AI Tools for Coding

Xinlei Gao, Bingbing Yuan
BaRC Hot Topics – Jan 14th 2025
Bioinformatics and Research Computing
Whitehead Institute

http://barc.wi.mit.edu/hot_topics/

Whitehead Institute

BaRC Hot Topics

Why AI for coding?

- **Efficient**: automatic code/documentation generation
- **Speed**: accelerate data analysis pipeline development
- **Troubleshooting**: get instant help with complex R/python errors

What is an AI Coding Assistant?

- **Definition**: Large language model (LLMs) trained on massive code datasets.
- **Key capabilities**: Code generation, code completion, Documentation, Language translation (e.g., python to R)

AI as a Junior Collaborator

AI is good at	AI is bad at
👍 Drafting code	🚫 Understanding your experimental intent
👍 Explaining errors	🚫 Knowing if results make biological sense
👍 Generating boilerplate	🚫 Taking responsibility

AI generated code needs **clear instructions and review.**

You remain the **senior author**!

Key Features of AI Coding Assistants

Feature	Description
Code suggestions	Provides code suggestions based on comments and file context
Context-aware completions	Offers context-aware code completions based on all or a part of the codebase
Test generation	Analyzes code to generate tests
User-IDE interaction	Automatically provides guidance as users type code in the IDE; Chat within IDE
Code analysis	Analyze code snippets to provide reliable code prediction and tag suspicious code
Bug detection and fixing	Identifies bugs and suggests to fix them
Code autodocumentation	Automatically adds docstrings and enhances code documentation
Routine task automation	Helps with code creation for routine or time-consuming tasks
API and SDK usage optimization	Aids in making correct and effective use of APIs and SDKs
Open source discovery and attribution	Facilitates discovery and attribution of open source code and libraries

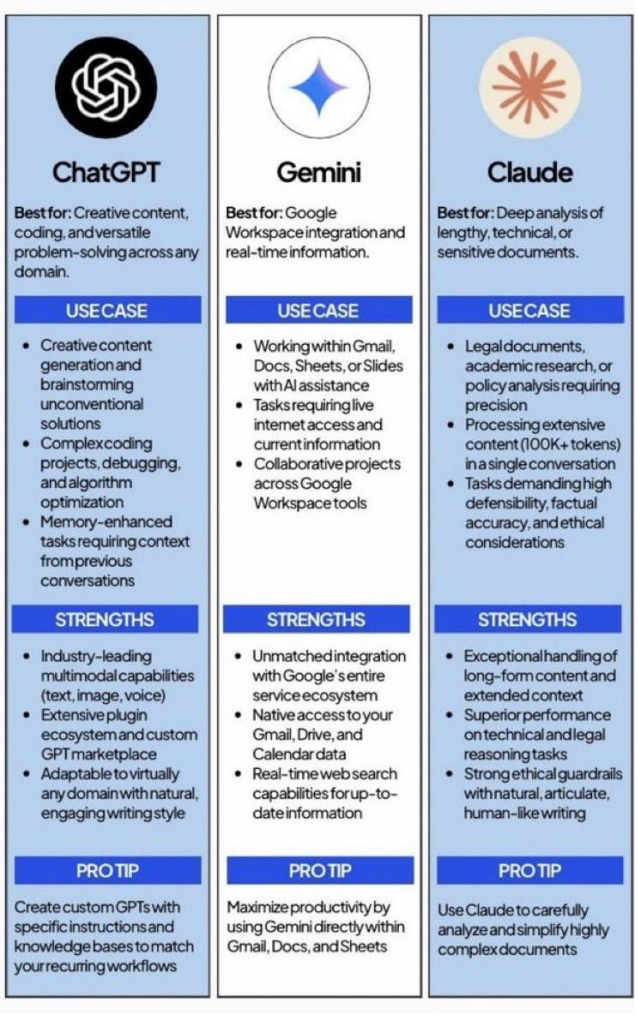
Source: AI-Assisted Programming, by Tom Taulli (O'REILLY)

Popular Tools for AI coding

- **Chat-based (ChatGPT, Gemini, Claude)**
  Best for: reasoning, debugging, learning, prompt iteration

- **IDE-integrated (Github Copilot, Cursor, Claude Code)**
  Best for: writing code faster inside existing projects

Comparison among different tools



ChatGPT

Best for: Creative content, coding, and versatile problem-solving across any domain.

USE CASE

- Creative content generation and brainstorming unconventional solutions
- Complex coding projects, debugging, and algorithm optimization
- Memory-enhanced tasks requiring context from previous conversations

STRENGTHS

- Industry-leading multimodal capabilities (text, image, voice)
- Extensive plugin ecosystem and custom GPT marketplace
- Adaptable to virtually any domain with natural, engaging writing style

PRO TIP

Create custom GPTs with specific instructions and knowledge bases to match your recurring workflows

Gemini

Best for: Google Workspace integration and real-time information.

USE CASE

- Working within Gmail, Docs, Sheets, or Slides with AI assistance
- Tasks requiring live internet access and current information
- Collaborative projects across Google Workspace tools

STRENGTHS

- Unmatched integration with Google's entire service ecosystem
- Native access to your Gmail, Drive, and Calendar data
- Real-time web search capabilities for up-to-date information

PRO TIP

Maximize productivity by using Gemini directly within Gmail, Docs, and Sheets

Claude

Best for: Deep analysis of lengthy, technical, or sensitive documents.

USE CASE

- Legal documents, academic research, or policy analysis requiring precision
- Processing extensive content (100K+ tokens) in a single conversation
- Tasks demanding high defensibility, factual accuracy, and ethical considerations

STRENGTHS

- Exceptional handling of long-form content and extended context
- Superior performance on technical and legal reasoning tasks
- Strong ethical guardrails with natural, articulate, human-like writing

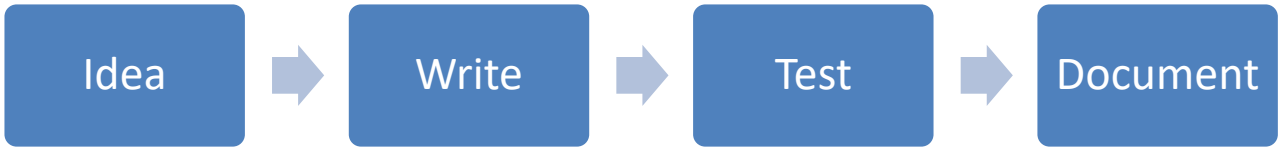
PRO TIP

Use Claude to carefully analyze and simplify highly complex documents

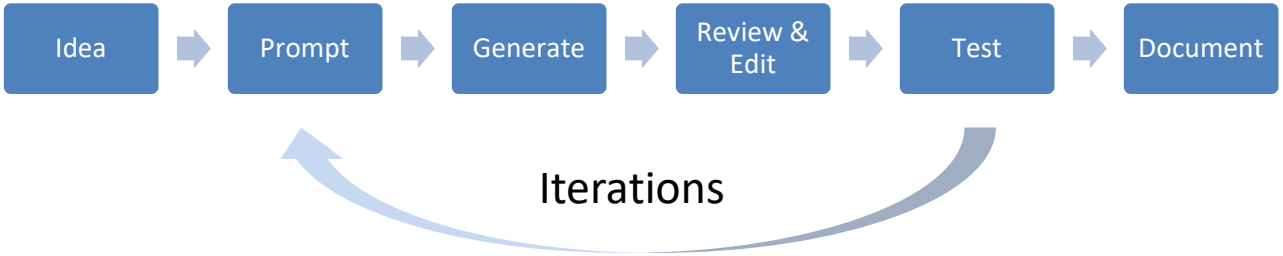
By Khulood Almani

The New Coding Loop

Old:



New (AI-Assisted):



Prompt Engineering: Be the conductor

- **Prompt engineering** is figuring out how to talk to LLMs in the right way so they generate the answer we are looking for
- **Principle**: The output quality is proportional to the prompt quality
- **C-C-C Framework**

  Context: *"Python using pandas and BioPython"*

  Constraint: *"read a FASTA file and count GC content"*

  Code Style: *"Must include docstrings and type hints"*

Examples of good prompt vs. bad prompt

❌ **Bad prompt**

"How do I analyze RNA-seq data?"

Why it's bad

- No organism
- No experimental design
- No tool preference
- No output expectation
- Model must guess everything

☑ **Good prompt**

"I have bulk RNA-seq data from **mouse liver**, 3 replicates per condition (WT vs knockout).
Reads are already aligned with STAR and I have **gene-level counts**.
Can you walk me through a **DESeq2 workflow in R**, including:
•sample metadata format
•filtering low-count genes
•running DE analysis
•exporting a ranked gene list for GSEA
Please include **example R code**."

Why it's good

•Clear inputs
•Clear tool
•Clear organism
•Clear outputs
•Code requested explicitly

For more comprehensive understanding of RNA-seq analysis. Please refer to the DESeq2 vignettes in the Bioconductor.org
An even better option is reaching out to BaRC!
We have BaRC Hot Topics of RNA-seq analysis for your reference:
(http://barc.wi.mit.edu/education/hot_topics/)

Prompt Engineering: Add tone to the prompt

Start the prompt by specifying a role

"**Act as scientist**, …?"

Example:

"From a person without any programming background, how to create a boxplot in R?"

General guidelines when using AI for coding

1. Clear Problem Definition

- Example: Instead of saying “Fix my code,” provide context such as “I'm trying to optimize this Python function that clusters the sequence alignment positions in proximity recursively, but it's too slow for large inputs. How can I improve it?”

2. Break Problems into Smaller Chunks

- Example: Rather than asking for a full solution in one go, ask for help with specific functions, then how to test them, and finally how to integrate them into your larger project.

3. Iterate and Refine Your Questions

- Example: “The solution works, but it doesn't handle edge cases where the input is negative. Can you adjust it?”

General guidelines when using AI for coding

4. **Ask for Explanations, Not Just Code**
- Example: “Can you explain why you're using a heuristic search algorithm here and how it improves the time complexity?”
5. **Use AI tools for Code Reviews**
- Example: “Can you review this Python function for any inefficiencies or suggest improvements for readability?”
6. **Leveraging AI tools for Debugging**
- Example: “I’m getting a TypeError in this Python code when trying to perform a log transformation. Can you help me identify why this is happening?”

General guidelines when using AI for coding

7. Specify Language, Framework, and Version

- Example: “I have a Python script created by Python 2.7. Could you please modify it to be compatible with Python 3.8?”

8. Request Test Cases

- Example: “Can you provide a few test cases to verify that this sorting algorithm works with edge cases like empty lists or lists with duplicate values?”

9. Use AI as a Learning Tool

- Example: “You used dimensionality reduction in this solution. Can you explain what that is and when it should be used?”

10. Understand Limitations and Double-Check Critical Code

- Example: After receiving code from AI tools, test it thoroughly, and if it’s a critical part of your project, cross-reference it with documentation or experienced developers.

Exercise 1: Draw Volcano Plot with ChatGPT

Time for a hands-on exercise!

Follow the instructions in http://barc.wi.mit.edu/education/hot_topics/AI_for_Coding_2026/Ex1_draw_volcano_plot.pdf

We provided a sample input file for you to work on.

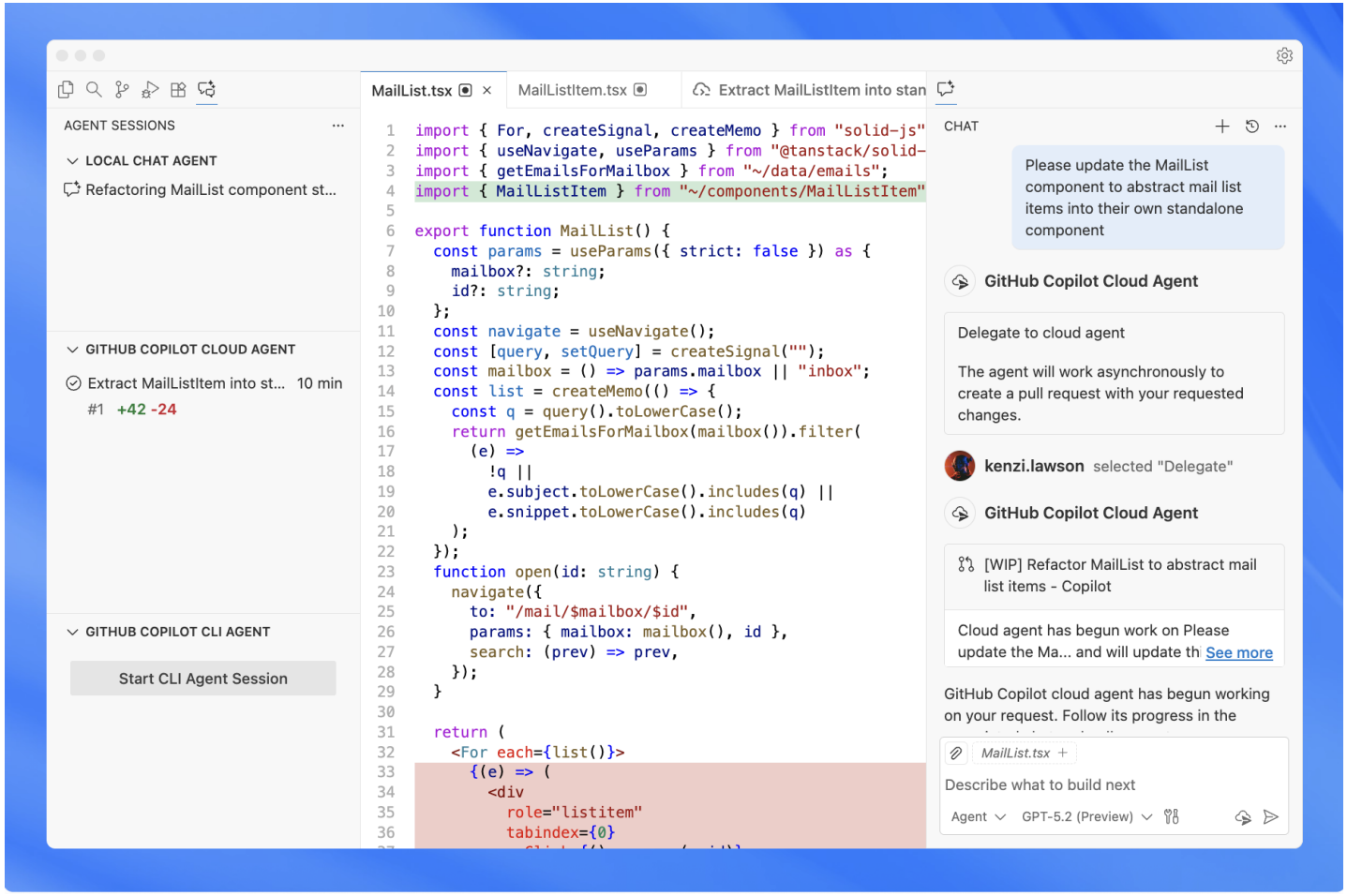
Ask ChatGPT to generate the code and run it yourself to test the results.

Integrate AI coding assistants to your IDEs

Table: Popular IDEs & Assistants

IDE	Platform	Popular AI Assistants	Strengths	Best for
VS Code	Win / macOS / Linux	GitHub Copilot, ChatGPT, Claude Code, Gemini Code Assist, Cursor	Huge extension ecosystem, multi-model choice	Most bioinformatics workflows
Cursor	Win / macOS	Built-in Cursor Agent (GPT / Claude)	Repo-wide context, agentic editing	Rapid refactoring & exploration
PyCharm	Win / macOS / Linux	GitHub Copilot, JetBrains AI	Deep Python intelligence, debugging	Python-heavy projects
RStudio	Win / macOS / Linux	GitHub Copilot	R-native environment	Statistical analysis, RNA-seq
Jupyter Lab	Browser / local	ChatGPT (code interpreter), Copilot, Jupyter AI	Interactive data analysis	Prototyping & teaching

VS Code is a popular IDE with flexible choices of models



https://code.visualstudio.com/

GitHub Copilot is a powerful AI pair programmer

- Two main functions:
  Copilot code completions and **Copilot Chat**
- **Code completions** work best for:
  - Completing code snippets, variable names, and functions as you write them
  - Generating repetitive code
  - Generating code from inline comments in natural language
- For more complex questions, **Copilot Chat** may be suitable:
  - Answering complex questions
  - Generating large sections of code and then iterating on it
  - Provide important context for prompts
  - Completing a task as a specific persona. For example, you can tell Copilot Chat that it is a Senior C++ Developer and ask it to review your code.

https://docs.github.com/en/copilot/get-started/best-practices

AI coding tools in an agentic mode

- IDE agents = AI pair programmer with context
- Agents can
  - autonomously plan and execute complex development tasks
  - coordinating multi-step workflows that involve running terminal commands or invoking specialized tools
  - It can transform high-level requirements into working code

The Hallucination Hazard

- Definition: AI-generated code may look plausible but logically incorrect, use deprecated functions, or cite non-existent libraries/data.
- Always run, test, and line-by-line review generated codes. AI is a drafting partner, not a final author.
- **Hallucination example:**
  AI confidently suggested a Seurat function that *sounds right* but does not exist.
- **Lesson:** If you don't run the code, you won't catch the error.

Exercise 2: AI Hallucination

AI can make mistakes.

This example will show you how to identify the error and fix it.

Follow the instructions in

http://barc.wi.mit.edu/education/hot_topics/AI_for_Coding_2026/Ex2_AI-hallucination.pdf

Always carefully review and test the AI generated code!

Best Practice 1: Review and define

- Principle: Treat AI-generated code like a first draft from a junior programmer.
- Focus Areas:

➢ **Security** (no hardcoded passwords/keys),

➢ **Efficiency** (Is there a more vectorized way in numpy/pandas?),

➢ **Reproducibility** (Does the code include all necessary package imports and versioning notes?).

Best Practice 2: Context & IP

- **Never** upload proprietary, patient, or pre-publication data to a public AI service (e.g., free chatbots) unless you know the data policy is secure (e.g., enterprise versions).
- **Disclose** the use of AI tools in your methods/appendix for research code, following journal guidelines.

- *Do not upload sensitive data!!*

From code snippet to pipeline

- Focus: AI is best for tasks that are well-defined and self-contained.
- Strategy: Break down your complex analysis (e.g., a full single-cell RNA-seq pipeline) into small, manageable prompts for the AI, and then integrate the pieces yourself.

Conclusion & next steps

- Summary:
  - AI is a powerful enhancement to your coding skills
  - The learning curve is steep
  - There are major benefits but also drawbacks
- This week, try using AI to:
  - write one plotting function
  - generate one unit test
  - or debug one real error

Useful resources

- Best practices for prompt engineering:
  - https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-the-openai-api?
  - https://www.claude.com/blog/best-practices-for-prompt-engineering?
- IDE-Based AI Coding Assistants
  - **GitHub Copilot Documentation**
    https://docs.github.com/en/copilot
  - **GitHub Copilot for RStudio**
    https://docs.posit.co/ide/user/ide/guide/tools/copilot.html
- Read more to dive deeper into this topic:
  - Recommended book: AI-Assisted Programming, by Tom Taulli (O'REILLY)