Data Science in Focus

Andrew Stewart

2025-04-12

Table of contents

Data Science In Focus

Introduction

Data Science In Focus is a reflective essay series on data science as a scientific discipline—what it is, how it's practiced, and what it has become. As the field matures beyond its early hype cycles and into a coherent form of applied research, this series aims to sharpen our collective understanding of the work itself.

Where a bountiful collection of earlier works have laid the groundwork for a newly forming field, these essays here revisit core questions with the benefit of hindsight:
- What does it mean to practice data science as science?
- How should teams, tools, and systems support inquiry over output?
- What kind of knowledge does data science produce—and for whom?

Rooted in the scientific method, structured around the research lifecycle, and steeped in the evolving norms of modern tech orgs, this series puts the discipline itself into focus.

Defining Data Science

What Is Data Science?

In the era of tech monoculture, the term *data science* has been stretched to near incoherence—absorbing everything from analytics engineering to AI research under its inflated halo. But if we strip away the branding and job title inflation, what remains is something much older, much simpler, and much more principled: **data science is the application of the scientific method to the study of data-generating systems.**

It is not a subfield of software engineering. It is not a synonym for machine learning. It is not a placeholder for "person who works with data." Data science, properly understood, is a **scientific discipline**—defined not by its tooling or domain, but by its epistemology. Its goal is to generate knowledge. Its process is experimental. Its currency is uncertainty. And its outputs are not products, but explanations.

A Discipline of Inquiry, Not Output

What distinguishes data science from engineering is not the data—it's the **orientation toward inquiry**.

- **Engineers** build systems that are designed to perform reliably and at scale.
- **Scientists** study systems to understand how and why they behave the way they do.

Data scientists may use engineering tools, work within engineering organizations, and produce artifacts that feed into engineering systems. But their foundational job is to ask—and answer—questions about system behavior. They design experiments, test hypotheses, analyze variation, and build explanatory models. In this sense, a data scientist is closer to a physicist studying turbulence than a developer deploying a feature.

This isn't a hierarchy. It's a **division of labor**—and misunderstanding it leads to broken workflows, misaligned expectations, and org charts that burn out good scientists by asking them to write production code full-time.

The Objects of Study

Data science is concerned with **data-generating processes**, especially those that arise within technological systems. Some examples:

- How does user engagement change in response to a new design?
- What latent behaviors drive churn in a subscription model?
- Why did model performance degrade last week?
- What features of a marketplace system produce price instability?

These are not engineering problems. They're **systems questions**. Answering them requires conceptual models, uncertainty quantification, domain awareness, and often a blend of statistical inference and simulation. They also often involve dead ends, ambiguous results, and theoretical exploration—things that are normal in science but foreign to many software workflows.

The Tools Are Not the Discipline

It's tempting to define data science by its stack: SQL, Python, pandas, Jupyter, etc. But that would be like defining chemistry by beakers and Bunsen burners. Tools enable the work—they aren't the work.

In fact, many of the tools used in data science are borrowed from engineering or software development. The difference is in **how they're used**. A data scientist doesn't write Python to deploy services; they use it to simulate a hypothesis, analyze system output, or validate statistical assumptions. SQL isn't a pipeline—it's a telescope.

Science Inside a Software Org

One of the greatest challenges facing data scientists today is that they are often the only scientists inside engineering organizations. That creates cultural friction. Deadlines prioritize shipping over understanding. Metrics are flattened into KPIs. Curiosity becomes a liability. Documentation is seen as overhead rather than intellectual scaffolding.

But despite these tensions, data science has a crucial role to play: it helps organizations **understand themselves**. It maps the terrain, exposes the mechanisms, and builds the mental models that engineering and product teams rely on to make informed decisions.

When practiced as science, data science becomes the **epistemic engine** of a tech company. It gives us confidence not just in what we're building, but in what we believe.

In Summary

- Data science is a **scientific discipline** rooted in the study of complex systems through data.

- Its central purpose is **explanation**, not output.
- Its methods are driven by **hypothesis, experimentation, and inference**.
- Its work supports and complements engineering by providing **clarity, context, and insight**.

By treating data science as science, we restore its rightful posture—an experimental partner to engineering, a conceptual partner to product, and a critical lens for understanding the systems we build and inhabit.

A Brief History of Data Science

The term *data science* has become a fixture in modern tech culture. It evokes a potent combination of mathematics, coding, and business impact. Yet for all its popularity, the origin and evolution of data science are often mischaracterized, reduced to buzzwords or simplistic analogies. To understand data science as a discipline, we must retrace its path—through statistics, computer science, and information theory—to see how it emerged and why it matters.

Origins in Statistics and Probability

The earliest roots of data science lie in the field of statistics, which arose from the needs of governance. The word "statistics" comes from the Latin *statista*, meaning "statesman," and its early usage revolved around collecting demographic and economic information for statecraft. Governments in 18th-century Europe tabulated population sizes, trade balances, and agricultural outputs—rudimentary analytics aimed at planning and control.

By the 19th century, statistics became mathematical. Probability theory formalized uncertainty, while inference developed tools for drawing conclusions from data. Thinkers like Laplace, Gauss, and Bayes provided the foundations for empirical science. The rise of frequentist and Bayesian paradigms in the early 20th century established two dominant schools of thought about how knowledge could be derived from observation.

Computing and the First Fusion

The advent of computing in the mid-20th century introduced a revolutionary capability: the mechanized manipulation of symbols at scale. Pioneers like Alan Turing and John von Neumann imagined machines that could simulate logic, calculation, and eventually decision-making. From these ideas came programmable computers, which changed how information could be handled.

In the 1960s and 70s, computation and statistics began to merge. Simulation-based methods, such as the bootstrap, Monte Carlo algorithms, and early machine learning models, emerged. The ability to process larger datasets enabled new techniques. But these developments remained within academic silos—few organizations were equipped to generate or exploit data at scale.

The Rise of Industrial Data

The 1980s and 90s saw the rise of enterprise computing. Data warehouses, relational databases, and business intelligence systems transformed organizational workflows. Meanwhile, algorithmic tools like decision trees, neural networks, and clustering matured. Yet "data mining," as it was called, was largely the domain of statisticians and operations researchers operating inside corporate IT departments.

Then came the internet. Starting in the late 1990s and accelerating into the 2000s, a torrent of user-generated data began to flood digital platforms. Clicks, searches, purchases, locations, and social signals were logged at previously unimaginable granularity. Suddenly, tech companies had both the need and the means to analyze behavior at scale. This catalyzed a redefinition of what data work required.

Naming the Discipline

The phrase "data science" gained prominence in the mid-2000s, as organizations sought roles that combined statistical expertise, computational fluency, and business relevance. In 2001, William Cleveland proposed data science as an independent discipline. In 2008, DJ Patil and Jeff Hammerbacher popularized it as the job title of the future. The term captured a shift: data work was no longer purely academic or operational—it was strategic.

By the 2010s, the archetype of a *data scientist* had taken shape: a hybrid skilled in statistics, coding, and domain expertise. The canonical "Data Science Venn Diagram" illustrated this convergence. Online platforms and bootcamps emerged to train a new workforce. The explosion of open-source tools—Python, R, Jupyter, scikit-learn—democratized access and accelerated innovation.

Fragmentation and Identity

As organizations scaled their data efforts, new bottlenecks emerged. Collecting and cleaning data, managing pipelines, and deploying models became formidable challenges. This led to the rise of *data engineering* and later *MLOps*—infrastructure practices that support the operationalization of data science. These subfields emphasized reproducibility, monitoring, and automation—less science, more systems.

Despite its successes, data science has struggled to define itself precisely. Is it applied statistics? Computational modeling? Business analytics? Software development? Different teams interpret the role differently. In some firms, data scientists build models; in others, they run SQL queries. The term has become elastic—useful for branding, but vulnerable to dilution.

The Scientific Core and Future Direction

At its best, data science embodies the scientific method applied to modern systems. It treats organizational behavior, customer actions, and system performance as empirical phenomena to be observed, modeled, and understood. This orientation—toward hypothesis, experimentation, and iterative refinement—distinguishes science from mere reporting or automation. Data science, properly practiced, is a mode of inquiry.

Today, data science is fragmenting and professionalizing. Specialized roles—machine learning engineer, data analyst, applied scientist—have emerged to reflect different emphasis areas. Meanwhile, generative AI, foundation models, and causal inference are reshaping the discipline's frontiers. The next chapter may look less like the monolith of "data science" and more like a federation of focused crafts.

Conclusion

Data science is young, but it stands on centuries of thought. It inherits questions from statistics, capabilities from computing, and relevance from the business world. Its evolution reflects a larger story: the increasing role of empirical reasoning in how we understand and shape the world. Whether or not the name sticks, the mindset will endure.

Where Data Science Fits in Technology Organizations

Data science is often misunderstood within the context of a modern technology company. While it shares similarities with software engineering, product analytics, and business intelligence, it is a distinct discipline with its own methods, goals, and organizational implications. To understand where data science belongs in a technology organization, we must understand what data science is and what kinds of work it produces. Only then can we structure teams and collaborations that let it thrive.

What Data Science Does — and Doesn't — Do

Data science is the application of the scientific method to understanding how systems behave. It is not a subfield of engineering. Nor is it just about making dashboards or building models. The core activity of data science is generating *knowledge*: explanations, predictions, hypotheses, theories, and evaluations about system behavior. These systems might be algorithms, business processes, user journeys, or product mechanisms. The output of a data science function is therefore scientific in nature: insights, frameworks, metrics, hypotheses, validated learnings.

By contrast, engineering teams are focused on *building* systems—software infrastructure, production algorithms, and user-facing features. Product teams are focused on *developing* systems—coordinating cross-functional work toward business objectives. Data science supports both, but its perspective is observational and interpretive rather than directive or generative. This distinction is essential.

Functional vs. Matrixed vs. Embedded Models

Where data science lives organizationally depends on how an organization balances centralization with specialization. There are three major models:

- **Functional Model**: All data scientists report to a centralized data science org. They may work on a range of projects across departments, with shared standards and a common leadership structure.

- **Embedded Model**: Data scientists report into the department or team they support (e.g., marketing, product, engineering), often leading to deeper integration but more fragmentation across the discipline.
- **Matrixed/Hybrid Model**: Data scientists report into a central data science org but are functionally embedded in teams across the company, typically with dotted-line relationships to those teams.

Each model has tradeoffs. Functional models promote strong peer support and shared practices but risk becoming disconnected from day-to-day product or business concerns. Embedded models encourage alignment with team priorities but often isolate data scientists and dilute standards. Matrixed models attempt to capture the best of both worlds but require clear role definitions and dual-accountability structures to succeed.

Data Science's Role in the Product Lifecycle

At a high-performing company, data science contributes throughout the product lifecycle:

- **Exploration & Ideation**: framing the problem space, sizing opportunities, identifying behavioral patterns, proposing hypotheses
- **Design & Planning**: defining metrics, estimating baselines, setting targets, designing experiments
- **Implementation & Launch**: building telemetry, supporting A/B tests, validating assumptions in real time
- **Monitoring & Evaluation**: analyzing outcomes, contextualizing results, diagnosing regressions, proposing next steps

Note that these contributions are not about shipping code or setting strategy. Instead, they help *inform* decision-making with a scientific perspective. This is true whether the project is product-facing, algorithmic, operational, or strategic. The data scientist is there to ask, "What do we know? How do we know it? What would change our belief?"

Data Science vs. Data Engineering vs. Analytics

Data science is often confused with related fields. Here's a rough breakdown:

- **Data Engineering**: builds pipelines, platforms, and infrastructure to make data accessible, reliable, and usable
- **Analytics / BI**: creates dashboards, reports, and KPIs to monitor and summarize key business or product metrics
- **Data Science**: investigates causality, uncertainty, and emergent behavior in systems using statistical and computational methods

These roles are complementary. In some companies, one person might wear multiple hats. But for mature teams, clarity between these roles helps assign responsibility appropriately and encourages the development of specialized methods and tools.

Organizational Allies and Tensions

Data scientists work across boundaries. Their natural collaborators include:

- **Engineers** (for instrumentation, telemetry, model integration)
- **Product Managers** (for hypothesis framing, decision-making support)
- **Designers** (for behavioral studies, experiment design)
- **Executives** (for forecasting, scenario analysis, strategy testing)
- **Operations** (for workflow optimizations and root cause analysis)

However, tensions may arise when stakeholders expect deliverables that fall outside the scientific function of data science. For example:

- When engineers expect production-ready code
- When PMs expect decisions rather than probabilistic evidence
- When leadership expects dashboards rather than research

These misalignments can be mitigated by educating peers about the scientific nature of the discipline and by clarifying expectations early in the collaboration.

Institutionalizing Research Practices

Because data science is fundamentally research-oriented, it benefits from structures that support long-term inquiry:

- **Hypothesis management systems** (e.g., Jira boards for research questions)
- **Documentation and reproducibility standards**
- **Knowledge repositories** for storing validated insights
- **Reading groups and peer review** to build shared epistemology
- **Metrics frameworks** that evolve with product understanding

These structures are difficult to maintain in organizations that treat data science as a service role rather than a core research discipline. But without them, data scientists spend too much time rediscovering known facts, fighting for prioritization, or maintaining dashboards instead of doing science.

Leadership and Career Development

Data science leadership differs from engineering or product leadership. Good data science managers:

- Advocate for scientific integrity over organizational convenience
- Create environments where critical thinking and skepticism are rewarded
- Invest in mentorship, research infrastructure, and knowledge management
- Protect time for exploration and open-ended investigations

Career ladders for data scientists should also reflect the dual technical-and-scientific nature of the role. Promotions should reward not just technical skills or number of projects delivered, but also clarity of thought, epistemic rigor, and contributions to collective understanding.

Conclusion: Science Inside the System

Data science belongs in technology organizations as the internal scientific arm. Just as a hardware company employs physicists and chemists, and a biotech company employs biologists and statisticians, a software company should employ data scientists to understand the behavior of its systems.

But for this function to work, the organization must recognize that data science is not just a set of tools, nor a dashboard factory, nor a subset of software engineering. It is a scientific discipline embedded in a socio-technical environment. It needs room to think, tools to investigate, and colleagues who understand its purpose.

Placed correctly, data science can reveal the system to itself—and help shape more intelligent, resilient, and adaptive technology.

Data Science as Applied Systems Science

Data science, as a term, is often mischaracterized as a synonym for statistical modeling or machine learning. While those are important tools in its toolbox, they do not define the discipline. To define data science properly, we must instead ask: what is the *object of study*? What is the data scientist ultimately trying to understand, model, or change?

The answer is systems. In the broadest sense, data scientists work to understand the behavior of complex systems through empirical observation, mathematical modeling, and computational experimentation. These systems might be physical, biological, economic, social, or technological—but they are systems nonetheless. That makes data science, at its core, a systems science.

But unlike theoretical systems science, data science is grounded in applied practice. It operates under real-world constraints: noisy data, unclear objectives, messy interfaces, and organizational politics. Its insights are judged not just by elegance or generality but by utility—whether they can be translated into improved decisions, predictions, interventions, or understanding in the system under study.

A useful analogy is to think of data science as the *engineering discipline* of systems science. Just as mechanical engineering applies physics to real-world machines, or chemical engineering applies chemistry to manufacturing and materials, data science applies systems theory and inference to the messy reality of organizations, products, and platforms.

This framing helps resolve confusion around the scope of data science work. A data scientist may spend weeks modeling user retention curves or anomaly detection systems, but that work only makes sense when placed in the context of a larger system—such as a digital product ecosystem, a marketing funnel, or a customer lifecycle. Without systemic context, the work risks becoming statistical navel-gazing.

Conversely, the data scientist's role differs from that of a pure software engineer, even if both work with code. The engineer builds systems; the data scientist studies them. The engineer implements features; the data scientist asks whether those features work, for whom, and why. These are distinct modes of thought, and conflating them can lead to misaligned expectations or misallocated talent.

One of the most important features of applied systems science is that it acknowledges and embraces feedback loops. When you deploy a model or recommendation, it changes the system. When you change incentives or alter measurement strategies, behaviors shift. A/B tests, recommender systems, forecasting models—all exert influence on the systems they measure. The

data scientist must reason not just about passive measurement, but about how interventions alter equilibrium states or generate unintended consequences.

This is also why experimental design is so central to data science. Experiments are not just tools for measuring lift; they are interventions in a system whose structure we are trying to uncover. If data science is systems science, then experiments are field studies—probes designed to elicit responses that reveal internal causal dynamics.

Another implication is that data scientists must understand how their systems are instrumented. This means more than reading a schema; it requires understanding how the data is generated, what biases are introduced by the collection mechanism, and what assumptions are embedded in pipeline logic. Without this understanding, any analysis rests on a shaky foundation.

In that sense, data science also inherits something from epistemology: the study of knowledge itself. What can we claim to know from this dataset? What causal claims are warranted? What generalizations hold beyond the observed context? These are systems questions, but they are also *scientific* questions. The term "data science" rightly includes that word: science.

From this perspective, much of the infrastructure of data science—dashboards, notebooks, logging frameworks, feature stores, metric layers—should be understood as the lab equipment of systems science. These tools make it possible to observe, hypothesize, test, and refine our models of how a system behaves. But they are not the end goal. The goal is understanding.

This systems framing also helps explain why the most effective data scientists tend to have broad interdisciplinary fluency. They draw from statistics, computer science, social science, behavioral economics, and sometimes domain-specific theory. Each contributes something to the system-level understanding: statistical tools offer validation, computer science offers scale and optimization, domain knowledge provides context and constraints.

Finally, treating data science as applied systems science invites us to think critically about our organizational role. We are not just analysts or modelers—we are theorists of the systems we inhabit. And with that role comes responsibility. If we misdiagnose the system, our recommendations may backfire. If we ignore feedback loops, our metrics may mislead. But if we do it well, we can illuminate the structure of systems that would otherwise remain opaque—and in doing so, help steer them toward better outcomes.

The Data Scientists

A Refined Venn Diagram of Data Science

The classic Venn diagram of data science—comprising "hacking skills," "math & stats," and "domain expertise"—has become a meme. While catchy, it lacks intellectual rigor. It neither reflects the disciplinary foundations of data science nor the epistemic processes by which knowledge is developed. If we're serious about defining data science as a discipline, we need to do better.

We propose a more precise model grounded in academic traditions and scientific method. Our refined Venn diagram of data science includes three intersecting domains:

- **Statistical Modeling** — the science of *validation* and inference,
- **Scientific Computing** — the engine of *optimization* and simulation,
- **Systems Research** — the generator of *applied intuition* and understanding.

Each brings its own principles, methods, and goals. The heart of data science lies in the interplay among them.

Statistical modeling provides the theoretical machinery to evaluate hypotheses, estimate uncertainty, and draw justified inferences from data. This includes not only classical frequentist tools but also Bayesian frameworks, experimental design, and causal inference. It is the domain of *epistemic humility*: accepting that claims must be backed by uncertainty-aware models, not confident guesswork.

Scientific computing brings numerical optimization, algorithm design, and simulation-based techniques. From gradient descent to MCMC, it allows models to be trained, tuned, and scaled. It also includes symbolic computation, differentiable programming, and high-performance computing—powering the machinery of modern ML.

Systems research includes software engineering, human-computer interaction, distributed systems, and sociotechnical modeling. It equips data scientists to work with real-world systems and real-world constraints. This is where questions are framed, where telemetry is designed, and where failure modes are diagnosed. It is the most neglected domain, yet it supplies the practical *intuition* for what data mean in operational contexts.

The intersections of these domains form key subfields:

- **Algorithms** emerge where statistical modeling meets scientific computing. Here we find regularization, optimization theory, kernel methods, and the mathematically grounded side of ML.

- **Experimentation** arises at the intersection of systems research and statistical modeling: the design and interpretation of tests and interventions in live environments.
- **Data Analysis** emerges between systems research and scientific computing: understanding how to extract interpretable signals from messy, large-scale, often poorly-instrumented systems.

Where all three meet—*where validation, optimization, and intuition converge*—is the space we should call data science proper.

This refined model has practical consequences. It can guide hiring and team design. A mature data science team should not just be a row of ML engineers. It should include researchers trained in experimental design, experts in large-scale systems instrumentation, and individuals who understand both the math and the meaning of the models they deploy.

Pedagogically, this triadic structure supports curriculum development. Courses can be mapped to ensure coverage of statistical methods, computational infrastructure, and systems reasoning. Degree programs can avoid overindexing on narrow optimization skills or purely theoretical training.

Epistemically, this model encourages a more honest engagement with how knowledge is constructed in practice. It acknowledges that neither statistical modeling nor ML algorithms are sufficient on their own. Insight arises not just from mathematical correctness or computational speed, but from embedding those tools within systems that generate meaningful observations and allow structured interventions.

To treat data science as a science is to acknowledge its multi-rooted nature. Not as a fusion of buzzwords, but as a structured intersection of disciplines, each with its own rigor, history, and methods. That intersection—carefully defined—is where data science lives.

The T-Model of Skill Development

The "T-Model" is one of the most enduring metaphors in the discussion of data science skill development. It illustrates a profile that combines both breadth and depth: a wide range of general competencies across domains (the top of the "T"), and deep expertise in at least one of them (the vertical stem). In data science, this structure is essential—not only for individual effectiveness but for the coherence and flexibility of data science teams.

What makes the T-Model especially relevant in data science is the fundamentally interdisciplinary nature of the field. A successful data scientist is not just a statistician or a programmer or a domain expert, but someone capable of navigating all three domains with fluency. However, few individuals can be deep experts in all dimensions simultaneously. The T-Model helps resolve this tension.

Horizontal Breadth

The horizontal portion of the T represents broad foundational knowledge. For data scientists, this means a working familiarity with:

- **Statistics and probability**: Understanding distributions, inference, experimental design, and uncertainty quantification.
- **Programming and software engineering**: Ability to write modular, testable, maintainable code; familiarity with data structures and algorithms.
- **Databases and data infrastructure**: Proficiency in querying data (typically via SQL), understanding of schemas, indexing, and performance concerns.
- **Machine learning**: Conceptual fluency with modeling techniques like regression, classification, clustering, and more complex architectures.
- **Domain knowledge**: Understanding the context in which data is collected, used, and interpreted.

This breadth allows data scientists to collaborate across disciplines and adapt their thinking to a wide range of problems. It makes them credible interlocutors to engineers, PMs, business stakeholders, and researchers alike.

Vertical Depth

The vertical stroke of the T represents depth in a particular area. This could be:

- Deep mathematical knowledge in statistical theory or Bayesian modeling
- Advanced software engineering skills (e.g., building ML infrastructure, production-grade pipelines)
- Specialized knowledge in a scientific or business domain (e.g., genomics, e-commerce, NLP, or ad tech)
- Deep understanding of causal inference or experimental methodology
- Fluency with specific tools or paradigms (e.g., probabilistic programming, time-series forecasting, graph analytics)

This depth enables the data scientist to innovate, create new techniques, or drive insight in a way that generalists cannot. It anchors their credibility and allows them to lead or mentor in that area.

The Team-Level T

The T-Model is not just a personal development guide—it's also a team design principle. In a well-composed data science team, each member may have a different area of vertical depth, but all members share a common language across the horizontal breadth.

This means that teams can share mental models, communicate clearly, and collaborate fluidly—even if their expertise differs. It also means that problems can be transferred or handed off between team members more easily, avoiding knowledge silos.

T-Model and Hiring

When hiring data scientists, companies often fall into the trap of searching for "unicorns" who are deep experts in everything. This is rarely realistic. Instead, hiring should aim to fill out the collective T-shape of the team. A team with redundant verticals but no breadth won't collaborate well; a team with breadth but no depth will lack edge.

A good hiring strategy identifies what verticals the team currently lacks, and which areas of breadth need strengthening. Junior hires might be broader generalists; senior hires may be expected to bring vertical depth.

T-Model vs. -Model

In some discussions, especially when extending from data science into broader fields like design or product, people use the **-model**—a profile with two verticals rather than one. In data science, this could reflect deep strength in both statistical modeling and software engineering, or in both machine learning and domain expertise.

However, the -model is better understood as a refinement of the T. It's helpful to recognize when someone has dual specialties, but still assumes a wide base of general competence. The goal is not to stack verticals indiscriminately, but to cultivate synergy.

Learning Pathways

The T-Model also guides self-development. Early-career data scientists should prioritize building out the horizontal bar—acquiring fluency across the stack, learning best practices in programming, solidifying statistical fundamentals, and becoming literate in ML workflows.

As one gains experience, the next step is to pursue vertical specialization. This might happen organically—by diving deeper into problems one encounters at work—or deliberately, through graduate study or focused personal projects.

What matters is not just learning in isolation, but applying that skill in practical, high-impact ways. The stem of the T is carved not by study alone, but by challenge, feedback, and iteration.

Pitfalls of Over-Specialization

There is a risk that depth in one area can lead to overconfidence, or a loss of collaborative flexibility. A data scientist who is deeply skilled in modeling but cannot explain results to stakeholders—or who cannot adapt models to practical constraints—risks becoming siloed.

The T-Model reminds us that depth must be built upon a foundation of breadth. Expert knowledge becomes valuable only when it is actionable in context.

Organizational Design Implications

Teams built around T-shaped individuals tend to be more resilient. They can shift responsibilities during turnover, adapt to changes in the tech stack or company priorities, and support mutual learning. A team full of deep specialists without shared language will fracture; a team full of generalists may stagnate.

T-shaped teams also support career growth. Junior members can learn by pairing with seniors in their verticals; seniors benefit from the questions and perspective that juniors bring across the horizontal.

Beyond the T

Some authors propose extensions to the T: the **comb-shaped** model (many shallow or moderate verticals), or the **E-shaped** model (breadth, depth, and experience in execution). These are useful elaborations, but they share the same core insight: collaboration requires common ground; expertise requires depth.

The T-Model is enduring because it's simple, flexible, and true. It doesn't prescribe what kind of data scientist one must be—it offers a structure for thinking about how to grow and how to build.

Diagram

Breadth: Stats, Coding, ML, Infra, Domain Depth: Specialization

Data Science as a Team Sport

The Lone Wolf Myth

The myth of the solitary data scientist is stubborn. The image of a lone genius hacking away at a notebook, conjuring insights from the abyss, still lingers in corporate lore and even in many hiring pipelines. But if you scratch the surface of real, functioning data science orgs—especially those that consistently deliver impact—you'll find something quite different: collaborative research labs (**passi2018problem?**).

Data science, at its core, is a collective enterprise. Its most productive form resembles the structure of a scientific research lab or an academic department, with a diversity of skills distributed across complementary team members. Each brings domain knowledge, statistical thinking, systems intuition, or engineering rigor to bear on the same fundamental goal: understanding and improving a complex system through evidence and iteration.

Roles and Structure

This framing runs counter to the way many companies think about "staffing" data science. Job descriptions often conflate incompatible expectations: the ideal candidate should design experiments, write Spark jobs, build dashboards, deploy models, wrangle stakeholders, and explain confidence intervals—all while maintaining a sunny disposition. It's not surprising that hiring mismatches are common and team morale can suffer (Donoho 2017).

Instead of looking for mythical unicorns, organizations should adopt a team-based model that recognizes data science as a research discipline. This model emphasizes specialization within a collaborative framework, in which roles are clearly defined, but outcomes are shared. A good analogy is the surgical team or film crew: you wouldn't expect the lighting director to write the script or the anesthesiologist to scrub in on post-op paperwork. But they all contribute to the success of the operation—or the story.

A practical data science team might include statistical modelers, data engineers, domain experts, and research leads. Some teams also include embedded analysts or product liaisons who maintain close contact with business stakeholders. These roles are not rigid, but the distinction helps avoid two common dysfunctions: underpowered modeling and overengineered pipelines.

Collaboration and Coordination

In a mature data science lab, project work is scoped collaboratively, but tasks are distributed according to strengths. Engineers design reliable data flows, analysts validate assumptions and communicate results, and scientists iterate on models or experiments. These activities are coordinated not by top-down command but by shared research goals, with regular design reviews and cross-functional critique (**lewis2020team?**).

Documentation, hypothesis logs, and design rationale are all critical in this setting. The lab model works best when intermediate progress is legible and reproducible. This makes the work reviewable, enables debugging, and allows knowledge to persist beyond the original researchers. It also makes onboarding easier and lets newcomers quickly contribute.

Importantly, a data science lab is not a "service desk" for other teams. It's not there to produce dashboards on request or answer one-off data questions. While analysts may occasionally do this kind of work, the lab's primary function is research: to pose and test hypotheses, evaluate mechanisms, and identify causes, tradeoffs, and opportunities. These insights feed into strategy, product development, or operational policy (**bailer2018data?**).

Research Programs and Culture

When structured properly, labs can organize themselves around subject matter areas rather than functional roles. For example, one lab might specialize in growth and customer acquisition, while another focuses on payments and fraud. This encourages both domain depth and methodological cross-pollination. Labs build expertise not only in their systems, but in the kinds of modeling, experimentation, and metrics most applicable to those systems.

Career development also benefits from this model. Junior data scientists can apprentice under experienced leads, contributing to real research projects while building their skills. Mid-level team members can rotate between labs or serve as methodological leads. The presence of mentors and a research ethos increases both satisfaction and retention (**willingham2021cultivating?**).

Even leadership changes in nature under this model. Instead of a flat analytics org reporting into product or engineering, the lab model supports a Director of Data Science who acts more like a Principal Investigator (PI) in a research institution. This leader sets vision, recruits talent, defines research programs, and secures institutional buy-in for long-term exploration.

The lab approach also makes room for genuine innovation. Because teams are not bound to narrow scopes or ticket-based workflows, they have space to explore unexpected questions, follow anomalous results, and develop novel methods. Some of the best insights in data science come from chasing curiosities that at first seem like outliers (Breiman 2001).

Conclusion

This isn't to say that structure and accountability disappear. On the contrary, lab work should be organized around well-scoped research programs with clear deliverables, hypotheses, and evaluation criteria. But unlike project-based models, where scientists are often roving utility players, the lab model promotes stability, continuity, and intellectual ownership.

There's also a cultural dimension. Labs foster a norm of critique and reflection, of shared intellectual responsibility. A failed experiment is not a personal failure—it's a data point. A surprising result is not a reason for dismissal—it's an opportunity for inquiry. When this culture is strong, even setbacks advance the team's understanding.

Finally, the lab model connects better to the broader scientific tradition. It encourages rigorous reasoning, documentation, and collaborative discovery. It situates data science not as a niche tech function, but as part of a centuries-old tradition of knowledge generation through structured inquiry (**peng2011reproducible?**).

In the end, data science isn't a solo sport. It's a team sport. And the better we design our teams—not just in composition but in mission, structure, and culture—the more likely we are to build organizations that learn, adapt, and thrive.

Research Management and Process

Managing Data Science Through Research Programs

Why Project Management Fails for Data Science

Data science defies traditional project management paradigms. While engineering is often structured around delivery timelines and dependency tracking, data science requires an environment conducive to inquiry, failure, and refinement of hypotheses. Managing this domain through short-term sprints or rigid milestone charts can distort its nature, incentivizing superficial wins and discouraging exploration. The more suitable alternative is to manage data science through long-term, open-ended research programs.

What Is a Research Program?

A research program is not merely a renamed roadmap. It is a structured approach to exploring a related class of questions under a shared intellectual objective. The boundary of the program is permeable and evolves as knowledge accumulates. Like an academic research lab, it supports inquiry across multiple threads while maintaining cohesion.

Imre Lakatos developed the idea of scientific research programs as a model for understanding how science progresses—not through isolated experiments but through evolving, structured commitments to a set of theories and methodologies that adapt over time (**lakatos1970falsification?**).

How the Research Lifecycle Aligns

A typical data science lifecycle begins with an intuition—often from a stakeholder—that a pattern might exist. The scientist explores, quantifies, models, and validates that intuition. At each step, new paths emerge. Traditional project management treats those forks as failures to plan. Research programs treat them as natural and necessary shifts in understanding.

Organizing work into programs allows data science teams to behave more like scientific research labs. A program might focus on "Buyer Elasticity" and include pricing models, segmentation

strategies, and simulations. Another might address "Content Personalization," from engagement modeling to fairness evaluations. The goal is not just output, but insight.

Progressive Elaboration and Knowledge Memory

Programs support what philosophers call *progressive elaboration*—the sharpening of questions and accumulation of explanatory power over time. As the program matures, it builds a memory: of hypotheses tested, approaches discarded, and models refined. This memory is vital to avoid repeating dead ends and helps onboard new team members with context and momentum.

This process echoes the cumulative vision of data science described by Donoho, who calls for systems that support replicability, distribution of knowledge, and shared intellectual infrastructure (**donoho201750?**).

Evaluating Programs Without Fixed Deliverables

Because research programs resist fixed timelines, they require alternative evaluations. Metrics like research *velocity*—number of hypotheses tested, models iterated, or experiments run—are one approach. *Influence* is another: did the program change product direction, stakeholder thinking, or team understanding?

Intermediate artifacts become key deliverables: decision memos, validated notebooks, and model cards. These should be as rigorous as papers, even if they don't follow a peer-review path. Internal publication models, versioned with Git and written in reproducible Markdown/Quarto formats, serve this purpose well.

Reducing Context Switching Through Program Cohesion

Programs reduce context switching. Instead of jumping across unrelated Jira tickets—e.g., churn analysis, attribution modeling, and experimental design—scientists focus on a coherent domain. They reuse code, deepen subject-matter understanding, and evolve hypotheses longitudinally.

The cost of context switching is real: degraded focus, fragmented knowledge, and lower reproducibility. This principle is well-recognized in cognitive psychology and productivity science.

Program Hierarchy and Organizational Mapping

At an organizational level, three to six programs may be enough to reflect major strategic themes. Within each, workstreams—like mini research projects—can be scoped, prioritized, and retired dynamically. This structure aligns resourcing, reporting, and planning to substantive areas of progress.

Netflix's approach exemplifies this: it treats applied research as a continuous investment area, with alignment between research programs and long-term product vision (**amatriain2016netflix?**).

Documentation as Infrastructure

Each research program should maintain a "program brief," continuously updated and version-controlled. It includes:

- Primary questions and hypotheses
- Current and planned investigations
- Links to code, dashboards, and past results
- Open questions and knowledge gaps

These briefs function like evolving lab notebooks, protecting against knowledge loss and enabling internal peer review.

Collaboration Across Teams

Programs create opportunities for cross-team collaboration. Two teams in different parts of the org working on, say, "forecasting under uncertainty," can contribute to the same program, align on shared metrics, and exchange tools and findings. This model echoes academic research, where labs co-publish or share data infrastructures.

Such sharing builds a culture of transparency and open inquiry, and encourages code standardization, statistical rigor, and institutional learning.

Insulating Programs from Operational Churn

For research programs to flourish, they need time and insulation. This doesn't mean detachment from business reality—it means permission to follow the scientific process without being pulled into every reactive business ask. Without this protection, teams become reactive rather than generative.

Leadership must recognize that most valuable insights come not from short-term optimization but from sustained programs that allow deep model understanding and system-level exploration.

Guarding Against "Research Theater"

Not all programs are equal. There is always a risk of "research theater"—where teams simulate inquiry without rigor, producing dashboards or papers that are never used or validated.

A good program is grounded in real-world systems. It engages with stakeholders but holds scientific standards: falsifiability, reproducibility, and relevance. It treats failure as informative and values insight over deliverables. The scientist is not a ticket-taker but an epistemic agent advancing the organization's understanding.

Conclusion

Managing data science through research programs reflects the epistemological reality of the work. It aligns with the scientific method, supports cumulative progress, and empowers researchers to engage deeply with their domains. Programs turn data science from a reactive function into a durable engine of discovery.

References

The Scientific Method in Data Science

The Method Is the Discipline

The scientific method isn't just a metaphor for how data science works — it is the method. If data science is to remain science and not drift into analytics theater, the discipline must center itself on systematic hypothesis testing, error analysis, and the incremental refinement of understanding. Models, dashboards, and even causal inference pipelines are tools. The method is the process.

At its core, the scientific method is an iterative loop that begins with observations, forms hypotheses, tests those hypotheses with data, and refines those ideas in light of the results. This framework, ancient in origin but modern in its rigor, enables the accumulation of reliable knowledge in the face of uncertainty and noise — the very conditions of production data (Popper 2002; Gelman et al. 2013).

Against the Ticket Queue

In most tech organizations, however, this process is implicitly degraded. Business stakeholders prefer answers to questions, and models to uncertainty. Data scientists are frequently pushed into a service model: pull the data, run the regression, answer the ticket. But such a mode of operation short-circuits the fundamental value proposition of science — not to produce answers per se, but to generate validated knowledge (Wilson et al. 2014).

To realign with the scientific method, data science teams must reconceptualize their work as research programs rather than ticket queues. This means developing research questions, designing hypothesis-driven experiments, and documenting outcomes, whether confirming or null. It also means resisting the allure of spurious correlations, post-hoc rationalization, and aesthetic dashboards with no inferential content (Beaulieu 2020).

Hypothesis Stories in Practice

One practical entry point is the introduction of hypothesis stories — a lightweight narrative form that frames any analysis as a falsifiable test. A hypothesis story includes: (1) the intuition or anomaly prompting inquiry, (2) the operational hypothesis to be tested, (3) the method for testing it, and (4) the expected implications of each possible outcome. It's a format that brings both rigor and clarity, and it can be embedded directly in a team's workflow.

Tools like Jira and Notion can be reappropriated for this purpose. Instead of task cards, each analysis ticket becomes a living experiment write-up. The story evolves as data is collected, tested, and interpreted. This approach preserves the continuity of thought across weeks or months of work and allows others to trace the intellectual lineage of decisions made.

Accumulating Knowledge

The output of this process should be cumulative. Data science, like any scientific discipline, depends on a shared corpus of knowledge. Each completed hypothesis story, once validated, becomes a building block — a finding that can be cited, revisited, or contradicted. Teams should maintain an internal knowledge repository, akin to a lab notebook, in which these results are captured and synthesized (Leek and Peng 2017).

Such a shift is cultural, not just procedural. It demands that leadership reward clarity over certainty, and depth over speed. It requires that product managers and analysts align on framing questions, not just interpreting results. And it implies that models are not the final output, but instruments of a broader research agenda (Passi and Barocas 2020).

Embracing Imperfection

Of course, the rigor of science cannot be absolute in a commercial setting. Time constraints, incomplete data, and organizational pressure all impinge on ideal practice. But that makes the scaffolding of the scientific method even more essential. It ensures that even when shortcuts are taken, they are acknowledged as such — and that findings are always framed in probabilistic, not deterministic, terms.

One important ally here is Bayesian reasoning. The Bayesian framework formalizes belief updating, allowing teams to express uncertainty, incorporate prior knowledge, and revise expectations in light of new evidence. It maps cleanly onto the scientific method and can be a bridge between intuitive judgment and formal inference (McGrayne 2011).

Another helpful structure is the distinction between exploratory and confirmatory analysis. Exploratory work is generative and open-ended; it's where ideas are born. Confirmatory analysis tests those ideas against the harsh discipline of data. When the two are conflated — as they often are — findings become less reliable. Separating them maintains epistemic hygiene (Tukey 1977).

A Smarter Team, Not a Slower One

Critically, this orientation doesn't slow teams down. It makes them smarter. A team that operates with a scientific mindset iterates more effectively, avoids repeated mistakes, and builds organizational memory. It knows what it knows, and more importantly, it knows the limits of what it knows.

In the long arc, this produces not just better models, but better understanding. And that, more than any one predictive output, is the lasting contribution of data science to the organization.

> *"The aim of science is not to open the door to infinite wisdom, but to set a limit to infinite error."* — Bertolt Brecht (via *Galileo*)

Using the Capability Maturity Model in Data Science

In many organizations, data science initiatives begin with enthusiasm but lack structured frameworks to assess or guide their evolution. The Capability Maturity Model (CMM), originally developed to evaluate software development processes, offers a powerful lens for assessing the maturity of data science practices. By adapting the CMM to data science, teams can better understand where they are, where they could go, and what steps are required to grow from ad hoc experimentation to rigorous, institutionalized innovation.

From Software to Science: A Rationale

The CMM was introduced by the Software Engineering Institute to assess the process maturity of software organizations, guiding them from chaotic, hero-driven development to optimized, quantitative refinement of practices (Paulk et al., 1993). Despite its origin in software, the staged model of maturity maps well onto data science because both are process-heavy disciplines that demand rigor, repeatability, and quality control.

Stages of Maturity in Data Science

An adapted CMM for data science typically includes five stages:

1. **Initial (Ad hoc)** – Efforts are unstructured, success is person-dependent, and reproducibility is rare.
2. **Repeatable** – Some basic processes are reused (e.g., standard SQL queries, common scripts), but documentation and metrics are inconsistent.
3. **Defined** – Processes are documented and standardized across teams. Models are versioned, reviewed, and tied to business objectives.
4. **Managed** – Quantitative metrics guide model performance, experimentation, and business impact. Experiment tracking and reproducibility are central.
5. **Optimizing** – The organization continuously refines practices using feedback loops, automated retraining, and post-deployment monitoring.

These stages build progressively on each other, but in practice, different aspects of a team (e.g., modeling vs. data engineering) may reside at different maturity levels simultaneously.

Diagnosing Maturity

Assessing data science maturity involves identifying key dimensions such as:

- **Experimentation discipline**: How rigorously are hypotheses defined and validated?
- **Model lifecycle management**: Are models reproducible, versioned, and monitored?
- **Knowledge retention**: Are results documented, shared, and used to inform future work?
- **Integration with product development**: Are models tied to measurable business outcomes?

A diagnostic framework across these axes can expose inconsistencies and help prioritize improvements.

Practical Applications

Teams at the lower stages often focus on local wins—delivering one-off analyses or models without long-term sustainability. However, maturity shifts the focus toward *institutional learning*—transforming one-off experiments into robust systems. For instance, implementing standardized experiment tracking (e.g., with MLflow or Weights & Biases) can move a team from Stage 2 to Stage 3 by enabling replicability and peer review (Zaharia et al., 2018).

At higher levels, maturity enables organizations to treat data science as a platform capability. Mature organizations invest in internal research programs, internal tooling, and shared knowledge repositories. At Stage 5, the organization continuously improves its research productivity and decision-making precision through rigorous feedback cycles and automation (Furterer, 2009).

Maturity as Leverage for Leadership

Executives often demand business impact, yet data science teams may struggle to translate their work into such terms. A maturity model provides language to frame conversations not just around outcomes, but capabilities: "We're not yet at a stage where we can deliver real-time uplift estimates because our model monitoring infrastructure is still at Stage 2."

This framing turns a capability gap into an investment roadmap, positioning data science as a strategic function, not a service desk. It also makes success legible across organizational boundaries by highlighting process improvements in addition to model results.

Conclusion

Using a Capability Maturity Model framework allows data science leaders to chart a realistic progression from chaos to optimization. It encourages structured growth, supports strategic investment, and anchors scientific curiosity in process discipline. As the field matures, such frameworks may become essential not only for managing teams, but for managing trust in data-driven decisions themselves.

References

Tooling and Lab Practice

The Data Scientist's Lab Workbench

(SQL as data collection, Python as instrumentation, Jupyter/Markdown/Quarto as lab notebooks.)

Documentation and Knowledge Repositories

Documentation and Knowledge Repositories

Good documentation is a cornerstone of effective data science practice. While the technical output of data scientists is often code, models, or dashboards, these artifacts cannot achieve full impact without accompanying narrative, context, and explanation. Documentation ensures that the logic behind analyses, model assumptions, limitations, and interpretive frameworks are preserved for both future practitioners and stakeholders outside the immediate team.

The Role of Documentation in Scientific Work

Data science is fundamentally a scientific discipline, and science requires clear records. The scientific method demands transparency, reproducibility, and iterative refinement. These are impossible without documentation. Whether it's a well-annotated Jupyter notebook, a design doc laying out hypothesis structure, or a Markdown file explaining the evaluation methodology of a model, these artifacts support the continuity of research programs and the accumulation of knowledge over time.

In traditional science, the lab notebook was the primary medium for preserving thought processes and experiment design. In modern data science, digital equivalents like Git repositories, Quarto documents, Confluence pages, or shared Google Docs serve similar purposes, though often with a broader audience and more collaborative intent.

Types of Documentation

Different forms of documentation serve different needs:

- **Reference documentation**: These are technical descriptions of systems, APIs, datasets, and procedures. They must be precise, up to date, and easily searchable.
- **Analytical narratives**: These are reports or research memos that explain the motivation, process, and conclusions of an analysis. They provide interpretive framing and are essential for sharing insights across teams.
- **Synthesis documents**: These summarize and contextualize knowledge across multiple analyses or systems, such as a README for an experimental project or a whitepaper justifying a model deployment strategy.
- **Procedural knowledge**: This includes onboarding guides, checklists for releasing models, data access instructions, and other forms of institutional memory that facilitate operational continuity.

The Knowledge Repository

A well-organized knowledge repository is more than a wiki or document archive. It is a living system that curates, surfaces, and preserves knowledge. It supports search, synthesis, and serendipitous discovery. When structured effectively, it accelerates the learning curve for new team members and enables reuse of past work. This repository often acts as the institutional memory of the data science function.

Successful repositories often balance centralized structure with decentralized contributions. Index pages, tagging systems, and clear naming conventions help maintain structure. Templates and documentation standards reduce friction for contributors.

Writing for Different Audiences

Not all documentation is written for other data scientists. Some is written for product managers, engineers, or leadership. This requires conscious adaptation of language and framing. Analytical documents should distinguish between **peer-facing analysis** (which might include exploratory code and detailed methodology) and **stakeholder-facing synthesis** (which should prioritize clarity, narrative, and decision-relevance).

Tools of the Trade

Modern documentation tools span code, prose, and visual interfaces:

- **Markdown and Quarto** allow integration of text, code, and plots in a coherent document.
- **Jupyter Notebooks** are useful for exploration and lightweight narrative, though often suffer from poor version control and reproducibility.
- **Wiki systems** like Confluence or Notion support collaborative documentation but require thoughtful maintenance to avoid sprawl.
- **Version control systems** like Git are indispensable for preserving code and history, especially when documentation lives alongside the codebase.

The best environments treat documentation as a first-class citizen in the development lifecycle, with documentation reviews treated as seriously as code reviews.

Institutionalizing Documentation

To move from ad-hoc practice to organizational habit, teams must create rituals and norms around documentation:

- Post-analysis writeups are expected and reviewed.
- Decisions are linked to analytical documents.
- Project kickoffs include documentation plans.
- End-of-quarter retrospectives review and update the knowledge base.

These practices not only improve operational discipline but also foster a culture of thoughtfulness, explanation, and humility.

Conclusion

In data science, documentation is not an afterthought—it is part of the science. It bridges the gap between individual analyses and organizational learning. A robust knowledge repository does not just store documents; it reflects the intellectual scaffolding of the team. Investing in it is investing in the future capability of the organization.

References

Statistical Thinking

Probability and Statistical Inference

(Quantifying uncertainty, validating observations, and the logic of belief.)

Beaulieu, Anne. 2020. "Data and Society." *Annual Review of Anthropology* 49: 151–67. https://doi.org/10.1146/annurev-anthro-102218-011429.

Breiman, Leo. 2001. "Statistical Modeling: The Two Cultures (with Comments and a Rejoinder by the Author)." *Statistical Science* 16 (3): 199–231.

Donoho, David. 2017. "50 Years of Data Science." *Journal of Computational and Graphical Statistics* 26 (4): 745–66.

Gelman, Andrew, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. 2013. *Bayesian Data Analysis*. 3rd ed. Boca Raton: CRC Press.

Leek, Jeffrey T., and Roger D. Peng. 2017. "Opinion: Five Ways to Improve Reproducibility." *Nature* 551 (7685): 768. https://doi.org/10.1038/d41586-017-07522-z.

McGrayne, Sharon Bertsch. 2011. *The Theory That Would Not Die: How Bayes' Rule Cracked the Enigma Code, Hunted down Russian Submarines, and Emerged Triumphant from Two Centuries of Controversy*. New Haven: Yale University Press.

Passi, Samir, and Solon Barocas. 2020. "Problem Formulation and Fairness." *Proceedings of the ACM on Human-Computer Interaction* 3 (CSCW): 1–22. https://doi.org/10.1145/3359228.

Popper, Karl. 2002. *The Logic of Scientific Discovery*. London: Routledge.

Tukey, John W. 1977. *Exploratory Data Analysis*. Reading, MA: Addison-Wesley.

Wilson, Greg, Arjun Aruliah, C. Titus Brown, Neil P. Chue Hong, Michael Davis, Richard T. Guy, Steven H. D. Haddock, et al. 2014. "Best Practices for Scientific Computing." *PLOS Biology* 12 (1): e1001745. https://doi.org/10.1371/journal.pbio.1001745.