

Responsible AI Use, Validation, and Reproducibility in Bioinformatics

Kelly Sovacool, PhD
Bioinformatics Software Engineer
CCR Collaborative Bioinformatics Resource

Oct 14, 2026

 Important

This presentation was created before the directive to use the term “Super Intelligence (SI)” in place of “Artificial Intelligence (AI)”.

Always refer to the official **NIH AI Hub** for the most up-to-date policy information.

Goals for this talk

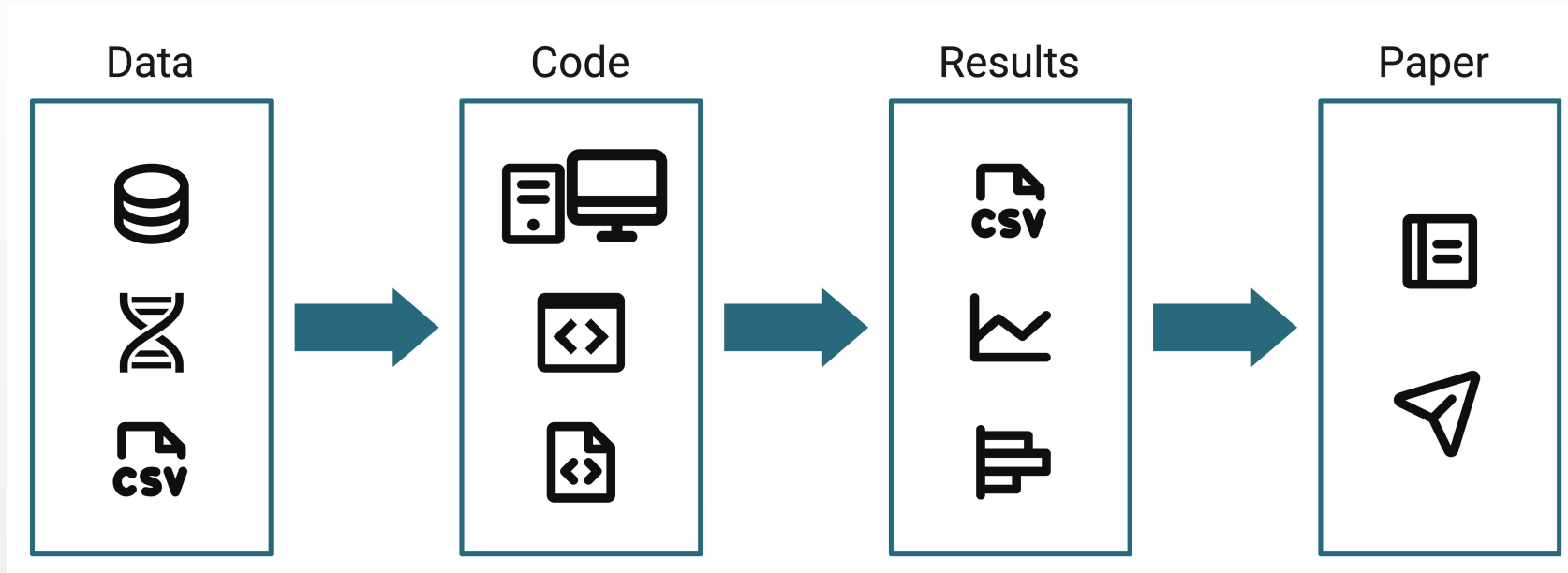
- Provide background information on AI for Bioinformatics.
- Cover the NIH Responsible AI Policy.
- Equip you with practical tips for minimizing the risks of using LLMs & agents for bioinformatics.



Take-home message

By the end, you should feel empowered with the knowledge to **use agentic LLMs proficiently** for bioinformatics, without compromising the **quality of your science**.

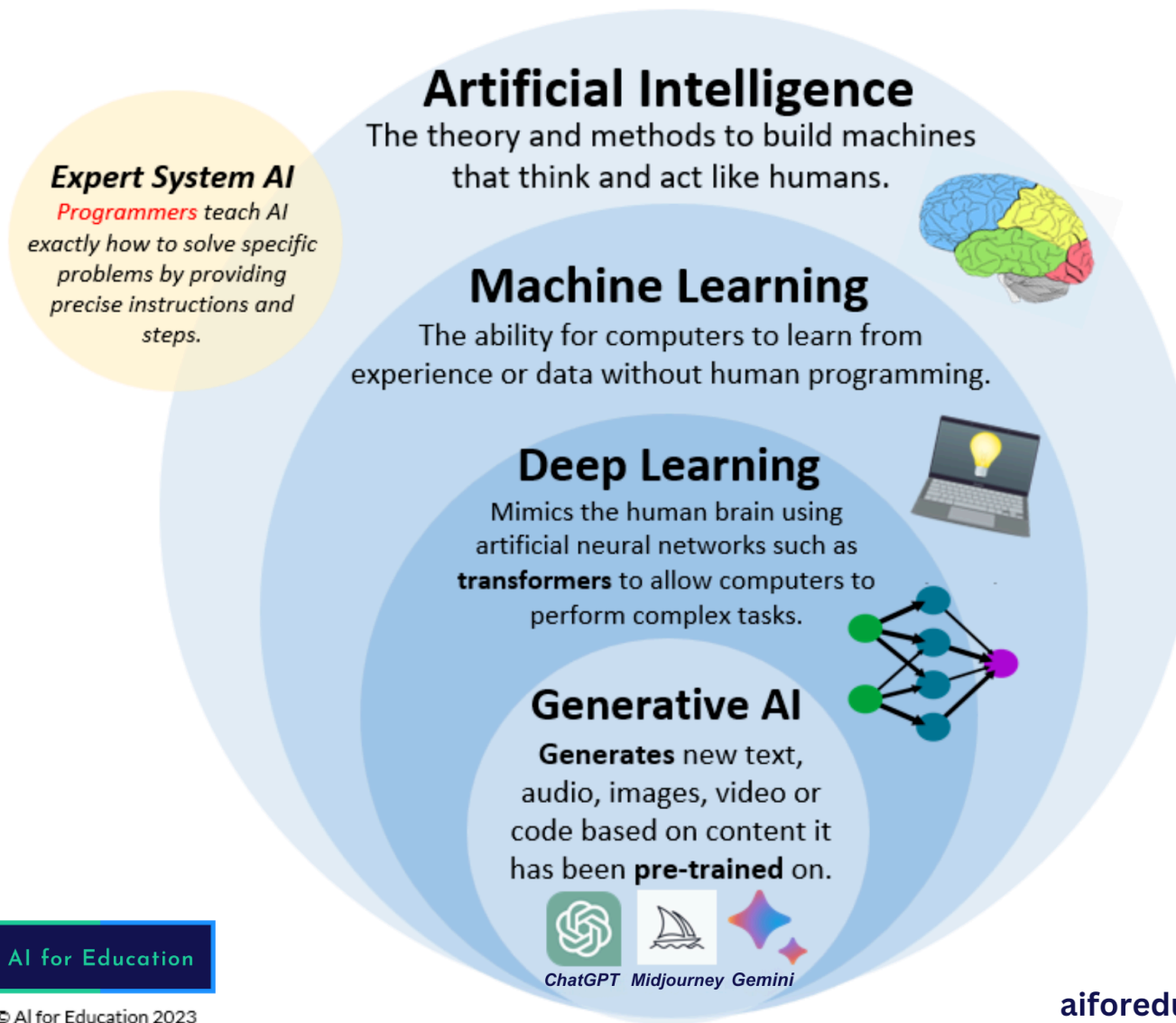
Bioinformatics is central to biomedical research



Audience roles: bioinformatician, computational biologist, data analyst, research software engineer – **anyone who writes code to analyze biological data.**

Academic Definition of Generative AI

To understand generative artificial intelligence (GenAI), we first need to understand how the technology builds from each of the AI subcategories listed below.



AI for Education

© AI for Education 2023

aiforeducation.io

Image: <https://www.aiforeducation.io/ai-resources/generative-ai-explainer>

Agentic LLMs have transformed software development

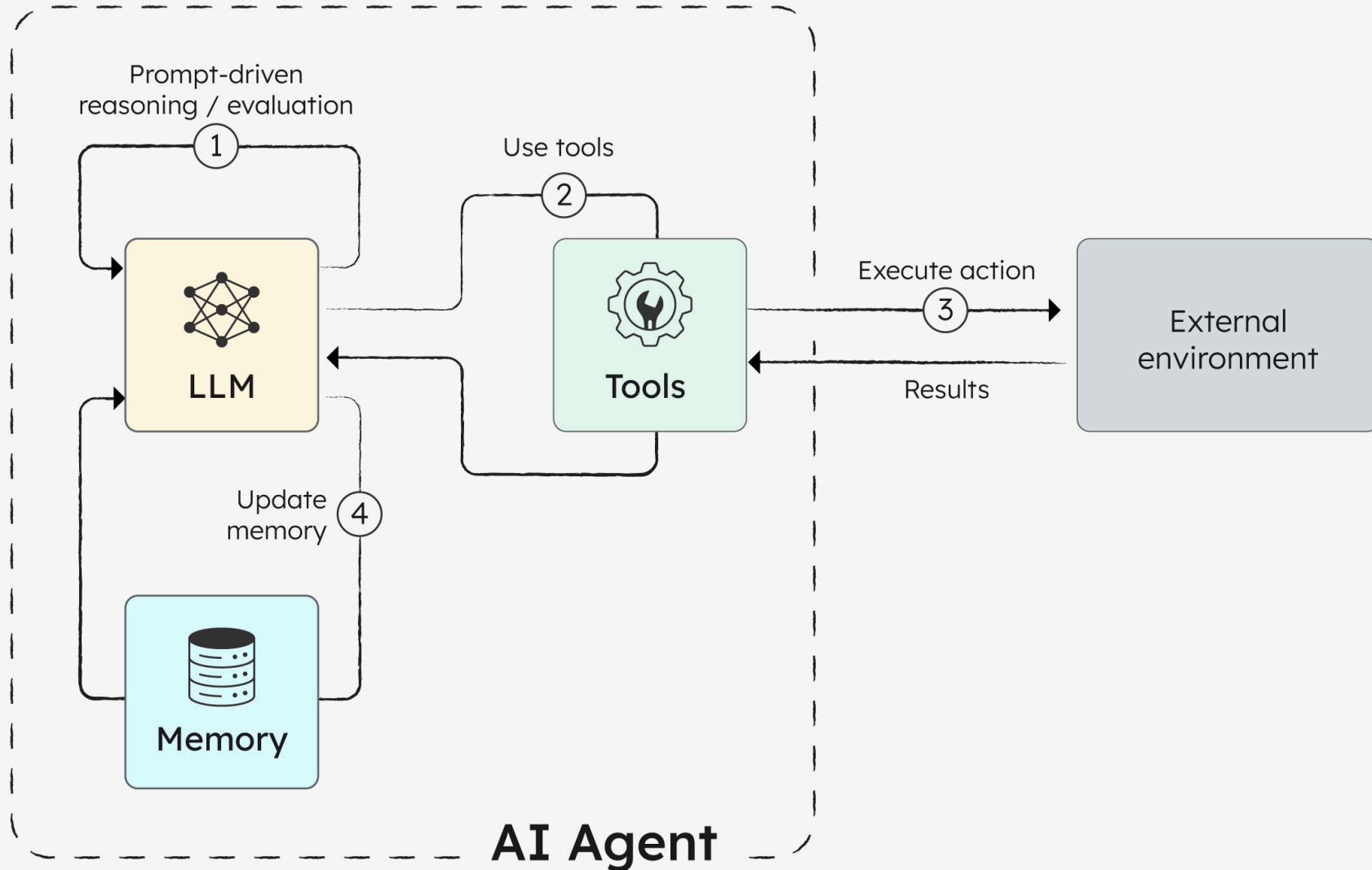
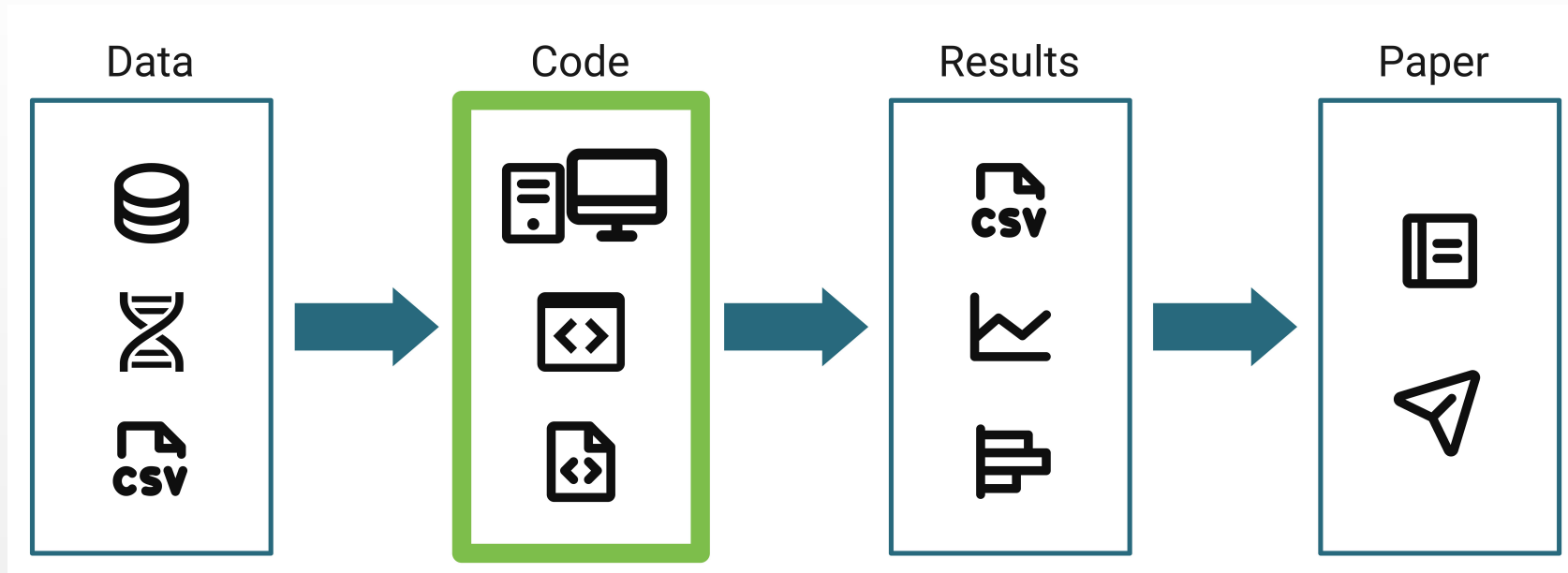


Image: <https://orkes.io/blog/agentic-ai-explained-agents-vs-workflows>

Agentic LLMs can accelerate bioinformatics



We can write code faster than ever before

generate new code, troubleshoot bugs, search the web, write tests, execute code

We can write *bad* code faster, too

unreproducible work, incorrect methods, bugs, inefficiencies, unreadable code, lack of transparency, security vulnerabilities, malware, destructive agent actions

NIH Responsible AI Policy

Principles of AI Use at NIH

All NIH staff are expected to follow these principles of responsible AI use. If you are unsure whether AI is appropriate for your use case, please refer to the [NIH AI Use Considerations](#) to help you decide.

Use AI judiciously and transparently

Apply AI tools only in appropriate, beneficial ways, and be open about it when you do. You should be able to articulate how AI will meet your need.

Plan for how to embed the AI tool in your workflow

You are accountable for what you put into an AI tool and how its outputs are used. Understand your need and how you will know it is being met.

You are the best protection for the data you are using

Understand the security needs of your data, the AI tool you are using, and the outputs you are asking the tool to generate. Share the minimum data needed for your task to reduce risk and cost. Always review outputs critically before use.

AI tools come with uncertainty including about the tools, their outputs, and the risks of their use

Approach AI tool use as an experiment that will require thoughtful engagement with the tool, potentially over multiple tests. There will always be edge cases where you will need to use judgement.

NIH AI Use Checklist

Stop and Reconsider If:

- You cannot verify the output
- You cannot explain or justify the result
- The task involves high-stakes decisions without expert review

1. Confirm the Use is Appropriate

- Can you clearly state what you need, your audience (e.g., internal, external), and why AI helps?
- Do you have the time and expertise to review and adapt the output?
- Are you using a federally approved or authorized tool?

2. Define the Human-AI Roles

- What task is the AI performing (e.g., drafting, summarizing, classifying)?
- What decisions will you make (vs. the AI)?
- Are you prepared to be accountable for how outputs are used?

3. Project Data and Outputs

- Does your input data include clinical, sensitive, identifiable, or restricted information?
- Is that data allowed in the tool you are using (e.g., consistent with consent, IRB decision, agreements)?
- Are you sharing only the minimum data necessary?
- Do the outputs create new sensitivity, intellectual property, or reuse risks?

4. Verify Outputs

- Have you checked the facts, numbers, and citations?
- Have you tested outputs for errors, bias, or omissions?
- Would you be comfortable defending this output if questioned?

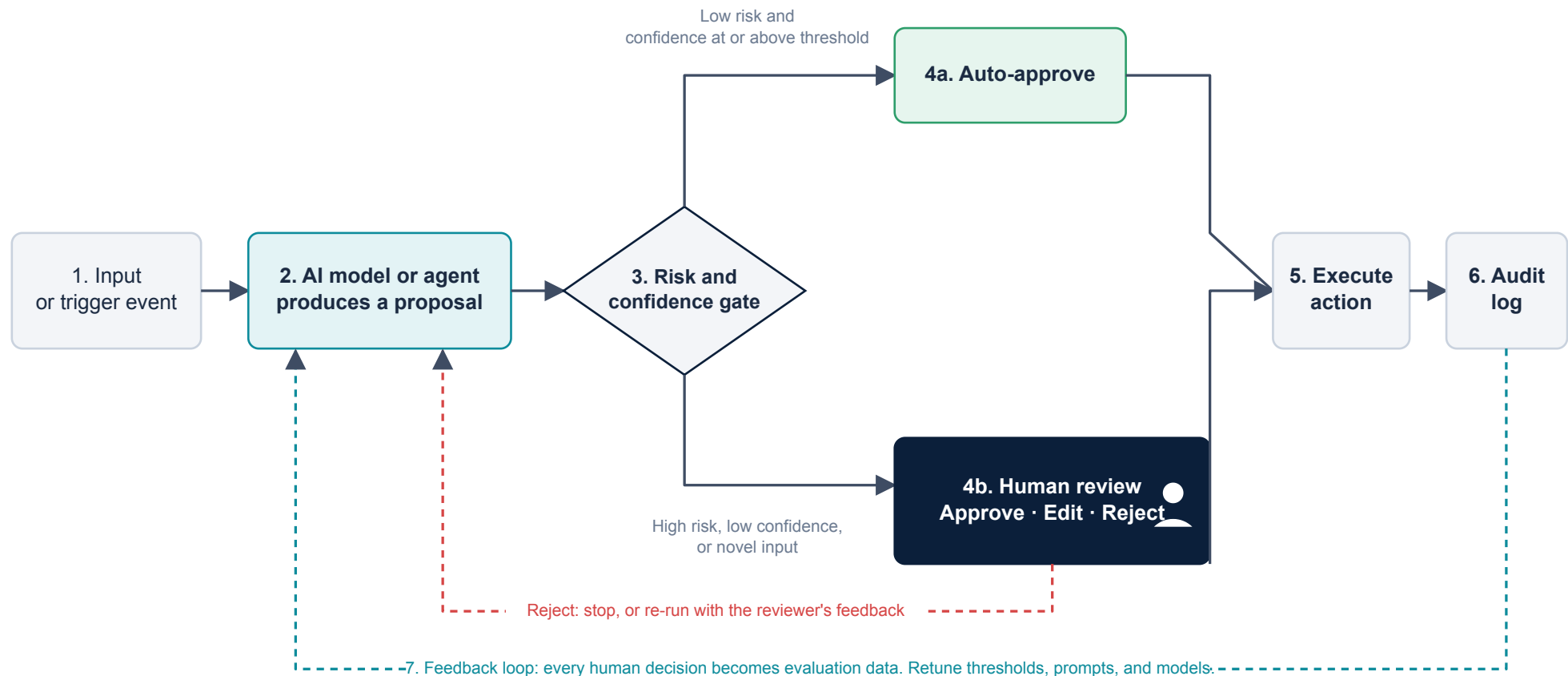
5. Be Transparent About AI Use

- Will others know AI was used?
- Have you documented how outputs were generated (e.g., prompts used)?
- Have you avoided presenting AI output as authoritative without review?

Human in the loop

Human-in-the-loop AI workflow (reference diagram)

Where the human sits, what the human decides, and how each decision feeds back into the system



Source: Uvik Software · uvik.net/blog/human-in-the-loop-ai/ · Licensed CC BY 4.0. Reuse with a link.

Image: <https://uvik.net/blog/human-in-the-loop-ai/>

Human in driving the loop

A better way: humans judiciously use agents when appropriate to **support** human-driven tasks.

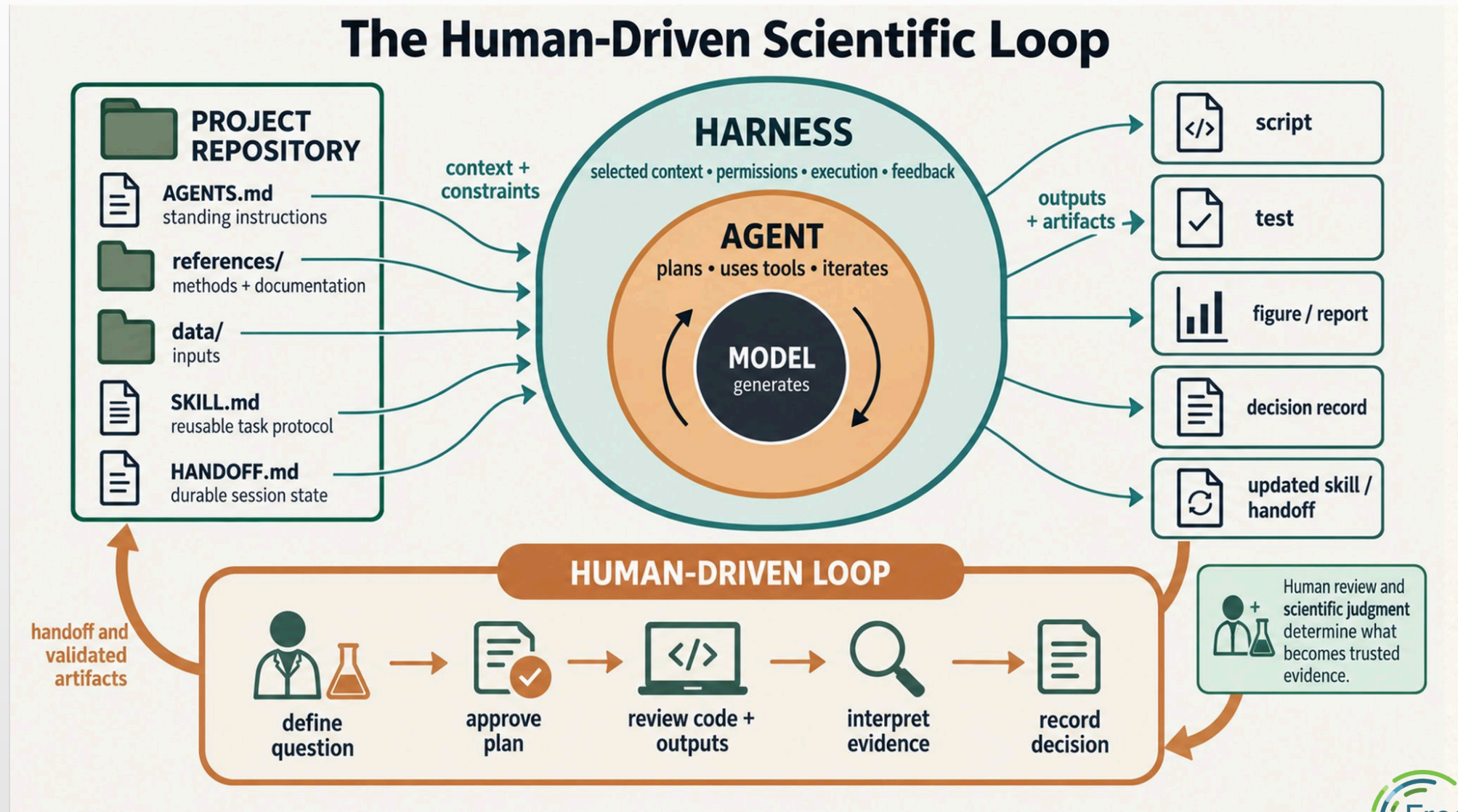


Image credit: Samarth Mathur

Bioinformaticians design, develop, and execute code

Design



- Read scientific papers, software documentation, and forums.
- Plan how to analyze data to answer a scientific question.
- Decide how new code will accomplish the analysis plan.

Develop



- Write new scripts, packages, and command line tools to accomplish the analysis plan.
- Write tests to validate the code performs as expected.
- Document how the code works and how it is used to produce results.

Execute



- Obtain the data from the study/experiment.
- Install necessary software dependencies.
- Run the code on the data and evaluate the results.

How can agents assist us in bioinformatics tasks?

Treat agents like just another tool in your toolbox.

Practical tips for responsible agent use

 **Follow reproducible practices**

 **Review, test, and validate all code**

 **Document everything**

Reproducibility is essential for science

Yourself → Future you → Your collaborators → Peers in your field

- Can each of the above people run your code on the same dataset and get the same results?
- How difficult would it be for each of them to reproduce your results?

You are your own most important collaborator

Reproducible practices for responsible AI use

Organize your projects

- One project per directory.
- Keep data, code, and outputs in separate subdirectories.
- Data is read-only; outputs should only be written by the code.

file system tree

```
/data/$USER/projects
├── my-rnaseq-project
│   ├── AGENTS.md
│   ├── data/
│   ├── docs/
│   ├── README.md
│   ├── results/
│   ├── scripts/
│   └── tests/
└── r-dataviz-explorer
    ├── references/
    ├── scripts/
    ├── SKILL.md
    └── tests/
```

Credit to Samarth Mathur for these project examples.

Reproducible practices for responsible AI use

Organize your projects

- One project per directory.
- Keep data, code, and outputs in separate subdirectories.
- Data is read-only; outputs should only be written by the code.

Version control & backup with git & GitHub

- Create a git repository for each project and upload it to GitHub.
- Commit meaningful code changes and sync with GitHub frequently.

file system tree

```
/data/$USER/projects
├── my-rnaseq-project
│   ├── AGENTS.md
│   ├── data/
│   ├── docs/
│   ├── README.md
│   ├── results/
│   ├── scripts/
│   └── tests/
└── r-dataviz-explorer
    ├── references/
    ├── scripts/
    ├── SKILL.md
    └── tests/
```

Credit to Samarth Mathur for these project examples.

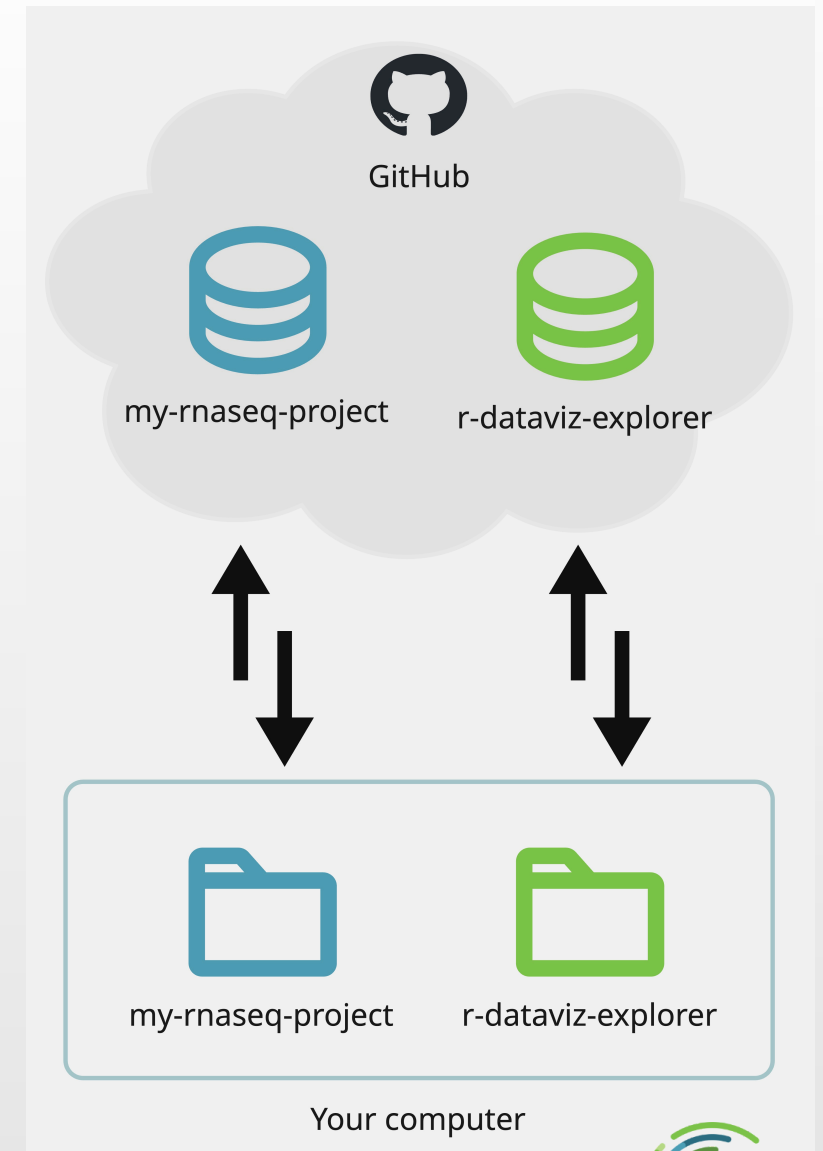
Reproducible practices for responsible AI use

Organize your projects

- One project per directory.
- Keep data, code, and outputs in separate subdirectories.
- Data is read-only; outputs should only be written by the code.

Version control & backup with git & GitHub

- Create a git repository for each project and upload it to GitHub.
- Commit meaningful code changes and sync with GitHub frequently.



Credit to Samarth Mathur for these project examples.

Review, test, and validate

From the NIH Responsible AI policy:

Do not rely on the technology to be a software developer by proxy:

- All well-written code must adhere to security design and ethical principles.
- All code output needs to be reviewed for completeness, quality, efficiency, and, most of all, security.
- Leverage manual and automated validation tools and testing technologies to help ensure these factors.
- If you cannot identify or understand what a piece of AI generated code does, you should not use it.

AI outputs may be incomplete, inaccurate, biased, or fabricated.

Review

- Read the entire output of each prompt and ensure you understand it.
- Be skeptical. Do not blindly trust that the output is complete or correct.
- Treat agents as untrustworthy, in terms of both inputs and outputs.
 - Do not expose passwords, access tokens, or sensitive information.
 - Refer to the [NIH Responsible AI page - approved tools and data types](#) for which types of data can be used with the tools we have access to.

Review

If you cannot identify or understand what a piece of AI generated code does, you should not use it.

Content retrieved September 2026 from the [NIH Responsible AI site](#). Please refer to the official site for the latest guidance.



Challenge yourself!

If you do not understand a piece of AI-generated code, dig into the documentation and tutorials to **learn how it works**.

- Come across a **package or function** you've never seen? Find the documentation and learn how it works!
 - R: use `?` and `help()`. Python: use `help()`
 - Docs websites: <https://rdr.io/>, <https://docspython.org/3/>
- Come across a **large chunk of code** with many patterns you don't understand?
 - Prompt the agent to explain it line-by-line. Verify the explanation by asking for sources and reading them.
- Use forums for in-depth help from people.
 - StackOverflow, biostars, Posit forum, GitHub discussions, your colleagues.

Testing & validation

Write tests to make sure the code behaves as expected.

- Unit tests for individual functions on trivial inputs/data.
- Integration tests run multiple functions or even an entire analysis on small & simulated datasets.
- R: `testthat`. Python: `pytest`, `unittest`

file system tree

```
my-python-project
├── data/
├── src/mypackage
│   └── main.py
└── tests
    └── test_main.py
```

src/mypackage/main.py

```
1 def gc_content(seq):
2     """Fraction of G/C bases in a DNA sequence."""
3     seq = seq.upper()
4     return (seq.count("G") + seq.count("C")) / len(seq)
```

tests/test_main.py

```
1 from mypackage.main import gc_content
2
3 def test_gc_content():
4     assert gc_content("GGCC") == 1.0
5     assert gc_content("ATAT") == 0.0
6     assert gc_content("atgc") == 0.5
```

Testing & validation

Run tests before accepting code.

```
sovacoolkl@arm64-apple-darwin20:~/projects/CCBR/gh-actions-sandbox
>(pydev-3.11) nci-mac → gh-actions-sandbox git:(init) python -m pytest
===== test session starts =====
platform darwin -- Python 3.11.9, pytest-8.3.2, pluggy-1.5.0
rootdir: /Users/sovacoolkl/projects/CCBR/gh-actions-sandbox
plugins: cov-5.0.0, anyio-4.4.0
collected 2 items

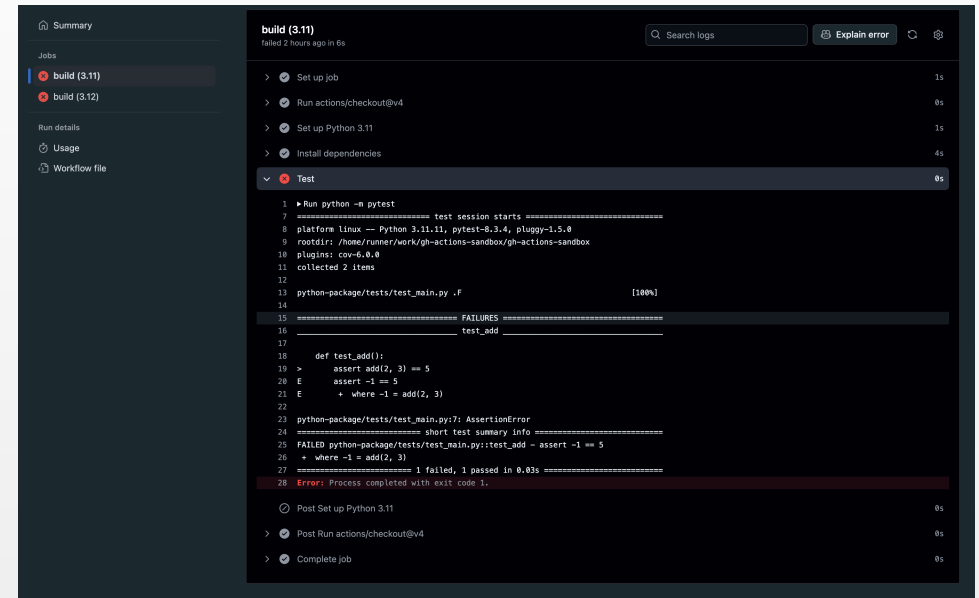
python-package/tests/test_main.py .F [100%]

===== FAILURES =====
test_add

>
def test_add():
>     assert add(2, 3) == 5
E     assert -1 == 5
E     + where -1 = add(2, 3)

python-package/tests/test_main.py:7: AssertionError
===== short test summary info =====
FAILED python-package/tests/test_main.py::test_add - assert -1 == 5
===== 1 failed, 1 passed in 0.07s =====
>(pydev-3.11) nci-mac → gh-actions-sandbox git:(init) □
```

Use GitHub actions to automate.



The screenshot shows a GitHub Actions workflow run for 'build (3.11)' that failed 2 hours ago. The workflow consists of several steps: 'Set up Job', 'Run actions/checkout@v4', 'Set up Python 3.11', 'Install dependencies', 'Test', 'Post Set up Python 3.11', 'Post Run actions/checkout@v4', and 'Complete job'. The 'Test' step is highlighted with a red error icon. The logs for the 'Test' step show the same pytest output as the terminal window on the left, indicating a failure in the 'test_add' function. The error message is 'AssertionError: assert -1 == 5'. The workflow run is shown in a dark theme with a sidebar on the left containing 'Summary', 'Jobs', 'Run details', 'Usage', and 'Workflow file'.

Testing & validation

Use linters and code quality checks.

- Ensure code is formatted for readability, quality, and syntax is correct.
- Security and vulnerability scanning with GitHub Advanced Security.
- R: **air**. Python: **ruff**.

Use **pre-commit** to run checks automatically on every change.

```
check for added large files.....Passed
fix end of files.....Passed
trim trailing whitespace.....Passed
codespell.....Passed
air format.....Passed
parsable-R.....Passed
use-tidy-description.....Passed
lintr.....Passed
- hook id: lintr
- duration: 10.68s

- The project is out-of-sync -- use `renv::status()` for details.
- The project is out-of-sync -- use `renv::status()` for details.
File `tests/testthat/test-normalize.R` is not lint free
/code/tests/testthat/test-normalize.R:148:7: warning: [return_linter] All functions must have an explicit return().
  ggplot2::ggplot()
  ~~~~~
```

Document everything

Inline comments, function docstrings, vignettes, tutorials, getting started guide

Rscript

```
1 #' Add together two numbers
2 #'
3 #' @param x A number
4 #' @param y A number
5 #' @returns The sum of `x` and `y`
6 #' @examples
7 #' add(1, 1)
8 #' add(10, 1)
9 add <- function(x, y) {
10   x + y
11 }
```

add {rvest}

R Documentation

Add together two numbers

Description

Add together two numbers

Usage

```
add(x, y)
```

Arguments

x A number

y A number

Value

The sum of x and y

Examples

```
add(1, 1)
```

```
add(10, 1)
```

Example from <https://r-pkgs.org/man.html>

README Contributing MIT license Security

UCSC Genome Browser session

<https://genome.ucsc.edu/s/kelly%2Dsovacool/CCBR1418>

- [Configuration files](#)

Genome Visualization

1. Edit [./integration/assets/py-genome-tracks.ini](#) with tracks for [pyGenomeTracks](#).
2. Edit [./integration/bin/plot_ATAC_genome_tracks.sh](#) with separate `pyGenomeTracks` commands for each figure.
3. Create the figures and re-render the quarto document:

```
pixi run ./integration/bin/plot_ATAC_genome_tracks.sh
```

View the figure:

- local: [./integration/assets/genome-browser-figure.html](#)
- on github pages: [ATAC-seq genome browser view](#)

Document how you are using agents

from the very beginning of the project:

Create a document outlining how you will and will *not* use LLMs and agents.

docs/ai-usage.md

- Human responsibilities: Formulate the problem statement, design the software, validate the results, write the interpretation.
- Agent roles: assist in generating code, unit tests, and function documentation.

Disclose use of LLMs and agents in commit messages, issue comments, and pull requests.

Generative AI Usage Statement

AI assistance was used for:

- drafting and refactoring code for MOObject integration,
- updating and reconciling test expectations,
- iterating on failing test diagnostics,
- drafting PR description text.

Tool/model used: GitHub Copilot with GPT-5.3-Codex.

Responsible agent use to minimize risks

	Practice	Risks minimized
	Follow reproducible practices	unreproducible work, destructive agent actions
	Review, test, and validate all code	incorrect methods, bugs, inefficiencies, malware
	Document everything	unreadable code, unusable code, lack of transparency

A checklist for agent-assisted Bioinformatics

Check off every item in this checklist before you trust LLM-generated code

1. Understand

- ☐ Can I explain what the code is doing?
- ☐ Did I review all functions, parameters, and methods?

2. Test

- ☐ Did I run it on a small or known dataset?
- ☐ Did I test edge cases and check expected outputs?
- ☐ Have I set up an automated testing framework (pytest, testthat, GitHub Actions)?

3. Sanity check

- ☐ Do the results make biological sense?
- ☐ Did I check for silent problems: missing samples/genes, genome build, coordinates, strandedness, IDs?

4. Make it reproducible

- ☐ Are code, parameters, software versions/environment, and inputs captured?
- ☐ Is the validated code committed & synced with GitHub?

5. Own it

- ☐ Am I comfortable putting my name on this analysis?
- ☐ Have I documented AI use where appropriate?

Final reminders

LLMs produce outputs that *look like* a plausible response to a prompt 🦜

- View these models as another tool in your toolbox alongside IDEs, HPCs, and other software.
- Interpreting the meaning of the generated outputs is your responsibility. Treat it as a learning experience!

Agentic tools can accelerate bioinformatics 🚀

- Just make sure you're facing the direction you want to go!

AI Use Statement

- **Human roles:** all text content was written by the author and reviewed by colleagues.
- **Generative AI use:** Copilot was used for reformatting and syntax lookup while drafting this presentation.
- **Images:** image sources are noted on the slides where they appear; images created by others may or may not have used GenAI for assistance. Images listed without a source were created by the author, without GenAI.

Resources

- NIH Guidelines for Responsible AI
- BTEP Coding Club
- NIH Library
- Related talks:
 - Practical introduction to GenAI - BTEP
 - From Scripts to Skills: Using Generative AI Without Losing Scientific Control - BTEP
 - Organizing and documenting NGS pipelines on GitHub - ABCS Programmer's Corner
- Tutorial: GitHub Copilot in VS Code
- GitHub Copilot in VS Code cheat sheet