

Is the AI Coding Assistant Catching Your Vibe?

A Practical Assessment of Generative AI
Tools in Clinical Data Analytics

Chendi Liao

PHUSE US Connect 2026 [ML22]
March 24, 2026



Disclaimer

- *The views and opinions expressed herein are solely those of the presenter and do not necessarily reflect the official position of the company*
- *Given the rapid advancement of AI technologies, please be advised that some of the content presented may be subject to obsolescence or diminished pertinence*

Background

-  Recursion. is a clinical-stage TechBio company. Our Recursion OS provides an AI-native, end-to-end drug discovery and development platform
- Tech-forward approach - state-of-the-art technology stack
- Clinical pipeline consists of mostly early phase and/or first-in-human programs
 - Close data monitoring, rapid turnaround
 - Frequent safety review and dose escalation decision meetings
- Built open-source based in-house analytics solutions to support rapid insight generation

Improving Patient Profile Report

Individual Patient Profile

Subject 01-701-1015

Demographics and Baseline

Treatment and Disposition Overview

Exposure

Adverse Events

Medical History

Concomitant Medications

Vital Signs

Labs

Responses

Subject 01-701-1023

Subject 01-701-1028

Subject 01-701-1033

Subject 01-701-1034

[DEMO] REC-XXX Clinical Data Review Report

Recursion Pharmaceuticals Confidential Document - For Internal Use Only.
Disclaimer - This demo/examples is created with dummy data

Prepared by Chendi Liao
Data Snapshot Date: 2025-XX-XX; Demo report generated on: 2025-10-22

Individual Patient Profile

Subject 01-701-1015

Demographics and Baseline

01-701-1015: Demographics and Baseline							
Demographics				Baseline			
Age(Sex/Race/Ethnic)	Country	Height (cm)	Weight (kg)	BM (kg/m2)	Diastolic BP (mmHg)	Systolic BP (mmHg)	Pulse (bpm)
63(F/White/H)	USA	147.32	54.43	25.07927	56	130	56

¹ NH = Not Hispanic or Latino, H = Hispanic or Latino

Treatment and Disposition Overview

01-701-1015: Treatment and Disposition							
Treatment Group			Treatment Info			Disposition Status	
Treatment Sequence	Safety-Evaluable ¹	Randomization Date	Treatment Start	Treatment End	Duration (Days)	End of Study Date ¹	Study Status ¹
Placebo	Y	2014-01-02	2014-01-02	2014-07-02	182	2014-07-02	COMPLETED

¹ NA is shown for subject who are ongoing for treatment or study follow-up.

Exposure

Copy CSV Excel Search:

01-701-1015: Exposure Listing

Drug Name	Visit	Start Date of Treatment	Start Day	End Date of Treatment	End Day	Dose (Unit)	Planned Dose	Dose Adjustment	Drug Form	Dosing Frequency	Route of Administration
PLACEBO	BASELINE	2014-01-02	1	2014-01-16	15	0	0		PATCH	QD	TRANSDERMAL
PLACEBO	WEEK 2	2014-01-17	16	2014-06-18	168	0	0		PATCH	QD	TRANSDERMAL
PLACEBO	WEEK 24	2014-06-19	169	2014-07-02	182	0	0		PATCH	QD	TRANSDERMAL

Showing 1 to 0 of 3 entries

Adverse Events

Copy CSV Excel Search:

01-701-1015: Adverse Events Listing

MedDRA Preferred Term	Start Day	End Day	Severity/Grade	Related?	Serious?	Action	Outcome
APPLICATION SITE ERYTHEMA	2		MILD	PROBABLE	N	DOSE NOT CHANGED	NOT RECOVERED/NOT RESOLVED
APPLICATION SITE PRURITUS	2		MILD	PROBABLE	N	DOSE NOT CHANGED	NOT RECOVERED/NOT RESOLVED
DIARRHOEA	8	10	MILD	REMOTE	N	DOSE NOT CHANGED	RECOVERED/RESOLVED

Showing 1 to 0 of 3 entries

Shortcomings with existing solution

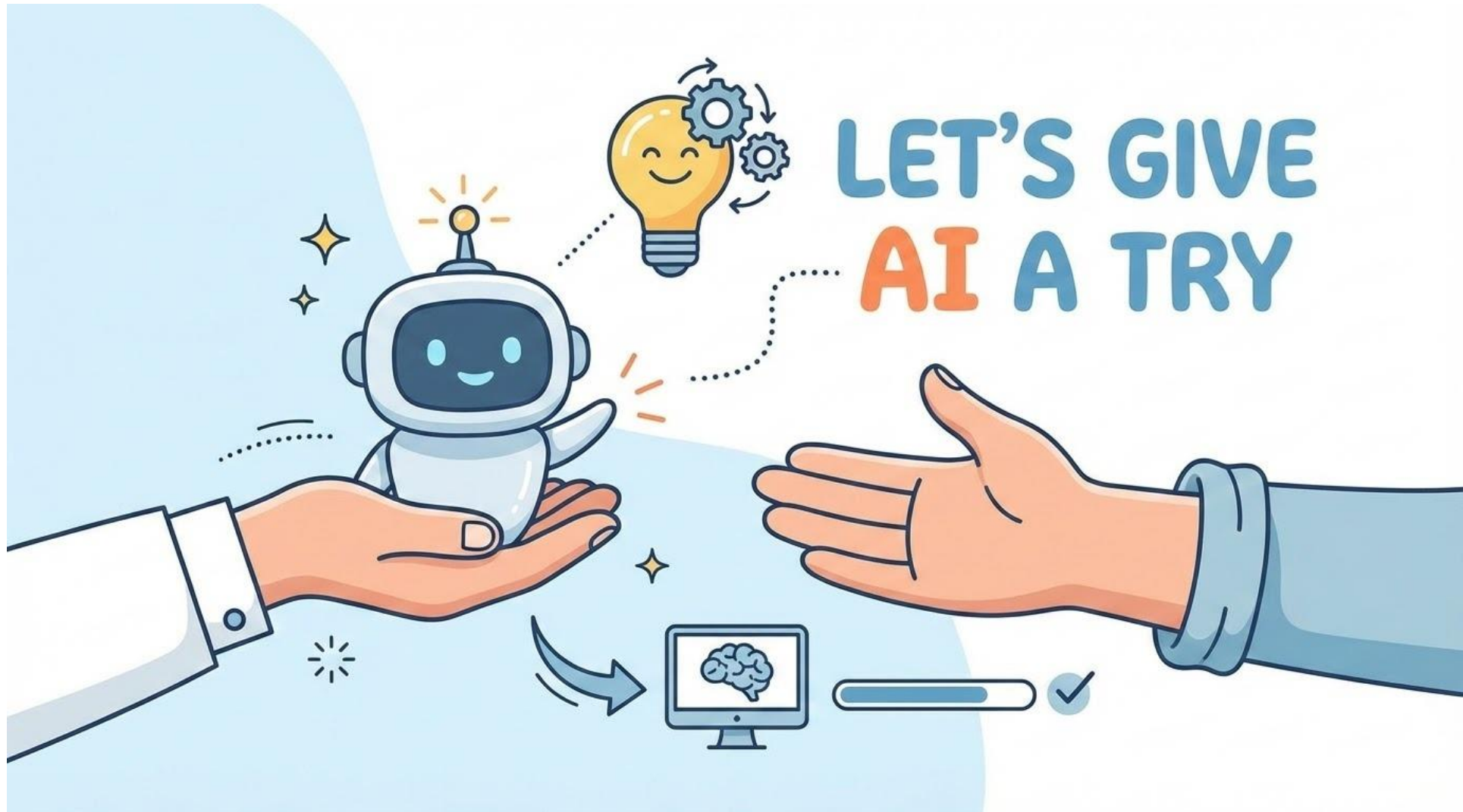
- Increasing data volume led to slower loading time
- Need for more summary statistics / visualizations
- Need for multiple format outputs



Transform report into dashboard

- Fast loading time, independent of trial size
- Reuse existing Rmarkdown code base
- Enable multiple output formats
- Balanced interactive vs static contents
- Minimize or avoid use of R-shiny

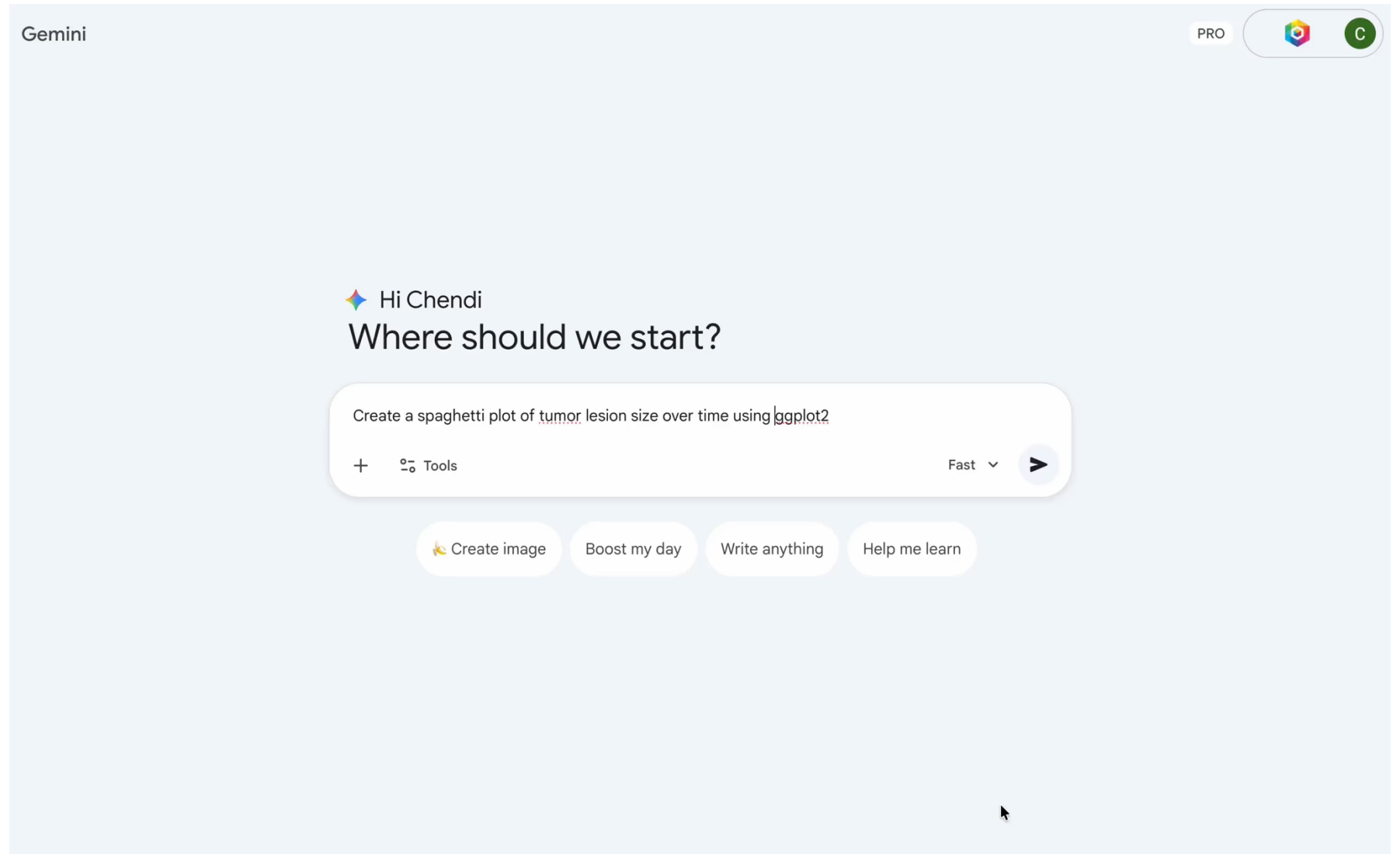
Can AI do it better/faster?



Four Main Types of AI Coding Assistants

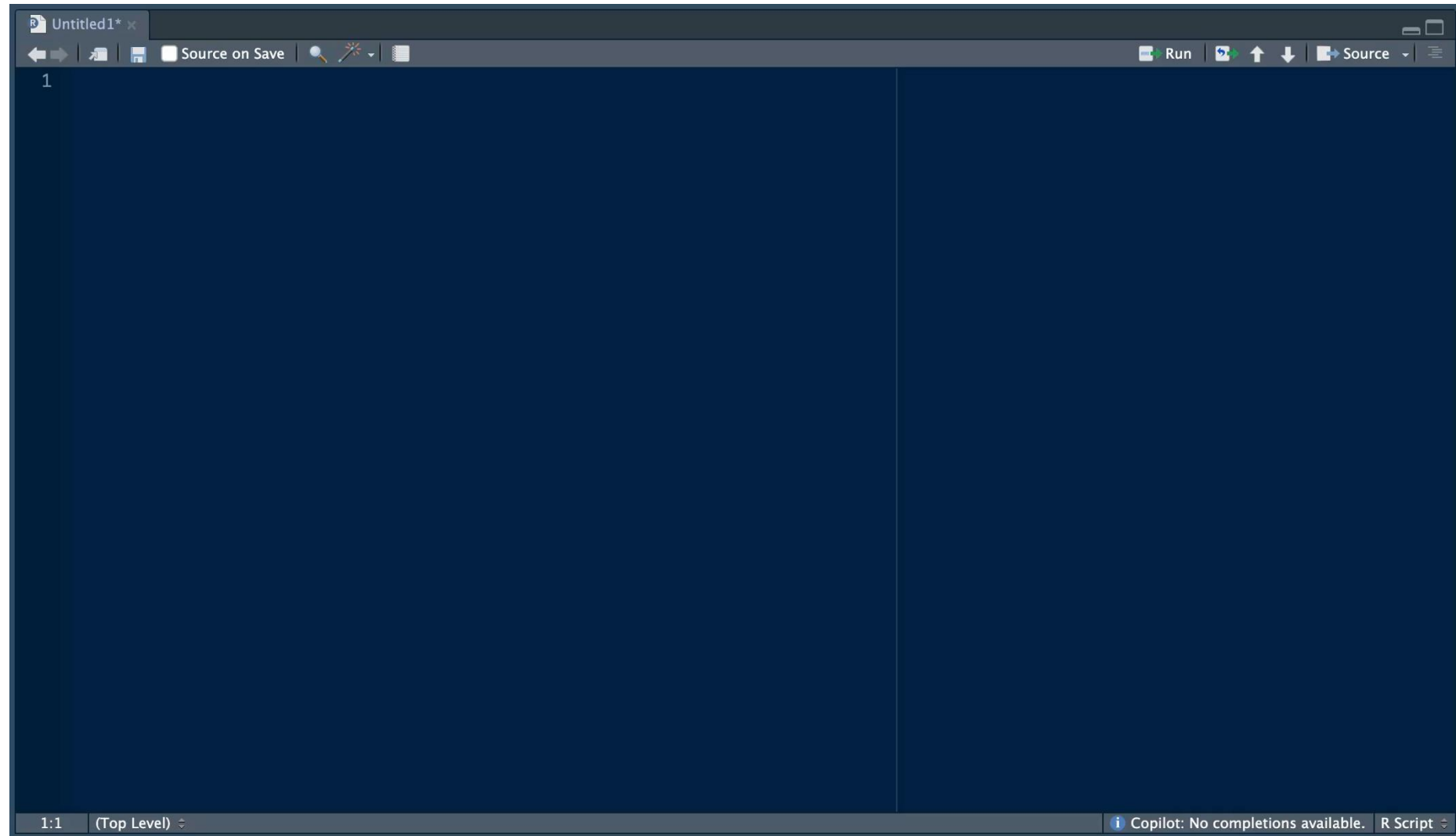
1. Conversational chatbots

- Google Gemini
- ChatGPT
- Perplexity AI



Four Main Types of AI Coding Assistants

1. Conversational chatbots
 - Google Gemini
 - ChatGPT
 - Perplexity AI
2. **Coding Partner (IDE extension)**
 - a. Github Copilot
 - b. Gemini Code Assist
 - c. Tabnine



Github Copilot (Jan 2026)

Four Main Types of AI Coding Assistants

1. Conversational chatbots

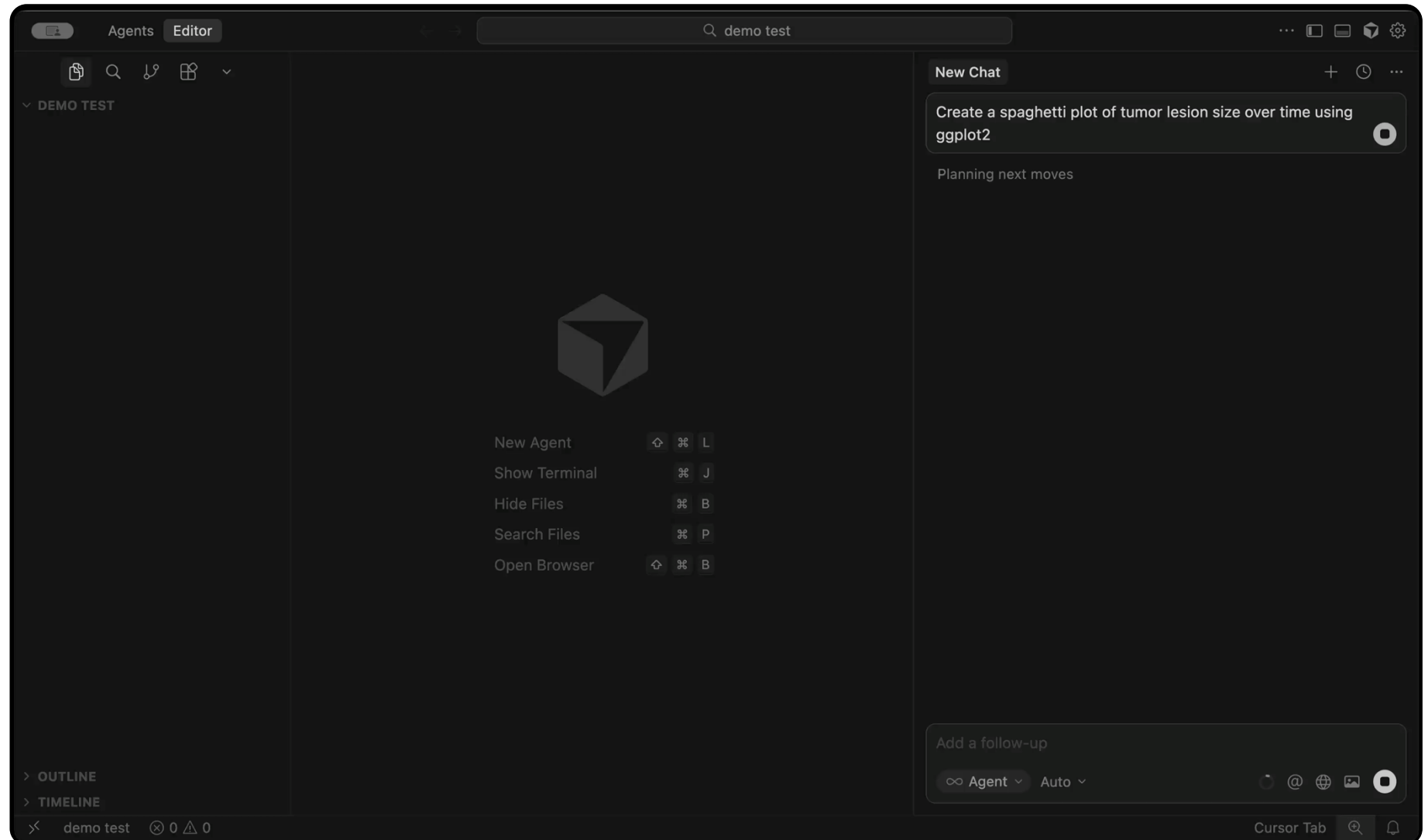
- Google Gemini
- ChatGPT
- Perplexity AI

2. Coding Partner (IDE extension)

- a. Github Copilot
- b. Gemini Code Assist
- c. Tabnine

3. AI-native IDEs

- a. Cursor AI
- b. Windsurf
- c. Aide



Cursor AI Auto Model (Jan 2026)

Four Main Types of AI Coding Assistants

1. Conversational chatbots
 - Google Gemini
 - ChatGPT
 - Perplexity AI
2. Coding Partner (IDE extension)
 - a. Github Copilot
 - b. Gemini Code Assist
 - c. Tabnine
3. AI-native IDEs
 - a. Cursor AI
 - b. Windsurf
 - c. Aide
4. **Command Line Interface (CLI)**
 - a. Claude Code
 - b. Gemini CLI
 - c. Codex CLI

A terminal window titled "demo test -- zsh -- 126x40". The terminal shows a login session for "chendi.liao@CLIA-4R1LGK-MBP" on "ttys000". The user has navigated to the directory "AIcoding/demo" and executed the command "test". The prompt is "chendi.liao@CLIA-4R1LGK-MBP demo test %".

```
demo test -- zsh -- 126x40
Last login: Fri Jan 30 19:58:25 on ttys000
chendi.liao@CLIA-4R1LGK-MBP ~ % cd AIcoding/demo\ test
chendi.liao@CLIA-4R1LGK-MBP demo test %
```

Comparisons of Popular AI Coding Assistants

Type	Conversational Chatbot	Coding Partner (IDE extension)	Standalone IDE	CLI
AI Tool Tested	Enterprise Google Gemini/ChatGPT	Github Copilot (via Rstudio)	Cursor AI	Claude Code
Embedded IDE	No	Yes	Yes	No
Multi-LLM support	No	No (possible in other IDEs)	Yes	No
Context Scope	None	Single file (can enable project-wide)	Full project / repo	Full local filesystem via Shell access
Multi-file edits	No	No	Yes	Yes
Code execution	No	Yes	Yes (automated with permission)	Yes (automated with permission)
Multi-application execution	No	No	Yes (require system permission)	Yes (require system permission)

Hit or Miss: Ideation

Task: Approaches to convert Rmarkdown into dashboard

Prompt

I currently have a set of Rmd code that generate an "html_document" based report for clinical trial data review. I publish this report to Posit Connect and share with stakeholders. The report has two main sections: 1) all data summaries, 2) by-patient data display. As time progress in the trial, the number of patients increases and data volume increase, and now the html report is getting too large, which takes a very long time to open and load. I want to make improvements to this reporting pipeline, potentially making it into a data review dashboard, with the following requirements:

1. Faster loading time, ideally independent of number of patients
2. Reuse existing code from Rmd as much as possible
3. Stay in R
4. Minimize or avoid use of R-shiny as it's difficult to maintain
5. Allow multiple format generation, for example html as well as pdf.

Can you suggest some potential solutions?

Gemini

It sounds like you've hit the "HTML bloat" wall. When an `html_document` embeds thousands of rows of data into a single file, the browser struggles to parse the massive JSON/JavaScript objects required to render tables.

To keep your R-based workflow while ditching the lag and avoiding the overhead of a Shiny server, here are three strategic solutions.

1. The "Quarto Dashboard" with OJS (Best for Speed)

If you haven't switched from R Markdown to **Quarto**, now is the time. Quarto Dashboards (released in v1.4) allow you to create professional layouts with very little code change.

2. Static Site Generation (The "Divide and Conquer" Method)

Instead of one massive file, move to a **multi-page website** structure using **Quarto Websites** or **R Markdown Sites**.

Recommendation: The "Hybrid Quarto" Approach

Given your requirements, I suggest moving to **Quarto**. You can maintain one project where:

1. A `_quarto.yml` file manages the structure.
2. Your summary statistics use your existing R code.
3. The "By-Patient" section uses **listings** or a **parameterized loop** to create individual pages.

This eliminates the "all-at-once" loading bottleneck while keeping you entirely within the R ecosystem and away from the complexities of Shiny reactivity.

Would you like me to provide a template code snippet for a Quarto loop that generates individual patient pages from a single dataset?

ChatGPT

I'll present 4 practical architectures, tradeoffs, and small code examples you can run now. After that I give a recommended path (minimal change, high ROI) and short next steps.

1) Best low-maintenance option — Pre-render per-patient HTML fragments + small index page (lazy-load)

2) Static index + precomputed JSON + client-side viewer (ideal for very fast search/filter)

How to reuse your current Rmd with minimal edits

- Turn the by-patient section into a parameterized chunk guarded by `if (!is.null(params$patient_id)) { ... }` so same Rmd can produce full summary (no param) or patient-only pages (with param).
- Keep all shared summary code in a `R/` script or an Rmd include (`child = "shared-summaries.Rmd"`) and source/include it from both the index-render and the patient-render — single source of truth.
- Use `rmarkdown::render()` programmatically to produce both the full summary HTML and per-patient HTMLs from the same source.

Concrete migration plan (recommended, incremental)

1. Parameterize your existing Rmd (add `params$patient_id`) and isolate patient display into a parameter-guarded chunk.
2. Add a small R script that: renders the Rmd (using `rmarkdown::render()`), or where you publish.
3. Create index page (small) that lists all patients (small, not all patient HTML).
4. Publish the `reports/` folder, prefer on-demand rendering.
5. Add PDF generation step (e.g., `pdf` output).

Example: parameterized Rmd guard pattern

```
R
---{r setup}
knitr::opts_chunk$set(echo = FALSE)
patient_id <- params$patient_id
data <- readRDS("precomputed/all_data.rds")
if (!is.null(patient_id)) {
  patient <- data$patients[[patient_id]]
}
```

All-patient summary (only when no param)

```
{r}
if (!is.null(patient_id)) {
  # summary tables/figures here
  print(summary_table)
}
```

Patient section (only when param set)

```
{r}
if (!is.null(patient_id)) {
  # render single patient detail (plots, tables)
  show_patient(patient)
}
```

Use with Caution: Vibe Coding

Task: Convert Rmarkdown script into Quarto website format

Prompt

Convert the Rmarkdown workflow in this directory into a Quarto website, save into a new subdirectory called `pp\_quarto`. The website should have two pages: one page for "All Subject Summaries", the other page for "Individual Patient Profile". The Patient Profile page should have a left sidebar that lists all patient IDs, each linked to a by-patient subpage generated with a loop, presenting data only related to the specified patient. Reuse all available R code from .Rmd files where possible.

The image displays a workflow for converting an R Markdown document into a Quarto website. On the left, a terminal window shows the R code used for rendering the website. The code includes comments and function calls like `quarto::quarto_render()` and `quarto::quarto_website()`. On the right, a preview of the resulting website is shown. The website has a blue navigation bar with links to 'Home', 'All Subject Summaries', and 'Individual Patient Profile'. The main content area features a title '[DEMO] REC-XXX Clinical Data Review Report', a disclaimer, and a 'Welcome' section. The 'All Subject Summaries' link is highlighted in the navigation bar.

```
pp_quarto > README.md > # Quarto Website for Patient Profile Reports
1 # Quarto Website for Patient Profile Reports
22 ## Building the Website
29
30 Or from R:
31
32 ```r
33 quarto::quarto_render()
34 ```
35
36 The website will be generated in the
37
38 ## Structure
39
40 - `index.qmd`: Homepage
41 - `all-subject-summaries.qmd`: Summ
42 - `patient-profile.qmd`: Main patie
43 - `patient-*.qmd`: Individual patie
44 - `patient-template.qmd`: Template
45 - `pp_func_analysis.R`: R functions
46 - `custom.css`: Custom styling
47 - `_quarto.yml`: Quarto website con
48
49 ## Notes
50
51 - Individual patient pages are gene
52 - Each patient page includes a side
53 - The current patient is highlighte
54 - All R code from the original `.Rm
55
```

Quarto website with patient profiles

Convert the Rmarkdown workflow in this directory into a Quarto website, save into a new subdirectory called `pp\_quarto`. The website should have two pages: one page for "All Subject Summaries", the other page for "Individual Patient Profile". The Patient Profile page should have a left sidebar that lists all patient IDs, each linked to a by-patient subpage generated with a loop, presenting data only related to the specified patient. Reuse all available R code from .Rmd files where possible.

[DEMO] REC-XXX Clinical Data Review Report Home All Subject Summaries Individual Patient Profile

[DEMO] REC-XXX Clinical Data Review Report

Recursion Pharmaceuticals Confidential Document - For Internal Use Only.

Disclaimer - This demo/examples is created with dummy data

AUTHOR
Prepared by Chendi Liao

PUBLISHED
October 22, 2025

Welcome

This website provides a comprehensive review of clinical data, including:

- [All Subject Summaries](#): Overview of demographics, baseline characteristics, and treatment-emergent adverse events across all subjects.
- [Individual Patient Profile](#): Detailed patient-level information with navigation to individual patient pages.

Confidential Document - For Internal Use Only

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE)
knitr::opts_chunk$set(out.width = '120%', fig.width = 10, fig.height = 6)
```

# Use with Caution: Creating New Analytical Contents

Task: Adding new summary table

## Prompt

For "All Subject Summaries" section, add a TEAE summary table by system organ class and preferred term using R package tern

```
pp_quarto > all-subject-summaries.qmd > TEAE Summary by System Organ Class and Preferred Term > (code cell)

78
79 ## TEAE Summary by System Organ Class and Preferred Term
80
81 ```{r aet02}
82 ## AET02 table - TEAE summary by SOC and PT
83 basic_table(
84 title = "Summary of Treatment Emergent Adverse Events by System Organ Class and Preferred Term",
85 show_colcounts = TRUE
86) |>
87 split_cols_by("TRT01A") |>
88 add_overall_col("All Subjects") |>
89 split_rows_by(
90 "AEBODSYS", child_labels = "visible",
91 split_fun = trim_levels_in_group("AEDECOD"),
92 label_pos = "topleft",
93 split_label = "System Organ Class"
94) |>
95 summarize_num_patients(
96 var = "USUBJID",
97 .stats = "unique",
98 .labels = c(unique = "n"),
99 .formats = list(unique = format_count_fraction_fixed_dp)
100) |>
101 count_occurrences(
102 vars = "AEDECOD",
103 .stats = "fraction",
104 .formats = c(fraction = format_count_fraction_fixed_dp),
105 .indent_mods = c(AEDECOD = 1L)
106) |>
107 build_table(df = ADAE_SE, alt_counts_df = ADSL_SE) |>
108 as_html(width = "90%")
109
110
```

Quarto website will TEAE summary table for all

For "All Subject Summaries" section, add a TEAE summary table by system organ class and preferred term

System Organ Class and Preferred Term. The table:

1. Groups by System Organ Class (AEBODSYS) — main row grouping
2. Shows Preferred Terms (AEDECOD) — nested under each SOC
3. Displays counts by treatment group — split by TRT01A with an "All Subjects" column
4. Shows subject counts and fractions — number and

```
processing file: all-subject-summaries.qmd
|.....| 77% [aet02] Error in `assert_proportio
n_value()` at tern/R/formatting_functions.R:148:3:
! Assertion on 'x' failed: Element 1 is not <= 1.
Backtrace:
 1. global .main()
 2. execute(...)
 3. rmarkdown::render(...)
 4. knitr::knit(knit_input, knit_output, envir = envir, quiet = quiet)
 at rmarkdown/R/render.R:768:5
 ...
 at evaluate/R/evaluate.R:141:5
17. base (local) withRestartList(expr, restarts)
18. base (local) withOneRestart(withRestartList(expr, restarts[-nr]), restarts[[nr]])
19. base (local) do.call(restart_handler, restart_args)
21. evaluate (local) fun(base::quote(<smplErrr>))
```

Summary of Treatment Emergent Adverse Events by System Organ Class and Preferred Term

System Organ Class	Placebo (N=86)	Xanomeline High Dose (N=72)	Xanomeline Low Dose (N=96)	All Subjects (N=254)
n	21 (24.4%)	36 (50.0%)	51 (53.1%)	108 (42.5%)
APPLICATION SITE BLEEDING	0/86	0/72	1/96 (1.0%)	1/254 (0.4%)
APPLICATION SITE DERMATITIS	5/86 (5.8%)	7/72 (9.7%)	9/96 (9.4%)	21/254 (8.3%)
APPLICATION SITE DESQUAMATION	0/86	0/72	1/96 (1.0%)	1/254 (0.4%)
APPLICATION SITE DISCHARGE	0/86	1/72 (1.4%)	0/96	1/254 (0.4%)
APPLICATION SITE DISCOLOURATION	0/86	0/72	1/96 (1.0%)	1/254 (0.4%)

Cursor Auto Model (Jan 2026)

# Use with Caution: Modifying Analytical Contents

Task: Convert analysis code into reusable functions

## Prompt

Convert all summary table code into reusable functions and save them in a new R script called *pp\_func\_safesum.R*. Then use the last newly created function to add a new table "Related TEAE Summary by SOC and PT".

The screenshot shows a code editor with two files: `all-subject-summaries.qmd` and `pp_func_safesum.R`. The `all-subject-summaries.qmd` file contains R code for generating summary tables. The `pp_func_safesum.R` file contains reusable functions for generating summary tables. The sidebar on the right provides instructions on how to use the new functions.

```
78
79 ## TEAE Summary by System Organ Class and Preferred
80
81 ```{r aet02}
82 ## AET02 table - TEAE summary by SOC and PT
83 basic_table(
84 title = "Summary of Treatment Emergent Adverse Events",
85 show_colcounts = TRUE
86) |>
87 split_cols_by("TRT01A") |>
88 add_overall_col("All Subjects") |>
89 split_rows_by(
90 "AEBODSYS",
91 split_fun = trim_levels_in_group("AEDECOD"),
92 label_pos = "topleft",
93 split_label = "System Organ Class"
94) |>
95 summarize_num_patients(
96 var = "USUBJID",
97 .stats = "unique",
98 .labels = c(unique = "n"),
99 .formats = list(unique = format_count_fraction)
100) |>
101 count_occurrences(
102 vars = "AEDECOD",
103 .stats = "fraction",
104 .formats = c(fraction = format_count_fraction),
105 .indent_mods = c(AEDECOD = 1L)
106) |>
107 build_table(df = ADAE_SE, alt_counts_df = ADSL_SE,
108 as_html(width = "90%")
109 ```
```

```
54
55 ## TEAE Summary by System Organ Class and Preferred
56
57 ```{r aet02}
58 ## AET02 table - TEAE summary by SOC and PT
59 makepp_teae_soc_pt(
60 адае_data = ADAE_SE,
61 адsl_data = ADSL_SE
62)
63 ```
64
65 ## Related TEAE Summary by System Organ Class and Preferred
66
67 ```{r aet03}
68 ## AET03 table - Related TEAE summary by SOC and PT
69 ADAE_REL_SE <- ADAE_SE |>
70 filter(REL == "Y") ## filter for related TEAEs only
71
72 makepp_related_teae_soc_pt(
73 адае_data = ADAE_REL_SE,
74 адsl_data = ADSL_SE
75)
```

Reusable summary table functions

Convert all summary table code into reusable functions, and save them in a new R script called `pp_func_safesum.R`. Then use newly created function to add a new "Related TEAE Summary by SOC and PT (new)"

2. Updated `all-subject-summaries.qmd` to:

- Source the new functions file
- Replace inline summary table code with function calls
- Add a new section "Related TEAE Summary by System Organ Class and Preferred Term" that filters for related TEAEs (`REL == "Y"`) and uses the new function

All functions include:

- Parameter documentation
- Default values matching the original code
- Customizable titles, labels, and widths
- Consistent structure and formatting

The code is now more maintainable and reusable. The new "Related TEAE Summary by SOC and PT" table appears after the existing TEAE summary table.

2 Files

- pp\_func\_safesum.R +162
- all-subject-summaries.qmd +27 -60

Plan, @ for context, / for commands

Agent Auto

Cursor Auto Model (Jan 2026)

# Use with Caution: *Better Approach* to Modifying Analytical Contents

Task: Convert analysis code into reusable functions, *by first creating a plan / specification*

## Prompt

Convert all summary table code into reusable functions and save them in a new R script called *pp\_func\_safesum.R*. Then use the last newly created function to add a new table "Related TEAE Summary by SOC and PT".

The screenshot shows a code editor with a file named `all-subject-summaries.qmd`. The code is R Markdown, starting with a title "TEAE Summary by System Organ Class and Preferred Term". It uses `basic_table()` to create a table with columns for treatment, system organ class, and preferred term. The code is split by "TRT01A" and rows by "AEBODSYS". It includes a `summarize_num_patients()` function to count unique subjects and a `count_occurrences()` function to count the fraction of subjects with at least one TEAE. The final output is an HTML table with a width of 90%.

On the right, a chat window titled "Reusable summary" contains a prompt: "Convert all summary table code into reusable functions, and save them in a new R script called 'pp\_func\_safesum.R'. Then use the last newly created function to add a new table 'Related TEAE Summary by SOC and PT'".

```
pp_quarto > all-subject-summaries.qmd > TEAE Summary by System Organ Class and Preferred Term

79 ## TEAE Summary by System Organ Class and Preferred Term
80 ```{r aet02}
81 basic_table(
82 title = "Summary of Treatment Emergent Adverse Events by
83 System Organ Class and Preferred Term",
84 show_colcounts = TRUE
85) |>
86 split_cols_by("TRT01A") |>
87 add_overall_col("All Subjects") |>
88 split_rows_by(
89 "AEBODSYS",
90 child_labels = "visible",
91 split_fun = drop_split_levels,
92 label_pos = "topleft",
93 split_label = "System Organ Class"
94) |>
95 summarize_num_patients(
96 var = "USUBJID",
97 .stats = "unique",
98 .labels = c(unique = "Total number of subjects with at
99 least one TEAE")
100) |>
101 count_occurrences(
102 vars = "AEDECOD",
103 .stats = "fraction",
104 .indent_mods = c(AEDECOD = -1L)
105) |>
106 build_table(df = ADAE_SE, alt_counts_df = ADSL_SE) |>
107 as_html(width = "90%")
108 ```
109
```

# Good Use: UI Customization

Task: Adding informational side-panel using CSS

## Prompt

Create a floating sidebar using CSS only for the page index.qmd. Name the new CSS file as `page-left-sidebar.css`. Place all contents including and below the callout into the floating sidebar. The sidebar style and color palette should match the defined bootstrap theme. The sidebar should contain two main parts: 1) main part showing the `callout-important` and `data-dco`, 2) footer showing the text about published date and author info.

index.qmd# page-left-sidebar.css

www > # page-left-sidebar.css > .floating-sidebar

```
1 /* Floating Left Sidebar for index.qmd */
2
3 /* Create the floating sidebar container */
4 .floating-sidebar {
5 position: fixed;
6 top: 0px;
7 left: 0px;
8 bottom: 0px; /* <-- UPDATED: This makes the height responsive */
9 width: 210px;
10 z-index: 1000;
11 background-color: var(--bs-body-bg);
12 border: 1px solid var(--bs-border-color);
13 border-radius: 0.25rem;
14 box-shadow: 0 0.5rem 1rem rgba(0, 0, 0, 0.15);
15 display: flex;
16 flex-direction: column;
17 }
18
19 /* Sidebar main content area */
20 .floating-sidebar-main {
21 flex-shrink: 0; /* Prevents the footer from shrinking */
22 text-align: left;
23 padding: 10px;
24 }
25
26 :: { .floating-sidebar-footer }
27
28 Published on `r format(Sys.Date(), "%B %d, %Y")`
29 Contact [Chendi Liao](chendi.liao@recursion.com) if you have any questions or
30 feedback.
31
32 ::
33
34 ::
```

Floating sidebar implementation

Create a floating sidebar using CSS only for the page index.qmd. Name the new CSS file as `page-left-sidebar.css`. Place all contents including and below the callout into the

Reviewing the current files to understand the structure.

Explored 3 files

Checking the Bootstrap Litera theme colors and structure. Creating the floating sidebar CSS file.

Searched bootstrap litera theme colors CSS variables

Creating the floating sidebar CSS file and updating index.qmd to include it and wrap the content appropriately.

# page-left-sidebar.css (new) +117 -1

/\* Floating Left Sidebar for index.qmd \*/

/\* Create a container for the page content

body > .quarto-container {

The CSS still works because it targets the same class names. The Quarto column syntax (:: { .class-name }) is cleaner and more native to Quarto. The floating sidebar behavior remains the same, with the content organized using Quarto's native syntax.

2 Files

Undo All

Keep All

Review

# page-left-sidebar.css +56 -1

index.qmd +15 -1

5 of 5

Undo %N

Keep %Y

Cursor Auto Model (Jan 2026)

## Rendered Result

[DEMO] REC-XXXX - Clinical Data

Confidential!

For Internal Use Only

This report summarizes and displays data from REC-XXXX.

Report is based on data received on:

Data	Snapshot Date
EDC	2025-08-01

Data cut-off date of 2025-07-01 has been applied for this report.

Published on Feb 06, 2026.  
Contact Chendi Liao if you have any questions or feedback.

All Sub

The following

For each subje

information. T

values indicat

Demog

Summary of D

Age

n

Mean (SD)

Median

Min - Max

Sex

n

F

M

Race

n

AMERICAN I

BLACK OR A

WHITE

# Good Use: Non-Analytical Repetitive Task

Task: Adding conditional Quarto blocks for pdf rendering

## Prompt

Help me update `\_allsum.qmd` so that all summary tables has a conditional content chunk for pdf rendering. Make sure to follow the example code in section `# Demographics and Baseline`. All summary tables are created with `maketbl\_` functions.

```
15 # Demographics and Baseline
16 ```{r}
17 # Demographic summary table
21 sumvars <- c("AGE", "SEX", "RACE", "ETHNIC", "COUNTRY")
22
23 ADSL_SE <- adam$se
24
25 dmt01 <- maketbl_
26 dmt01 |> as_html
27 ```
28
29 ::: {.content-visible when-format="pdf"}
30
31
32 ```{r}
33 #| label: "all-demographics-baseline"
34 dmt01 |> rtables::
35 ```
36
37 :::
38
39 # Adverse Events
40
41 Currently, AE is
42
43 - Date of AE onset
44 - Date of AE onset
```

Help me update `\_allsum.qmd` so that all summary tables has a conditional content chunk for pdf rendering. Make sure to follow the example code in section `# Demographics and Baseline`.

Review: Allsum.qmd summary tables PDF rendering

1 Pending Change

Undo All Keep All Find Issues Commit

52 hidden lines

```
53 ADAE_SE <- adam$adae |>
54 filter(TRTEMFL == "Y") |> ## filter for
55 left_join(ADSL_SE |> select(USUBJID, TI
56
57 # Layout for variables from adsl dataset
58 maketbl_aet01(ADAE_SE, ADSL_SE, aesi_v
59
60 as_html(width = "70%")
61 ```
62
63
64 ::: {.content-visible when-format="pdf"}
65
66 ```{r}
67 #| label: "all-aet01-pdf"
68 aet01 |> rtables::officer::tt_to_flex
69 ```
70
71 :::
72
73
74 ## TEAE Summary Tables
```

Help me update `\_allsum.qmd` so that all summary tables has a conditional content chunk for pdf rendering. Make sure to follow the example code in section `# Demographics and Baseline`.

All PDF chunks follow the same pattern as the example in the `# Demographics and Baseline` section:

- Use the `::: {.content-visible when-format="pdf"}` div wrapper
- Include a code chunk with label `*-pdf`
- Convert the table using `rtables::officer::tt_to_flex`

All 4 summary tables (including the existing `dmt01`) now have conditional PDF rendering chunks. No linting errors were found.

1 File Undo Keep Review

\_allsum.qmd +36 -3

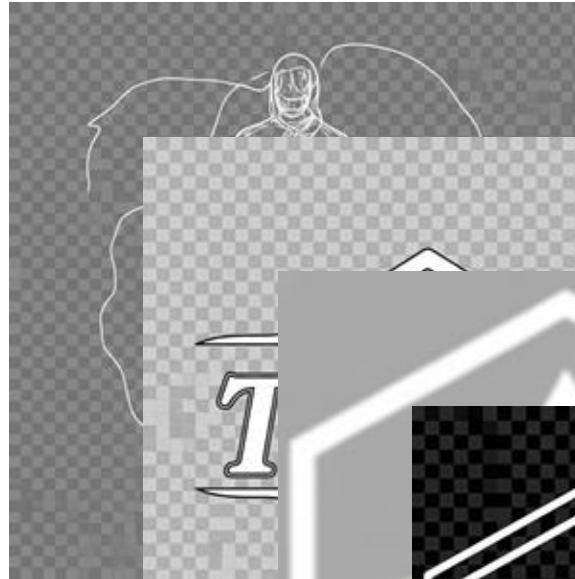
# An Anecdote

GPT seems to perform better in image processing than Gemini

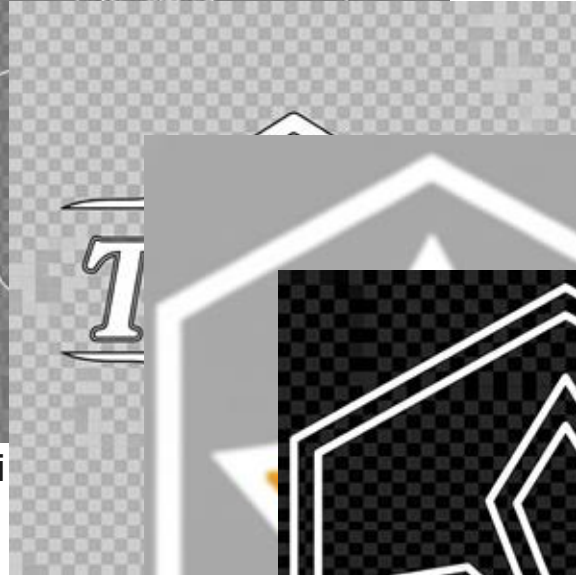
## Prompt



Convert this image to transparent background, replace dark blue outline with white color and all other colors removed



Gemini



Gemini



Gemini



Nano Banana Pro (Jan 2026)



ChatGPT 5.1 (Jun 2025)  
ChatGPT 5.2 (Jan 2026)

# Revamped Clinical Data Dashboard

All Subject Summaries – [DEMO]

connect.valid.rxx.io/content/5bf77a85-9092-470f-973c-4505bab35e61/

Work

[DEMO] REC-XXXX - Clinical Data Dashboard

All Subject Summaries

Patient Profiles

Contact

Confidential!

For Internal Use Only

This report summarizes and displays data from REC-XXXX.

Report is based on data received on:

Data

Snapshot Date

EDC

2025-08-01

Data cut-off date of 2025-07-01 has been applied for this report.

Published on Feb 06, 2026.

Contact [Chendi Liao](#) if you have any questions or feedback.

All Subject Summaries

The following analyses are restricted to subjects in the **safety-evaluable population** only.

For each subject, the study start date is the non-missing treatment start date, or the enrollment date in the case of missing treatment information. The study start date is used for all downstream calculation of the relative study day, unless otherwise indicated. Negative values indicate the number of days prior to subject's study start.

Demographics and Baseline

Summary of Demographics and Baseline Characteristics

	Placebo (N=86)	Xanomeline High Dose (N=72)	Xanomeline Low Dose (N=96)	All Patients (N=254)
Age				
n	86	72	96	254
Mean (SD)	75.2 (8.6)	73.8 (7.9)	76.0 (8.1)	75.1 (8.2)
Median	76.0	75.5	78.0	77.0
Min - Max	52.0 - 89.0	56.0 - 88.0	51.0 - 88.0	51.0 - 89.0
Sex				
n	86	72	96	254
F	53 (61.6%)	35 (48.6%)	55 (57.3%)	143 (56.3%)
M	33 (38.4%)	37 (51.4%)	41 (42.7%)	111 (43.7%)
Race				
n	86	72	96	254
AMERICAN INDIAN OR ALASKA NATIVE	0	1 (1.4%)	0	1 (0.4%)
BLACK OR AFRICAN AMERICAN	8 (9.3%)	9 (12.5%)	6 (6.2%)	23 (9.1%)
WHITE	78 (90.7%)	62 (86.1%)	90 (93.8%)	230 (90.6%)

On this page

Demographics and Baseline

Adverse Events

Overview of TEAE

TEAE Summary Tables

Lab Summaries

Aggregated Lab Results Over Time

Listing of Quantitative Lab Abnormalities

Other Formats

PDF

 Recursion.

19

# Optimal Human-AI Collaboration in Clinical Data Analytics

## Better Success

- Automate repetitive, non-analytical, and mundane tasks
- Refine UI (HTML/CSS customization)
- Scaffolding for well-defined outcome
- Straight-forward regular expression
- Write documentation, example codes, unit tests

## Caution and Require Oversight

- Domain or analysis-heavy codes
- High-risk high-impact tasks
- Clinical data processing/derivation
- Repetitive tasks involving modifying analysis codes
- Tasks with increasing complexity
- Avoid “generate insights from data”

# Takeaway

*In a highly regulated field where accuracy, reproducibility and traceability are non-negotiable...*

- Commercial AI coding tools are improving at “catching the vibe” of a statistical programmer
  - Vibe coding should be limited to non-analytical tasks
- Better ways to incorporate AI coding tools
  - Use on small, pieced, or low-risk tasks
  - Spec-driven code development
  - Models trained specifically for clinical data analytics - e.g. with MCP, RAG, PEFT
- Always include human-in-the-loop review and validation step
- AI-coding tools is best reserved for experienced/senior analysts
- Future of AI coding is capturing intent and writing better instructions/specification



Recursion.®

**Decoding biology to radically improve lives.®**