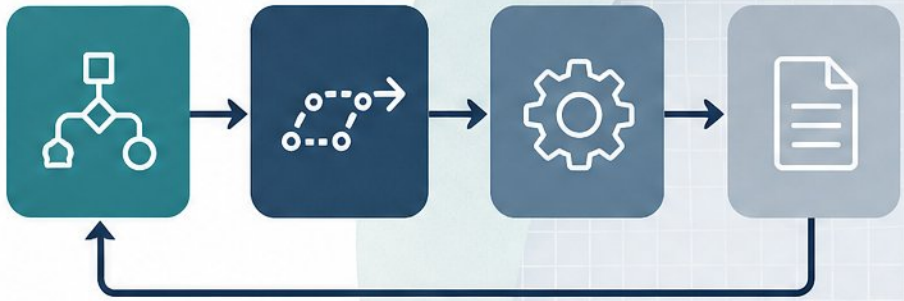
Statistics Software: Selection, Requirements, and Workflow

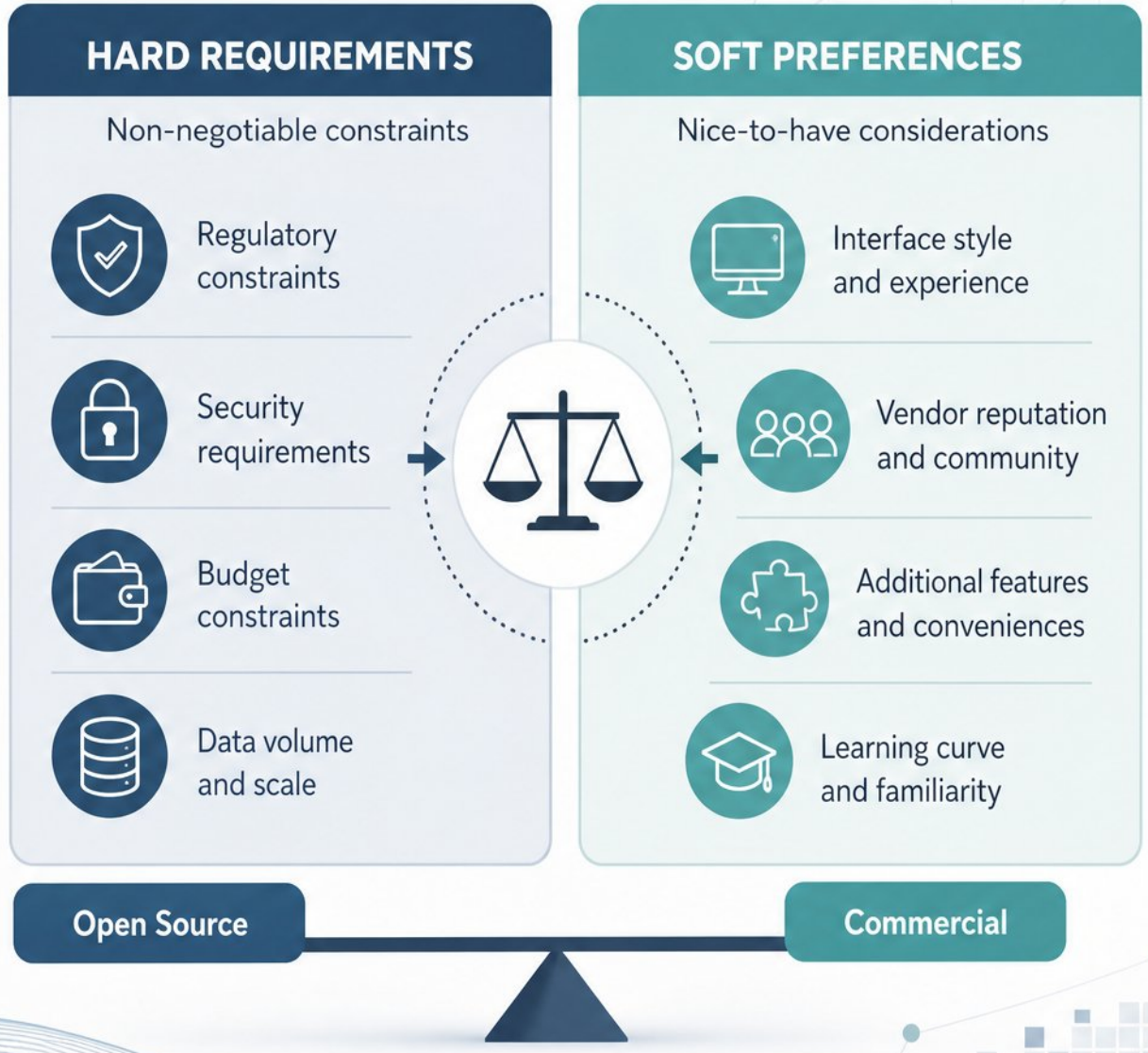
- **Scope:** choosing statistical software, defining requirements, building reproducible workflows
- **Audience:** analysts, researchers, educators, students, data teams, organizations
- **Core promise:** a decision framework linking software choice to data, collaboration, governance
- **Roadmap:** requirements, landscape comparison, evaluation, workflow design, validation, adoption, metrics
- **Outcome:** defensible software decisions and reproducible reporting



Start with Requirements, Not Feature Lists

Statistical Software Selection: Why Requirements Come First

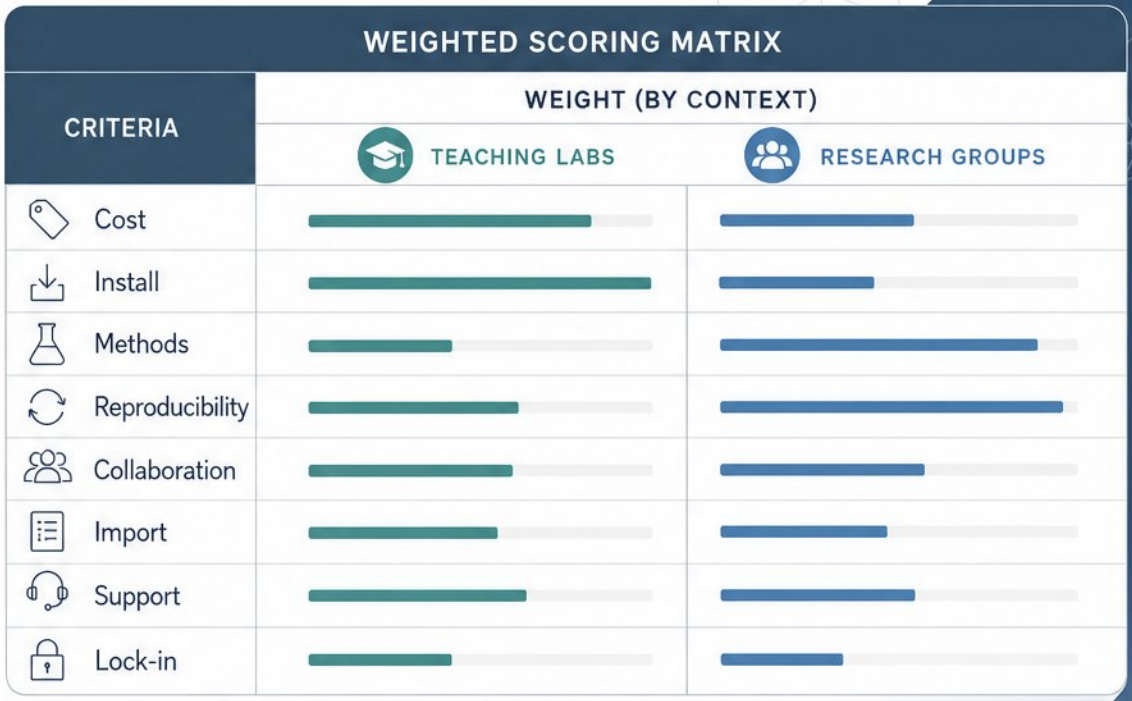
- Separate hard requirements — regulatory, security, budget, data volume — from soft preferences.
- Map needs to roles: occasional analyst, programmer, methodologist, educator, student, decision-maker.
- Define non-functional needs: privacy, reproducibility, auditability, performance, licensing, support, interoperability.
- Use a weighted scoring matrix; document trade-offs instead of comparing feature checklists.
- Per FDA guidance, software must be reliable and fully documented, including version and build.



A Decision Framework for Software Selection

- Eight-criteria checklist: cost, install, methods, reproducibility, collaboration, import, support, lock-in
- Weight criteria by context — teaching labs favor cost and install; research groups favor methods and reproducibility
- Model three-year total cost of ownership, not sticker price
- Include add-on modules, IT maintenance, and retraining after licensing changes
- Document rationale with fixed weights across finalists to defend the decision

WEIGHTED SCORING MATRIX

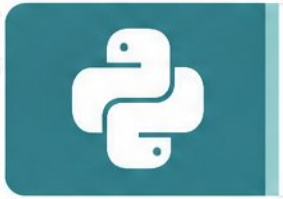


THREE-YEAR TOTAL COST OF OWNERSHIP

- Include add-on modules
- IT maintenance
- Retraining after licensing changes
- Model three-year total cost of ownership, not sticker price

Landscape Overview: General-Purpose Statistical Environments

- **R:** free, open-source, broadest statistical coverage; ggplot2 graphics; steep learning curve



- **Python:** free, open-source; best when analysis connects to automation, ML, or production systems

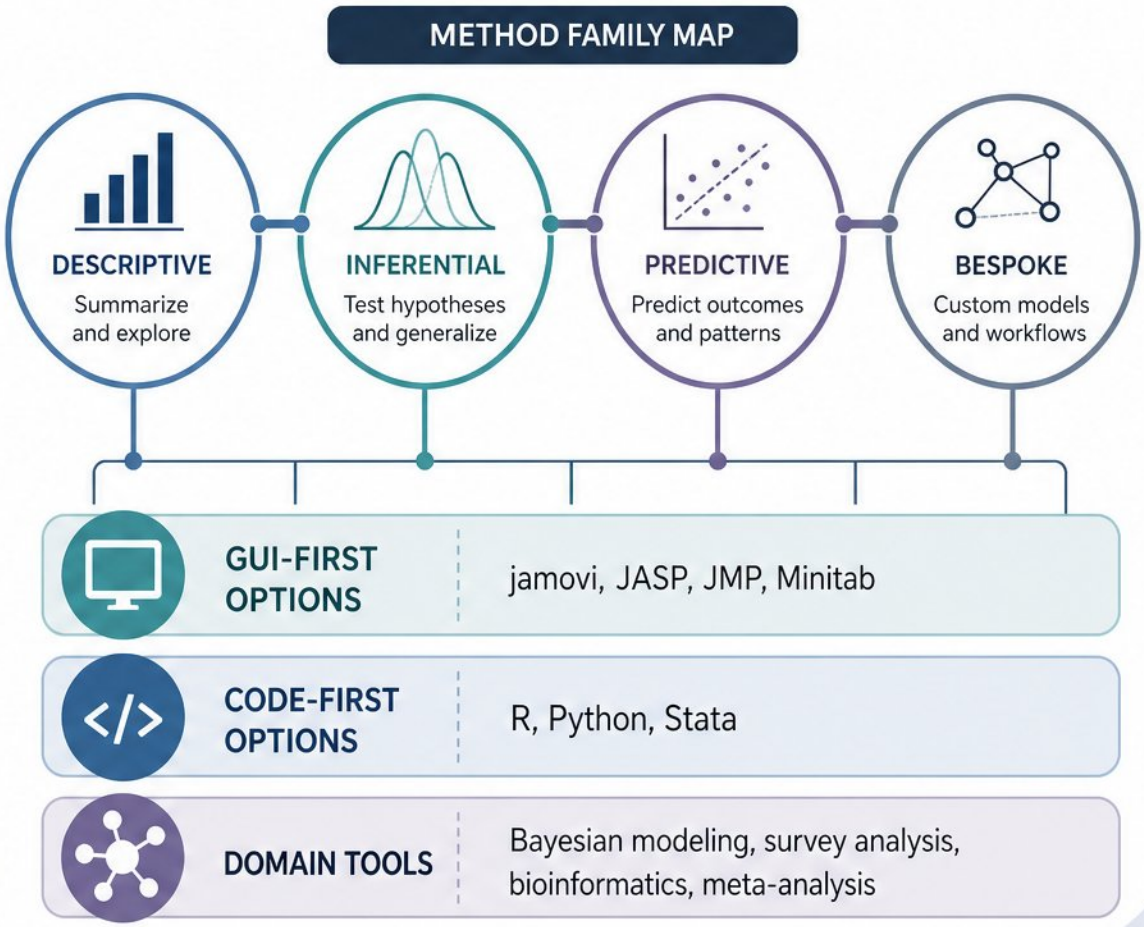
- **SAS:** enterprise platform with validated procedures, audit trails, and vendor support for regulated work

- **SPSS:** menu-driven interface for survey and social-science analysis; paid add-on modules for advanced methods

- **Stata:** command-driven environment for econometrics, panel data, survival analysis, and reproducible applied research

Specialized and GUI-First Tools by Analysis Need

- Match the tool to the method family: descriptive, inferential, predictive, or bespoke modeling.
- GUI-first options lower the barrier: jamovi, JASP, JMP, Minitab.
- Code-first options allow custom methods and automation: R, Python, Stata.
- Domain tools cover Bayesian modeling, survey analysis, bioinformatics, meta-analysis.
- Follow field conventions to stay compatible with collaborators, reviewers, and regulators.



Open Source vs. Commercial Trade-offs

Open Source

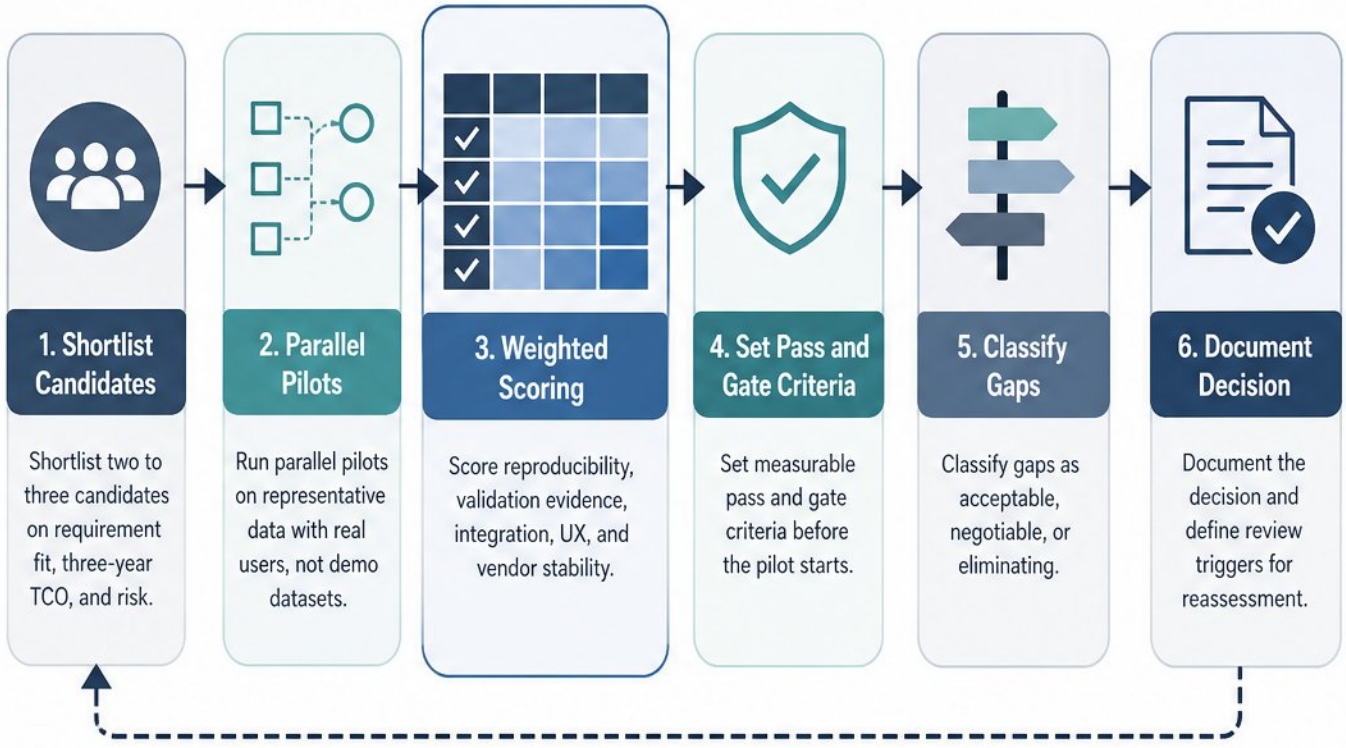
- Free licenses: R, Python, JASP, jamovi
- "Free" covers licensing only — budget training, compute, engineering time
- Open source relies on community support and internal validation
- Talent pool, extensibility, and maintenance costs differ sharply
- Most teams run two tools: one open-source, one licensed

Commercial

- Paid licenses: SPSS, SAS, Stata, Minitab, JMP, GraphPad Prism
- Commercial vendors offer SLAs and validated procedures
- Open source relies on community support and internal validation
- Talent pool, extensibility, and maintenance costs differ sharply
- Most teams run two tools: one open-source, one licensed

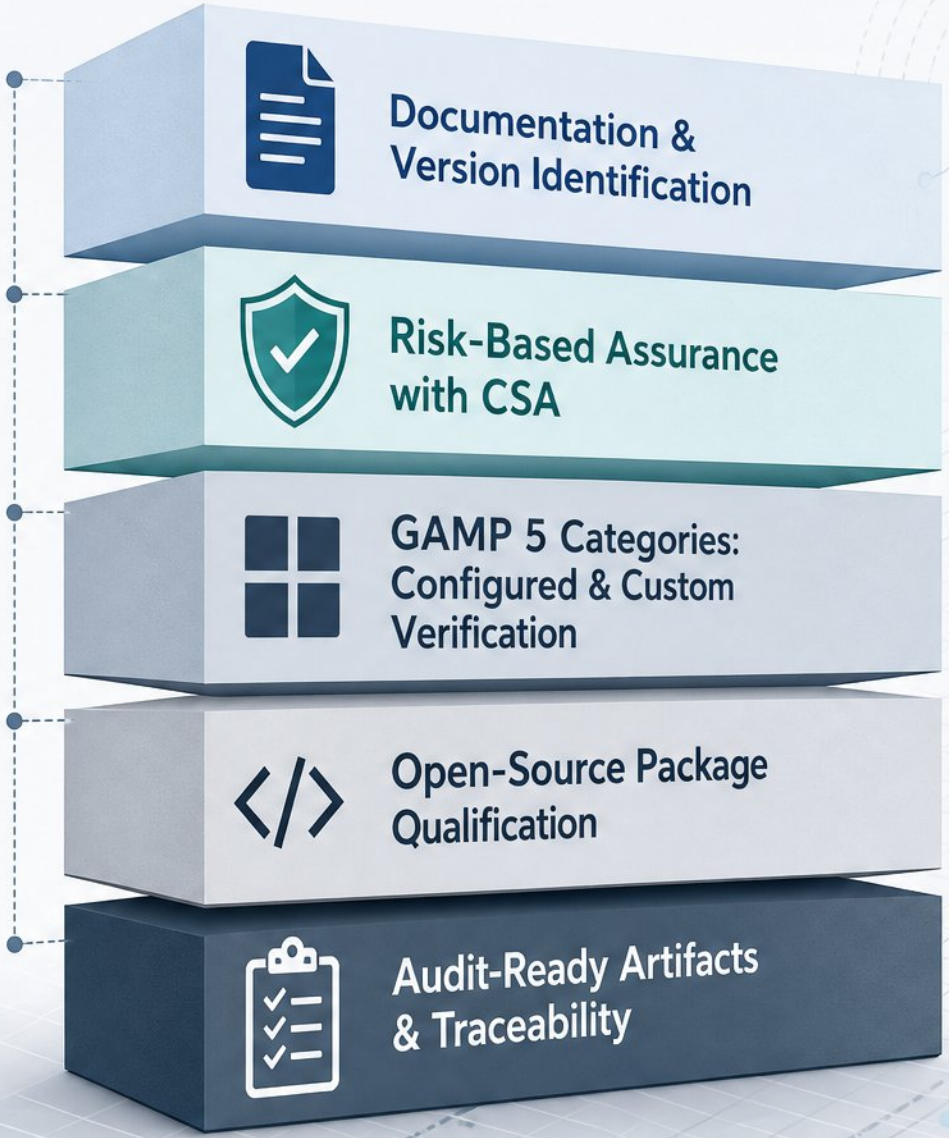
Running a Structured Evaluation and Pilot

- - Shortlist two to three candidates on requirement fit, three-year TCO, and risk
- - Run parallel pilots on representative data with real users, not demo datasets
- - Score reproducibility, validation evidence, integration, UX, and vendor stability
- - Set measurable pass and gate criteria before the pilot starts
- - Classify gaps as acceptable, negotiable, or eliminating
- - Document the decision and define review triggers for reassessment

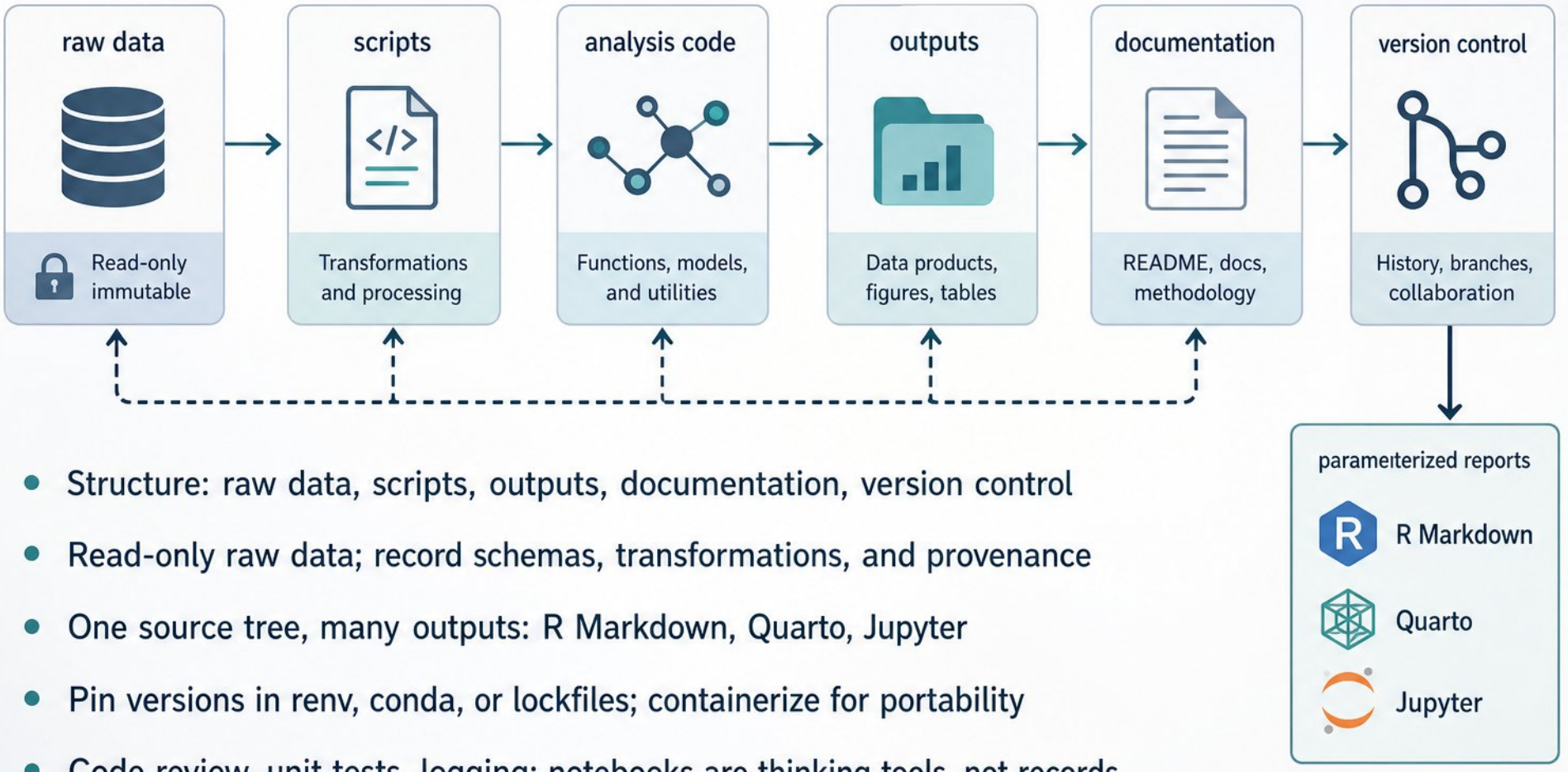


Validation and Regulatory Considerations

- FDA requires no specific software; document version and build for every analysis
- CSA (final Sept 2025, updated Feb 2026) replaces exhaustive CSV with risk-based assurance
- GAMP 5 categories: configured products and custom apps need rigorous verification
- Qualify open-source packages by repository checks, maintenance, usage, and testing evidence
- Auditors expect traceability matrices, validation summary reports, and change-control records



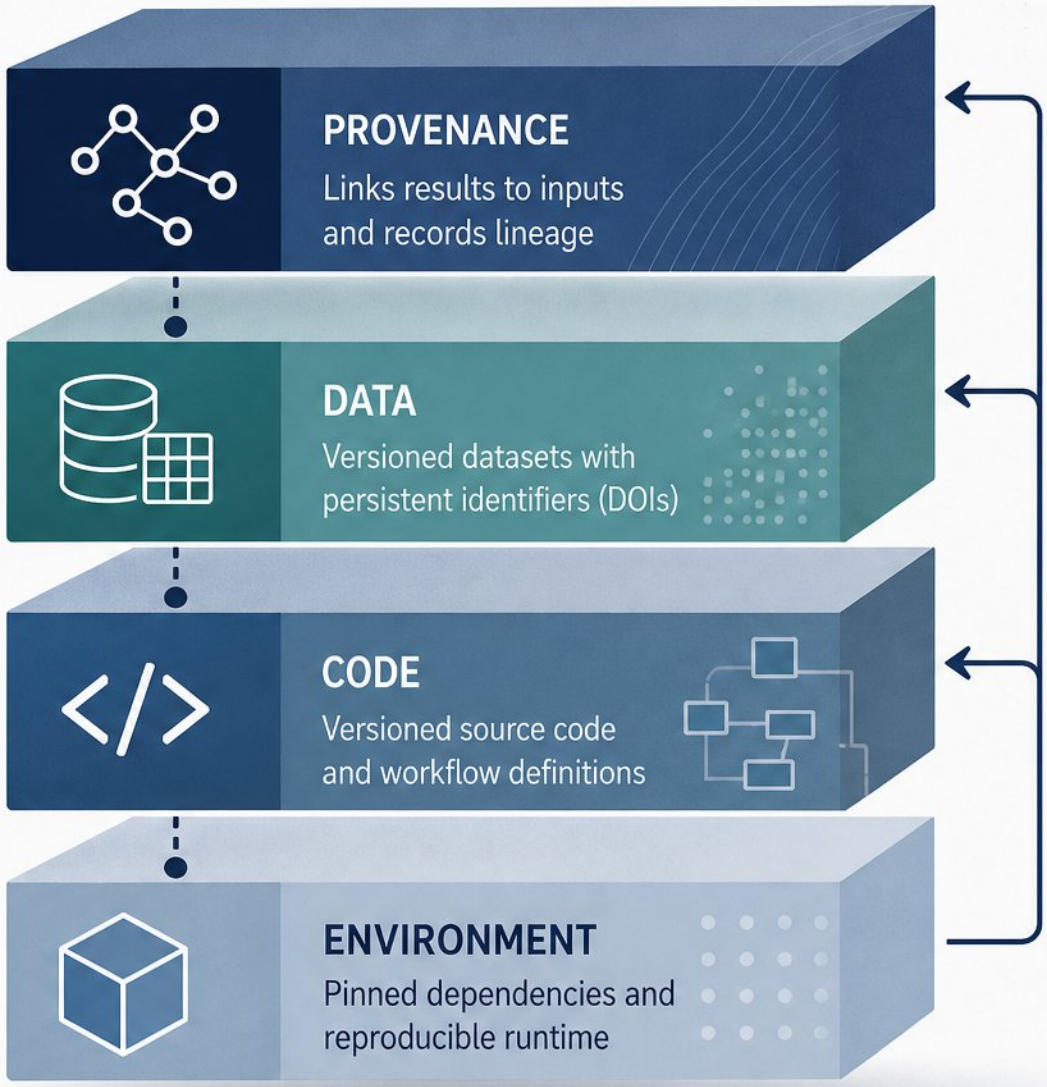
Workflow Pattern: Project Structure and Reproducible Reporting



- Structure: raw data, scripts, outputs, documentation, version control
- Read-only raw data; record schemas, transformations, and provenance
- One source tree, many outputs: R Markdown, Quarto, Jupyter
- Pin versions in renv, conda, or lockfiles; containerize for portability
- Code review, unit tests, logging; notebooks are thinking tools, not records

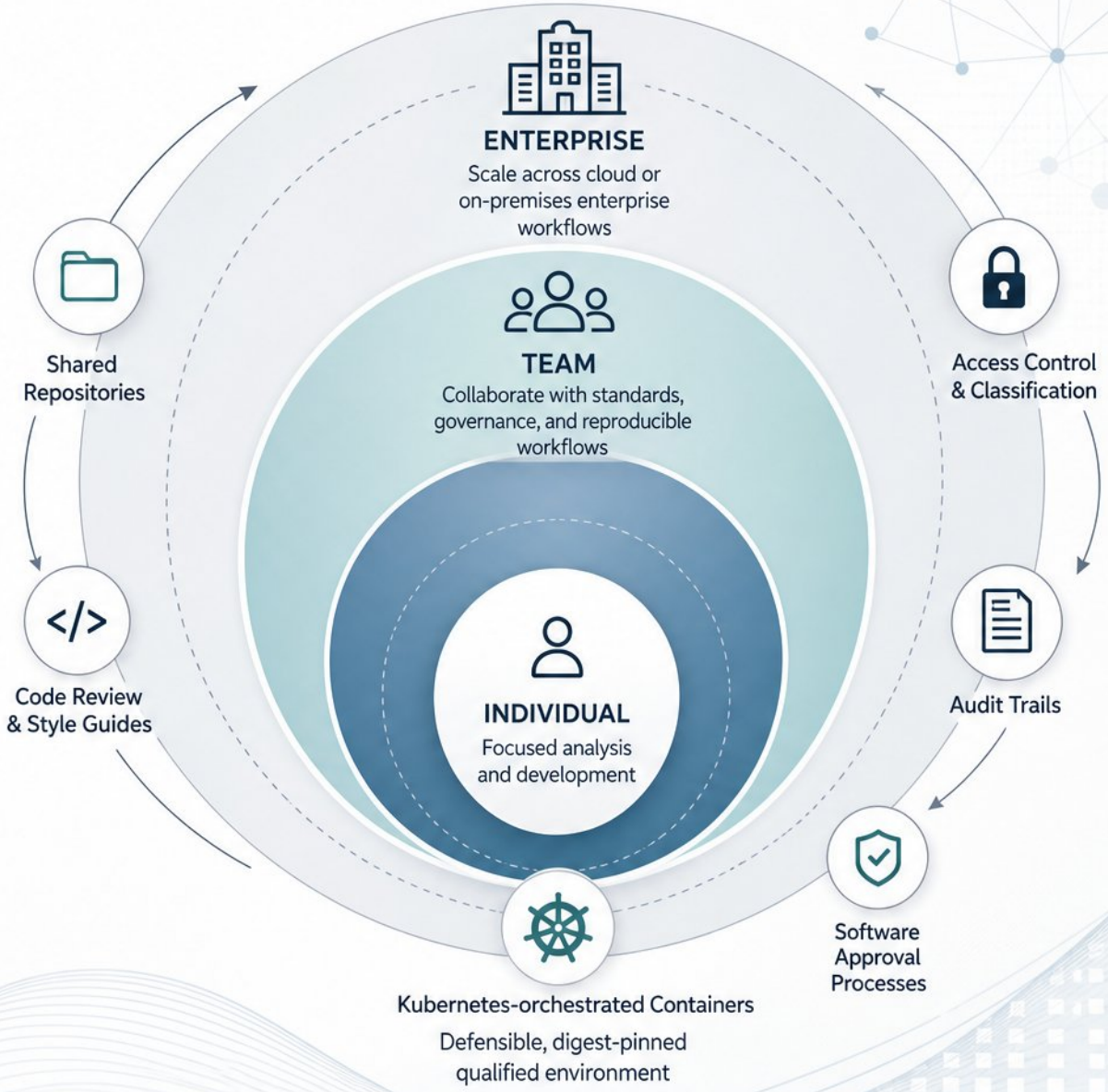
Reproducibility Infrastructure: Environment, Data, and Provenance

- Four layers: environment, code, data, and provenance tying results to inputs
- Pin exact versions in a lockfile, or ship a container image pinned by digest
- A mutable tag such as *latest* is not a reproducibility guarantee
- Assign persistent identifiers (DOIs) so the exact dataset version stays citable
- Workflow managers express pipelines as dependency graphs; one command regenerates
- Keep hyperparameters in versioned config files; fix and record random seeds



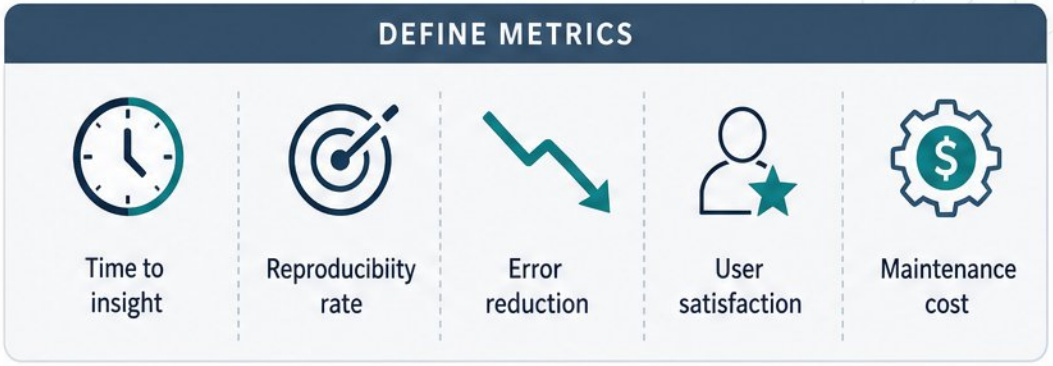
Collaboration, Governance, and Scaling

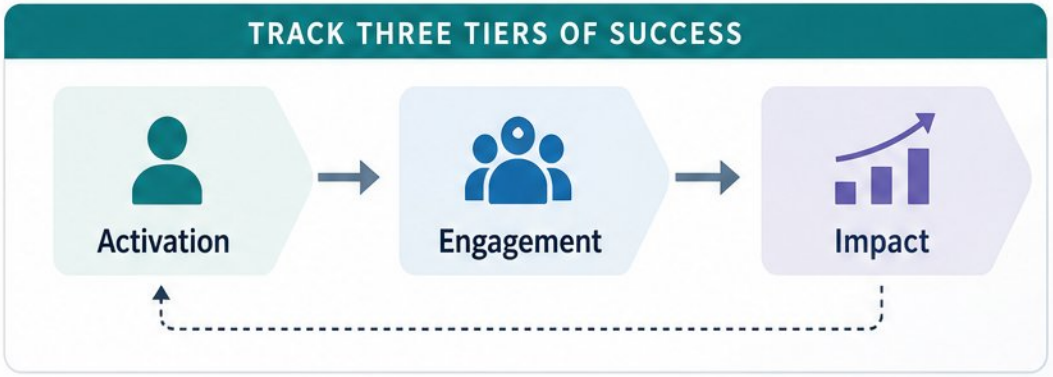
- Shared repositories, code review, style guides, and reproducible handoffs
- Access control, data classification, audit trails, software approval processes
- Scale from individual analysis to cloud or on-premises enterprise workflows
- Kubernetes-orchestrated containers give regulated teams a defensible, digest-pinned environment
- Training and support plans reduce tool fragmentation and build sustainable skills

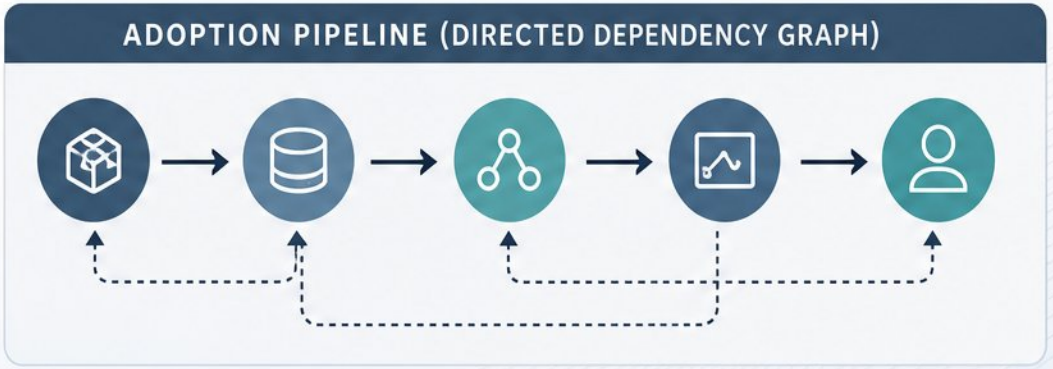


Adoption Planning and Measuring Success

- Build an adoption plan with owners, milestones, and training
- Define metrics: time to insight, reproducibility rate, error reduction
- Track three tiers: activation, engagement, impact — not logins alone
- Review the software portfolio as requirements and ecosystems shift
- Avoid over-customization, ignored total cost, and poor workflow fit

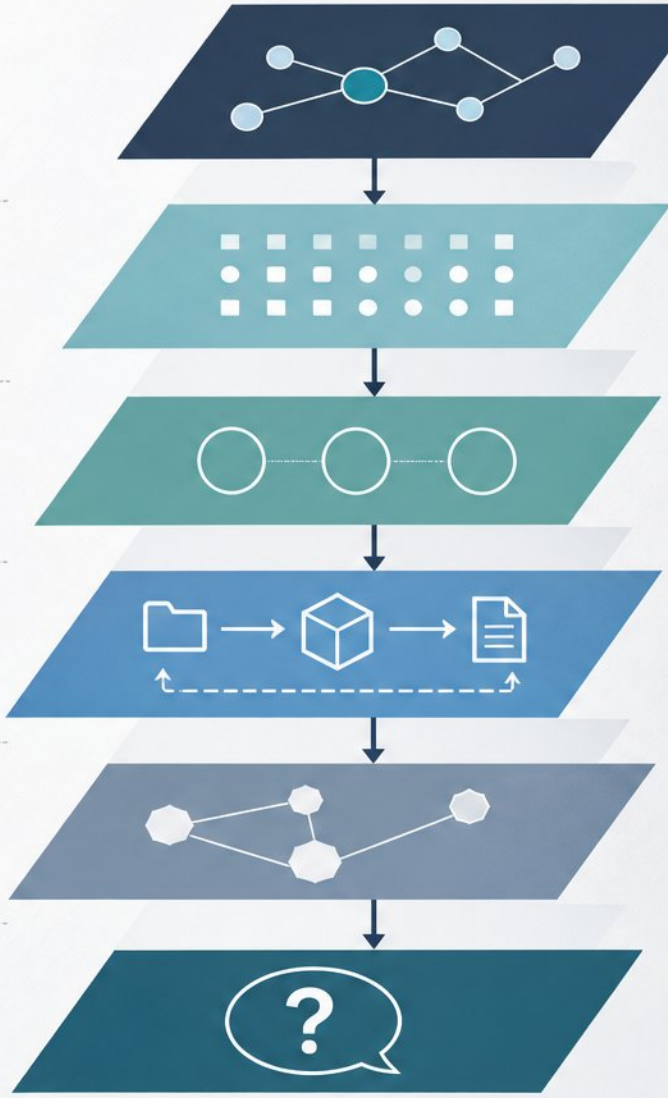






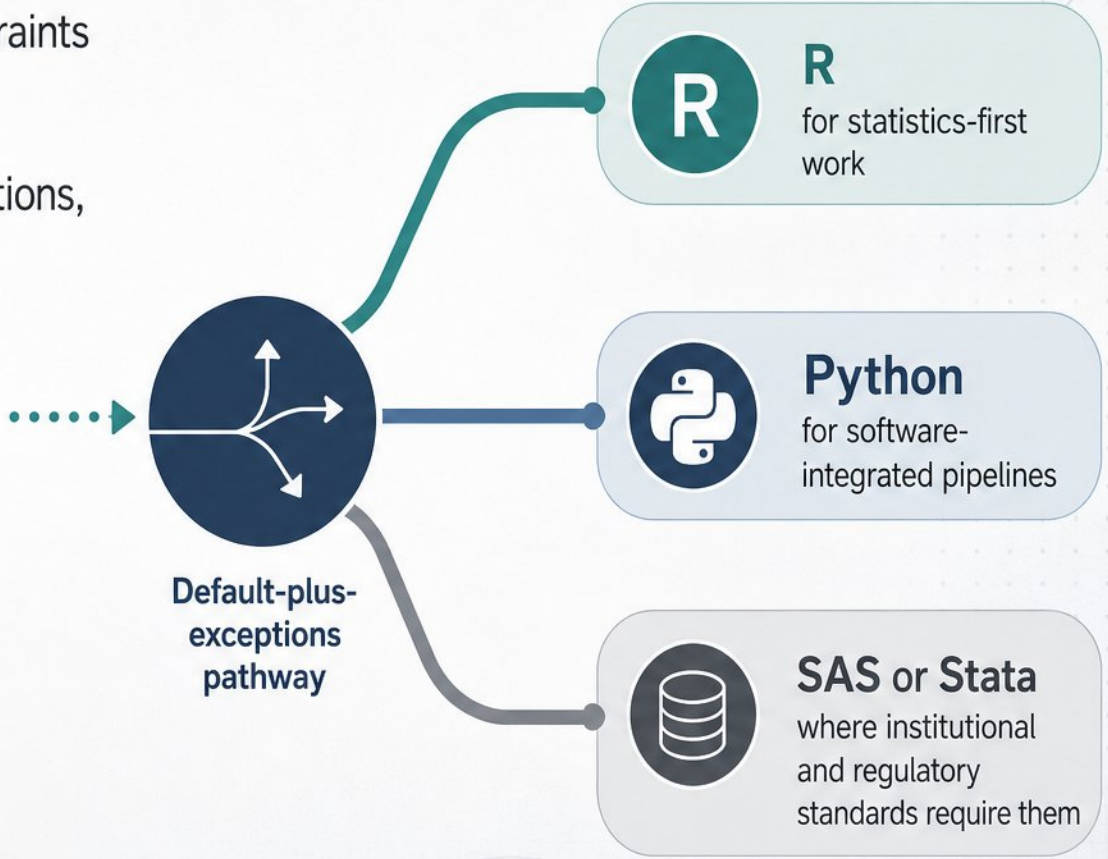
Practical Exercise: Requirements to Workflow

- Write 5–10 hard and soft requirements for the sample scenario
- Score criteria 1–5, keep weights fixed across finalists
- Shortlist two or three tools that pass the gate criteria
- Sketch a reproducible workflow: project structure, environment, reporting
- Compare trade-offs across cost, reproducibility, collaboration, governance
- Debrief: which requirement would change the software decision?



Key Takeaways and Next Steps

- Requirements come first: hard constraints and user roles drive the shortlist
- Start with one default and two exceptions, matched to your context
- R for statistics-first work; Python for software-integrated pipelines
- SAS or Stata where institutional and regulatory standards require them
- Build reproducibility from the first experiment, not submission week
- Match validation effort to risk; define success metrics before rollout



PersonWise.

PersonWise • AI digital-human course platform

Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/babb7968/public