here is the mechanism that drives the score down: 39 of the 70 rubrics contain both a completeness criterion (every member of a set) and an exclusivity criterion (nothing extra). The answer needs full recall and full

precision at the same time. Strict pass@1 means an attempt passes only if every criterion is met, so a response that is right about nine things and misses one item of a list scores zero.

The failure analysis in section 5.2 shows what that looks like. On the rent task, Muse Spark 1.3 listed the tenancies with ending discounts and left out the three that qualify only through a tenant break, which sit on the BreakClauses tab with one of them hidden behind an AutoFilter. On the data-dictionary task, the answer omitted five of the eight housebound codes. Both answers had perfect precision. Both failed.

**38.3%**
strict pass@1, Gemini 4 Argon
best of 20 configurations, launch leaderboard 30 Sep 2026; task-sampling SE at most 5.8 points

**19.3%**
median configuration
arXiv:2611.17766, 20 configurations from 12 developers

**39 / 70**
tasks whose rubric demands full recall and full precision on a set
completeness plus exclusivity criteria, paper §3.5

**21 / 70**
tasks where the answer depends on layout, formatting, comments or embedded objects
paper §3.3, structure-and-format demand

Each attempt had up to 60 minutes, 200 turns, and a container with Python (openpyxl, pandas), LibreOffice for recalculation and rendering, and Tesseract OCR. Vision-capable models could view images extracted from a workbook. The container could reach only the model provider's API, so every piece of evidence had to come from the file. These models were not starved of tools. They had an hour with the file and the libraries needed to enumerate every tab, comment, filter and drawing in a few lines of code, and they still missed the evidence.

## Gemini 4 Argon sits 28 points above Gemini 3.1 Pro and still fails three attempts in five

Second and third place go to Claude Opus 5.5 (30.3%) and Claude Sonnet 5.5 (29.1%). No other configuration in the launch set exceeds 25%. Where the leaderboard carries two versions of the same model (Claude Sonnet 5 and 5.5, GPT 6 Sol and GPT 6.1 Sol), the newer

one scores higher.

So the curve is moving. A secondary leaderboard tracker, reading the official board on 26 November 2026, lists 36 models with Gemini 4 Neon on top at 43.1%, Gemini 4 Hydrogen at 40.9% and Gemini 4 Carbon at 38.9%. Those figures come from the live leaderboard rather than the paper's tables, and the paper's own evaluation covers the 20 configurations as of September 2026. Treat 43.1% as current board state, not as a result verified against the paper.

Even taking the best of them, the shape holds. The strongest known configuration fails well over half of attempts on questions a competent analyst with the file in front of them would answer.

The paper states a task-sampling standard error of at most 5.8 points on the 38.3% figure. The AI Eval Watch review for 11 to 17 November puts it more bluntly: a pass rate near 38% on 70 items carries a 95% margin of error of roughly 11 percentage points, before any run-to-run variation. A three-point gap on this leaderboard is not a ranking. The same review notes that the abstract does not say whether the judge's verdicts were checked against human graders, and that holding 67 of 70 tasks back (reasonable for contamination control) means nobody outside can rerun the full evaluation.

The paper is candid about headroom. It is also worth reading the scoring rules closely. Attempts that error out, including the 60-minute timeout and provider failures, are excluded from the task's pass rate, while an attempt that hits the 200-turn limit or trips the SDK's stuck detector is graded as an empty response. Since every rubric contains an affirmative or completeness criterion, an empty response passes nothing. Those two rules pull in opposite directions, and I have not seen the counts for either category in the extract I could retrieve.

WHERE I LAND

*My take: this is a harness result as much as a model result. The paper says so almost explicitly in its discussion. For vision-capable models the harness already provides a path to all the missed evidence, because a few lines of Python list every tab, comment, filter and embedded image. The models had the tools and did not use them systematically. That gap between capability available and capability exercised is what scaffolding closes, and it is the*

*same gap I wrote about in the SWE-bench harness piece. My prediction, and it is only that: a spreadsheet-specific agent that enumerates structure before answering, then checks its answer set against each condition in the request, lands materially above 38.3% on these same tasks without a new base model. I would be surprised if nobody ships that within two quarters.*

## What a 19.3% median means for an automation budget

The median configuration is the relevant number for most buyers, because most deployments do not run the single best frontier configuration at maximum reasoning effort with an hour per query. A 19.3% strict pass rate means roughly four in five answers fail at least one criterion a domain professional said mattered.

The failure mode is worse than the rate suggests. These are not obviously broken outputs. Both analyzed failures had perfect precision: every item listed was correct. The answer looks complete, cites real rows, uses the right vocabulary, and silently omits three tenancies or five form codes. A reviewer who does not already know the answer cannot tell the difference without redoing the work. **An answer with perfect precision and missing set members reads exactly like a correct answer.** That makes this failure pattern unsuited to any workflow where a human gives the output a quick sanity check and moves on.

Three things I would change in how a team specifies this work.

Separate retrieval from reasoning in your pipeline and evaluate them separately. The paper's failure analysis points at retrieval every time: evidence on an unexamined tab, a diagram the agent never looked at, a filter hiding a row. If your spreadsheet agent loads a workbook into DataFrames and reasons over values, it has already discarded the evidence for the hardest 21 of 70 tasks. Make structure extraction an explicit, logged step: enumerate tabs, comments, filters, data validation, shapes and embedded images before anything else.

Score at the unit the user cares about. The same week's eval roundup covered SWE-CoWork, where the two strongest coding agents passed about 63% of fail-to-pass tests but fully completed fewer than 8% of assignments. Same runs, two stories, depending on

the unit. A per-criterion score on GDP.xlsx would look far healthier than 38.3%, and it would mislead you. For comparison on difficulty, Visual Cipher Bench (arXiv:2611.16861, 12 November 2026) had its best agent solve 10.7 of 40 long-horizon visual reasoning tasks (26.7%), with GPT-6.1 Sol and Claude Opus 5.5 tied at 21.7%.

Build the completeness check into the task rather than the review. What the models get wrong is deciding when a set is finished. If the deliverable is a list, the agent should state the rule that defines membership, then show the scan it ran over every candidate row against that rule. Make it produce that artifact and a reviewer has something falsifiable to check in a minute. Without it they have to redo the query.

## What GDP.xlsx does not measure

It does not say spreadsheet automation is worthless. GDP.xlsx grades one hard thing: answering an underspecified question about an unfamiliar workbook, where the answer is usually a set and the evidence is scattered. Building a model, writing formulas, transforming known data, generating a report from a known layout: none of that is measured here, and separate benchmarks (SpreadsheetBench, BlueFin, Finch) cover parts of it.

It also does not include a human-expert baseline I could retrieve from the indexed extract. The authors are domain professionals who wrote tasks from workflows where models had failed them, so the tasks are selected for difficulty against models. Whether a professional unfamiliar with a given workbook would hit 100% under the same rubric and the same hour is not something the retrievable sections answer. My assumption is that a professional would score well above 38.3% and below 100%, but that is an assumption, not a finding, and anyone citing this paper in a procurement argument should say so.

What is confirmed: as of September 2026, across 20 configurations from 12 developers, with five attempts per task, an hour per attempt, a full toolchain and complete files, no configuration answered even two in five of these questions to a professional's satisfaction. If your 2027 plan assumes an agent reads the workbook and tells the team what it says, that plan is ahead of the evidence by a wide margin, and the evidence was published this month.

WHAT TO DO WITH THIS

**Before you budget for spreadsheet agents, run your own ten hardest real requests through your stack and grade them with a binary rubric written by the person who normally answers them, completeness criterion included. If you want help designing that rubric so the score survives scrutiny, book a call →**



WRITTEN BY

Artur Markus

Software engineer in Basel building AI infrastructure and agentic systems, with a background in pharmaceutical automation, GxP compliance, and validation. I help teams turn AI from a chatbot into a reliable operational layer.

Book a meeting →