

Next-Gen Statistical Programming: AI as Your Coding Co-Pilot in R

Mark Rothe & Sumesh Kalappurakal, Johnson & Johnson, New Jersey, United States

刘晓畅 Xiaochang (Jack) Liu, Johnson & Johnson, Shanghai, China

1. ABSTRACT

As clinical trial data grows in complexity, AI-assisted code generation offers a transformative approach to statistical programming in R. This presentation explores how large language models can automate routine tasks—such as dataset creation, table generation, and exploratory analysis¹—while maintaining regulatory compliance. Through practical examples, we demonstrate how AI tools enhance productivity, reduce coding errors, and foster collaboration. We also address key challenges including validation, workflow integration, change management and governance. Attendees will gain insights into responsibly leveraging AI to augment human expertise in clinical data analysis.

2. DISCLAIMER

The content of Paper ML08 and its corresponding presentation has not been officially endorsed by Johnson & Johnson. Thus, the views, information, and opinions discussed in Paper ML08 and its corresponding presentation are those of the authors and do not necessarily reflect the official policy or position of Johnson & Johnson.

3. GENERATIVE & AGENTIC AI USAGE DISCLOSURE

The Generative AI model **Anthropic Claude™ Opus 4.5** ([Anthropic, 2025](#)) was used via the **AWS™** Agentic AI tool **Kiro™ CLI** ([Amazon Web Services, 2026](#)) during the authoring of this paper ([Section 5.2.1](#)). **Anthropic Claude Opus 4.6** ([Anthropic, 2026a](#)) was also used during some of the final QC as noted below. Kiro CLI and Claude Opus 4.5/4.6 were employed together in January and February 2026 as “**co-pilots**” by the authors of this paper as follows:

1. *De novo* generation of some paper content. Sections that were predominantly authored by AI have ([Anthropic, 2025](#)) ([Amazon Web Services, 2026](#)) citations in the section headers.
2. *De novo* generation of **ALL** R code for in-text tables (via `flextable` ([Gohel and Skintzos, 2025](#))) and *non*-Mermaid plots (via `dplyr` ([Wickham et al., 2023](#)), `ggplot2` ([Wickham et al., 2025](#)), `patchwork` ([Pedersen, 2025](#))) as well as *LaTeX* ([LaTeX Project, 2025](#)) equation code for **Total Score**. (Final plot QC performed by **Anthropic Claude Opus 4.6** ([Anthropic, 2026a](#)).)
3. *De novo* generation of **ALL Mermaid** ([Mermaid, 2025](#)) code (`*.mmd`) for in-text workflow plot PNG files displayed via **Quarto** ([Allaire et al., 2025](#)).
4. Adjudication of all content and code regarding correctness, clarity, consistency, and redundancy. (Final adjudications performed by **Anthropic Claude Opus 4.6** ([Anthropic, 2026a](#)).)
5. Editing and refactoring of all content and code as a result of the adjudications. (Final refactoring performed by **Anthropic Claude Opus 4.6** ([Anthropic, 2026a](#)).)
6. Searches of the World Wide Web for supporting content—including confirmation of each citation’s relevance. (Final WWW QC performed by **Anthropic Claude Opus 4.6** ([Anthropic, 2026a](#)).)

The human paper authors have collaborated with Kiro CLI, Claude Opus 4.5, and Claude Opus 4.6 and take full responsibility for the paper’s content. Multiple rounds of AI and human fact-checking were employed to confirm the paper’s content.

¹ This paper and corresponding presentation focus on **table**, **listing**, and **graph** generation. Future work will explore automation of other tasks.

4. INTRODUCTION

4.1 Skepticism, Curiosity, and Human-In-The-Loop (HITL) Validation

Throughout the course of the statistical programming profession, a few of our stakeholders have, at times, mistakenly believed that sweeping derived SAS® data set and summary table changes were effortlessly possible simply by “pressing a button”. Recent advances in AI have “moved the needle” a few millimeters closer to making this mistaken fantasy more plausible. However, the salient quote about AI from Posit™’s CTO Joe Cheng ([Cheng, 2024](#)),

Stay skeptical, but also, let yourself be curious.,

should make every statistical programmer’s “Spidey Sense” ([Wiktionary, 2026b](#)) tingle—signaling the inherent pitfalls of AI’s “Siren’s Song” ([Wiktionary, 2026a](#)), as illustrated by the following chat dialogue with Kiro CLI²:

Human User 🧑

Are you aware of the `ggtangle` R package? You **CAN** explore the internet (e.g., CRAN) to help answer this question.

Kiro CLI 🤖 : *Using Anthropic Claude Opus 4.5*

Searching the web for: `ggtangle` R package CRAN (using tool: `web_search`)

Here’s what I found:

`ggtangle` is an R package available on CRAN that extends `ggplot2` to support network visualization. It’s inspired by the “Method 1” approach from the `ggtree` package (Yu, 2018) and is designed to work with network-associated data.

Key details:

- Current version: **0.0.9**
- Purpose: Drawing networks with data using the `ggplot2` framework
- Author: Guangchuang Yu (same author as `ggtree`)

You can install it with:

```
r  
install.packages("ggtangle")
```

References:

1. Package `ggtangle` - <https://cran.r-project.org/web/packages/ggtangle/index.html>
2. `ggtangle` README - <https://cran.r-project.org/web/packages/ggtangle/readme/README.html>

Human User 🧑

Are you certain that **0.0.9** is the current version?

Kiro CLI 🤖 : *Using Anthropic Claude Opus 4.5*

You’re right to question that. The current version is **0.1.1**, not **0.0.9**. The search results I initially referenced were slightly outdated. Thanks for the catch.

✓ **0.1.1** was independently confirmed by a human to be the correct version as of *16-January-2026*.

The numeric difference between `ggtangle` ([Yu, 2026](#)) versions **0.1.1** and **0.0.9** is likely not show-stopping—even though there could be breaking changes with the up-versioning. However, similar numeric errors in p-values, confidence intervals, survival estimates, risk differences, etc. could translate into unseen safety signals, efficacious investigational product results

² See [Section 5.2.1](#) for a definition of **Kiro CLI**.

becoming equivalent to placebo, patients being projected to live longer than really possible, or failed clinical trials. “**Stay(ing) skeptical**” through **human-in-the-loop (HITL) validation** and **independently fact-checking** were important to correcting Kiro CLI’s initial answer, and this same healthy doubt and independent validation must remain as cornerstones embedded in any AI-assisted code generation solution.

4.2 Context

Simple chat dialogues with Kiro CLI do not only underscore the importance of HITL healthy doubt and independent validation, but also demonstrate the importance of the **context** used by AI to generate responses. Context is “a frame (Goffman 1974) that surrounds the **(communicative) event** being examined and provides resources for its **appropriate interpretation**” (Goodwin and Duranti, 1992). The **communicative event** is the **question-answer** or **request-response** interaction with AI:

4.2.1 Independent Question-Answer Round 1

Human User 🗣️

Are you aware of the `ggtangle` R package? You **CANNOT** explore the internet (e.g., CRAN) to help answer this question.

Kiro CLI 🤖 : *Using Anthropic Claude Opus 4.5*

Searching knowledge base for: `ggtangle` R package across all contexts (using tool: `knowledge`)

I don’t have any information about the `ggtangle` R package in my indexed knowledge bases. The search results returned content about general R package development, `ggplot2`, and SAS programming, but nothing specifically about `ggtangle`.

In the above question-answer round, Kiro CLI was instructed **NOT** to search the internet. Thus, context available via the World Wide Web was off-limits, but Kiro CLI was undeterred and conducted queries of Kiro CLI’s indexed knowledge bases for `ggtangle`. **Knowledge bases** are vectorized stores of text and code that are created using the Kiro CLI command `/knowledge`.

4.2.2 Independent Question-Answer Round 2

Human User 🗣️

Are you aware of the `ggtangle` R package? You **CANNOT** explore the internet (e.g., CRAN) to help answer this question. You **CANNOT** explore any local knowledge bases to help answer this question.

Kiro CLI 🤖 : *Using Anthropic Claude Opus 4.5*

I’m not familiar with a package called `ggtangle`. It’s not something I have knowledge of from my training data.

If it exists, it may be a newer or niche package that was released or became notable after my knowledge cutoff, or it could be a less widely-used package that wasn’t well-represented in my training data.

To learn about it, you could check CRAN directly or run `?ggtangle / help(package = "ggtangle")` in R if you have it installed.

Context was not made available to Kiro CLI during the second question-answer round, forcing it to rely on knowledge from the training data used for Anthropic’s Claude Opus 4.5. Kiro CLI was unable to provide the desired answer from its training data, stating “it may be a newer or niche package that was released or became notable after my knowledge cutoff”, thus underscoring the critical importance of providing the right context to AI.

4.3 Model Context Protocol (MCP) (Anthropic, 2025) (Amazon Web Services, 2026)

In two of the previous question-answer sessions with Kiro CLI, Kiro included the following two parenthetical statements:

- (using tool: `web_search`)
- (using tool: `knowledge`)

`web_search` and `knowledge` are **built-in tools** that Kiro CLI used to search the World Wide Web and knowledge bases, respectively. While these particular tools are native to Kiro CLI, the **Model Context Protocol (MCP)** enables external MCP Servers to expose *additional* tools, resources, and prompts to AI—helping AI systems overcome the knowledge cutoff of the underlying LLM. In short, **MCP** enables AI **Models** to receive **Context** via a standardized **Protocol**.

4.3.1 What is MCP? (Anthropic, 2025) (Amazon Web Services, 2026)

MCP is an open standard introduced by Anthropic on November 25, 2024 (Anthropic, 2024). Major cloud providers including AWS, Azure™, and Google Cloud™ have built MCP-compatible servers and integrations, while AI platforms like Claude Desktop™ and Kiro CLI³ have adopted MCP as hosts. Think of MCP as the “**USB-C for AI**”—a commonly-used AI community analogy: instead of building custom connectors for every integration, MCP provides a universal protocol for connecting AI agents to any compatible external service.

For statistical programmers, this means AI assistants can query clinical trial databases, access regulatory documentation, and integrate with clinical data systems without the need to write custom *API* code.

4.3.2 Architecture (Anthropic, 2025) (Amazon Web Services, 2026)

MCP follows a **host-client-server** architecture (Model Context Protocol, 2025)—as displayed in Figure 1 and Table 1:

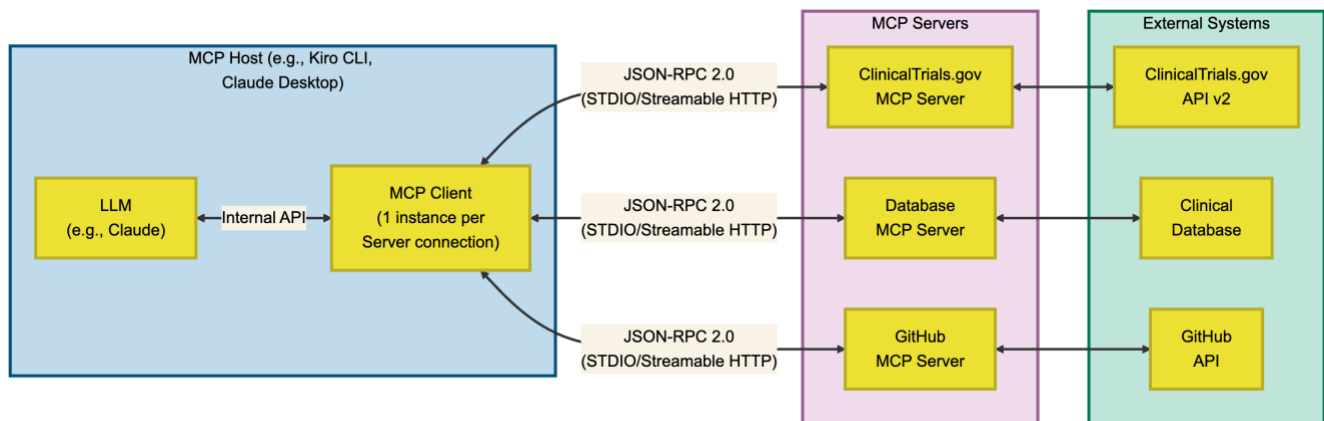


Figure 1: MCP Architecture: Host, Client, Server, and External Systems

³ See Section 5.2.1 for a definition of Kiro CLI.

Table 1: MCP Architecture Components

Component	Role
MCP Host	The outer application (Kiro CLI, Claude Desktop) that runs the AI and orchestrates connections
MCP Client	Created by the Host; one Client instance per Server connection; handles JSON-RPC communication
MCP Server	Program exposing tools, resources, or prompts to clients
Transport Layer	JSON-RPC 2.0 via STDIO (for local processes) or Streamable HTTP (for remote/web access)

4.3.3 Why MCP Matters (Anthropic, 2025) (Amazon Web Services, 2026)

MCP eliminates the “**M×N integration problem**” (Django Stars, 2025)—the need for **custom connectors** between every MCP host and external system. Instead of M×N connections, MCP requires only **M+N**: each MCP host (e.g., Kiro CLI, Claude Desktop) implements MCP client support once, and each external system implements one MCP server. For example, connecting 3 AI assistants to 5 external systems would require **15** custom integrations (3×5) without MCP, but only **8** (3+5) with MCP.

To further illustrate the **M+N** benefits: suppose a pharmaceutical company wants their AI assistants to access [ClinicalTrials.gov](#), an internal safety database, and a document management system. Without MCP, integrating these 3 external systems with 3 MCP hosts (Kiro CLI, Claude Desktop, GitHub Copilot™) would require **9** custom integrations—each with its own authentication, data formatting, and error handling. With MCP, the company builds **2** MCP servers and uses the ClinicalTrials.gov MCP server, namely `clinicaltrials.gov-mcp-server`⁴. All 3 MCP hosts can immediately use them—**6** components (Figure 3) instead of **9** custom integrations (Figure 2), with each new MCP host or external system requiring only 1 additional implementation rather than N or M new connectors.

All 9 host-to-server connections in Figure 3 use the identical MCP protocol. Each of the 3 MCP servers is built once and serves **ALL** hosts—no per-host customization required. Adding a fourth MCP host would require **0 new server code**; adding a 4th data source would require only **1** new MCP server that all existing hosts could immediately use.

For clinical research, this means the ClinicalTrials.gov MCP server can serve Claude, Kiro, GitHub Copilot, and any other MCP host (MCP-compatible AI assistant) without additional integration work.

Important Caveat: When using MCP servers with clinical data, statistical programmers should validate and trust the server source. Open-source servers like `clinicaltrials.gov-mcp-server` can be audited for compliance with data handling requirements, but any MCP server accessing sensitive or regulatory data should undergo appropriate security review and validation before use in production workflows.

⁴ The `clinicaltrials.gov-mcp-server` (Hand, 2025) is an open-source MCP server developed by Casey Hand that provides AI agents with programmatic access to the [ClinicalTrials.gov](#) database.

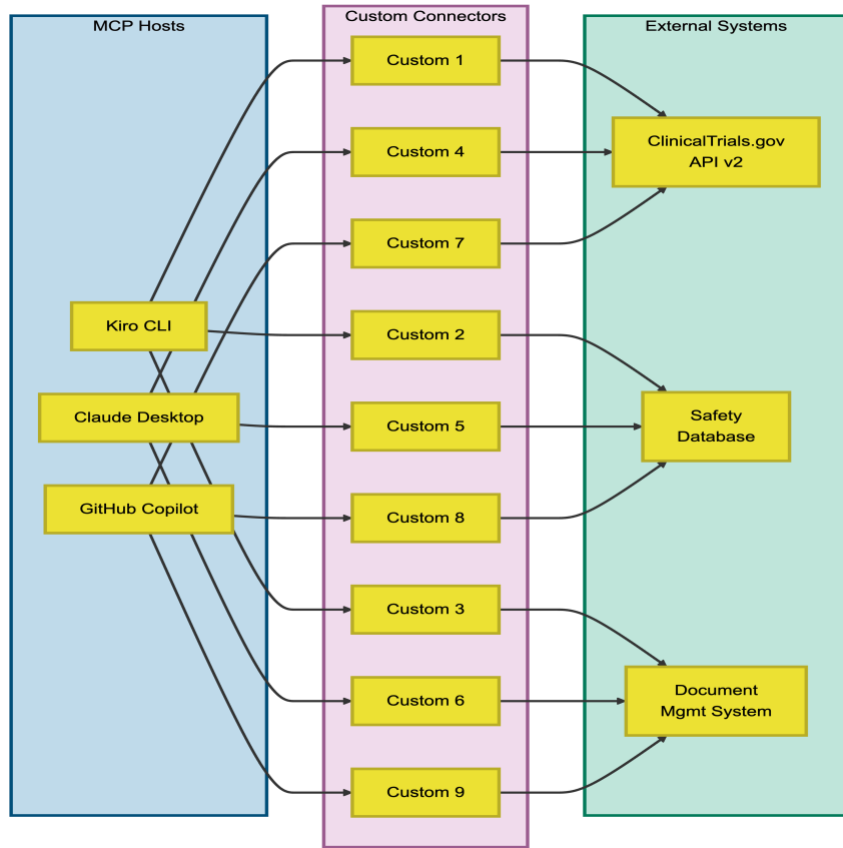


Figure 2: Without MCP: 9 Custom Integrations Required

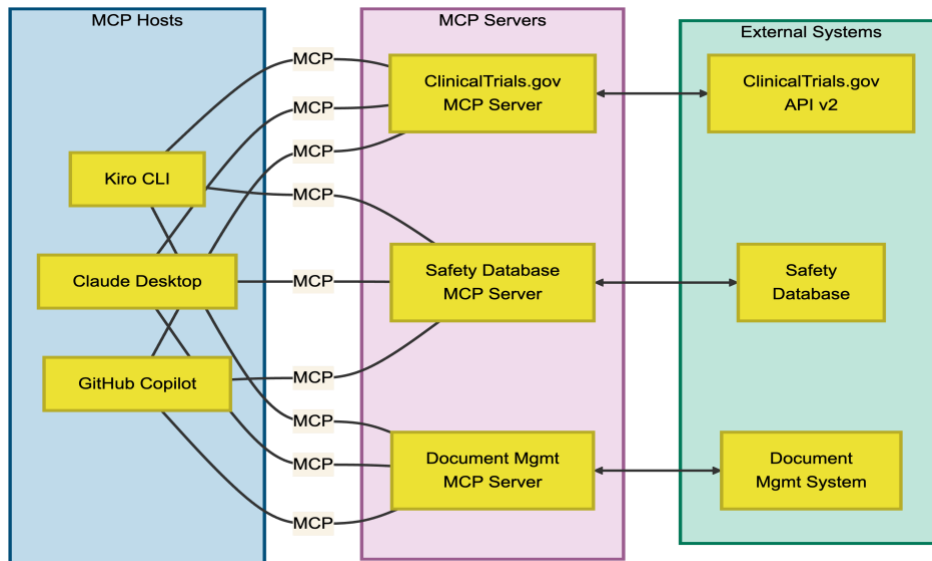


Figure 3: With MCP: 6 Components (3 Hosts + 3 Servers)

5. METHODS

5.1 Generative AI Upskilling Sessions

R and other open source languages like Python have given statistical programmers opportunities to change **HOW** they solve problems as well as have **expanded the types of problems** statistical programmers can routinely tackle—such as authoring & sharing interactive data visualizations with stakeholders and running reproducible analyses via literate programming. AI-assisted code generation offers a transformative approach to statistical programming that potentially will have an impact **significantly greater** than open source adoption—necessitating the incorporation of a change management process that sets up statistical programmers for success as “**strategic change enthusiasts**” who leverage the right AI tools and technologies to solve the right problems at the right time. The latest AI craze may not translate into tangible ROI. Thus, the 3 AI bellwethers discussed in the Introduction ([Section 4](#)),

1. **Skepticism, Curiosity, and Human-In-The-Loop (HITL) Validation**
2. **Context**
3. **Model Context Protocol (MCP)**

were chosen as the cornerstones for Generative AI upskilling sessions conducted among ~90 statistical programmers, statisticians, and other colleagues. Five training sessions were held in Fall 2025 as follows:

- i. Overview of the [VS Code IDE \(Microsoft Corporation, 2025\)](#)⁵
- ii. Intro to Code Development via Prompting
- iii. Effective Prompting
- iv. Using Context
- v. Using MCP Tools

Each upskilling session was recorded and comprised of 3 sections:

- a. Background of the concept—presented by a subject matter expert.
- b. Live demo of the concept in-action—supported by reference [Quarto™](#) HTML documentation published via [Posit Connect™](#).
- c. Assignment of hands-on exercises to help learners gain a better understanding.

Open office hours were made available to learners—including a dedicated MS Teams™ channel for questions/issues, and exercise solutions were shared between sessions.

5.2 Agentic AI ([Anthropic, 2025](#)) ([Amazon Web Services, 2026](#))

Agentic AI systems promise to transform many aspects of human-machine collaboration, especially in areas of work that were previously insulated from AI-led automation, such as proactively managing complex IT systems to pre-empt outages; dynamically re-configuring supply chains in response to geopolitical or weather disruptions; or engaging in realistic interactions with patients or customers to resolve issues.—Mark Purdy, Harvard Business Review (Purdy, 2024)⁶

The Generative AI upskilling sessions not only provided attendees with the experience of prompting LLMs to synthesize code but also introduced them to collaborating with Kiro CLI⁷ ([formerly Amazon Q Developer™ CLI](#)) in a “**pilot**” / “**co-pilot**” relationship—an example of “**Agentic AI**”(Section 5.2.1).

⁵ IDE: [Integrated Development Environment](#); When the curriculum for the upskilling sessions was originally developed, it was decided to use the Amazon Q Developer plug-in extension for the VS Code IDE. The VS Code IDE is very new for many learners, so the first training session was dedicated to understanding and using it—including how to use R and Q in the VS Code IDE.

⁶ Quote sourced via Atera’s compilation of agentic AI quotes ([Brown, 2025](#)).

⁷ Kiro CLI (formerly Amazon Q Developer CLI), Amazon Q Developer CLI, and the Amazon Q Developer plug-in extension for the VS Code IDE are all examples of Agentic AI from AWS and have very similar levels of AI functionality. To prevent unnecessary redundancy, “**Kiro CLI**” will be used as the synonymous term for the remainder of this paper.

5.2.1 Kiro CLI, An Example of Agentic AI (Anthropic, 2025) (Amazon Web Services, 2026)

Kiro CLI is an AI-powered development assistant built by AWS that runs directly in your terminal⁸. It combines conversational AI with practical development tools to help you:

- Read, write, and modify files in your codebase.
- Execute shell commands.
- Search and navigate code semantically (via built-in Tree-sitter (Wikipedia, 2026e) analysis or optional LSP (Wikipedia, 2026c) integration).
- Perform AST-based (Wikipedia, 2026a) pattern search and code transformations.
- Manage AWS resources through CLI calls.
- Query indexed knowledge bases for context-aware answers (Sueiras, 2025).
- Search the web and fetch page content.
- Delegate complex tasks to specialized subagents that work in parallel.
- Supports extending capabilities with external tools via MCP Servers.

What makes it different from a chatbot (i.e., Generative AI): (Table 2)

Unlike a standalone chat interface, Kiro CLI is deeply integrated with your local development environment. It has access to your filesystem, can run commands, understands your project structure through code intelligence, and can take action—not just give advice. One can think of it as a **co-pilot** or **coding partner** that lives in your terminal, capable of both answering questions and performing tasks.

5.2.2 Definition of Agentic AI (Anthropic, 2025) (Amazon Web Services, 2026)

Agentic AI refers to AI systems capable of making autonomous decisions on how to achieve a goal, then executing on those decisions—though responsible deployment requires HITL checkpoints at critical decision points (Oracle, 2025). Unlike traditional generative AI that responds to individual prompts, agentic systems:

- **Pursue goals autonomously:** Given a high-level objective, the agent determines *how* to achieve it.
- **Orchestrate multi-step workflows:** Tasks are decomposed into sequential or parallel sub-tasks.
- **Use tools dynamically:** Agents invoke external tools (databases, APIs, MCP servers) as needed—MCP is the core protocol that makes this possible (Anthropic, 2024), providing the standardized “USB-C for AI” interface (Django Stars, 2025) discussed in Section 4.
- **Maintain context across steps:** Memory and state persist throughout the workflow.⁹
- **Coordinate with other agents:** Complex tasks may involve delegation to specialized sub-agents (Zamanian, 2024).

Table 2: Generative AI vs. Agentic AI¹⁰

Characteristic	Generative AI	Agentic AI
Interaction model	Single prompt → single response	Goal → autonomous workflow → result
Task complexity	Atomic tasks	Multi-step, decomposable tasks
Tool use	None or user-directed	Autonomous tool selection and invocation
Context management	Within conversation window	Persistent across workflow stages

⁸ End-users type prompts and commands directly in the terminal; hence, the term “Command Line Interface” or the acronym “CLI” (Wikipedia, 2026b).

⁹ An agentic system maintains state beyond a single conversation turn—across multiple phases of a workflow. The key distinction: Gen AI context is **ephemeral** and **bounded** (*lives and dies with the conversation*). Agentic AI context is **persistent** and **cumulative** (*builds up across stages and can be resumed*).

¹⁰ Characteristics synthesized from Oracle (2025), Merit Data & Technology (2025), and general AI literature.

Human involvement	Required for each step	At defined checkpoints
Error handling	Returns error or hallucinates; user must retry	Can retry, adapt strategy, or escalate to human
Output	Text, code, or content	Completed objectives with artifacts

5.2.3 Responsible Agentic AI: Human-in-the-Loop Design (Anthropic, 2025) (Amazon Web Services, 2026)

While agentic AI offers significant productivity gains, autonomy without oversight poses risks—particularly in regulated environments like clinical research where errors can affect patient safety, regulatory submissions, and trial integrity. **Human-in-the-loop (HITL)** design can address this by embedding human oversight at critical decision points (Merit Data & Technology, 2025).

HITL is not a single mechanism but a design philosophy that determines how much autonomy agents have and when humans must intervene. Autonomy levels fall into three categories as delineated in Table 3:

Table 3: Autonomy Levels in Agentic AI Systems¹¹

Level	Description	Clinical Applicability
Supervised	Agents perform tasks but require human approval before execution	High-risk outputs requiring sign-off (e.g., efficacy tables, safety summaries, regulatory submission components)
Semi-Autonomous	Agents execute within predefined boundaries; out-of-bounds actions are escalated to humans	Routine derivations with automated validation rules (e.g., ADSL variable derivations with programmatic checks)
Fully Autonomous	Agents act independently with minimal oversight	Low-risk, well-defined transformations with established patterns (e.g., date format conversions, unit standardization)

5.2.4 HITL Design Patterns for Statistical Programming (Anthropic, 2025) (Amazon Web Services, 2026)

In regulated industries, HITL takes the form of design patterns that blend efficiency with control:

- **Approval Checkpoints:** Humans sign off before high-impact actions (e.g., finalizing table specifications, committing generated code).
- **Exception Escalation:** Agents flag anomalies or ambiguities for human review rather than making assumptions.
- **LLM-as-a-Judge:** A separate LLM evaluates and validates agent outputs before they proceed downstream, providing scalable quality assessment (Maxim AI, 2025).
- **Audit Trails:** All agent decisions, tool invocations, and human interventions are logged in persistent artifacts for traceability and regulatory compliance.

These patterns ensure that agentic AI systems remain accountable and explainable—requirements that regulators increasingly demand (U.S. Food and Drug Administration, 2025). As Carter Cousineau, Vice President of Responsible AI at Thomson Reuters, has emphasized: “Human-in-the-Loop is critical at every stage: design, development, and deployment” (Merit Data & Technology, 2025).

5.2.5 Supervised Agentic AI in Regulatory Reporting: The DPS2R Workflow

DPS2R (Data Presentation Specifications to R) is an example of a Supervised Agentic AI that works in concert with Kiro CLI and was developed to generate draft R code for regulatory reporting grade **TLG (Table, Listing, Graph)** deliverables using the

¹¹ “Fully Autonomous” operations in clinical settings should include automated validation checks and audit logging. Risk classifications and approval requirements vary by organization.

`rtables` (Becker et al., 2025), `junco` (Becker et al., 2026), `tern` (Zhu et al., 2025), and `tidytlg` (Masel et al., 2025) packages.

Note: The DPS2R workflow described below reflects the current production version with 6 HITL checkpoints and a 7-phase architecture. The version evaluated in the Results section (Section 6) used an earlier 2-checkpoint design; the current architecture incorporates enhancements informed by those evaluation results.

Data Presentation Specifications (DPS) are mockshells augmented with prose describing some of the statistical programming algorithms to be used¹². However, even with this additional context, a gap remains: DPS defines **WHAT** the output should look like and **HOW** to compute it conceptually, while ADaM metadata defines **WHICH VARIABLES** are available. In reality, neither sufficiently specifies the granular detail needed to completely bridge the gap between technical specifications and R code (Figure 4). DPS2R addresses this gap through a seven-phase, state-machine workflow (freeCodeCamp, 2020) that builds intermediate specification artifacts before generating code (Figure 4). Each artifact captures the decisions and rationale at each phase: `layout-spec.md` documents the table structure; `data-spec.md` identifies required variables and filtering conditions; `plan.md` outlines the implementation approach; and `tasks.md` provides a step-by-step checklist. These artifacts make the process auditable & debuggable and allow the workflow to resume after interruptions. The construction of these intermediate specification artifacts is a hallmark of a **Spec-Driven Development (SDD)** approach.¹³ DPS2R focuses on three goals when applying SDD: (1) **Clarification**: Resolving ambiguities in existing specs; (2) **Elaboration**: Extracting actionable implementation details; (3) **Decoupling**: Separating the **“WHAT”** (specs) from the **“HOW”** (code).

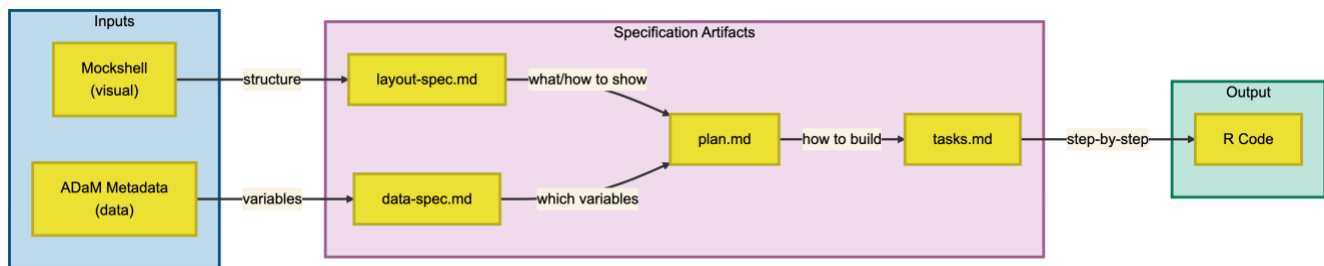


Figure 4: DPS2R bridges visual intent and variable availability through incremental specification artifact generation from the inputs to draft R code—where each artifact builds upon the previous one and decouples information from the source specifications.

During execution of DPS2R, Kiro CLI pauses at **6 key decision points** and waits for user feedback (see Figure 5):

1. After **mockshell understanding**
2. After **mockshell comparison** (Outsourced to the Large Language Model (LLM) `gpt-o4-mini`TM from OpenAI for the JJCS workflow only¹⁴)
3. After **metadata review**
4. After **variable analysis**
5. After **planning**

¹² Some statistical programming departments include detailed programming notes in their mockshells while others rely on institutional knowledge or separate specification documents.

¹³ A similar project, `Spec Kit` by GitHubTM, emerged around the same time as DPS2R and enables SDD across LLM agent platforms. While DPS2R’s core design predates Spec Kit, some concepts were later aligned—such as deeper integration with the user’s local filesystem and the constitution Markdown file `constitution.md` (though the content and intent differ). The `constitution.md` captures study-specific knowledge that helps guide the DPS2R workflow—including (a) **Study Design**: type, phases, treatment arms; (b) **Population Definitions**: flag variables and filtering criteria; (c) **Function Preferences**: preferred packages, functions to avoid; (d) **Variable Conventions**: treatment variables, key identifiers; (e) **Formatting Standards**: decimal places, date formats, missing value handling; (f) **Learnings**: study-specific insights discovered during development.

¹⁴ **JJCS** ≡ **J**ohnson & **J**ohnson **C**onected **S**tandards, a database of reference TLG outputs with associated R scripts. The `gpt-o4-mini` compares the mockshell in question with those in the JJCS database. The workflow automatically detects which type applies based on whether a reference TLG exists for the given output ID. If there is no reference example available, the comparison is skipped, and it relies on mockshell analysis only.

6. After code generation¹⁵

—ensuring that each subsequent workflow step does not commence without prior user confirmation that Kiro CLI can proceed. Pre-defined prompt commands assist the user to unambiguously navigate the workflow:

- `@clarify`: Pause for discussion or request minor adjustments; user input determines the next action.
- `@pass`: Confirm code is complete and correct; triggers DPS2R *memory* updates.¹⁶
- `@fail`: Indicate major issues; user specifies which phase to restart from.

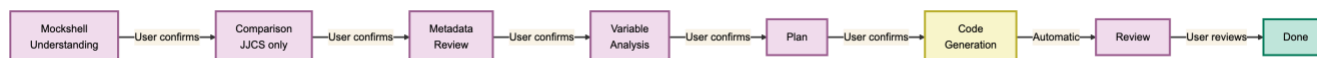


Figure 5: Human checkpoints throughout the DPS2R workflow

Workflow navigation is additionally tracked via an audit trail where all MCP tool calls are logged with **username + session ID**, **tool name + parameters**, and **timestamp + response summary** for traceability.

These 6 checkpoints illustrate the application of the AI bellwether **Skepticism, Curiosity, and Human-In-The-Loop (HITL) Validation** in DPS2R. The Kiro CLI is also exposed to a number of tools via **Model Context Protocol Servers** (Table 4) that connect it to external systems—enabling it to seek DPS2R workflow instructions to guide next steps, obtain study metadata from the DPS/APT systems¹⁷, locate relevant code via the **BM25 information retrieval algorithm**¹⁸, etc.:

Table 4: Some DPS2R MCP Server Tools

Tool	Purpose
<code>guidance_on_dps2r</code>	Get instructions for current workflow state
<code>retrieve_data</code>	Fetch study metadata from DPS/APT systems
<code>get_cached_data</code>	Retrieve cached session data (permits workflow resumption after interruptions and supports offline operation)
<code>compare_mockshell</code>	AI-powered mockshell comparison (via OpenAI's gpt-o4-mini LLM)
<code>analyze_variable_needs</code>	Identify required ADaM variables
<code>search_relevant_adam_variables</code>	Semantic search for variables
<code>find_relevant_code_chunks</code>	BM25-based code chunk retrieval

These MCP-server-supplied tools not only enable Kiro CLI to communicate with external systems but also inject it with **THE RIGHT CONTEXT AT THE RIGHT TIME** in the DPS2R workflow, preventing overwhelming Claude's context window.

¹⁵ Before user confirmation, each code section of the *proto*-draft R script is deterministically validated—undergoing a **Syntax Check**: R code parses without errors; **Function Attribution**: All functions are traceable to known packages; **Namespace Verification**: Functions exist in the specified R *Docker™ container*. If any validation step fails, Kiro CLI will automatically retry until escalation to the user is warranted.

¹⁶ The term “memory” is standard AI agent terminology referring to an AI system's ability to store and recall past experiences to improve decision-making (IBM, 2025).

¹⁷ The DPS Helper Tool and ADaM Planning Tool (APT) are internal J&J systems that statistical programmers use to author augmented mockshells as well as ADaM metadata + variable derivation algorithms, respectively.

¹⁸ **BM25** or **Okapi BM25** (Wikipedia, 2026d) is a keyword **best matching** (“BM”) algorithm that excels at locating query terms within documents, while semantic search finds conceptually similar content even with different wording.

Reusable “know-hows” for well-defined problems (“**skills**”) are additionally interjected as context to Kiro CLI from a skills library.¹⁹ Thus, the 3 AI bellwethers discussed in the Introduction ([Section 4](#))—**Skepticism, Curiosity, and Human-In-The-Loop (HITL) Validation; Context; and Model Context Protocol (MCP)**—have informed DPS2R design.

5.2.5.1 The DPS2R Workflow: Overview (Anthropic, 2025) (Amazon Web Services, 2026)

DPS2R generates R code for pharmaceutical TLG (Table, Listing, Graph) outputs through a guided, state-machine workflow ([Figure 6](#)). The system is comprised of **7 steps or phases**—grouped into **5 different purposes**:

1. Retrieves study metadata and mockshells from DPS/APT systems: **(i) Initialization**
2. Analyzes mockshell structure: **(ii) Understanding**
3. Compares with reference mockshells (JJCS workflow only): **(ii) Understanding**
4. Identifies required ADaM variables: **(iii) Analyze & Plan**
5. Creates an implementation plan: **(iii) Analyze & Plan**
6. Generates R code section by section: **(iv) Implementation**
7. Reviews and validates the output: **(v) Review**

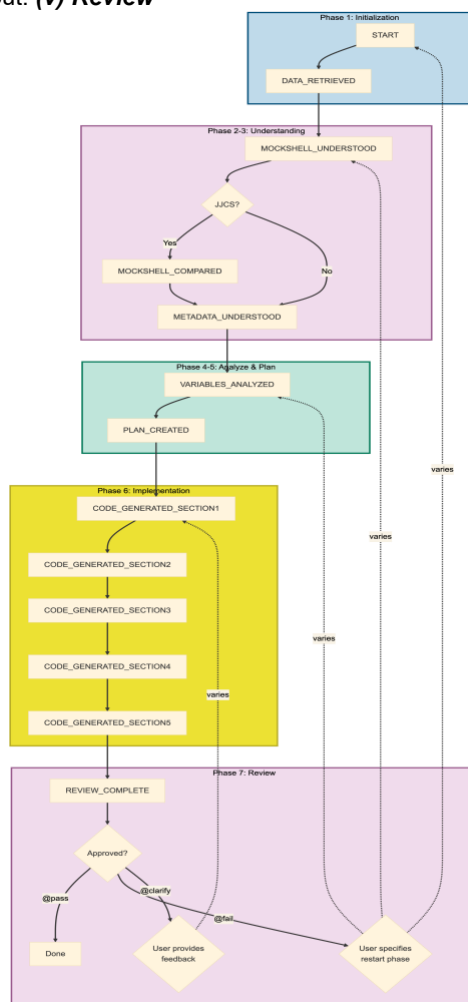


Figure 6: Complete DPS2R workflow state diagram

¹⁹ The skills concept follows Anthropic’s definition ([Anthropic, 2026b](#)): “Skills are folders of instructions, scripts, and resources that Claude loads dynamically to improve performance on specialized tasks.” Each skill consists of two parts: (1) a YAML header with metadata (name, description, category, tags), and (2) a “Solution” section describing how to resolve a specific problem ([Anthropic, 2026b](#)). DPS2R skills are organized by category: *planning, setup, data wrangling, table/listing/graph, exporting, quality control, and debugging*.

6. RESULTS: THE DPS2R WORKFLOW (Anthropic, 2025) (Amazon Web Services, 2026)

Note: The DPS2R version evaluated in this study used 2 HITL checkpoints. The current 6-checkpoint version described in the Methods section (Section 5) reflects subsequent enhancements informed by these results.

Nine experienced statistical programmers—each with substantial industry tenure—were selected from the ~90 colleagues who attended the Generative AI upskilling sessions described in the Methods section (Section 5). Each programmer was assigned multiple outputs (a combination of tables, listings, and graphs) across three therapeutic areas and 9 completed studies, using both JJCS and non-JJCS standards to reflect real-world workflows where both coexist. This resulted in **329** total output attempts across all programmers and iterations.

Each programmer followed the DPS2R workflow through successive iterations. A single **iteration** consisted of: (1) invoking the workflow and receiving generated code, at which point timing stopped; (2) running the code to verify execution; and (3) checking the output against the previously authored & validated output for correct statistics and layout. Subsequent iterations were intended to address any identified issues. **Accuracy** was measured on a 0-100% scale reflecting the degree to which the generated output matched the original output with respect to the displayed **statistics**. Accuracy was deemed unevaluable for attempts where code failed to generate evaluable output. While accuracy captures the percentage match between generated output and original output, **Total Score** incorporates process quality alongside output quality:

$$\text{Total Score} = 0.25 \times I(\text{Runs without Errors}) + 0.25 \times I(\text{Produced Output}) + 0.50 \times \frac{\text{Accuracy}}{100}$$

where $I(\cdot)$ is an indicator function returning **1** if the condition is met (i.e., Y) and **0** otherwise. This weighting scheme allocates **50%** to accuracy and **25%** each to two binary process indicators: **whether the generated code executed without errors** and **whether it produced any output at all**. Additionally, each programmer estimated the time required to complete the same output using traditional programming methods, drawing on their professional experience to provide realistic baseline comparisons. Results are summarized in Figure 7.

Figure 7 illustrates key metrics: **Median Accuracy** and **Median Total Score** by **Iteration** with interquartile range (bars and error bars) and **Mean Efficiency Gains Versus Traditional Programming Estimates** (x-axis labels).²⁰

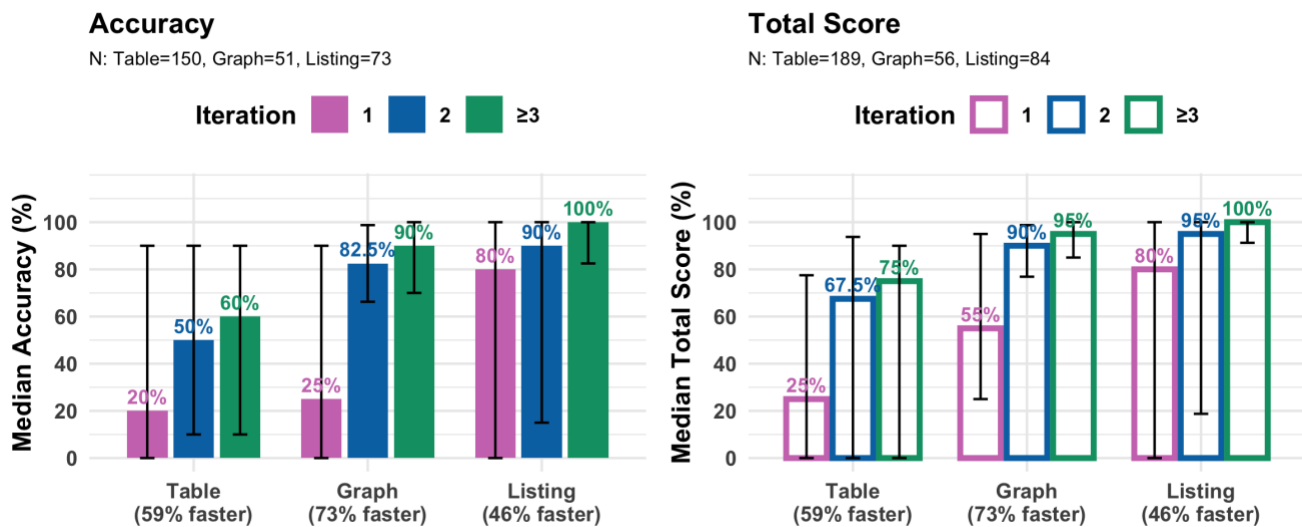


Figure 7: DPS2R performance improves with iteration. **Left:** Median Accuracy by output type. **Right:** Median Total Score by output type. Mean efficiency gains shown on x-axis compare DPS2R vs. estimated traditional programming time: Graphs: **73%** faster (0.8 vs 3.0 hours); Tables: **59%** faster (0.9 vs 2.2 hours); Listings: **46%** faster (0.7 vs 1.3 hours).

²⁰ Two salient points: (1) Sample sizes differ between **Accuracy (N=274)** and **Total Score (N=329)**: Accuracy was deemed unevaluable (i.e., missing) for 55 attempts where code failed to generate evaluable output. (2) Iterations were grouped as 1, 2, and ≥3 due to increasing sparsity of higher iteration counts. Analysis was limited to iterations 1–5, excluding 24 Table observations from iterations 6–10.

7. DISCUSSION

As per [Figure 7](#),

- **Accuracy improves with iteration across output type.**
- **Total Score improves with iteration across output type.**
- **DPS2R had the greatest success with generating listings, then graphs, and finally tables.**
- **Sizeable mean efficiency gains exist across output type**, specifically
 - Graphs: **73%** faster (0.8 vs 3.0 hours)
 - Tables: **59%** faster (0.9 vs 2.2 hours)
 - Listings: **46%** faster (0.7 vs 1.3 hours)

The challenges DPS2R and the 9 statistical programmers experienced generating viable tables relative to graphs and listings are logical—given the natural complexity of tables. This is also reflected by the fact that some Table data—specifically, 24 Table observations from iterations 6-10—were **excluded** from the analysis as a result of the sparsity of higher iteration counts—ensuring sufficient data density in iterations 1 through 5 for comparing Tables, Listings, and Graphs. The Supervised Agentic AI DPS2R workflow was deemed successful enough to move it into a scaling-up phase that includes installation in J&J’s regulatory reporting system known as **SPACE (Statistical Programming And Computing Environment)**. Kiro CLI is planned to be in SPACE by the beginning of **Q2-2026**, and teams are working to have DPS2R in SPACE by the beginning of **Q3-2026** or earlier.

7.1 Challenges, Limitations, and Opportunities

While the results are promising, **the efficiency gains are inherently subjective** (although informed by each evaluator’s seasoned work experience)—given that the 9 statistical programmers who performed the TLG iterations estimated the length of time needed to generate the same TLGs on their own. Regardless, DPS2R’s efficiency potential is very real—especially when one considers that (1) the current JJCS TLG library is far from sufficiently mature to address the myriad of different TLG mockshells needed to support 3 therapeutic areas, all 4 phases of clinical trials, and both safety and efficacy domains²¹; and (2) the use of `rtables`, `junco`, `tern`, and `tidytlg` for TLG generation is in its infancy in SPACE—severely limiting the amount of viable context for DPS2R.

The version of DPS2R that underwent evaluation had 2 HITL checkpoints, while the current version has 6. The decision to add these was made to help mitigate the challenges DPS2R experienced when generating Tables. As the JJCS TLG library develops and more studies begin using `R` for TLG generation, the size and heterogeneity of relevant context may become substantial enough where fewer HITL checkpoints are necessary to maintain draft TLG success—defined by little to no debugging or rework required in subsequent iterations.

The architecture of DPS2R has undergone substantive refactoring since formal TLG evaluation and is now a much more mature 7-phase workflow ([Figure 6](#)). Room for improvement remains, and work to improve Kiro CLI’s understanding of Table generation via `rtables` is underway.

7.2 The DPS2R Workflow and Existing Processes

DPS2R is Supervised Agentic AI that works in concert with Kiro CLI and generates draft R code for regulatory reporting grade TLGs. Upon receipt of the draft R code, statistical programmers are responsible for ensuring that it runs successfully and generates the correct output in SPACE’s development environment. This step constitutes the 6th and final HITL checkpoint of the DPS2R workflow and, if the draft R code correctly executes, marks the end of AI’s involvement in code generation. The existing process whereby self-checked code is promoted to the SPACE QA environment, undergoes risk-based validation, and, if it successfully passes, is then promoted to the SPACE production environment remains the same. Code that fails risk-based validation and is demoted to the SPACE development environment may undergo future DPS2R iterations until successful remediation of the identified issues. DPS2R and Kiro CLI remain fully focused on drafting code in the SPACE development environment—serving as the statistical programmer’s coding co-pilot. All existing processes remain unchanged.

7.3 AI Governance and Change Management

Kiro CLI and its various underlying Anthropic LLMs are ring-fenced within J&J—ensuring a secure and private enterprise experience without any data leakage to external AI systems. The sole purpose of Kiro CLI is to act as the statistical programmer’s / **THE DOMAIN EXPERT’S** coding co-pilot—facilitating code generation, review, and refactoring. This collaborative “**pilot**” / “**co-pilot**” relationship along with the 3 AI bellwethers—**Skepticism, Curiosity, and Human-In-The-**

²¹ It was decided to include both JJCS and non-JJCS standards to reflect the current real-world regulatory reporting practice in SPACE.

Loop (HITL) Validation, Context, and Model Context Protocol (MCP)—will be the focus as the previous Generative AI upskilling sessions are rolled out for all statistical programmers at J&J in 2026, helping them to responsibly leverage AI to augment their expertise in clinical data analysis.

7.4 Serendipity

R and the packages `rtables`, `junco`, `tern`, and `tidytlg` are relatively new for some statistical programmers at J&J. The DPS2R workflow empowers each statistical programmer to actively apply their **DOMAIN EXPERTISE** while interacting with Kiro CLI at each of the 6 HITL checkpoints. Carefully inspecting the resulting draft R code and how it handles both the creation of the tabular layout and the associated statistical analyses effectively acts as hands-on upskilling—an unexpected and powerful benefit of using the DPS2R workflow.

8. REFERENCES (*Anthropic, 2025*) (*Amazon Web Services, 2026*)

Allaire, J. *et al.* (2025) *quarto: R interface to 'quarto' markdown publishing system*. Available at: <https://CRAN.R-project.org/package=quarto>. Version 1.5.1, 2025-09-04. R interface to Quarto markdown publishing system.

Amazon Web Services (2026) “Kiro CLI: AI-powered development assistant.” Available at: <https://github.com/aws/amazon-q-developer-cli>. Version 1.24.0, 2026-01-15. Formerly Amazon Q Developer CLI. Runs directly in the terminal with conversational AI and development tools.

Anthropic (2024) “Introducing the Model Context Protocol.” Available at: <https://www.anthropic.com/news/model-context-protocol>. Published 2024-11-25.

Anthropic (2025) “Introducing Claude Opus 4.5.” Available at: <https://www.anthropic.com/news/claude-opus-4-5>. Published 2025-11-24. Anthropic’s flagship AI model for coding, agents, and computer use.

Anthropic (2026a) “Introducing Claude Opus 4.6.” Available at: <https://www.anthropic.com/news/claude-opus-4-6>. Published 2026-02-05. Upgrade to Opus 4.5 with improved coding, agentic tasks, 1M token context window, and state-of-the-art reasoning.

Anthropic (2026b) “Skills: Public repository for agent skills.” Available at: <https://github.com/anthropics/skills>. Skills are folders of instructions, scripts, and resources that Claude loads dynamically to improve performance on specialized tasks.

Becker, G. *et al.* (2025) *rtables: Reporting tables*. Available at: <https://CRAN.R-project.org/package=rtables>. R package version 0.6.15, 2025-12-14. R package providing a framework for declaring complex multi-level tabulations. Developed by F. Hoffmann-La Roche AG.

Becker, G. *et al.* (2026) *junco: Create common tables and listings used in clinical trials*. Available at: <https://CRAN.R-project.org/package=junco>. R package version 0.1.4, 2026-02-11. R package providing additional tooling for use alongside `rtables`, `rlistings`, and `tern` packages. Developed by Johnson & Johnson Innovative Medicine.

Brown, J. (2025) “18 motivational quotes on agentic AI.” Atera Blog. Available at: <https://www.atera.com/blog/agentic-ai-quotes/>.

Cheng, J. (2024) “Shiny x AI.” Available at: <https://www.youtube.com/watch?v=AP8BWGhCRZc>. Presented at posit::conf(2024): Shiny Talks, August 2024. YouTube video published 2024-10-31. R code available at GitHub: <https://github.com/jcheng5/r-sidebot>.

Django Stars (2025) “The future of MCP: Why the model context protocol is becoming the ‘USB-C for AI’.” Available at: <https://djangostars.com/blog/the-future-of-mcp-why-the-model-context-protocol-is-becoming-the-usb-c-for-ai/>. Discusses the N×M fragmentation problem that MCP solves.

freeCodeCamp (2020) “Finite state machine explained.” Available at: <https://www.freecodecamp.org/news/finite-state-machines/>. Describes FSM as a software design pattern where a model transitions between predefined behavioral states through external input.

Gohel, D. and Skintzos, P. (2025) *flextable: Functions for tabular reporting*. Available at: <https://CRAN.R-project.org/package=flextable>. R package version 0.9.10, 2025-08-24. R package for creating tables for Word, PowerPoint, and HTML output.

Goodwin, C. and Duranti, A. (1992) “Rethinking context: An introduction,” in C. Goodwin and A. Duranti (eds.) *Rethinking context: Language as an interactive phenomenon*. Cambridge: Cambridge University Press, pp. 1–42. Available at:

<http://www.sscnet.ucla.edu/anthro/faculty/duranti/reprints/rethco.pdf>. PDF of "Rethinking Context: An Introduction" sourced via Wikipedia references: <https://en.wikipedia.org/wiki/Context>.

Hand, C. (2025) "Clinicaltrials.gov-mcp-server: ClinicalTrials.gov Model Context Protocol server." Available at: <https://github.com/cyanheads/clinicaltrials.gov-mcp-server>. Version 1.5.0, 2025. Apache-2.0 License. Also available on npm: <https://www.npmjs.com/package/clinicaltrials.gov-mcp-server>. Author GitHub: <https://github.com/cyanheads>. Author website: <https://caseyjhand.com/>.

IBM (2025) "What is AI agent memory?" Available at: <https://www.ibm.com/think/topics/ai-agent-memory>. Defines AI agent memory as the ability to store and recall past experiences to improve decision-making.

LaTeX Project (2025) "LaTeX: A document preparation system." Available at: <https://www.latex-project.org/>. Free software for high-quality typesetting, especially mathematical formulas.

Masel, N. *et al.* (2025) *tidytlg: Create TLGs using the "tidyverse"*. Available at: <https://CRAN.R-project.org/package=tidytlg>. R package version 0.11.0, 2025-12-19. R package providing a framework for creating TLG outputs for clinical study reports. Developed by Johnson & Johnson Innovative Medicine.

Maxim AI (2025) "LLM as a judge: A practical, reliable path to evaluating AI systems at scale." Available at: <https://www.getmaxim.ai/articles/llm-as-a-judge-a-practical-reliable-path-to-evaluating-ai-systems-at-scale>. Describes using LLMs to evaluate other models or agents for validation.

Merit Data & Technology (2025) "Human-in-the-loop design for agentic AI in critical industries." Available at: <https://meritdata-tech.com/resources/human-in-the-loop-agentic-ai>. Discusses HITL design patterns including approval checkpoints, exception escalation, and autonomy levels.

Mermaid (2025) "Mermaid CLI: Command line interface for mermaid diagrams." Available at: <https://github.com/mermaid-js/mermaid-cli>. Version 11.12.0, 2025-09-25. Generates diagrams from Mermaid markdown-like syntax.

Microsoft Corporation (2025) "VS Code - Open Source (Code - OSS)." Available at: <https://github.com/microsoft/vscode>. The open source repository for Visual Studio Code.

Model Context Protocol (2025) "Architecture overview." Available at: <https://modelcontextprotocol.io/docs/learn/architecture>. Official MCP documentation.

Oracle (2025) "What is agentic AI?" Available at: <https://www.oracle.com/artificial-intelligence/agentic-ai/>. Defines agentic AI as systems capable of making autonomous decisions on how to achieve a goal.

Pedersen, T.L. (2025) *patchwork: The composer of plots*. Available at: <https://CRAN.R-project.org/package=patchwork>. R package version 1.3.2, 2025-08-25. R package for combining multiple ggplot2 plots into a single figure.

Purdy, M. (2024) "What is agentic AI, and how will it change work?" *Harvard Business Review* [Preprint]. Available at: <https://hbr.org/2024/12/what-is-agentic-ai-and-how-will-it-change-work>. Mark Purdy is co-founder and director of Beacon Thought Leadership.

R Core Team (2025) *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Available at: <https://www.R-project.org/>. Version 4.5.2, 2025-10-31.

Sueiras, R. (2025) "Manage context rot by exploring new experimental features in Amazon Q CLI." Available at: <https://dev.to/aws/manage-context-rot-by-exploring-new-experimental-features-in-amazon-q-cli-10ki>. DEV Community (dev.to).

U.S. Food and Drug Administration (2025) "Considerations for the use of artificial intelligence to support regulatory decision-making for drug and biological products." Available at: <https://www.fda.gov/regulatory-information/search-fda-guidance-documents/considerations-use-artificial-intelligence-support-regulatory-decision-making-drug-and-biological>. FDA draft guidance on AI credibility assessment framework.

Wickham, H. *et al.* (2023) *dplyr: A grammar of data manipulation*. Available at: <https://CRAN.R-project.org/package=dplyr>. R package version 1.1.4, 2023-11-17. R package for data manipulation.

Wickham, H. *et al.* (2025) *ggplot2: Create elegant data visualisations using the grammar of graphics*. Available at: <https://CRAN.R-project.org/package=ggplot2>. R package version 4.0.1, 2025-11-14. R package implementing the grammar of graphics.

Wikipedia (2026a) "Abstract syntax tree." Available at: https://en.wikipedia.org/wiki/Abstract_syntax_tree. Tree representation of the abstract syntactic structure of source code.

Wikipedia (2026b) "Command-line interface." Available at: https://en.wikipedia.org/wiki/Command-line_interface. Text-based user interface for interacting with computer programs.

Wikipedia (2026c) "Language server protocol." Available at: https://en.wikipedia.org/wiki/Language_Server_Protocol. Open protocol for communication between source code editors and language servers.

Wikipedia (2026d) "Okapi BM25." Available at: https://en.wikipedia.org/wiki/Okapi_BM25.

Wikipedia (2026e) "Tree-sitter (parser generator)." Available at: [https://en.wikipedia.org/wiki/Tree-sitter_\(parser_generator\)](https://en.wikipedia.org/wiki/Tree-sitter_(parser_generator)). Incremental parsing library for programming tools.

Wiktionary (2026a) "Siren song." Available at: https://en.wiktionary.org/wiki/siren_song. An enticing but dangerous appeal; from Greek mythology where Sirens lured sailors to destruction with their singing.

Wiktionary (2026b) "Spidey sense." Available at: https://en.wiktionary.org/wiki/Spidey_sense. An intuitive feeling, usually of something being wrong or suspicious; from the comic book superhero Spider-Man.

Yu, G. (2026) *ggtangle: Draw network with data*. Available at: <https://CRAN.R-project.org/package=ggtangle>. R package version 0.1.1, 2026-01-16. R package for network visualization extending ggplot2.

Zamanian, K. (2024) "A comparative study of AI agent orchestration frameworks." Available at: <https://medium.com/@kzamania/a-comparative-study-of-ai-agent-orchestration-frameworks-f61cd49b687e>. Compares multi-agent orchestration frameworks including delegation patterns where main agents coordinate sub-agents. Also available on ResearchGate: <https://www.researchgate.net/publication/386083531>.

Zhu, J. *et al.* (2025) *tern: Create common TLGs used in clinical trials*. Available at: <https://CRAN.R-project.org/package=tern>. R package version 0.9.10, 2025-12-18. R package containing analysis functions to create tables and graphs used for clinical trial reporting. Developed by F. Hoffmann-La Roche AG.

9. SOFTWARE DISCLOSURES

1. This paper was produced using **Quarto** (Allaire *et al.*, 2025).
2. The in-text diagrams were created with **Mermaid CLI** (Mermaid, 2025).
3. All in-text tables were generated using the R package **flectable** (Gohel and Skintzos, 2025).
4. **R** (R Core Team, 2025), **dplyr** (Wickham *et al.*, 2023), **ggplot2** (Wickham *et al.*, 2025), and **patchwork** (Pedersen, 2025) were used to generate the in-text accuracy & total score plots.

10. ACKNOWLEDGEMENTS

In recognition of the 9 statistical programmers who conducted the DPS2R testing: **Manjusha Alapati, Yufan Chen, Yu (Emily) Cheng, Abhyuday Dwivedi, Steven Haesendonckx, Jiaqi Song, Chengeng Tian, Kishore Vengala, Yingshan You**. *The success of DPS2R would NOT have been possible without you! Thank You!* 😊 Sincere gratitude also goes to **Marc Drucker** and **Tadeusz Lewandowski** for devising the scoring algorithm and their diligent efforts to realize its implementation. *Thank You!* 😊

11. CONTACT INFO

Your comments and questions are valued and encouraged. Contact the authors at:

Mark Rothe Johnson & Johnson 920 U.S. Route 202 South Raritan, New Jersey United States 08869 mrothe@its.jnj.com	Sumesh Kalappurakal Johnson & Johnson 920 U.S. Route 202 South Raritan, New Jersey United States 08869 skalappu@its.jnj.com	刘晓畅 Xiaochang (Jack) Liu Johnson & Johnson Building A, Xinyan Mansion No. 65, Guiqing Rd., Xuhui Shanghai 200233 Shanghai China xliu328@its.jnj.com
--	---	--

Brand and product names are trademarks of their respective companies.