A Mac Workflow for Tracking Daily Research Progress

Using ChatGPT dictation, bash scripts, and Git to build a searchable research log

Ronald 'Ryy' G. Thomas

2025-04-12

Table of contents

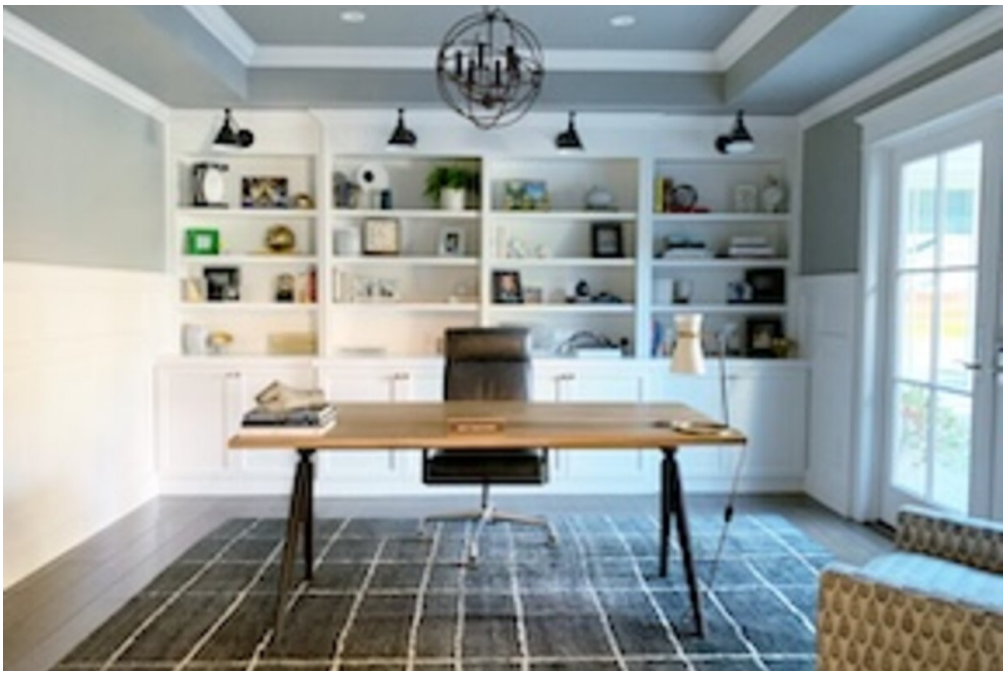

Figure 1: A well-organized research workspace ready for daily progress tracking

Structured workflows turn scattered research notes into a searchable knowledge base.

1 Introduction

I did not really know how to maintain a consistent research log until I started losing track of what I had accomplished across ten concurrent projects. The problem was not motivation; it was friction. Opening a text editor, remembering the right file, formatting the entry, and committing the changes felt like too many steps for something that should take sixty seconds.

The breakthrough came when I realized that macOS dictation could capture my thoughts hands-free, and ChatGPT could turn rambling voice notes into concise, structured summaries. By wrapping these two capabilities in a few short bash scripts and pushing results to Git, I had a daily research logging system that required almost no manual typing.

This post documents that workflow from start to finish. It covers the folder structure, the dictation prompt script, the log-appending script, the search script, and the Git integration that ties everything together. The entire system runs from the terminal with three short commands.

More formally, this post documents the file-system layer of the Workflow Construct described in post 52. The convention that all research projects live under `~/prj/NN-name/`, are synced continuously by Dropbox, and are snapshotted intentionally by Git, is the substrate on which every other layer depends. The dictation-driven research-log workflow described below is one specific use of that substrate.

1.1 Motivations

- I was managing ten research projects simultaneously and could not remember which analyses I had run two weeks earlier on any given project.
- Handwritten lab notebooks did not support full-text search, which made retrieval slow and unreliable.
- Existing note-taking applications introduced too many dependencies and did not integrate with my Git-based version control workflow.
- I wanted a system that could produce a structured log entry in under ninety seconds, including the commit and push to a remote repository.
- Voice dictation felt like the lowest-friction input method, but raw dictation output is too messy to store directly.
- I needed per-project search capability so that running a single command from any project directory would surface all related log entries.

1.2 Objectives

1. Set up a structured directory layout for managing multiple concurrent research projects.
2. Create a bash script (`dp`) that generates a formatted ChatGPT prompt, priming the model for research note summarization.
3. Build an append-and-push script that copies a ChatGPT summary from the clipboard into a centralized daily log and commits the update to Git.
4. Write a search script (`sp`) that retrieves all log entries for the current project using `ripgrep`.

I am documenting my learning process here. If you spot errors or have better approaches, please let me know.

2 Prerequisites and Setup

This workflow assumes a macOS environment with the following tools available:

- **Terminal**: Any terminal emulator (iTerm2, Kitty, or the built-in Terminal.app)
- **Bash or Zsh**: The scripts use standard POSIX shell features and work in either shell
- **Git**: For version control of the daily log repository

- **ripgrep** (`rg`): For fast, project-scoped search across log entries
- **pbcopy / pbpaste**: macOS clipboard utilities (built in)
- **ChatGPT**: A browser tab or desktop application for dictation summarization
- **macOS Dictation**: Enabled in System Settings under Keyboard (press the microphone key or double-tap the Function key)

No R packages are required for this workflow. The entire system runs through the shell. If you do not have `ripgrep` installed, you can add it with Homebrew:

```
brew install ripgrep
```

3 What is a ChatGPT-Assisted Research Log?

A ChatGPT-assisted research log is a daily note-taking system where the researcher dictates progress notes using macOS voice input, submits the raw dictation to ChatGPT for summarization and formatting, and stores the resulting structured summary in a version-controlled text file. Think of it as having a research assistant who listens to your stream-of-consciousness update and hands back a clean, dated, project-tagged paragraph.

The key insight is that the summarization step solves the primary pain point of voice-to-text workflows: raw dictation is verbose and poorly structured, but passing it through a language model produces concise, searchable entries that are suitable for long-term storage. The project-tagging convention (prefixing every line with the directory name) enables per-project retrieval from a single centralized log file.

4 Getting Started: Research Folder Structure

A structured directory keeps research files accessible and prevents clutter. The following command creates a workspace with a central logging directory and ten project folders:

```
mkdir -p ~/prj/research_update \
  ~/prj/{X1,X2,X3,X4,X5,X6,X7,X8,X9,X10}
```

Each project folder should contain a consistent internal structure:

- `notes.md` – Running log of project-specific notes
- `references.bib` – Citation management
- `data/` – Datasets and related files
- `coding/` – Code and analyses
- `figures/` – Graphs and visualizations
- `tables/` – Data summaries
- `archive/` – Storage for non-current files

The centralized daily log lives at a fixed location that all scripts reference:

```
~/prj/research_update/daily_log.md
```

This single file accumulates entries from every project. The project-tagging convention described in the next section makes it possible to filter entries by project despite storing everything in one place.

Initialize the logging repository with Git so that every update is tracked:

```
cd ~/prj/research_update
git init
git add daily_log.md
git commit -m "Initialize daily research log"
```

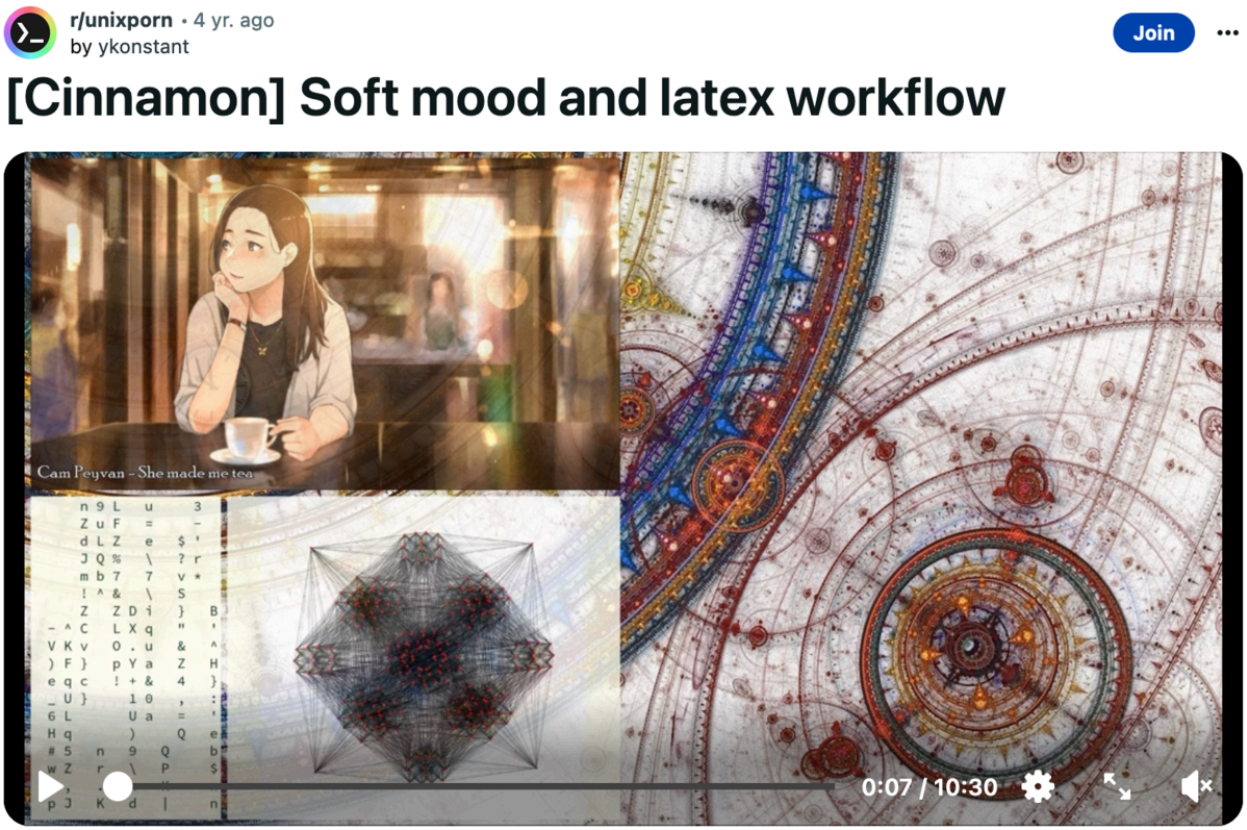

Figure 2: Workspace tools arranged for a productive research session

5 Deeper Analysis: The Three-Script Workflow

The entire system consists of three short bash scripts that chain together: `dp` generates the prompt, the append script commits the result, and `sp` searches the log. Each script is designed to be run from within a project directory so that the project name is captured automatically.

5.1 Script 1: The Dictation Prompt (`dp`)

This script constructs a ChatGPT prompt that includes the current date, time, and project directory name. It copies the prompt to the clipboard so that the researcher can paste it directly into ChatGPT.

```
#!/bin/bash

current_time=$(date +"%Y-%m-%d %H:%M:%S")
current_dir=$(basename "$PWD")

prompt="I'm an academic biostatistician. I'm working
on a data analysis project. I'm about to dictate
daily research progress notes.
When I'm done, provide a concise summary that
includes:

1. The date and time of dictation
($current_time). The line with date and time
should be the second line of the summary. The
first line should be blank. The date and time
line should be enclosed in a box of ascii
characters to set it apart.
2. The name of the current research project
directory ($current_dir).
3. Each line of the summary including the blank
line and the date and time line and enclosing
box lines should begin with \"$current_dir:\"
so that it can be extracted using ripgrep.

The notes start here: "

echo -n "$prompt" | pbcopy

echo "Prompt copied to clipboard."
echo "Paste it into ChatGPT when ready."
```

The project-prefixing instruction in item 3 is the critical design decision. By requiring every line of the summary to begin with the directory name followed by a colon, the `sp` script can later extract all entries for a given project using a single `ripgrep` call. The ASCII box around the date and time creates a visual separator between entries when reviewing the raw log file.

5.2 The Dictation Process

After running `dp`, the workflow proceeds as follows:

1. Open ChatGPT in a browser or desktop application.

2. Paste the prompt from the clipboard into the ChatGPT input field.
3. Submit the prompt so that ChatGPT is primed for summarization.
4. Click the ChatGPT microphone icon and dictate your research notes. Speak naturally; there is no need to say "new line" or "period" because ChatGPT will handle formatting.
5. Submit the dictated text for summarization.
6. Review the summary. Iterate with ChatGPT (either by voice or by typing) to refine the output.
7. Copy the final summary to the clipboard.

5.3 Script 2: Append and Push

This script reads the summary from the clipboard, appends it to the centralized daily log, and pushes the update to the remote Git repository. Save it somewhere on your `$PATH` (for example, `~/bin/append_log`).

```
#!/bin/bash

current_dir=$(basename "$PWD")
current_time=$(date +"%Y-%m-%d %H:%M:%S")

clipboard_content=$(pbpaste)

echo "$clipboard_content" \
  >> ~/prj/research_update/daily_log.md
echo "" >> ~/prj/research_update/daily_log.md

echo "Update for $current_dir appended to"
echo "daily_log.md in ~/prj/research_update"

cd ~/prj/research_update
git add .
git commit -a \
  -m "Daily log update $(date +'%Y-%m-%d')"
git push
```

The blank line appended after the clipboard content ensures that consecutive entries are visually separated in the log file.

5.4 Script 3: Search Project (sp)

This script searches the daily log for all entries tagged with the current project directory name. The `cut -c6-` strips the directory prefix from each line for cleaner output. Adjust the cut range if your project directory names are longer than four characters.

```
#!/bin/bash
current_dir=$(basename "$PWD")
rg "$current_dir" \
  ~/prj/research_update/daily_log.md \
  | cut -c6-
```

Running `sp` from within `~/prj/X3/` will display every log entry tagged with `X3:`, regardless of when it was written. This is the retrieval mechanism that makes the single-file approach practical.

5.5 Things to Watch Out For

1. **Clipboard conflicts.** If you copy something else to the clipboard between running `dp` and pasting into ChatGPT, the prompt is lost. Run `dp` immediately before switching to ChatGPT.
2. **Project directory naming.** The `ripgrep` search matches on directory name, so avoid names that are substrings of other names. `X1` will match `X10` entries. Use names like `proj_alpha` instead of sequential numbers if you have many projects.
3. **Git remote setup.** The append script assumes a remote is configured. Run `git remote add origin <url>` in `~/prj/research_update/` before using the push feature for the first time.
4. **Dictation accuracy.** macOS dictation works best in a quiet environment. Technical terms (biostatistics jargon, package names, function names) are often misheard. Review the ChatGPT summary carefully before appending.
5. **Cut column offset.** The `cut -c6-` in the search script assumes a four-character project name plus a colon and space. Adjust this value if your directory names are longer.

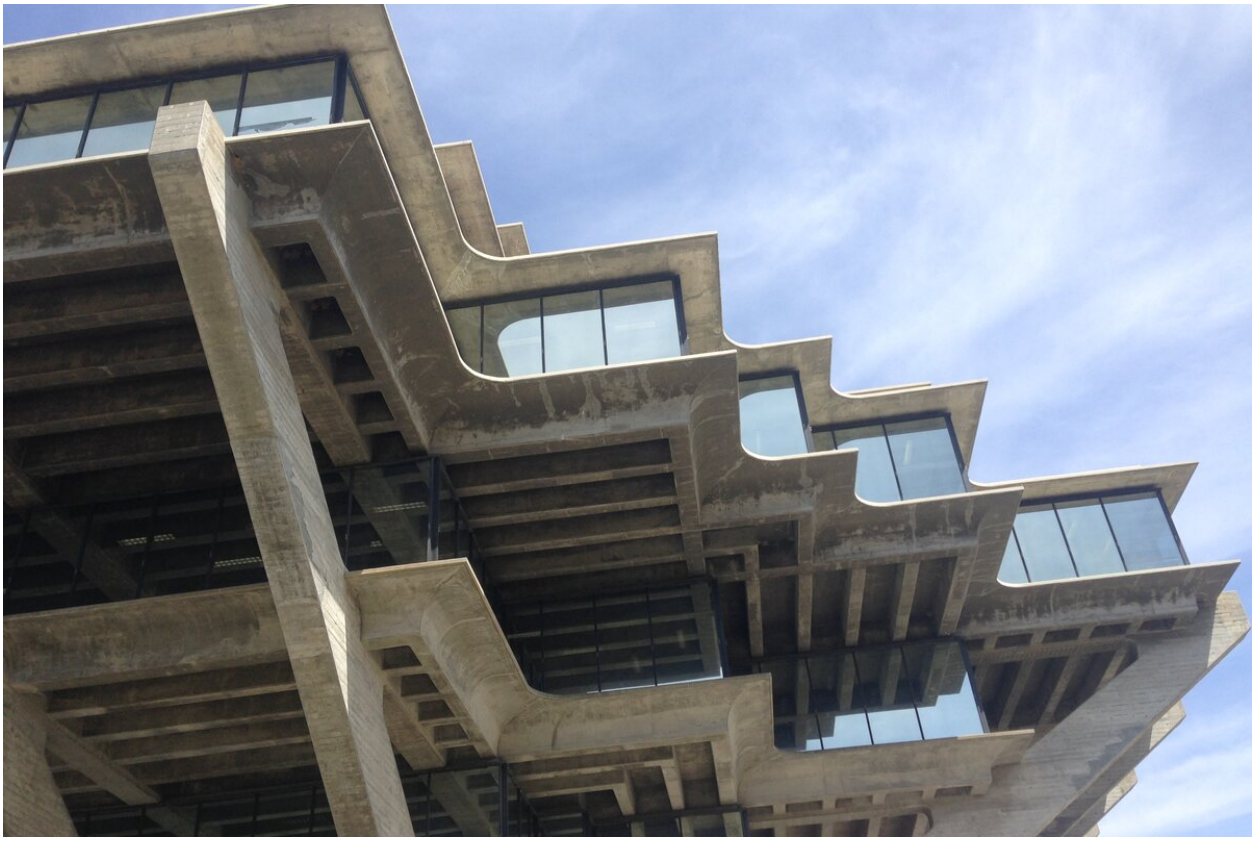

Figure 3: Organized notes and tools for daily research tracking

6 What Did We Learn?

6.1 Lessons Learnt

Conceptual Understanding:

- A single centralized log file with project-tagged lines is more practical than per-project log files because it supports cross-project search and provides a chronological record of all research activity.
- Voice dictation followed by language model summarization produces higher-quality log entries than direct typing because the researcher can speak freely without worrying about formatting.
- The project-prefix convention (every line begins with the directory name) is a simple but effective tagging system that enables structured retrieval from an unstructured text file.
- Version-controlling the daily log with Git provides an immutable audit trail that satisfies reproducibility requirements for academic research.

Technical Skills:

- Using `pbcopy` and `pbpaste` to pass data between shell scripts and GUI applications (ChatGPT) through the macOS clipboard.

- Writing bash scripts that extract context automatically (`basename "$PWD"` for the project name, `date` for timestamps) to minimize manual input.
- Using `ripgrep` with simple string patterns for fast retrieval from large text files.
- Chaining `git add`, `git commit`, and `git push` in a single script to reduce version control friction to a single command.

Gotchas and Pitfalls:

- The clipboard is a shared resource: any copy operation between `dp` and the paste step will overwrite the prompt. Minimize context switching.
- macOS dictation sometimes fails silently if the microphone permissions are not granted to the active application. Verify in System Settings.
- The `cut -c6-` column offset in the search script is fragile. If project directory names vary in length, consider using `sed` to strip the prefix instead: `sed "s/^$current_dir: //"`.
- If the Git remote is unreachable (network issues), the push will fail but the local commit still succeeds. The log entry is preserved locally; push manually when connectivity returns.

6.2 Limitations

- This workflow is macOS-specific due to its reliance on `pbcopy`, `pbpaste`, and macOS dictation. Adapting it to Linux would require `xclip` or `xsel` and a different speech-to-text tool.
- The system depends on ChatGPT availability. If the service is down or the API changes, the summarization step fails, though dictated notes can still be appended manually.
- There is no automated backup beyond Git. If the local repository is corrupted and the remote is lost, all log entries are unrecoverable.
- The project-prefix tagging convention does not support hierarchical organization (sub-projects, milestones, or categories within a project).
- The workflow does not handle attachments, images, or structured data. It is text-only by design.
- Long-running projects will accumulate large numbers of entries in a single file. Performance of `ripgrep` remains excellent at scale, but human readability of the raw file degrades over time.

6.3 Opportunities for Improvement

1. **Direct ChatGPT API integration.** In theory, it should be possible to send dictated text directly to the ChatGPT API from a bash script, bypassing the browser entirely. This would reduce the workflow from three manual steps to one command.
2. **Whisper for local transcription.** Replacing macOS dictation with OpenAI Whisper running locally would improve transcription accuracy for technical terminology and remove the dependency on macOS-specific features.
3. **Structured output format.** Switching the log from plain text to a structured format (JSON or YAML) would enable richer queries, date-range filtering, and integration with analysis dashboards.

4. **Automated daily reminders.** A cron job or `launchd` agent that opens a terminal prompt at a set time each day would reduce the likelihood of forgetting to log.
5. **Cross-platform support.** Abstracting the clipboard commands behind environment detection (`pbcopy` on macOS, `xclip` on Linux, `clip.exe` on WSL) would make the scripts portable.
6. **Log visualization.** A simple R or Python script that parses the daily log and generates a timeline or heatmap of research activity per project would add a useful retrospective view.

7 Wrapping Up

This workflow turns daily research logging from a chore into a lightweight habit. The entire process – dictating notes, generating a summary, appending to the log, and pushing to Git – takes under ninety seconds once the scripts are in place.

What I found most valuable was the reduction in friction. Before this system, I would skip logging on busy days because the overhead felt too high. With voice dictation and automated summarization, the barrier dropped low enough that daily logging became sustainable.

If you are managing multiple concurrent research projects and struggling to maintain a consistent record of progress, I would encourage you to try this approach. The three scripts are short enough to customize in an afternoon, and the payoff in searchable, version-controlled documentation accumulates rapidly.

Main takeaways:

- Three bash scripts (`dp`, append, `sp`) provide a complete dictation-to-search workflow.
- ChatGPT summarization converts raw voice dictation into concise, structured log entries.
- Project-prefixed lines enable per-project retrieval from a single centralized log file.
- Git integration provides version control and remote backup with no additional effort.

8 See Also

Related posts:

- Configure the Command Line for Data Science Development – Terminal and Zsh setup that complements this workflow
- Creating a GitHub Dotfiles Repository – Managing configuration files across machines

Key resources:

- ripgrep documentation – The search tool used in the `sp` script
- OpenAI Whisper – Local speech-to-text alternative to macOS dictation
- Git documentation – Reference for the version control commands used in the append script
- macOS Dictation guide (Apple Support) – Setup instructions for the built-in dictation feature

9 Reproducibility

This post describes a shell-based workflow with no R analysis pipeline. To reproduce the system, create the three scripts described above and place them on your `$PATH`:

```
mkdir -p ~/bin

cp dp.sh ~/bin/dp
cp append_log.sh ~/bin/append_log
cp sp.sh ~/bin/sp

chmod +x ~/bin/dp ~/bin/append_log ~/bin/sp

export PATH="$HOME/bin:$PATH"
```

System requirements:

- macOS 13+ (Ventura or later) for reliable dictation
- Git 2.30+
- ripgrep 13.0+
- A ChatGPT account (free tier is sufficient)

Files in this post:

File	Purpose
`dp`	Generate ChatGPT dictation prompt
`append_log`	Append summary and push to Git
`sp`	Search log by project name

10 Let's Connect

- **GitHub:** rgt47
- **Twitter/X:** @rgt47
- **LinkedIn:** Ronald Glenn Thomas
- **Email:** rgtlab.org/contact

I would enjoy hearing from you if:

- You spot an error or a better approach to any of the code in this post.
- You have suggestions for topics you would like to see covered.
- You want to discuss R programming, data science, or reproducible research.
- You have questions about anything in this tutorial.
- You just want to say hello and connect.