How I learned to stop worrying and love plain text

Alán Muñoz

January 9, 2024

Goal

Showcase plain text and its tools as a valid approach in place of more complex formats and tools.

What do I mean by plain text?

- I define “plain text” as data represented by readable characters and in a human-readable format.
- It is understandable without an specialised tool.

Examples

Plain text is

- Markdown
- CSV files
- Source code (pre-compilation)

Examples

Plain text is

- Markdown
- CSV files
- Source code (pre-compilation)

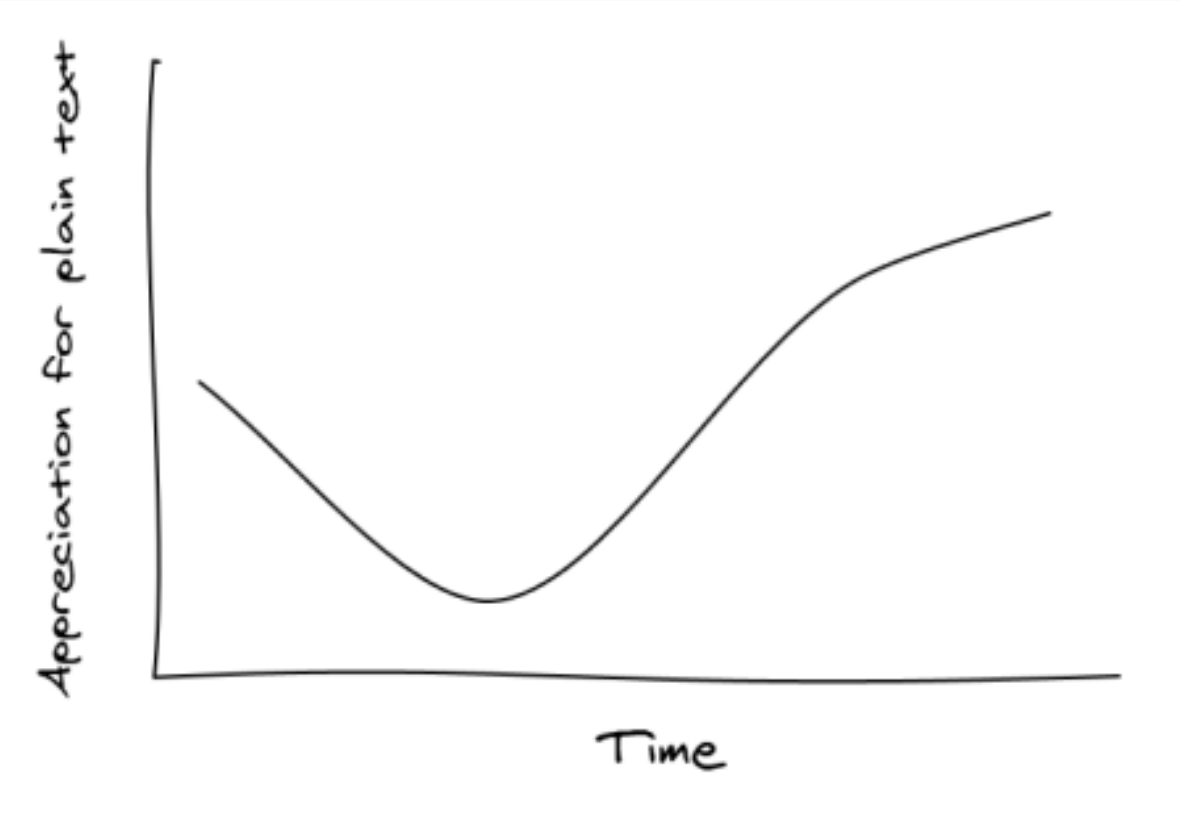
And is not

- Data containing floating points
- Images
- Html

It offers multiple benefits

- Universal
- Lightweight
- Portable
- The maintenance cost of its toolsets is lower than GUIs

Plain text may sound like an outdated notion



Essential text processing tools

- cat (+tac): print file (or reverse line order)
- head/tail: print first/last n lines of file
- rev: Reverse lines
- wc: Count words/lines/characters
- sort: Sort lines
- uniq: Show unique lines
- tr: Replace/delete characters
- grep (+ripgrep): Select lines
- cut: Select columns
- paste: Combine documents or the lines in one

Additional useful tools

- tee: Read from stdin and write to stdout and files.
- sponge: Recycle input file when processing (moreutils package)
- jq: Query json files/stdin
- sed: Standard editor, useful for simple edits with no variables
- awk: Versatile programming language for more complex queries
- perl (+python): Languages for complex one-line queries

Overview of plain text tools/interfaces

Interfaces minimise friction between processes

- Modal editing
- Git interfaces: Magit (Emacs), Fugitive (Vim), LazyGit (CLI)
- File management interfaces
- Smart text editing

Modal editing

- Use regular expressions to find and replace regions
- change, surround, replace, exchange, text becomes objects to edit
- Code folding

File management

- We can rename files in bulk
- Access server data remotely

File management

- Most operations do not require complex graphical interfaces
- Anything a terminal could ssh into, but pretend it is local

Plain text can be a git interface

- Stage-Commit-Push files to a repository
- Merging and solving conflicts

Plain text can be a git interface

- Git operations become routine
- Select specific code regions to commit
- Explore previous versions
- Issues and requests included
- Learn git as you go, it shows the commands you invoke
- Interfaces: lazygit (TUI), fugitive (Vim), magit (Emacs)

Diff and Undo

- Visualise file differences in undo

There is a markup language for cooking

Cooklang – Recipe Markup Language

```
Then add @salt and @ground black pepper{} to taste.

Poke holes in @potato{2}.

Place @bacon strips{1%kg} on a baking sheet and glaze with @syrup{1/2%tbsp}.
```

Many text tools improve the Python development experience

- black: auto-formatting
- isort: Sort imports
- jupytext: Jupyter Notebook <-> Python script
- LSP (Language Server Protocol): Code linter
- Live coding interface

Plain text can:

Run notebooks with multiple languages

Include bibliography

Task management

Basic spreadsheets

Run timers and deltas

Note forwards and backlinks

Where does plain text underperform?

- Review/Collaboration (e.g., GDrive review tools)
- Visualisation of complex data (e.g., plotting dense data)
- Replacing structured data (e.g., spreadsheets, big data frames)

Other things that plain text is capable of

- Bibliography management
- Database of personal notes
- RSS (Really Simple Syndication)

Why is fast feedback important for data scientists and software engineers?

1. Load data into memory
2. Process the data
3. Bring functions together to give it structure
4. Iterate until data has been processed enough
5. - Feedback speed from where the data is matters

Does it actually make a difference?

Depends on one's workflow, but it reduces friction usually caused by switching between a multitude of apps.

Conclusions:

Plain text universal and portable

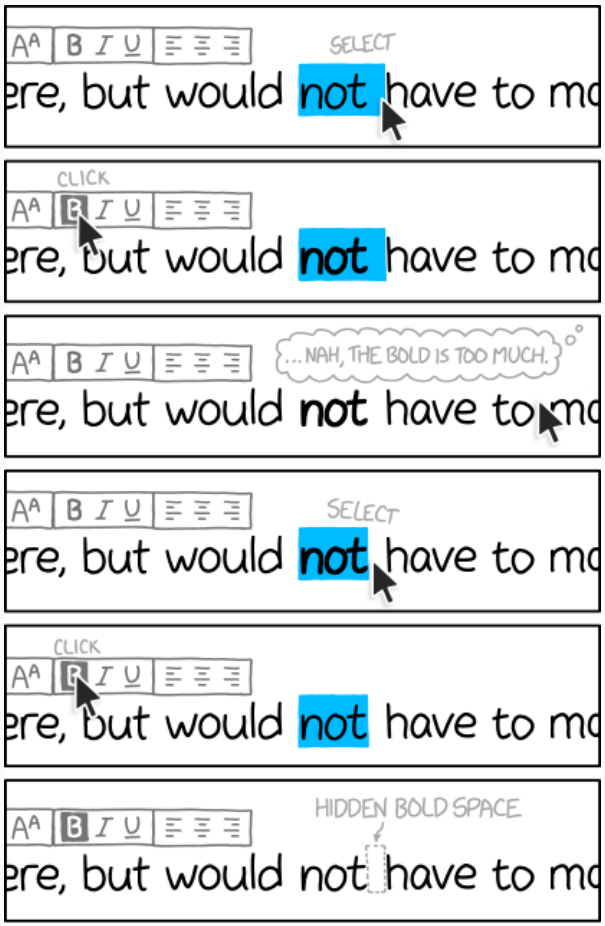
Can be converted to any other text format

Enables version control

Lowers the feeling working on a server vs a local machine

There are already toolkits to process it in most imaginable ways

Relevant XKCD



Resources

- tldr: DIsplay simple pages for command line tools
- lazygit: Git command line interface
- jupytext: Jupyter notebooks <-> plain text
- moreutils: Additional CLI tools
- mermaid: Generate diagrams from plain text (Github renders)
- pandoc: Convert markup languages into each other
- plain-text-everything: List of other projects that use plain text

Other fun tools

- more/less: Look at file, also interactively
- screen: Run background sessions and restart them
- du: Check folder size
- fzf: command fuzzy finder
- fish: bash with QOL improvements (not always compatible with bash/zsh)
- htop/btop: process management
- rsync: Synchronise copies of the same files
- parallel: Use multiple cores
- csvtools: Tools for managing CSV