In Data Analysis We Trust

~~Best~~ Good Enough Practices for Reproducible Computing ~~in the Age of AI~~

Ming 'Tommy' Tang, PhD

- Director of Bioinformatics, AstraZeneca
- Divingintogeneticsandgenomics.com
- Nextflow summit keynote
- April 29, 2026

THE PROBLEM

95%

of bioinformatics analyses cannot be reproduced.

Of papers I read, approximately **95% fail to include details necessary for replication.** It's very hard to build off of research like this. — *Titus Brown*

THE PROBLEM

95%

of bioinformatics analyses cannot be reproduced.

Of papers I read, approximately **95% fail to include details necessary for replication.** It's very hard to build off of research like this. — *Titus Brown*

THE PROBLEM

Method Matters — both for academia and industry

Detection of gene fusions

We detected gene fusions in regions of genomic complexity using an approach that integrates multiple independent fusion algorithms, and then removed those found in normal tissue. Putative fusions were validated by de novo assembly. A total of 1277 normal (nonneoplastic) samples from 43 different tissues were obtained from the NHGRI GTEx consortium (database version 4) and used to remove artifacts. All fusions were visually inspected if one or both genes involved chromoplexy or were adjacent (up to 1 Mbp). Fusions were further filtered by quality of the realigned transcript, breakpoint coverage, and gene expression.

⚠ Vague methods like "we integrated multiple algorithms" are not reproducible. Always specify tools, versions, and thresholds.

THE PROBLEM

Method Matters — both for academia and industry

Detection of gene fusions

We detected gene fusions in regions of genomic complexity using an approach that integrates multiple independent fusion algorithms, and then removed those found in normal tissue. Putative fusions were validated by de novo assembly. A total of 1277 normal (nonneoplastic) samples from 43 different tissues were obtained from the NHGRI GTEx consortium (database version 4) and used to remove artifacts. All fusions were visually inspected if one or both genes involved chromoplexy or were adjacent (up to 1 Mbp). Fusions were further filtered by quality of the realigned transcript, breakpoint coverage, and gene expression.

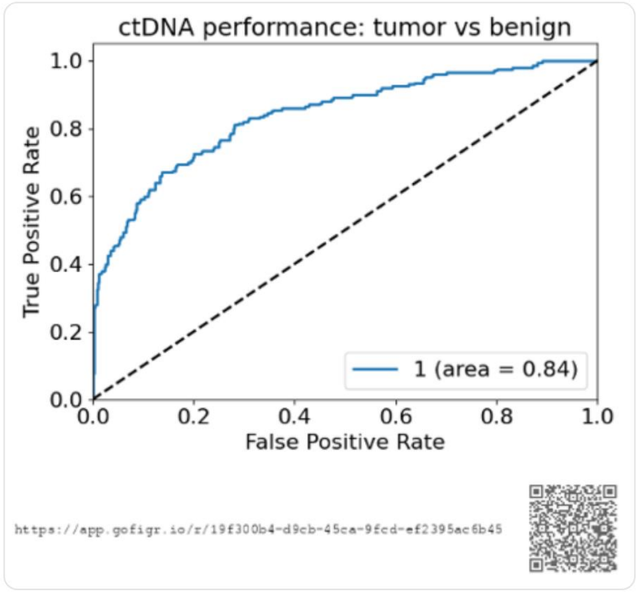
⚠ Vague methods like "we integrated multiple algorithms" are not reproducible. Always specify tools, versions, and thresholds.

Monday, February 17th

Alyssa 5:31 AM

Hi, can I have this figure broken down by cancer stage? Investor presentation at 10am

image.png



THE PROBLEM

The Stakes: A Real Clinical Catastrophe

Flawed Cancer Trial at Duke Sparks Lawsuit

By **Jennifer Couzin-Frankel** | Sep. 9, 2011 , 3:38 PM

A dozen plaintiffs have filed a **lawsuit** against Duke University and administrators, researchers, and physicians there, alleging that they engaged in fraudulent and negligent behavior when they enrolled cancer patients in a clinical trial compromised by faulty data. The lawsuit, filed Wednesday in a North Carolina court, comes 14 months after a **scandal erupted at Duke** that finally exposed the extent of the trial's problems: in July 2010, Duke oncologist Anil Potti, whose work was central to the trial, admitted that he had embellished his resume and later **resigned**.

— Science, Aug 2011

$3B–$5B

Average cost to develop a single drug

Real People

Enrolled in trials based on flawed bioinformatics

⚠ The root cause was not complex statistics. It was sloppy data handling that could have been prevented with basic reproducibility practices.

[The Importance of Reproducible Research in High-Throughput Biology]

By Dr.Keith A. Baggerly from MD Anderson Cancer Center

THE PROBLEM

Why Is Reproducibility So Hard?

Data Not Available

Raw data withheld or not version controlled. "Available upon reasonable request" rarely works in practice.

Scripts Not Shared

Analysis code omitted entirely, or buried in supplementary materials with no context.

Missing Method Detail

Parameter choices, filtering steps, and tool versions left undocumented.

Environment Drift

Different OS, R/Python versions, or package releases produce silently different results.

SO WHAT · WHY IT MATTERS

You Can't Build Solid Science on Shaky Foundations

Irreproducible results don't just waste time — they contaminate downstream decisions, mislead collaborators, and ultimately put patients at risk.

SO WHAT · WHY IT MATTERS

"Your closest collaborator is you six months ago — and they don't answer emails."

If you can't reproduce your own analysis next year, no one else can either. And you *will* need to rerun it — when a reviewer asks, when a dataset grows, when a new collaborator joins.

Reproducibility isn't just for others. It's professional self-defense.

NOW WHAT · THE ROADMAP

Six Pillars of Reproducible Bioinformatics

Tidy Data

Structure datasets for clarity and analysis

Project Organization

Consistent files, folders, and naming

Version Control

Track changes with Git for provenance

Environment Management

Pin dependencies for reproducibility

Literate Programming

Weave code, narrative, and results together

Clean Code

Write readable, reusable, and well-tested functions

No single practice transforms reproducibility overnight. Applied together, these six pillars form a reliable foundation — from the moment you receive raw data to the moment you submit a manuscript or deliver a regulatory package.

PILLAR 1: TIDY DATA

Data & Spreadsheets

PILLAR 1: TIDY DATA

Bad Data In, Bad Science Out

https://divingintogeneticsandgenomics.com/post/my-odyssey-of-obtaining-scrnaseq-metadata/

PILLAR 1: TIDY DATA

Bad Data In, Bad Science Out

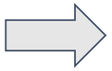
Can you spot what's wrong? Merged cells, color-coded meaning, mixed data types in one column, free-text notes in data fields — all common spreadsheet mistakes that break downstream analysis.

Excel silently converts gene names like `SEPT2` to dates and `MARCH1` to numbers. These errors have contaminated thousands of published genomics datasets.

https://divingintogeneticsandgenomics.com/post/my-odyssey-of-obtaining-scrnaseq-metadata/

PILLAR 1: TIDY DATA

Best practices using spreadsheets



	A	B	C	D	E
1	strain	genotype	min	replicate	response
2	A	normal	1	1	147
3	A	normal	1	2	139
4	B	normal	1	1	246
5	B	normal	1	2	240
6	A	mutant	1	1	166
7	A	mutant	1	2	179
8	B	mutant	1	1	178
9	B	mutant	1	2	172
10	A	normal	5	1	334
11	A	normal	5	2	354
12	B	normal	5	1	514
13	B	normal	5	2	611
14	A	mutant	5	1	451
15	A	mutant	5	2	474
16	B	mutant	5	1	412
17	B	mutant	5	2	447

https://www.tandfonline.com/doi/full/10.1080/00031305.2017.1375989

PILLAR 1: TIDY DATA

Five Rules for Spreadsheets That Actually Work

1

One value per cell

Never combine multiple pieces of information in a single cell.

2

Use good field names

No spaces, no special characters, no leading numbers. Use `janitor::clean_names()` to fix existing headers.

3

Be consistent

Pick one value per category: "male" not sometimes "M", "Male", or "male". Same variable name across all files.

4

Use correct null values

Leave them blank. Never use -999 or other values.

5

Save as plain text

CSV or TSV — not .xlsx — for long-term portability and version control compatibility.

PILLAR 1: TIDY DATA

Consistency Is Not Optional

❌ What breaks pipelines

- `Glucose_10wk` vs `gluc_10weeks` vs `10 week glucose`
- `153` vs `mouse153` vs `mouse-153F`
- `M` vs `male` vs `Male`

```
```{r}

LUAD_index<- !is.na(TCGA_LUAD_data$paper_expression_subtype)
LUSC_index<- !is.na(TCGA_LUSC_data$paper_Expression.Subtype)
```

✅ What works

- One canonical name per variable — documented in a data dictionary
- One subject ID format, applied everywhere
- One value per category, applied consistently across all batches

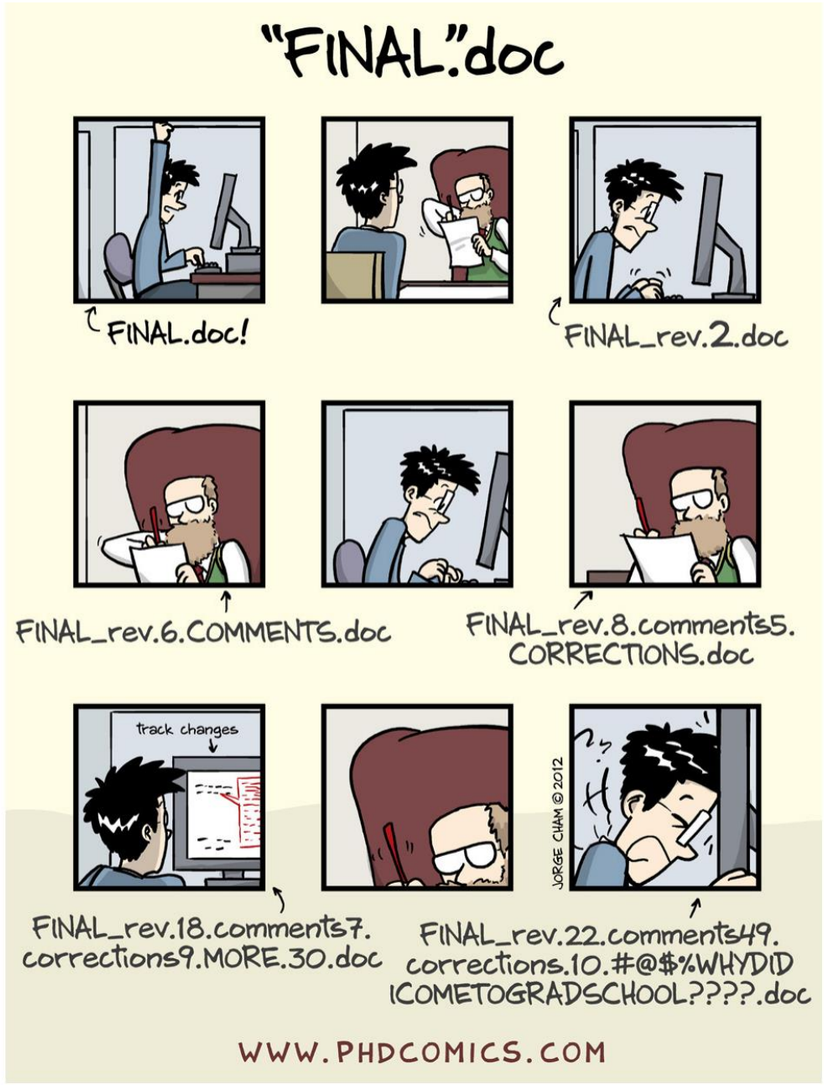
Inconsistency forces every downstream analyst to reverse-engineer your intent. Don't make them guess.

It is not rocket science, but learning how to use spreadsheet effectively will greatly benefit your research, avoid errors and have a happy collaborator (me)!
```

PILLAR 2: FILE NAMING and PROJECT ORGANIZATION

File Naming & Project Organization

Naming files is hard



PILLAR 2: FILE NAMING and PROJECT ORGANIZATION

Bad File Names Are a Reproducibility Crisis

NO

myabstract.docx
Joe's Filenames Use Spaces and Punctuation.xlsx
figure 1.png
fig 2.png
JW7d^(2sl@deletethisandyourcareerisoverWx2*.txt

YES

2014-06-08_abstract-for-sla.docx
joes-filenames-are-getting-better.xlsx
fig01_scatterplot-talk-length-vs-interest.png
fig02_histogram-talk-attendance.png
1986-01-28_raw-data-from-challenger-o-rings.txt

We've all seen it — and probably done it:

- `analysis_final.R`
- `analysis_final_v2.R`
- `analysis_FINAL_USE_THIS.R`
- `analysis_really_final.R`

⚠ If your file names include "final," "v2," or "USE_THIS" — you need Git and better naming conventions.

PILLAR 2: FILE NAMING and PROJECT ORGANIZATION

Three Principles for File Names That Last

Machine Readable

No spaces, no special characters. Use underscores or hyphens. Glob and regex friendly.

Human Readable

Add a slug that describes the content. `2024-01-15_rnaseq_qc_report` beats `output_v3` every time.

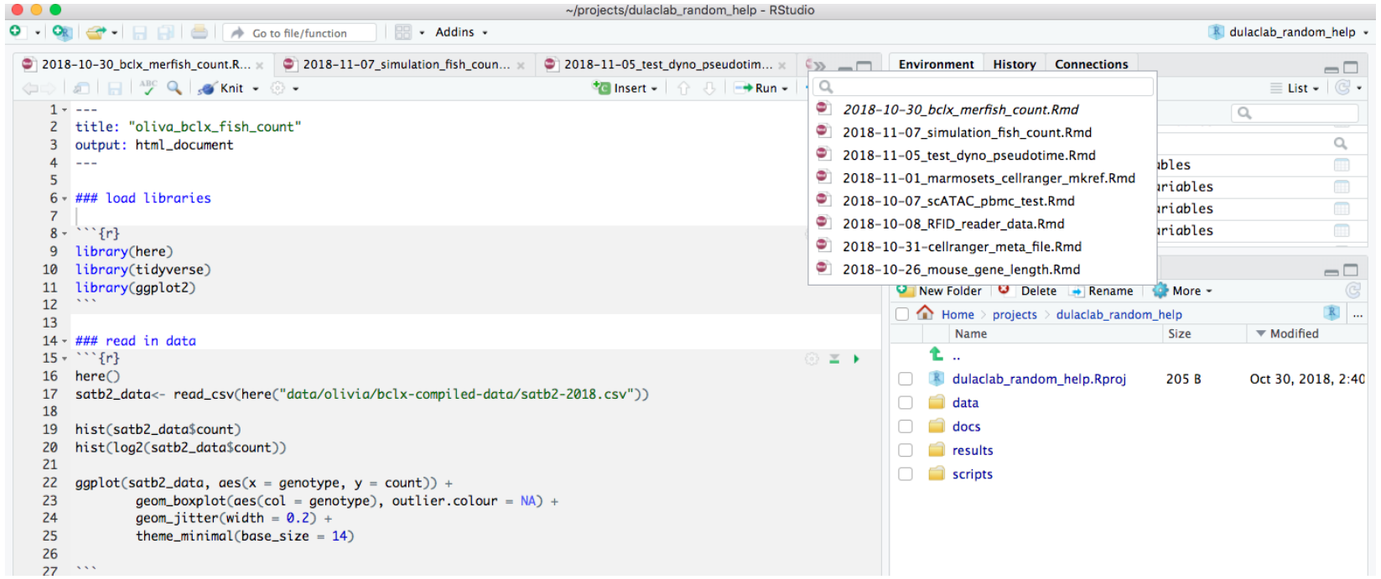
Default Ordering

Lead with ISO 8601 dates (YYYY-MM-DD) or zero-padded numbers. Files sort correctly in any OS.

Credit: Jenny Bryan — http://www2.stat.duke.edu/~rcs46/lectures_2015/01-markdown-git/slides/naming-slides/naming-slides.pdf

PILLAR 2: FILE NAMING and PROJECT ORGANIZATION

A Consistent Project Structure Saves Hours



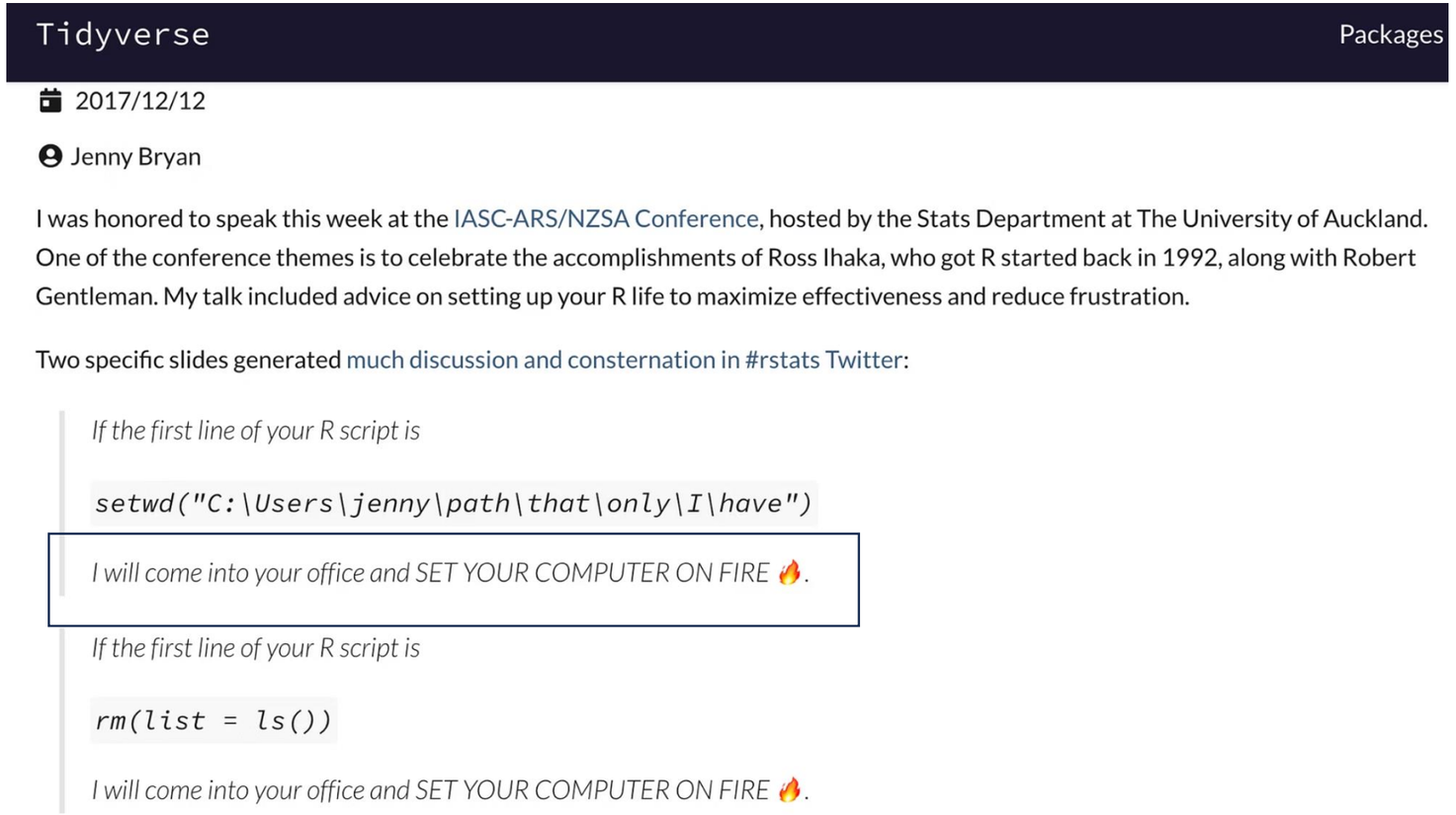
A standard folder layout means any collaborator (including future-you) can navigate the project instantly.

- `data/` — raw, never modified
- `scripts/` — all analysis code
- `results/` — outputs, figures, tables
- `docs/` — reports, README

Protect raw data: `chmod u-w -R data/` — make it physically impossible to overwrite.

PILLAR 2: FILE NAMING and PROJECT ORGANIZATION

Use `here()` — Never Use Absolute Paths Again



Tidyverse Packages

2017/12/12

Jenny Bryan

I was honored to speak this week at the IASC-ARS/NZSA Conference, hosted by the Stats Department at The University of Auckland. One of the conference themes is to celebrate the accomplishments of Ross Ihaka, who got R started back in 1992, along with Robert Gentleman. My talk included advice on setting up your R life to maximize effectiveness and reduce frustration.

Two specific slides generated much discussion and consternation in #rstats Twitter:

> *If the first line of your R script is*
>
> `setwd("C:\Users\jenny\path\that\only\I\have")`
>
> *I will come into your office and SET YOUR COMPUTER ON FIRE 🔥.*
>
> *If the first line of your R script is*
>
> `rm(list = ls())`
>
> *I will come into your office and SET YOUR COMPUTER ON FIRE 🔥.*

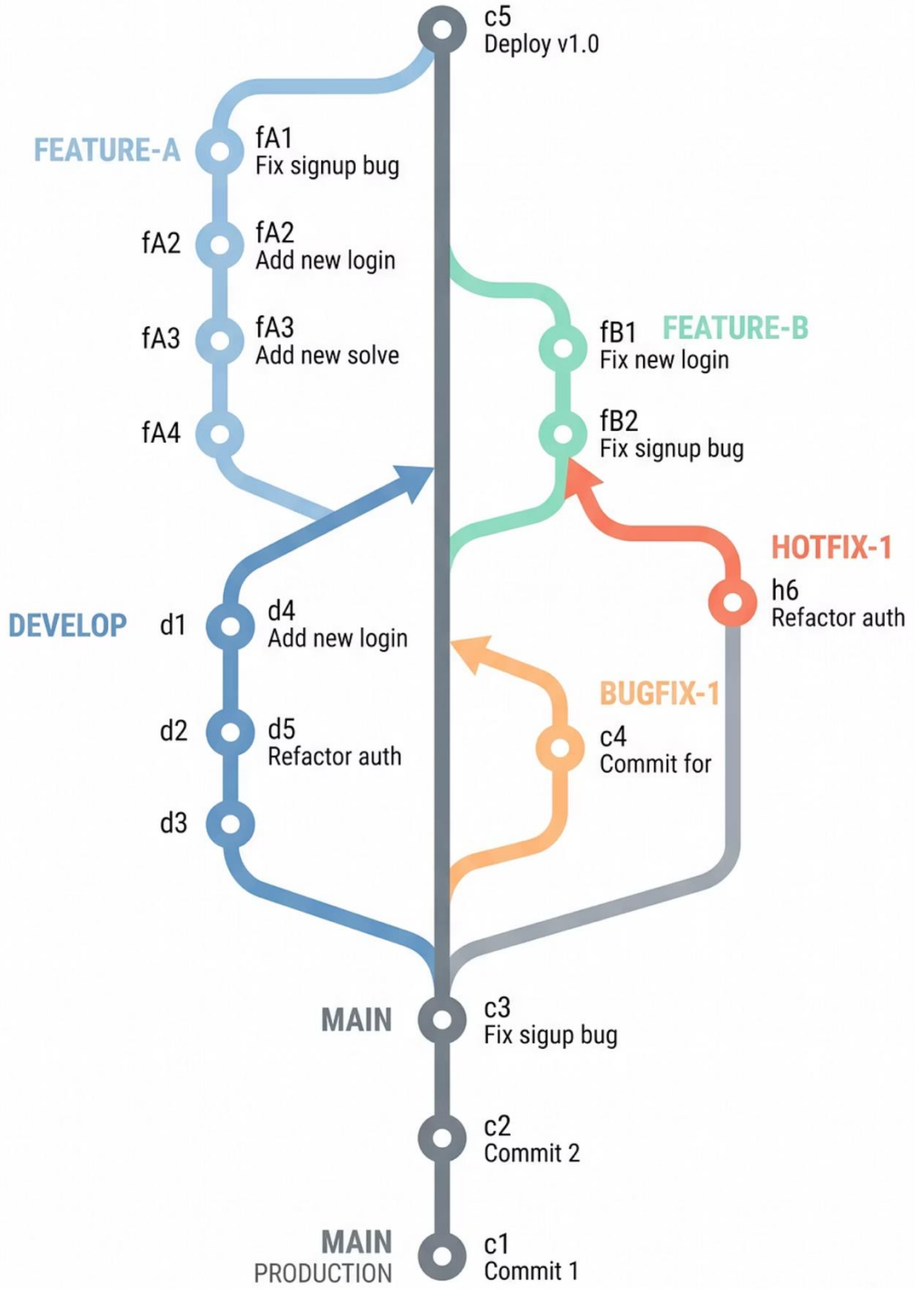
Absolute paths are the #1 reason code fails on a collaborator's machine. The `here()` package in R constructs paths relative to the project root — automatically, on any OS.

- R: `library(here); read_csv(here("data", "samples.csv"))`
- Python: `pip install pyhere`

Use R Projects in RStudio to anchor the root. One project = one working directory.

PILLAR 3: VERSION CONTROL

Git: Version Control for Code



PILLAR 3: VERSION CONTROL

Git Is Your Time Machine for Code

Recover Anything

Broke something? Roll back to any previous commit instantly. No more "analysis_FINAL_v7_USE_THIS.R."

Code Review

Pull requests let collaborators catch bugs before they propagate into results. We all make mistakes.

Fearless Experimentation

Create a branch, try a new approach, discard it if it fails — without touching your working code.

PILLAR 3: VERSION CONTROL

Six Git Commands Cover 90% of Your Needs

`git init`

Start tracking a project

`git clone`

Copy a remote repo

`git add`

Stage your changes

`git commit`

Save a snapshot

`git push`

Upload to GitHub

`git pull`

Sync from remote

The Golden Rule

Commit often. Push by the end of the day. Think of commits as checkpoints in a video game — the more frequently you save, the less you lose when something goes wrong.

Start with happygitwithr.com and learngitbranching.js.org — both are excellent, free, and interactive.

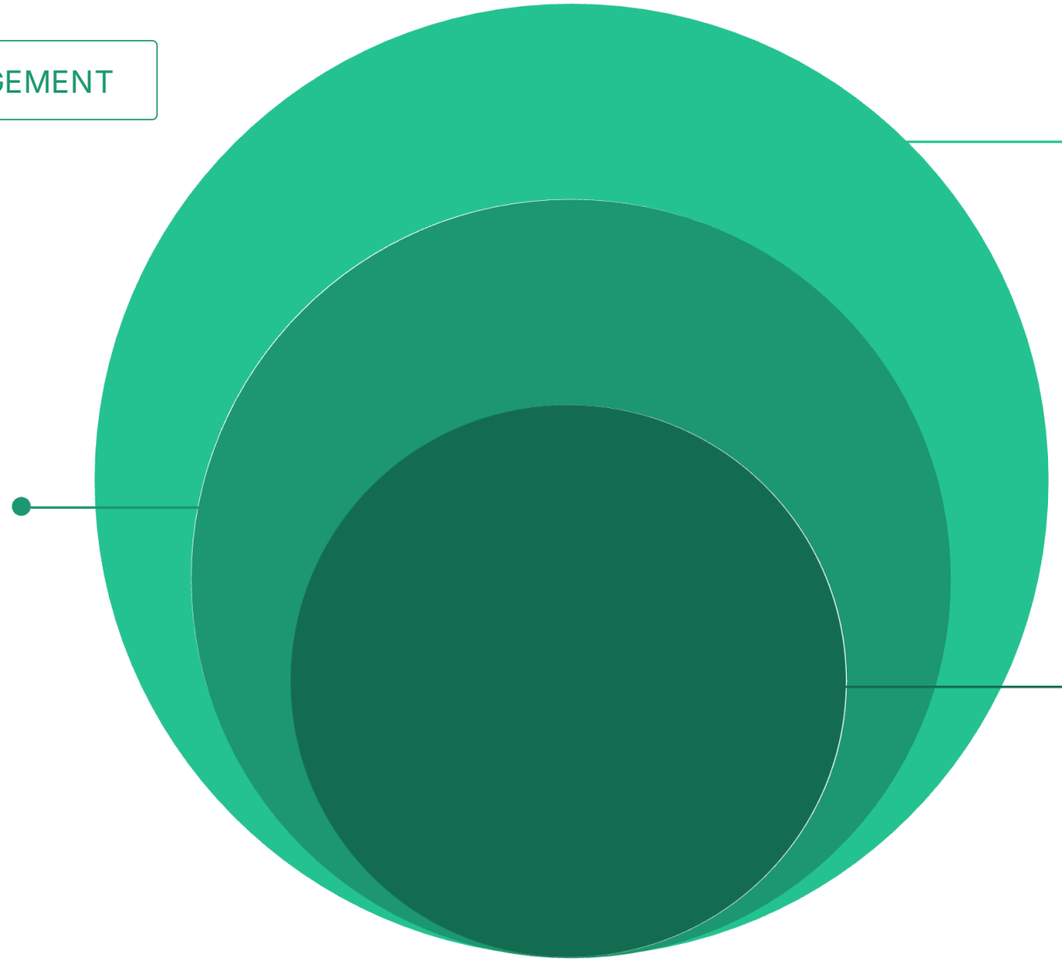
PILLAR 4: ENVIRONMENT MANAGEMENT

Managing Your Computing Environment

Same Code, Different Machine → Different Results?

A collaborator runs your exact R script on their machine and gets different clustering results. Why? A different version of Seurat. Or a different UMAP random seed. Environment drift is silent and deadly.

PILLAR 4: ENVIRONMENT MANAGEMENT



OS Container

Encapsulate system libs with Docker/Singularity

Full Stack

Use Biocontainers registry for end-to-end reproducibility

Package Layer

Pin R/Python packages via renv/uv/conda

Lock every layer — packages, system libraries, and OS — to guarantee identical results across machines and time.

PILLAR 4: ENVIRONMENT MANAGEMENT

Conda/Mamba: Reproducible Python & Bioinformatics Environments

Conda creates isolated environments with pinned tool versions. Share the `environment.yml` and anyone can recreate your exact setup.

- **Mamba**: drop-in conda replacement, dramatically faster solves
- **Pixi**: modern, lock-file-first alternative for reproducible conda envs
- Always export: `conda env export > environment.yml`

PILLAR 4: ENVIRONMENT MANAGEMENT

uv for Python · renv for R

uv (Python)

Blazing-fast Python package manager from Astral. Replaces pip + venv. Generates a `uv.lock` file — commit it to Git and anyone can restore your exact environment with one command.

renv (R)

Creates a private, project-local R library. `renv::init()` to start, `renv::snapshot()` to record, `renv::restore()` to recreate. The `renv.lock` file is the source of truth.

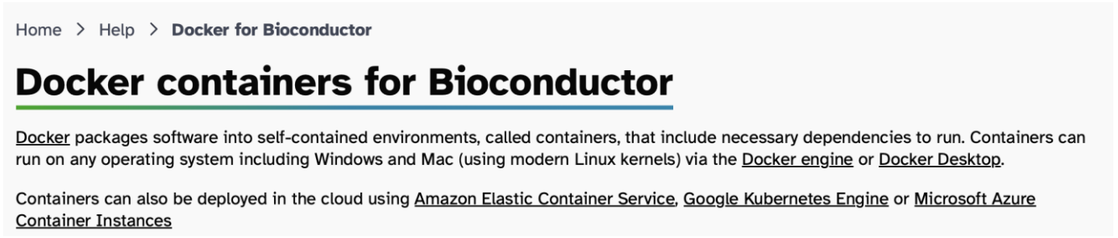
Commit your lock files (`renv.lock`, `uv.lock`) to Git. This is the single highest-leverage habit for environment reproducibility.

PILLAR 4: ENVIRONMENT MANAGEMENT

Docker: Freeze Your Entire Computing Environment

The Rocker Project

Docker Containers for the R Environment



Home > Help > **Docker for Bioconductor**

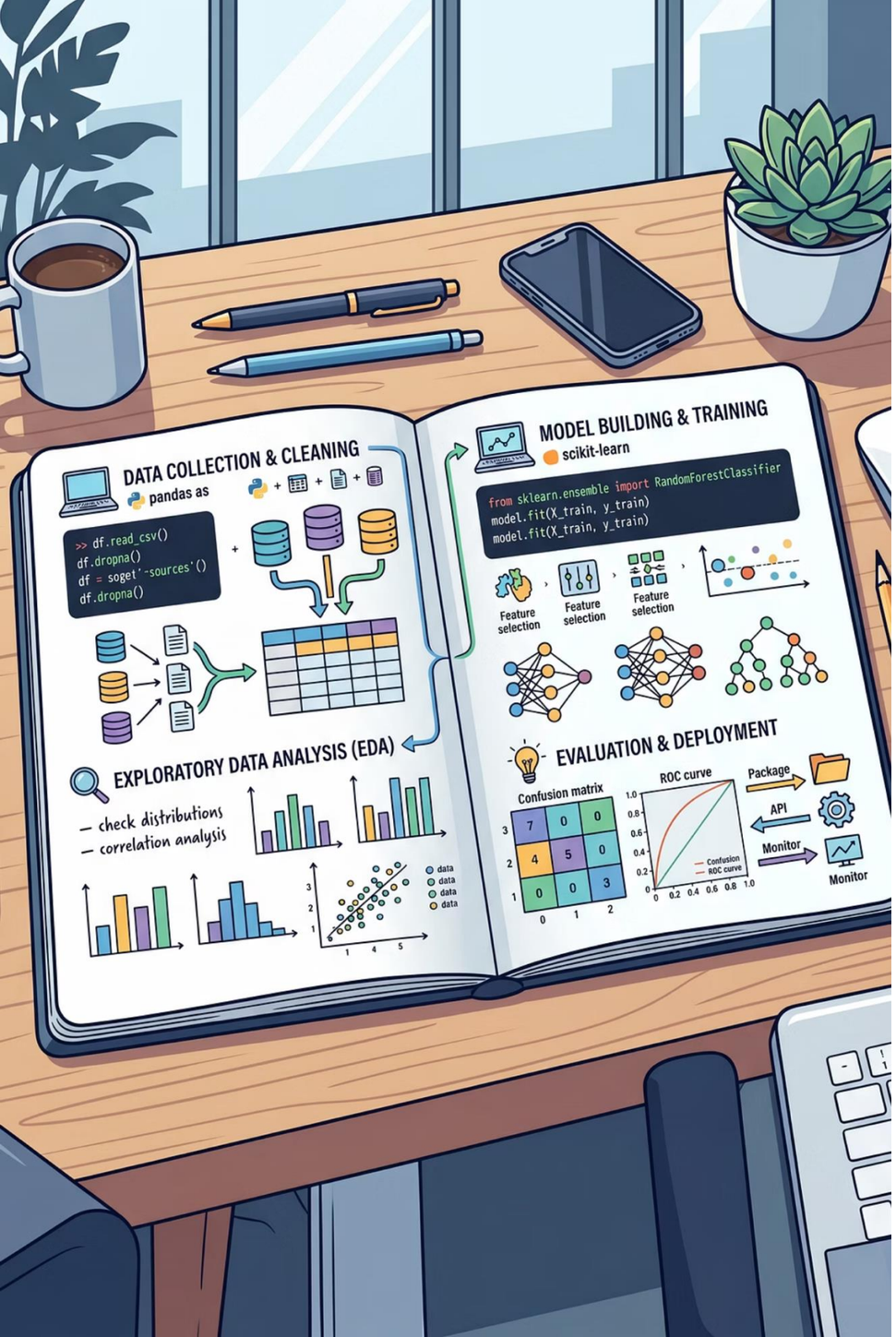
Docker containers for Bioconductor

Docker packages software into self-contained environments, called containers, that include necessary dependencies to run. Containers can run on any operating system including Windows and Mac (using modern Linux kernels) via the Docker engine or Docker Desktop.

Containers can also be deployed in the cloud using Amazon Elastic Container Service, Google Kubernetes Engine or Microsoft Azure Container Instances

Docker packages your code, all dependencies, system libraries, and OS into a single portable image. When you share the image, you share the *exact* environment — not just a list of packages.

Think of it as a sealed virtual machine that runs identically on any hardware. For HPC environments, use **Singularity/Apptainer** instead. Pre-built bioinformatics containers: biocontainers.pro and rocker-project.org.



PILLAR 5: LITERATE PROGRAMMING& AUTOMATION

Literate Programming & Automation

Code Without Context Is an Artifact, Not Science

PILLAR 5: LITERATE PROGRAMMING& AUTOMATION

Mix Code With Prose — Make Your Analysis Self-Explaining

Jupyter Notebook

The standard for Python workflows. Interleave code, output, and narrative. Note: not natively Git-friendly — consider **Marimo** as a reactive alternative.

R Markdown

Weave R (and Python via Reticulate) with prose to produce HTML, PDF, or Word reports. Knitr renders everything reproducibly.

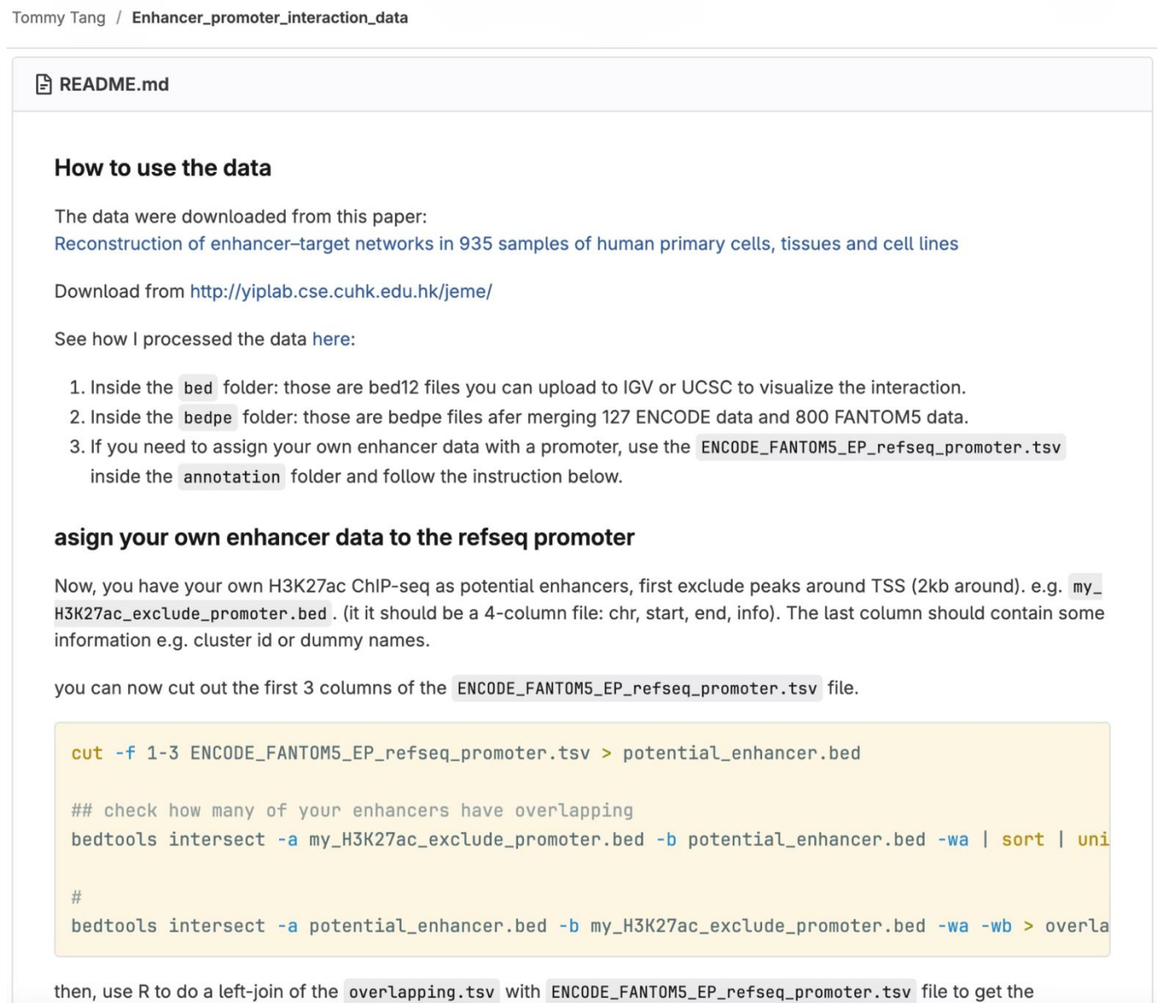
Quarto

The next generation of R Markdown. Supports R, Python, Julia, and Observable. One framework for articles, slides, websites, and books.

The goal: anyone should be able to open your notebook and re-run it top-to-bottom to get identical results.

PILLAR 5: LITERATE PROGRAMMING& AUTOMATION

Write a README. Then Write a Better One.



Tommy Tang / **Enhancer_promoter_interaction_data**

README.md

How to use the data

The data were downloaded from this paper:
Reconstruction of enhancer–target networks in 935 samples of human primary cells, tissues and cell lines

Download from http://yiplab.cse.cuhk.edu.hk/jeme/

See how I processed the data here:

1. Inside the `bed` folder: those are bed12 files you can upload to IGV or UCSC to visualize the interaction.
2. Inside the `bedpe` folder: those are bedpe files afer merging 127 ENCODE data and 800 FANTOM5 data.
3. If you need to assign your own enhancer data with a promoter, use the `ENCODE_FANTOM5_EP_refseq_promoter.tsv` inside the `annotation` folder and follow the instruction below.

asign your own enhancer data to the refseq promoter

Now, you have your own H3K27ac ChIP-seq as potential enhancers, first exclude peaks around TSS (2kb around). e.g. `my_H3K27ac_exclude_promoter.bed`. (it it should be a 4-column file: chr, start, end, info). The last column should contain some information e.g. cluster id or dummy names.

you can now cut out the first 3 columns of the `ENCODE_FANTOM5_EP_refseq_promoter.tsv` file.

```
cut -f 1-3 ENCODE_FANTOM5_EP_refseq_promoter.tsv > potential_enhancer.bed

## check how many of your enhancers have overlapping
bedtools intersect -a my_H3K27ac_exclude_promoter.bed -b potential_enhancer.bed -wa | sort | uni

#
bedtools intersect -a potential_enhancer.bed -b my_H3K27ac_exclude_promoter.bed -wa -wb > overla
```

then, use R to do a left-join of the `overlapping.tsv` with `ENCODE_FANTOM5_EP_refseq_promoter.tsv` file to get the

Good documentation outside the code includes:

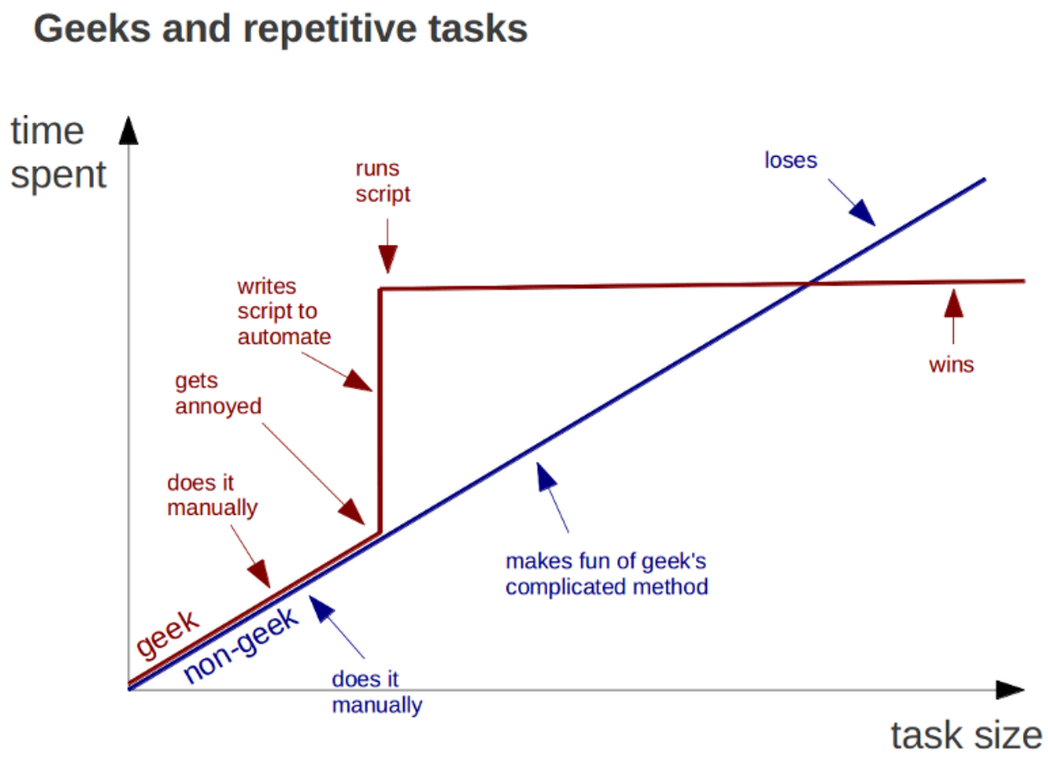
- A project-level `README.md` with context, data sources, and how to run the analysis
- A per-analysis Quarto or Rmd file with narrative explaining *why*, not just *what*
- A changelog or lab notebook entry for each major analysis update

Use Claude Code or ChatGPT to draft README files from your code. Then review and refine. AI-assisted documentation is still better than no documentation.

https://gitlab.com/tangming2005/Enhancer_promoter_interaction_data

PILLAR 5: LITERATE PROGRAMMING& AUTOMATION

Automate Everything You'll Do More Than three times



> The best documentation is automation. — Twitter-verse wisdom

If you ran it manually three times, Write a script, and future-you gets it for free every time after.

PILLAR 5: LITERATE PROGRAMMING& AUTOMATION

Workflow Managers: Beyond Shell Scripts

1

Shell Scripts

Simple repetitive tasks. `my_routine.sh` as a starting point.

2

GNU Make / 

`{targets}` in R for dependency-aware pipelines. Reruns only what changed.

3

Production-grade workflow managers for HPC and cloud. Scales to thousands of samples.

Pick the level of complexity that matches your use case. Snakemake and Nextflow have large bioinformatics ecosystems and community-maintained pipelines (nf-core).

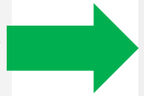
PILLAR 6: CLEAN CODE

Clean Code & Functional Programming

PILLAR 6: CLEAN CODE

Don't Repeat Yourself — Wrap It in a Function

```
ht_list2 =
  partition_hp2 +
  EnrichedHeatmap(mat5, pos_line = FALSE,  column_title = "ER", row_title_rot = 0, name = "ER",
                  col= col_fun5,
                  top_annotation = HeatmapAnnotation(enriched = anno_enriched(gp = gpar(col = 2:4), ylim=c(0, 1))),
                  show_row_dend = FALSE) +
  EnrichedHeatmap(mat6, pos_line = FALSE,  column_title = "ER Capi", row_title_rot = 0, name = "ER Capi",
                  col= col_fun6,
                  top_annotation = HeatmapAnnotation(enriched = anno_enriched(gp = gpar(col = 2:4), ylim=c(0, 1))),
                  show_row_dend = FALSE) +
  EnrichedHeatmap(mat7, pos_line = FALSE,  column_title = "ER Ful", row_title_rot = 0, name = "ER Ful",
                  col= col_fun7,
                  top_annotation = HeatmapAnnotation(enriched = anno_enriched(gp = gpar(col = 2:4), ylim=c(0, 1))),
                  show_row_dend = FALSE) +
  EnrichedHeatmap(mat8, pos_line = FALSE,  column_title = "ER CF", row_title_rot = 0, name = "ER CF",
                  col= col_fun8,
                  top_annotation = HeatmapAnnotation(enriched = anno_enriched(gp = gpar(col = 2:4), ylim=c(0, 1))),
                  show_row_dend = FALSE) +
  EnrichedHeatmap(mat1, pos_line = FALSE,  column_title = "ATAC", row_title_rot = 0, name = "ATAC",
                  col= col_fun1,
                  top_annotation = HeatmapAnnotation(enriched = anno_enriched(gp = gpar(col = 2:4), ylim=c(0, 2.5))),
                  show_row_dend = FALSE) +
  EnrichedHeatmap(mat2, pos_line = FALSE,  column_title = "ATAC Capi", row_title_rot = 0, name = "ATAC Capi",
                  col= col_fun2,
                  top_annotation = HeatmapAnnotation(enriched = anno_enriched(gp = gpar(col = 2:4), ylim=c(0, 2.5))),
                  show_row_dend = FALSE) +
  EnrichedHeatmap(mat3, pos_line = FALSE,  column_title = "ATAC Ful", row_title_rot = 0, name = "ATAC Ful",
                  col= col_fun3,
                  top_annotation = HeatmapAnnotation(enriched = anno_enriched(gp = gpar(col = 2:4), ylim=c(0, 2.5))),
                  show_row_dend = FALSE) +
  EnrichedHeatmap(mat4, pos_line = FALSE,  column_title = "ATAC CF", row_title_rot = 0, name = "ATAC CF",
                  col= col_fun4,
                  top_annotation = HeatmapAnnotation(enriched = anno_enriched(gp = gpar(col = 2:4), ylim=c(0, 2.5))),
                  show_row_dend = FALSE)
```



```
```{r}
make_single_hp<- function(mat, sample_name, color_map, top_y_limit, kmeans_color){
  hp<- EnrichedHeatmap(mat, pos_line = FALSE,
                       column_title = sample_name,
                       row_title_rot = 0,
                       name = sample_name,
                       col= color_map,
                       top_annotation = HeatmapAnnotation(enriched = anno_enriched(gp = gpar(col = kmeans_color),
                                                                                   ylim= top_y_limit)),
                       show_row_dend = FALSE)
  return(hp)
}


H3K4me1_hps<- purrr::map2(H3K4me1_mats, names(H3K4me1_mats),
                          ~make_single_hp(mat = .x, sample_name = .y,
                                          color_map = col_fun2,
                                          top_y_limit = c(0, 0.6),
                                          kmeans_color = 2:4))

KMT2D_hps<- purrr::map2(KMT2D_mats, names(KMT2D_mats),
                        ~make_single_hp(mat = .x, sample_name = .y,
                                        color_map = col_fun1,
                                        top_y_limit = c(0, 0.8),
                                        kmeans_color = 2:4))

ht_list<-  purrr::reduce(H3K4me1_hps, `+`) + purrr::reduce(KMT2D_hps, `+`)
```

The Rule of Three

If you copy-paste a block of code three times, it's time to write a function.

Copy-paste code is fragile. If your logic changes, you must update every copy. A function means you fix it once, everywhere.
```

PILLAR 6: CLEAN CODE

Use `purrr::map()` to Replace Loops

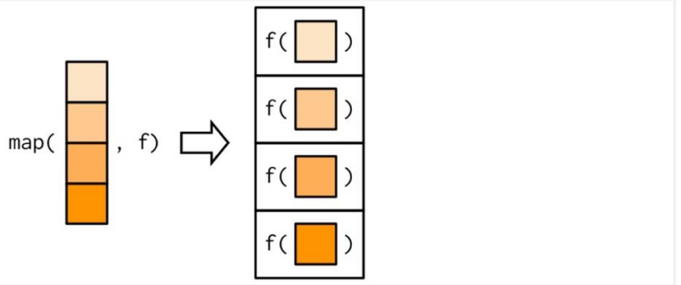
9.2 My first functional: `map()`

The most fundamental functional is `purrr::map()` [53]. It takes a vector and a function, calls the function once for each element of the vector, and returns the results in a list. In other words, `map(1:3, f)` is equivalent to `list(f(1), f(2), f(3))`.

```
triple <- function(x) x * 3
map(1:3, triple)
#> [[1]]
#> [1] 3
#>
#> [[2]]
#> [1] 6
#>
#> [[3]]
#> [1] 9
```

Copy

Or, graphically:



- Use `purrr::map()` in R to apply functions across lists — no more copy-paste loops
- Python equivalent: `map()`, list comprehensions, or `functools`
- Document with Roxygen2 in R so your functions are self-explaining

PILLAR 6: CLEAN CODE

Use AI to Refactor Your Code — Not to Write It Blind

```r
# Function to align a query dataset to a reference dataset using CCA and perform label transfer
align_cca <- function(reference_data, query_data, reference_labels, k_cca = 20, k_mnn = 10) {
  # Normalize and scale
  centered_reference <- t(scale(t(reference_data), center = TRUE, scale = TRUE))
  centered_query <- t(scale(t(query_data), center = TRUE, scale = TRUE))

  # Covariance matrix
  sigma_xy <- (1 / (min(ncol(centered_query), ncol(centered_reference)) - 1)) *
              t(centered_query) %*% centered_reference

  # Perform SVD
  cca_svd <- irlba::irlba(sigma_xy, nu = k_cca, nv = k_cca)

  # Get canonical variates
  canonical_variates_query <- l2_normalize(cca_svd$u)  # Query canonical variates
  canonical_variates_reference <- l2_normalize(cca_svd$v)  # Reference canonical variates

  # Set row names to sample names and column names based on selected dimensions
  rownames(canonical_variates_query) <- colnames(query_data)
  colnames(canonical_variates_query) <- paste0("CCA", seq_len(k_cca))

  rownames(canonical_variates_reference) <- colnames(reference_data)
  colnames(canonical_variates_reference) <- paste0("CCA", seq_len(k_cca))

  # Find Nearest Neighbors using RANN
  nn_indices_reference <- nn2(data = canonical_variates_reference, query = canonical_variates_query, k =
k_mnn)$nn.idx
  nn_indices_query <- nn2(data = canonical_variates_query, query = canonical_variates_reference, k =
k_mnn)$nn.idx

  # Identify Mutual Nearest Neighbors
  mnn_indices <- find_mnn(nn_indices_reference, nn_indices_query, nrow(canonical_variates_query))

  # Perform Label Transfer
  label_result <- label_transfer(reference_labels, mnn_indices, nn_indices_reference)

  # Return the results
```

AI tools like Claude Code and ChatGPT can dramatically accelerate code quality — when used correctly:

- Paste messy, repetitive code → ask for a functional refactor
- Ask for unit tests to lock in expected behavior
- Request documentation strings for every function
- Use AI to scaffold R or Python packages from your existing scripts

⚠ Always review AI-generated code. AI doesn't know your biological context — you do.

PILLAR 6: CLEAN CODE

Package Your Functions for Maximum Reuse and Reproducibility

R Package

Roxygen2 documentation + devtools. Wrap project-specific functions into a versioned, installable package.

Python Package

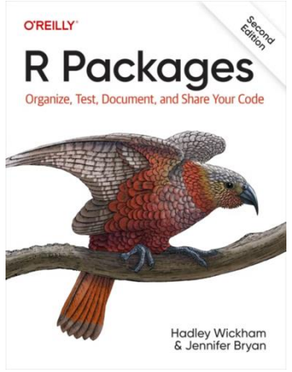
Use Claude Code to scaffold a `pyproject.toml` package from your scripts. Instantly installable via pip or uv.

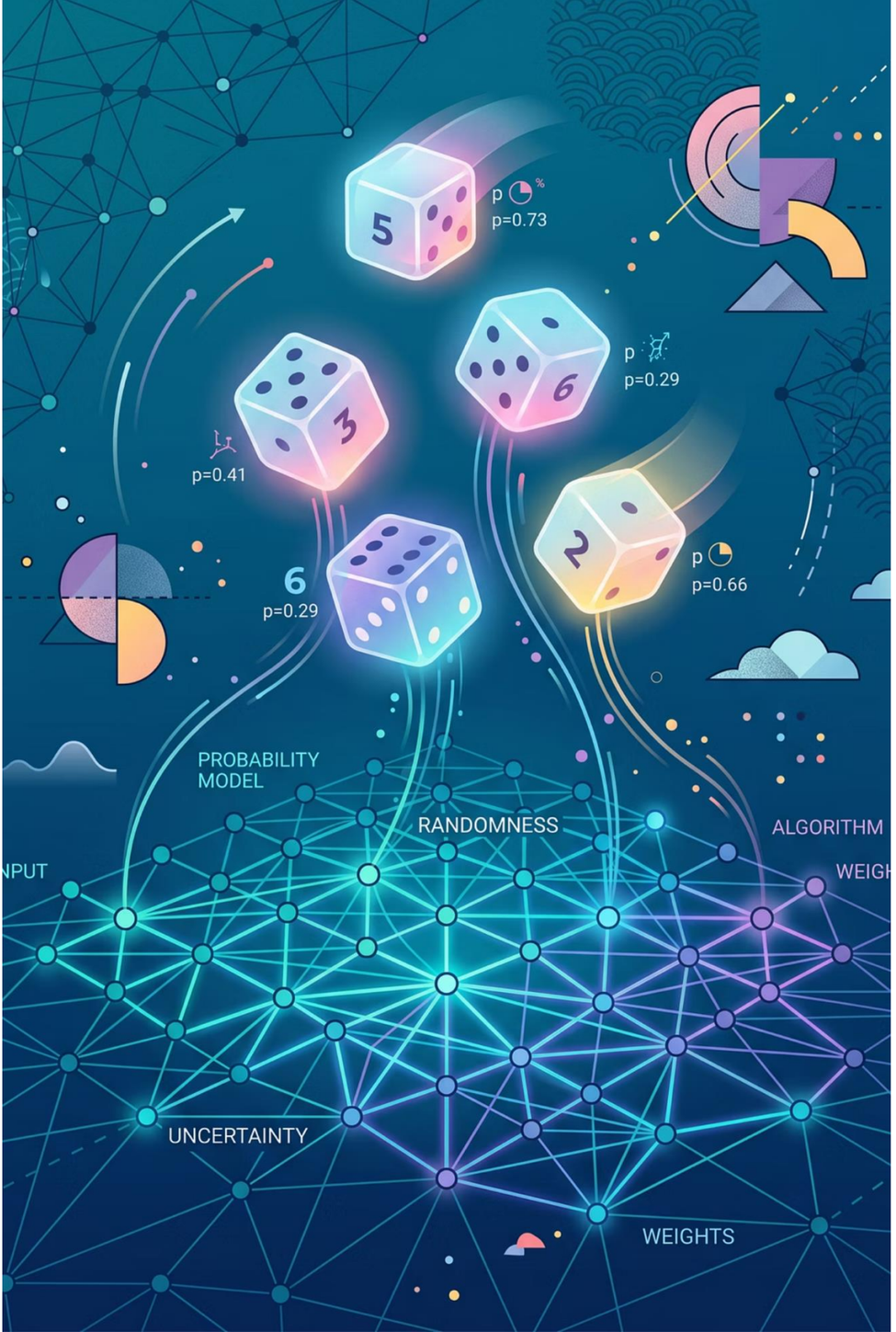
A package is the most reproducible form of code. It forces you to document inputs and outputs, write tests, and version your logic explicitly. Once packaged, anyone can install and reuse your exact analysis functions.

Start with r-pkgs.org — Hadley Wickham's free, comprehensive guide.

https://github.com/crazyhottommy/BETA2

https://github.com/crazyhottommy/ROSE2





AI IN THE WORKFLOW

AI in Bioinformatics Is Probabilistic — Here's What That Means for Reproducibility

AI IN THE WORKFLOW

How Tommy Uses Claude Code in Practice

Load Context

Gather papers, slides, old Rmd

Claude Writes

Plan and generate Rmd scaffold

Write Question

Define the biological question clearly

Knit Report

Render HTML from Quarto/Rmd

The key: Claude writes the *scaffold*, you provide the *context and judgment*. The code lives in Git. The environment is locked with renv. The report is a Quarto template. **The human stays in the loop.**

AI IN THE WORKFLOW

AI Reports Drift — Even When the Prompt is Identical

If you use an LLM to generate an analysis report, running the same prompt twice produces *different* figures, different thresholds, different section order. Why?

The model re-decides everything

Which plots (volcano vs MA?), which thresholds (padj < 0.05 or 0.01?), which genes to label, which colors — all filled in "creatively" each run.

Temperature ≠ the only source of variance

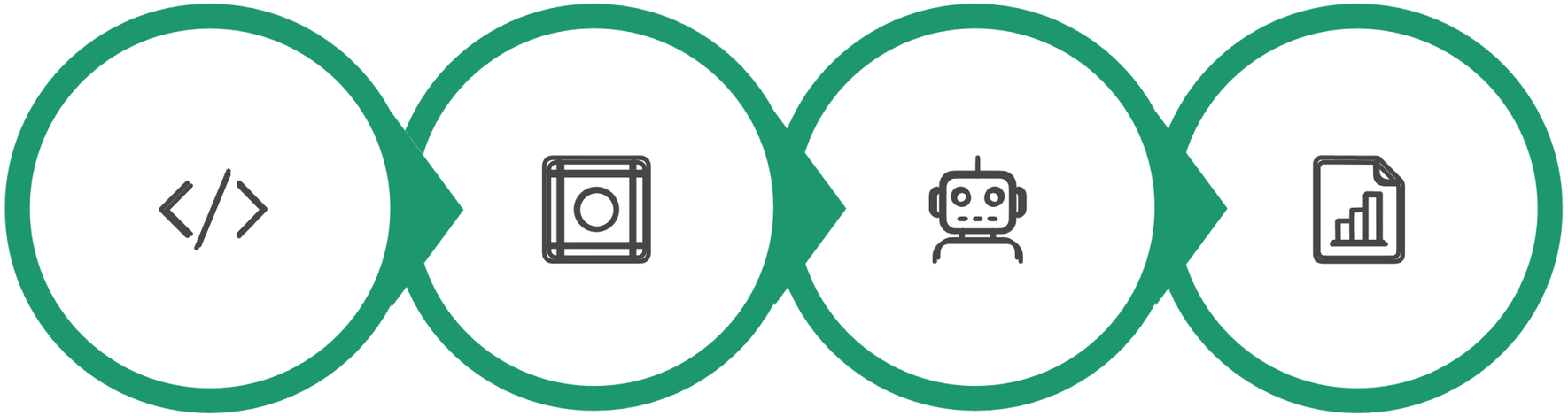
Even at temperature 0, the prompt under-specifies the analysis. The model fills gaps differently every time.

Version drift

The model itself is updated over time. The same prompt in GPT-4o today vs six months from now may yield a different analysis structure.

AI IN THE WORKFLOW

The Fix: LLM as Orchestrator, Not Analyst



Deterministic Scripts

Quarto Template

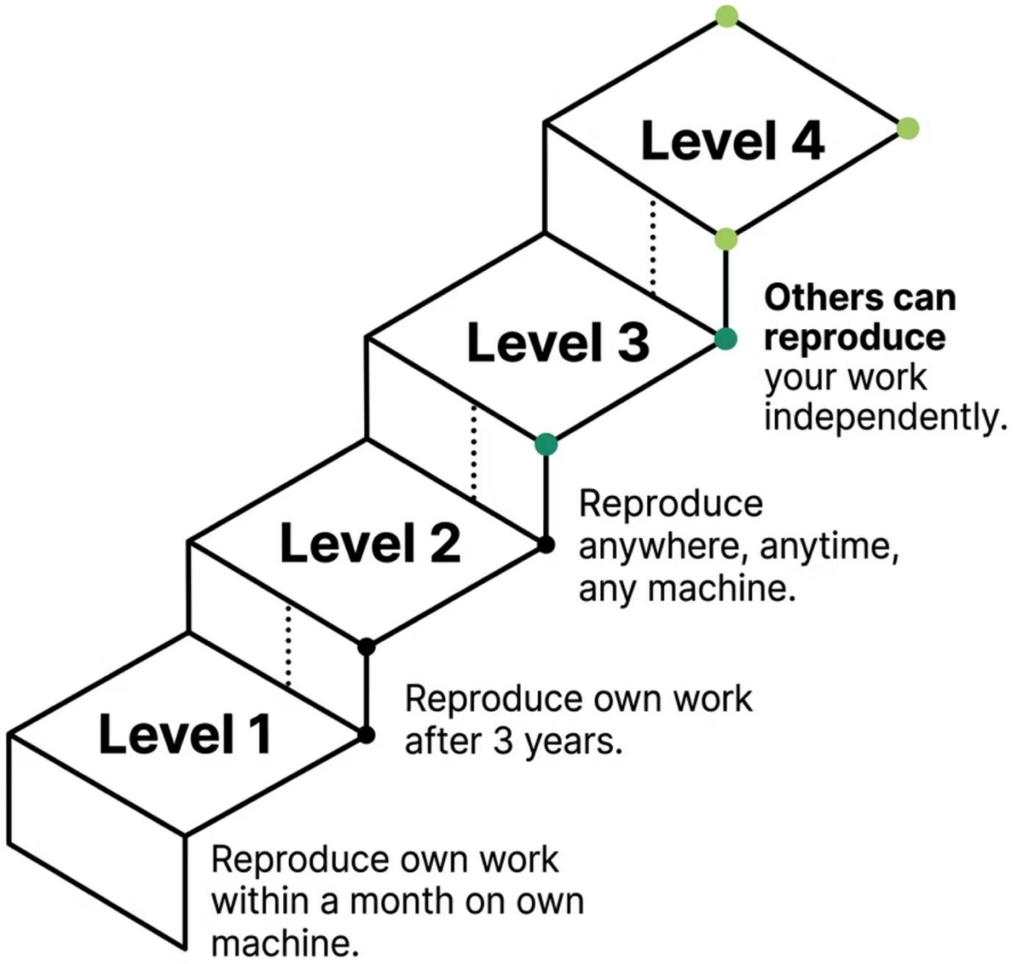
LLM Orchestrates

Reproducible Report

The principle: Move all analytical decisions into pinned, versioned scripts with fixed random seeds (`set.seed(42)`, `random_state=42`). The LLM's only job is to pick inputs, call your scripts, and assemble the report into pre-defined template slots.

Recommended structure: `scripts/` (deterministic, testable) + `templates/report.qmd` (fixed layout with slots) + locked package environment.

The Reproducibility Spectrum: Where Are You?



Most bioinformaticians live at Level 1 or 2. The practices in this talk move you toward Level 3 and 4 — which is where clinical translation and collaborative science require you to be.

You don't need to achieve Level 4 overnight. Each practice you adopt moves you up one step and dramatically reduces risk.

The Good Enough Practices Checklist

data

Versioning of data, follow best practices for spreadsheets

project organization

ISO 8601 dates, no spaces, machine + human readable. Zero-pad numbers. Consistent folder structures.

Git

Version control all code. Commit often. Push daily. Code review via PRs.

Environments

renv for R, uv for Python, Conda/Mamba/pixi for tools. Commit lock files. Use Docker containers

Notebooks and Automate

Quarto or Rmd mixing code + narrative. Render to HTML and share the link.

Clean Code

Write functions and packages. Use functional programming

Data Analysis We Can Believe In

Reproducibility isn't a bureaucratic checkbox. It's how we earn the right to make decisions that affect patients, shape policy, and advance human health.

> Start with one practice today. Your collaborators — and your future self — will thank you.

Blog

divingintogeneticsandgenomics.com

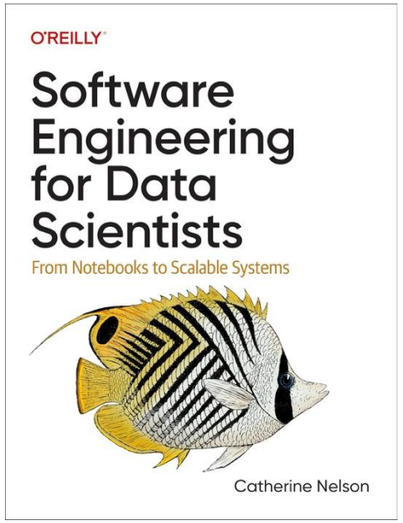
YouTube

Chatomics

X / Twitter

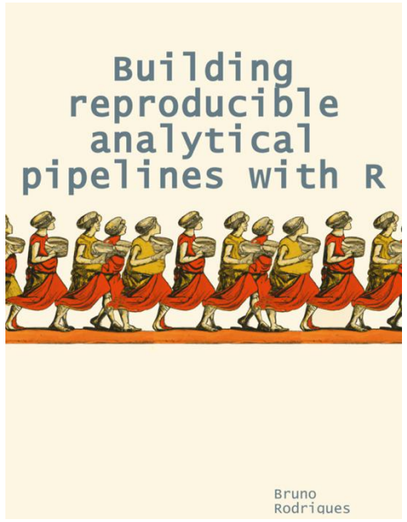
@tangming2005

Further Reading to Go Deeper



Unit test

CI/CD



https://raps-with-r.dev