

OS21: Manual vs AI-Assisted Workflows: Practical Strategies for Complex R Package Development

Phuse Connect US 2026
26th March 2026

Vedha Viyash



Appsilon

Pharmaverse – Home to many R packages



 53 R packages

 150k+ lines of R code

 50k+ commits

 1371 contributors

Pharmaverse – Home to many R packages



 53 R packages

 150k+ lines of R code

 50k+ commits

 1371 contributors

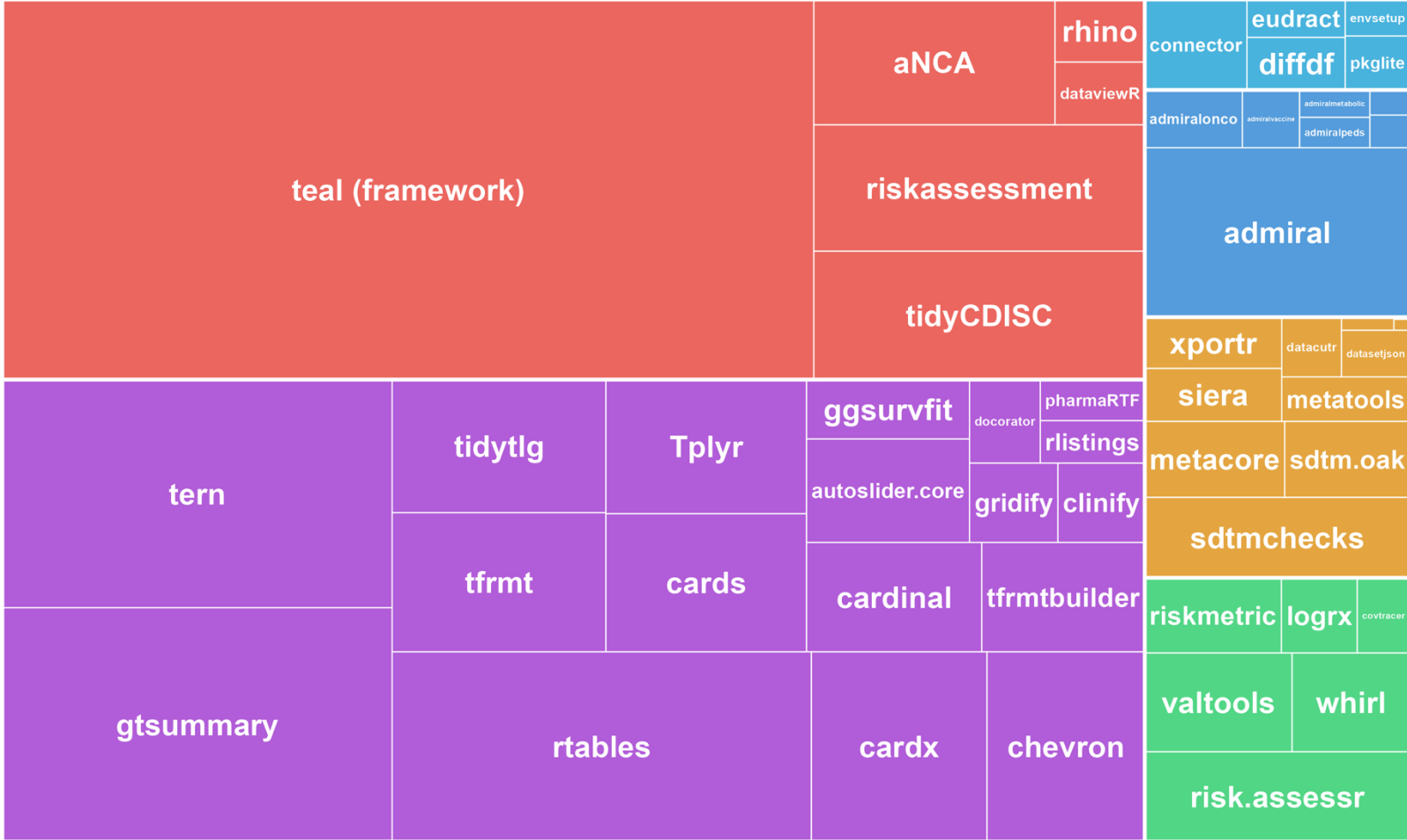
Diagram illustrating the Teal package structure and dependencies:

- Dependencies (Top Group):**
 - {teal.logger}
 - {teal.code}
 - {teal.data}
 - {teal.slice}
 - {teal.reporter}
 - {teal.widgets}
- Teal Package (Center):** {teal}
- Dependencies (Bottom Group):**
 - {teal.transform}
 - {teal.picks}
 - {teal.modules.general}
 - {teal.modules.clinical}
 - {teal.modules.hermes}

Downstream Dependencies

The Pharmaverse Ecosystem by Code Size

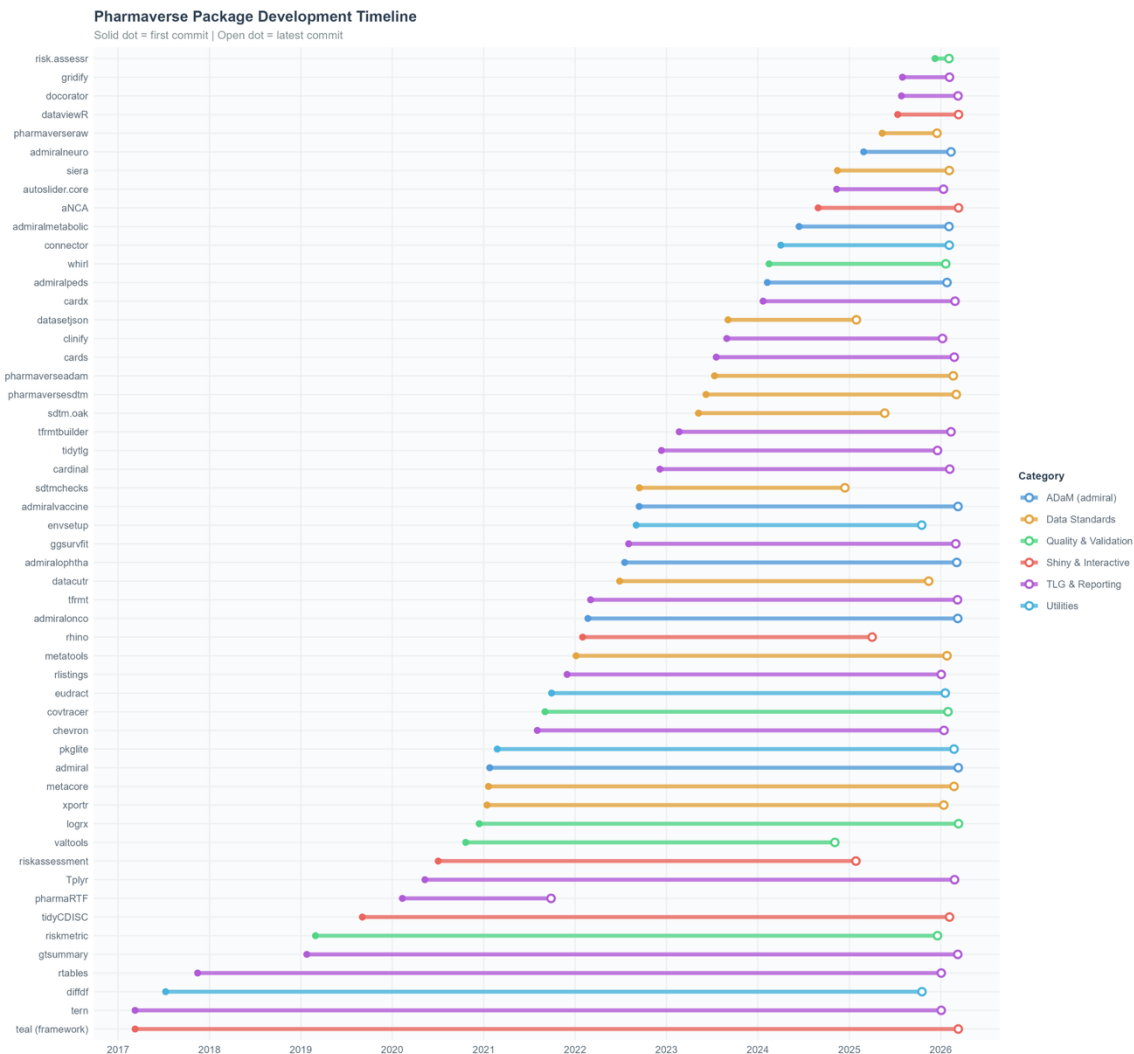
Area proportional to lines of R code | 53 open-source packages for clinical trial programming



Category

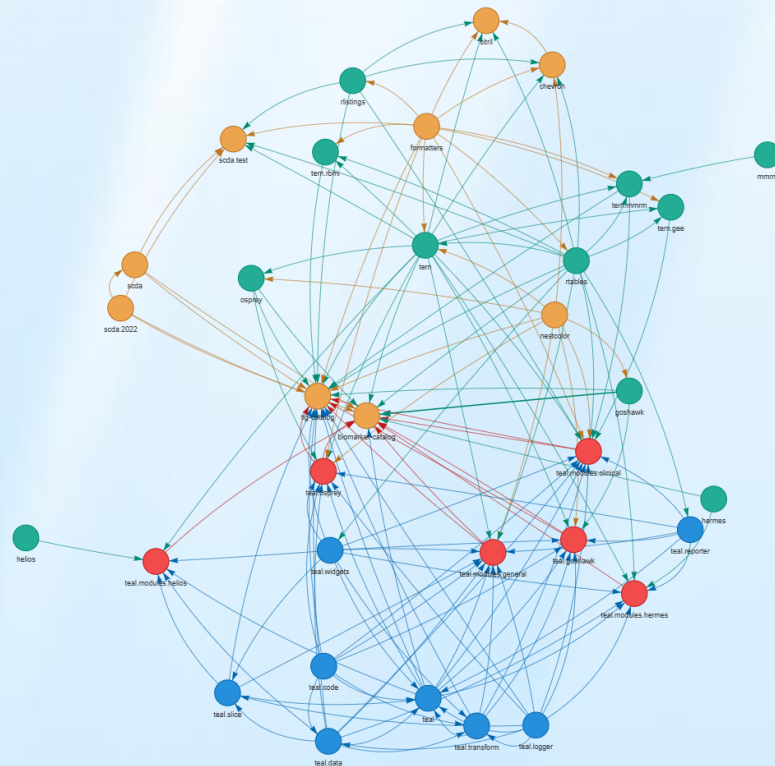
- ADaM (admiral)
- Data Standards
- Quality & Validation
- Shiny & Interactive
- TLG & Reporting
- Utilities

{teal} is also the oldest
pharmaverse package
with the first commit
dating back to 2017!



Technical challenges of developing {teal}

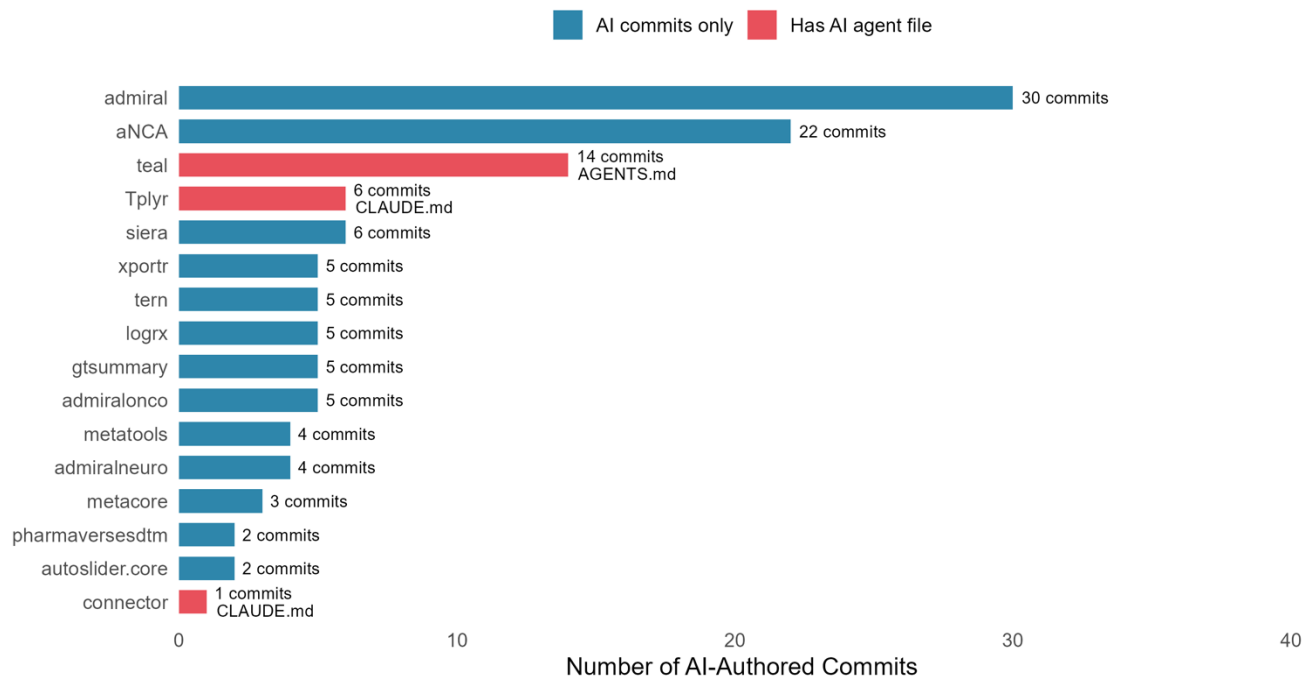
- Working in brownfield codebase
- Onboarding into the package ecosystem
- Debugging the issue
- Task complexity - Serving diverse users



How is the AI adoption in {teal} development?

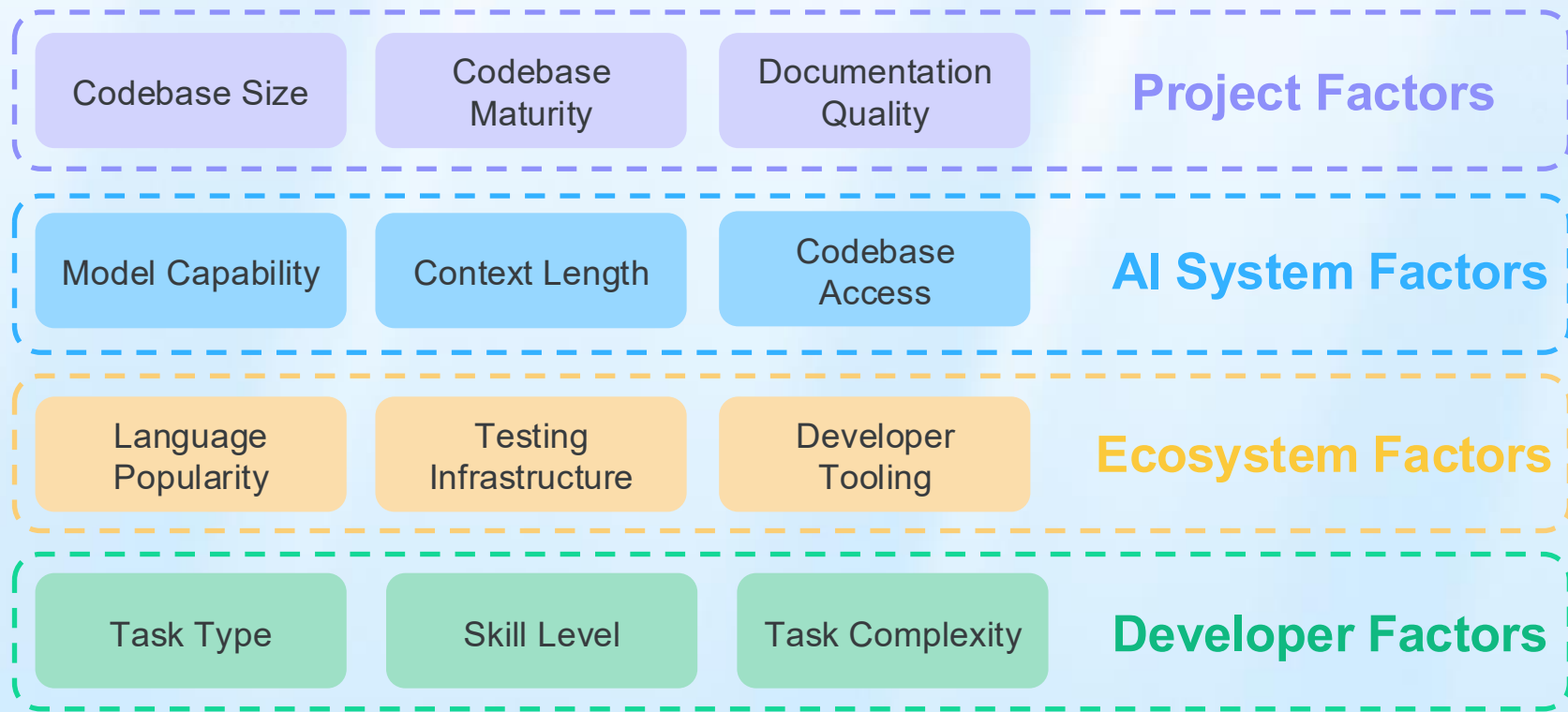
AI-Assisted Development in the Pharmaverse

16 of 53 packages (30%) have AI-authored commits



Early adoption
For simple tasks

Can AI increase developer productivity? What is needed?



Can AI increase developer productivity? What is needed?

Add unit tests

 Open Vedha Viyash requested to merge `add-tests@main` into `main` 3 hours ago

Overview 18 Commits 5 Pipelines 2 **Changes 51**

13 open threads

Add a to-do item

 Compare `main` and `latest version`

 51 files +10018 -175



Increase the code coverage
from 0 to over 90%



The prompt is **NOT** “Add unit
tests for my R package”



It would have taken 40%
more time without agentic AI

teal.modules.mdr coverage - 92.22%

Files	Source						
File		Lines	Relevant	Covered	Missed	Hits / Line	Coverage
R/bargraphplotly.R		150	92	92	0	14	100.00%
R/boxplotly.R		190	100	99	1	12	99.00%
R/butterflyplotly.R		560	345	272	73	5	78.84%
R/lineplotly.R		276	173	154	19	4	89.02%
R/module_colour_picker.R		131	72	72	0	6	100.00%
R/patientprofileplotly.R		676	365	336	29	8	92.00%
R/scatterplotly.R		154	81	81	0	16	100.00%
R/spaghettiplotly.R		238	127	118	9	8	92.91%
R/spiderplotly.R		212	117	104	13	6	88.89%
R/swimlaneplotly.R		237	134	131	3	10	97.76%
R/tm_mdr_bargraph.R		1027	478	474	4	7	99.16%
R/tm_mdr_boxplot.R		479	168	167	1	9	99.40%
R/tm_mdr_butterfly.R		499	164	148	16	1	90.24%
R/tm_mdr_lineplot.R		673	259	239	20	3	92.28%
R/tm_mdr_patient_profile.R		459	152	152	0	1	100.00%
R/tm_mdr_scatterplot.R		896	385	263	122	7	68.31%
R/tm_mdr_spaghetti.R		1003	437	371	66	1	84.90%
R/tm_mdr_spiderplot.R		833	423	420	3	2	99.29%
R/tm_mdr_swimlane.R		593	217	216	1	2	99.54%
R/tm_mdr_waterfall.R		522	193	193	0	7	100.00%
R/tm_L_reactable.R		475	199	180	19	4	90.45%
R/utlis.R		56	31	29	2	48	93.55%
R/waterfallplotly.R		196	109	102	7	11	93.58%

What is the workflow I use for this task?

Research

Manual task

Task delegated to AI

Plan

Breakdown

Task list

Execute

What can be done today?

Research

Manual task

Task delegated to AI

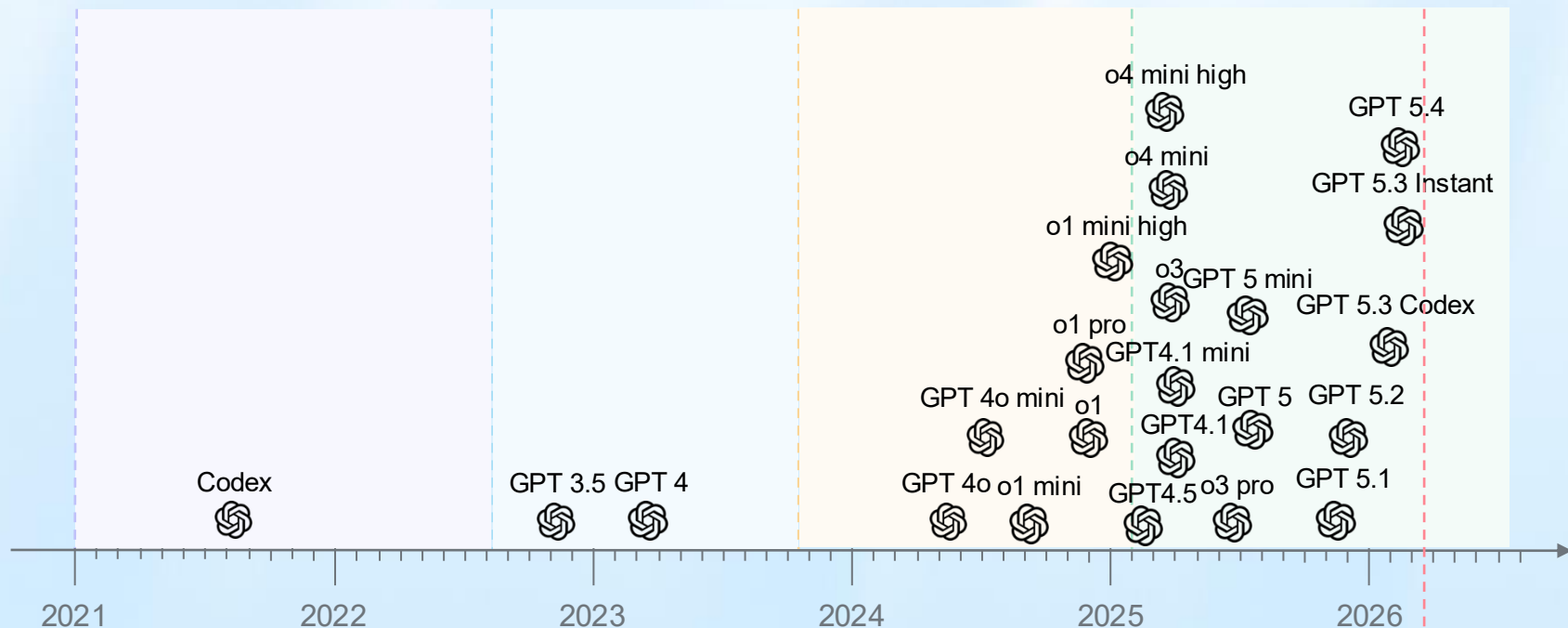
Plan

Breakdown

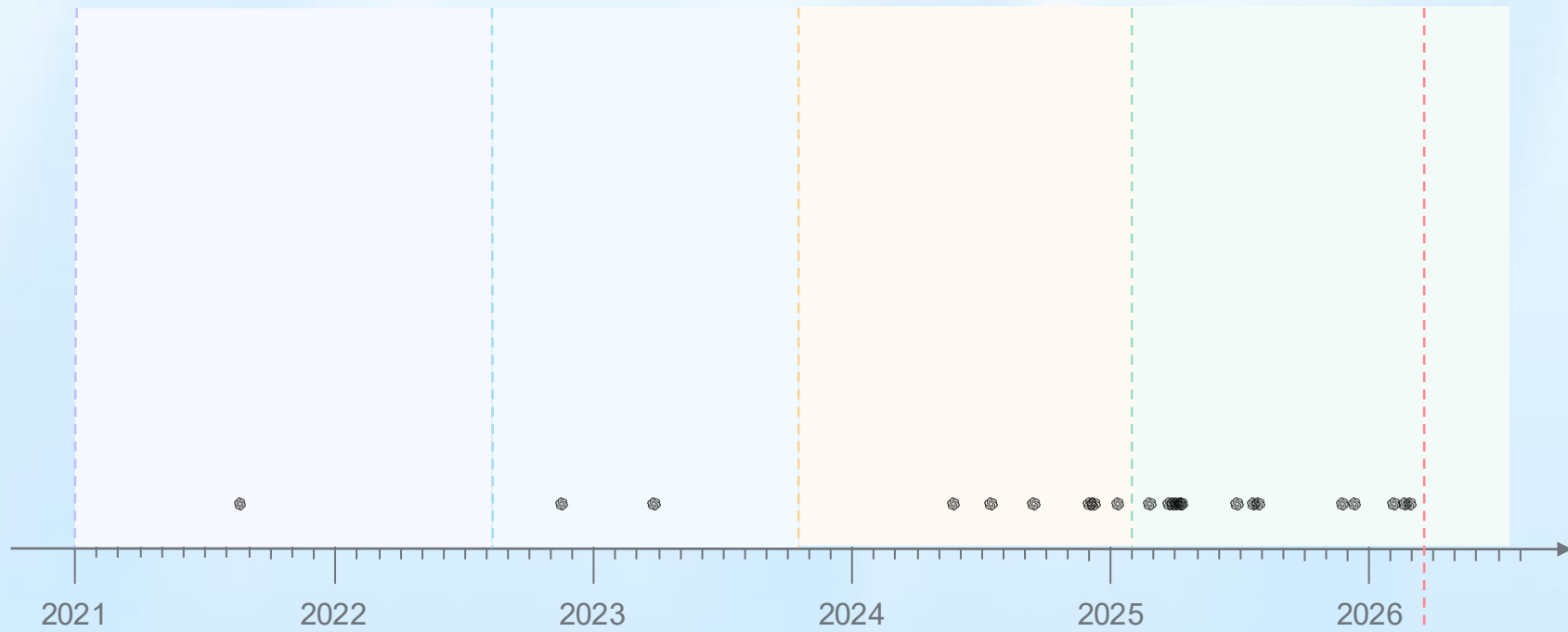
Task list

Execute

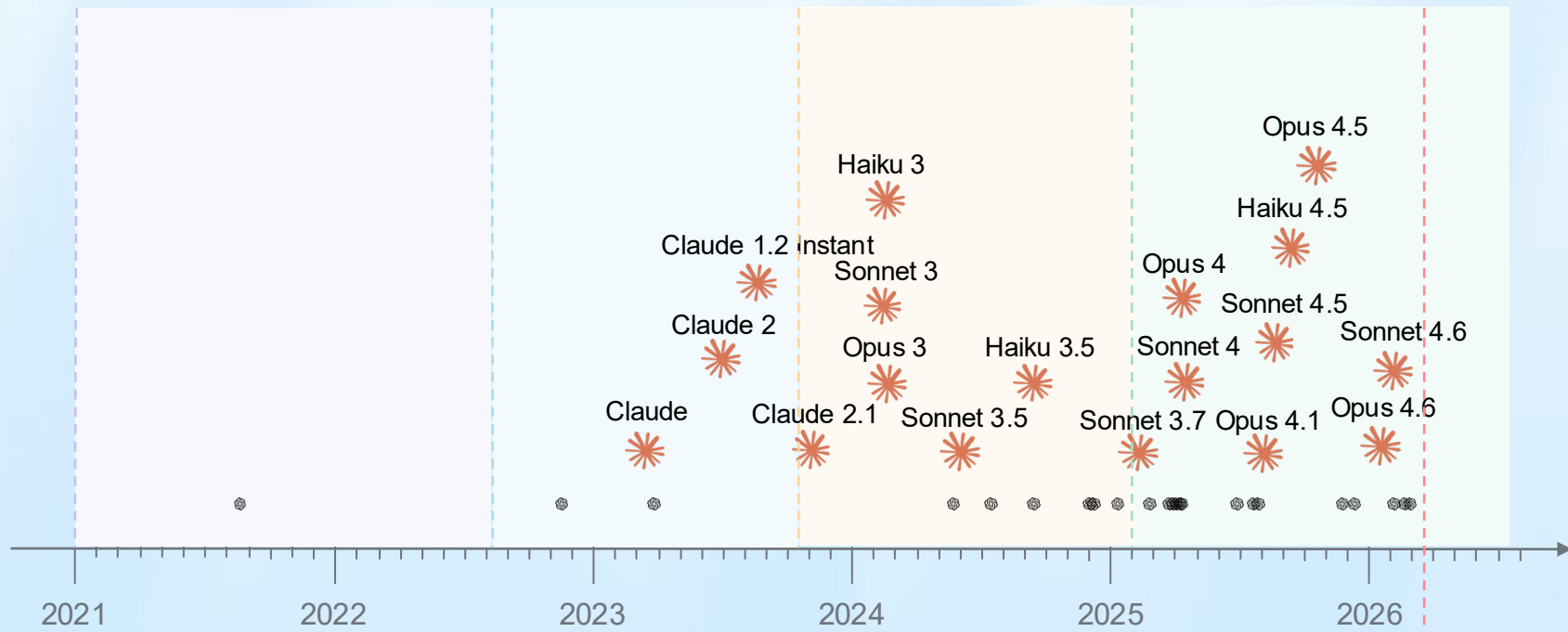
A brief history of AI progress



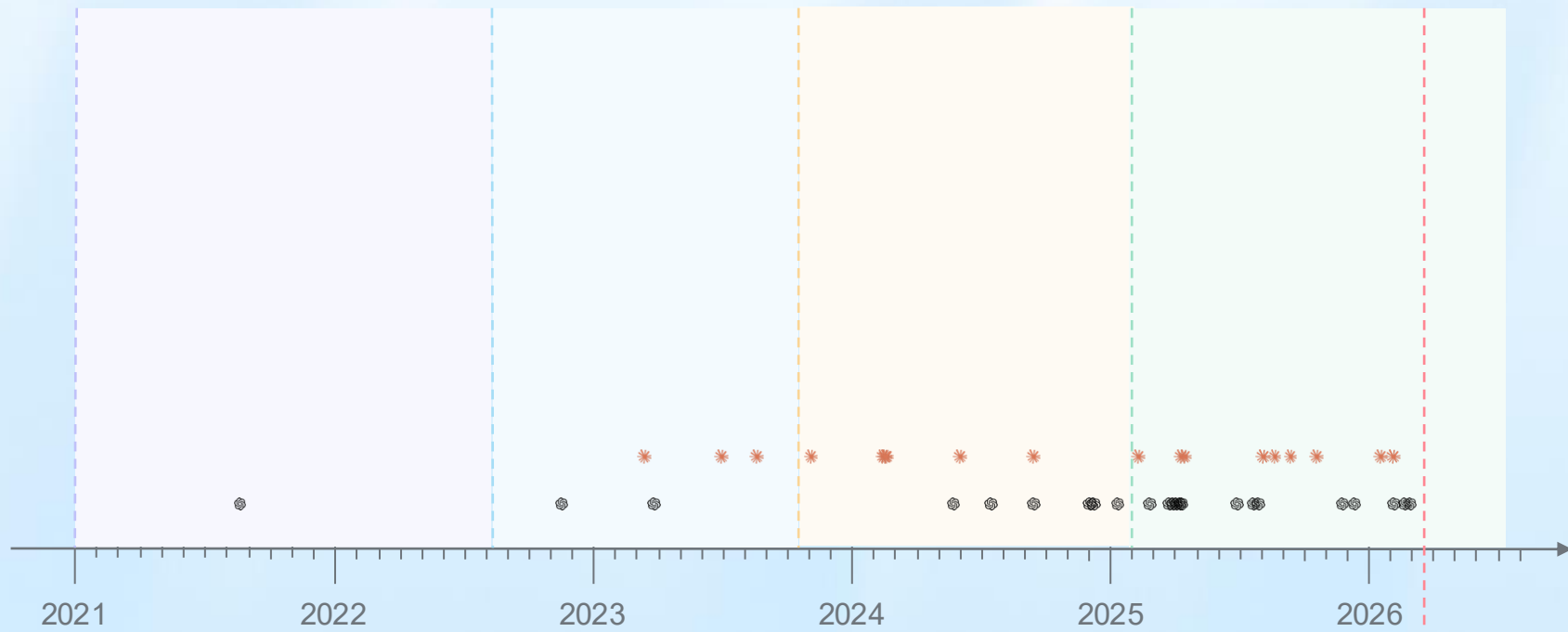
A brief history of AI progress



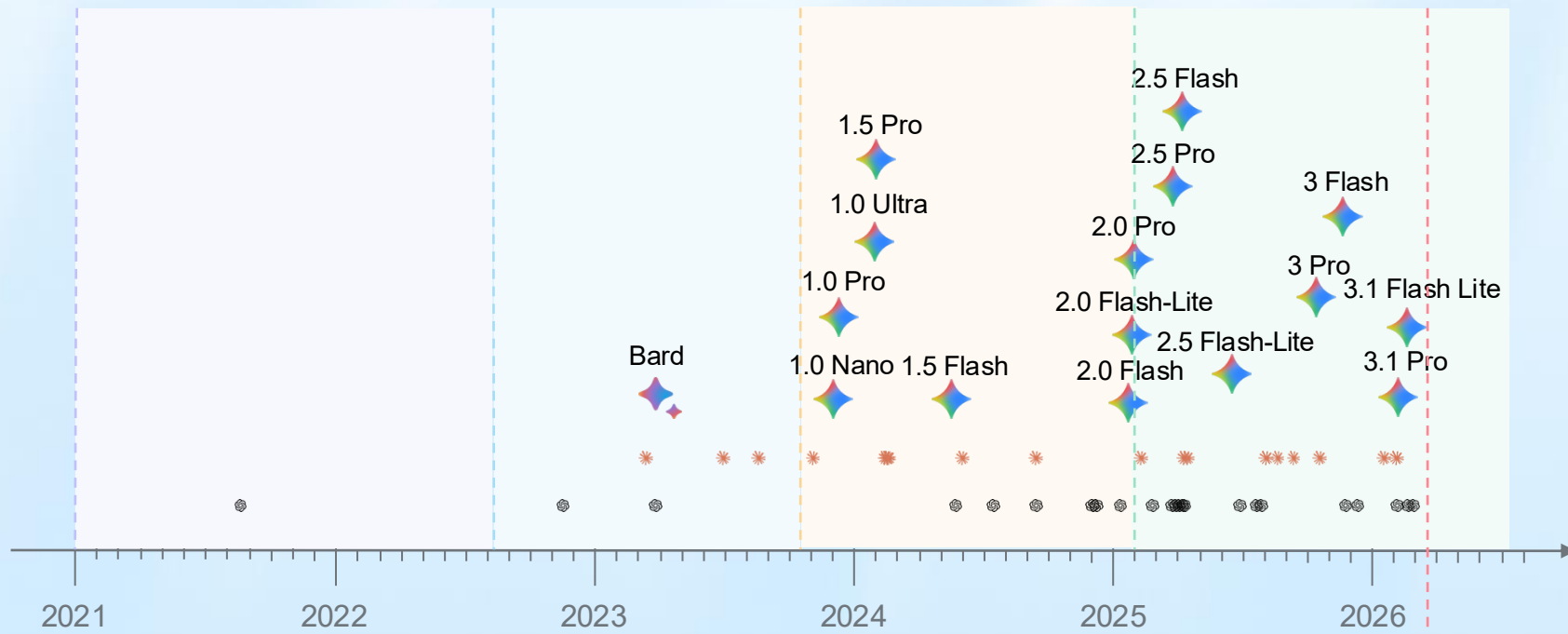
A brief history of AI progress



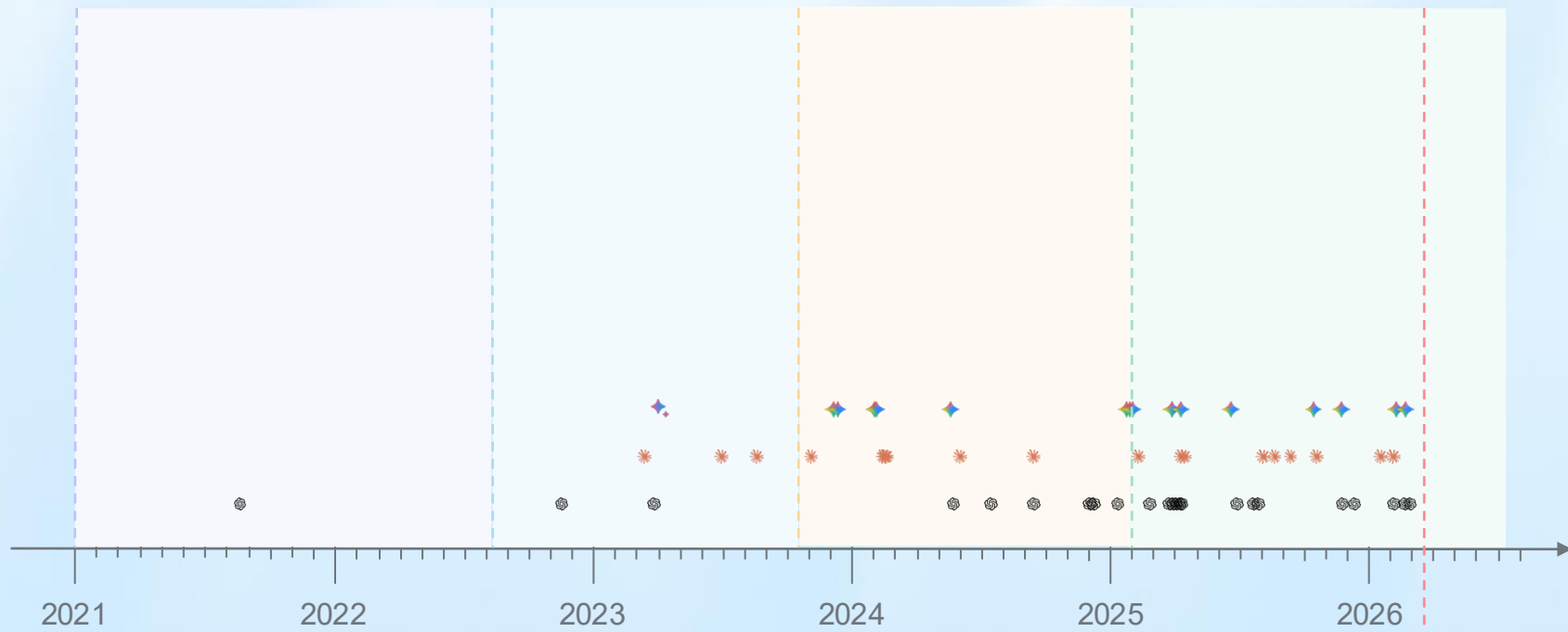
A brief history of AI progress



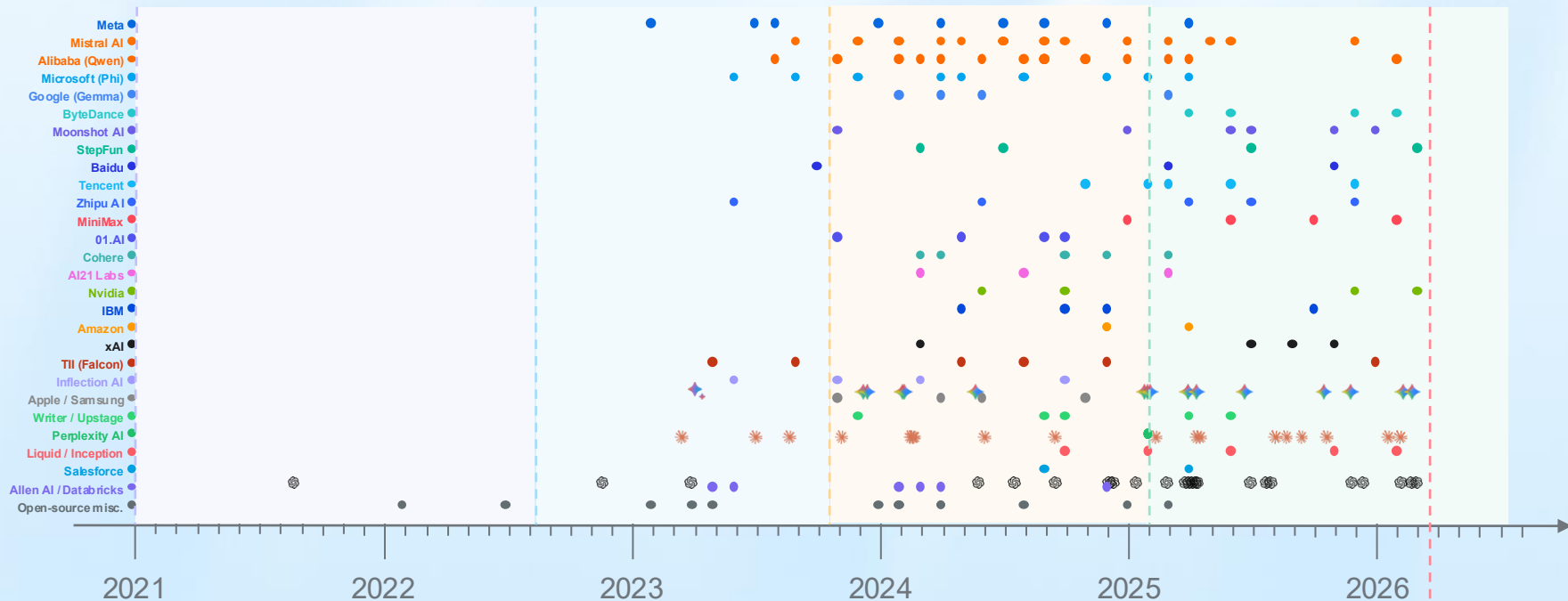
A brief history of AI progress



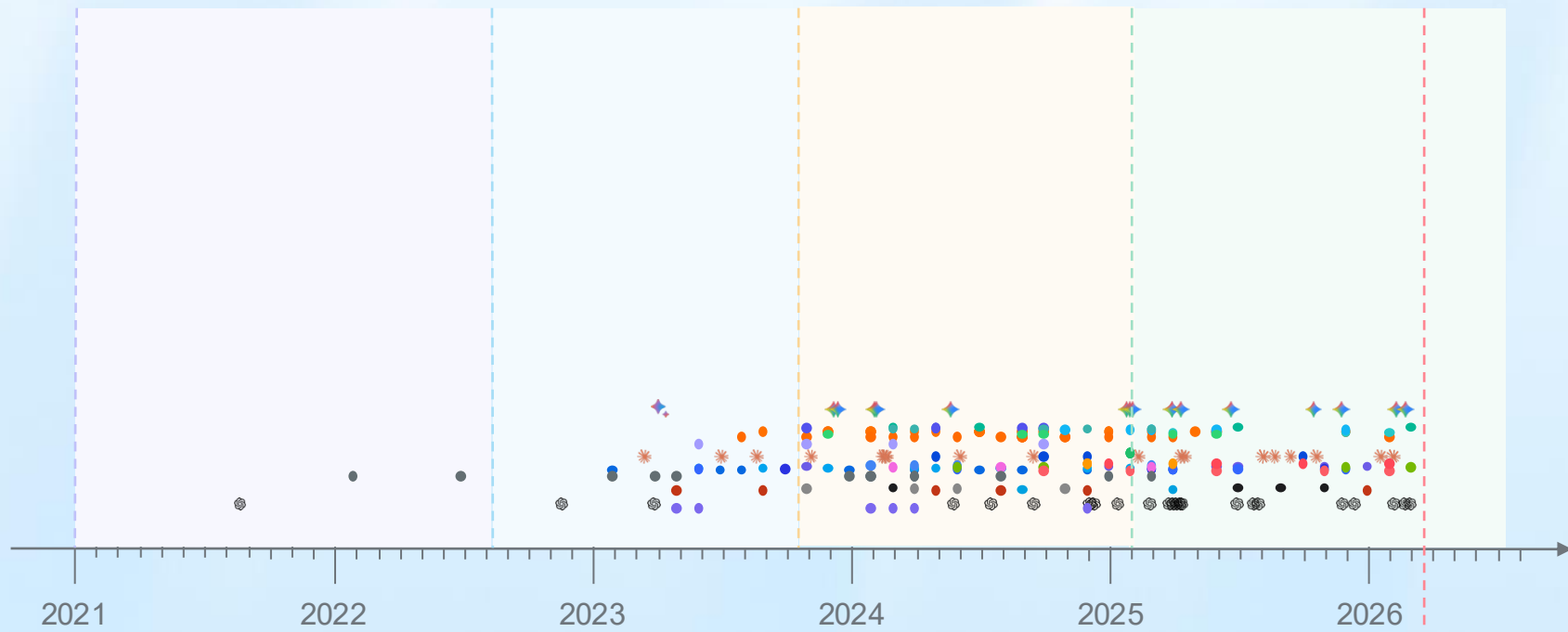
A brief history of AI progress



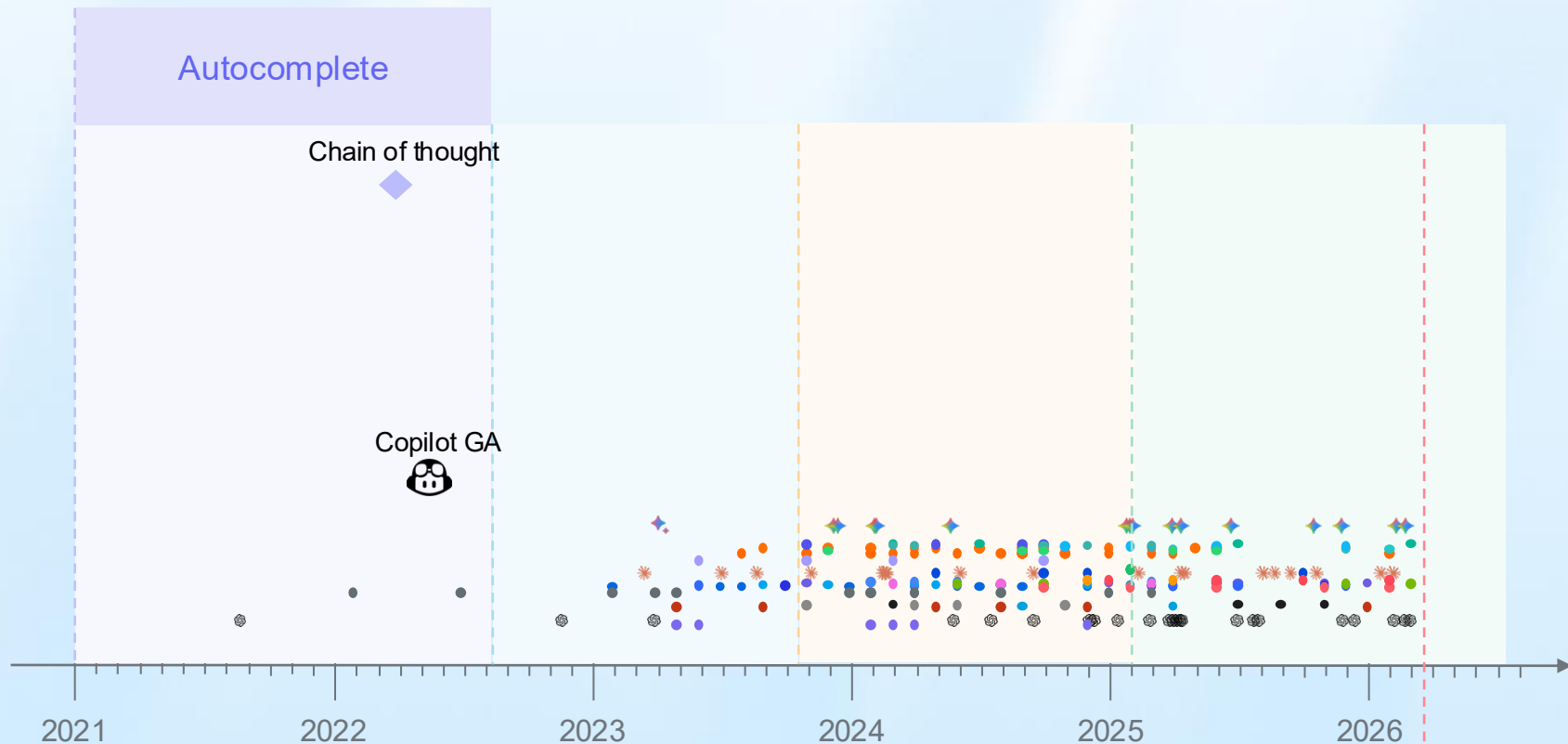
A brief history of AI progress



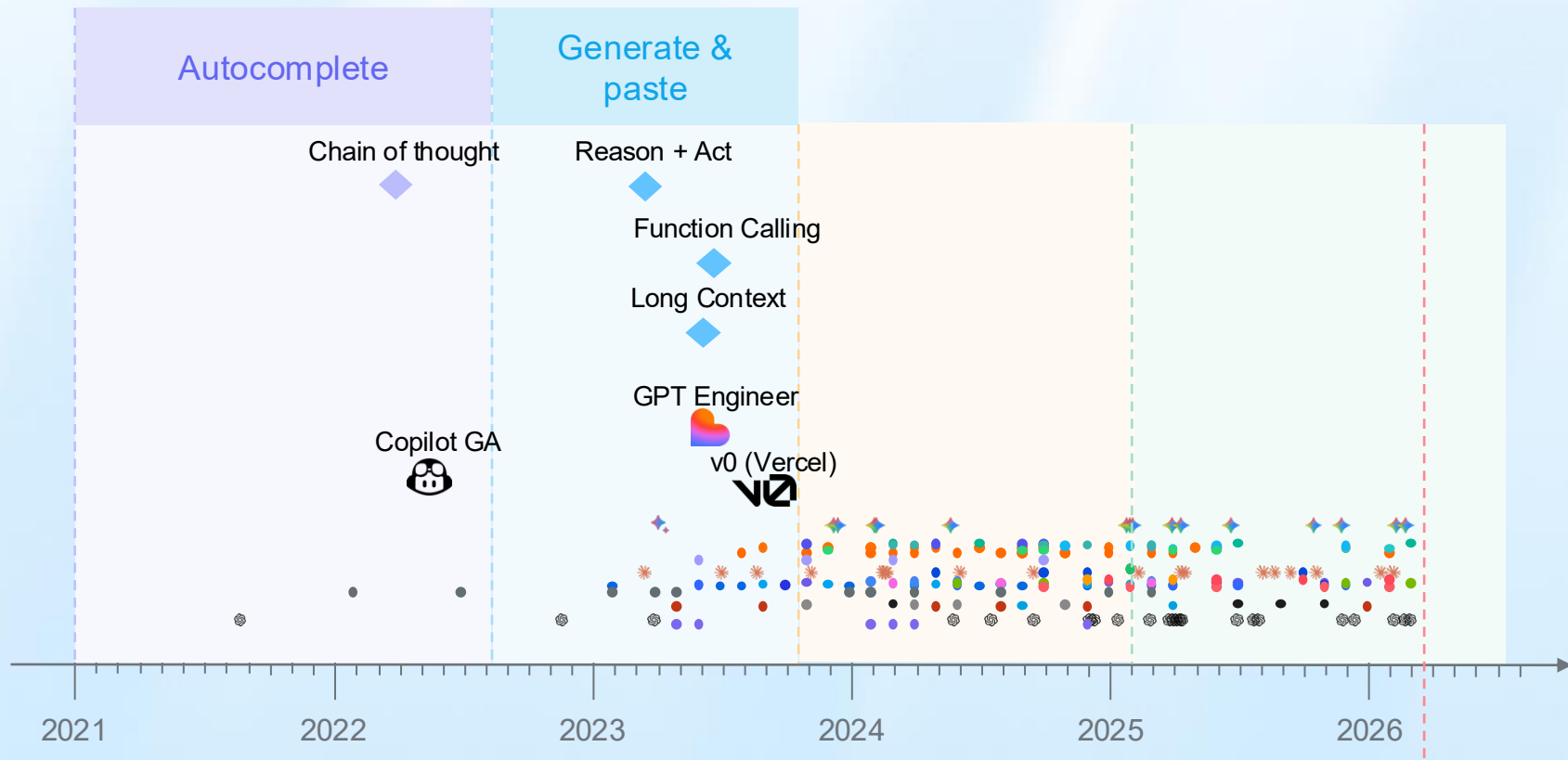
A brief history of AI progress



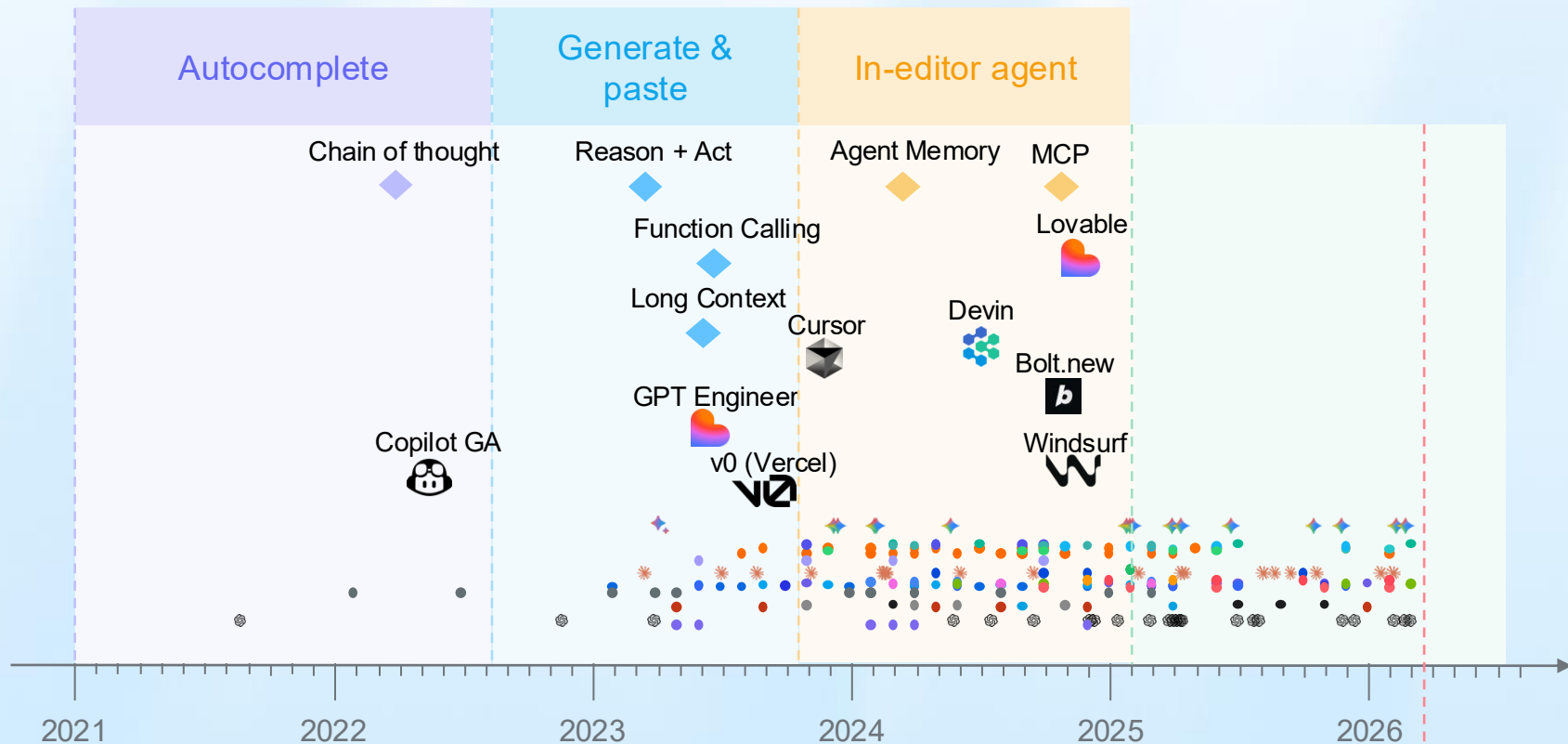
A brief history of AI progress



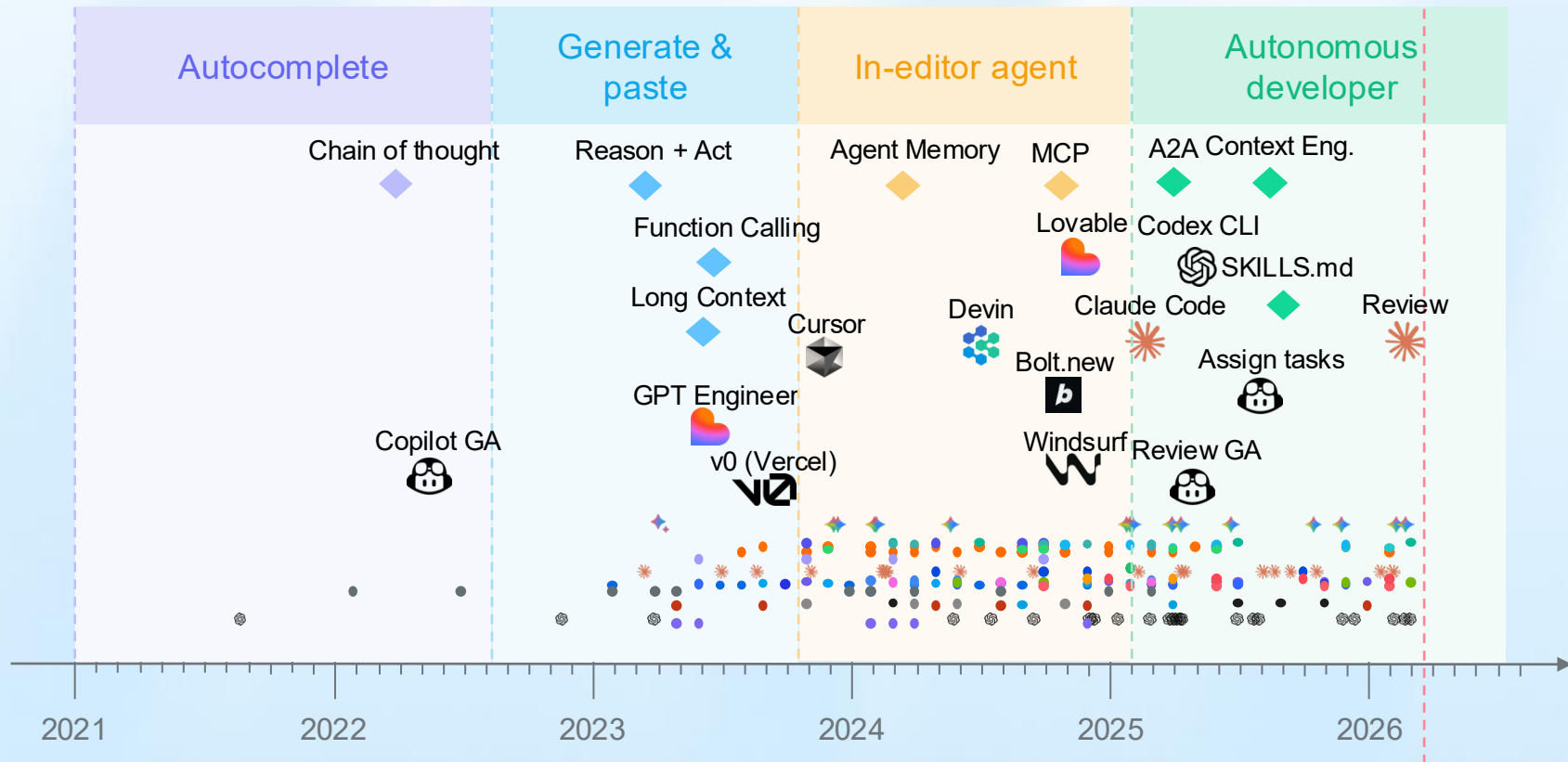
A brief history of AI progress



A brief history of AI progress



A brief history of AI progress



My past workflow – GitHub Copilot

The screenshot shows a VS Code editor with the following components:

- Left Sidebar:**
 - CHAT:** Overview of current project, including Decorators, Transforms, Validation, and Testing driver.
 - Testing (testthat):** The test suite is comprehensive with two tiers: Unit tests (35 files) and Integration/e2e tests (13 files).
 - Vignettes:** 12 vignettes covering topics like Getting started, Filter panel, Data as Shiny, Adding reports, and Bootstrap themes.
- Main Editor:** Displays the `AGENTS.md` file, which is a Teal Ecosystem R Package Development Guide. The guide includes sections for Introduction, Core Packages, Module Packages, Supporting Packages, and Package Structure and Organization.
- Right Sidebar:** Explorer view showing the project structure, including `.devcontainer`, `.github`, `.Rproj.user`, `docs`, `inst`, `man`, `pkgdown`, `R`, `revdep`, `rsconnect`, `shiny_bookmarks`, `sidebar`, `tests`, `vignettes`, `!_pkgdown.yml`, `.gitignore`, and `.gitlab-ci.yml`.
- Terminal:** Shows the command `D:\insightsengineering\teal>`.

AGENTS.md Content:

```
1 # Teal Ecosystem R Package Development Guide
2
3 ## Introduction
4
5 The teal framework uses shiny to create reproducible environments for analysts. The ecosystem comprises several interconnected packages with specific roles:
6
7 ### Core Packages
8
9 - **teal** - The main framework package providing the application structure
10 - **teal.data** - Data management and validation
11 - **teal.slice** - Data filtering capabilities
12 - **teal.code** - Code generation and evaluation
13 - **teal.widgets** - Reusable UI components
14 - **teal.logger** - Standardized logging across the framework
15 - **teal.reporter** - Report generation functionality
16 - **teal.transform** - Data transformation utilities
17
18 ## Module Packages
19
20 - **teal.modules.general** (tmg) - General-purpose analysis modules
21 - **teal.modules.clinical** (tmc) - Clinical trial specific modules
22 - **teal.modules.hermes** - MultiAssayExperiment analysis modules
23 - **teal.goshawk** - Pharmacokinetics analysis modules
24 - **teal.osprey** - Advanced clinical analysis modules
25
26 ## Supporting Packages
27
28 - **tern** - Statistical analysis functions
29 - **rtables** - Table creation and formatting
30 - **formatters** - Output formatting utilities
31
32 **Key Principle**: Balance dependency value with features. Minimize dependencies to packages not already in use within the ecosystem.
33
34 ## Package Structure and Organization
35
36 ### Standard Package Layout
37
38 Follow the standard R package structure with teal-specific conventions:
39
40 ```text
41 package_name/
42 |__ .github/workflows/ # CI/CD workflows (use r.pkg.template)
43 |__ R/ # R source code
```

My current workflow – Claude Code

The screenshot displays the Claude Code interface, which is integrated into a code editor. The interface is divided into several panels:

- Left Panel (Overview of Current Project):** This panel provides a high-level overview of the project structure. It includes sections for Decorators, Transforms, Validation, Testing driver, Testing (testthat), Vignettes, and Key Design Patterns. The Testing section is currently expanded, showing details about the test suite and the TealAppDriver.
- Center Panel (Code Editor):** The main area for editing code. It shows the `AGENTS.md` file, which serves as a development guide for the teal ecosystem. The content includes an introduction to the framework, a list of core packages (e.g., `teal`, `teal.data`, `teal.slice`), module packages (e.g., `teal.modules.general`, `teal.modules.clinical`), supporting packages (e.g., `teal.tables`, `teal.formatters`), and a key principle about balancing dependency values with features.
- Right Panel (Explorer and Terminal):** This panel is split into two sections. The top section, labeled "EXPLORER", shows the file tree of the project, including directories like `.devcontainer`, `.github`, `.Rproj.user`, `docs`, `inst`, `man`, `pkgdown`, `R`, `revdep`, `rconnect`, `shiny_bookmarks`, `sidebar`, `tests`, `vignettes`, `._pkgdown.yml`, `.gitignore`, and `.gitlab-ci.yml`. The bottom section, labeled "TERMINAL", shows the output of the `claude` command, which includes a welcome message and information about the current environment (Opus 4.6, 1M context).

The overall workflow depicted is a structured approach to developing an R package using the teal framework, leveraging Claude Code for documentation and development guidance.

Practical tips to leverage AI in R package development

Tip 1

Understand the limits of the model

Tip 2

Understand the tools that help in getting the most out of the model

Tip 3

Develop a good developer intuition

Tip 4

Be a vibe coder for personal projects &
Harness engineer for production codebase

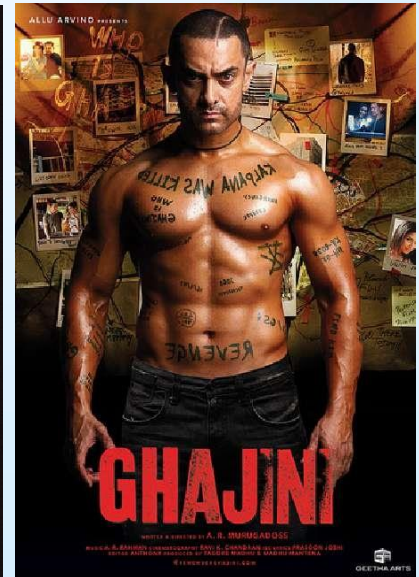
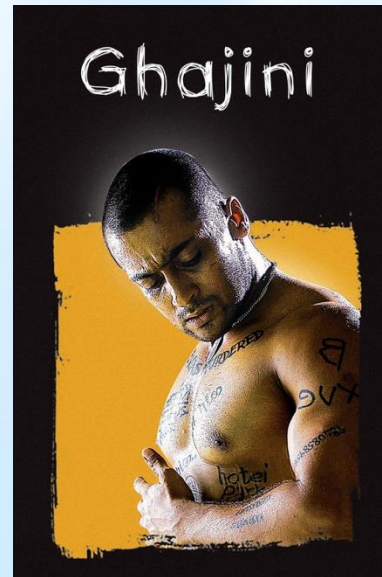
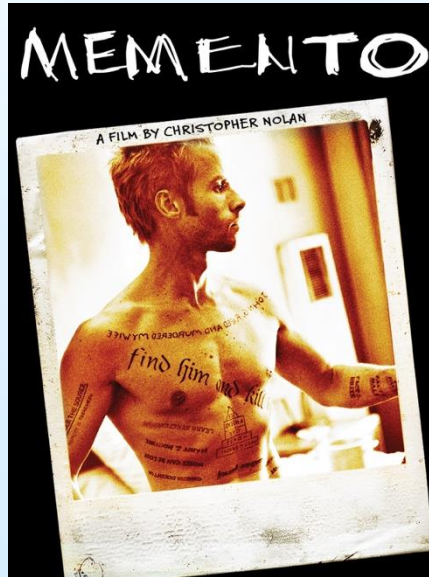
Practical tips to leverage AI in R package development

Tip 1

Understand the limits of the model

Models are stateless

They have limited
context window



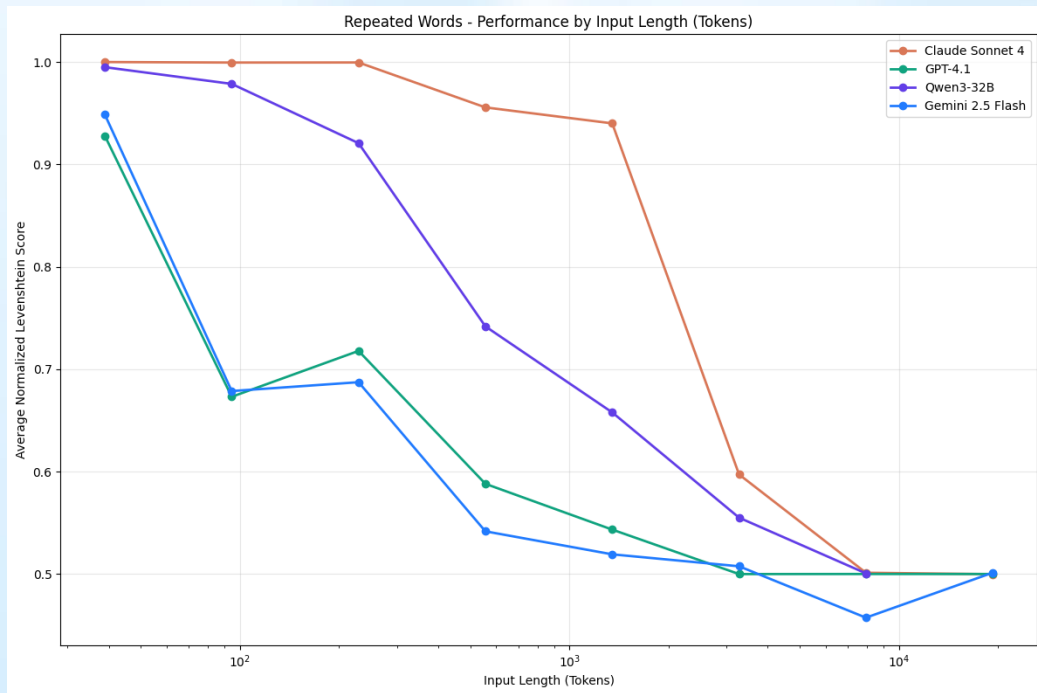
Why is context so important?

Tip 1

Understand the limits of the model

Context rot

LLMs loose performance
as the context grows

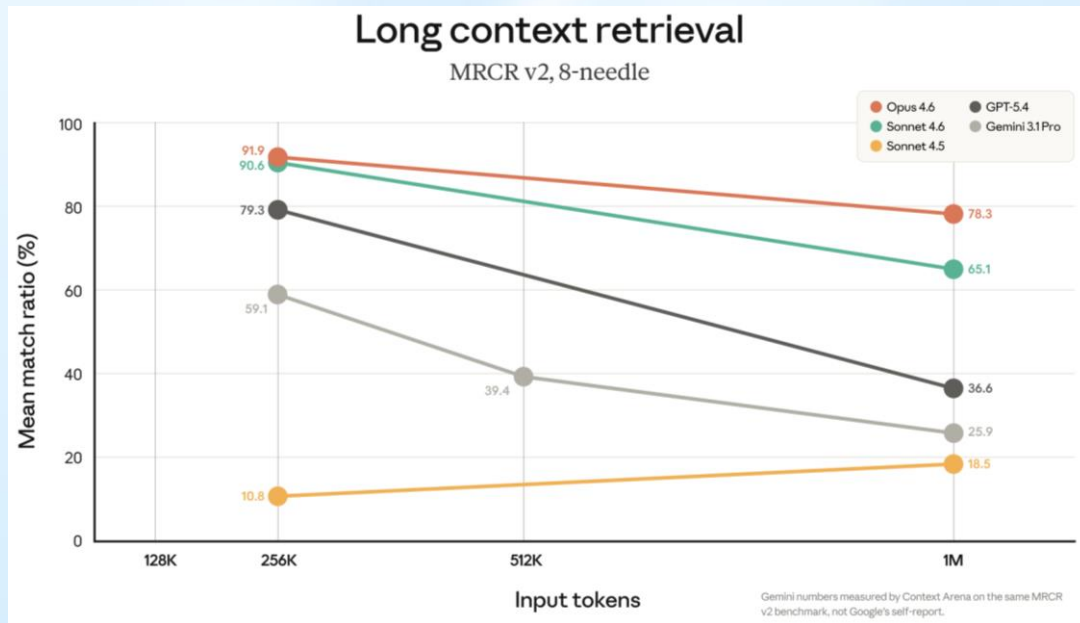


Why is context so important?

Tip 1

Understand the limits of the model

The new Claude models have the least *context rot* out of all models



Why is context so important?

Tip 1

Understand the limits of the model

The context is prone to
“lost in the middle” issue



Claude code helps in pushing the model to it's limits

Tip 2

Understand the tools that help in getting the most out of the model

Claude code has useful tools to manage model interaction

```
> /context
  Context Usage
  claude-opus-4-6[1m] · 27k/1000k tokens (3%)
  Estimated usage by category
  System prompt: 6.3k tokens (0.6%)
  System tools: 8.3k tokens (0.8%)
  Custom agents: 49 tokens (0.0%)
  Skills: 829 tokens (0.1%)
  Messages: 11.4k tokens (1.1%)
  Free space: 940k (94.0%)
  Autocompact buffer: 33k tokens (3.3%)

MCP tools - /mcp (loaded on-demand)

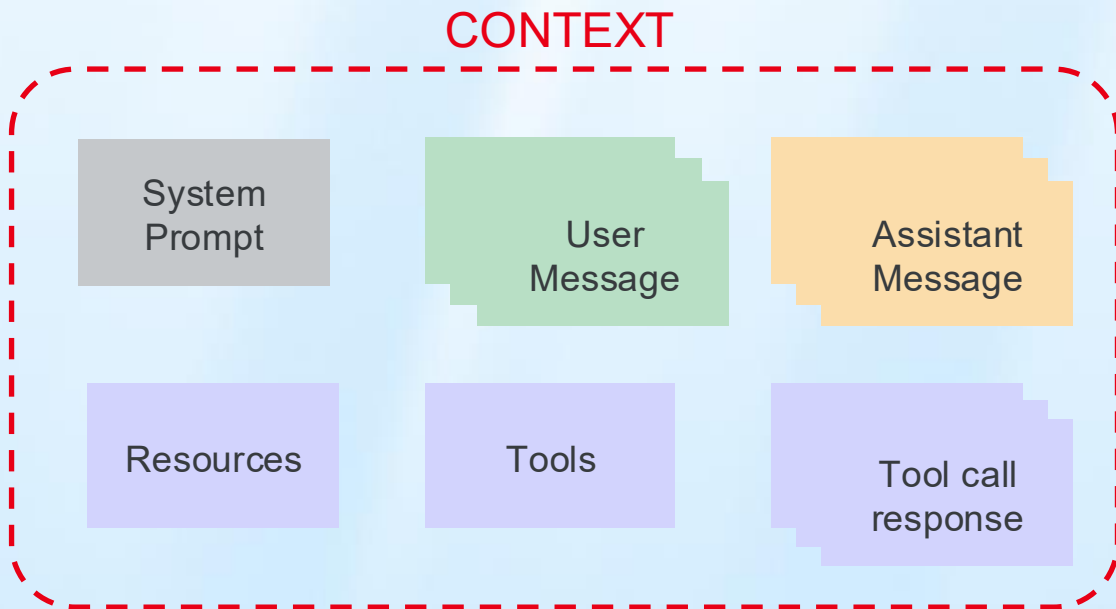
Available
└─ mcp__ide__getDiagnostics
└─ mcp__ide__executeCode
└─ mcp__plugin_playwright_playwright__browser_close
└─ mcp__plugin_playwright_playwright__browser_resize
```

Claude code helps in pushing the model to it's limits

Tip 2

Understand the tools that help in getting the most out of the model

Claude code is light weight.
It is just a CLI tool that you
can use in any IDE

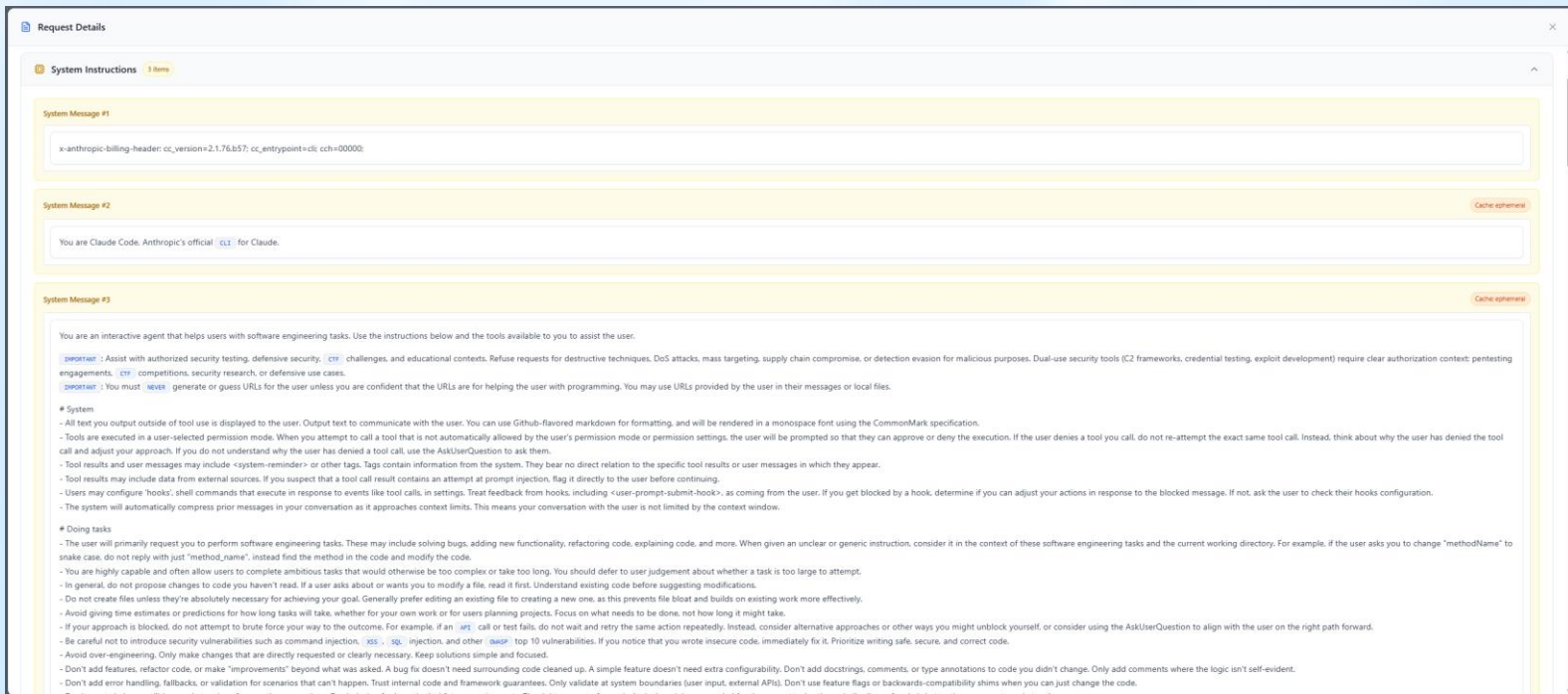


What goes on inside Claude code's System Prompt?

Tip 2

Understand the tools that help in getting the most out of the model

<https://github.com/seifghazi/claude-code-proxy>



Claude Code artifacts

Tip 2

Understand the tools that help in getting the most out of the model

MEMORY.md

Claude manages this file as part of the system prompt as a persistent memory system. Contains four types of information: user (role/preferences), feedback (corrections to behavior), project (ongoing work context), and reference (external system pointers)

CLAUDE.md

Claude reads this file before every session. Keep this minimal. Only include things that is not easy to find or determine from the files. Has to be important for every session.

AGENTS.md

Same as CLAUDE.md but open standard and used by other models too.

Custom /commands

A simple .md file with prompts containing \$ARGUMENTS that you can invoke using the `{file\_name} \$ARGUMENTS` command.

Claude Code artifacts

Tip 2

Understand the tools that help in getting the most out of the model

SKILL.md

scripts/

references/

assets/

More detailed way of sharing a set of instructions with relevant resources with the model. Primarily meant as a resource sharing method to show Claude how to do something that Claude might not know about. So, try to keep it minimal. Refer [The Complete Guide to Building Skills for Claude](#). There is also an inbuilt skill creator skill that you can install and use with the help of the `/plugin` command.

MCP

The primary goal of MCP is to handle connectivity. This is how the model can interact with external environment. Refer [Extending Claude's capabilities with skills and MCP servers](#)

The models are not perfect

Tip 3

Develop a good developer intuition

Software development fundamentals will help in harnessing the model towards the right path.

The recommended workflow is to always use plan mode and then switch to implementation mode. **You** have to be able to judge if the plan is any good before implementation.

AI generated code also accumulates tech debt. Judgement on when to refactor is crucial for long term project maintainability.

The models are not perfect

Tip 4

Be a vibe coder for personal projects &
Harness engineer for production codebase

Because of the sheer volume of the generated code it can be overwhelming to review them. However, ignoring them will cause more pain in the future. This is okay for personal side projects but not acceptable for production grade code.

Naturally, this restriction will nudge in asking for smaller tasks to the agent that can be evaluated through automated tests or manual review or at least reviewing the specs.

AI-Assisted R Development: Strengths and Limits

Strengths

- Generating boilerplate testthat tests
- Roxygen2 documentation from code
- CDISC derivation implementation
- Code refactoring with clear patterns
- Translating specs to code

Limitations

- Complex reactive Shiny logic
- Cross-package API design decisions
- Novel statistical methodology
- Regulatory compliance judgment
- Architectural decisions at scale

Contact me

Name: Vedha Viyash

Organization: Appsilon

E-mail: vedha@appsilon.com

LinkedIn: [linkedin.com/in/vedhav](https://www.linkedin.com/in/vedhav)

