



THE LAB: BUILD YOUR PERSONAL AI BENCHMARK

BUILD YOUR PERSONAL AI BENCHMARK · FRONTIER LAB · OCTOBER 1, 2026

Nufar Gaspar. Frontier Lab with The AI Daily Brief and Superintelligent. October 2026.

A new AI model or tool arrives every few weeks, with test scores and a week of hot takes. None of it tells you whether it is better at your work. A **personal AI benchmark** does. It is a small set of your own real requests. You give them to several AI options side by side, with the names hidden, and see which answer you would use.

This lab is self-paced, and you can do all of it without the recording. You leave with three things:

1. **A saved task list:** four to six real requests you reuse every time something new comes out.
2. **A blind comparison:** every option's answer to every request, names hidden, with your pick and one line on why.
3. **A decision:** your AI lineup, or a verdict on something new.

One rule before you start: nothing private. You will paste these requests into many tools over the years. Use realistic made-up material with invented names and numbers. Prompt 1 writes it for you.

Parts 1 and 2 are the core, complete on their own. Then come an extra for early finishers and two optional parts for going deeper, with their prompts included.

PART 1. CHOOSE TASKS THAT COVER WHAT YOU REALLY DO

This is the most important step. Your benchmark can only tell you about the work it contains. If every task is an email, you learn which AI writes the best email, and that is all you learn.

The Work Mix: six questions

The **Work Mix** is six questions that check your set covers what you really do. A good set of five or six tasks touches most of them.

QUESTION	COVER BOTH ENDS	FOR EXAMPLE
What kind of work?	Writing, analysis, research, decisions and advice, planning and admin, making things	A board update. Reading a sales report. A brief on a competitor. A second opinion on a hire. A trip plan. A slide or an image.
How often?	Daily habits, and rare moments that matter	Your weekly team email, and the annual budget memo
What's at stake?	Quick, low-risk asks, and ones where a weak answer costs you	A lunch poll for the team, and a note to a client who is about to leave
Where does the AI start?	A blank page, or your own material	"Draft a welcome note" compared with "turn these meeting notes into a list of decisions"
Which part of life?	Work and personal	A hiring plan, and a birthday party for an eight-year-old

QUESTION	COVER BOTH ENDS	FOR EXAMPLE
What's on your wish list?	Something you wanted AI to do and it couldn't	"Turn my messy strategy memo into slides I would present as they are"

Four habits make the set stronger:

- **Pick tasks where you can tell when an answer is off.** You will judge the answers yourself, so choose work you know well enough to spot a weak, generic, or wrong answer quickly. A task where every answer looks fine to you cannot separate the options.
- **Include one matter-of-taste task,** where the best answer is simply the one you prefer. A message in your own voice is the classic. This is where differences between options are easiest to feel.
- **Include one task you can check,** where an answer is plainly right or wrong. Figures pulled from a table work well.
- **Weight the set toward what fills your week.** If most of your AI use is writing, two writing tasks is a fair share. Five would crowd out everything else.

A well-covered set: Maya's six tasks

Maya is chief operating officer of a 300-person logistics company. She uses AI most days, mainly for writing and for thinking through decisions. Here is a set that covers her real work.

#	TASK	WHY IT EARNS ITS PLACE	WHAT IT COVERS
1	Turn her rough notes into the weekly leadership update, in her voice	Her most frequent request, and a matter of taste	Writing, weekly, her own material, taste
2	Find the story in a monthly table of on-time deliveries by region: three findings, and one question for each regional lead	Numbers she can check	Analysis, monthly, her own material
3	Argue against a proposal to close the smallest warehouse, and list what she would need to know before deciding	A rare decision with real consequences	Decisions and advice, rare, high stakes
4	Plan a difficult conversation with a manager who has missed targets twice	Judgment and tone, where a weak answer hurts	Advice, rare, high stakes, blank page, taste
5	Plan a three-day family weekend with two teenagers and a grandparent who walks slowly	Her main personal use of AI	Planning, occasional, personal
6	Turn a four-page strategy memo into five slides she would present as they are	Her wish: it has never worked well for her	Making things, rare, her own material, wish list

The set has no research task. That fits, because Maya rarely asks AI to research. The set looks like her life.

A lopsided set, and how to fix it

Here is the set Maya first wrote down:

1. Summarize a long email thread.
2. Draft a reply to a customer complaint.
3. Turn meeting notes into action items.
4. Rewrite a paragraph to sound more professional.
5. Draft the weekly leadership update.

All five are writing, daily, low stakes, from her own material, and work. There is no wish. Every good AI handles these well, so the result would be a tie that tells her very little.

The fix: keep the two she does most (3 and 5). Swap the other three for the warehouse decision, the delivery table, and the strategy slides. Each swap fills an empty row of the Work Mix.

Prompt 1: Find my tasks

Paste this into the AI tool that knows you best. If it has memory of you or access to your files, it starts from what it knows. If it knows nothing about you, it interviews you from the start. Answer in your own words, and correct it when it guesses wrong.

COPY THIS PROMPT

I want to build my personal AI benchmark. That is a small saved set of real requests from my own life, which I will give to several AI tools side by side, with the names hidden, to see which one does my work best. I will reuse the same requests, unchanged, every time a new AI model or tool comes out. Help me choose those requests by interviewing me.

A good set covers the mix of what I really do. Use these six questions as your map of my work:

1. What kind of work is it? Writing; analysis of numbers or documents; research; decisions and advice; planning and admin; making things such as images, slides, or web pages.
2. How often does it happen? Daily habits, and rare moments that matter a lot when they come.
3. What is at stake? Quick, low-risk requests, and ones where a weak answer would cost me.
4. Where does the AI start? From a blank page, or from my own material such as notes, a draft, a spreadsheet, or a long document.
5. Which part of my life is it? Work, or personal life.
6. Is it on my wish list? Things I have wanted AI to do for me that it could not do, or could not do well.

How to run the interview:

- Start with what you already know. If you have memory of me, our past conversations, or access to my files, begin by summarizing in a few lines what you think I use AI for, including the requests I seem to repeat or go back and forth on. Then ask me what you got wrong. If you know nothing about me, say so and start asking.
- Ask one question at a time, in plain language, and wait for my answer. Keep each question short.
- Start broad: my role, what a normal week looks like, and what I already use AI for. Then go deeper on the parts of the map my answers have not reached.
- Every few answers, tell me in one sentence which parts of the map are covered so far and which are still empty. Then ask about an empty one.
- Useful questions to draw on: Which request do you make most often? Where do you usually rewrite the answer, or ask several times before you are happy? Which rare task matters a lot when it comes, such as a board update or a difficult conversation? Where would a wrong answer embarrass you or cost money? What do you use AI for outside work, if anything?
- Near the end, always ask these three, one at a time: What have you always wanted AI to do for you that it could not do, or could not do well? What has frustrated you with AI recently? Which parts of your work do you still not trust AI with, and what would change your mind?
- Ask twelve questions at most. Stop earlier if the map is well covered or I say I am done.

Then propose my set: three to five everyday tasks (things I really do, regularly or at important moments) plus one or two wish-list tasks, with six tasks at most in total. Choose them so the set looks like my real life, and prefer tasks where I know the work well enough to tell quickly when an answer is off, because I will be the judge: several kinds of work, at least one rare but important moment, at least one where a weak answer would cost me, at least one that starts from my own material, and at least one where the best answer is a matter of taste, such as a message in my own voice. Weight the set toward the work that fills my week. Include a personal task only if I use AI outside work. If I use AI tools that can work with my files or take several steps on their own, include one task of that kind.

For each task, give me:

- A number and a short name.
- One line on why it earns its place, and which parts of the map it covers.
- The complete request I will paste, ready to use as it is. Where the request needs background material, such as meeting notes, a draft, figures, or an email thread, write realistic made-up material directly into it, as messy and specific as the real thing, so I never need to paste anything private or confidential. Keep each request short enough to paste into any chat window.
- A short scoring guide: two or three plain lines on what a great answer must include and what would make me

reject it. To write it, ask me one question per task: how would I tell a great answer from a weak one here? Use my own words. If I cannot answer, tell me the task may not belong in the set, because I will be the judge.

- If the task asks for an image, slides, or a web page, a reminder that I will need to save those results in a folder to compare them.

Finish with a short table showing each task against the six map questions. Name any part of the map that is still missing, and say whether that matters for someone like me.

Then save it in the shape that fits where we are. If you can create files, make a folder called `my-ai-benchmark` with three things in it: a `README.md` that lists my tasks by number and name and leaves space for my lineup; a `tasks` folder with one file per task, named like `01-weekly-update.md`, each with three headings in this order: "Why it is here" (one line), "Request" (the complete request to paste, with any made-up material written into it), and "Scoring guide" (the two or three lines we agreed); and an empty `answers` folder for the answers I collect later. If you cannot create files, give me everything in one document, numbered, with the requests complete, and I will save it myself. Before you save anything, ask me to review the set: for each request, is this something I would really send, in these words, with material like this? Ask what to swap, what to sharpen, and what is missing from my real week. Revise until I say the set is mine. Only then save it, and remind me to keep these exact requests unchanged from that point on, because I will paste them every time something new comes out.

Make it yours before you save it. The list is a draft of your work, written by a tool that knows only what you told it. Read every request as if you were about to send it for real. Ask three things of each: Would I really ask this, in these words? Is the material close enough to my real material that a weak answer would show? Does this set, taken together, look like my week? Would I notice quickly if an answer here was off? Swap anything you would never send. Sharpen a request that is vaguer than your real ones. Add material that is closer to yours, with names and numbers changed. Then hold the set up against Maya's lopsided set: if most of your tasks are the same kind of work, swap one for an empty part of the map. The prompt asks you for these corrections before it saves; give it real ones. A benchmark that tests tasks you never do tells you nothing, however neat it looks.

Say what good looks like, before you see a single answer. For each task the prompt asks you one question: how would you tell a great answer from a weak one? Your reply becomes a short scoring guide, two or three lines, saved with the task. In the taste test you only glance at it before you pick. Written now, it is a commitment. Written after you have read the answers, it would only explain the choice you already made. And if you cannot say what good looks like for a task, you will struggle to judge it, so swap it for one where you can.

Done: four to six numbered requests, each ready to paste and each with a line on why it is there, saved in one of two shapes:

- **In a chat tool:** one document with the numbered requests and a results section at the bottom. Keep it where you keep notes.
- **In a tool that can create files** (Claude Code, Codex, Cursor, Cowork): a folder called `my-ai-benchmark`. It holds a `README` that lists the tasks, a `tasks` folder with one file per task, and an empty `answers` folder. Each task file holds why it is here, the request, and a short scoring guide. Prompt 1 builds it for you. The same folder feeds the blind-shuffle script in Step 3, and later the automated version, with no reformatting.

The requests are plain text with the material written in, so they work in any chat window. You never need an agent tool to run the taste test.

PART 2. THE TASTE TEST

The **taste test** is the core method. You run your saved requests in every option, hide the names, pick the answer you would use, and write one line on why. It is enough to make a real decision, and it sharpens your sense of a good answer far better than one quick try.

A word before you start: this part is done by hand. You paste, you collect, you hide the names, you pick. For most people that is the right way, and doing it by hand once teaches you what a good answer looks like faster than anything else. If you have more tooling, here is where it helps:

Chat tools only	Nothing. Parts 1 and 2 are built for you.
An AI agent tool such as Claude Code, Codex, or Cursor, but the options you are comparing are chat products	Steps 2 and 4 stay by hand, because no tool can paste into another product's chat for you. Your agent tool can file each answer into the <code>answers</code> folder and run the blind-shuffle script, which takes the fiddly part away.
A key that lets a program use AI models (Part 4 shows how to get one), and your options are models it can reach	Part 4 runs every request in every option and hides the names. You score and reveal on a page in your browser. Skip ahead if you want, and come back here for anything that only runs in a chat product.

Step 1. Choose your options

An **option** is anything you can compare. It can be a model, such as Claude, ChatGPT, Gemini, or Copilot. It can be a setting inside one product, such as its fast, thinking, or top option. Or it can be an AI agent tool, such as Claude Code, Codex, or Cowork. Agent tools work through several steps on their own, with your files and apps.

Pick two to four options you can open today. Four is the most that stays pleasant by hand. Which ones depends on why you are here:

- **Finding your lineup.** You are doing this for the first time, or you have no clear favorite. Compare, for example, the fast, thinking, and top options in the product you already use, or the top option from three companies.
- **Testing something new.** A new model or tool just came out. Compare it, or several new releases at once, with what you use today.

Done: the names of your options are written at the top of a new document.

Step 2. Run every request in every option

For each saved request, open a fresh chat in each option and paste the request exactly as saved. Keep the settings the same everywhere: web search on or off, memory on or off, and the same thinking setting where the product offers one. Copy each full answer into your document with the option's name above it. If you have the folder, save each answer as a file named after the option inside `answers/<task name>/`. Paste quickly and try not to read yet.

If one option searched the web and another did not, make a note. It is a difference in the test, and it can explain a surprise later.

Done: every request has one full answer from each option in your document.

Step 3. Hide the names

Brand preference is louder than most people expect. Once the names are hidden, all that is left to judge is the answer. Choose whichever of these four ways suits you.

- **A friend.** Send a colleague the answers with the names on. They send them back in a shuffled order, labeled A, B, C, and keep the key until you have picked. Return the favor.
- **A chat.** Paste the named answers into a fresh chat in a tool you are *not* testing, with Prompt 2 below. It removes the names, shuffles, relabels, and keeps the key until you type "reveal". No setup needed.
- **A spreadsheet.** Use one tab per request and one row per answer. Put the option's name in the first column, the full answer in the second, and the formula `=RAND()` in the third. Sort the rows by the third column, then hide the first. The top answer is A, the next is B, and so on. Unhide the first column to reveal.
- **A script,** for people comfortable running a small program. The blind-shuffle script in the materials copies your answers with the names hidden, shuffled separately for each request. It keeps the key in a separate file, records your picks, and counts the wins.

The script, step by step. Make a folder called `answers`, with one subfolder per request. In each subfolder, save one file per option, named after the option. Agent tools can save their answers straight into it. Put `answers` next to the `blind-shuffle` folder from the materials. Open the text window where you type commands in the folder that holds both, and run:

RUN THESE COMMANDS

```
python blind-shuffle/anonymize.py answers
python blind-shuffle/anonymize.py answers --pick 01-weekly-update B "sounded like me"
python blind-shuffle/anonymize.py answers --reveal
```

The first makes `answers-blind`, with every answer relabeled A, B, C, D. The second records one pick and your line on why; repeat it per request. The third shows the key, your picks, and the count of wins. Details are in `blind-shuffle/README.md`.

Prompt 2: Hide the names

Run this in a fresh chat, in a tool that is not one of your options.

COPY THIS PROMPT

I am running a blind taste test of AI tools, and I need you to hide which AI wrote which answer, so I can judge the answers on their own merits. We will go one request at a time. For each request, I will paste several answers, each labeled with the name of the AI model or tool that wrote it. I will paste quickly without reading them.

For each request: remove the names, and remove any line where an answer mentions who wrote it. Shuffle the order, using a fresh shuffle for every request. Relabel the answers A, B, C, and so on. Show me only the relabeled answers, in full, and otherwise exactly as written. Do not comment on them, rank them, or hint at which is which.

Keep the key to yourself. When I tell you my pick for a request, confirm that you have recorded it and ask for the next request. When I type "reveal", show me the full key, my pick for each request with the real names, and a count of wins for each AI.

Images, slides, and websites go in a folder. A chat or a spreadsheet cannot hold them. Save one file per option, or one folder per option for a website, named after the option, and let the script or a friend relabel them. Judge a website from a full-page screenshot, because a live link shows which tool made it in the web address. After the reveal, open the real thing: does it work, does it match what you asked, and how many follow-ups did it take? If a tool leaves a visible mark or a recognizable style, note it next to your pick.

Done: for each request you see answers labeled A, B, C, and you do not know which is which.

Step 4. Pick, and write one line on why

Read every answer before you judge any. It is hard to say how good the first answer is until you have seen the second and the third. Judging is a comparison, and the answers teach you what good looks like for this request. Your scoring guide is a checklist to glance at; the side-by-side read is where the decision happens.

Then ask yourself one question: which of these would I use as it is? Choose one letter. With three or four options, you pick a single winner.

Want more than one winner per request? Score them. Give each answer a score from 1 to 5: 5 means you would use it as it is, 3 means usable after real edits, 1 means it misses the point. Scores show you how close the race was, and they let a second and third place count. The review page in the automated version works this way. One warning from experience: be stingy with top marks. If everything gets a 4 or a 5, the scores stop telling you anything.

Then write one line on why, in concrete words: "sounded like me", "caught the risk I missed", "half the length and nothing lost". Also note whether the win was clear or close.

These one-line reasons are your taste, written down. After a few rounds they start to repeat, and the repeats tell you what you value in an answer.

Done: for every request, one letter, one line on why, and the word "clear" or "close".

Step 5. Reveal and count

Reveal the names: ask your friend for the key, type "reveal" in the chat, unhide the column, or run the script's reveal. Count the wins for each option.

If an option you were rooting for lost, notice the feeling. It is the reason you hid the names.

Done: a count of wins for each option.

Step 6. Decide

The taste test judges one thing: the finished answer. Cost, speed, and how much you enjoy a tool are real reasons to choose, and they come in now, with the names visible. Think in three steps:

1. **Must-haves.** Am I allowed to use it at work? Am I comfortable with its data and privacy terms? Is it on my plan? An option that fails here is out, however good its answers were.
2. **The answers.** Your taste test: who won, and was each win clear or close?
3. **Tie-breakers.** Cost, speed, the features around the model (voice, connections to your other apps, memory, a good mobile app), plan limits, and the effort of changing your habits.

The rule: **a clear win on tasks that matter moves you. A close call goes to the tie-breakers.**

For example, say a new model comes a close third, and you love the voice mode in its app. Staying in that app is a sound decision. If it had come a clear first on one kind of task, you would send that kind of task to it and keep the rest where they are.

In one line: *the benchmark judges the answers, you judge the experience, and the decision uses both.*

What you end with depends on why you ran the test:

- **Finding your lineup** ends with **your AI lineup:** a short card that names your default AI and any kinds of task you send to another one.
- **Testing something new** ends with one verdict for each new option. **Switch:** it becomes your new default. **Split:** it joins your lineup for specific kinds of task. **Stay:** keep your lineup and test again next time.

Prompt 3 walks you through the decision and writes the result down. Run it in any chat.

Prompt 3: Make my decision

COPY THIS PROMPT

I just ran a blind taste test of AI options. I gave the same saved requests to several of them (AI models, settings inside one product such as fast or thinking, or AI tools), hid which answer came from which, picked the answer I would use for each request, wrote one line on why, and then revealed the names. Help me turn that into a decision. Interview me one question at a time and wait for each answer.

First, ask which situation I am in. Finding my lineup: I compared a few options to choose my default and to see which kinds of task to send elsewhere. Testing something new: I compared a new release with what I use today.

Next, for each request in turn, ask which option won, my one line on why, and whether the win was clear or close.

Then ask about what sits outside the answers. Must-haves: am I allowed to use each option at work, am I comfortable with its data and privacy terms, and is it on my plan? An option that fails a must-have is out, whatever its answers. Tie-breakers: cost, speed, features of the tool I enjoy (such as voice, connections to my other apps, memory, or a mobile app), plan limits, and the effort of changing my habits.

Decide with this rule: a clear win on a task that matters to me should move me, and a close call should be settled by the tie-breakers.

Then give me:

1. My AI lineup: a short card naming my default AI and which kinds of task, if any, go to another option.
2. If I was testing something new, one verdict for each new option, with one sentence of reasoning: Switch (it becomes my new default), Split (it joins my lineup for specific kinds of task; name them), or Stay (keep my lineup as it is and test again at the next release).
3. Three short notes on what I seem to value in an answer, written as qualities that will still make sense after products are renamed, such as "gets to the point in the first line".

Finish with the result that surprised me most, and one thing to retest the next time something new comes out.

Done: your AI lineup, plus a verdict for each new option if you were testing something new, saved under today's date next to your task list.

A finished result: Maya's lineup

Maya compared the top models from three companies, A, B, and C, on her six tasks. Here are her picks after the reveal.

TASK	WINNER	CLEAR OR CLOSE	HER LINE ON WHY
1. Weekly leadership update	Company A	Clear	Sounded like me and needed no edits
2. On-time delivery table	Company A	Close	Same findings; its questions for the leads were sharper
3. Against closing the warehouse	Company B	Clear	Found the lease cost I had missed and said so plainly
4. Difficult conversation	Company A	Close	Kept the ask firm and the tone warm
5. Family weekend	Company C	Close	Planned rest stops for my father
6. Strategy memo to slides	Company C	Clear	Five slides I would present as they are

All three passed her must-haves. The slides win surprised her, so she reran it, and company C won again. The family weekend was close, so a tie-breaker settled it: she plans trips by voice in company A's app.

Maya's lineup: her default is the top model from company A. Decisions she wants argued against go to company B. Documents into slides go to company C.

Months later, company B releases a new model. It wins the delivery table clearly and loses the rest. Her verdict: "Split. The new model takes my number work. Everything else stays where it is."

Four habits that keep it fair

1. **Same request, same settings, fresh chat** for every option, so the option is the only thing that changes.
2. **Hide the names before you pick.** Look afterwards.
3. **Notice the cost of getting to an answer you accept.** A cheaper model that needs three extra rounds of follow-up ends up costing more.
4. **Test on day one, and wait a week before trusting what people online say.** Launch-week posts mostly repeat the announcement and show off demos. Reports from real work arrive a few days later.

The same AI can answer the same request differently twice. If a result surprises you, run that request again before you act on it, as Maya did.

IF YOU FINISH EARLY: SCORE ONE TASK

One task teaches a new ability: turning your taste into a written standard, and checking whether an AI can judge by it. Pick the task where your win was closest.

1. **Write a scoring guide.** A **scoring guide** is a few lines on what a great answer includes, what earns top marks, and what is an automatic fail. It turns a wish into lines anyone can check. "Sounds like me" becomes: no sentence over 25 words, opens with a concrete example, no hedging phrases. Use Prompt 5 in Part 3 with your one task. When it offers more tasks, say one is enough for now. **Done:** a scoring guide that says what earns each score from 1 to 5.
2. **Score the hidden answers yourself,** from 1 to 5. **Done:** your scores, written down before you see anyone else's.
3. **Bring in an AI judge.** An **AI judge** is a second AI, from a company that made none of your options. It scores the same hidden answers against your guide. Use Prompt 6 in Part 3, in a fresh chat. **Done:** two sets of scores, and one line on where you and the judge differed.

A disagreement usually has one of three causes. Your guide is missing something you care about. The judge misread. Or length and polish swayed you.

YOU CAN STOP HERE

Parts 1 and 2 are the whole method. If you want more rigor, or you want a program to run it for you, the two parts below go deeper. Each is optional and complete on its own, and each starts from the task list you made in Part 1.

PART 3 (OPTIONAL). THE SCORED VERSION

For the extra diligent. You keep the same tasks and add three things.

1. **A scoring guide for each task.** A few written lines that say what a great answer must include, what earns top marks, and what counts as an automatic fail. A good guide is specific enough that two people, or a person and an AI, would give the same answer the same score. For example, "sounds like me" is a wish. A scoring guide turns it into lines anyone can check: "no sentence over 25 words, opens with a concrete example, no hedging phrases such as 'it may be worth considering'."
2. **Scores from 1 to 5,** which you give yourself, with the names hidden.
3. **An AI judge.** A second AI, from a company that made none of the options you are testing, scores the same hidden answers against your scoring guide. You then compare its scores with yours. An AI judge left to itself gives top marks to almost everything, so Prompt 6 asks it to act as a tough editor. Your own scores decide.

Picking by taste is fast and builds your judgment. A scoring guide makes results comparable from one release to the next, and once you trust the judge, it can do the scoring for you.

Three kinds of saved task

- **Everyday tasks** are the regular work you chose in Part 1. Keep them word for word between runs, so this month's results compare with last month's. If you do change one, save it as a new, dated version.
- **Wish-list tasks** are things you want AI to do and it has not managed yet. Refresh them at each new release. When testing something new, add one or two built around what the release says it can now do.
- **Matter-of-taste tasks** are the ones where the best answer is simply the one you prefer, such as advice on a tricky conversation. Judge them on four points: pushes back when it should, right length, sounds as sure as it should be, and usable as it is. These results give you a read on personality. Take them with a pinch of salt.

Also mark each task as a **chat task** or an **agent task**. A chat task works in any chat window with the material pasted in. In an agent task, the AI has to open files, use tools, or take several steps on its own. When you compare agent tools, run the agent tasks first, because that is where the tools differ most.

You and the judge

Score first, then read the judge's scores, then reveal the names. When you and the judge disagree, one of three things happened:

- **Your scoring guide is missing something you care about.** Add it, and date the change.
- **The judge misread.** Try a different judge, or score that task yourself from now on.
- **Length or polish swayed you.** Reread the scoring guide before you score next time.

Check the judge now and then. When you first set it up, and again every few releases, score a few answers yourself and compare. If you mostly agree on a kind of task, you can let the judge score that kind alone. The benchmark checks your judge, too.

The decision rule

Write it down before you see any answers, so a polished surprise cannot talk you into anything. A good default:

- **Switch** when the new option matches or beats what you use today on every everyday task, and wins at least one wish-list task.
- **Split** when it wins some tasks and loses others. List the tasks it wins.
- **Stay** in every other case.

When finding your lineup, the option with the most wins becomes your default, and each other option takes the kinds of task it won. Matter-of-taste tasks shape your notes on personality and stay out of the verdict. Cost, speed, and how you like each tool settle the close calls, exactly as in the taste test.

The scored version, step by step

1. **Testing something new? Research the release** with Prompt 4, in a tool with web search. Skip this step when finding your lineup. **Done:** a short brief on what the company claims, what people report, and which tasks to watch.
2. **Build your scored benchmark** with Prompt 5. Paste your task list from Prompt 1, and the research brief if you have one. **Done:** one document with every task, its material, its scoring guide, an empty scorecard, and your decision rule.
3. **Run every task in every option, and hide the names**, exactly as in the taste test. **Done:** lettered answers for every task.
4. **Score each answer yourself** from 1 to 5, rereading the scoring guide first. Write your scores down. **Done:** your scores, recorded before you see anyone else's.
5. **Bring in the judge** with Prompt 6, in a fresh chat with an AI from a company you are not testing. **Done:** a filled scorecard with both sets of scores, how often you agreed, and a verdict or a lineup.

Prompt 4: Research a new release

COPY THIS PROMPT

A new AI model or AI tool has just come out, and I want to decide whether it matters for my work. Help me research it. Use web search throughout and cite a source for every claim.

Before you start, ask me three questions, one at a time, and wait for each answer: what was released and by which company (it may be a model, a new option inside a product, or a tool that works on its own with files and apps); how many days ago it came out; and which AI tools I can open today.

If the release is a tool, read its release notes and help pages, which play the part a model announcement plays for a model. Its claims will be features, such as memory, connections to other apps, or the ability to work on its own for longer. Tell me which kinds of work those features would change.

Timing: if it came out less than five days ago, label everything people online are saying as **EARLY AND UNRELIABLE**, because the first days are mostly reposts of the announcement and showy demos. Tell me to run this research again in a week.

Then give me four short sections, in plain language:

1. What the company claims. A table with the claim, the evidence offered, whether that evidence comes from a test the whole industry uses or one the company created itself, and the kind of everyday task where I would notice the difference. Add a few lines on changes to cost, to speed, and to how much material it can work with at once. Add one line on anything the company is quiet about, such as tests it stopped reporting.
2. What people report. Search X, LinkedIn, Reddit, Hacker News, Substack, and independent blogs. Give more weight to posts that show the actual request and answer, to people who work in a named field such as law, finance, or marketing, and to detailed reports of failures. Leave out posts that only repeat the company's numbers, and do not let any single popular account decide. A table with what people say got better, what got worse, which reports show evidence, and which field they come from.
3. Tasks to watch. Eight to twelve kinds of task where this release is most likely to be noticeably better or worse than what came before, each with one line on why, and whether the evidence comes from the company, from users, or both.
4. Personality. One short paragraph on what people say about its style: how long its answers are, whether it pushes back, how often it hedges, and how it handles unclear requests. Label it "impressions only".

End with three questions I should ask myself about my own work before I decide what to test.

Prompt 5: Build my scored benchmark

COPY THIS PROMPT

Help me build a scored personal AI benchmark. A personal AI benchmark is a saved set of my own real requests that I give to several AI options side by side, with the names hidden, to see which one does my work best. An option can be an AI model, a setting inside one product (such as fast or thinking), or an AI tool that works through several steps on its own. The scored version adds a written scoring guide to every task, scores from 1 to 5, and an AI judge whose scores I compare with my own.

Interview me first, one question at a time, and wait for each answer:

- Ask me to paste my saved task list. If I do not have one, tell me to create it first with a task-finding interview, and stop there.
- Ask which situation I am in. Finding my lineup means comparing a few options to choose my default and which kinds of task go elsewhere. Testing something new means comparing a new release with what I use today. If I am testing something new, ask me to paste any research I have on the release.
- Ask me to name the options, and which one I use today, if any.
- Ask whether my tools show me the cost or usage of each conversation.

Then build one document I will still understand a year from now. Explain every part inside the document in plain words.

Part 1, everyday tasks. Use my regular tasks, five to eight in total. If I have fewer than five, suggest more from what you have learned and ask me to approve them. For each task, write down: the exact request, word for word; any material it needs, made up and realistic so nothing private is included; what a great answer must include; the common ways an answer fails; a scoring guide from 1 to 5 that says what earns a 5 and what is an automatic fail, specific enough that two people would give the same answer the same score; and a tag, either chat task (works in any chat window with the material pasted in) or agent task (the AI must open files, use

tools, or take several steps on its own). Tell me these tasks stay word for word between runs, and that any change becomes a new dated version.

Part 2, wish-list tasks. Two to four tasks I have wanted AI to do and it has not done well. Take them from my list. If I am testing something new, build one or two around what the release claims it can now do, shaped so that the claimed ability has to show up for the task to succeed. Write the same fields as for the everyday tasks, plus one line on where each task came from. Tell me to refresh these at each release.

Part 3, matter-of-taste tasks. Choose up to three tasks where the best answer is simply the one I prefer, such as writing in my voice or advice on a tricky conversation. Judge them on four points: pushes back when it should; right length; sounds as sure as it should be; usable as it is. If I am comparing agent tools, add two more points: asks before taking actions that matter, and reports clearly what it did. Label this part "a read on personality; take with a pinch of salt".

The scorecard. An empty table with one row per task. For each option, two columns: my score and the judge's score. Then columns for the winner, whether the judge and I picked the same winner, the cost per task where I can see it, speed, and notes.

The decision rule, written now, before any answers exist. If I am testing something new, one verdict per new option: Switch (it becomes my default) when it matches or beats what I use today on every everyday task and wins at least one wish-list task; Split (it joins my lineup for specific kinds of task, which I list) when it wins some tasks and loses others; Stay (keep my lineup and test again next release) in every other case. If I am finding my lineup, the option with the most wins becomes my default, and each other option takes the kinds of task it won. Matter-of-taste tasks shape my notes on personality and stay out of the verdict. Cost, speed, and how much I like each tool settle close calls.

A short run checklist: a fresh chat for every option; the same request and material; the same settings for web search, memory, and thinking; the same level of model when comparing companies (top against top, fast against fast) unless comparing levels is the point; names hidden before any scoring; I score first; the judge is an AI from a company that made none of the options; the names are revealed only after scoring. For agent tools: the same model inside each tool where the tool allows it, the finished result is what gets judged, and agent tasks run first. Say which tasks to run first if I am short on time.

Refer to AI options by company and level, such as "the top model from company X", so the document still makes sense after products are renamed.

Prompt 6: The AI judge

Score the answers yourself first. Then run this in a fresh chat with an AI from a company that made none of your options.

COPY THIS PROMPT

I ran my personal AI benchmark: the same saved requests given to several AI options, with the names hidden and the answers labeled with letters. I want you to act as a tough, independent editor who has to pick what actually gets used, and who has seen a great deal of competent, forgettable work. I will paste my benchmark document first, so you have the tasks, the scoring guides, and my decision rule. Then I will paste the lettered answers one task at a time. Do not guess which AI wrote which answer, and do not ask.

For each task, read every letter first and rank them from best to worst. Then score every letter from 1 to 5 against the scoring guide, and keep the scores in the order of your ranking. Competent work earns a 3. A 5 means I would use it exactly as it is and it clearly beats the others. Give at most one 5 per task, and give none if no answer earns it. Write one line for each score that names the answer's main weakness, including for the winner, and name the winner. Judge the finished result, and do not reward length or polish that the request did not ask for. If an answer is a web page, remember that you can read only its code and cannot see the page, so leave the look of it to me. If I also paste a record of the steps an agent tool took, use it only for the points about asking before acting and reporting what it did. For matter-of-taste tasks, rate the four points in the document and add one line on each letter's personality: what kind of work it seems suited to, and what I should watch out for.

After each task, ask me for my own scores, which I gave before reading yours. Tell me where we picked the same winner and where we differ. Where we differ, help me work out which of three things happened: my scoring guide is missing something I care about; you misread the answer or the guide; or I was swayed by length or polish. Push back if I try to move the goalposts or explain away a failure.

When all tasks are done, fill in the scorecard, including any cost and speed figures I give you, and tell me how often we agreed. For each task, say whether the win was clear or close. Ask me whether anything beyond the

answers matters to my choice: whether I am allowed to use each option at work, its privacy terms, my plan, cost, speed, a feature I love, or the effort of switching. Then apply the decision rule from my document to my own scores. My scores decide, and yours are a second opinion. Let clear wins on tasks that matter move me, and let what I told you settle the close calls. If I am testing something new, give one verdict per new option: Switch, Split, or Stay, as defined in my document. If I am finding my lineup, give my AI lineup: my default, and which kinds of task go elsewhere. Add three notes on what I seem to value, written as qualities that will still make sense after products are renamed.

I will reveal which letter was which only after your decision. Then help me update my lineup and note what to retest next time.

PART 4 (OPTIONAL). THE AUTOMATED VERSION

The Benchmark Runner is a small program that runs your personal AI benchmark for you. You give it your saved tasks and the names of two to four AI models. It sends every task to every model, hides which model wrote which answer, and opens a page in your browser where you read the answers and score them blind. Then it reveals the names and tells you what your own scores say: switch, split, or stay, with what each model cost and how long it took. An AI judge can score alongside you if you want a second opinion.

It is for people who are comfortable pasting an API key into a file and double-clicking a launcher. You do not need to write code. If that is not you, the taste test in the lab handout gets you the same answer by hand.

Get your key

A **key** (often called an API key) is a private code that lets a program use an AI model directly, billed to your own account. Treat it like a password.

The easy route: one key for many models. OpenRouter is one account that reaches models from most companies, and it reports what each answer cost and how long it took. Three steps:

1. Create an account at openrouter.ai.
2. Add a small amount of credit.
3. Create a key, and paste it into the `key.env` file, as the START-HERE file in the download shows.

Other routes. Another model hub that offers the same kind of single key also works. So does a key directly from the company whose models you want to compare, such as OpenAI, Anthropic, Google, or xAI. A direct key works the same way, with one address changed in the settings file. Cost then shows as "n/a", and only time and length are reported. A direct key reaches one company's models, so comparing across companies in one run takes a hub.

What it costs. You pay the model companies for what you use, through your own account. The reveal page shows the exact amount for every run.

Keep it safe. Never paste a key into a chat, an email, or a shared document.

Run it

The download includes a START-HERE file with four steps: get a key, choose your options and tasks, run the tryout, and review blind, then reveal. Each step says what "done" looks like. The task files `Prompt 1` saved in your `my-ai-benchmark` folder work as they are.

Choose two to four options. The program can run more than you can judge well. Reading seven answers to one request and holding them all in your head is hard work, and your judgment gets worse as the pile grows. At the next release, you add the new option to the settings file and run it again.

Score on the review page, then reveal

The review page opens in your browser. For each task it shows the request, your scoring guide, and every answer under a letter, in a different order for each task. Read every answer to a task before you score any. Then give each one a score

from 1 to 5, with an optional note, and be stingy with top marks. Your scores save on every click, so you can stop and come back. An answer that is a web page is shown as a real page, with buttons to see it at phone width or open it full size.

When you are done, press one of the two buttons at the top:

- **Reveal with the AI judge.** The judge scores the same answers blind first, then you see the names.
- **Reveal, my scores only.** You see the names with no judge at all.

If you named the option you use today, the reveal shows a verdict card for each other option: Switch, Split, or Stay, decided by your own scores. Below the cards, it shows what each option cost and how long it took, for each option and for each task. Cost and time appear only after the reveal, because they are tie-breakers: they settle close calls once the answers are judged. "You and the judge" shows where you picked the same winner and where you differed. "The raw scores" lists every score, yours and the judge's, with the reasons.

The AI judge

The AI judge is a second reader, from a company that made none of your options. The first version gave top marks to almost everything, so it was rewritten as a tough editor. It ranks the answers first, gives at most one top mark per task, and treats competent work as a 3. It cannot see a web page. It reads only the page's code, so the look of a page is yours to judge. Where you and the judge differ, your own scores decide.

What stays by hand

Tasks that need web search or a build tool stay by hand, because the runner sends each request to the model as text. A task that asks for a web page can be answered as one file, and the review page shows it as a real page. Agent tools such as Claude Code or Cowork are compared by hand too. One tool cannot drive another, so you run each task inside each tool and save the answers into a folder for the blind-shuffle script.

OpenRouter's chat page can help with the by-hand route. It sends one request to several models at once and shows the answers side by side, with the names visible. It is a quick way to collect answers before you hide the names with one of the four ways in Step 3.

Let an agent tool set it up

An agent tool such as Claude Code, Codex, or Cursor can set the runner up with you, with Prompt 7.

Prompt 7: Set up the runner

Open your agent tool in a folder that holds your benchmark document, any made-up material your tasks use, and the benchmark-runner folder from the download. Paste this.

COPY THIS PROMPT

Help me set up the Benchmark Runner for my personal AI benchmark. In this folder you will find my benchmark document (my saved tasks, each with its exact request, its material, and its scoring guide, plus my decision rule), the made-up material the tasks use, and a benchmark-runner folder. The Benchmark Runner is a small program that sends each task to every AI option and saves the answers under random letters. It opens a review page in my browser where I score the answers with the names hidden. Then it reveals the names, with a verdict from my own scores and what each option cost and how long it took. An AI judge can score alongside me as a second opinion.

Before changing anything, read my benchmark document and the START-HERE and README files in the benchmark-runner folder. Where those files are more specific than this message, follow them.

Then interview me, one question at a time, and wait for each answer:

- Which options do I want to compare, and how is each one named on the service I am using? Keep it to two to four, because I will read every answer myself. If I name more, ask me which ones to drop.
- Is one of them what I use today? If so, I will get Switch, Split, or Stay for each other option. If not, I will get a lineup: my default, and which tasks go elsewhere.
- Do I want an AI judge as a second opinion? If so, it must come from a company that made none of the options. Refuse any judge that matches an option.

- Is my key already set up the way START-HERE describes? Remind me never to paste the key into this chat.
- Does any task need material you cannot find in this folder?

Then do the work. Convert each task into the format the runner expects, one file per task, keeping my request and scoring guide word for word. If a task needs web search or a build tool, tell me it stays by hand and leave it out. If a task asks for a web page, set it up to be answered as one file, so the review page can show it as a real page. Fill in the settings from my answers. Run one everyday task first and show me the lettered answers before running anything else, so I can check that the setup is fair. Once I approve, run everything. Then open the review page for me and explain how to use it: read every answer to a task before scoring any, score each answer from 1 to 5, and be stingy with top marks. My scores save on every click. I press a Reveal button myself when I am done.

After the reveal, save a copy of the reveal page and my decision next to my benchmark document. Then tell me in plain words: the total cost, the slowest task, the tasks where the judge and I disagreed, and any task where the answers differed so much in length that I should check the settings. Tell me how to rerun everything the next time something new comes out.

Done: the reveal page saved next to your benchmark document, and the steps to rerun it.

Platforms that do this for teams. Some products already run a side-by-side comparison with a scoring guide and an AI judge. LangSmith, Braintrust, and Langfuse each let you run your saved requests across several models, score the answers with an AI judge, and review them by hand. promptfoo does the same as a free, open-source tool for people comfortable with a command line. They are built for teams testing prompts inside software they ship. For one person comparing a few options on real tasks, they are more than you need, and none of them hides the names on options you choose. That is why the taste test and the small script exist. If your company already uses one, it will run the scored version for you.

FROM HERE ON: BUILD ONCE, REUSE FOR MONTHS

You have now done the hard part. The thinking about how to evaluate an AI is finished, and it is saved in one document. From here on, a new release costs you no thinking at all.

When something new comes out, you do Part 2 again with the new option and what you use today: the same saved requests, unchanged, names hidden, pick, decide. That is the whole routine. You do it as often as releases interest you, and you never start from a blank page again.

Refresh the benchmark itself only rarely. Three moments call for it:

- **Your work has changed.** A new role, a new kind of task filling your week. Run Prompt 1 again and swap in what is missing.
- **A step change in what AI can do.** When a new kind of ability arrives, add one wish-list task that tests it.
- **A task stopped telling you anything.** If every option handles it perfectly, replace it with a harder one from your real work.

Between those moments, leave the benchmark alone. That is what makes results comparable across releases. Built well, a benchmark serves you for a good few months, and often longer.

For a new agent tool the routine is the same, with the tasks that use files and tools going first.

Done: one saved document, a lineup you trust, and a routine you can repeat at every release.