Flexibility vs. Waterfall

- Having chosen this all-inclusive format, I am free to experiment with different (automated) rewritings, knowing that nothing will ever destroy the original information.
- I may waste my own time, or waste CPU power, but that is less important than having a solid footing for the starting point.
- This is part of the “plan”. This arrangement allows me to change my mind (multiple times, if needed) knowing that the hard work (entering and collecting data) will be reusable.
- Goal: I will never be boxed in.

Flexibility vs. Waterfall (con't)

- I can throw all of my code away and re-start, if I discover that that is the best choice.
- I like Lisp because it lets me be cavalier with my code.
- I dislike C because it makes me calcify my decisions early in the process
  - Calcification is a hidden form of *waterfall* design
  - I imagine that Rust has this same problem, as do most statically-typed languages
- Normalizing data achieves design flexibility (I favour triples)