Birth, Life, and Death of AI Models*

Aaron P. Kaye† **Kazimier Smith** **Neil Thompson**

Boston University & MIT MIT MIT

September 9, 2026

Abstract

We characterize the lifecycles of open-weight AI models using a weekly panel of the 108,074 most-downloaded models on Hugging Face, which together account for 99.5 percent of the platform's more than 53 billion downloads. We link these data to token usage from OpenRouter and performance rankings from Arena and Artificial Analysis. We establish key institutional details about the supply of and demand for AI models, and we organize our analysis into three phases: birth, life, and death. In the "birth" phase, we show that major developers often release models in batches, concurrently introducing models that are both vertically differentiated (e.g., by size, capability, and latency) and horizontally differentiated (e.g., by task specialization, hardware compatibility, and alignment). Because weights are open, developers can also build on one another's models. We document cumulative innovation through fine-tuned, quantized, merged and adapted descendants of source models. In the "life" phase, we quantify how demand evolves after release. Even within the sample of top models, demand is concentrated, and the concentration persists: the median model loses ∼68 percent of its release-week downloads within ten weeks, while the most popular models decline gradually; as a result, the top percentile's advantage over the median model is ∼600-fold at release, and ∼900-fold six months later. Further, we estimate spillover effects from new popular model releases on related models and find that the first popular third-party descendant is associated with a 76 to 99 percent increase in weekly downloads of its source model. In the "death" phase, we construct measures of technical obsolescence and usage obsolescence. We find that technical obsolescence is associated with a slow but persistent decrease in demand, while usage obsolescence coincides with an immediate but temporary drop in demand. Our findings characterize open-weight AI as a market of complementary model portfolios, one in which third-party developers building on a source model can raise demand for the original, and in which demand is slow to respond to technical obsolescence.

Keywords: Artificial Intelligence, Open-Source, Open-Weights, Technology Diffusion, Platform Economics, Cumulative Innovation.
JEL: O33, L17, L86, O31

*First version: September 9, 2026. We thank Isaiah Andrews, seminar participants at MIT FutureTech, and conference participants at CESifo for helpful feedback and suggestions. This paper uses publicly available data and is not subject to a data-sharing agreement or editorial review from Hugging Face, OpenRouter, Arena, Artificial Analysis, or any other platform. We also thank all researchers at Hugging Face who helped make this research possible, including Avijit Ghosh, Yacine Jernite, and Giada Pistilli. All analyses, conclusions, and any remaining errors are our own.

†Contact: apkaye@bu.edu.

1 Introduction

Artificial intelligence is increasingly the subject of debates about productivity, competition, energy use, safety, and industrial policy, yet the market's basic institutional details remain largely undocumented. In mature industries, researchers typically know what the product is, how firms version it, how they introduce new vintages, and when old vintages effectively exit. Although much of the policy discussion treats the AI market as one that consists of a few firms competing to develop ever-larger frontier models, these basic questions about the structure of the AI market remain open. The open-weight market is broad, economically relevant, and the subject of policy discussion.[1] Hugging Face hosts more than three million open-weight models, with over 53 billion cumulative downloads. On OpenRouter, the largest marketplace between inference providers and users, open-weight models account for nearly 30% of token use (Aubakirova et al., 2026).[2]

Training a model from scratch is expensive, but developers can reuse nearly the same underlying training investment to produce a family of related models, which permits cost synergies across a firm's portfolio of models. Such portfolios can also be expanded: once a model's weights are released, they are relatively cheap to copy and modify, allowing both the original developer and third parties to create descendants through fine-tuning, quantizing, distilling, adapting, and merging. Understanding technological change in this market requires evidence on how models enter, how developers use and build on them after release, and when models effectively exit. To fill this gap, we establish basic institutional details about the open-weight AI market and empirically document the lifecycle of open-weight AI models using novel data from Hugging Face, OpenRouter, Arena, and Artificial Analysis. We organize our analysis into three phases: birth (the entry and differentiation of new models), life (the evolution of use and cumulative innovation after release), and death (the process of obsolescence and persistence).

In markets where the relevant products, decisions, and competitive margins are not yet well understood, establishing key institutional details can be critical in shaping research that furthers our understanding of the new market, and better informs policy. Economics research has a tradition that highlights the importance of establishing institutional details. Griliches (1957) documented how hybrid corn spread across U.S. states at different rates, establishing the empirical patterns that shaped decades of further work on technology adoption. Gort and Klepper (1982) traced the entry and exit of firms across product markets, identifying the lifecycle that became a benchmark for theories of industry evolution. Dunne et al. (1988) measured the basic patterns of firm turnover in manufacturing, providing the facts that motivated dynamic models of industry structure. Lerner and Tirole (2002) laid out the institutional logic of open-source software, clarifying the incentive structures that standard industrial organization frameworks had not yet addressed. De Loecker et al. (2020) and Conlon et al. (2023) documented the evolution of firm markups across the U.S. economy, providing empirical evidence to inform policy debates on inflation. In each example, correctly establishing institutional details enabled further research and informed policy. We aim to do something similar for AI by establishing descriptive institutional details and providing empirical analysis to characterize how models enter, compete, diffuse, and persist.

[1] See Letter on "Open Weights and American AI Leadership," organized by Nvidia and Microsoft, July 24, 2026; Anthropic's "Position on open-weight models," July 27, 2026; and Thinking Machines' "A Safe Path to Open Weights," July 31, 2026. Using open-weight models requires either self-hosting or using an inference provider, for example through a marketplace like OpenRouter.

[2] After the end of our sample, Stripe announced on August 19, 2026, that it had agreed to acquire OpenRouter at a $7.5 billion valuation, and NVIDIA announced on September 3, 2026, that it had agreed to acquire Hugging Face at a $12.9 billion valuation. The agreements do not affect our empirical analysis or findings. See Stripe, "Stripe Agrees to Acquire OpenRouter to Help Businesses Optimize Token Routing and Usage," August 19, 2026, and NVIDIA, "NVIDIA to Acquire Hugging Face," September 3, 2026.

We study these questions in the context of open-weight models. This is an especially appealing setting to understand the AI market, as it offers an unusually visible window into a broader AI market that is otherwise difficult to observe. Open-weight models are models where the weights are publicly available even when their full training code or data are not.[3] A few platforms act as intermediaries between model developers, compute providers, and users. These publicly provide detailed information on the models and their demand. Additionally, there is a growing market of platforms dedicated to publicly benchmarking model capabilities. Our primary data come from Hugging Face, the largest platform for hosting and distributing open-weight models. We supplement these data with usage data from OpenRouter, a platform that connects users to compute providers for a variety of open-weight and proprietary models, and with performance rankings from Arena and Artificial Analysis. Together, these sources let us observe supply-side product-market choices and demand-side adoption patterns. On the supply side, developers choose what to release, how many versions to release together, and how to differentiate those versions. Further, on the supply side, we explicitly observe cumulative innovation: a source model can be fine-tuned, quantized, adapted, or merged with other models to create a new, potentially competing, descendant model.[4] On the demand side, we observe attention and adoption over time, which we can use to document the model lifecycles and how adoption changes after the release of related, competing, or more capable models. Our main measures of model demand trade off between coverage and directness. Hugging Face provides broad measurement of an upstream margin of open-weight model demand, while OpenRouter directly measures downstream inference for a narrower part of the market. We provide an econometric framework under which within-model changes in downloads identify proportional, and consequently unitless, changes in latent model demand, without requiring a download to represent the same amount of use across models.

Following our birth-life-death framework, we provide a set of stylized facts and empirical analysis of how open-weight AI models enter, evolve, and exit the market. In the "birth" phase, we find that major developers frequently release models in batches, introducing families of models that are vertically differentiated by size, capability, and latency, and horizontally differentiated by domain specialization, hardware compatibility, and alignment. This implies that the relevant unit of product design is often the model family rather than serial versions of individual models. Further, we find that third-parties play a substantial role in expanding these model families. From our sample of popular models, 30,191 are descendants (27.9%), meaning model entry is shaped by both product-line design and cumulative innovation.[5]

In the "life" phase, we show that hype and demand follow different patterns: Attention peaks quickly, but downloads persist much longer. Across models, demand is concentrated in a subset of ultra popular models. Of the roughly 3 million models on Hugging Face, the top 10% account for 99.8% of downloads, the top 1% account for 98.8%, and the top 0.1%, just 2,922 models, account for 93.3% of downloads. Within a model, demand is concentrated near release, with the median model losing 68% of its release-week downloads within ten weeks.[6] Further, we investigate demand spillovers between source models and their descendants. Theoretically, the effect of a popular descendant is ambiguous: it could cannibalize demand for the source model by offering an upgraded alternative, or it could boost demand by drawing attention to the source in an otherwise crowded market. We find strong evidence of the latter: the release of a first popular third-party

[3]When a model is open-weight, anyone with the appropriate hardware can deploy it. By contrast, using closed models requires a compute provider with access to the model.

[4]Notably, since the models are open-weight, third party developers can build off of another developer's work.

[5]We consider 27.9% a lower bound, as developers self-report source-descendant relationships, including quantizing, fine-tuning, merging, and adapting, but not model distillation.

[6]The most popular models decline more gradually, so the top percentile's advantage over the median grows from roughly 600-fold at release to 900-fold six months later.

descendant is associated with a 76 to 99 percent increase in weekly downloads of its source model. However, for subsequent releases, we find generally negative, but statistically insignificant, spillovers, meaning we cannot definitively rule out either force as the number of descendant models grows.

In the "death" phase, we define usage obsolescence as the point at which a model is delisted or no longer attracts meaningful token usage on OpenRouter, and technical obsolescence as the point at which the model loses competitive standing on benchmark or preference-based rankings. Among models that reach each threshold, the median age is 9 weeks at technical obsolescence and 36 weeks at usage obsolescence. We evaluate the impact of reaching technical obsolescence with an event study design, finding that demand often persists long after a model falls behind the technical frontier. By contrast, we find that demand for downloading the model drops significantly after usage fades on Open Router (usage obsolescence). A number of mechanisms might explain these results. For example, open-weight models could be embedded in larger processes like software packages. Users and firms might face switching costs that prevent them from updating their AI workflows at the same frequency as technological improvements. We find that usage and technical obsolescence have different timings and different demand responses, consistent with the switching cost explanation.

Both the institutional details and the empirical results we establish are valuable for AI researchers and policymakers, whose assumptions about market features shape their research conclusions. For example, consider the assumptions underlying these three questions: first, what is the value of the data used to train AI models? Second, how do open weight models affect competition and innovation in the AI market? Third, what does model adoption reveal about demand for improvements in AI capability? On the value of data, assuming firms train economically relevant models one at a time implicitly treats each model as the outcome of a separate training decision. The batch releases and descendants we document show that this model-by-model assumption ignores a training investment's potential synergies across models in the same and subsequent batches, as well as across developers through descendant models. On how open weights affect competition, it is tempting to assume that open-weight models face fierce competition from third-party descendant models, ignoring the possibility of positive spillovers. On technical progress and adoption, evaluating models only on benchmark performance ignores potential lock-in, whereby demand persists long after technical obsolescence. Similar assumptions underpin other important questions, like market definitions in antitrust analysis and safety proposals for older models. The empirical facts we document can inform specific research questions about AI and advance our understanding of the broader AI market.

1.1 Related Literature

This paper relates to three strands of the literature. First, a growing economics and policy literature studies foundation models as a potentially general-purpose technology and asks how openness, complementary assets, and market structure shape diffusion. Second, research in computer science studies the open-weight AI market, focusing on interdependent supply chains created through model reuse, fine-tuning, and cumulative innovation. Third, a budding empirical literature measures pricing and demand in markets for model inference.

The first strand emphasizes that, in addition to technical progress of frontier AI models, the economic significance of AI also depends on downstream innovation, organizational changes, and market structure. Bresnahan (2024) suggests that AI becoming a general-purpose technology requires complementary co-invention and digital transformation. Azoulay et al. (2024) show that openness does not guarantee entry if incumbents control complementary assets, and Korinek and Vipra (2024) highlight how economies of scale and scope, as well as vertical integration, can push AI-model markets toward concentration. Similarly,

Kapoor et al. (2024) discuss the properties of open foundation models associated with innovation benefits and governance risks. We contribute to this literature by providing new evidence on how open-weight models enter and compete. Major developers maintain vertically and horizontally differentiated model portfolios, with major releases often arriving in batches of differentiated models. Our findings suggest that, for questions related to competition and innovation, the relevant unit of analysis is often a firm's portfolio of models rather than an individual model release.

The second strand studies open-weight AI as an ecosystem of reuse and cumulative innovation. Jiang et al. (2023) document that reuse of pretrained models on Hugging Face depends on attributes such as provenance, reproducibility, and portability, but also faces problems of missing metadata, performance discrepancies, and model risk. Dettmers et al. (2023) show that QLoRA sharply lowers the resource cost of adapting large models, making downstream fine-tuning feasible on much more modest hardware. Laufer et al. (2025) adopt an approach from evolutionary biology to document family trees of AI models on Hugging Face. We connect to this literature by adding an economic lifecycle perspective: descendant creation, inherited lineage, and downstream adaptation shape how models create value after release.

The third strand of the literature centers on demand for AI models. Demirer et al. (2025) document rapid growth in models, creators, and providers, along with price declines, persistent price heterogeneity, and both horizontal and vertical differentiation in demand. Fradkin (2025) shows that, on OpenRouter, new model releases are adopted rapidly, but differ substantially in whether they substitute for incumbents or expand the market, and that multihoming across models is common. We build on this literature by making two contributions to the measurement of AI model demand: new data and a complementary econometric framework. On data, we are the first, to our knowledge, to construct a panel of model-week level downloads using publicly available data from Hugging Face. We use publicly reported 30-day rolling and cumulative totals to reconstruct implied weekly downloads for each model.[7] Further, we show that weekly download data are especially useful for answering questions about demand and the impacts of related model releases. Previous papers have relied on 30-day rolling downloads, which mask the precise timing of changes in demand around events of interest and make it especially difficult to understand shorter model lifecycles. This granularity also allows us to demonstrate that "likes," a common proxy for adoption in prior studies, and downloads follow considerably different patterns.[8] On the methods side, we provide an econometric framework and a modest set of assumptions that formally map our constructed weekly download measure to overall changes in market demand.[9]

1.2 Outline

The remainder of the paper is organized as follows. Section 2 describes the institutional setting, data sources, and sample construction. Section 3 examines model entry and differentiation, including batch releases, model families, and descendant models. Section 4 documents how attention and demand evolve after release and estimates spillovers from subsequent same-author releases and third-party descendants. Section 5 distinguishes usage from technical obsolescence and studies how demand and attention change around each form of obsolescence. Section 6 discusses our results on how they can inform some of the major questions

[7] In other recent work, Longpre et al. (2025) examine changes in model concentration over time. They also use weekly download data. However, they use internal Hugging Face data from an earlier period, June 2020 to August 2025.

[8] Kadasi et al. (2025) notes the limitation of the raw publicly available download data, and instead proposes using likes as a measure of adoption.

[9] The primary assumption is that any event of interest can change demand, but not the relationship between downloads and demand. We call this assumption modest, as it still allows for the relationship between demand and downloads to vary across models, across time, and across model age.

facing the AI industry. Finally, Section 7 summarizes the findings and discusses their implications for AI policy and future research.

2 Data and Setting

Our primary data source is Hugging Face, the largest online platform for hosting machine learning models.[10] Before the rapid rise of large language models, Hugging Face offered an interface through which developers could use popular machine learning methods. As transformer-based models began to dominate the machine learning ecosystem, Hugging Face became a platform for storing and accessing AI models. Today, Hugging Face hosts more than three million models. Because of its popularity and ease of use, it is one of the most complete records of open-weight models available (Laufer et al., 2025). The key qualifier is *open-weight*. Hugging Face makes each model available to any developer that wants to use it by storing the model "weights," a set of numbers learned during training that the model uses to convert input to output. Developers download these weights in a specific format (typically this is done in a straightforward way through the Hugging Face API). Developers can then use the model on their local hardware. Models whose weights are publicly available are called *open-weight*. The term differs from the more common term *open-source*, which refers to software whose *code* is publicly available. Critically, the code used to train and develop an open-weight model is not necessarily made public. Only the core model architecture need be available for developers to use the model. For example, Meta's Llama models are open-weight but not open-source. In contrast, models like OpenAI's GPT-5 are *proprietary*: their weights are not publicly available and users must interface with the model through OpenAI-approved channels. Even the exact parameter count of proprietary models is often a closely guarded secret (Wan et al., 2025). Hugging Face is an excellent environment to study the lifecycle of AI models because of its scale. We have sufficient data to reliably estimate lifecycle patterns that would be difficult to infer from a small sample of proprietary models.

We retrieve data on all models on Hugging Face from the "Model Hub."[11] The model hub provides daily snapshots of every model, along with additional metadata, going back to July 24th, 2024. There are 587 snapshots in total. This is the primary source of our raw data. Each snapshot contains a list of models and many metadata fields. We construct a weekly panel of Hugging Face models by reading one snapshot every seven days starting on July 24th, 2024. While there is a wealth of metadata available, a few key fields are documented in Table 2.1.

[10] huggingface.co
[11] https://huggingface.co/datasets/cfahlgren1/hub-stats

Table 2.1: Key variables from Hugging Face Model Hub

Variable	Description
ID	Unique identifier for the model. Consistent over time.
Creation date	Date the developer uploaded model weights to Hugging Face.
Downloads (rolling)	Number of times the model was downloaded in the last 30 days.
Downloads (cumulative)	Total number of downloads since creation. Only available starting February 2025.
Likes	Total number of user "likes" accumulated since creation. Users click a button to express interest; no download or usage is required.
Trending score	Platform-calculated popularity score incorporating downloads, likes, recency, and other factors. Determines model ranking on the Hugging Face models page. Exact algorithm is not publicly available.

Note: Data retrieved from weekly snapshots of the Hugging Face Model Hub, July 2024 to February 2026. See Section 2 for details.

Each model has a unique identifier and a creation date recording when the developer first uploaded the model weights. Downloads record requests to Hugging Face's servers for specific model files. We observe two measures of downloads. The first is a rolling 30-day count available throughout our sample. The second is a cumulative total available only from February 2025 onward. Using these two measures, we impute cumulative downloads and use the resulting series to construct weekly downloads for the full sample. We detail this approach in Section 2.2. Likes function similarly to social media engagement: users click a like button and the platform displays a running total. Users can only like a model once and are not required to download or use a model before liking it. Finally, Hugging Face assigns each model a trending score that determines its visibility on the platform's main page.[12] Conversations with Hugging Face employees indicate that the trending score depends on downloads, likes, recency, and other factors, but the exact algorithm is not public and may have changed over time. We observe each of these variables on a weekly basis.

2.1 Incentives to Release Open-Weight Models

> "We have been having extensive discussions around open source strategy. We will discuss it more at our next board meeting, but one thing we'd like to do soon is to create a language model with the approximate capability of GPT-3 that can run locally on consumer hardware and release that. We'd like to do it soon, before Stability or someone else does. In general, we think this helps discourage others from releasing similarly-powerful models, and makes it harder for new efforts to get funded."
>
> — Sam Altman, Email to OpenAI's board, October 1, 2022. Released in Musk v. Altman (2026)[13]

Given that Hugging Face downloads do not directly generate revenue for developers, and that releasing a model's weights makes it easier for other developers to build potentially competing models, one natural question is why developers make their model weights public. Some developers cite scientific or altruistic motives, like advancing AI research.[14] Firms and individual developers also face a diverse set of economic incentives to release open-weight models, and these incentives are not mutually exclusive. For individuals,

[12] https://huggingface.co/models
[13] Reported in Simon Willison's Webblog
[14] The Allen Institute, for example, focuses on open-weight models Allen Institute research principles

many of the career-based incentives associated with developing open-source software, established in Lerner and Tirole (2002), may also apply to open-weight model development, with model performance serving as a signal of developer competence. For firms, releasing open-weight models can expand demand for their complementary goods and services. NVIDIA, for example, is a major contributor of open-weight models, and profits from selling GPUs used for model training and inference.[15] (This fits into a category of strategies sometimes called "commoditizing complements"). MoonshotAI, the developer of the popular Kimi models, releases model weights while also selling hosted inference through their API and offering enterprise contracts.[16]

Firms may also face dynamic incentives to release open-weight models. For AI labs that are not yet at the frontier, releasing their current models may allow them to learn from how people use and build on their models, which in turn can inform their future model development decisions. Firms can also use the success of their open-weight models to attract investors.[17] Firms' dynamic incentives may also be based on how releasing open weight models impacts their competition. The email at the start of this subsection suggests that producing a capable open-weight model may reduce the expected returns to potential rival releases and entry, thereby deterring competition. This particular type of strategy is sometimes called "preemption." Related mechanisms work by raising rivals' costs or otherwise lowering their current profits, meaning rivals have fewer resources for R&D impacting future competition. Coverage of Meta's release of Llama could erode the market shares of Meta's rival closed-model providers.[18] These examples, though not exhaustive, point to an extensive set of economic incentives shaping the open-weight AI market. Many of these incentives depend on the relative popularity of the model releases, meaning that the outcomes we consider, like model downloads can be helpful in understanding firms' expected returns.

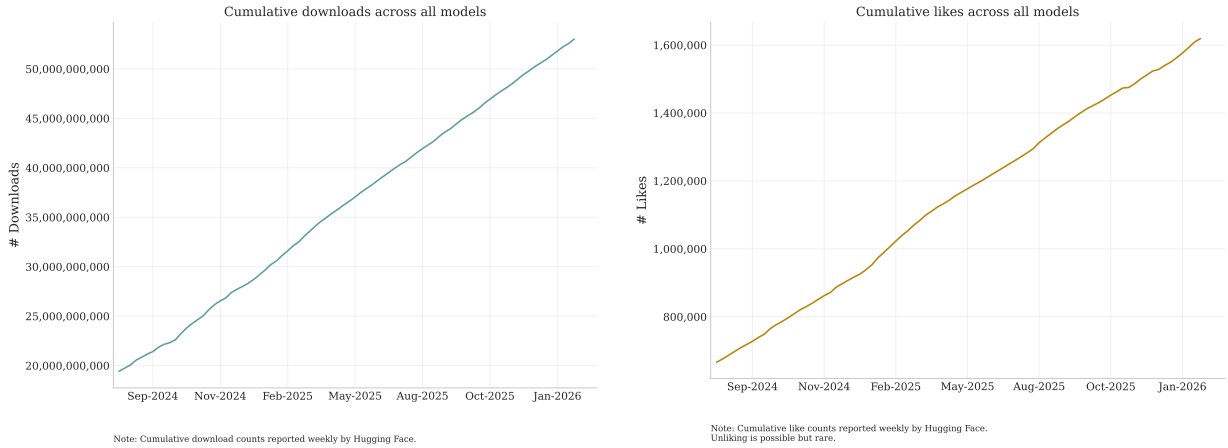
2.2 Model Downloads

We use model downloads on Hugging Face as our main outcome of interest. Hugging Face records downloads based on requests to their servers for designated files in a model repository.[19] We think of downloads as an upstream measure of demand because downloading a model generally precedes using the model for inference. Our data include every model on Hugging Face, so we observe all downloads recorded on the largest platform for hosting and distributing open-weight models. This gives us an especially comprehensive view of upstream demand for most of the world.[20] Figure 2.1 plots cumulative downloads and likes over time for the economically relevant models we describe below. By the end of our sample, models on Hugging Face had been downloaded more than 53 billion times.

The first challenge related to using download data is practical: Hugging Face does not directly report weekly downloads, so we construct a weekly panel from the available information. Hugging Face reports rolling 30-day downloads throughout our sample and cumulative downloads beginning in February 2025. For our analysis, the main problem is that the rolling measure obfuscates the timing of downloads. We use both the rolling and cumulative counts to construct the implied weekly downloads for each model from July 24, 2024 to February 11, 2026. We detail this process in Section A.1. Our approach addresses limitations of the

[15] See NVIDIA's Hugging Face developer page.
[16] See Moonshot AI's Hugging Face developer page and the Kimi API platform.
[17] Some of the top developers, including DeepSeek, have recently been conducting funding rounds. (See WSJ "AI startup deepseek poised to reach 74 billion valuation", August 27th, 2026).
[18] See Katie Paul, "Meta opens AI model to commercial use, throwing nascent market into flux," *Reuters*, July 18, 2023.
[19] For details on how Hugging Face counts downloads, see https://huggingface.co/docs/hub/en/models-download-stats.
[20] One important exception is mainland China, where Hugging Face is inaccessible without a VPN and ModelScope is a major domestic alternative.

Figure 2.1: Downloads and likes over time for economically relevant models



publicly available download measures noted in prior work.[21] We similarly construct weekly likes from the reported cumulative totals.

The second challenge is conceptual: a download is not necessarily a direct measure of downstream use or demand in general. The relationship between model use and downloads may vary across models, users, and inference providers. For example, a user might download a small model one time locally and then use the model for inference repeatedly. Alternatively, a user repeatedly deploying the model with third-party compute might use a container that downloads the same model at the start of each run.[22] We address this measurement issue by focusing much of our analysis on proportional differences in downloads rather than their raw levels. In Section 4.2.3, we detail a modest set of conditions under which our estimates can be more broadly interpreted as changes in overall model demand.[23]

2.3 Sample Selection of Economically Relevant Models

Hugging Face currently hosts nearly three million open-weight models. Anyone, including firms, professional developers, academics, and hobbyists, can upload a model to Hugging Face, and the model remains stored on the platform unless its creator removes it. This means that many of the models on the platform receive few or no downloads. Including these models could introduce noise and obscure the true nature of model lifecycles and demand for AI. We restrict our sample to the top 108,074 models ranked by lifetime downloads, which we call *economically relevant*. These models account for 99.5% of all downloads on Hugging Face (Figure A.2).[24]

We create the sample of economically relevant models by calculating the total downloads each model receives over our sample and dropping models that are never downloaded. Of the 3,054,279 unique models in our data, 1,462,212 are never downloaded.[25] Figure 2.2 plots the distribution of cumulative lifetime downloads across the remaining models that receive at least one download. While the median model receives

[21] For example, Osborne et al. (2024) note the limitations of the reported download counts and find that likes are more strongly correlated than downloads with model usage in Hugging Face Spaces.

[22] AI models can also be dependencies in software packages, meaning they are downloaded whenever a user installs the package.

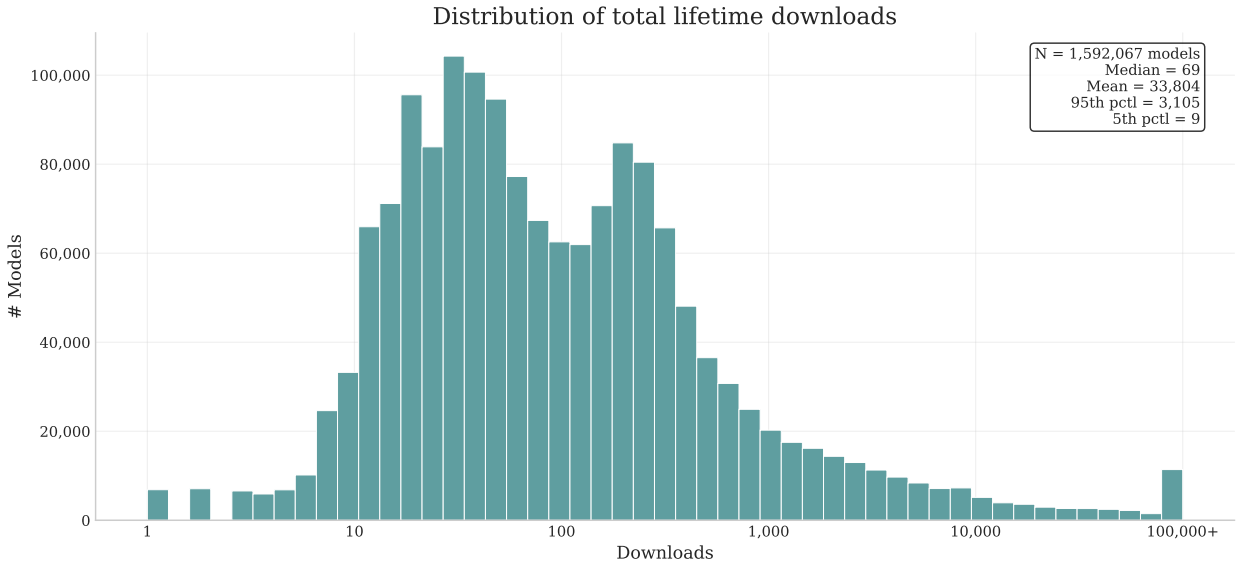
[23] We also compare downloads with token usage on OpenRouter in Figure A.1. While token usage is a direct measure of downstream inference, OpenRouter data only covers a small share of global inference.

[24] Even the 108,074th most popular model has over 2,000 lifetime downloads. To ensure our estimates are robust to the choice of cutoff, we include initial popularity bin controls throughout much of our analysis and repeat the estimation for more popular subsamples.

[25] Since we observe a panel of models going back to 2024, our data include models that are no longer hosted on Hugging Face.

69 downloads, there is a long tail of models that receive larger numbers. For example, DeepSeek-R1 has accumulated 46,818,923 downloads since its launch in January 2025, and Nvidia's Nemotron 3 Ultra has received 757,477 downloads since its release in June 2026.[26] For the remainder of the paper, we restrict our analysis to economically relevant models, except where explicitly noted.

Figure 2.2: Distribution of total downloads



Note: Plots the distribution of maximum cumulative downloads observed for each model. Excludes 1,462,212 models that are never downloaded.

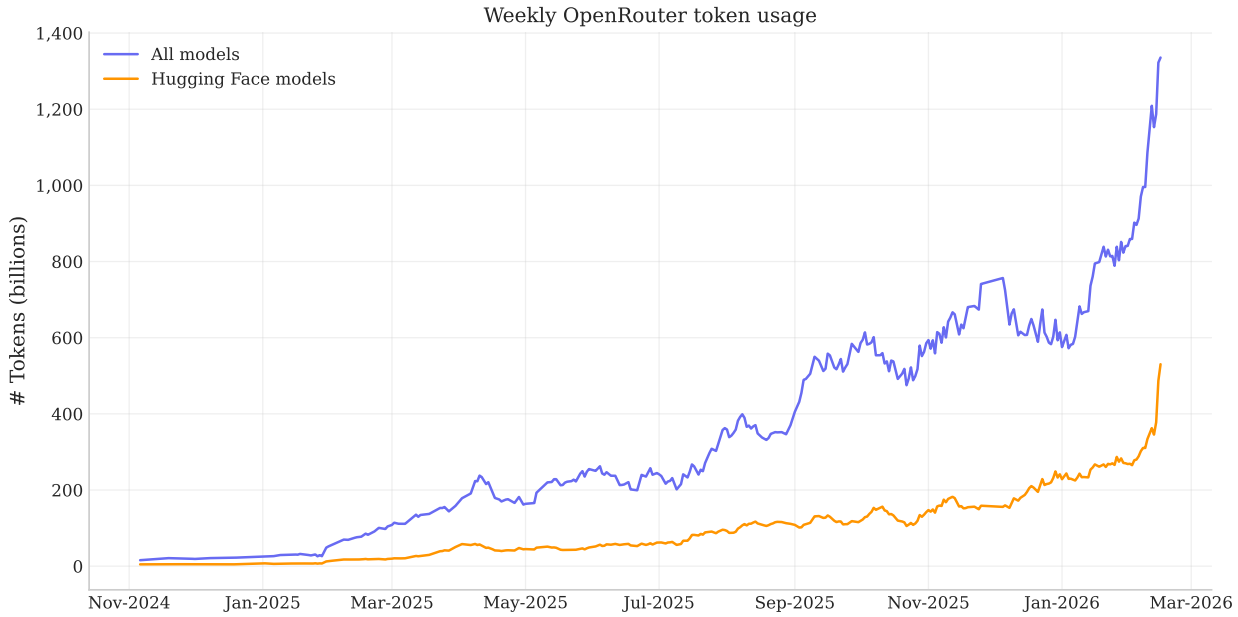
2.4 Inference Data

To measure downstream model usage, we collect data from *OpenRouter*, a large intermediary platform that connects users to AI inference providers.[27] During our sample period, OpenRouter provides API access to 717 models, including both open-weight and proprietary models.[28] On OpenRouter users can directly choose their model and inference provider or use routing services that automate selection of models and providers. We observe daily input and output tokens for each model beginning in November 2024.[29]

We use the OpenRouter data primarily to measure usage obsolescence in Section 5.2. Because the platform includes both open-weight and proprietary models, we can measure whether an open-weight model continues to attract meaningful usage relative to the broader set of models available on OpenRouter. Open-weight models account for a substantial share of activity on the platform. Figure 2.3 plots total token usage on OpenRouter and usage by the models we match to Hugging Face. The matched models account for 27.5% of total tokens during our sample. Among models observed in both data sources, cumulative token usage is positively correlated with cumulative Hugging Face downloads (Figure A.1).

[26] For more information on these models see model cards: "DeepSeek AI's DeepSeek-R1" and "Nvidia's Nemotron 3 Ultra"
[27] openrouter.ai
[28] This number includes models that were previously available, but no longer on the platform.
[29] Tokens are the units into which model inputs and outputs are divided. We measure usage as the sum of input and output tokens.

Figure 2.3: OpenRouter model usage



Note: Plots daily sum of prompt and completion tokens. Values on missing dates are filled in with linear interpolation.

2.5 Model Capability Benchmarks

We collect data on model benchmark scores from *Artificial Analysis*, an independent firm that has tracked model performance since 2023.[30] One would expect model demand to depend on model capability, relative to other available models. Benchmarks provide a set of questions or tasks with clearly defined answers. Models are evaluated by the number of questions they answer correctly. For example, the Google-Proof Q&A Benchmark contains multiple-choice questions about physics, biology, and chemistry and tries to measure a model's ability to do genuine scientific reasoning, rather than searching the web (Rein et al., 2023). For each of more than 700 models, Artificial Analysis collects price, latency, and scores on a variety of standard benchmarks that test coding, math, tool use, and related tasks. These data are aggregated into an *Intelligence Index* for each model and more specific Coding, Math, and Agentic indices. We collect these index scores as a measure of technical quality. Table 2.2 shows the top models in each index.

Table 2.2: Artificial Analysis Benchmark Indices

	All models			Hugging Face models		
Category	Models	Highest score	Best model	Models	Highest score	Best model
Intelligence index	594	59.9	Claude Fable 5	338	51.1	GLM-5.2
Coding index	533	78.3	GPT-5.6 Sol	305	68.8	GLM-5.2
Math index	369	99.0	GPT-5.2	220	96.7	DeepSeek V3.2 Speciale
Agentic index	413	54.0	GPT-5.6 Sol	238	43.1	GLM-5.2

Note: Data as of July 9, 2026.

[30] artificialanalysis.ai

2.6 Elo Scores

Benchmarks capture how well models perform on domain-specific tasks, but users may also care about aspects that are difficult to measure, like the tone of a model's responses. To obtain a measure of model performance that is more directly related to user preferences, we collect "Elo scores," which is a composite score based on how well a model performs in a series of pairwise comparisons with competitor models. We collect Elo scores from both Artificial Analysis and from *Arena*, one of the most widely used platforms for evaluating large language models through direct human judgment.[31] To calculate the scores, both platforms host "model arenas" in which a user compares the outputs of two models given a single prompt and chooses their preferred result.[32] Figure A.4 depicts a typical contest: given a text prompt, two models generate images, and the user does not observe the identity of either model. The user then selects the image they prefer. Thousands of users make these pairwise comparisons, and the platform aggregates the resulting votes into the Elo score for each model.[33] The evaluations come from real users, so they directly capture each model's quality as perceived by a broad user base.

Arena maintains leaderboards across categories including overall performance, coding, mathematics, and instruction following. Artificial Analysis leaderboards focus on media models, including image generation, video generation, and text-to-speech. Table 2.3 shows the number of models in each category and the number we are able to match to our main Hugging Face sample. Note that both platforms include proprietary models in their arenas which are not present on Hugging Face. The Elo score data complement the benchmark data.

Table 2.3: Elo scores

	All models			Hugging Face models		
Category	Models	Highest score	Best model	Models	Highest score	Best model
Artificial Analysis						
Text to image	156	1339.0	GPT Image 2	41	1226.2	Cosmos3-Super-Text2Image
Image editing	71	1291.4	Riverflow 2.0	24	1230.0	HunyuanImage 3.0 Instruct
Text to video	84	1218.8	Dreamina Seedance 2.0 720p	12	973.5	LTX-2.3 Fast
Image to video	80	1194.6	Dreamina Seedance 2.0 720p	9	953.7	LTX-2.3 Fast
Arena.ai						
Text	376	1563.6	Claude Fable 5	124	1487.8	GLM 4.7
Vision	136	1366.8	Claude Opus 4.6	24	1211.1	Qwen3 VL Thinking
Search	31	1238.5	Claude Fable 5	0	–	–
Document	29	1500.1	Claude Fable 5	0	–	–
Webdev	92	1653.9	Claude Fable 5	6	1440.1	GLM 4.7
Text to image	73	1362.8	GPT Image 2	8	1167.9	Hunyuan Image 3.0
Image edit	53	1455.8	GPT Image 2	7	1240.6	Qwen Image Edit
Text to video	41	1527.5	Gemini Omni Flash	1	1170.0	Hunyuan Video 1.5
Image to video	42	1473.7	Dreamina Seedance 2.0	1	1196.1	Hunyuan Video 1.5
Video edit	7	1377.1	Dreamina Seedance 2.0	0	–	–

Note: Data as of July 8, 2026.

The benchmark scores are based on capability for a coarse set of domains, while the Elo scores are based on user preferences and cover a more granular set of domains. By combining benchmarks and Elo scores across domains, we are better able to capture when a model is technologically relevant and when it has been superseded by a newer model.[34]

[31] See arena.ai. Previously named LMArena and Chatbot Arena.

[32] On Artificial Analysis, the platform pre-selects the prompt and the user only chooses their preferred output. On Arena, the user inputs their own prompt. Both choices could bias the Elo scores, so we combine the data from both platforms.

[33] Arena uses the Bradley-Terry model for mapping head-to-head preferences into model ranking scores. For details, see "Arena AI FAQ".

[34] More specifically, we focus on the domain-benchmarks and domain-Elo scores, where the model initially performs best. For details on how we aggregate these scores, see Section 5.1

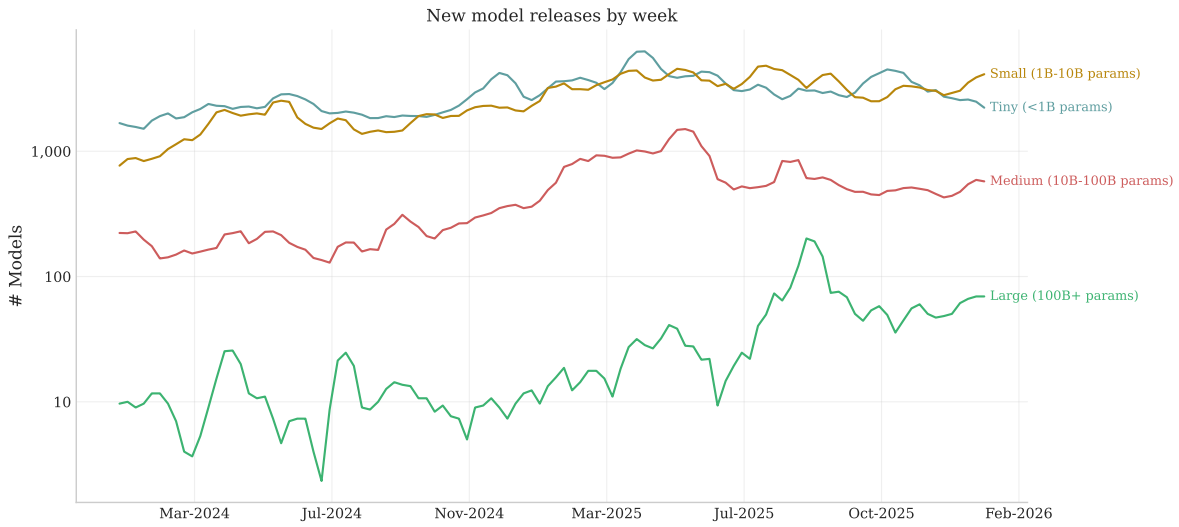
3 Birth

This section establishes several basic facts about how developers create and release open-weight models. We are particularly interested in whether developers make investments model by model or across portfolios, and in how open-weights allow other developers to build on those investments. These choices affect the returns to training, the products brought to market, and how those products compete. We first document overall release patterns and product differentiation. We then study batch releases and the creation of descendant models by original developers and third parties.

3.1 Model Entry and Model Sizes

Model entry follows different patterns by model size. Figure 3.1 plots the volume of model releases by size, and Figure B.1 plots the same for the sub-sample of economically relevant models. For both samples, the tiny and small models are released more frequently than larger models. One might expect this pattern, as training compute costs and other significant fixed costs tend to increase with size. Optimally training a 100 billion parameter model incurs compute costs orders of magnitude higher than a 100 million parameter model (Rosenfeld et al., 2019; Kaplan et al., 2020; Hoffmann et al., 2022). At the same time, this scaling of compute and size is one of the primary approaches to improving model capability (Brown et al., 2020; Wei et al., 2022; Owen, 2024; Mertens et al., 2026), making model size an important measure of vertical differentiation.[35]

Figure 3.1: New model releases by week



Note: new models are models uploaded to Hugging Face for the first time in a given week. Plots 3-week rolling average.

For tiny, small, and medium models, the pace of new model releases increased from March 2024 to March 2025, and has since stabilized. For large models, the rate of releases is noisier as large model releases are relatively rare, but has trended up since November 2024. This size breakdown of model releases highlights the volume of new model releases each week and shows that these rates substantially differ by model size. However, it is worth noting that these releases are interconnected. New large models can facilitate the

[35]The cost scaling applies to training original models. These relationships could be different when building on top of a pre-trained model, for example with fine-tuning, which we cover in Section 3.3.

development of smaller models both indirectly through distillation or augmenting development (e.g. coding), and directly through model descendants, which we cover in Section 3.3.

3.2 Model Portfolios and Product Differentiation

The AI market is one of differentiated goods, so it is important to understand, given the breadth of models available, how these models differ and ultimately compete. As highlighted above, more than 1,000 open-weight models are released each week and these models can differ in size and, as a result, capability. This is one of a number of dimensions of product differentiation that fall under two categories:

1. **Vertical Differentiation:** Variation in capability and latency. Improving capability is not free: it either requires a larger model in terms of parameter count, additional training, or more inference, all of which can increase user-costs. As with many vertically differentiated goods, some users might prefer the lower quality product due to differences in these user-costs.

2. **Horizontal Differentiation:** Variation in hardware compatibility, licensing, alignment, and domain-specific capabilities. These are generally matters of taste, where users have different preferences for these features.

Product differentiation is important to understand how models compete, but also how a firm strategically builds their product portfolio. Figure 3.2 illustrates the open-weight model portfolios of four developers: DeepSeek, Meta, IBM and OpenAI. For each developer, there are clear examples of both vertical and horizontal differentiation. Vertical differentiation is easiest to see with different model sizes. OpenAI's gpt-oss-120b and gpt-oss-20b are vertically differentiated by capability. The larger, gpt-oss-120b is generally more capable than gpt-oss-20b, but more costly and slower to use (OpenAI et al., 2025).[36] IBM's 2B parameter versions of Granite 3.3 Speech and Granite 3.3 Vision illustrate horizontal differentiation, as each model has its own domain specialization.[37]

[36] Many developers use a naming convention where they include a model's parameter count in its name. gpt-oss-120b is a 120 billion parameter model. Similarly, gpt-oss-20b is a 20 billion parameter model.

[37] Granite 3.3 Speech and Granite 3.3 Vision derive from the same base model and share the same training data, but they are fine-tuned to specialize in different tasks (speech-to-text versus image classification, in this case).

Figure 3.2: Model releases over time

3.2.1 Example Model Portfolios

Figure 3.2 highlights not only within-firm product differentiation, but also important patterns in portfolio composition and release timing. DeepSeek, Meta, and IBM frequently release *batches* of five or more models. While OpenAI's frontier models are closed, it maintains the popular "Whisper" text-to-speech models and open-weight GPT models. The typical batch from these developers includes two to four model variants that differ by parameter count and, within each parameter count, two or three variants differentiated by task specialization (e.g. language and vision). Size differentiation ensures the model appeals to users with different preferences for performance versus cost and with varying hardware limitations. Task differentiation provides specialized models for specific needs. The individual model variants within a batch are released close to one another in time, while the gap between batches is larger. Meta's Llama 3 and LLama 4 were released about one year apart, as were IBM's Granite 3.0 and Granite 4.0. Within these major model generations, there are smaller technological increments: Meta's Llama 3.1 launched a few months after Llama 3, and Llama 3.2 a few months later. The patterns we describe suggest major developers invest and strategize at the portfolio level rather than at the individual model level. Training a new model from scratch takes time, but the process yields multiple new products that can be released in the same batch, which alters the return to training and may affect the long-term strategies of developers.

One way developers produce batches of models is by building on top of a specific model, which we call a *source* model. This commonly involves starting with a "base" model, the fundamental next-token-predictor version, and fine-tuning it with reinforcement learning to produce an "instruct" version more suitable for answering user prompts. Developers can also use their base model to fine-tune smaller models. DeepSeek, for example, used synthetic data from its powerful DeepSeek-R1 to fine-tune several versions of Llama and Qwen. We refer to these variations on the source model as *descendants*. Creating descendants is not trivial, but it is far less costly than training a large model from scratch, so the process allows developers to turn a single training run into multiple products, increasing the return on their initial training investment. There are additional methods available to build one model using another, which we detail in Section 3.3.

We have described two patterns in model releases. First, models are often released not as standalone products but in groups we call *batches*. Second, the batch release phenomenon partly derives from developers' ability to build on top of models to create variants called *descendants*. These are only descriptive examples from major developers, so we now turn to characterizing model release patterns more broadly.

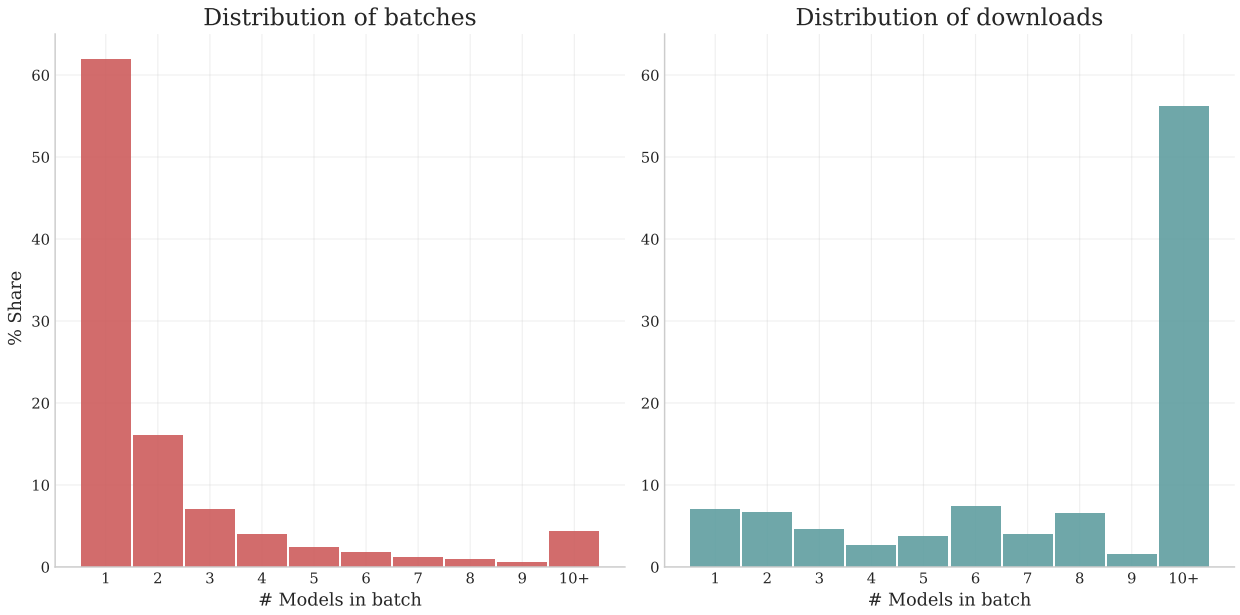
3.2.2 Batch Releases

Above, we highlight a pattern of diverse model portfolios and batch releases for a select set of developers. Next, we analyze these batch release patterns across all developers. These patterns are important for understanding the timing of supply-side production incentives. While conventional wisdom focuses on how one model generation impacts the next, these batch release strategies reveal important clues about spillovers between models within the same generation.

To analyze patterns in batch releases, we assign all models in our sample to batches by constructing 30 day windows around their release dates, as detailed in Section B.1. Figure 3.3 plots the distribution of batch size (the number of models in a batch) and of model downloads accruing to models in batches of different sizes. While the typical model is released on its own, i.e. not in a batch, most downloads on the platform go to models released in large batches.

The contrast between the two distributions illustrates a recurring theme of concentrated model use. Even

Figure 3.3: New model releases by batch size



Note: Batches are groups of models released by the same author within 30 days of each other.
Left: share of models in batches of each size. Right: share of total platform downloads accruing to models in each batch size.

in our sample of economically relevant models, the typical model is released alone, but most downloads go to a small number of models that were released as part of major batches (for example, the many concurrently released variants of DeepSeek-R1).

Given the concentration in downloads among models released in large batches, it is worth noting the economic intuition for why one might observe these batch release patterns. Both demand- and supply-side economic forces might shape the batch release patterns we observe. On the demand side, we have the typical incentives of multi-product firms developing portfolios of differentiated goods to meet diverse user preferences (Wollmann, 2018). On the supply side, economies of scale and scope could allow training investments in one model to spill-over to other models in the same batch. In particular, one model can be directly used as the starting point for building another, as we describe in Section 3.3. Additionally, within the same firm, common architecture, common training data, model distillation, and shared organizational knowledge could generate cost synergies within and across model batches.

3.3 First- and Third-Party Descendant Models

Unlike proprietary models, the core architecture of an open-weight model is publicly available. This means developers can build new models on top of the original, improving its performance or adapting it for specific tasks. We refer to the original model as the "source model" and to the adapted version as the "descendant model." Both the original author of the source model and third-party developers can create descendants. Developers self report these relationships on Hugging Face.[38] There are four types of relationships between source and descendant models based on how the descendant was created:

[38] On Hugging Face the relationships are reported in the "model tree" section of the model card.

1. **Quantize**: reduces numerical precision in the underlying weights. This reduces inference costs and memory use at the cost of model capability.[39]

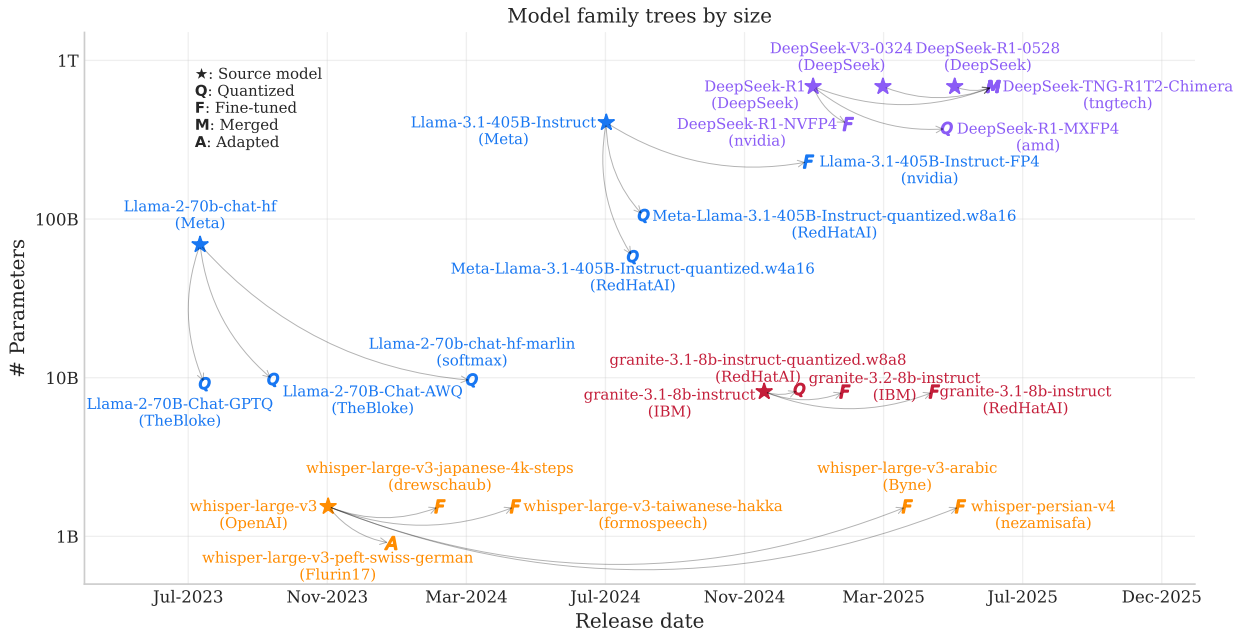
2. **Fine-tune**: use additional data to adapt the source model to a specific setting.

3. **Merge**: combine multiple source models into one descendant.

4. **Adapt**: adjust the hardware compatibility or add specific capabilities to the source model.

Figure 3.4 illustrates a subset of the model trees for select example models from Meta, OpenAI, IBM, and DeepSeek. We note the timing of the source and descendant releases (horizontal axis), the size of each model (vertical axis), and the type of descendant (marker type). For example, third party developers quantized Meta's Llama models both within days of release and months later. OpenAI released a popular text-to-speech model "Whisper Large V3" in late 2023. Other developers subsequently fine-tuned the English-focused source model on data in other languages, with descendants continuing to appear well into 2025. Major developers also are involved in creating descendants of each other's models, for example, NVIDIA creating descendants of Meta's Llama models.

Figure 3.4: Model family trees

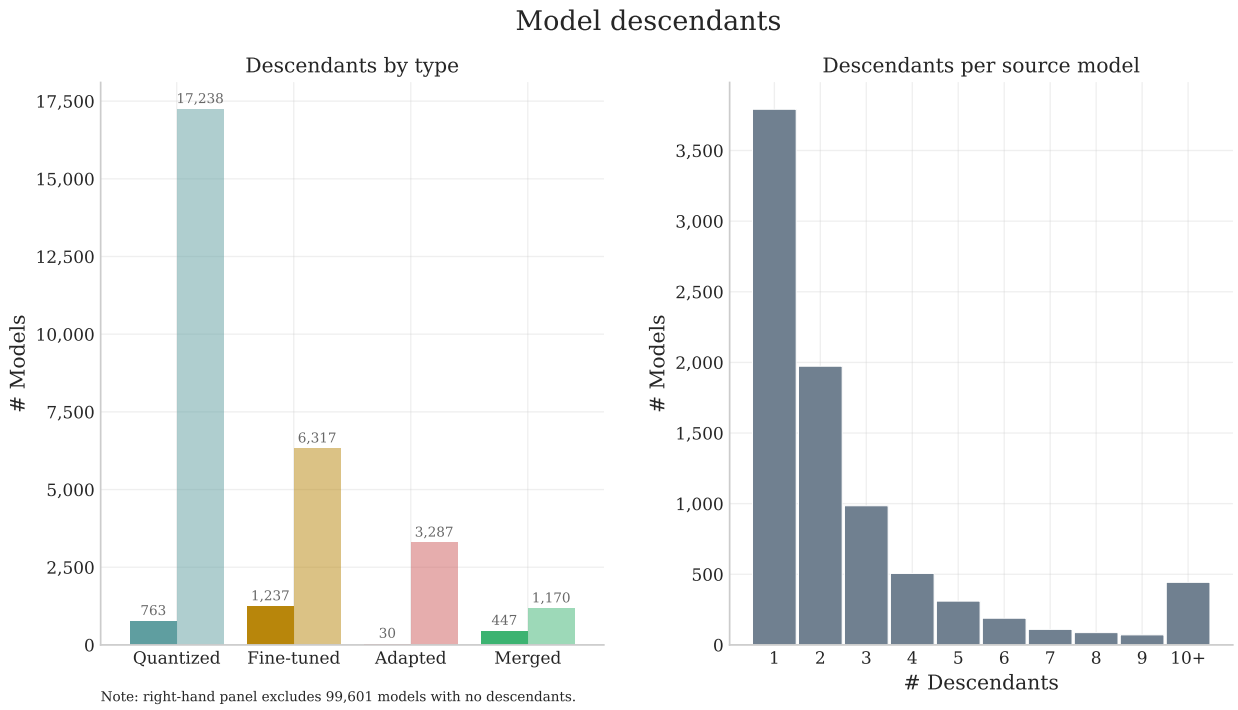


Note: Does not show all descendants.
Quantizing reduces the size of the source models by reducing numerical precision.
Fine-tuning adjusts the source model to perform well on specific tasks.
Merging combines multiple source models into one.
Adapting adds specific functionalities to the source models.

[39] Hugging Face provides a useful explanation of quantization here https://huggingface.co/docs/transformers/en/main_classes/quantization

We document descendant relationships overall in Figure 3.5, which characterizes the types and number of economically relevant descendants on Hugging Face. Third-party descendants are more common than same-author descendants. Quantization is by far the most common method for creating descendant models. Of the economically relevant models, 30,191 are descendants (27.9%).[40] Only 8,473 unique models in our data (7.83%) ever have a descendant.

Figure 3.5: Descendant model types



4 Life

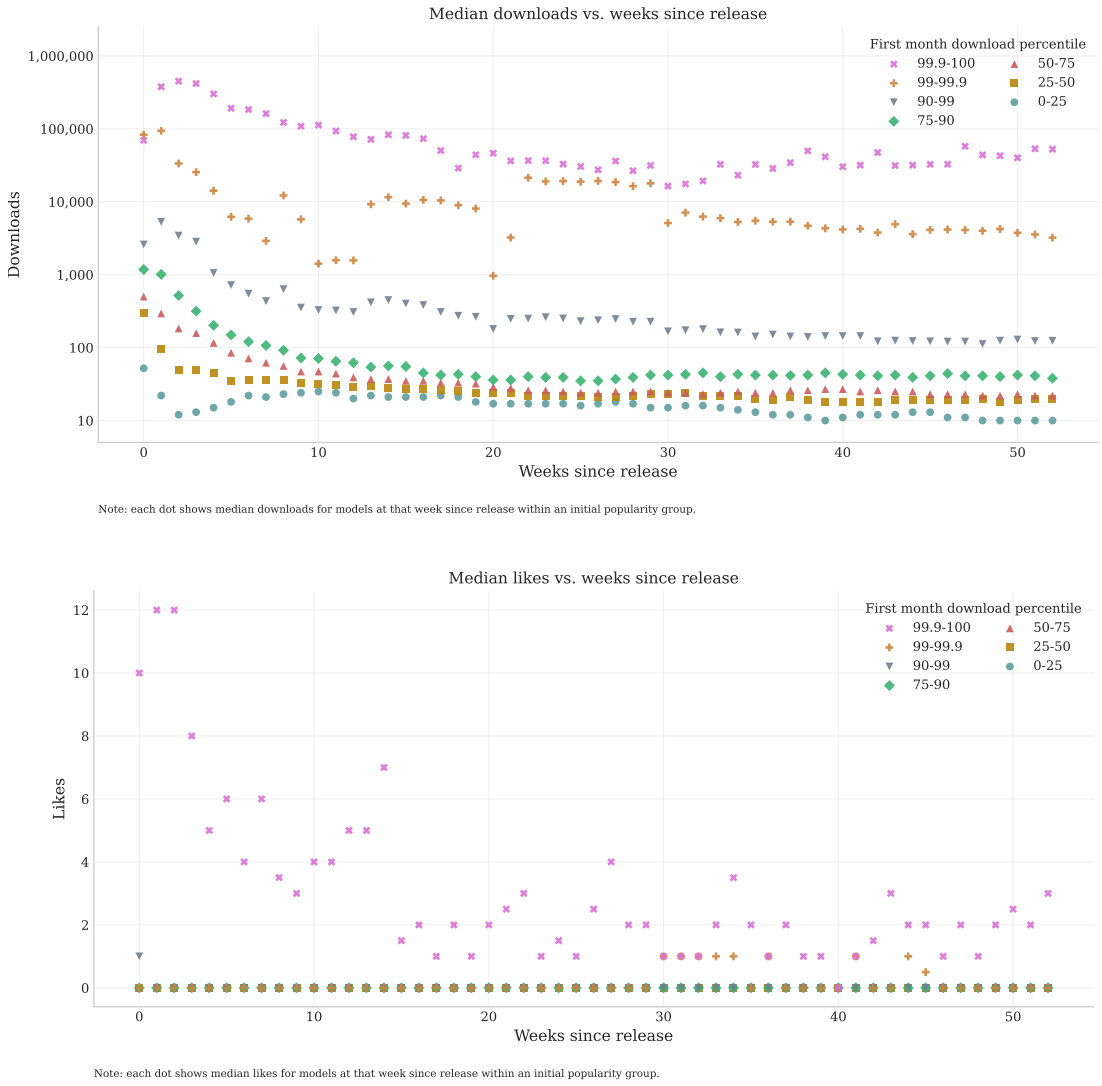
This section focuses on the "life" phase of the AI model lifecycle. We first look at descriptive patterns in downloads and likes as models age. We find concentration in downloads that persists over time, and that downloads follow different patterns depending on a model's initial popularity. We also document notably different relationships for likes versus downloads. Then, we provide an econometric framework that, under a modest set of assumptions, allows us to estimate changes in *demand* and changes in *attention*, based on proportional changes in observed downloads and likes, respectively. We use this framework to estimate the spillover effects on demand and attention from new model releases from the same developer, as well as from first- and third-party descendant models. Our findings inform developers' strategic decisions, as optimal release timing depends on how long they expect demand to persist. Moreover, spillovers from subsequent model releases change the returns to model development, which could alter developers' optimal choice of model portfolios, and impact decisions to open-weight their models.

[40] Since these relationships are self reported, 27.9% can be thought of as a lower bound.

4.1 Model Lifecycle

We describe the model lifecycle using weekly downloads and likes during each model's first year. As models experience different levels of initial popularity, one might also expect different trajectories based on initial success. To capture this, we group models into bins based on their initial popularity, specifically their percentile rank by number of downloads in the first month after their release.[41] Section C.2 repeats this analysis, instead binning models by parameter count.[42]

Figure 4.1: Median model lifecycle by initial popularity



The top panel of Figure 4.1 reports the lifecycle in terms of downloads. First-month downloads are highly concentrated, and the concentration persists. Among models we observe at release, 12% of all downloads are attributed to new models in their first month. Of those early downloads, the top 0.1% of models account for

[41] We specifically use the first four weeks after release to define the first month. For the remainder of the paper, we refer to this initial period as the first month.

[42] Because popularity groups are defined using downloads during the first month, differences across groups during this initial period are partly mechanical. The subsequent trajectories show whether those initial differences persist.

23%, the top 1% account for 53%, and the top 10% account for 85%. The models in each age bin see their highest median downloads in the first weeks then median downloads tend to decline, but at different rates. The median model receives 68% fewer downloads ten weeks after release than in its release week. The decline in downloads is much slower among initially popular models. The top 1% of models see thousands of weekly downloads for a full year after release. We can see this in differences in download shares as well; models in the top percentile receive approximately 600 times as many downloads as the median model at release and approximately 900 times as many after six months.

The bottom panel of Figure 4.1 shows that likes are sparse and follow notably different patterns than downloads. The median model receives no new likes in any week, including its release week. Even among models in the top 0.1%, the median number of likes in the release week is only ten, and new likes fall quickly toward zero. This highlights that for most model-weeks, likes provide little signal of contemporaneous demand. In nearly every week, the median number of likes is zero for every initial-popularity group outside the top 0.1%, even though median downloads remain positive and differ substantially across those same groups. Likes are also more concentrated near release than downloads, so the relationship between the two measures changes over the model lifecycle. This also suggests that neither cumulative nor weekly model likes are good proxies for cumulative or current demand across models of different ages. These patterns suggest that likes primarily record attention around release, while downloads continue to capture activity over a much longer period. For our remaining analysis, we model likes and downloads separately, mapping likes to attention, and downloads to demand and unobserved usage.

Separately identifying model age effects from cohort and time effects raises an age–period–cohort identification problem common in lifecycle analysis.[43] Because model age equals calendar time minus release time, with release time absorbed by model fixed effects, a decomposition of the three sets of fixed effects is not identified without additional restrictions (McKenzie, 2006; Schulhofer-Wohl, 2018). We do not have a compelling basis for choosing such restrictions in this setting, so we present the medians above as a descriptive exercise. Section C reports regression-adjusted profiles for comparison, but we do not interpret those coefficients as pure effects of model age.[44]

4.2 Spillover effects

Next, we investigate the relationships among new and existing models throughout the lifecycle. We first discuss the potential channels through which *spillover effects* from new model releases could impact demand and attention. We then develop an econometric framework mapping proportional changes in downloads to proportional changes in demand, and we use this framework to estimate changes in demand associated with the releases of popular related models from first- and third-party developers. The direction of spillovers, and the implied mechanisms, inform the returns to model development, which could alter developers' optimal choice of model portfolios, and impact their decisions to open-weight their models. This also impacts how researchers and policymakers structure their analysis, which depends on model developer behavior and incentives.

[43] The language for this problem uses the term "cohort," which typically refers to a group fixed effect based on the same treatment date. In settings with model fixed effects, each model is essentially its own cohort.

[44] In our spillover and obsolescence analysis, model, calendar-week, and model-age-by-initial-popularity fixed effects enter as controls rather than parameters of interest, while our parameters of interest rely on other identifying variation.

4.2.1 Spillover Mechanisms

A subsequent model release can affect demand for an existing related model, but the direction of the effect is not obvious. Different markets can be helpful in highlighting these competing mechanisms. One concern for developers might be that new related models—their own or from a third party—could cannibalize demand from their related source model; this effect is often called business stealing or substitution. A new mobile phone draws sales away from the previous generation, and similarly, new car models make the inventory of the previous model less competitive. Business stealing might be an especially notable concern in the open-weight environment since third parties can build descendant models from a source model, as described in Section 3.3.

In the open-weight market, there are other mechanisms worth considering including, *backward spillovers*, *market expansion*, and other effects driven by complementarity. Examples of backward spillovers can be seen in the music industry, where a new hit album draws listeners to an artist's earlier catalog (Hendricks and Sorensen, 2009).[45] Similarly, there is some evidence of spillovers across creators, for example, Schuster et al. (2019) find that songs sell more after they are sampled in a new recording.[46] Examples of market expansion can be seen in the pharmaceutical industry, where advertising for one drug raises demand for competing drugs in the same class (Shapiro, 2018; Sinkinson and Starc, 2019).[47] In the AI market, another possible complementarity occurs if users demand bundles of models, in which case a new related model could increase demand for the bundle that includes the source model.[48] Together, these mechanisms could have a net effect that is positive, negative, or zero. The direction and timing of the net effect offer clues as to which mechanisms exist and are most relevant in the AI market.

4.2.2 Related Model Releases

We focus on understanding net spillover effects from three types of relationships between a focal model and subsequent popular related releases. The type of relationship depends on who produced the subsequent release and if the subsequent model is a descendant of the focal model. Specifically, we categorize the relationships between a focal models and a related subsequent releases as follows:

1. Same-author non-descendant: a later model by the same author that is not a descendant of the focal model.
2. Same-author descendant: a later model by the same author that is a descendant of the focal model.
3. Third-party descendant: a later model by a different author that is a descendant of the focal model.

The first two relationships capture releases within a developer's portfolio. The third captures a different developer building directly on the focal model, a relationship enabled by open-weights.[49]

One might expect different net effects depending on the relationship. New models from the same developer could generate awareness of other products in the same portfolio, and similarly, a popular third-party release can bring attention to the source model. Models are imperfect substitutes, and the degree of business stealing

[45] Hendricks and Sorensen (2009) call this "backward spillovers" and attribute it to consumer discovery.

[46] Watson (2017) evaluates the same claim looking at Spotify streams, and Santos and Figueiredo (2024) report similar finding in a subset of cases for sampling, remixes, and covers using Google Trends as the outcome.

[47] Sinkinson and Starc (2019) decompose the effect into business stealing and market expansion, and find that the spillover comes mostly from new patients rather than from switching.

[48] Pharmaceutical markets provide an example of a specific type of this complementary. In combination therapies, promotion of one drug can increase demand for other drugs with which it is commonly prescribed (Liu et al., 2017).

[49] Source–descendant relationships are self-reported by model authors in the Hugging Face metadata. We use the reported source model and do not infer more distant ancestry.

could depend on the degree of differentiation. First- and third-party developers may make different choices about how to differentiate descendant models from their source models. The net effect may also differ both across these relationships and across releases of the same type. A focal model can experience several related releases. For example, the first popular descendant may generate more discovery than subsequent releases. The response may also depend on the popularity of the new model.[50] To account for this variation, our analysis in Section 4.2.3 allows the net effect to vary by relationship type, release order, and the popularity of the related releases. We also repeat the analysis by popularity of the focal models.

Examples Figures 4.2 to 4.4 provide descriptive examples of same author non-descendant, same author descendant, and third party descendant relationships. These examples highlight specific realizations of these relationships and help illustrate specific ways a related release could impact demand for the focal model. They also provide motivation for our econometric approach to estimating spillovers. While source-model downloads sometimes move around related model release dates, the plots also show that downloads are noisy, and that related releases may happen at high volume and in close succession.

Figure 4.2 provides an example of same-author non-descendant releases. It plots weekly downloads for several DeepSeek models around the January 2025 release of DeepSeek-R1, a reasoning model developed through supervised fine-tuning and reinforcement learning on DeepSeek-V3-Base.[51] DeepSeek-R1 ranked first in Hugging Face's trending score for several weeks following its release. Downloads of DeepSeek-V3 increased by an order of magnitude immediately following R1's release. Some V2 variants also exhibit similar immediate increases. One possible explanation is that attention to R1 brought users to DeepSeek's portfolio, including existing models better suited to other applications. We can think of this as descriptive evidence of a positive same-author spillover. However, this individual time series does not, on its own, establish spillover effects since we do not know counterfactual downloads in the absence of R1's release. Also, the impact of R1 might not be representative of same-author spillover effects in general.

Figure 4.3 illustrates same-author descendants. Both Qwen2.5-Coder-32B and QwQ-32B report Qwen2.5-32B as their source model, but they extend it in different directions. Qwen2.5-Coder-32B continues training on code and related data to specialize the source model for coding tasks. QwQ-32B uses supervised fine-tuning and reinforcement learning to add reasoning capabilities to the same 32-billion-parameter source model.[52] The reasoning descendant, QwQ-32B, was popular at release and reported performance competitive with leading reasoning models, including DeepSeek-R1, and its weekly downloads immediately after release were an order of magnitude greater than those of its source models, whose own weekly downloads slightly increased immediately following QwQ-32B's release.

Figure 4.4 illustrates third-party descendants of Black Forest Labs' FLUX.1-dev, a 12-billion-parameter text-to-image model. Third-party developers extended FLUX.1-dev along a number of dimensions. Examples include FLUX.1-Turbo-Alpha, a distilled adapter that reduces image generation to eight inference steps, Flux-ip-adapter-v2, which allows users to condition generation on an input image, and In-Context-LoRA which supports the generation of related sets of images, including film storyboards and other multi-image designs.[53] Other descendants specialize FLUX.1-dev for other applications and visual styles, including toy-like

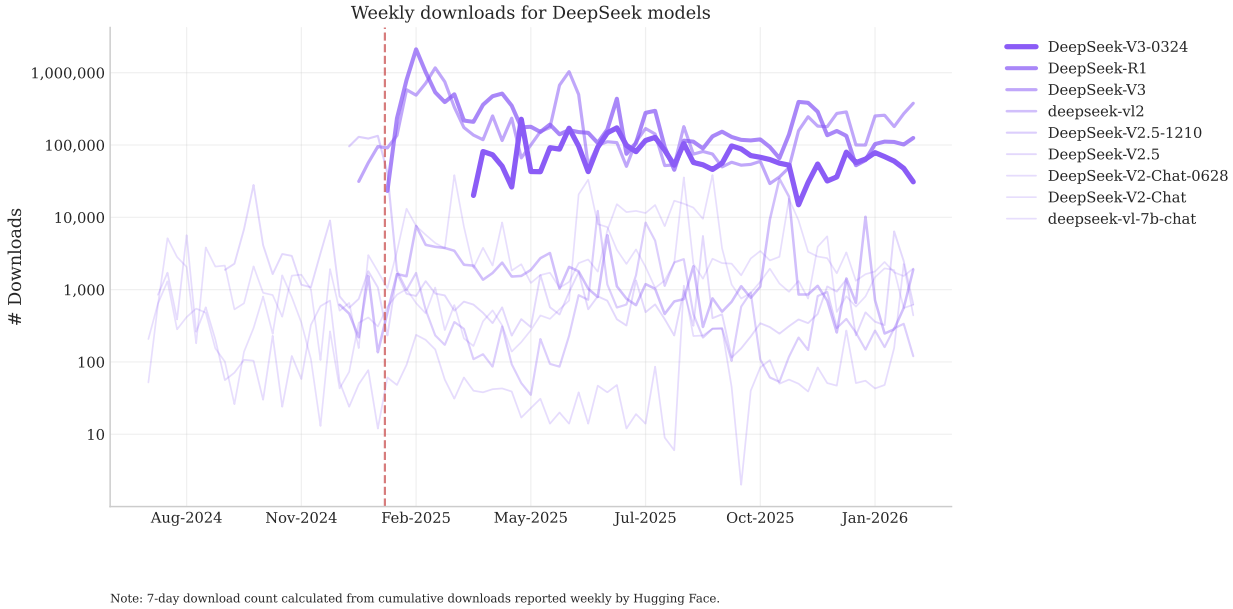
[50] We focus on popular new releases. A release that attracts little activity is unlikely to generate economically meaningful substitution or positive spillovers.

[51] See the DeepSeek-R1 model card. DeepSeek reports that R1 was trained from DeepSeek-V3-Base. Because we use the reported direct source model, R1 is classified as a same-author non-descendant relative to the DeepSeek-V3 and V2 repositories shown in the figure.

[52] See the model cards for Qwen2.5-Coder-32B and QwQ-32B. Both identify Qwen2.5-32B as their base model.

[53] See the model cards for FLUX.1-dev, FLUX.1-Turbo-Alpha, flux-ip-adapter-v2, and In-Context-LoRA.

Figure 4.2: Weekly downloads for DeepSeek models around the release of DeepSeek-R1



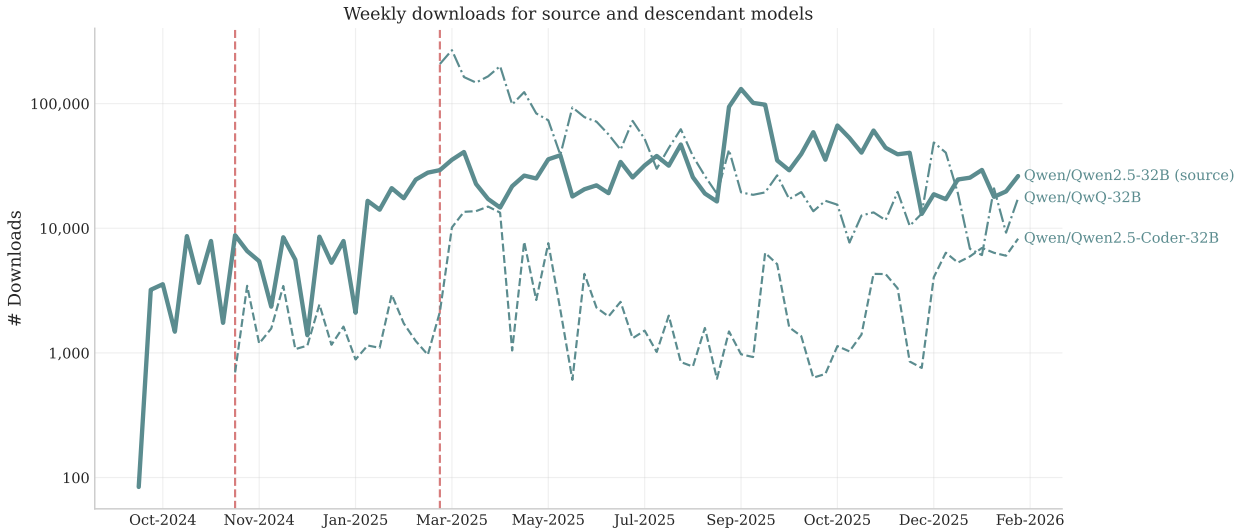
Note: 7-day download count calculated from cumulative downloads reported weekly by Hugging Face.

three-dimensional renders and pencil sketches.

Some of these descendants are adapters that are by definition complements to Flux.1-dev, since their use also requires FLUX.1-dev (the source model). For example, using alimama-creative/FLUX.1-Turbo-Alpha requires loading both its own and FLUX.1-dev's model weights. Third-party releases begin shortly after FLUX.1-dev's launch and continue many months later, while the source model remains widely downloaded throughout the period. Figure 4.4 also shows substantial variation in the popularity of descendants. Extensions related to speed, image conditioning, and multi-image generation receive thousands or tens of thousands of weekly downloads, while many style-specific descendants form a smaller long tail.

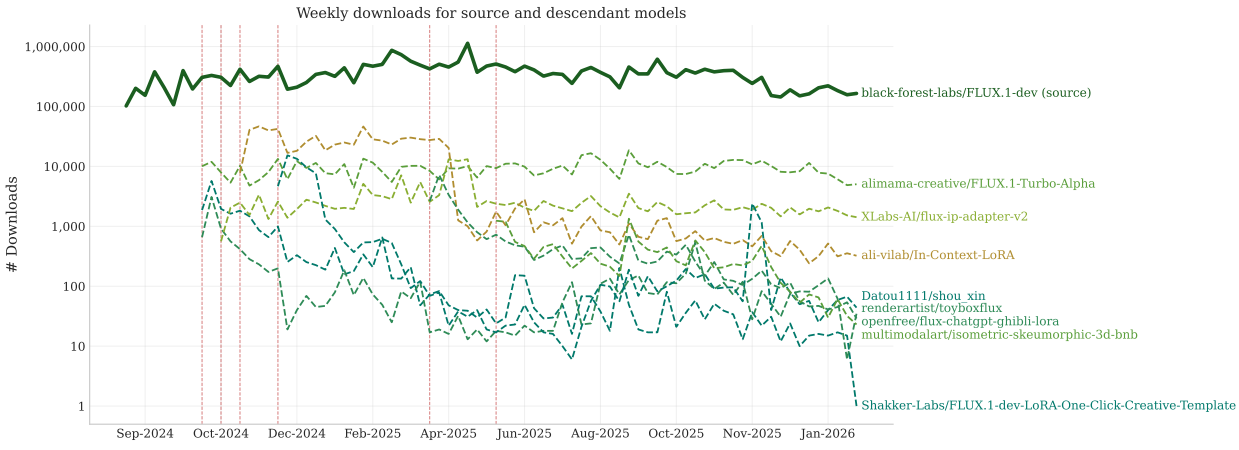
Together, these examples illustrate realizations of the three types of relationships we study, how related models can impact demand for source models, and the limits of inferring spillovers from individual time series. We next develop an empirical strategy to estimate the net effects of related releases while accounting for calendar-time shocks, model lifecycle dynamics, and overlapping releases.

Figure 4.3: Weekly downloads for Qwen2.5-32B and same-author descendants



Note: 7-day download count calculated from cumulative downloads reported weekly by Hugging Face.

Figure 4.4: Weekly downloads for FLUX.1-dev and third-party descendants



Note: 7-day download count calculated from cumulative downloads reported weekly by Hugging Face.

4.2.3 Empirical Strategy

Econometric framework mapping differences in downloads to differences in demand. Weekly downloads are our primary observed outcome. While downloads on their own are of interest, our primary object of economic interest is demand more broadly, in other words, the actual demand for AI models. While downloads do not necessarily map one-to-one to demand, under a modest set of assumptions, proportional changes in downloads map directly to proportional changes in demand.[54] This framework is especially helpful for evaluating spillovers, which can be modeled as changes in demand.

The econometric framework works as follows. Let D_{jt} denote weekly downloads and let Q_j denote baseline demand for model j. We can write the conditional mean of downloads as a function of the level of demand, Q_j, and proportional changes in both the level of demand and the mapping from demand to expected downloads:

$$\mathbb{E}[D_{jt} \mid \mathcal{T}_{jt}] = \left(\lambda_j \mu_t \nu_{b(j),t-t_0^j} \mathcal{T}_{jt}\right) Q_j \tag{4.1}$$

where λ_j captures model-specific differences in downloads per unit of demand, μ_t captures common changes in demand and the mapping to downloads over calendar time, and $\nu_{b(j),t-t_0^j}$ captures bin–age–specific differences in demand and the mapping to downloads over model lifecycles.[55] Additionally, $\mathcal{T}_{jt}$ captures the proportional changes in demand associated with spillovers from related-model releases.[56]

Under the setup in Section 4.2.3, we can recover estimates for the elements of $\mathcal{T}_{jt}$ using Poisson Pseudo Maximum Likelihood (PPML), with product fixed effects that absorb $\lambda_j Q_j$, time effects that absorb μ_t, and bin–age effects that absorb $\nu_{b(j),t-t_0^j}$. This setup allows the relationship between downloads and demand to vary by model, calendar time, and bin–age, while the release coefficients for spillovers are unchanged by differences in the level of the downloads-to-demand relationship across these dimensions.

Our primary restrictive assumption is that the release of related models cannot, on its own, change the relationship between downloads and demand. In other words, a related release must affect downloads only through its impact on demand. Under these assumptions, the resulting coefficients can be interpreted as proportional, and as a result unitless, changes in demand.

Specification. We focus on estimating net spillover effects from related releases in the top 30 trending in their first week and released at least seven days after the focal model. As described in Section 4.2.2, we jointly estimate effects from same-author non-descendant, same-author descendant, and third-party descendant releases. We allow these net effects to vary by popularity of the new release (top 10 versus top 11-30 trending in the first week). We also estimate accumulating immediate (first four weeks) and long term (after four weeks) effects. Further, the net effects can vary by the relative order of the related releases.[57] The econometric framework in Section 4.2.3 allows us to jointly estimate each of these net-spillover effects with the single PPML specification in Equation (4.2). We repeat the analysis on increasingly popular subsets of focal models including the full set of economically relevant models, the top 10%, and the top 1% by initial downloads.[58]

[54] Specifically, we focus on proportional changes in a unit-less index of demand. This can map to other latent-demand measures, such as usage or adoption, so long as the mapping assumptions hold for those outcomes.

[55] In our primary specifications, these bins are defined based on initial popularity.

[56] In the absence of any related releases, $\mathcal{T}_{jt}$ is normalized to one.

[57] For example, separate sets of effects for a focal model's first third-party descendant in the top 10 trending, and the second third-party descendant in the top 10 trending.

[58] Effects from differently popular related models are jointly estimated in the same regression, while effects by the popularity of the focal model use separate regressions.

Let j index models and t index calendar weeks. We estimate the following specification:

$$y_{jt} = \sum_{m \in \mathcal{M}} \left(\sum_{k=1}^{K_m} \tau_k^m \mathbb{1}(r_{jt}^m \geq k) + \mathbb{1}(r_{jt} > K_m) \left(\kappa_1^m + \kappa_2^m \log\left(r_{jt} - K_m \right) \right) \right) + \xi_j + \delta_t + \alpha_{b(j),t-t_0^j} + \varepsilon_{jt} \quad (4.2)$$

where y_{jt}, the outcome, is the number of downloads model j received in week t. Our parameters of interest, τ_k^m, denote the impact of subsequent releases on the outcome. We index τ_k^m based on the relationship between the focal model and related models and the within-relationship relative order of release: we use m to denote the type of relationship between models, and we use k to denote the relative release order of the related models (within group m). We estimate up to K_m separate τ_k^m terms for each m, and use κ_1^m and κ_2^m to control for any additional spillovers for releases beyond K_m.[59] The parameter ξ_j is a model fixed effect. δ_t is a calendar week fixed effect. We control for the heterogeneous model lifecycle patterns, as documented in Figure 4.1, with $\alpha_{b(j),t-t_0^j}$, which are initial popularity bin ($b(j)$) and age ($t - t_0^j$) specific fixed effects.

For each relationship type, we divide models into those whose release-week trending score was in the top 10 and those in the top 11-30. We separate the effects of a recent release (up to four weeks old) and a mature release (more than four weeks old). In total, $\mathcal{M}$ includes twelve release types, and we estimate the effect of all types jointly. For each type, m, we choose a maximum number of subsequent releases, K_m, based on the number of releases of that type. For same-author non-descendants, we set $K_m = 9$. For same-author descendants, we set $K_m = 2$, and we exclude top-coding since we never observe more than two subsequent releases of this type. For third-party descendants, we set $K_m = 2$. The dimensionality of types of treatment is quite large relative to settings where one might use standard event study approaches. Standard event study designs typically focus on a single discrete or continuous type of treatment, such as those outlined in Miller (2023). In our setting, a single specification estimates net spillover effects associated with 26 types of related releases, before accounting for timing (immediate and mature), and also not including the net spillovers associated with related releases past K_m. Figure C.6 presents motivating evidence for our choices of K_m.[60]

Even popular models have weeks with zero downloads or likes. We estimate the coefficients using Poisson pseudo maximum likelihood (PPML), which is well suited for our setting. PPML is consistent for the conditional mean when the count outcome has zeros (Silva and Tenreyro, 2006; Cohn et al., 2021), and allows for high-dimensional fixed effects without an incidental parameters problem (Wooldridge, 1999; Correia et al., 2020).[61] We include model fixed effects to absorb time-invariant differences across models, and calendar week fixed effects to control for platform-wide trends. We include initial popularity bin–age effects, to control for the different life-cycle patterns suggested by Figure 4.1.

It is worth re-emphasizing two additional benefits of our PPML specification in Equation (4.2). First, our parameters of interest avoid the age–period–cohort issue discussed above. While this impacts the model fixed effects, bin-age effects, and calendar time effects, the problem only impacts their interpretability, not their suitability as controls, which allows us to identify the net spillover terms. Second, as discussed in Section 4.2.3, those same fixed effects can capture both variation in downloads and variation in the mapping of downloads to demand.[62]

[59] For the top 10% and top 1% subsamples, there are a few cases where either κ_1 or κ_2 cannot be estimated due to a lack of data. In these cases those coefficient estimates are missing in the corresponding regression table.

[60] Essentially, we top-code further releases once the number of observations with at least that number of releases has dropped to one tenth of the number of observations with at least one release.

[61] Recent work shows that, for count outcomes with zeros, regressions with a $\log(1+x)$ or inverse hyperbolic sine transformation have a variety of issues (Chen and Roth, 2024; Mullahy and Norton, 2024)

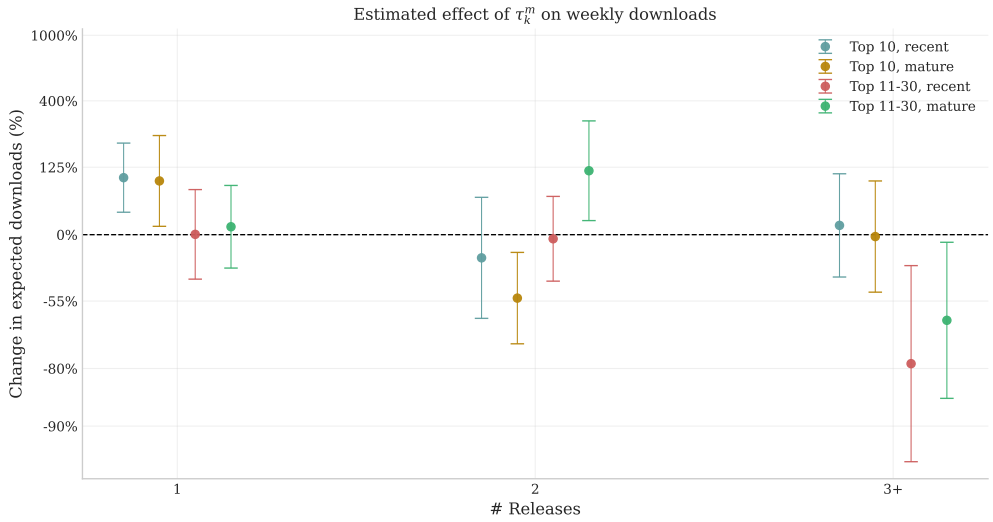
[62] And similarly the mapping of likes to attention.

4.2.4 Spillover Results

We find heterogeneous net spillover effects across relationship types and release order. The clearest results are positive net spillovers associated with popular third-party descendants, consistent with backward spillovers. We present the parameters of interest associated with each treatment type and release order in coefficient plots and regression tables. Figure 4.5 plots the recent and mature net spillover effects implied by the estimated parameters $\tau_1^m, \tau_2^m, \kappa_1^m$ associated with the release of third-party descendant models using the full sample of economically relevant focal models. We present the full set of underlying parameter estimates associated with third-party descendants in Table 4.1. The net spillover effect estimates from same-author descendants are in Table C.1, and same-author non-descendants are in Table C.2.[63] The coefficient plots for all relationships and for the different subsets of focal models are also presented in Section C.3, and we discuss the results for likes in Section C.3.2.

Demand Spillovers from Third-Party Descendants. The net effects of third-party descendants are of particular interest given that the ability of third-party developers to build on top of focal models is a defining characteristic of the open-weight market. We find large and statistically significant effects for the first release of top-10 descendants, and these net effects persist. Mapping the parameter estimates in Figure 4.5 to proportional changes in demand, we find that the first top-10 third-party descendant is associated with an initial 76–99% increase in focal-model demand, depending on the popularity of the focal model, and that the effect persists.[64]

Figure 4.5: Effects of subsequent model releases: third-party descendant models on demand



Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly downloads on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects.

While the first highly prominent descendant is associated with positive net spillovers, additional descendants are not associated with comparable incremental increases. Their estimates are mostly negative or statistically indistinguishable from zero. This pattern is consistent with a number of possible mechanisms.

[63] All three relationship types are estimated jointly. We report the estimates in separate tables for readability.

[64] The recent-period indicator remains active after four weeks, so the mature-period effect is the sum of the recent and mature coefficients. This combined effect is positive and statistically significant in all three focal-model samples, although the incremental mature coefficient is statistically significant only for the top 1% sample.

For example, the awareness effect could diminish quickly as the number of potential users unaware of the focal model shrinks with each new descendant (Ackerberg, 2001; Hendricks and Sorensen, 2009).

We find similar results across focal-model popularity, presented in Table 4.1. For each sample of focal models, both the initial effect and the combined mature effect are statistically significant.[65] Mapping the PPML coefficients to percent spillover effects, the full-sample estimates imply an initial 99% and long-term 91% increase in demand associated with the first top-10 third-party release. The corresponding results for the top 10% sample of focal models are an initial 94% and long-term 89% increase in demand. For the top 1% of focal models, the corresponding results are an initial 76% and long-term 173% increase in demand. Note that the difference between the immediate and long-term effects is only significant for the top 1% subsample. The top 1% subsample also shows positive and significant net spillovers associated with third-party descendants in the top 11–30.

Demand Net Spillovers from Same-Author Releases. Unlike estimates for third-party releases, we do not find systematic persistent net spillover effects within developers' own portfolios. Table C.1 presents the results for same-author descendant releases. The first same-author descendant has no statistically distinguishable effect on the focal model. We see a positive but insignificant effect for the first top-10 release.[66] Table C.2 presents the results for same-author non-descendant releases. For same-author non-descendants, we find a large but temporary net spillover associated with top 11–30 releases, while the corresponding effect associated with top-10 releases has no clear pattern. Estimates for later same-author releases vary in sign and, for same-author descendants, rely on only a small number of cases. Consequently, we do not find robust evidence of persistent same-author spillovers in either direction.

Interpretation of Demand Net Spillovers. These estimates provide net spillover effects from a variety of relationships between new models and related focal models. The patterns in the net effects can offer clues into the potential mechanisms driving AI demand and, as a result, strategic release decisions. The results rule out substitution as the only channel through which related-model releases affect realized demand. If substitution were the only channel, net spillover effects would be non-positive. Instead, we find that a focal model's first top-10 third-party descendant is associated with a substantial increase in source-model demand, consistent with positive backward spillovers. The variation in net spillover estimates across relationship types and release order suggests that we cannot rule out a sophisticated set of mechanisms driving AI demand and that the balance between positive and negative spillovers depends on details about the focal and related models. Further, these estimates show that in the open-weight market, downstream innovation can create backward spillovers to the open-weight models on which it builds.

[65] For the top 1%, the mature effect is also, on its own, significant.
[66] This could be due to the lower frequency of same-author descendants.

Table 4.1: Spillover effect estimates for third-party descendant models

	Downloads			Likes		
	Full	Top 10%	Top 1%	Full	Top 10%	Top 1%
Top 10						
1 release (recent)	0.6857***	0.6606***	0.5671***	0.3474***	0.3443***	0.3096***
	(0.2121)	(0.2084)	(0.1641)	(0.1258)	(0.1145)	(0.1000)
2 releases (recent)	-0.2786	-0.2477	-0.2497	-0.3508*	-0.3493*	-0.6457**
	(0.3713)	(0.3599)	(0.3658)	(0.2080)	(0.2073)	(0.2688)
3+ releases (recent)	0.1117	0.0902	0.0182	0.2324	0.2212	0.2641*
	(0.3168)	(0.3343)	(0.3426)	(0.1809)	(0.1728)	(0.1404)
Log(3+ releases) (recent)	-0.0083	0.0607	-0.8659	0.2211	0.2654	-0.9646
	(1.251)	(1.277)	(0.8034)	(0.7263)	(0.7345)	(0.7095)
1 release (mature)	-0.0393	-0.0223	0.4385**	-0.3649**	-0.3537**	-0.3100
	(0.1851)	(0.1847)	(0.1985)	(0.1626)	(0.1672)	(0.1936)
2 releases (mature)	-0.4842	-0.4346	-0.8381**	0.0472	0.0329	0.0668
	(0.3450)	(0.3337)	(0.3337)	(0.1959)	(0.1896)	(0.1825)
3+ releases (mature)	-0.1344	-0.1776	0.1626	-0.2312	-0.2181	0.0739
	(0.2395)	(0.2451)	(0.2259)	(0.2068)	(0.2094)	(0.1500)
Log(3+ releases) (mature)	0.3820	0.3302	0.8585	-0.3149	-0.1487	0.0601
	(0.9054)	(0.9117)	(0.9167)	(0.4475)	(0.4629)	(0.4863)
Top 11–30						
1 release (recent)	0.0038	0.0602	0.5080*	-0.0868	-0.1094	-0.0598
	(0.2748)	(0.2731)	(0.2710)	(0.1498)	(0.1500)	(0.1347)
2 releases (recent)	-0.0493	-0.0774	-0.5003	0.2076	0.2379	0.7392**
	(0.2600)	(0.2523)	(0.3416)	(0.2562)	(0.2499)	(0.2922)
3+ releases (recent)	-1.551***	-1.627**	-1.961***	-0.5946**	-0.5933**	-1.310***
	(0.6016)	(0.6499)	(0.2565)	(0.2619)	(0.2618)	(0.3099)
Log(3+ releases) (recent)	1.624	1.821		3.858***	3.928***	
	(3.157)	(3.309)		(0.9443)	(0.9286)	
1 release (mature)	0.0915	0.0972	0.1942	-0.3227**	-0.3427***	-0.4606***
	(0.2254)	(0.2202)	(0.2855)	(0.1253)	(0.1273)	(0.1191)
2 releases (mature)	0.8183***	0.8194***	0.9783***	0.4234	0.3617	0.1635
	(0.2959)	(0.2976)	(0.3602)	(0.3069)	(0.3037)	(0.3233)
3+ releases (mature)	0.5214	0.5626	1.974***	-0.1698	-0.0723	0.5810***
	(0.9205)	(0.9428)	(0.1670)	(0.1744)	(0.1590)	(0.1291)
Log(3+ releases) (mature)	1.704	1.605		-2.350***	-2.535***	
	(4.118)	(4.190)		(0.6302)	(0.6323)	
Model	Yes	Yes	Yes	Yes	Yes	Yes
Date (week)	Yes	Yes	Yes	Yes	Yes	Yes
Age*initial popularity	Yes	Yes	Yes	Yes	Yes	Yes
Observations	6,749,333	634,679	57,536	2,178,219	410,241	43,400
Pseudo R^2	0.91392	0.89650	0.90211	0.76988	0.82151	0.86886

Standard errors in parentheses clustered at the model level
*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1*

5 Death

In this section, we document facts about how and when models become obsolete. Developers frequently release new models (Figure 3.1), and sometimes new models perform better than existing ones, or offer comparable capabilities at lower cost. Given the significant improvements in model capabilities over the past few years, we are interested in defining and measuring how and when models become obsolete. For example, Meta's Llama 3 surpassed Llama 2 on most benchmarks, but this does not mean all users of Llama 2 immediately switched to Llama 3. Switching costs, embedding in software packages, preferences for specific model features, or other factors might mean demand for older models persists after they fall behind the technological frontier.

Using performance ranking data from Arena and Artificial Analysis, we define two types of obsolescence. *Technical obsolescence* means the model has been surpassed by other models on performance-based metrics. *Usage obsolescence* is defined based on downstream demand on OpenRouter, and occurs either when a model is delisted or the model's downstream market share permanently drops below a threshold. These measures allow us to distinguish when a model falls behind the technological frontier, from when it no longer sees meaningful use, and we show that they follow different patterns. A developer's decisions about the capabilities and release timings of its models depend on how each model compares technologically to other models, and on how long demand will persist for each model, so our results inform strategic choices about model development. In addition, patterns in how and when demand responds to obsolescence offer additional insights into how users choose models.

5.1 Technical Obsolescence

New models sometimes improve on the capabilities of existing models. For example, Meta released Llama 2 70B in July 2023 and released Llama 3 70B in April 2024. Llama 2 scored 25.6 on the HumanEval coding benchmark. Less than a year later, Llama 3 easily surpassed it, scoring 81.7.[67] While benchmark scores measure performance on specific tasks, they may not capture other model features that users care about, like writing tone and conciseness. Model arenas, which host pairwise comparisons between models, capture user preferences more directly.[68] We define technical obsolescence to capture these two measures of model capabilities: benchmark-based index scores and pairwise user preferences from the model arenas. Our metric characterizes how and when new, more capable models surpass older models. Our definition of technical obsolescence combines three data sources described in Section 2:

1. Benchmark-based index scores from Artificial Analysis
2. Elo scores from pairwise model comparisons on Artificial Analysis
3. Elo scores from pairwise model comparisons on Arena

Artificial Analysis and Arena are major platforms that track model capabilities and relative user preferences. Other platforms, where users choose models, reference these rankings.[69] Combining benchmark scores with Elo scores gives us a comprehensive measure of model performance relative to competitors. A

[67] Llama 3 model card found at huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct and HumanEval documentation found at deepeval.com/docs/benchmarks-human-eval.

[68] Llama 3 ranked fifth on Arena's leaderboards at release. Llama 2 ranked seventh at its first appearance on the leaderboards, about six weeks after its release

[69] For example, DigitalOcean uses both Arena and Artificial Analysis data in its inference routing process (docs.digitalocean.com/products/inference/how-to/use-inference-router/).

model might score highly on benchmarks but perform poorly on prompts submitted by real users, or vice versa. Models might also perform well on a specific task and poorly on others. To account for this potential specialization, our technical obsolescence definition uses a model's ranking in its best category.

For each model in each data source, we identify the category in which it ranked highest in its first observed week (Table D.2).[70] We then track the model's rank in that category against its competitors over time. The set of competitors changes as new models enter.[71] Once a model's rank, in its best category, falls below 30 in any of the three data sources, we classify it as technically obsolete. Table 5.1 summarizes the technical obsolescence classification. Most models eventually reach technical obsolescence, consistent with

Table 5.1: Technical obsolescence summary

	All models (n=773)		Hugging Face models (n=322)	
Platform	Number	Share	Number	Share
Any	616	79.7%	309	96.0%
Artificial Analysis Benchmarks	314	40.6%	194	60.2%
Arena.ai	301	38.9%	150	46.6%
Artificial Analysis Arena	121	15.7%	37	11.5%
Multiple (2+)	118	15.3%	71	22.0%

Note: Data as of February 11, 2026.

the recent fast pace of technological improvements in AI industry. The share of models that reach technical obsolescence based on benchmark scores is slightly larger than the share that reach technical obsolescence based on Arena rankings, which is understandable, given that benchmarks and pairwise model comparisons may capture different dimensions of model performance. These platforms also include closed source models, including high ranking frontier models from Anthropic, Google DeepMind, and OpenAI, which explains the lower technical obsolescence rate among all models relative to the subsample matched to Hugging Face.

The economic returns to a new model depend on its technical relevance and how demand reacts to newer, more capable models. These forces, in turn, impact the strategic decisions of developers who face tradeoffs between expected cost and capability when training new models, as well as which areas to specialize in and how to differentiate their models. Documenting the patterns of technical obsolescence helps us understand these economic incentives.

5.2 Usage Obsolescence

We define usage obsolescence to measure the point at which a model is no longer widely used for inference. This demand-based measure of obsolescence is particularly useful because, while technical obsolescence captures model capabilities, it ignores other features, like cost, that might impact a user's choice of model, as well as other features of the market that would lead to differences between the timing of technical obsolescence and meaningful changes in usage. We use data from OpenRouter, described in Section 2.4, which records inference, measured in input and output tokens, for a segment of the AI market. For each model we observe on OpenRouter, we record the first date on which either (1) the model is delisted from OpenRouter or (2) the share of tokens accruing to the model falls permanently below the 10th percentile.[72] Table 5.2 summarizes

[70] We focus on a model's best ranking to account for the fact that some models might be quite specialized.

[71] Sometimes models are removed from Artificial Analysis or Arena rankings, when this happens, we fill its most recently observed benchmark or Elo score forward to the present.

[72] When calculating the distribution of token shares, we use all models on OpenRouter to capture each model's relevance compared to its full set of competitors, including proprietary models.

Table 5.2: OpenRouter usage obsolescence summary

	All models (n=717)		Hugging Face models (n=321)	
	Number	Share	Number	Share
Dropout	72	10.0%	16	5.0%
Token share	153	21.3%	79	24.6%
Both	20	2.8%	6	1.9%

Note: Data as of February 11, 2026.

how models reach usage obsolescence. Among the 717 total models we observe on OpenRouter, 205 of them (28.6%) ever reach usage obsolescence, and of the 321 models matched to Hugging Face, 89 (27.7%) ever reach usage obsolescence. It is more common for models to reach usage obsolescence because of falling token shares than because of delisting from OpenRouter. For models that both drop below the minimum market share and are delisted, we use the earlier of the two dates.[73]

For a market to exist for an AI model, it must be profitable for inference providers, and it must have demand from users. Since OpenRouter is one of the most popular inference routing platforms, it provides a useful signal about these conditions. Delisting from OpenRouter signals that a model is no longer economically viable for providers, who face an opportunity cost for allocating their finite compute resources. Similarly, a model's token share dropping below the 10th percentile threshold provides evidence that users have switched to other models that offer better performance, lower cost, or both. Since demand for these models can have high week-to-week variance, we require the token share to permanently drop below the threshold to count as usage obsolete. The low token share condition suggests that, while the model still sees a small amount of use, most demand accrues to other models. Together, the two conditions offer evidence that the model has reached obsolescence.

Our measure of usage obsolescence allows us to study how users choose AI models and what information impacts their choices. OpenRouter represents a selected portion of downstream demand for inference in the AI market, and we can use our obsolescence metric to examine whether the platform provides a signal that impacts demand more broadly. The presence of a meaningful effect on demand would suggest that usage obsolescence provides additional information that leads users to adjust their choices, and that users are not fully informed about the models available in the market.[74]

Defining and measuring usage obsolescence also informs the nature of competition in the AI market. As with technical obsolescence, the economic returns to new model development depend on how long a developer expects its inference to be competitive in terms of quality, preferences, and costs.[75] The length of time models are used for inference also affects policy decisions. For example, antitrust regulation in the AI market requires knowledge of which models compete with one another. While older models may be technologically inferior, if they continue to be used intensively, they can remain important competitors in the market. In the remainder of this section, we analyze the timing and effects of obsolescence.

[73] Table D.1 lists the models that meet both criteria.

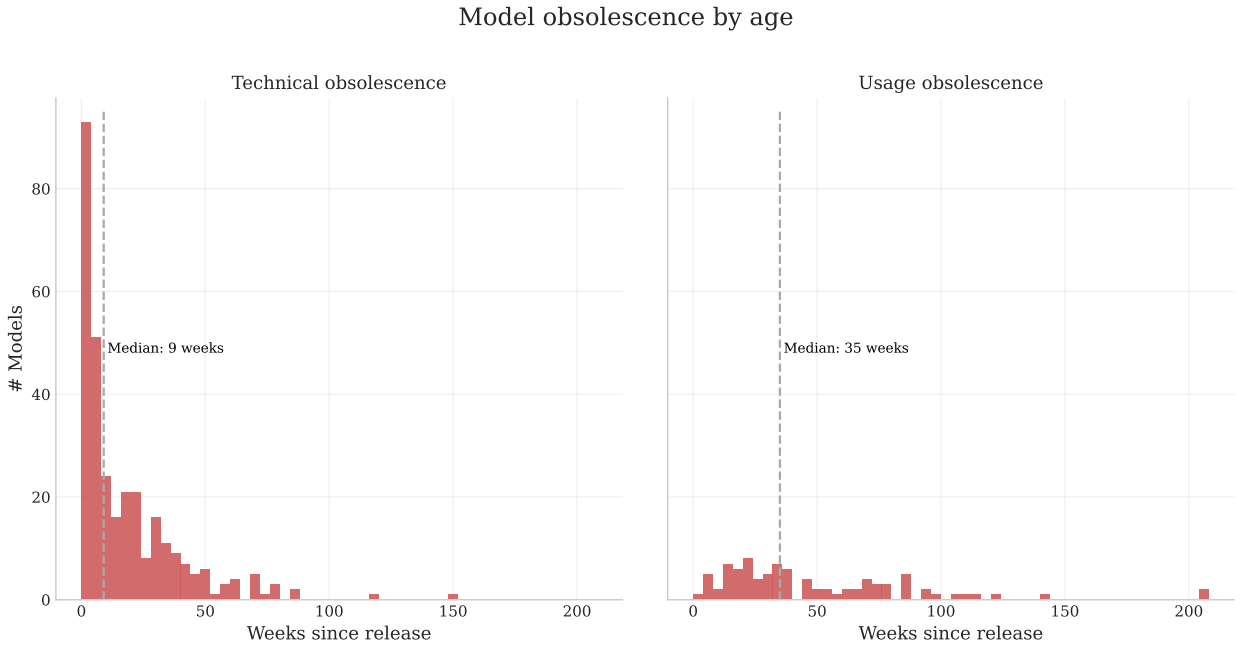
[74] For example, one might expect a large effect if uninformed users depend on other consumers choices to make decisions, as in Cai et al. (2009)

[75] Since open-weight model inference is often provided by third party inference providers, developers may not benefit directly from model usage. However, it is reasonable to assume that developer rewards are tied to the popularity of their models. Lerner and Tirole (2002) cover the more general case of incentives in open-source software development.

5.3 Age at Obsolescence

Having defined technical and usage obsolescence, we are interested in how quickly technical and usage obsolescence typically happen, how the patterns of obsolescence evolve after a model's release, and whether these patterns differ for usage versus technical obsolescence.[76] First, we examine how long after release models become obsolete. Figure 5.1 plots the distribution of model age at obsolescence, measured as weeks

Figure 5.1: Model age at obsolescence



Note: Usage obsolescence indicates token share below 10th percentile on OpenRouter or removal from OpenRouter.
Technical obsolescence indicates dropping below rank 30 in highest-ranked category at release on Artificial Analysis or Arena.
Restricted to models matched to Hugging Face.

since release. For most values of model age, more models become technically obsolete than become usage obsolete, reiterating our finding that technical obsolescence is more common than usage obsolescence. The median model becomes technically obsolete nine weeks after release, while for usage obsolescence the median is 36 weeks after release. The difference suggests that the technological frontier advances quickly, consistent with hundreds of new open-weight models releases each month (Figure 3.1). The two obsolescence measures also follow different early patterns. For example, 93 models (28%) become technically obsolete within their first four weeks, while none of the models on OpenRouter reach usage obsolescence that early. A number of mechanisms could explain these different patterns. First, benchmarks and arenas might not capture all of the important product features, like costs, meaning a market could exist for technically obsolete models. Second, the samples do not fully overlap, so selection of which models make it onto OpenRouter versus Artificial Analysis and Arena could drive some of these results.[77]

Among models in our sample that ever become obsolete, most reach obsolescence less than one year after

[76] We restrict our analysis to the models we are able to match to our sample of economically relevant models on Hugging Face. The matching yields 322 models for technical obsolescence and 321 models for usage obsolescence (Tables 5.1 and 5.2). While many models are present in both samples, they do not perfectly overlap. Section D.2 presents results including proprietary models.

[77] This could explain why so many models are immediately technically obsolete (but not usage obsolete), but not why so many see long term usage.

release, but there are some outliers. For example, one embedding model from the Sentence Transformers project was in the 23rd percentile of token shares on OpenRouter 203 weeks, nearly four years, after its release.[78] Stable Diffusion 2.1, a popular image generation model released in 2022, became technically obsolete 149 weeks after release.[79]

The fact that many models are technically obsolete soon after release, but fewer are usage obsolete in that period, suggests that there is some downstream demand for inference with models that are not at the technological frontier. For model developers, this means developing models that are not cutting edge, but that are still useful for some tasks, could be a viable business model. The outliers that do not become obsolete until a few years after release suggest that the relevant set of competing models for market definition could include models that are quite old.

5.4 Pace of Obsolescence

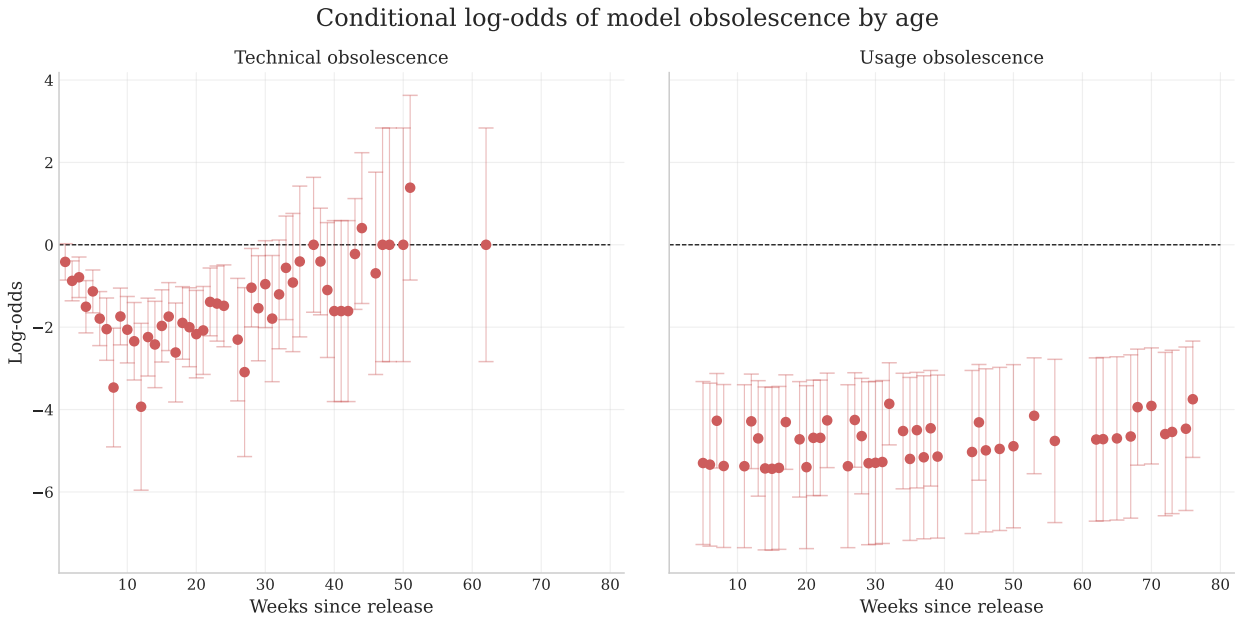
Next, we estimate the probability that a model becomes obsolete at a given age, and how that probability evolves over time. The sample does not include the full lifecycle of all models – some models were only recently added to OpenRouter, Artificial Analysis, or Arena, and some are not yet obsolete even years after release.[80] To address this problem, we estimate a hazard model for the conditional probability of obsolescence. The resulting estimates answer questions of the following form: conditional on a model still being in use 40 weeks after release, what is the probability that it reaches obsolescence in week 41? Conditional probabilities are useful because, for example, many models become technically obsolete quickly, so cumulative probabilities of technical obsolescence will be high at most model ages. However, conditional on not being technically obsolete at a given age, the probability of becoming technically obsolete in the next week might be relatively high or low. To measure this, we estimate the following specification:

$$\Pr(o_{jt} = 1 \mid o_{j,t-1} = 0) = \Lambda\Big(\alpha_{t-t_0^j}\Big) + \varepsilon_{jt}, \tag{5.1}$$

where $o_{jt} \in \{0, 1\}$ is an indicator equal to 1 if model j reaches obsolescence in calendar week t and 0 otherwise, t_0^j indexes the calendar week in which model j was released, and $\Lambda(x) = \frac{1}{1+e^{-x}}$ is the logistic function. The $\alpha_{t-t_0^j}$ terms are model-age fixed effects, with α_{52} as the excluded baseline. The estimation sample is restricted to model-weeks in which model j has not yet reached obsolescence at the start of week t, so the final observation for a given model is the week in which it becomes obsolete. Standard errors are clustered at the model level. Figure 5.2 plots the estimated coefficients $\hat{\alpha}_0, \ldots, \hat{\alpha}_{76}$. We exclude coefficients beyond age 76 due to insufficient obsolescence data. We also exclude coefficients for model-ages at which we observe no variation in obsolescence, since we cannot reliably estimate them. The log-odds increase (that is, move closer to zero) at later model ages, which implies that conditional on surviving to a given age, an older model is more likely to reach obsolescence in the following week. The coefficient estimates reinforce the differences between usage and technical obsolescence, since the conditional probability of usage obsolescence at each age is generally smaller than the probability of technical obsolescence. In addition, the technical obsolescence plot exhibits a U-shape in log-odds, which means, conditional on not being immediately technically obsolete, models become less likely to reach technical obsolescence and bottoms out around three to six months. One possible explanation is that some models are sufficiently technologically superior to their competitors that they are able to remain at or near the frontier for some time.

[78] huggingface.co/sentence-transformers/multi-qa-mpnet-base-dot-v1
[79] huggingface.co/sd2-community/stable-diffusion-2-1
[80] For example, we observe fewer models at 60 weeks since release than at 40 weeks since release.

Figure 5.2: Log-odds of obsolescence by age.



Note: Plots estimated log-odds of obsolescence from the following logit model:
Outcome: Binary indicator for a model becoming obsolete in a given week.
Sample: Model-weeks up to and including the obsolescence week.
Parameter of interest: coefficient on week since release.
Log-odds of zero implies 50% probability of obsolescence.
Plot shows coefficients on every other week, excluding weeks with no variation in obsolescence.

Using the hazard model estimates, we compute the implied cumulative probability of model obsolescence at each age.[81] The implied obsolescence curve as a function of model age is:

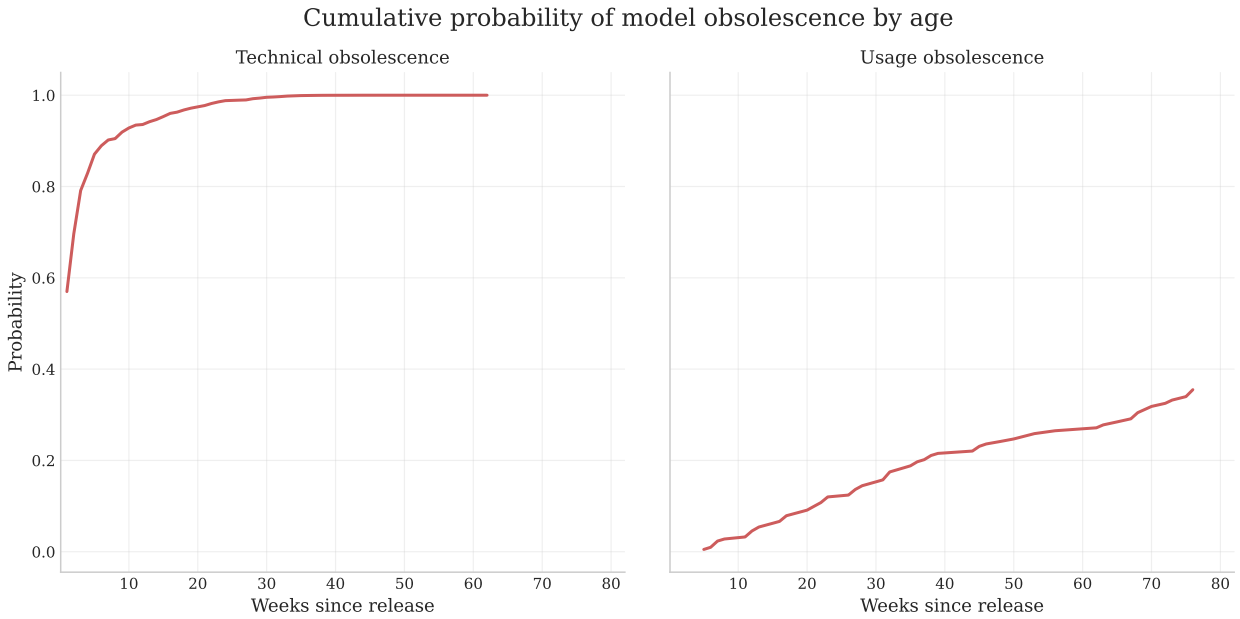
$$\widehat{O}(a) = 1 - \underbrace{\prod_{s=0}^{a} \Big(1 - \Pr\Big(o_{jt} = 1 \mid o_{j,t-1} = 0,\ t - t_0^j = s\Big)\Big)}_{\text{survival rate}} \tag{5.2}$$

Figure 5.3 plots these calculated cumulative probabilities. At one year since model release, the cumulative probability of usage obsolescence is about 25%, but it is nearly 100% for technical obsolescence. The cumulative probability of usage obsolescence is less than 40% even 1.5 years after release, suggesting that demand for inference can persist for older models.

These results also have implications for the nature of competition in the AI industry. We have established that technical obsolescence typically occurs earlier and with a higher probability than usage obsolescence. For developers, who incur large fixed costs to train models, their models will likely only remain at the technological frontier for a limited time, and this impacts their decision of how much to invest in training each model and how often to train a new model. On the other hand, downstream demand does not dissipate as quickly, which means developers might earn returns on their training investment for a longer period of time. For policymakers, research on market power often hinges on correctly defining the market, in this case the set of models that compete with one another. Since demand for inference from a model can persist for

[81] As an additional check, we estimate cumulative probabilities of obsolescence using a linear probability model (LPM) in Section D.1. We obtain qualitatively similar results in terms of relative differences between technical and usage obsolescence. The LPM-implied cumulative survival rates differ in absolute value, so we defer to the results of the hazard model, which is the more appropriate method in this setting.

Figure 5.3: Cumulative probability of usage obsolescence by weeks since release.



Note: Cumulative probability implied by estimated conditional probabilities.

years after release, the relevant set of competitors might include older, technologically obsolete models, as well as recent releases. Given the relevance of obsolescence for these economic questions, we now turn to explicitly measuring the effects of obsolescence on upstream demand.

5.5 Effects of Obsolescence on Demand and Attention

Next, we study the effects of technical and usage obsolescence on upstream demand for AI models. Our main outcome measure is downloads, which are a broad measure of upstream demand. Understanding how and when demand responds to obsolescence informs us about the nature of competition in the AI market, for example, by telling us whether obsolete models continue to compete with newer models. Further, the timing and duration of the demand response provides information about how and when users choose models. A delayed response, for example, would suggest that users face frictions in adopting models, and these frictions might affect counterfactual analyses and developer strategies. Technical obsolescence directly captures model capabilities, so it offers a clean measure of when a fully informed user, with no preference for cost, would abandon a model. Usage obsolescence is a complementary metric which captures preferences for model features beyond benchmarked capability, and which provides additional information about the structure and timing of demand.

5.5.1 Event Study Specification

We use an event study design to estimate deviations in expected demand (as measured by downloads) and attention (as measured by likes) in the periods surrounding the time of usage and technical obsolescence. We rely on the same econometric framework from Section 4.2.3 that maps proportional changes in downloads to proportional changes in demand to interpret our results. We implement a standard panel event study

specification described in Miller (2023) and implemented frequently in prior literature, such as Autor (2003). An event study design is a suitable approach to quantify the effects of model obsolescence, because, within a specific type of obsolescence (usage or technical), a model becomes obsolete once and remains obsolete forever.[82] Because downloads and likes are not part of our obsolescence definitions, we can study the effects of obsolescence on those outcomes with less concern that the patterns are mechanical.[83]

Our event study uses the following specification:

$$y_{jt} = \sum_{k=-15}^{20} \tau_k \mathbb{1}\left(t - t_d^j = k\right) + \tau_{21} \mathbb{1}\left(t - t_d^j \geq 21\right) + \xi_j + \delta_t + \alpha_{b(j),t-t_0^j} + \varepsilon_{jt} \tag{5.3}$$

where y_{jt} is the number of downloads (or likes) model j received in week t, t_0^j is the calendar week of model j's release, and t_d^j indexes the calendar week in which model j reaches obsolescence. Our parameters of interest, τ_k, capture the effect associated with obsolescence, where k indexes the weeks before or after obsolescence.[84] We pool the effects beyond 20 weeks into a single coefficient τ_{21}, which includes models that are already obsolete when we first observe them.[85] Observations more than 15 weeks before obsolescence, combined with models for which we never observe obsolescence, constitute the reference group when estimating the τ_k coefficients.[86] The specification includes model fixed effects ξ_j, calendar-week fixed effects δ_t, and model-age fixed effects $\alpha_{t-t_0^j}$. We estimate the coefficients using Poisson pseudo-maximum likelihood (PPML), for the same reasons described in Section 4.2.3, and we cluster standard errors at the model level. As in our spillover estimation strategy, we include fixed effects for model, calendar week, and model age interacted with initial popularity group to isolate how downloads and likes change around obsolescence rather than around secular trends or the ordinary lifecycle of model usage.[87] Since the specification includes model, time, and bin-age effects, the event study parameters can be interpreted as deviations from the expected model life-cycle.[88] Variation in the timing of obsolescence, along with models that never reach obsolescence in our sample period, separately identify the event study parameters, τ_k, from the controls.[89]

5.5.2 Obsolescence Event Study Results

Estimates. Figure 5.4 plots the estimated coefficients $\hat{\tau}_{-15}, \ldots, \hat{\tau}_{21}$ for demand (downloads).[90] For technical obsolescence (left), our results suggest a delayed and gradual effect on demand. We find no significant immediate effect in the week of obsolescence. However, we see evidence of a trend of decreasing negative deviations in demand that become statistically significant three weeks after obsolescence and are negative and mostly significant through weeks 21+. These event study estimates can be interpreted as proportional deviations from expected demand (based on the model, time, and bin-age effects). The effects suggest a 40–53% proportional decrease in demand in the 4–17 weeks after technical obsolescence, and a 60% proportional

[82] As described in Section 4.2.3, an event study design is not suitable for estimating spillover effects because each focal model is potentially affected by a large number of related model releases, which occur close together in time. By contrast, obsolescence is one-time absorbing state, which makes the event study design more appropriate.

[83] In other words, we are not using the outcome measure (downloads) to define the event (obsolescence)).

[84] Figure D.8 shows the number of models on which each coefficient is estimated.

[85] These models help pin-down the time and bin-age fixed effects in the regression.

[86] We think of the "never treated" models as "not yet treated", in the sense that they will eventually become obsolete.

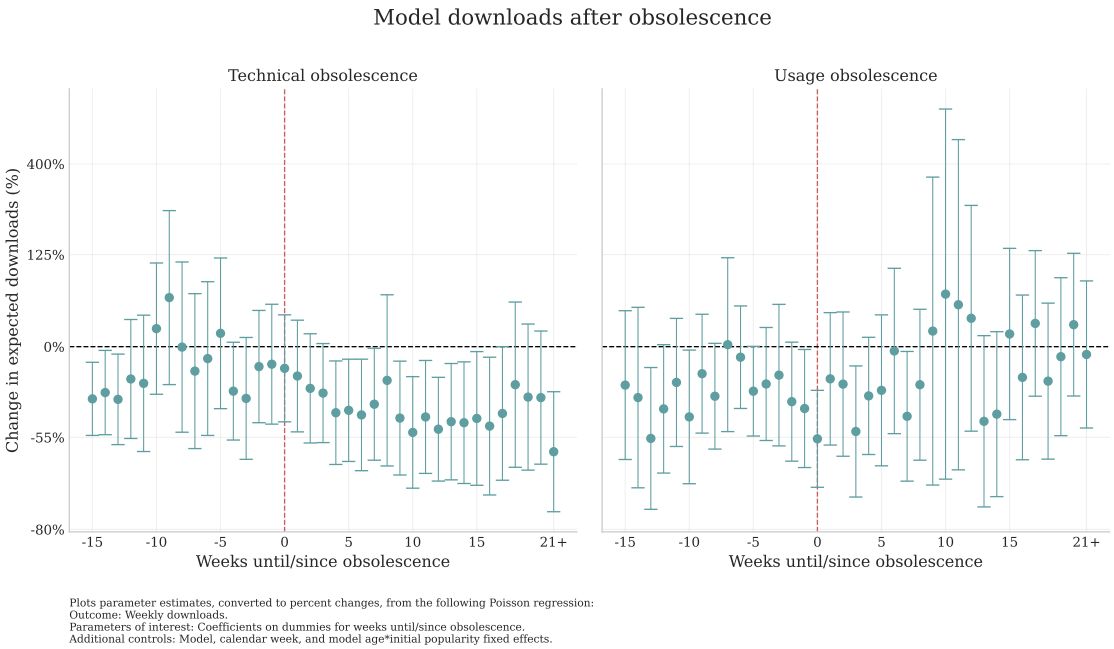
[87] The age–time–cohort problem could affect the estimates of these fixed effects, but we are interested in the estimated values of τ_k.

[88] For example, a parameter of zero means that the model demand is at the expected level. Two equal parameters at different relative timing imply the same proportional deviation from expected demand, not that demand is at the same level in those two periods.

[89] We present additional evidence motivating the event study design in Section D.3.

[90] We present the result for attention (likes) in Section D.4

Figure 5.4: Demand around obsolescence



decrease beyond 20 weeks after technical obsolescence.

For usage obsolescence (right), we find immediate and temporary effects associated with usage obsolescence, with negative and significant estimates for the week of usage obsolescence. We also report negative coefficients for 1 to 8 weeks after obsolesce, but only a subset of these are significant.[91] These results imply that, in the week of usage obsolescence, demand is 56% lower than otherwise expected. However, the effect is short-lived, as the coefficient estimates beyond seven weeks have mixed signs and are mostly statistically insignificant, indicating that upstream demand does not deviate from its expected level beyond seven weeks. We also see a negative and significant effect for the week preceding usage obsolescence, which could be explained both by how usage and downloads both measure demand, as well as differences in the timing of how downloads and inference map to contemporaneous demand.

The level and significance of these coefficients is relative to 16 or more weeks before obsolescence (for the treated) and all weeks from the never treated. This is a reasonable interpretation as all event study parameters are identified, and the model, time, and bin-age effects address the typical composition concerns one might have with an event study design. As an alternative interpretation, we also report these effects relative to the mean of τ_{-15} to τ_{-5}, and report them in Section D.5. The general findings for technical obsolesce are the same. For usage obsolescence only the effect for the week of obsolescence remains significant, which is still consistent with a negative but short lived effect.

Interpretation. Our technical obsolescence results show that, after an open-weight model falls behind the technological frontier, user demand declines (deviates from the expected lifecycle), but the response is gradual rather than immediate. Several possible mechanisms could explain this pattern. For example, technically obsolete models might be embedded in software packages, so that switching models requires either

[91] The parameters for weeks 0, 3, and 7 are negative and significant. The parameters for weeks 1, 2, 4, 5, and 6 are negative and insignificant.

adopting a new package or waiting for the package author to update the underlying model. Users might also face uncertainty about the quality of new models, for example, if a firm's internal processes rely on specific assumptions about model behavior, it might spend time evaluating a new model before adopting it. These barriers could explain our finding that demand does not respond to technical obsolescence immediately. However, a few weeks after technical obsolescence, we find drops in demand, which suggests that users do eventually substitute away from obsolete models. In terms of user decision making, this suggests a set of possible micro-founded mechanisms, such as default effects, switching costs, information frictions, and dependencies in complementary goods, which can make it suboptimal to continuously update choices, and lead to these types of slow demand responses. These patterns are inconsistent with some common structural assumptions in demand estimation.[92]

For usage obsolescence, we find negative effects in the week of obsolescence and up to eight weeks after obsolescence, which suggests that delisting from OpenRouter and low downstream demand are associated with a decrease in upstream demand. The coefficient estimates beyond eight weeks are statistically insignificant and vary in sign, indicating that upstream demand does not deviate from its expected level beyond eight weeks.[93] One explanation for this pattern is that usage obsolescence does not have long term effects on upstream demand, and instead, it changes the timing of users' active decision making. In this scenario, the users who stop using a model at the time of usage obsolescence would have otherwise stopped using that model in a few weeks. Essentially, this is a pulling forward of substitution away from the obsolete model. If users were continuously updating their choices, we would expect either a long term effect or no effect. Instead we see an immediate but temporary effect, which, like the technical obsolescence results, is consistent with a market where users are not continuously updating their model choices. This is also consistent with a market in which OpenRouter provides information about the relevance of models, and low usage or delisting from the platform serves as a signal of obsolete model's quality and cost relative to competing models, which impacts demand.

Together, these results suggest that users update their model choices periodically, but not at high frequencies. This could be a behavioral error, or it could be rational with information frictions or switching costs. If it is costly for users to research and choose new models, they might wait until technical capabilities have improved sufficiently to justify this cost. Signals from the market, like benchmark scores and downstream model popularity, might lead them to update their model choices more quickly.

Our obsolescence results have implications about the structure of the AI market and about model developers' strategic decisions. Since models do not immediately lose demand when they fall behind the technological frontier, a developer's optimal decisions depend not only on the capabilities of their models, but also on other factors that might generate demand, like integration in software packages. If users update their model choices infrequently, developers might pursue different release strategies than they would if users immediately adopted the latest frontier model. In terms of market structure, the market for open-weight models seems to be one where new products may compete with a variety of existing products, rather than immediately superseding them. Since the AI market has several competing firms, this affects their decisions, and it also informs competition policy.

[92] For example, standard applications of Berry et al. (1995), which typically assume unitary demand, full information, no frictions, and no complementarities.

[93] The timing is shorter under an alternative normalization (Section D.5), though the same intuition of a temporary effect holds.

6 Discussion

Our findings can better inform investigations into some of the biggest questions facing the AI industry. To highlight a few examples, consider categories of questions related to technical catch-up, the pacing of frontier development ("slowdown"), competition regulation, and the structure of AI demand.

Technical catch-up. A well established literature studies how firms behind the technological frontier can catch-up, and this impacts competition (e.g., Fudenberg et al., 1983; Aghion et al., 2001). Forecasts of the future of the AI market often point to the phenomenon where, task-by-task, specialized small models can reach similar capabilities as larger frontier models at a fraction of the inference cost (e.g., Belcak et al., 2025). Based on this phenomenon, some might jump to the conclusion that frontier AI firms face grim economic prospects in the future. However, as our results highlight, this kind of conclusion risks ignoring that developers produce portfolios of models, and that frontier labs could produce their own smaller and specialized models, and potentially benefit from within portfolio and within batch cost synergies (economies of scope). Our findings suggest that technological catch-up at the model level does not necessarily imply competitive catch-up at the firm level, meaning predictions about competition in the AI industry could benefit from considering firms' portfolio-level production costs and business models, not just individual model performance.

Pacing of frontier development. A growing policy debate at the forefront of the AI industry concerns "AI pacing." The debate entails whether and how to slow down AI development. It includes a broad set of policy objectives ranging from giving labor markets and other institutions adjustment time to reducing catastrophic risk from increasingly capable AI. Some of the proposed mechanisms include regulatory review and capability-dependent safeguards, as well as coordinated pauses in frontier development. These proposals face clear coordination problems, as an individual firm or country that slows development may fall behind competitors that do not. Our findings show that firms develop portfolios of models, and that there are potentially sizable inter-model dependencies within and across developers. This highlights an additional consideration for AI pacing: the models targeted by pacing policies would also impact non-targeted models due to the interdependency. As a result, pacing policies could have larger than predicted effects on aggregate AI progress. At the same time, they may also slow progress in lower-risk applications away from the frontier if those applications depend on frontier development.

Competition regulation. Regulatory questions about competition and market power rely on correctly defining the relevant markets, including which firms and which products compete with one another. In the case of AI, niche model developers that focus on specific applications of AI might be thought not to compete with large frontier labs building generalist models. Instead, we find that major developers often release portfolios of both vertically and horizontally differentiated models, which might include small, specialized models that could compete with offerings from more niche developers. Our results on the timing of obsolescence also inform the nature of competition in the AI market. Intuition might suggest that consumers primarily choose among a small set of the latest frontier models. Instead, we find that demand can persist for older, technically obsolete models, suggesting a broader set of competing products in the market.

Regulation and competition policy are often guided by quantifications of welfare, which rely on estimates of demand from a framework that correctly incorporates the specific features of the relevant industry. In the

open-weight AI market, we find economic mechanisms that traditional full-information unitary demand models (such as Berry et al. (1995)) cannot capture. Our positive demand spillover results indicate complementarities or an awareness effect. The responses to technical and usage obsolescence we find suggest that users are either not fully informed about the full set of models available in the market, face switching costs, or other market frictions. These mechanisms impact the welfare calculations that inform regulatory policy.

Structure of demand. In addition to the regulatory concerns noted above, a variety of research questions depend on correctly understanding demand for AI models and how people choose which models to use. While a variety of economic models can accommodate our results, we can also rule out some common assumptions about the structure of demand. Since developers release portfolios that include horizontal differentiation, and since users may prefer different models for different tasks, a model of demand for AI that restricts users to choosing a single model may not capture demand patterns correctly. Our positive spillover results indicate that a demand model should allow complementarities among the relevant goods. Finally, information frictions and other switching costs seem to play a role in model adoption, and a demand model should account for how these affect users' choices. These economic forces shape firms' model development decision making, as firms must anticipate demand for their products, and they also inform policy tradeoffs considered by researchers and regulators.

7 Conclusion

Economic and policy research is most informative when grounded in the relevant market's institutional details. This paper provides stylized facts and empirical results that establish key institutional details about the AI market. We construct a weekly panel of the 108,074 most-downloaded models on Hugging Face, which account for 99.5 percent of more than 53 billion downloads, and link these data to inference data from OpenRouter and performance ranking data from Arena and Artificial Analysis. Open-weight models on Hugging Face provide an ideal setting to further our understanding of the AI industry, as Hugging Face downloads provide broad coverage of an upstream margin of demand. The OpenRouter token share data complements this by providing a downstream measure of demand for a narrower set of models and users, while the benchmark and preference rankings measure capability relative to available alternatives. We develop an econometric framework that, under flexible assumptions, maps proportional changes in downloads to proportional changes in demand, and we use this framework to understand the nature of demand for AI models.

The paper is organized around three phases of AI life-cycles: birth, life, and death. In the birth phase, we show that major developers have portfolios of vertically and horizontally differentiated models and often release multiple new models in batches. Although the typical model is released alone, most downloads accrue to models released in large batches. A defining characteristic of the open-weight market is that first and third party developers can build on top of the source models through fine-tuning, quantization, merging, and adapting. We show that 27.9% of economically relevant models are such descendants of an open-weight source model. These patterns highlight differentiated model portfolios, batch releases, and model descendants, raising questions about how AI firms make strategic portfolio decisions, and the incentives that shape the AI market. On the developer side of the market, it is worth considering if developers optimize at the portfolio level and to what extent they internalize cost synergies and demand side spillovers from portfolio development and potential descendant models.

In the life phase, we describe the model lifecycle using our constructed weekly panel of downloads, and

compare these patterns to likes. We show that the set of economically relevant models is broad, with over 100,000 models receiving over 2,000 downloads. However, within this sample, downloads are concentrated among a subset of models both initially and over time. We also show that these patterns in downloads differ substantially from patterns in likes. Relative to downloads, likes are sparse and decline quickly, making them a poor proxy for contemporaneous demand. Using an econometric framework that maps proportional changes in downloads to proportional changes in demand, we estimate net spillovers to demand for a focal model from the release of new related models. We find positive net spillovers associated with the first top-10 trending third-party descendant. The net spillover effect persists, meaning a 76–99% increase in demand for the source model depending on the source model's initial popularity. We find no comparably systematic persistent spillovers from same-author releases or later third-party descendants. These positive net spillovers are especially notable given that they are associated with descendant third party releases, and a defining feature of the open weight ecosystem is that first and third party developers can build on an open-weight source model. A number of mechanisms could explain these patterns, including awareness-driven backward spillovers. However, a number of common structural assumptions in the empirical literature are not flexible enough to capture positive spillovers.[94] Researchers developing models to understand and analyze this market should consider structural assumptions and parameterizations that can capture positive net spillovers. Downloads are not themselves developer revenue. However, broader adoption can enter developer payoffs in a number of ways, including paid inference, demand for complementary goods, signaling for fundraising, and the value of expanding the ecosystem around the model. As such, our estimated positive spillovers identify a potential private benefit of third-party development to first party developers.

In the death phase, we evaluate how and when models become obsolete and how demand reacts to these obsolescence events. We evaluate obsolescence from two perspectives, defining technical and usage obsolescence. Technical obsolescence is based on a broad set of categories using both capability based benchmark rankings, and user preference based Elo scores, with a model reaching technical obsolescence once it drops below the top 30 in its best category. We use data from OpenRouter to document usage obsolescence, which occurs when a model is de-listed or permanently drops below the 10th percentile in terms of market share. It is useful to consider both measures as technical obsolescence alone does not capture product features users might care about, like costs. Similarly, differences in the timing between technical and usage obsolescence, along with how demand reacts to these events, are informative of how long models stay economically relevant and the mechanisms through which users choose models. We find that many models are immediately technically obsolete, and our hazard model shows that the representative model has a near 100% chance of obsolescence after 30 weeks, though top models remain technically relevant far longer. The likelihood of usage obsolescence follows a different pattern, occurring at a lower rate and increasing slowly.[95] We then use an event study design to evaluate changes in demand associated with these obsolescence events. We find decreases in demand after both technical and usage obsolescence, with somewhat different patterns. For technical obsolescence, our estimates indicate a drop in demand every week after obsolescence occurs, with smaller effects immediately afterwards and larger effects later on. For usage obsolescence, we see an immediate, but temporary drop in demand that lasts about seven weeks, before demand returns to its expected level. This suggests that the lost demand could be driven by users who would have stopped demanding the obsolete model in a few weeks anyway. Combined, the two obsolescence findings suggest a

[94] For example, standard applications of Berry et al. (1995) type demand estimation require assumptions inconsistent with this phenomenon.

[95] In part this result is a function of selection for which models are ever listed on OpenRouter versus the benchmarking and arena platforms. As more data become available, powered analysis of jointly listed models will be possible.

market where users do not continuously update their decisions with every new release, and instead periodically update their model choices.

Together, these findings establish key institutional details of the open-weight AI market. They suggest that the AI market is one where developers have differentiated multi-product portfolios, with interdependent development within and across developers. A substantial portion of models in the AI market are the result of cross-developer innovations, where third party developers build off of existing models. The structure of demand is such that related models generate positive spillovers, and that demand is slow to adjust to technical obsolescence. These findings can inform investigations into some of the biggest questions facing the AI industry. These include questions of competition that depend on market definition, as well as questions about innovation and the returns to R&D, which depend on portfolio-level decision-making and spillovers. These findings also apply to questions about the advancement of AI and the role of open-weight models more broadly. Our findings open new avenues for future research. These include understanding model development synergies within and across developers, quantifying developers' diverse dynamic incentives and spillovers associated with open-weight releases, and how to appropriately model observed and latent demand for AI.

References

Ackerberg, D. A. (2001). Empirically Distinguishing Informative and Prestige Effects of Advertising. *The RAND Journal of Economics 32*(2), 316.

Aghion, P., C. Harris, P. Howitt, and J. Vickers (2001, July). Competition, Imitation and Growth with Step-by-Step Innovation. *Review of Economic Studies 68*(3), 467–492.

Aubakirova, M., A. Atallah, C. Clark, J. Summerville, and A. Midha (2026, January). State of AI: An Empirical 100 Trillion Token Study with OpenRouter. arXiv:2601.10088 [cs.AI].

Autor, D. (2003, January). Outsourcing at Will: The Contribution of Unjust Dismissal Doctrine to the Growth of Employment Outsourcing. *Journal of Labor Economics 21*(1), 1–42.

Azoulay, P., J. Krieger, and A. Nagaraj (2024, May). Old Moats for New Models: Openness, Control, and Competition in Generative AI. Technical Report w32474, National Bureau of Economic Research, Cambridge, MA.

Belcak, P., G. Heinrich, S. Diao, Y. Fu, X. Dong, S. Muralidharan, Y. C. Lin, and P. Molchanov (2025, September). Small Language Models are the Future of Agentic AI. arXiv:2506.02153 [cs].

Berry, S., J. Levinsohn, and A. Pakes (1995). Automobile Prices in Market Equilibrium. *Econometrica 63*(4), 841–890.

Bresnahan, T. (2024, March). What innovation paths for AI to become a GPT? *Journal of Economics & Management Strategy 33*(2), 305–316.

Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei (2020, July). Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL].

Cai, H., Y. Chen, and H. Fang (2009, May). Observational Learning: Evidence from a Randomized Natural Field Experiment. *American Economic Review 99*(3), 864–882.

Chen, J. and J. Roth (2024, March). Logs with Zeros? Some Problems and Solutions. *The Quarterly Journal of Economics 139*(2), 891–936.

Cohn, J. B., Z. Liu, and M. Wardlaw (2021). Count Data in Finance. *SSRN Electronic Journal*.

Conlon, C., N. H. Miller, T. Otgon, and Y. Yao (2023, May). Rising Markups, Rising Prices? *AEA Papers and Proceedings 113*, 279–283.

Correia, S., P. Guimarães, and T. Zylkin (2020, March). Fast Poisson estimation with high-dimensional fixed effects. *The Stata Journal: Promoting communications on statistics and Stata 20*(1), 95–115.

De Loecker, J., J. Eeckhout, and G. Unger (2020, May). The Rise of Market Power and the Macroeconomic Implications*. *The Quarterly Journal of Economics 135*(2), 561–644.

Demirer, M., A. Fradkin, N. Tadelis, and S. Peng (2025). The Emerging Market for Intelligence: Pricing, Supply, and Demand for LLMs.

Dettmers, T., A. Pagnoni, A. Holtzman, and L. Zettlemoyer (2023). QLoRA: Efficient Finetuning of Quantized LLMs. Version Number: 1.

Dunne, T., M. J. Roberts, and L. Samuelson (1988). Patterns of Firm Entry and Exit in U.S. Manufacturing Industries. *The RAND Journal of Economics 19*(4), 495.

Fradkin, A. (2025). Demand for LLMs: Descriptive Evidence on Substitution, Market Expansion, and Multihoming. Version Number: 1.

Fudenberg, D., R. Gilbert, J. Stiglitz, and J. Tirole (1983, June). Preemption, leapfrogging and competition in patent races. *European Economic Review 22*(1), 3–31.

Gort, M. and S. Klepper (1982, September). Time Paths in the Diffusion of Product Innovations. *The Economic Journal 92*(367), 630.

Griliches, Z. (1957, October). Hybrid Corn: An Exploration in the Economics of Technological Change. *Econometrica 25*(4), 501.

Hendricks, K. and A. Sorensen (2009, April). Information and the Skewness of Music Sales. *Journal of Political Economy 117*(2), 324–369.

Hoffmann, J., S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. v. d. Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre (2022). Training Compute-Optimal Large Language Models. Version Number: 1.

Jiang, W., N. Synovic, M. Hyatt, T. R. Schorlemmer, R. Sethi, Y.-H. Lu, G. K. Thiruvathukal, and J. C. Davis (2023). An Empirical Study of Pre-Trained Model Reuse in the Hugging Face Deep Learning Model Registry. Version Number: 1.

Kadasi, P., S. R. Kondam, S. V. Chaturvedula, R. Sen, A. Saha, S. Sikdar, S. Sarkar, S. Mittal, R. Jindal, and M. Singh (2025, April). Model Hubs and Beyond: Analyzing Model Popularity, Performance, and Documentation. arXiv:2503.15222 [cs.CL].

Kaplan, J., S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei (2020). Scaling Laws for Neural Language Models. Version Number: 1.

Kapoor, S., R. Bommasani, K. Klyman, S. Longpre, A. Ramaswami, P. Cihon, A. Hopkins, K. Bankston, S. Biderman, M. Bogen, R. Chowdhury, A. Engler, P. Henderson, Y. Jernite, S. Lazar, S. Maffulli, A. Nelson, J. Pineau, A. Skowron, D. Song, V. Storchan, D. Zhang, D. E. Ho, P. Liang, and A. Narayanan (2024). On the Societal Impact of Open Foundation Models. Version Number: 1.

Korinek, A. and J. Vipra (2024, November). Concentrating Intelligence: Scaling and Market Structure in Artificial Intelligence. Technical Report w33139, National Bureau of Economic Research, Cambridge, MA.

Laufer, B., H. Oderinwale, and J. Kleinberg (2025). Anatomy of a Machine Learning Ecosystem: 2 Million Models on Hugging Face. Version Number: 1.

Lerner, J. and J. Tirole (2002, June). Some Simple Economics of Open Source. *The Journal of Industrial Economics 50*(2), 197–234.

Liu, H., Q. Liu, and P. K. Chintagunta (2017, May). Promotion Spillovers: Drug Detailing in Combination Therapy. *Marketing Science 36*(3), 382–401.

Longpre, S., C. Akiki, C. Lund, A. Kulkarni, E. Chen, I. Solaiman, A. Ghosh, Y. Jernite, and L.-A. Kaffee (2025, November). Economies of Open Intelligence: Tracing Power & Participation in the Model Ecosystem. arXiv:2512.03073 [cs.CY].

McKenzie, D. J. (2006, August). Disentangling Age, Cohort and Time Effects in the Additive Model*. *Oxford Bulletin of Economics and Statistics 68*(4), 473–495.

Mertens, M., N. Fischl-Lanzoni, and N. Thompson (2026). Is there "Secret Sauce" in Large Language Model Development? Version Number: 2.

Miller, D. L. (2023, May). An Introductory Guide to Event Study Models. *Journal of Economic Perspectives 37*(2), 203–230.

Mullahy, J. and E. C. Norton (2024, April). Why Transform Y ? The Pitfalls of Transformed Regressions with a Mass at Zero*. *Oxford Bulletin of Economics and Statistics 86*(2), 417–447.

OpenAI, S. Agarwal, L. Ahmad, J. Ai, S. Altman, A. Applebaum, E. Arbus, R. K. Arora, Y. Bai, B. Baker, H. Bao, B. Barak, A. Bennett, T. Bertao, N. Brett, E. Brevdo, G. Brockman, S. Bubeck, C. Chang, K. Chen, M. Chen, E. Cheung, A. Clark, D. Cook, M. Dukhan, C. Dvorak, K. Fives, V. Fomenko, T. Garipov, K. Georgiev, M. Glaese, T. Gogineni, A. Goucher, L. Gross, K. G. Guzman, J. Hallman, J. Hehir, J. Heidecke, A. Helyar, H. Hu, R. Huet, J. Huh, S. Jain, Z. Johnson, C. Koch, I. Kofman, D. Kundel, J. Kwon, V. Kyrylov, E. Y. Le, G. Leclerc, J. P. Lennon, S. Lessans, M. Lezcano-Casado, Y. Li, Z. Li, J. Lin, J. Liss, Lily, Liu, J. Liu, K. Lu, C. Lu, Z. Martinovic, L. McCallum, J. McGrath, S. McKinney, A. McLaughlin, S. Mei, S. Mostovoy, T. Mu, G. Myles, A. Neitz, A. Nichol, J. Pachocki, A. Paino, D. Palmie, A. Pantuliano, G. Parascandolo, J. Park, L. Pathak, C. Paz, L. Peran, D. Pimenov, M. Pokrass, E. Proehl, H. Qiu, G. Raila, F. Raso, H. Ren, K. Richardson, D. Robinson, B. Rotsted, H. Salman,

S. Sanjeev, M. Schwarzer, D. Sculley, H. Sikchi, K. Simon, K. Singhal, Y. Song, D. Stuckey, Z. Sun, P. Tillet, S. Toizer, F. Tsimpourlas, N. Vyas, E. Wallace, X. Wang, M. Wang, O. Watkins, K. Weil, A. Wendling, K. Whinnery, C. Whitney, H. Wong, L. Yang, Y. Yang, M. Yasunaga, K. Ying, W. Zaremba, W. Zhan, C. Zhang, B. Zhang, E. Zhang, and S. Zhao (2025, August). gpt-oss-120b & gpt-oss-20b Model Card. arXiv:2508.10925 [cs.CL].

Osborne, C., J. Ding, and H. R. Kirk (2024, October). The AI Community Building the Future? A Quantitative Analysis of Development Activity on Hugging Face Hub. *Journal of Computational Social Science 7*(2), 2067–2105. arXiv:2405.13058 [cs.SE].

Owen, D. (2024, January). How predictable is language model benchmark performance? arXiv:2401.04757 [cs.LG].

Rein, D., B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman (2023, November). GPQA: A Graduate-Level Google-Proof Q&A Benchmark. arXiv:2311.12022 [cs.AI].

Rosenfeld, J. S., A. Rosenfeld, Y. Belinkov, and N. Shavit (2019, December). A Constructive Prediction of the Generalization Error Across Scales. arXiv:1909.12673 [cs.LG].

Santos, G. S. S. d. and F. Figueiredo (2024, November). Assessing the Impact of Sampling, Remixes, and Covers on Original Song Popularity. arXiv:2411.01242 [cs.SI].

Schulhofer-Wohl, S. (2018). The age-time-cohort problem and the identification of structural parameters in life-cycle models. *Quantitative Economics 9*(2), 643–658.

Schuster, M., D. Mitchell, and K. Brown (2019, March). Sampling Increases Music Sales: An Empirical Copyright Study. *American Business Law Journal 56*(1), 177–229.

Shapiro, B. T. (2018, February). Positive Spillovers and Free Riding in Advertising of Prescription Pharmaceuticals: The Case of Antidepressants. *Journal of Political Economy 126*(1), 381–437.

Silva, J. M. C. S. and S. Tenreyro (2006, November). The Log of Gravity. *The Review of Economics and Statistics 88*(4), 641–658.

Sinkinson, M. and A. Starc (2019, March). Ask Your Doctor? Direct-to-Consumer Advertising of Pharmaceuticals. *The Review of Economic Studies 86*(2), 836–881.

Wan, A., K. Klyman, S. Kapoor, N. Maslej, S. Longpre, B. Xiong, P. Liang, and R. Bommasani (2025, December). The 2025 Foundation Model Transparency Index. arXiv:2512.10169 [cs].

Watson, J. (2017). What is the Value of Re-Use? Complementarities in Popular Music. *SSRN Electronic Journal*.

Wei, J., Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus (2022, October). Emergent Abilities of Large Language Models. arXiv:2206.07682 [cs.CL].

Wollmann, T. G. (2018, June). Trucks without Bailouts: Equilibrium Product Characteristics for Commercial Vehicles. *American Economic Review 108*(6), 1364–1406.

Wooldridge, J. M. (1999, May). Distribution-free estimation of some nonlinear panel data models. *Journal of Econometrics 90*(1), 77–97.

Appendix

A Data

A.1 Constructing Weekly Downloads

If we observed every model in the data starting on its creation date, we could use the rolling 30 day download count to calculate cumulative downloads. From that we could calculate the number of times the model was downloaded each week. However, our data is truncated at July 2024, so we do not observe initial values of rolling 30 day downloads for all models. We observe cumulative downloads directly starting in February 2025, which creates a six month gap during which we do not observe cumulative downloads. We remedy this with the following observation. Let D_t represent cumulative model downloads at time t and let R_t be the number of downloads the model received in the 30 days leading up to and including t. Then

$$D_{t-30} = D_t - R_t \tag{A.1}$$

Since we observe D_t and R_t at the daily level, we can use Equation A.1 to calculate D_t for each day between July 2024 and February 2025. Then we can calculate $d_t = D_t - D_{t-1}$, the number of downloads the model received on day t. Specifically, we use the following algorithm:

1. Initialize $\tilde{D}_{jt} = D_{jt}$ wherever D_{jt} is observed.
2. Repeatedly enforce monotonicity on the current series. If the series decreases between some date and the next non-missing date, find the largest such drop, set both endpoints of that drop to missing, and repeat until the remaining non-missing series is weakly increasing.
3. Interpolate the values removed in step 2 using shape-preserving cubic interpolation between observed points and linear extrapolation at the ends. If only one non-missing value remains, fill the series by carrying that value forward and backward. Replace only the values removed in step 2 with these interpolated values.
4. Use the 30-day identity $\tilde{D}_{j,t-30} = \tilde{D}_{jt} - R_{jt}$ to backfill missing cumulative downloads. To allow for small timing mismatches across snapshots, form

   $$B_{jt} = \text{mean}\left\{\tilde{D}_{jt} - R_{j,t-1}, \tilde{D}_{jt} - R_{jt}, \tilde{D}_{jt} - R_{j,t+1}\right\},$$

   where the mean is taken over available terms, and use B_{jt} to fill $\tilde{D}_{j,t-30}$ whenever that earlier value is missing.
5. Repeat steps 2–4 until no new values are filled. Then fill any remaining gaps by the interpolation rule in step 3.
6. After processing all models, any negative imputed values are set to missing, the code checks that each model's cumulative download series is weakly increasing, and the final series is rounded to integers before it is merged into the panel as cumulative downloads.

After completing the backfilling process, we observe a monotonically increasing series of cumulative downloads for each model j and each day t in the data. Note that we observe cumulative likes directly for the duration

of our sample, so the likes variable does not suffer from this missing data problem. All analysis proceeds with these data aggregated to the weekly level.

A.2 Matching OpenRouter and Hugging Face

We match OpenRouter models to Hugging Face models using Hugging Face model IDs reported in the OpenRouter data.

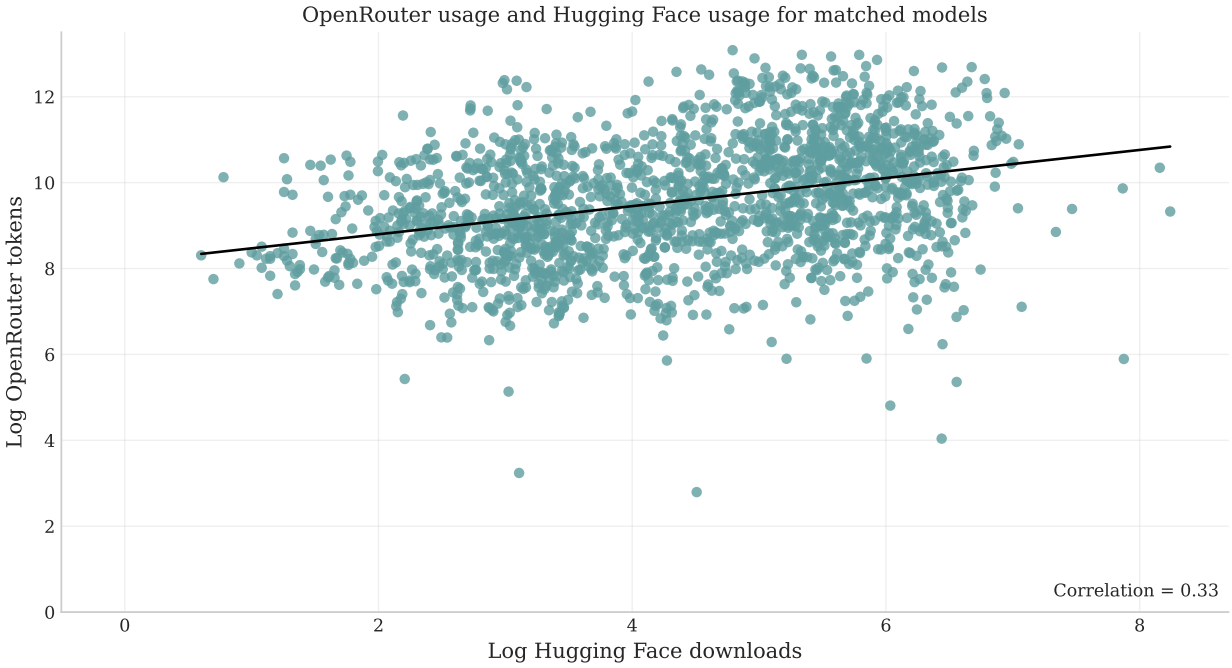
1. For each OpenRouter snapshot, we extract the Hugging Face model ID when OpenRouter reports one. In the raw data this appears in fields such as `hf_slug` or `hugging_face_id`.

2. We standardize OpenRouter model IDs before propagating matches. In particular, we drop variant suffixes that appear after a colon and strip trailing date stamps of the form "-YYYYMMDD." We assign the same Hugging Face model ID to all OpenRouter observations that share this cleaned OpenRouter ID.

3. After constructing the daily OpenRouter panel, we keep observations with a non-missing Hugging Face ID and aggregate the OpenRouter data to one observation per Hugging Face model ID and date.

4. Finally, we split the Hugging Face model ID into author and model name and merge the OpenRouter panel to the weekly Hugging Face panel by author, model name, and date. Once a Hugging Face model is matched in one week, we carry that match across all weeks for that model in the Hugging Face panel.

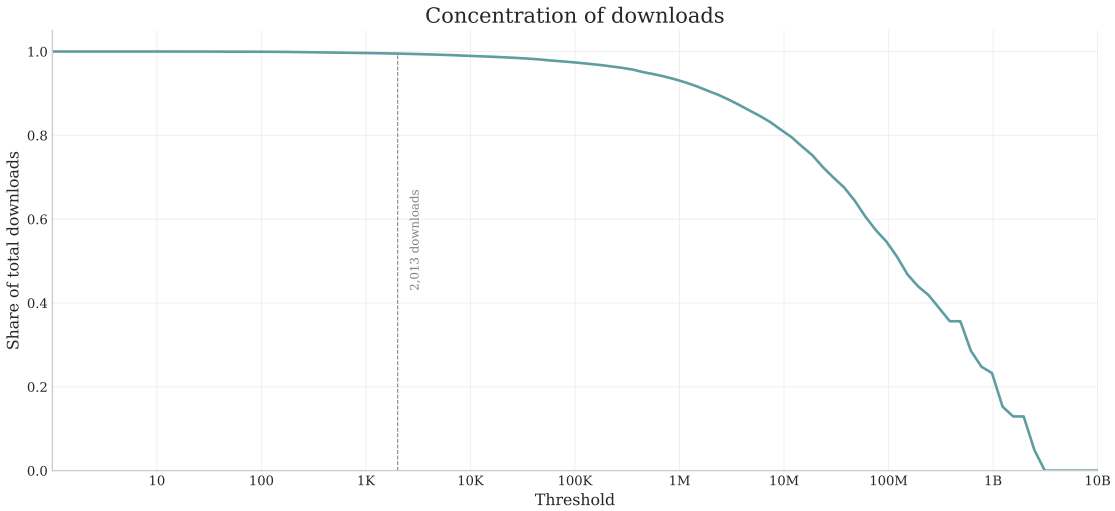
Figure A.1: Matched-model cumulative Hugging Face downloads and cumulative OpenRouter token use



Note: Plots 1,916 model-month observations for 251 models with non-zero download and token counts on Hugging Face and OpenRouter.

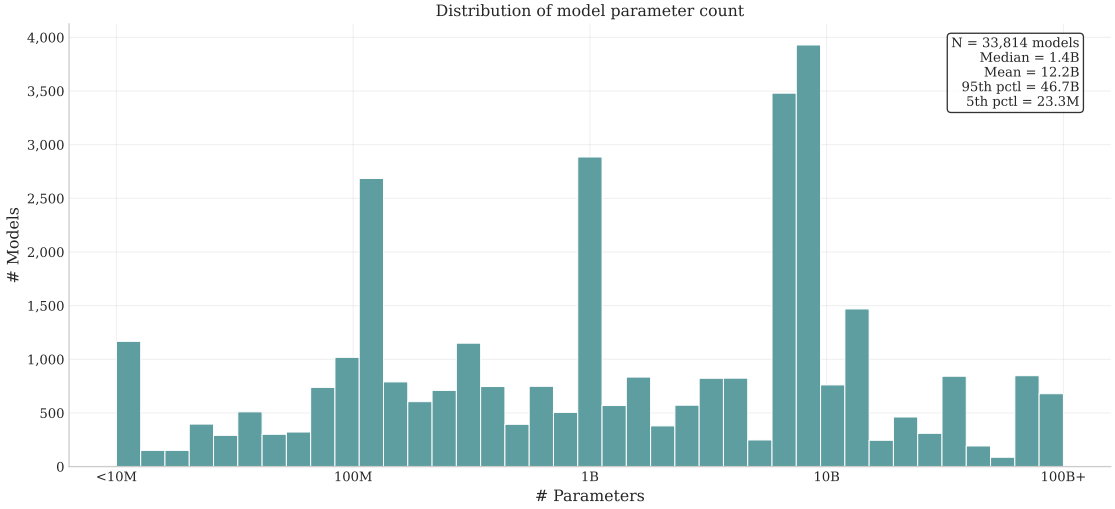
A.3 Additional figures

Figure A.2: Downloads concentration



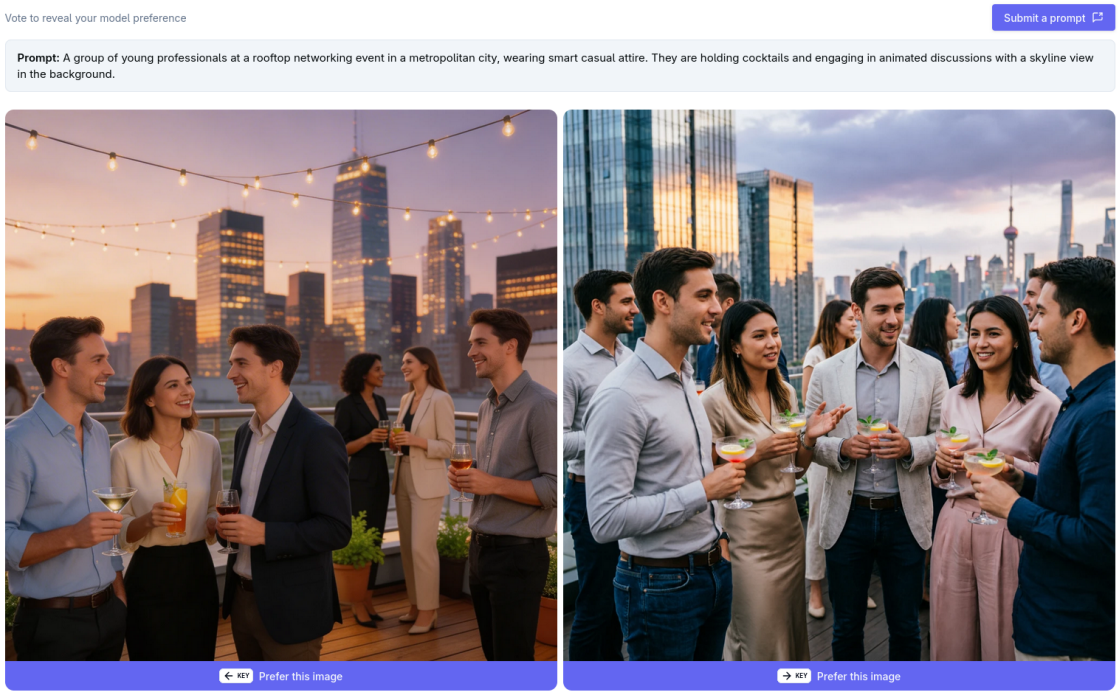
Note: plots the share of total downloads accounted for by models with more than M downloads for increasing values of M.
Models with more than 2,013 downloads account for 99.5% of all downloads on Hugging Face.

Figure A.3: Distribution of model size



Note: Excludes 74,260 models with no reported parameter count.

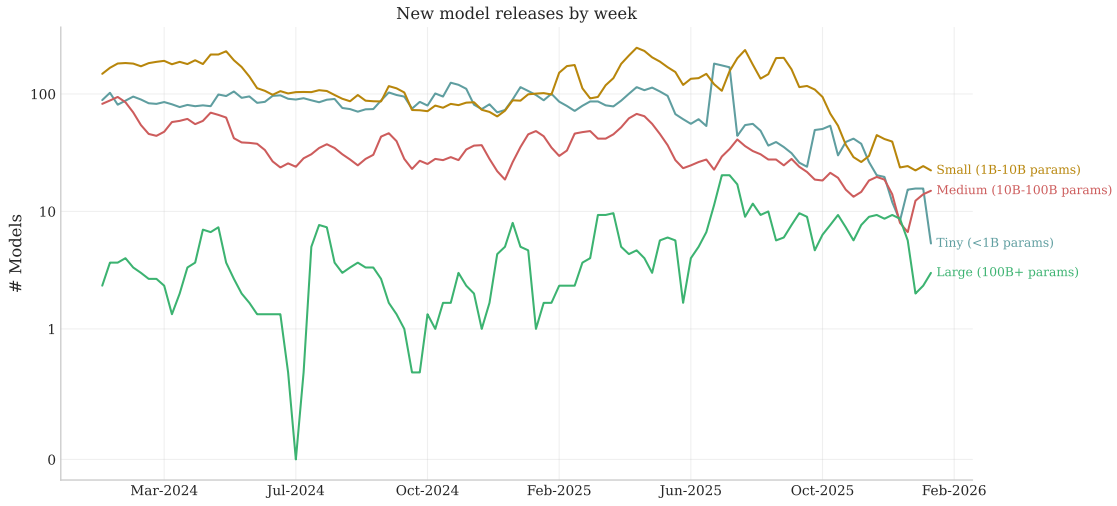
Figure A.4: Artificial Analysis Image Arena



B Birth

Figure B.1 plots model releases for economically relevant models, mirroring Figure 3.1 which covers all models. Both figures follow similar trends outlined in Section 3.1, with two differences worth highlighting. First, the relative gap between medium and small models is less pronounced in the set of economically relevant models. Second, it looks like there is a drop off in economically relevant models near the end of our sample, which is an artifact of the requirements to be economically relevant, where a number of models might not have had enough time to reach 2,013 downloads. This selection should have little impact on our analysis. Model downloads are concentrated in a model's early weeks, so these marginal models would largely belong in the lowest popularity bins.

Figure B.1: New model releases by week: economically relevant models



Note: new models are models uploaded to Hugging Face for the first time in a given week. Plots 3-week rolling average.

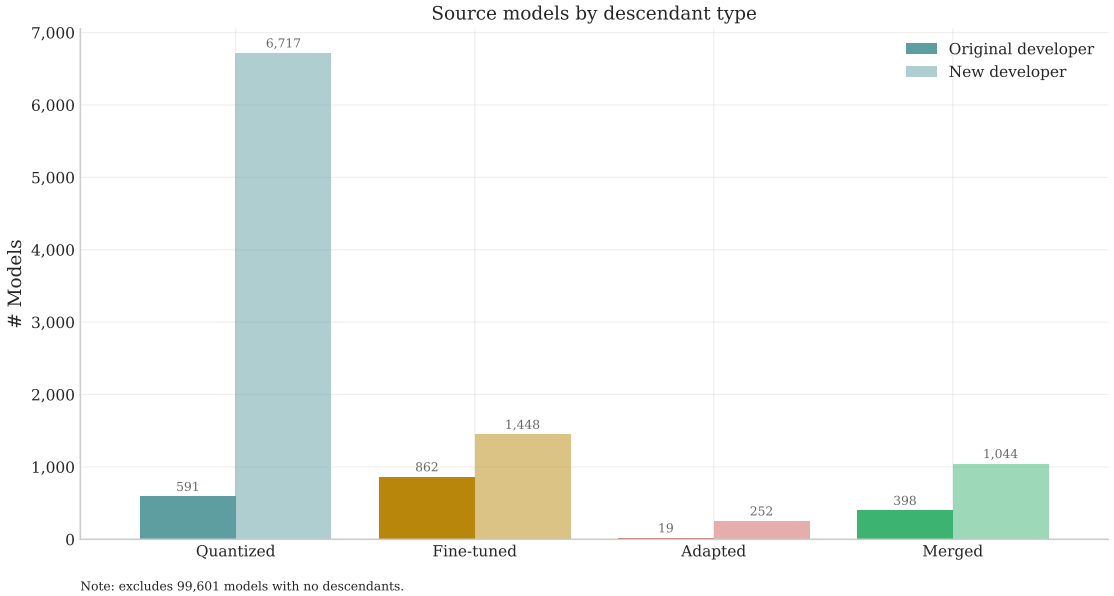
B.1 Batch releases

We assign all models in our sample to batches as follows:

1. Find the model with the earliest release date t_0 and assign it to batch 0
2. Assign all models with a release date in $[t_0, t_0 + 30]$ to batch 0.
3. Among unassigned models, assign the one with the earliest release date t_1 to batch 1.
4. Assign all models with a release date in $[t_1, t_1 + 30]$ to batch 1.
5. Repeat the above process until all models are assigned to batches.

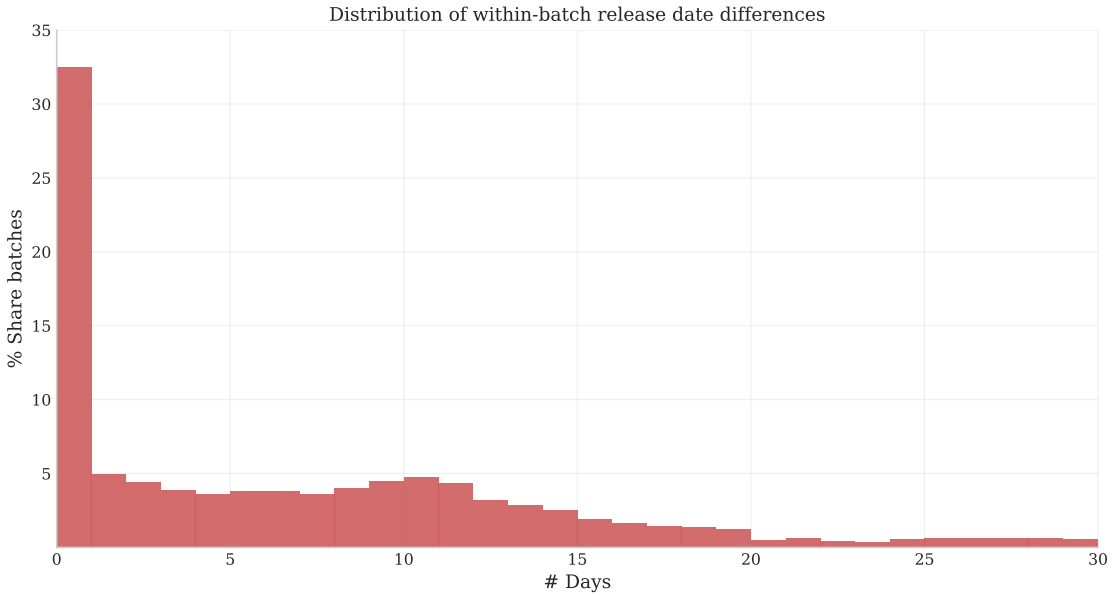
More than 30% of batches have a mean pairwise distance of one day, confirming that batches typically include models released close together.

Figure B.2: Source models by descendant type



Note: excludes 99,601 models with no descendants.

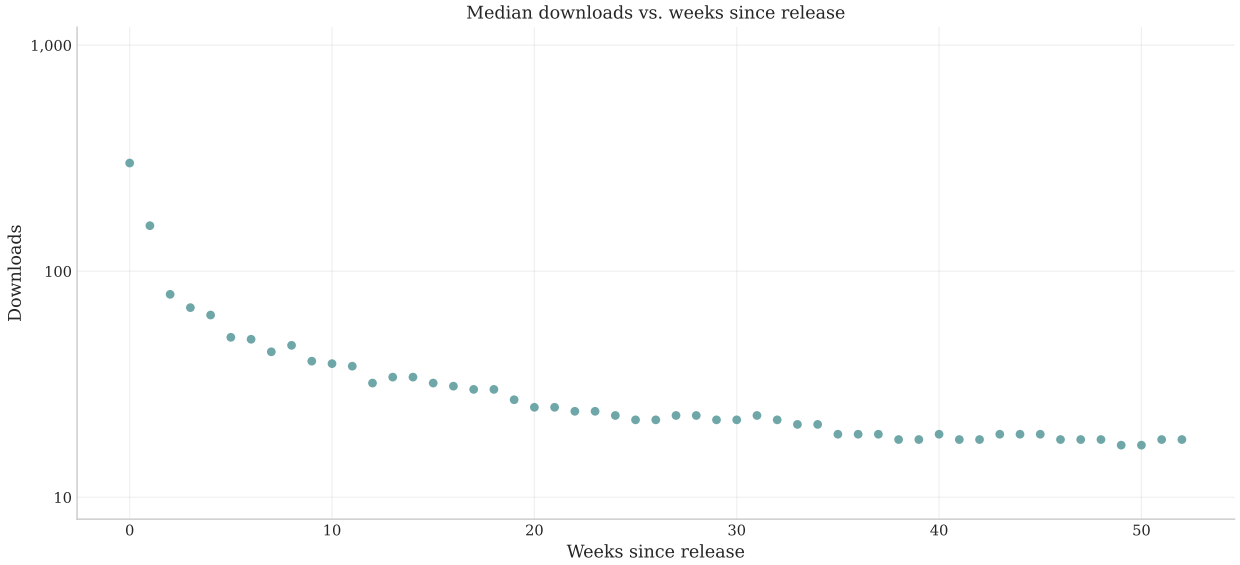
Figure B.3: Within-batch release date differences



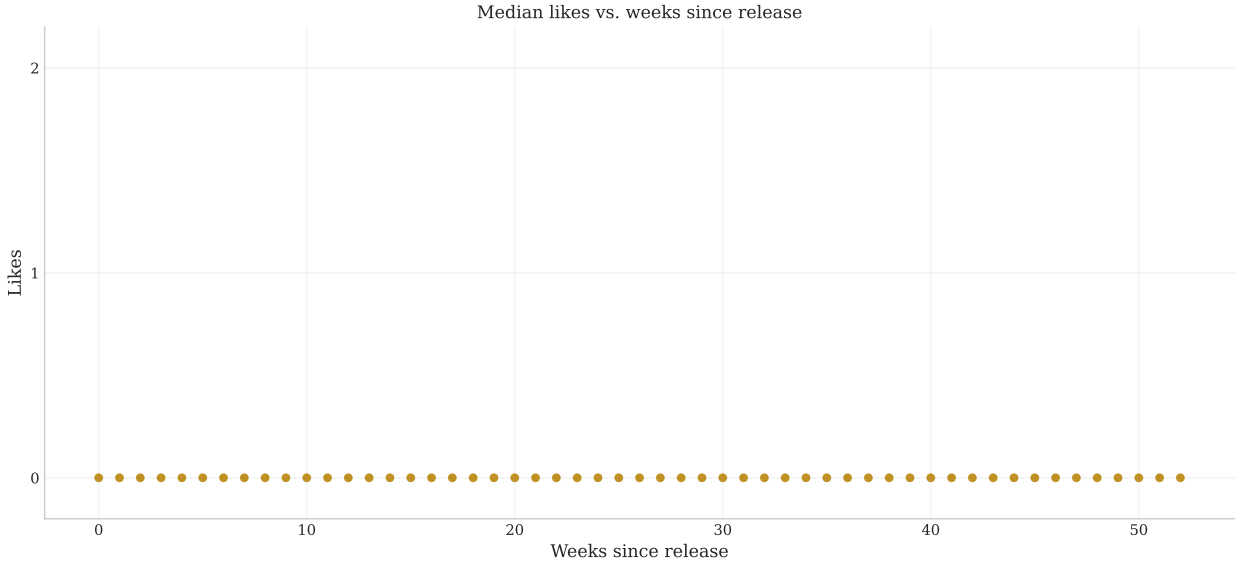
Note: Plots the distribution of the mean pairwise difference between model release dates in each batch.
Batches are groups of models released by the same author within 30 days of each other.
Includes batches with at least two models.

C Life

Figure C.1: Demand and attention at model release



Note: each dot shows median downloads for models at that week since release.



Note: each dot shows median likes for models at that week since release.

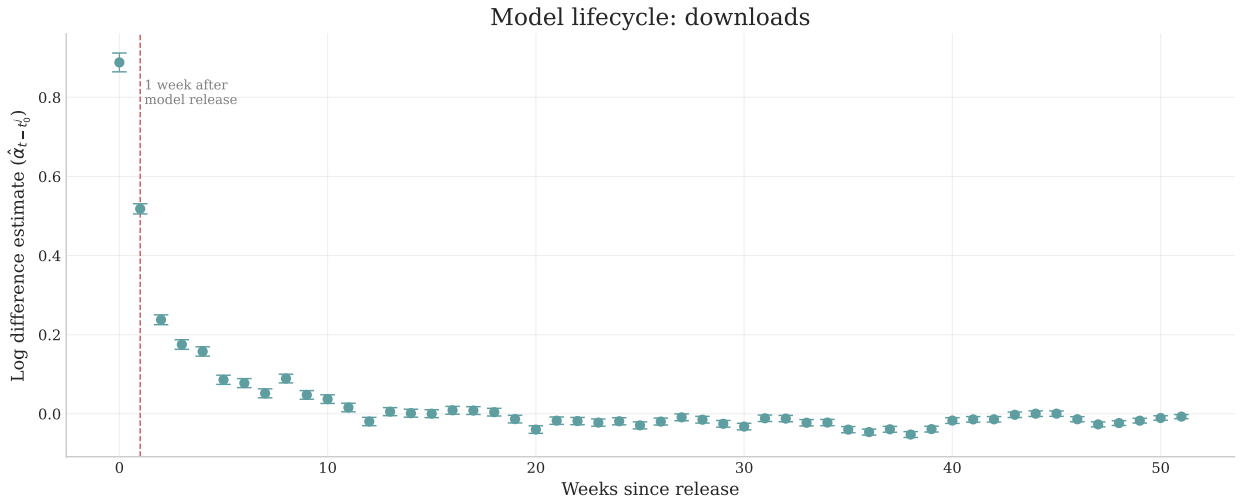
Although major model releases generate a lot of hype, it remains unclear whether the hype translates into actual usage that lasts beyond the initial release. When estimating this "hype-cycle" versus the lifecycle, simply averaging likes or downloads across models would mask important variation and trends. Instead, we allow for model age to more flexibly impact attention and demand. We control for model-specific features since models might have features that affect their use but do not reflect true utility, like better branding which attracts users but does not affect model performance. We also control for calendar time to address

broad trends in model use and adoption over time. For example, Hugging Face and large language models in general became more popular over our sample period, and the release of DeepSeek-R1 represented a unique moment in the history of transformer models. We want our picture of the model lifecycle to capture features inherent to the models rather than picking up broader trends in the data. We finally control for the day of week on which a model was released to address, for example, the possibility that Hugging Face users download models for work, in which case downloads would be less likely on weekends. Our full specification is the following:

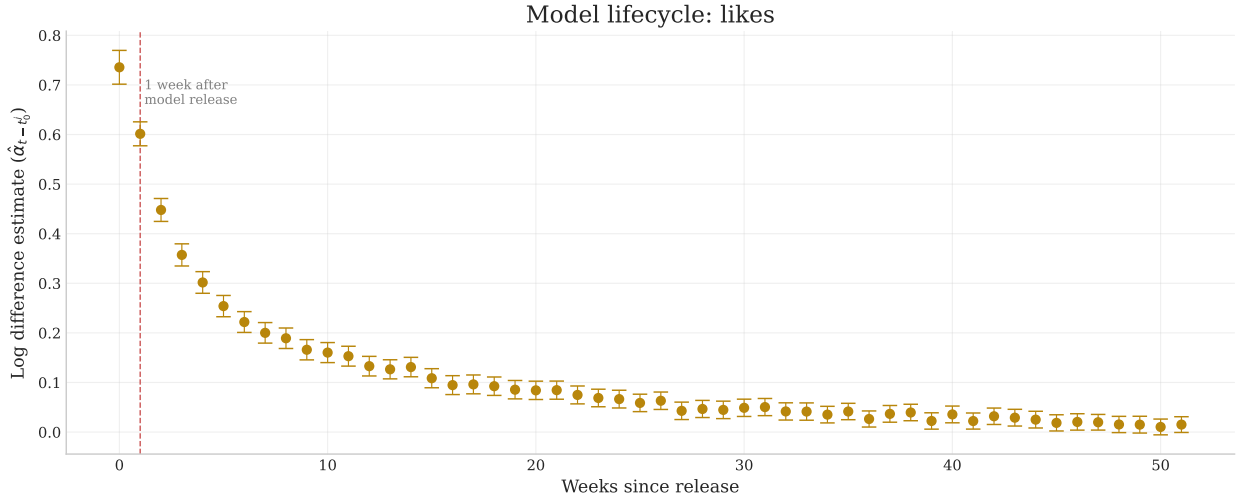
$$y_{jt} = \alpha_{t-t_0^j} + \alpha_{53}\mathbb{1}(t - t_0^j > 52) + \zeta_{dow}\mathbb{1}(t = t_0^j) + \xi_j + \delta_t + \varepsilon_{jt} \tag{C.1}$$

where the outcome is $y_{jt} = \log_{10} p_{jt}$ and p_{jt} is the number of downloads model j received in week t, and t_0^j indexes the calendar week in which model j was released. Our parameters of interest are the $\alpha_{t-t_0^j}$ terms, which are model age fixed effects for each $t - t_0^j \in \{0, 1, \ldots, 51\}$. The specification also includes model fixed effects ξ_j, calendar week fixed effects δ_t, and a day-of-week-of-release fixed effect ζ_{dow}. We cluster standard errors at the model level, and the excluded (baseline) model age is 52 ($\alpha_{52} = 0$). Because of the length of our sample, estimates of the model lifecycle beyond one year of release become increasingly noisy. We therefore focus our analysis on the first year of a model's life; this explains our choice to pool the model age variables beyond week 52.

Figure C.2: Demand and attention at model release



Note: Plots parameter estimates from the following linear regression:
Outcome: Base 10 log of weekly downloads.
Parameter of interest: Coefficient on week since release.
Additional controls: Day of week of release, calendar week fixed effects, model fixed effects.
Vertical axis shows change in log downloads relative to excluded week (52).



Note: Plots parameter estimates from the following linear regression:
Outcome: Base 10 log of weekly likes.
Parameter of interest: Coefficient on week since release.
Additional controls: Day of week of release, calendar week fixed effects, model fixed effects.
Vertical axis shows change in log likes relative to excluded week (52).

C.1 Lifecycle analysis by popularity

We estimate the following specification:

$$y_{jt} = \sum_{k=1}^{K} \left(\alpha_{t-t_0^j} + \alpha_{53} \mathbb{1}(t - t_0^j > 52) \right) \mathbb{1}(j \in Q_k) + \zeta_{dow} \mathbb{1}(t = t_0^j) + \xi_j + \delta_t + \varepsilon_{jt} \tag{C.2}$$

where

- $y_{jt} = \log_{10} p_{jt}$ where p_{jt} is the number of downloads model j received in week t.
- t_0^j is the calendar week in which model j was released.
- $\alpha_{t-t_0^j}$ is a model age fixed effect for each $t - t_0^j \in \{0, 1, \ldots, 51\}$.
- Q_k is the set of models that fall in a particular percentile of week-1 downloads when compared to other models at the same calendar time.
- ζ_{dow} is a day of week of release fixed effect.
- ξ_j is a model fixed effect.
- δ_t is a calendar week fixed effect.
- Standard errors are clustered at the model level.
- The excluded model age is α_{52}.

Note that there are actually multiple excluded fixed effects in Equation C.2, namely $\alpha_{52}\mathbb{1}(j \in Q_k)$ for each k. To understand this, first write a more explicit version of the regression equation without any excluded variables

$$y_{jt} = \sum_{k=1}^{K} \left(\sum_{\tau=0}^{52} \alpha_\tau \mathbb{1}(t - t_0^j = \tau) + \alpha_{53} \mathbb{1}(t - t_0^j > 52) \right) \mathbb{1}(j \in Q_k) + \zeta_{dow} \mathbb{1}(t = t_0^j) + \xi_j + \delta_t + \varepsilon_{jt} \tag{C.3}$$

which, without any excluded variables, collapses to

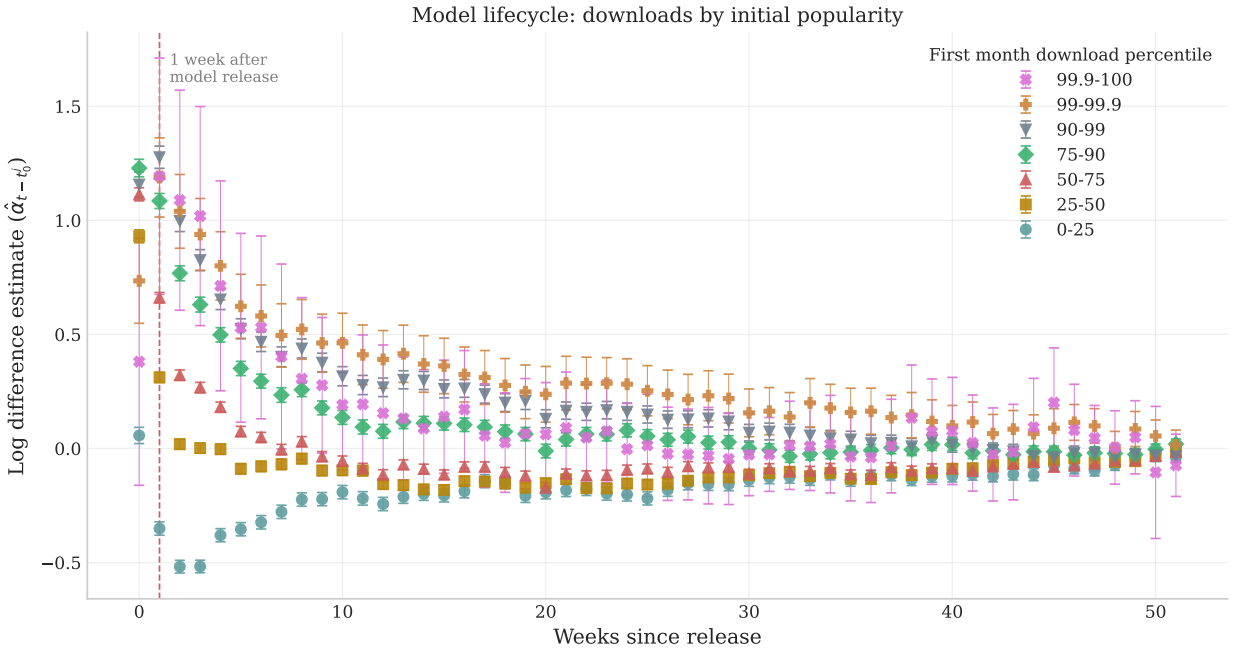
$$y_{jt} = \sum_{k=1}^{K} \alpha_k \mathbb{1}(j \in Q_k) + \zeta_{dow} \mathbb{1}(t = t_0^j) + \xi_j + \delta_t + \varepsilon_{jt}$$

This equation cannot be estimated because the indicator $\mathbb{1}(j \in Q_k)$ is equal to the sum of the model fixed effects ξ_j over all $j \in Q_k$. Now suppose we exclude a single variable, say $\alpha_{52}\mathbb{1}(t - t_0^j = 52)\mathbb{1}(j \in Q_K)$. Then Equation C.3 collapses to

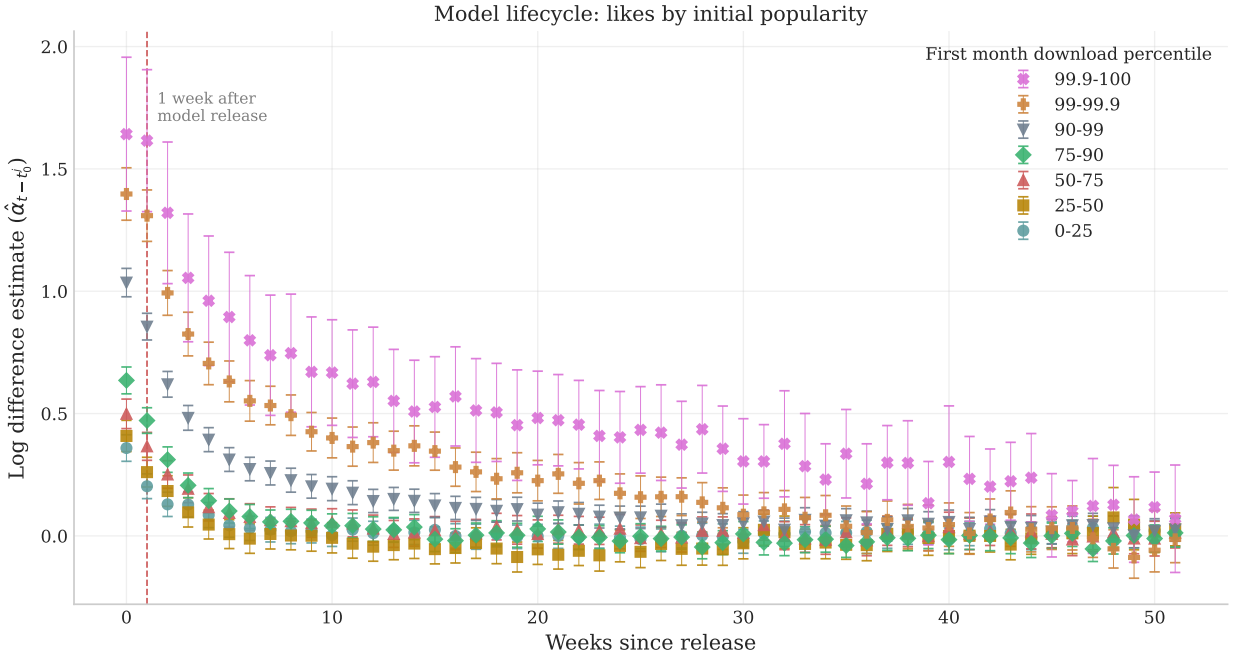
$$y_{jt} = \sum_{k=1}^{K-1} \alpha_k \mathbb{1}(j \in Q_k) + \left(\sum_{\tau=0}^{51} \alpha_\tau \mathbb{1}(t - t_0^j = \tau) + \alpha_{53} \mathbb{1}(t - t_0^j > 52) \right) \mathbb{1}(j \in Q_K) + \zeta_{dow} \mathbb{1}(t = t_0^j) + \xi_j + \delta_t + \varepsilon_{jt}$$

This eliminates the collinearity problem for models $j \in Q_K$, but not for $j \in Q_1, \ldots, Q_{K-1}$. Therefore we need to exclude $\alpha_{52}\mathbb{1}(t - t_0^j = 52)\mathbb{1}(j \in Q_k)$ for *all* $k \in \{1, \ldots, K\}$.

Figure C.3: Demand and attention at model release



Note: Plots parameter estimates from the following linear regression:
Outcome: Base 10 log of weekly downloads.
Parameter of interest: Coefficient on week since release interacted with first month download percentile.
Additional controls: Day of week of release, calendar week fixed effects, model fixed effects.
Vertical axis shows change in log downloads relative to excluded week (52).
Models with zero downloads in the first 4 weeks are included as a separate group in the regression but omitted from the plot.



Note: Plots parameter estimates from the following linear regression:
Outcome: Base 10 log of weekly likes.
Parameter of interest: Coefficient on week since release interacted with first month download percentile.
Additional controls: Day of week of release, calendar week fixed effects, model fixed effects.
Vertical axis shows change in log likes relative to excluded week (52).
Models with zero downloads in the first 4 weeks are included as a separate group in the regression but omitted from the plot.

C.2 Lifecycle analysis by model size

As an additional check, Figure C.4 plots the median model lifecycle by parameter count rather than initial downloads. While there is a clear relationship between the shape of the lifecycle curve and model popularity, the relationship is less one-to-one with size. For example, mid-sized models with one billion to ten billion parameters see a faster decay in downloads than the smallest models (with under one billion parameters). Attention, though, remains rare. One month after release, the median model in all size classes receives zero likes.

Figure C.4: Median model lifecycle by size

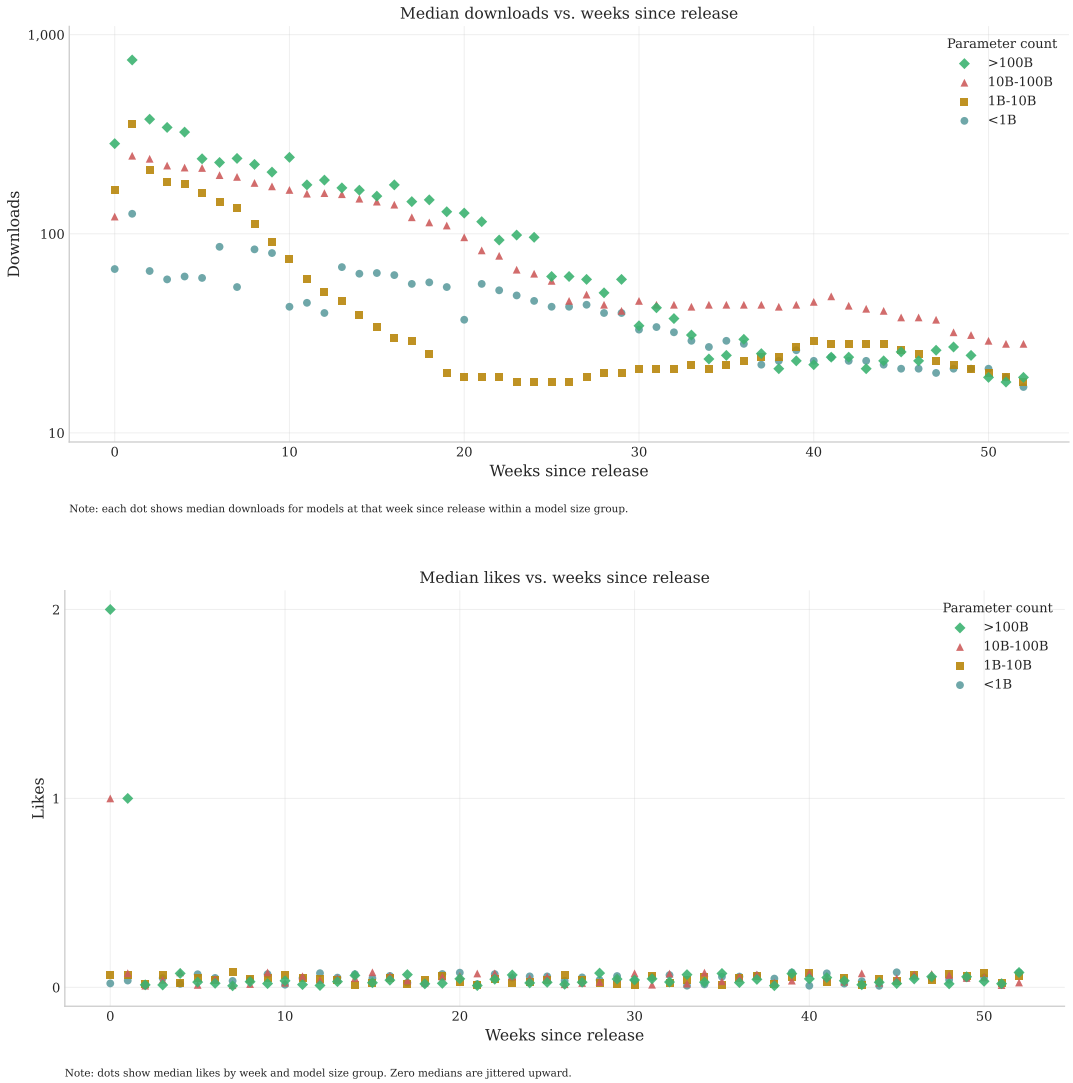
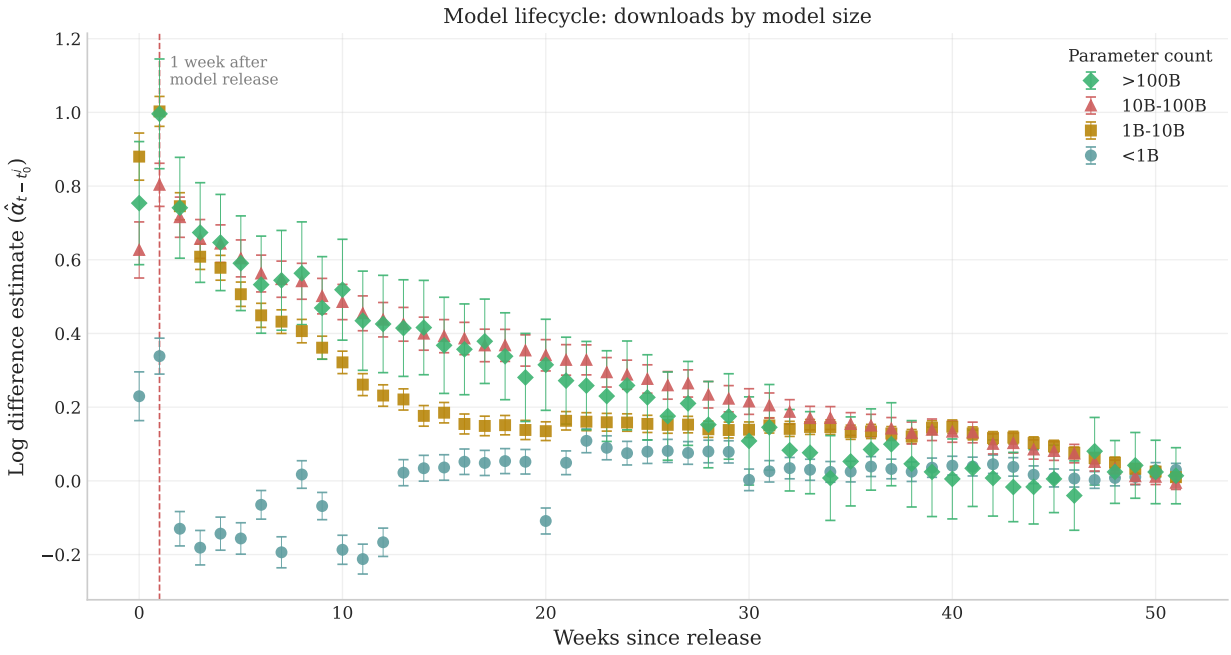
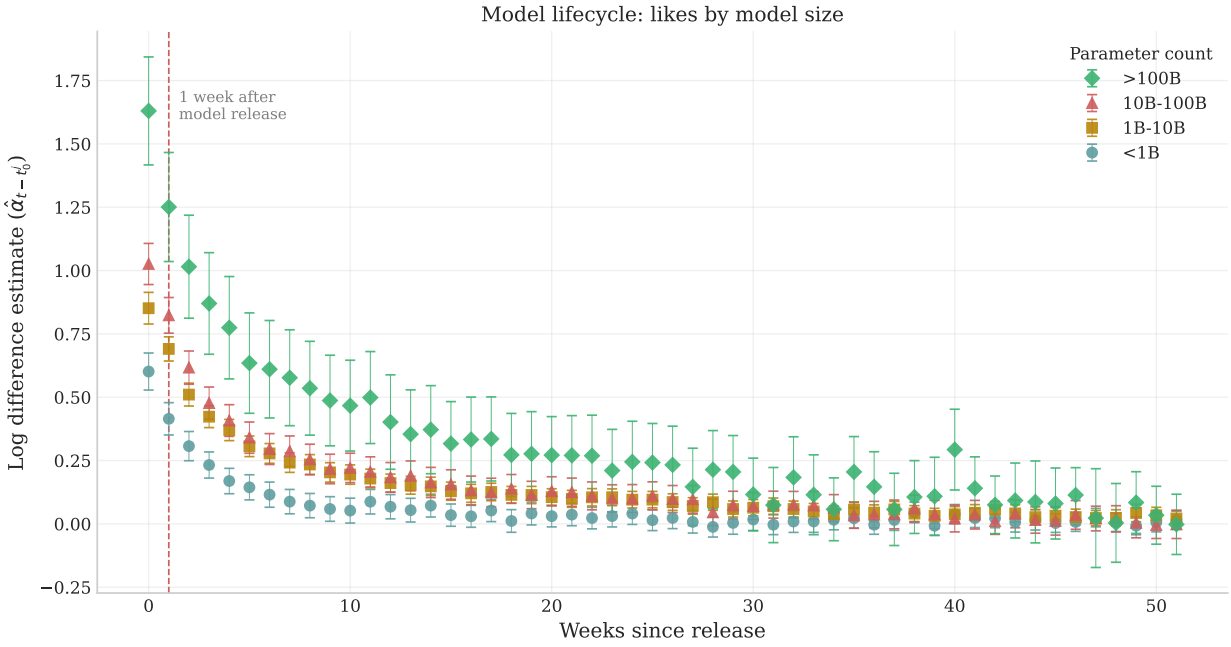


Figure C.5: Demand and attention at model release



Note: Plots parameter estimates from the following linear regression:
Outcome: Base 10 log of weekly downloads.
Parameter of interest: Coefficient on week since release interacted with model size group.
Additional controls: Day of week of release, calendar week fixed effects, model fixed effects.
Vertical axis shows change in log downloads relative to excluded week (52).

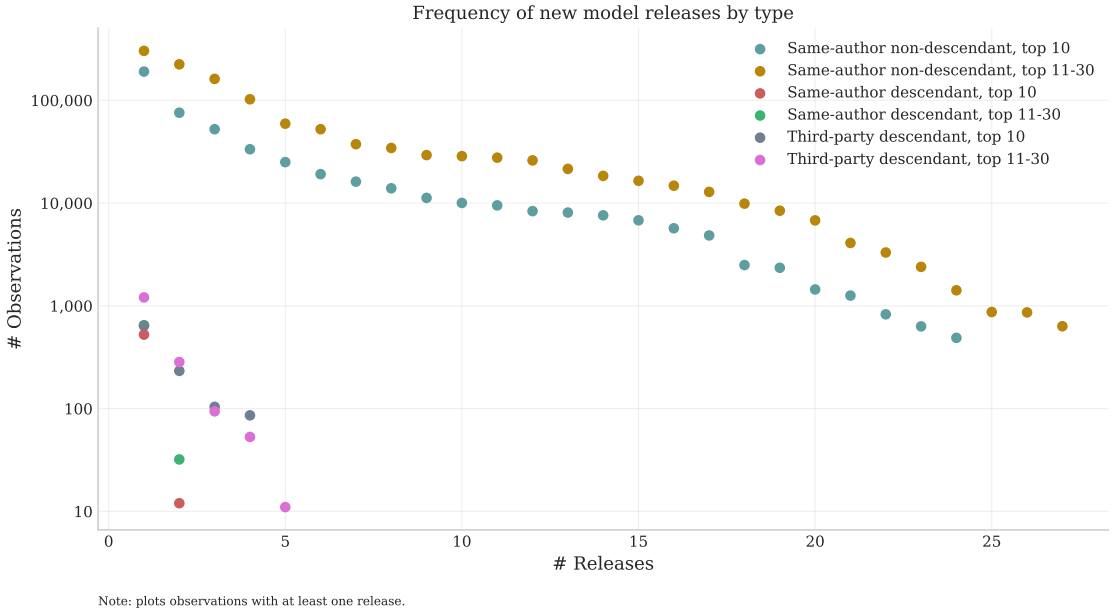


Note: Plots parameter estimates from the following linear regression:
Outcome: Base 10 log of weekly likes.
Parameter of interest: Coefficient on week since release interacted with model size group.
Additional controls: Day of week of release, calendar week fixed effects, model fixed effects.
Vertical axis shows change in log likes relative to excluded week (52).

C.3 Spillover effects

Here we present additional analysis of spillover effects from Section 4.2.

Figure C.6: Subsequent model release counts



Note: plots observations with at least one release.

C.3.1 Net Spillover Results from Same Author

Table C.1: Spillover effect estimates for same-author descendant models

	Downloads			Likes		
	Full	Top 10%	Top 1%	Full	Top 10%	Top 1%
Top 10						
1 release (recent)	0.3635	0.3402	0.2865	-0.0709	-0.1278	0.3194
	(0.2833)	(0.3004)	(0.3678)	(0.2825)	(0.2948)	(0.2117)
2 releases (recent)	-0.8493**			0.9314***		
	(0.3567)			(0.3478)		
1 release (mature)	0.0857	0.1566	-0.1157	-0.3648**	-0.4017**	-0.5073***
	(0.2505)	(0.2521)	(0.2803)	(0.1499)	(0.1587)	(0.1612)
2 releases (mature)	1.463***			-1.406***		
	(0.3169)			(0.2368)		
Top 11–30						
1 release (recent)	0.1372	0.1892	0.4415	0.7655***	0.8144***	0.2752
	(0.5274)	(0.5236)	(0.6687)	(0.1736)	(0.1628)	(0.2148)
2 releases (recent)	0.2301	0.1298		-9.777***	-9.763***	
	(0.4757)	(0.4857)		(0.3300)	(0.3507)	
1 release (mature)	0.2516	0.2103	-0.6173	-0.6654***	-0.6247***	-0.3283
	(0.4958)	(0.4909)	(0.8549)	(0.2059)	(0.1857)	(0.3925)
2 releases (mature)	-0.3958**	-0.3739*		9.368***	9.357***	
	(0.1715)	(0.1941)		(0.1873)	(0.2248)	
Model	Yes	Yes	Yes	Yes	Yes	Yes
Date (week)	Yes	Yes	Yes	Yes	Yes	Yes
Age*initial popularity	Yes	Yes	Yes	Yes	Yes	Yes
Observations	6,749,333	634,679	57,536	2,178,219	410,241	43,400
Pseudo R^2	0.91392	0.89650	0.90211	0.76988	0.82151	0.86886

Standard errors in parentheses clustered at the model level
*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1*

Table C.2: Spillover effect estimates for same-author non-descendant models

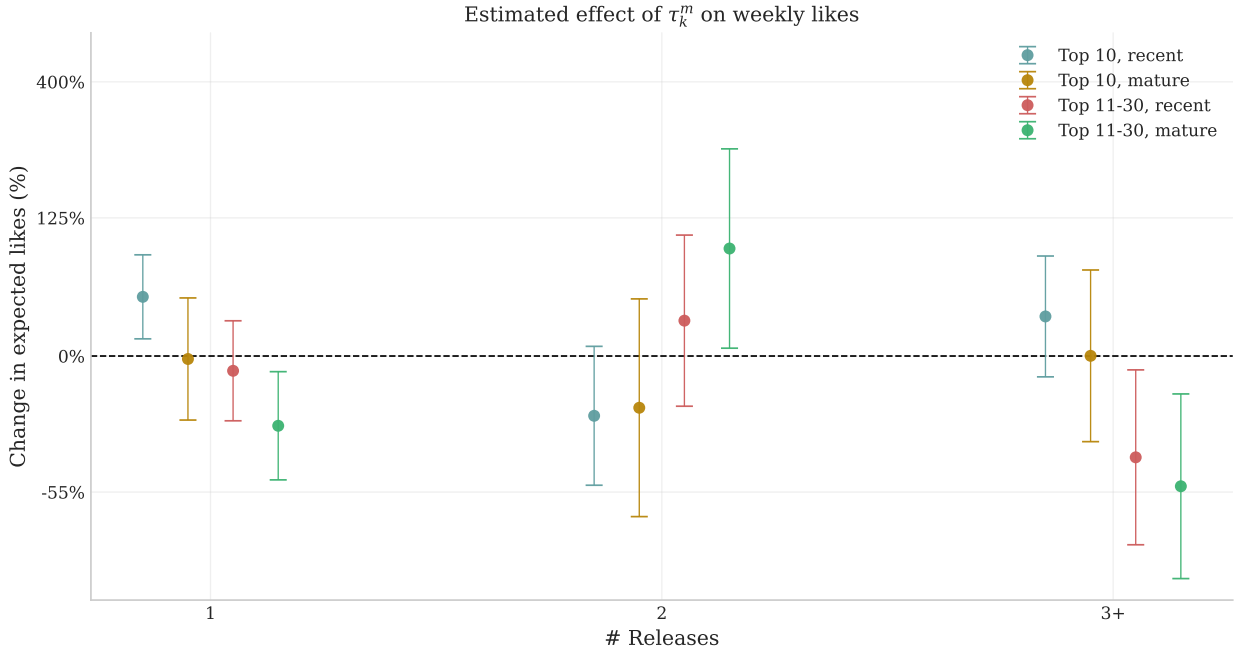
	Downloads			Likes		
	Full	Top 10%	Top 1%	Full	Top 10%	Top 1%
Top 10						
1 release (recent)	-0.1519	-0.2230	-0.5003*	-0.1860***	-0.1930***	-0.3835***
	(0.2010)	(0.2169)	(0.2831)	(0.0508)	(0.0554)	(0.0658)
2 releases (recent)	0.0312	0.0744	0.2334	0.1419**	0.2309***	0.2808***
	(0.1289)	(0.1351)	(0.1618)	(0.0640)	(0.0659)	(0.0759)
3 releases (recent)	-0.1618	-0.1929	-0.3373	-0.1101	-0.1961***	-0.2365***
	(0.1537)	(0.1598)	(0.2268)	(0.0685)	(0.0708)	(0.0880)
4 releases (recent)	-0.2207	-0.3384**	-0.1624	0.0201	0.0081	0.1018
	(0.1415)	(0.1614)	(0.2549)	(0.0730)	(0.0847)	(0.1486)
5 releases (recent)	0.0556	0.2699*	0.1310	0.0833	0.0919	0.1518
	(0.1360)	(0.1445)	(0.1911)	(0.0810)	(0.0883)	(0.1308)
6 releases (recent)	0.0005	-0.0616	0.1120	0.2835**	0.2696**	0.3143***
	(0.1348)	(0.1534)	(0.1929)	(0.1257)	(0.1264)	(0.1157)
7 releases (recent)	0.6717**	0.7430**	0.4871	0.1895*	0.1284	0.1418
	(0.2920)	(0.3220)	(0.4279)	(0.1127)	(0.1097)	(0.1229)
8 releases (recent)	0.0212	0.0450	0.0443	0.0954	-0.1826	-0.2619
	(0.2696)	(0.2752)	(0.4109)	(0.1654)	(0.1251)	(0.2062)
9 releases (recent)	0.3928	0.3447	0.1751	0.0497	0.2577**	0.1905
	(0.2804)	(0.2914)	(0.4769)	(0.1582)	(0.1274)	(0.1986)
10+ releases (recent)	-0.5059***	-0.4955***	-0.7170***	0.0381	0.0732	0.0877
	(0.1796)	(0.1860)	(0.2375)	(0.1343)	(0.1362)	(0.2041)
Log(10+ releases) (recent)	1.454***	1.519***	1.472**	0.3775*	0.3217	0.2985
	(0.4585)	(0.4749)	(0.6994)	(0.2089)	(0.2164)	(0.4455)
1 release (mature)	0.0232	0.0592	0.0013	-0.0900	-0.0120	0.1365
	(0.1751)	(0.1903)	(0.2455)	(0.0708)	(0.0792)	(0.0971)
2 releases (mature)	0.0642	0.0490	0.0460	-0.0767	-0.0953	-0.1268*
	(0.1024)	(0.1097)	(0.1234)	(0.0656)	(0.0681)	(0.0730)
3 releases (mature)	-0.2815**	-0.2760**	-0.1556	-0.0003	-0.0352	-0.1344
	(0.1217)	(0.1266)	(0.1497)	(0.0684)	(0.0733)	(0.1033)
4 releases (mature)	0.0021	-0.2435	-0.5057*	-0.2461**	-0.2173*	-0.3308**
	(0.1747)	(0.1939)	(0.2991)	(0.1004)	(0.1131)	(0.1534)
5 releases (mature)	-0.1036	0.0405	0.2636	-0.1410	-0.0366	0.0847
	(0.1700)	(0.1870)	(0.3081)	(0.1056)	(0.1099)	(0.1452)
6 releases (mature)	-0.2709	-0.2782	-0.1946	-0.2903***	-0.3503***	-0.3650***
	(0.2319)	(0.2524)	(0.3463)	(0.0921)	(0.0993)	(0.1040)
7 releases (mature)	0.5801**	0.5659**	0.9188**	-0.0706	0.0614	0.1161
	(0.2684)	(0.2835)	(0.3825)	(0.1563)	(0.1687)	(0.2579)
8 releases (mature)	-0.2508	-0.2156	-0.5829	-0.1370	-0.1992	-0.2486
	(0.2916)	(0.3007)	(0.4993)	(0.1609)	(0.1732)	(0.2681)
9 releases (mature)	-0.2425	-0.2629	0.3161	-0.0053	0.0278	0.0849
	(0.2658)	(0.2761)	(0.4303)	(0.1351)	(0.1523)	(0.2512)
10+ releases (mature)	0.1674	0.2171	-0.0284	-0.2858*	-0.2988*	-0.0496
	(0.2010)	(0.2094)	(0.2803)	(0.1536)	(0.1634)	(0.2530)
Log(10+ releases) (mature)	-0.5756	-0.6961*	-0.4313	0.1334	0.1016	-0.0012
	(0.3754)	(0.3944)	(0.5370)	(0.1976)	(0.2085)	(0.4323)
Top 11–30						
1 release (recent)	0.8501***	0.8806***	0.6061**	0.0625	0.0671	0.1450
	(0.2421)	(0.2397)	(0.2597)	(0.0485)	(0.0573)	(0.0950)
2 releases (recent)	-0.3495*	-0.3881*	-0.7292***	0.0844	0.0954	0.1579*
	(0.1867)	(0.2020)	(0.2546)	(0.0539)	(0.0587)	(0.0891)
3 releases (recent)	0.0369	0.1270	0.2413	0.0573	-0.0284	0.0225
	(0.1730)	(0.1853)	(0.2762)	(0.0718)	(0.0798)	(0.1036)
4 releases (recent)	0.1393	0.1110	0.2673	0.2350***	0.2586**	0.3898***
	(0.1341)	(0.1478)	(0.1734)	(0.0864)	(0.1028)	(0.1347)
5 releases (recent)	-0.0160	-0.0375	-0.5377*	-0.0405	-0.0039	-0.2248*
	(0.1830)	(0.1920)	(0.3014)	(0.0838)	(0.0925)	(0.1360)
6 releases (recent)	-0.2448*	-0.2466*	0.2988	-0.1148	-0.1167	0.1136
	(0.1391)	(0.1476)	(0.3415)	(0.0968)	(0.0966)	(0.1388)
7 releases (recent)	0.2953**	0.3040**	0.0990	0.3606***	0.2927***	0.1754
	(0.1249)	(0.1313)	(0.2667)	(0.0974)	(0.1013)	(0.1312)
8 releases (recent)	-0.3619*	-0.3624	-0.5517**	-0.0398	0.0288	-0.0017
	(0.2089)	(0.2204)	(0.2444)	(0.0804)	(0.0909)	(0.1307)
9 releases (recent)	0.1956	0.1956	0.3946**	-0.2127*	-0.1961	-0.4152**
	(0.1584)	(0.1659)	(0.1736)	(0.1145)	(0.1276)	(0.1877)
10+ releases (recent)	-0.4385*	-0.4615*	-1.069***	-0.1930*	-0.1957	-0.0644
	(0.2439)	(0.2565)	(0.3044)	(0.1159)	(0.1263)	(0.1495)
Log(10+ releases) (recent)	-0.2841	-0.3880	0.4489	0.5365***	0.6228***	0.6367
	(0.4671)	(0.5046)	(0.4726)	(0.1900)	(0.2123)	(0.4191)
1 release (mature)	-1.119***	-1.136***	-0.5641	-0.1153**	-0.0839	-0.1747**
	(0.4314)	(0.4310)	(0.3681)	(0.0454)	(0.0546)	(0.0874)
2 releases (mature)	0.6580**	0.6134**	0.4313*	0.0338	0.0465	0.0270
	(0.2876)	(0.2979)	(0.2617)	(0.0471)	(0.0549)	(0.0887)
3 releases (mature)	0.1297	0.1283	0.1971	-0.0467	-0.0261	-0.1700*
	(0.1453)	(0.1591)	(0.2373)	(0.0541)	(0.0651)	(0.0943)
4 releases (mature)	-0.0527	-0.0583	0.0385	0.0738	0.0014	0.0492
	(0.1534)	(0.1663)	(0.2068)	(0.0814)	(0.0941)	(0.1303)
5 releases (mature)	-0.1200	-0.1426	-0.6901	0.1536	0.1329	-0.0983
	(0.2076)	(0.2190)	(0.4370)	(0.1027)	(0.1160)	(0.1621)
6 releases (mature)	0.4819	0.5651	0.6297	-0.3000***	-0.2325***	-0.0509
	(0.3789)	(0.3990)	(0.3896)	(0.0830)	(0.0889)	(0.1177)
7 releases (mature)	0.1144	0.1012	0.3672	0.1965**	0.1963**	0.4128***
	(0.2549)	(0.2655)	(0.2336)	(0.0827)	(0.0911)	(0.1325)
8 releases (mature)	-0.0606	-0.0531	0.0535	0.3470***	0.3470***	0.3676***
	(0.0922)	(0.0984)	(0.1891)	(0.0812)	(0.0864)	(0.1287)
9 releases (mature)	-0.2334	-0.2954	-0.3806	0.0013	-0.0239	-0.3311*
	(0.1863)	(0.2092)	(0.3236)	(0.1213)	(0.1336)	(0.2009)
10+ releases (mature)	0.0008	0.0924	0.3859	-0.2768**	-0.3036**	0.1134
	(0.2547)	(0.2767)	(0.3226)	(0.1283)	(0.1479)	(0.2349)
Log(10+ releases) (mature)	0.4730	0.5142	-1.053***	0.0462	0.0137	-1.007**
	(0.4180)	(0.4483)	(0.2953)	(0.2023)	(0.2333)	(0.4630)
Model	Yes	Yes	Yes	Yes	Yes	Yes
Date (week)	Yes	Yes	Yes	Yes	Yes	Yes
Age*initial popularity	Yes	Yes	Yes	Yes	Yes	Yes
Observations	6,749,333	634,679	57,536	2,178,219	410,241	43,400
Pseudo R^2	0.91392	0.89650	0.90211	0.76988	0.82151	0.86886

Standard errors in parentheses clustered at the model level
*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1*

C.3.2 Likes Net Spillovers

Figure C.7 presents the results for third-party descendants, and the full set of parameter estimates is included in the regression tables alongside the demand results. The response of likes is more varied than for downloads. The sign and significance vary by relationship, popularity, order, and subsample. In part, this variation could be driven by the low number of likes most economically relevant models receive, as highlighted in Figure 4.1.

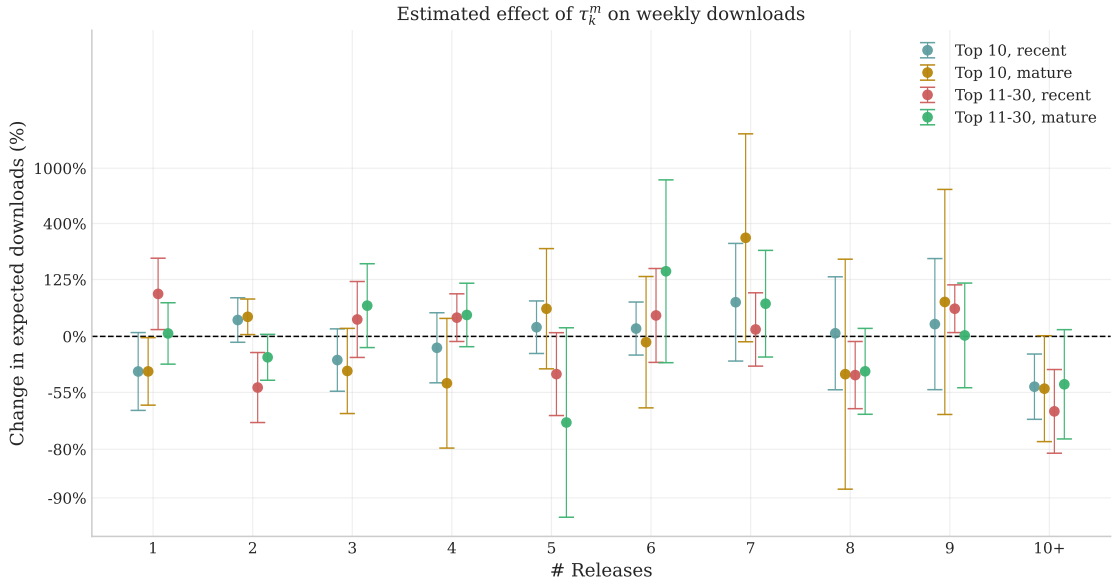
Figure C.7: Effects of subsequent model releases: third-party descendant models on likes



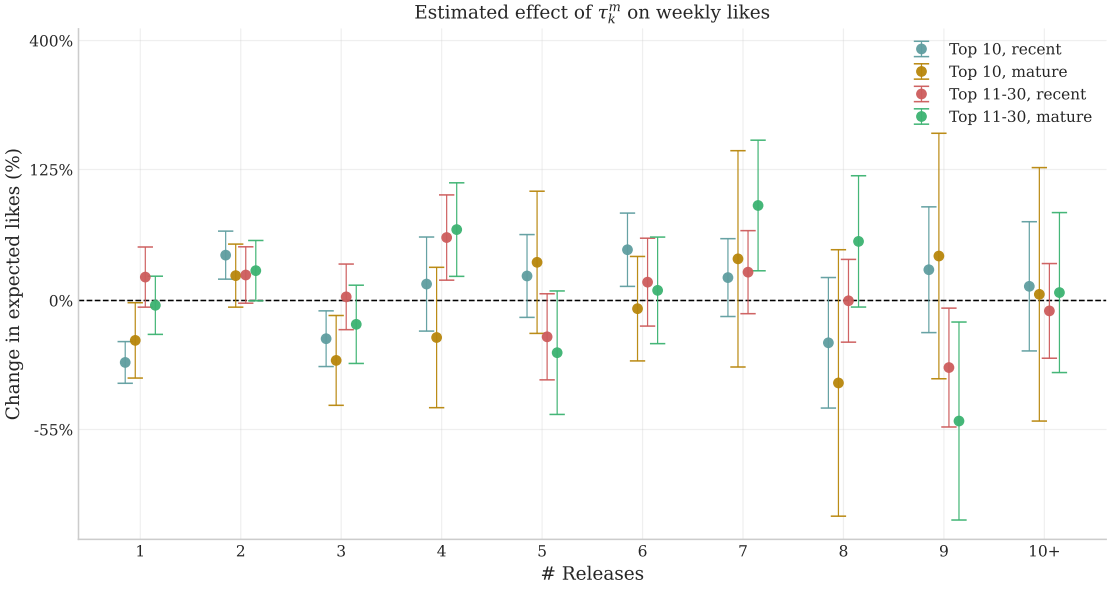
Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly likes on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects.

C.3.3 Net Spillover Results Coefficient Plots

Figure C.8: Spillover effects: same-author non-descendant releases, top 1% of focal models

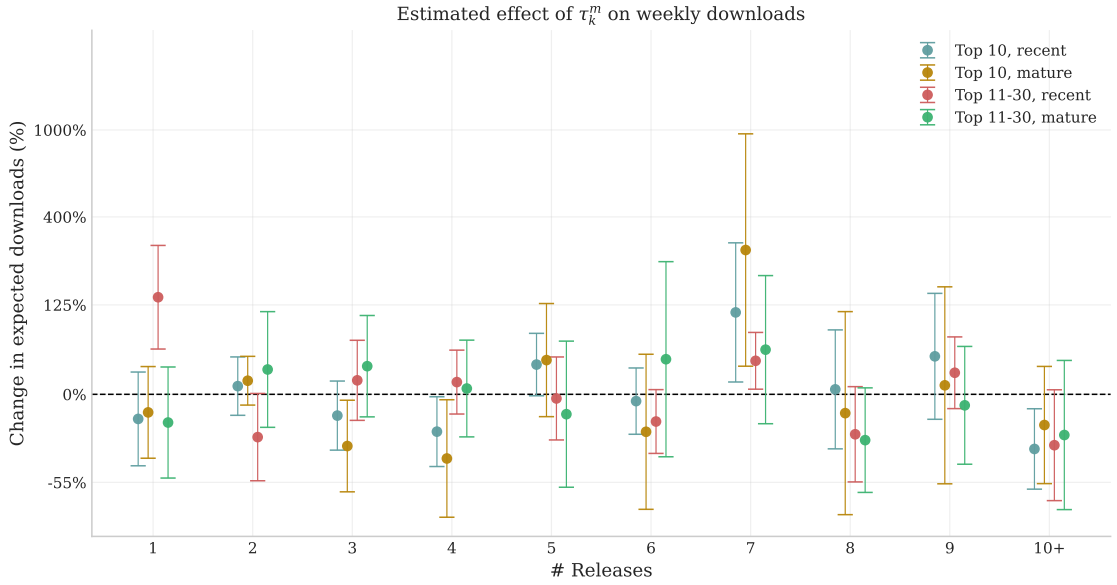


Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly downloads on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects, restricting to top 1% of models by initial popularity.

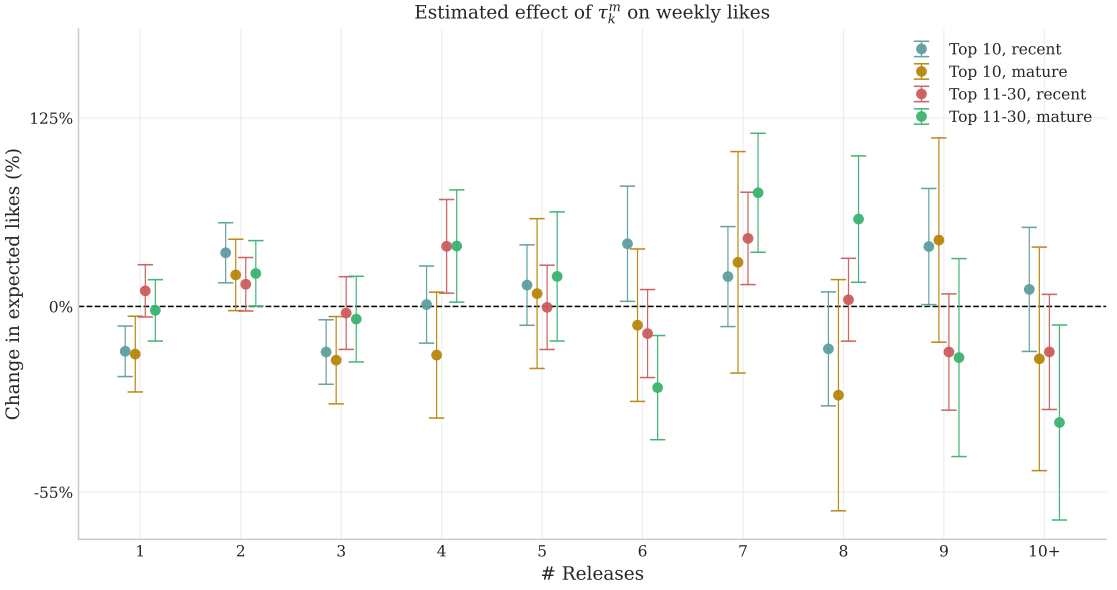


Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly likes on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects, restricting to top 1% of models by initial popularity.

Figure C.9: Spillover effects: same-author non-descendant releases, top 10% of focal models

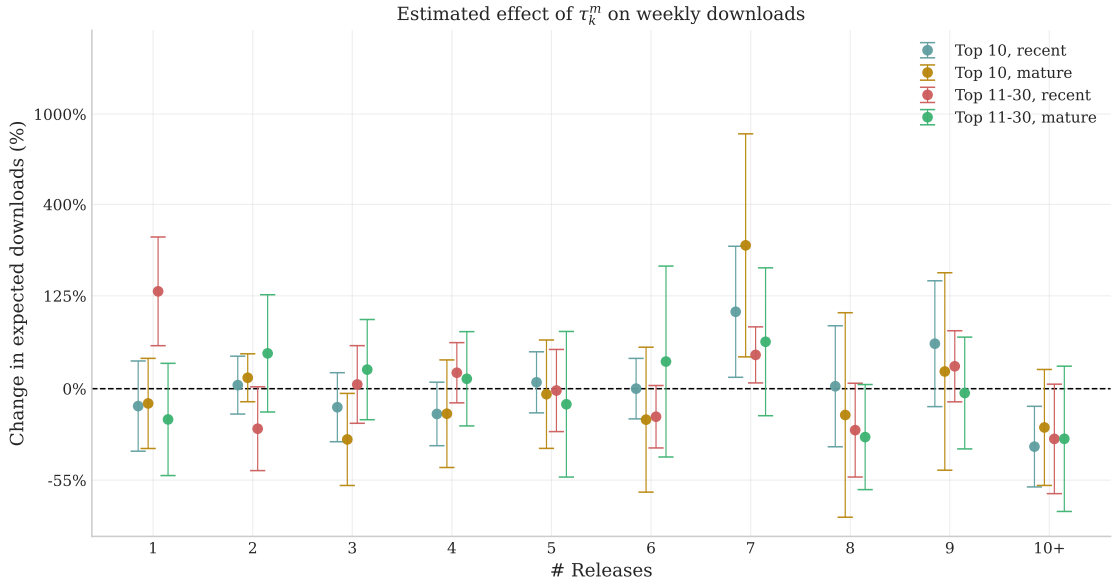


Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly downloads on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects, restricting to top 10% of models by initial popularity.

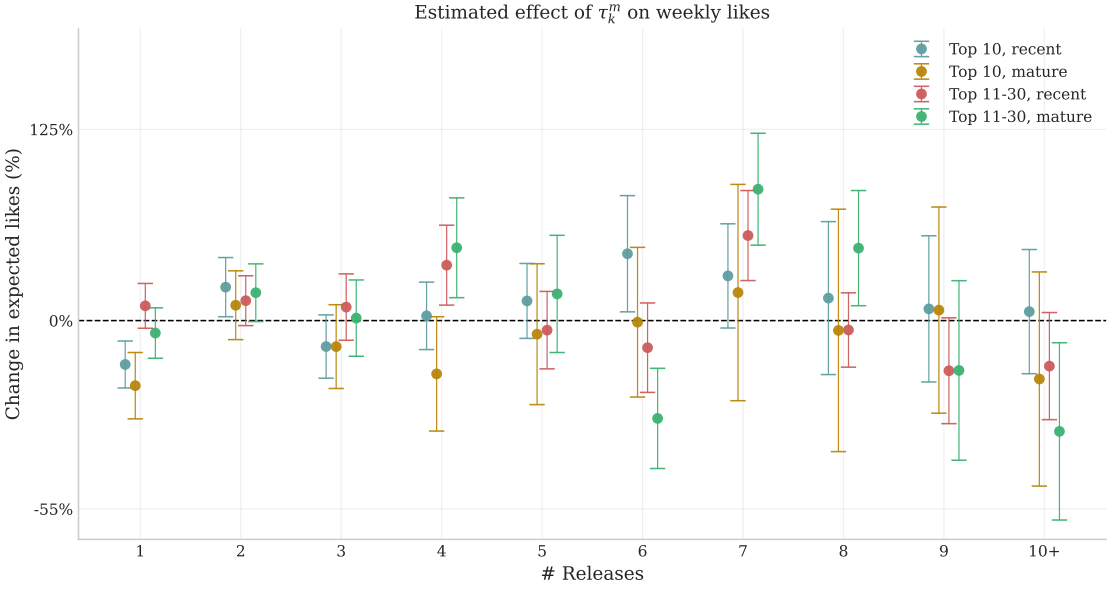


Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly likes on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects, restricting to top 10% of models by initial popularity.

Figure C.10: Spillover effects: same-author non-descendant releases, full sample

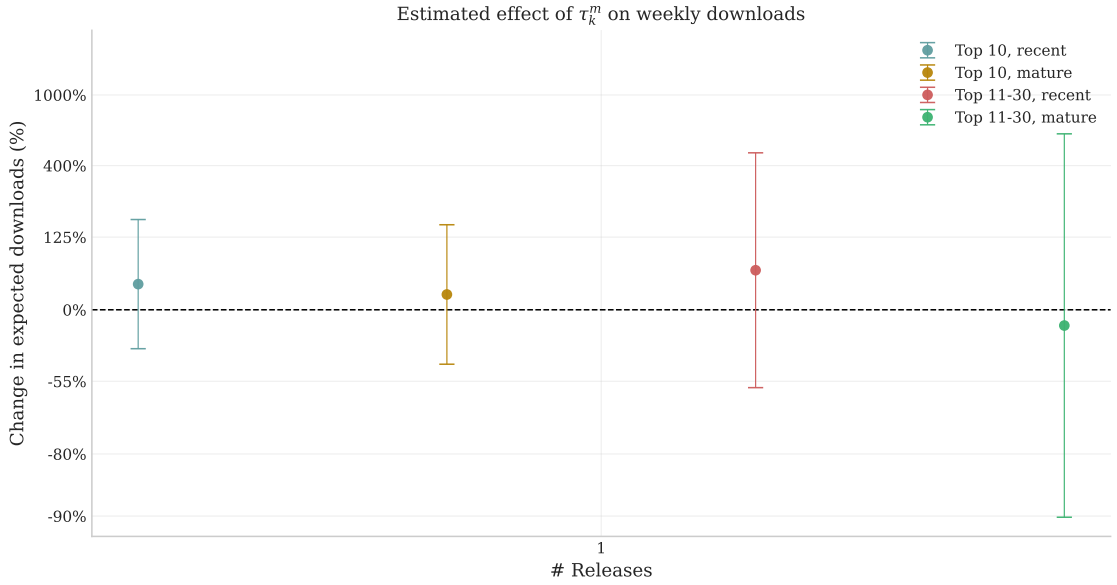


Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly downloads on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects.

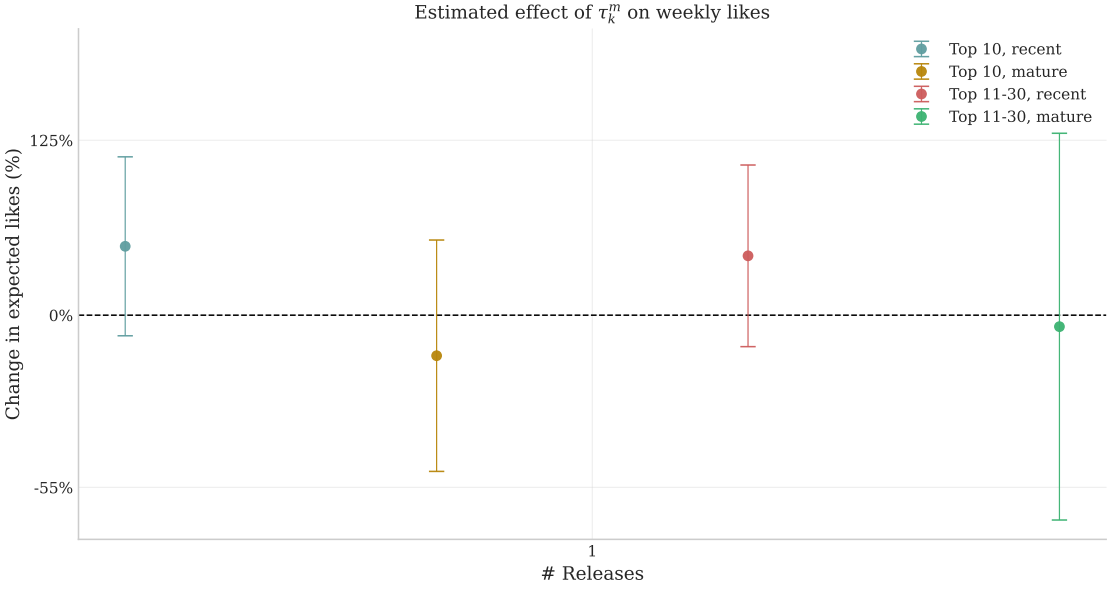


Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly likes on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects.

Figure C.11: Spillover effects: same-author descendant releases, top 1% of focal models

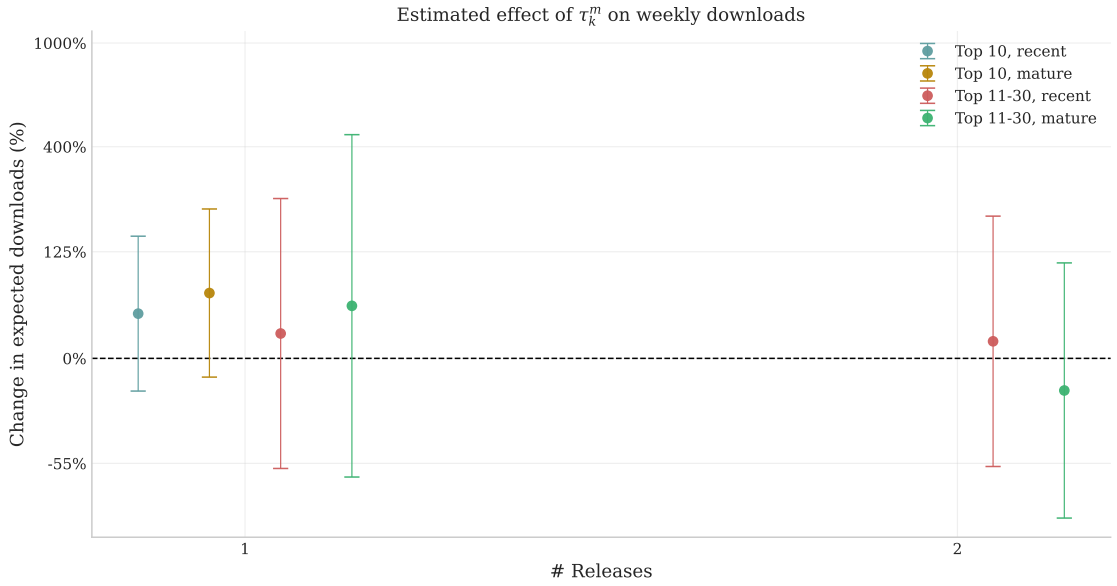


Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly downloads on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects, restricting to top 1% of models by initial popularity.

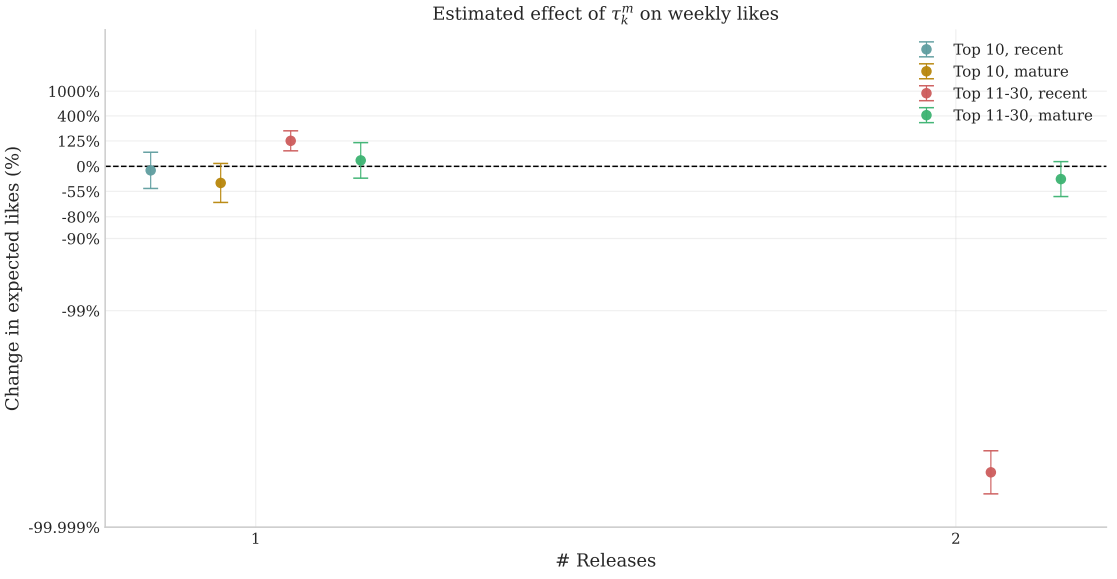


Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly likes on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects, restricting to top 1% of models by initial popularity.

Figure C.12: Spillover effects: same-author descendant releases, top 10% of focal models

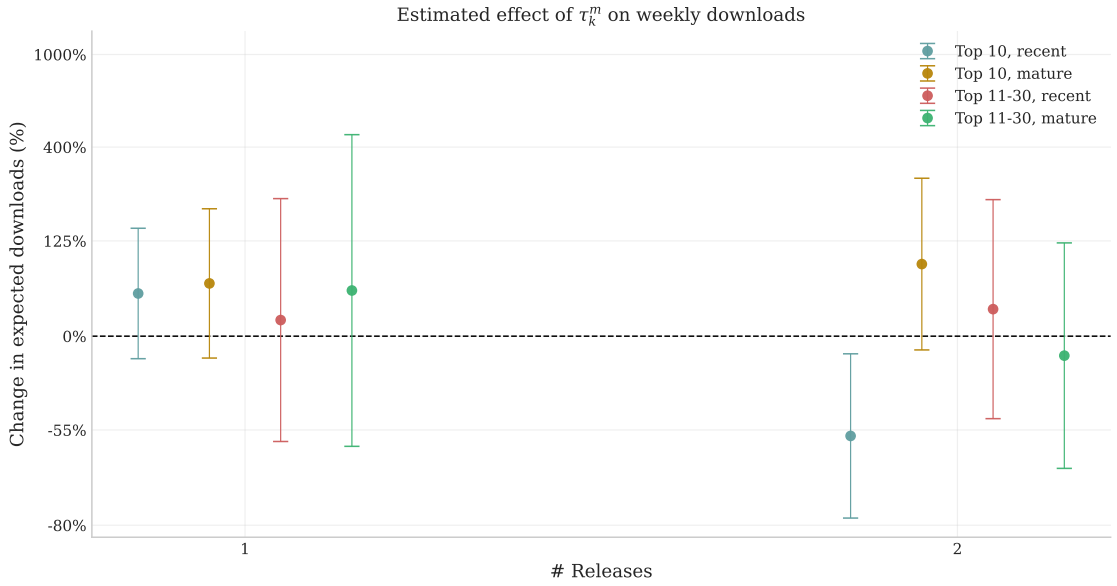


Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly downloads on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects, restricting to top 10% of models by initial popularity.

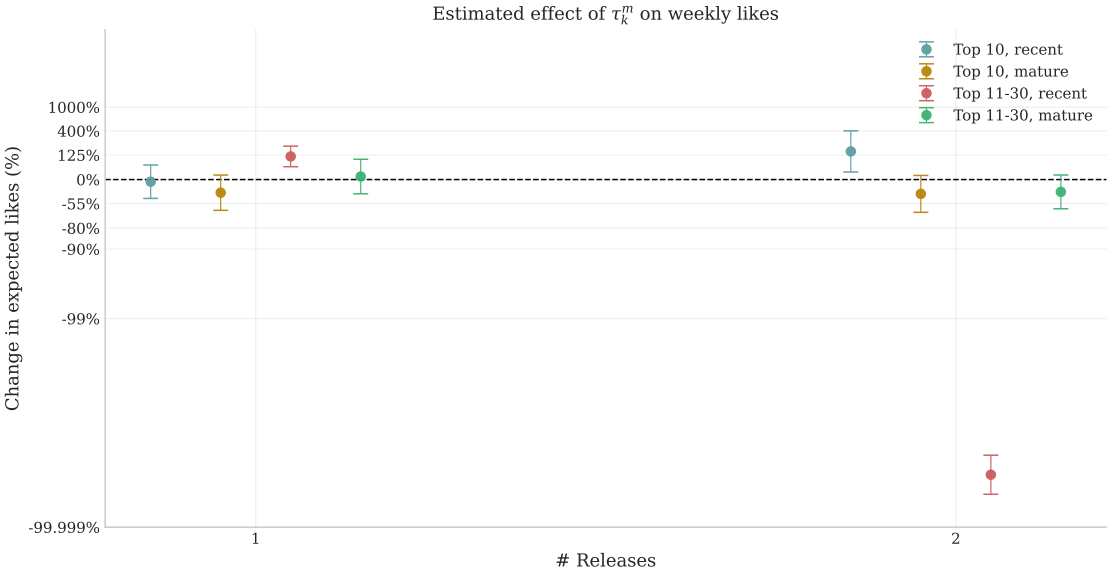


Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly likes on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects, restricting to top 10% of models by initial popularity.

Figure C.13: Spillover effects: same-author descendant releases, full sample

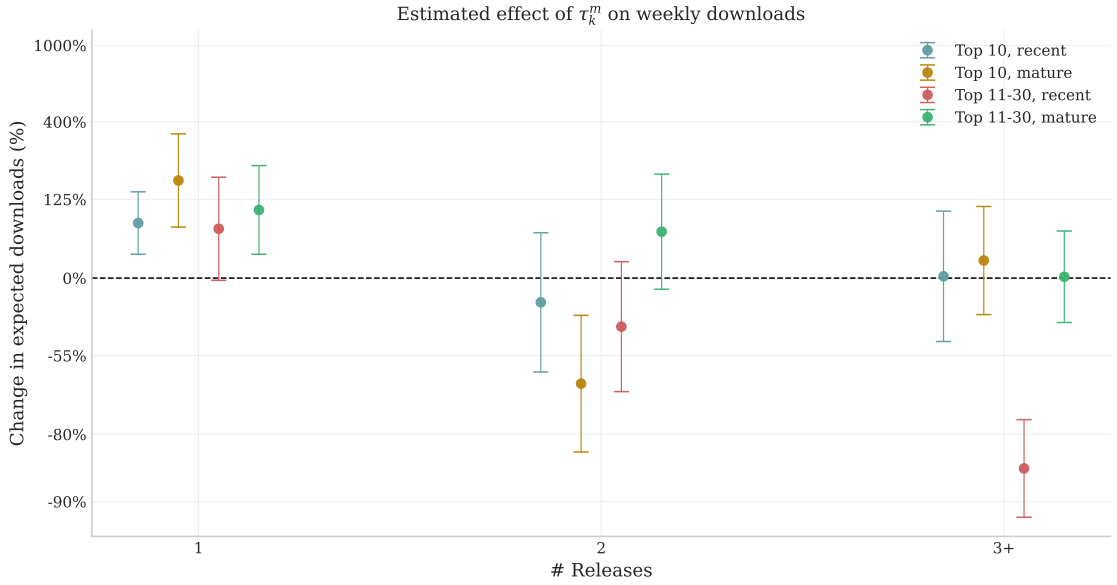


Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly downloads on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects.

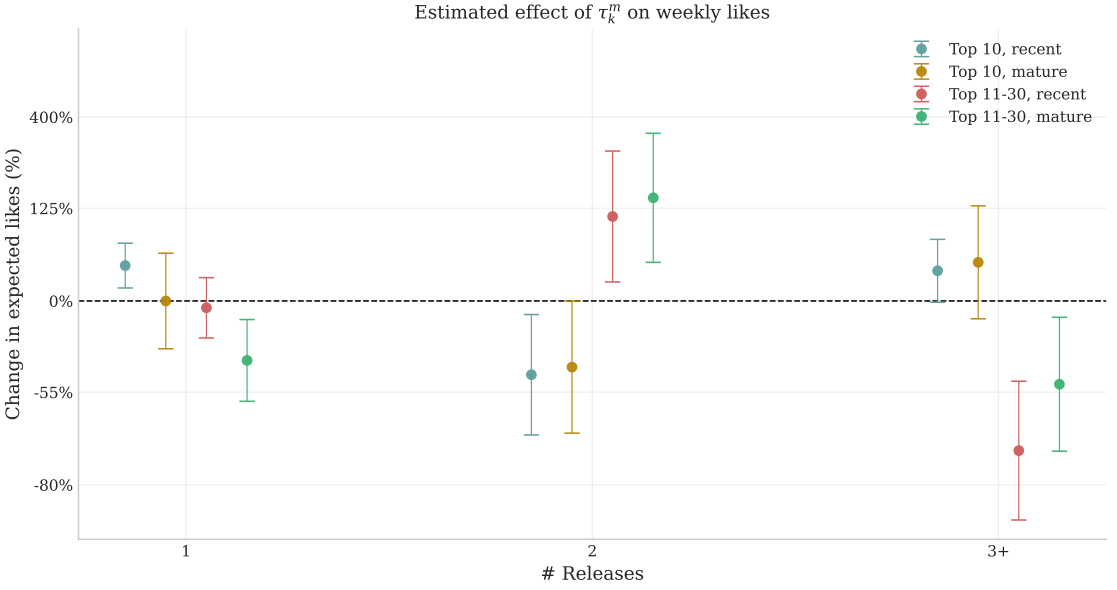


Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly likes on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects.

Figure C.14: Spillover effects: third-party descendant releases, top 1% of focal models

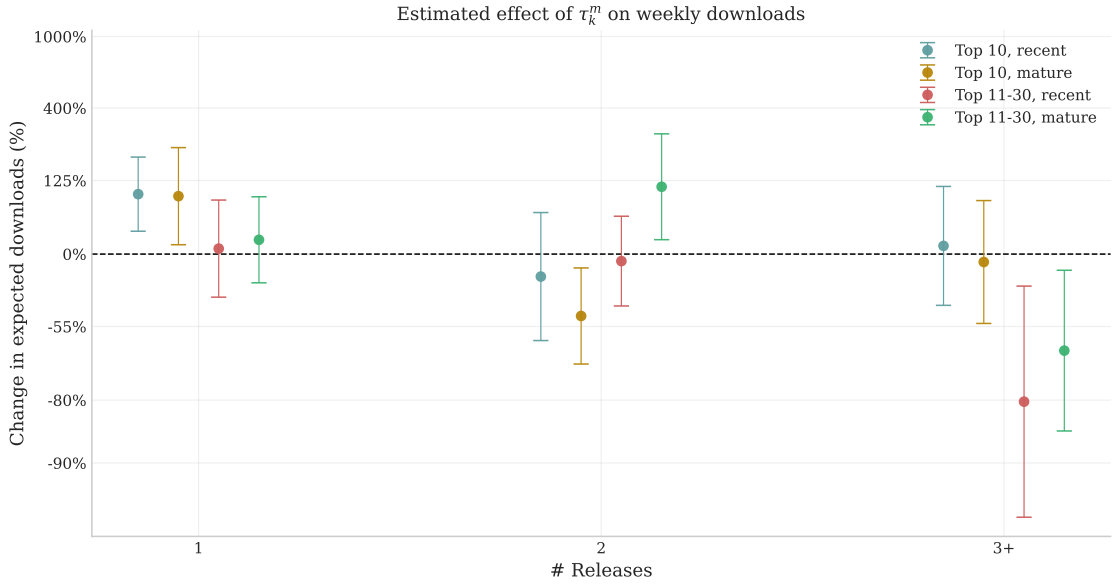


Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly downloads on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects, restricting to top 1% of models by initial popularity.

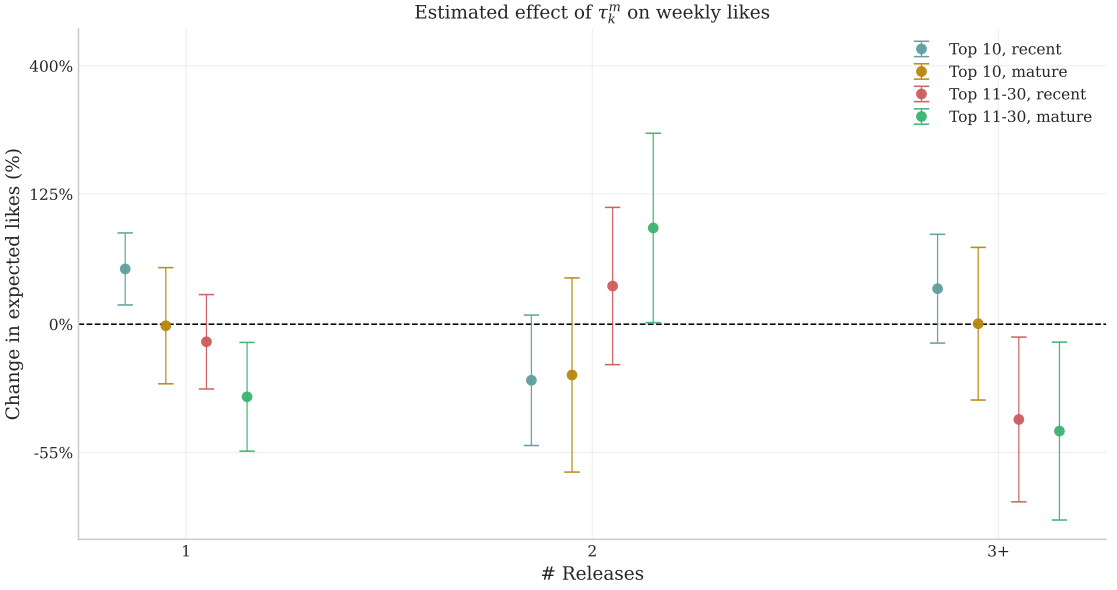


Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly likes on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects, restricting to top 1% of models by initial popularity.

Figure C.15: Spillover effects: third-party descendant releases, top 10% of focal models



Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly downloads on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects, restricting to top 10% of models by initial popularity.



Note: Plots estimated coefficients, converted to percent changes, from a Poisson regression of weekly likes on dummies for cumulative number of top 10 and top 11-30 model releases, split into short-run and long-run (4+ weeks) effects. Includes model fixed effects, calendar week fixed effects, and model age fixed effects, restricting to top 10% of models by initial popularity.

D Death

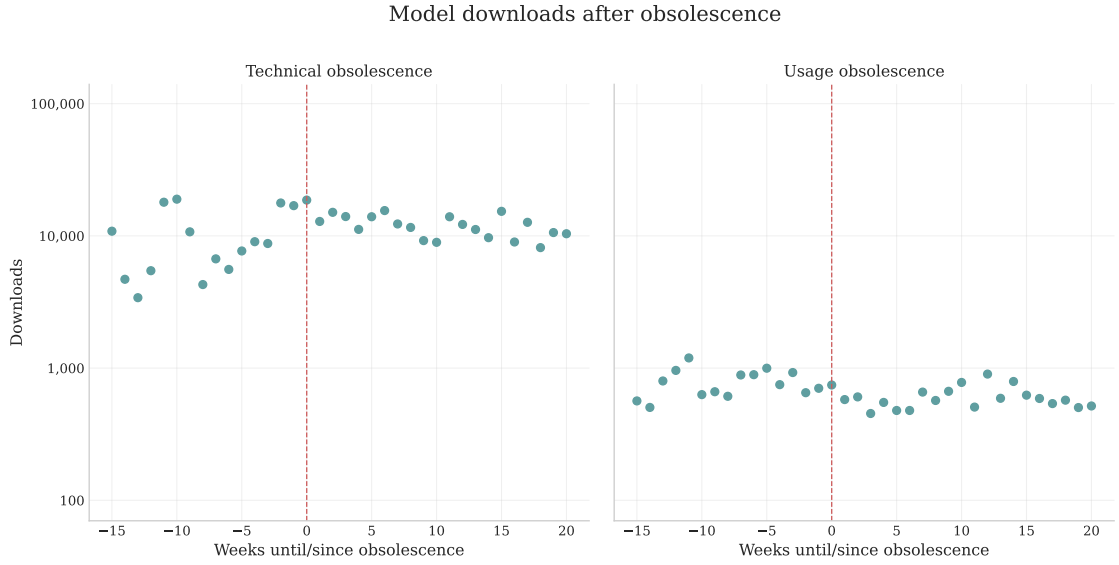
Table D.1: Usage obsolescence date differences

Model	Token share date	Dropout date	Difference (weeks)
TheDrummer/Anubis-70B-v1.1	2025-12-24	2025-12-31	1
allenai/molmo-7b-d-0924	2025-05-21	2025-11-05	24
google/gemini-flash-1.5-8b-exp	2025-04-30	2025-05-28	4
google/gemini-pro-vision	2025-05-14	2025-05-21	1
inclusionAI/Ling-1T	2025-11-26	2025-12-03	1
inclusionAI/Ring-1T	2025-11-26	2025-12-03	1
microsoft/phi-4-reasoning-plus-04-30	2026-01-28	2026-02-04	1
minimax/minimax-m1-40k	2025-06-18	2025-07-02	2
minimax/minimax-m1-80k	2025-06-18	2025-07-02	2
mistral/mistrall-small-2501	2025-02-05	2025-02-12	1
mistralai/Mixtral-8x7B-v0.1	2025-04-23	2025-04-30	1
moonshotai/Kimi-Linear-48B-A3B-Instruct	2025-12-24	2025-12-31	1
openai/shap-e	2024-11-06	2025-04-09	22
qwen/qwen2.5-7b-instruct	2025-05-21	2025-06-04	2
qwen/qwen3-1.7b-04-28	2025-06-04	2025-06-11	1
test/model	2026-01-14	2026-01-21	1
thudm/glm-4-32b-0414	2025-09-17	2025-10-01	2
thudm/glm-4-9b-0414	2025-05-28	2025-06-11	2
thudm/glm-z1-32b-0414	2025-08-27	2025-11-12	11
thudm/glm-z1-rumination-32b-0414	2025-07-16	2025-07-23	1

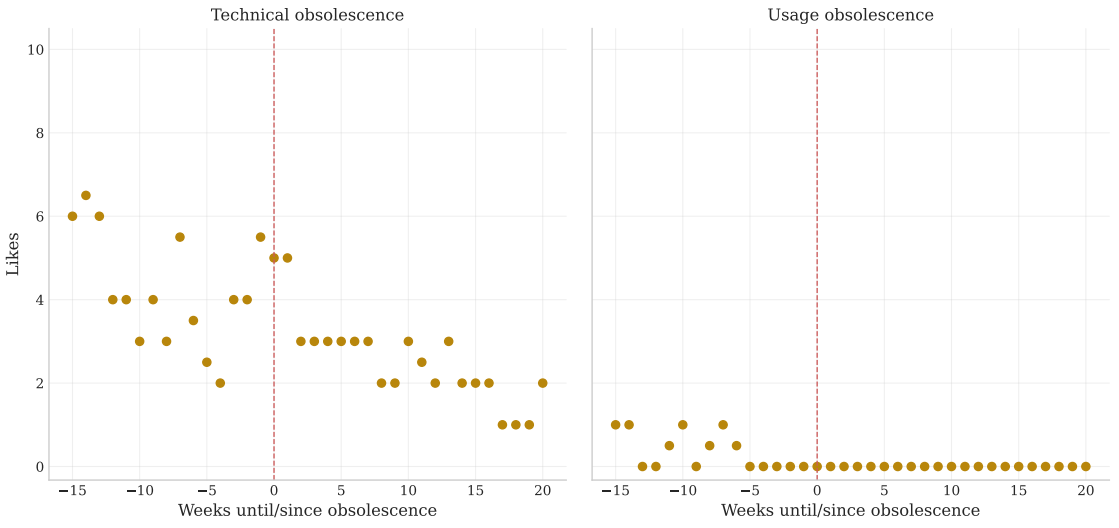
Table D.2: Best categories

Best category	Models
Artificial Analysis Benchmarks	
Intelligence index	162
Coding index	59
Math index	106
Agentic index	0
Total	327
Arena.ai	
Text style control	249
Vision style control	49
Search style control	19
Document style control	0
Webdev	15
Text to image	36
Image edit	23
Text to video	22
Image to video	19
Video edit	0
Total	432
Artificial Analysis Arena	
Text to image	92
Image editing	26
Text to video	54
Image to video	22
Total	194

Figure D.1: Demand and attention around obsolescence (medians)



Note: Plots median downloads by weeks until/since obsolescence.



Note: Plots median likes by weeks until/since obsolescence.

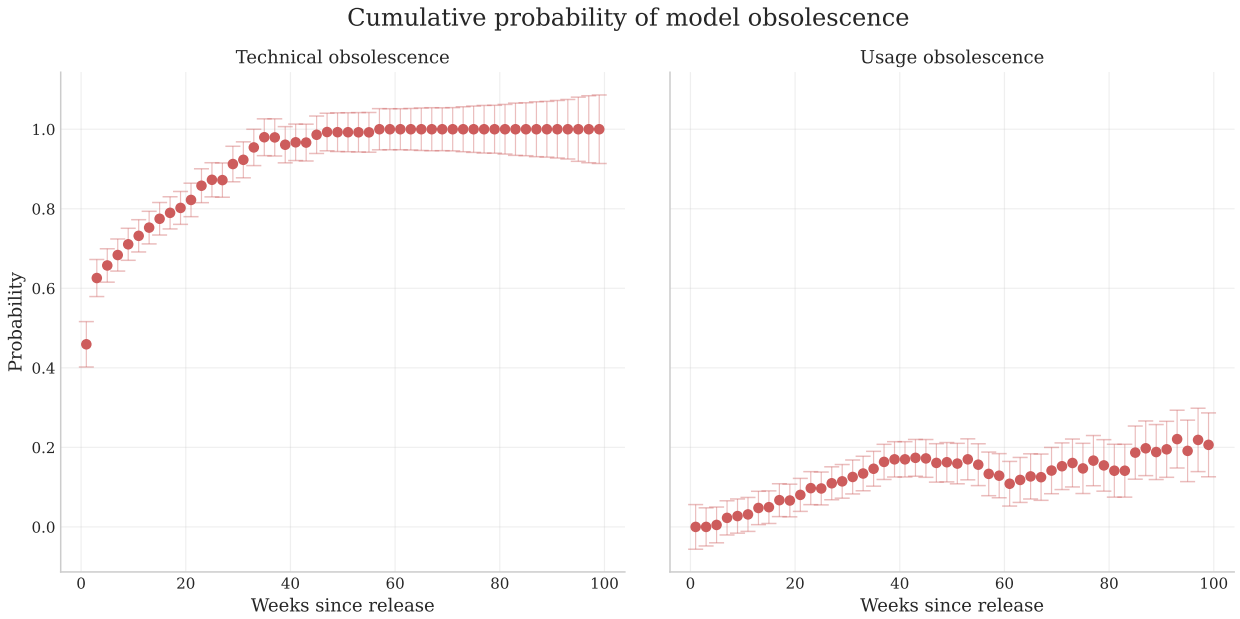
D.1 OLS Estimates of Cumulative Probability of Obsolescence

We estimate a linear probability model in which the cumulative probability that a model has reached obsolescence depends on its age. Specifically, we implement the following specification separately for usage and technical obsolescence:

$$\text{obsolete}_{jt} = \beta_{t-t_0^j} + \varepsilon_{jt}, \tag{D.1}$$

where $\text{obsolete}_{jt} \in \{0, 1\}$ is an indicator equal to 1 if model j has reached obsolescence by calendar week t and 0 otherwise. t_0^j indexes the calendar week in which model j was released. The $\beta_{t-t_0^j}$ terms are model-age dummies for each observed value of $t - t_0^j$. The regression is estimated by OLS with no intercept, so each estimated coefficient $\widehat{\beta}_{t-t_0^j}$ is the sample mean of $\texttt{obsolete}_{jt}$ at age $t - t_0^j$. Figure D.2 plots the estimated

Figure D.2: Share of models reaching usage obsolescence by weeks since release.



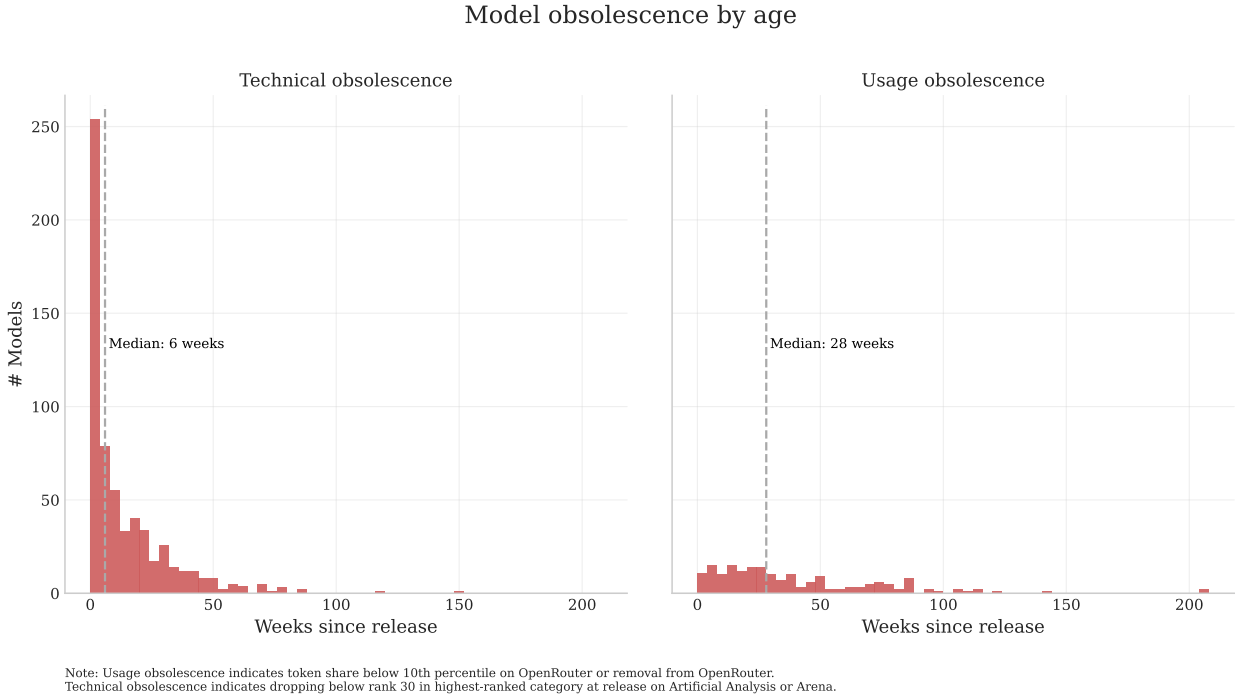
Note: Plots coefficients from the following ordinary least squares regression:
Outcome: Binary indicator for whether a model is obsolete in a given week.
Parameter of interest: Coefficient on week since release dummy (week 52 excluded).
Plot shows coefficients on every other week.

coefficients $\hat{\beta}_0, \ldots, \hat{\beta}_{99}$; we do not observe enough technical-obsolescence events to estimate probabilities beyond 100 weeks of age. The probability of obsolescence increases with age. Usage obsolescence peaks around 20%, while technical obsolescence is practically guaranteed by one year after release. If we observed all models over their full lifecycle, these probabilities would increase monotonically. In practice, however, some models were only recently added to OpenRouter, Artificial Analysis, or Arena, so we do not observe their eventual obsolescence. As a result, we observe fewer models at 60 weeks since release than at 40 weeks since release, for example.

D.2 Results for all models

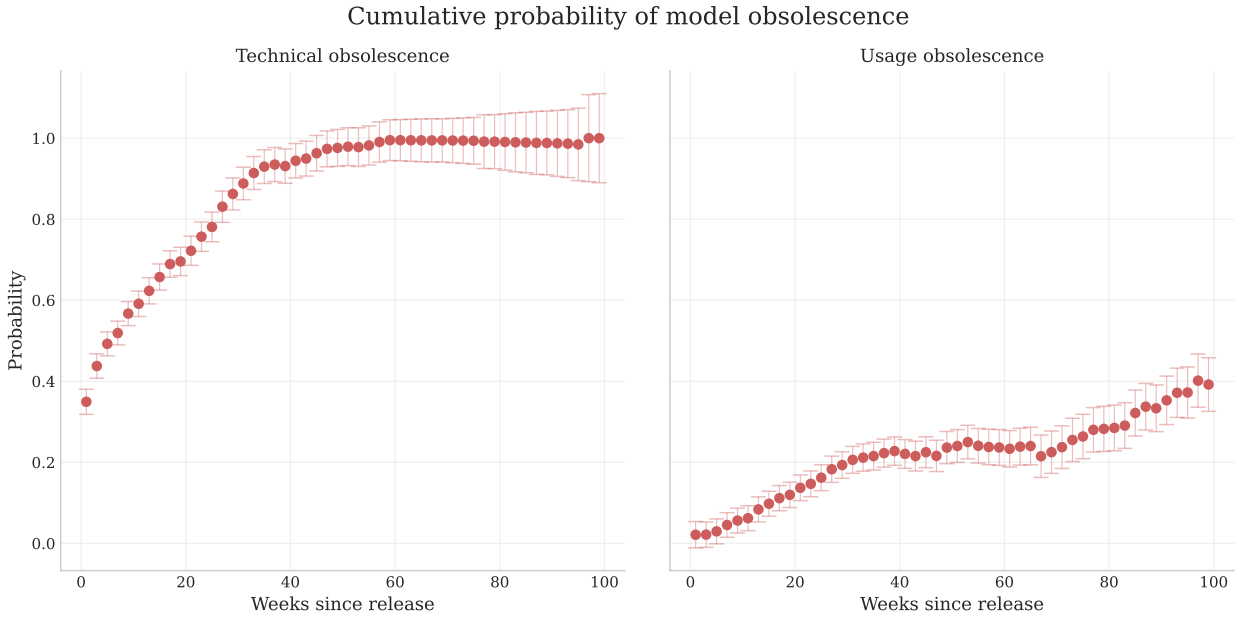
Here we present results from the analysis using all models on OpenRouter, Artificial Analysis, and Arena.ai, in contrast to the main body which shows results restricted to our primary sample of economically relevant models on Hugging Face. The starkly different patterns for age at obsolescence persist.

Figure D.3: Model age at obsolescence



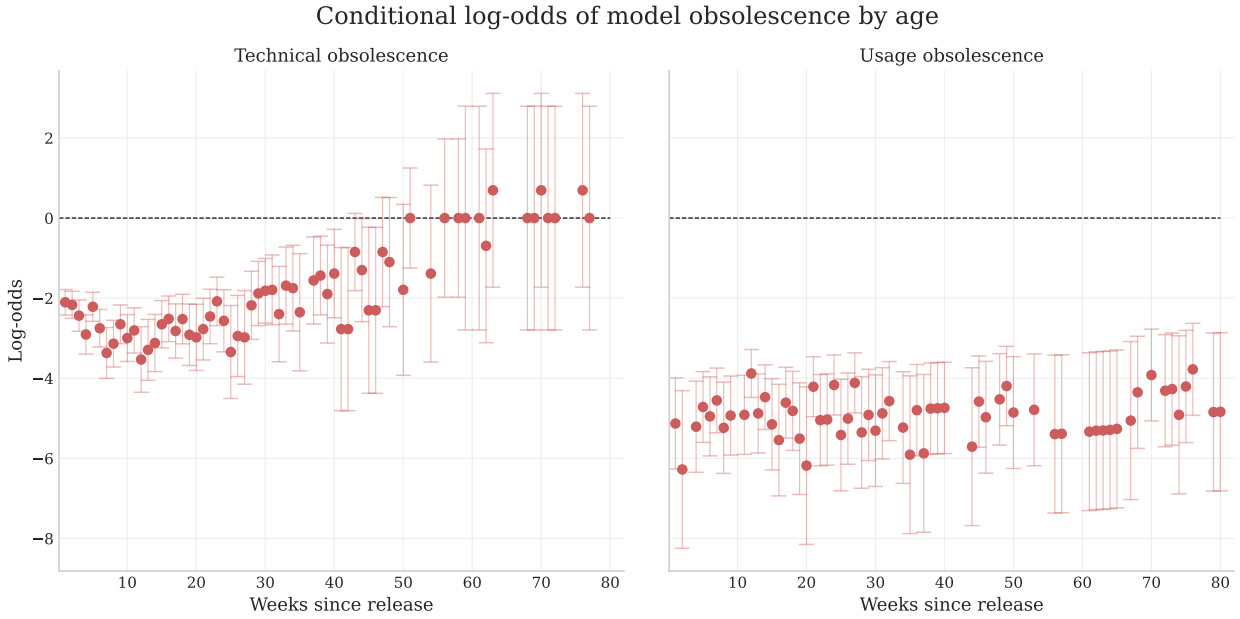
Note: Usage obsolescence indicates token share below 10th percentile on OpenRouter or removal from OpenRouter.
Technical obsolescence indicates dropping below rank 30 in highest-ranked category at release on Artificial Analysis or Arena.

Figure D.4: Share of models reaching usage obsolescence by weeks since release.



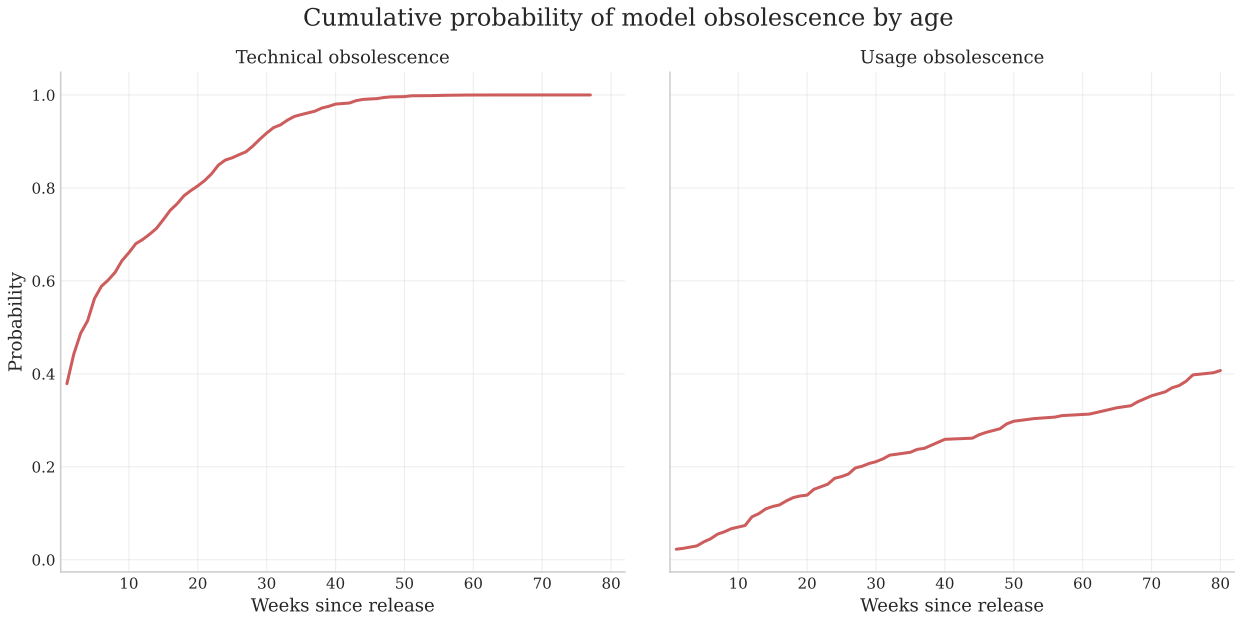
Note: Plots coefficients from the following ordinary least squares regression:
Outcome: Binary indicator for whether a model is obsolete in a given week.
Parameter of interest: Coefficient on week since release dummy (week 52 excluded).
Plot shows coefficients on every other week.
Includes models not matched to Hugging Face.

Figure D.5: Log-odds of obsolescence by age.



Note: Plots estimated log-odds of obsolescence from the following logit model:
Outcome: Binary indicator for a model becoming obsolete in a given week.
Sample: Model-weeks up to and including the obsolescence week.
Parameter of interest: coefficient on week since release.
Log-odds of zero implies 50% probability of obsolescence.
Plot shows coefficients on every other week, excluding weeks with no variation in obsolescence.
Includes models not matched to Hugging Face.

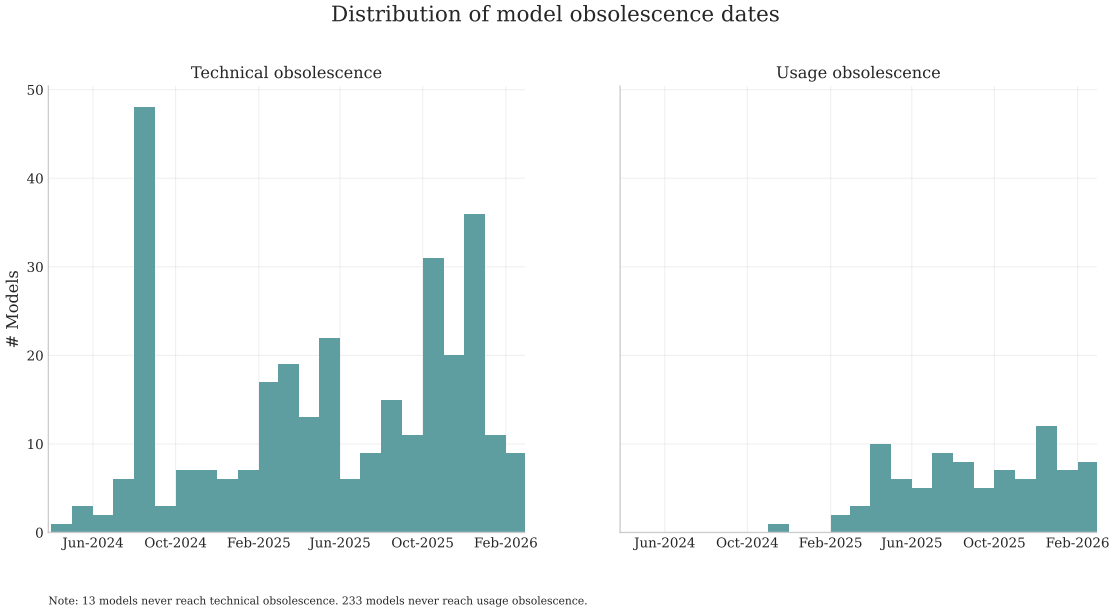
Figure D.6: Cumulative probability of usage obsolescence by weeks since release.



Note: Cumulative probability implied by estimated conditional probabilities.
Includes models not matched to Hugging Face.

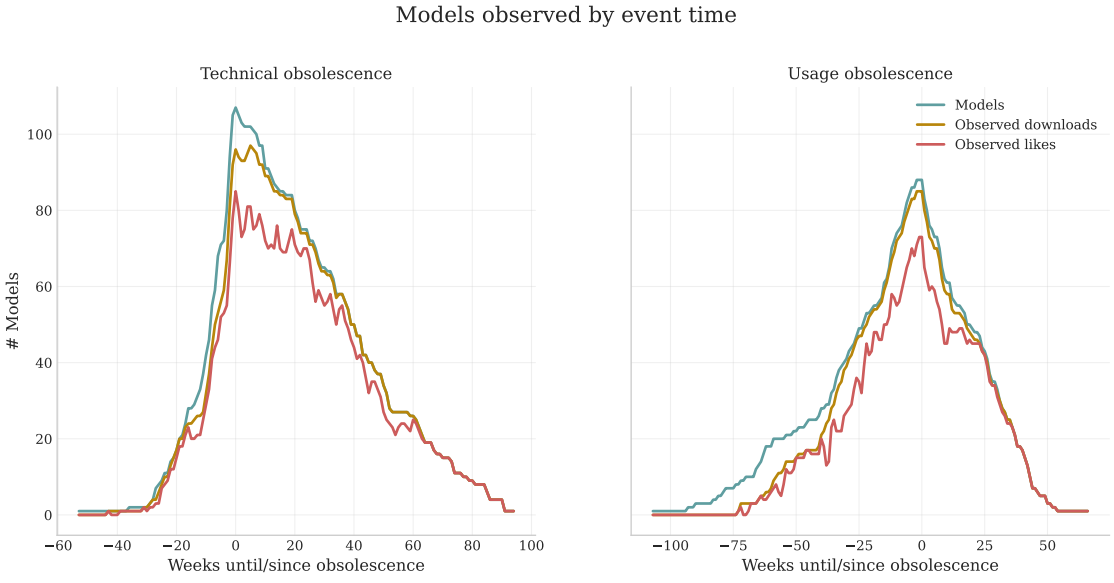
D.3 Obsolescence Event Study Motivating Figures

Figure D.7: Obsolescence timing



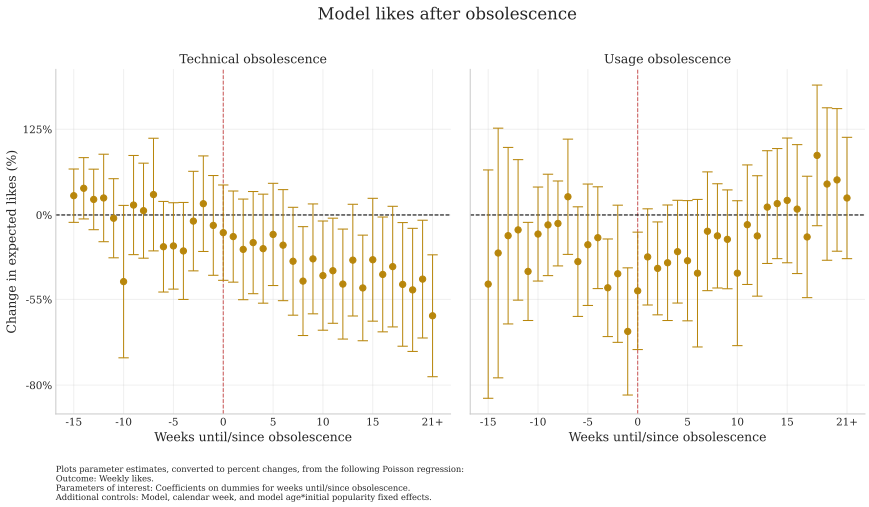
Note: 13 models never reach technical obsolescence. 233 models never reach usage obsolescence.

Figure D.8: Obsolescence event study data



D.4 Obsolescence Event Study Results: Attention

Figure D.9: Obsolescence Event Study Results: Attention



D.5 Obsolescence Event Study Results: Normalized

Figure D.10: Obsolescence Event Study Results: Demand

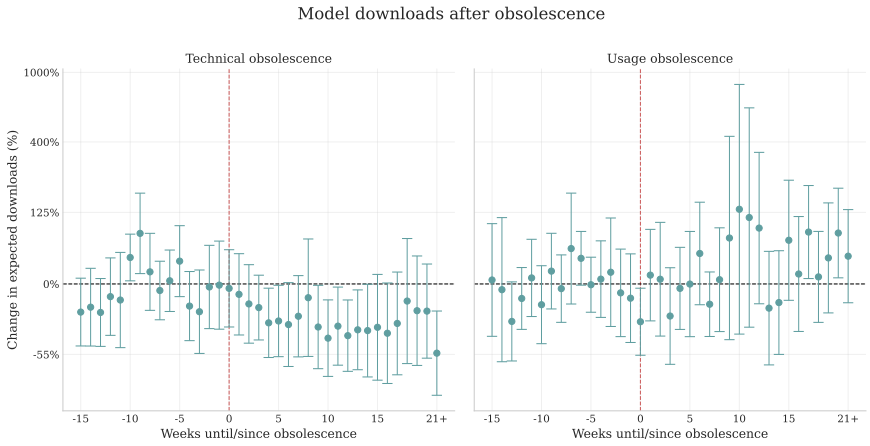
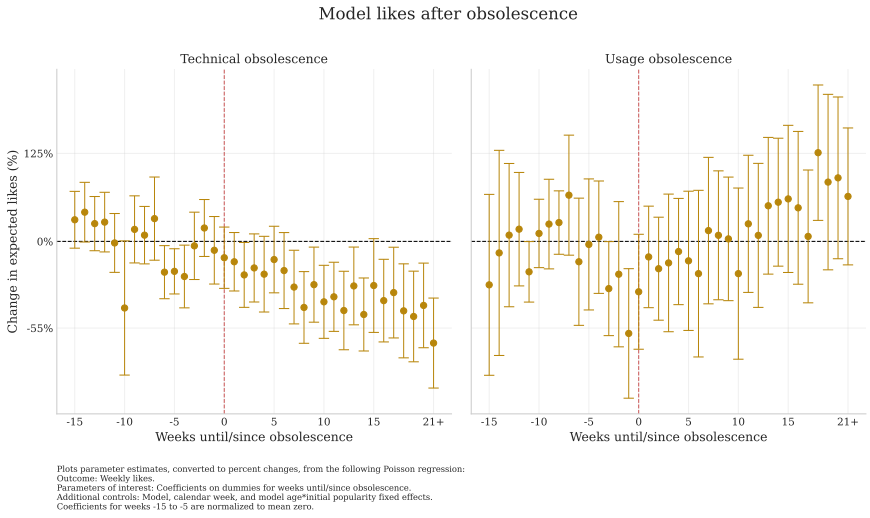


Figure D.11: Obsolescence Event Study Results: Attention



D.6 Obsolescence Event Study Results: Estimated Coefficients

Table D.3: Coefficient estimates from obsolescence event studies

	Downloads		Likes	
Weeks until/since obsolescence	Technical	Usage	Technical	Usage
-15	-0.460***	-0.338	0.182	-0.654
	(0.164)	(0.334)	(0.129)	(0.551)
-14	-0.404**	-0.448	0.252*	-0.360
	(0.189)	(0.405)	(0.148)	(0.603)
-13	-0.464**	-0.808**	0.147	-0.196
	(0.203)	(0.319)	(0.146)	(0.426)
-12	-0.285	-0.548*	0.161	-0.141
	(0.267)	(0.288)	(0.211)	(0.339)
-11	-0.323	-0.315	-0.031	-0.534**
	(0.306)	(0.288)	(0.191)	(0.237)
-10	0.159	-0.619**	-0.630*	-0.181
	(0.295)	(0.300)	(0.368)	(0.227)
-9	0.432	-0.238	0.094	-0.094
	(0.391)	(0.267)	(0.239)	(0.245)
-8	-0.004	-0.436*	0.041	-0.080
	(0.382)	(0.238)	(0.229)	(0.205)
-7	-0.215	0.018	0.193	0.172
	(0.348)	(0.391)	(0.272)	(0.278)
-6	-0.105	-0.093	-0.301	-0.441*
	(0.345)	(0.230)	(0.219)	(0.265)
-5	0.117	-0.392*	-0.293	-0.282
	(0.338)	(0.201)	(0.208)	(0.293)
-4	-0.392*	-0.329	-0.340	-0.215
	(0.220)	(0.254)	(0.234)	(0.246)
-3	-0.456*	-0.251	-0.058	-0.689***
	(0.274)	(0.318)	(0.240)	(0.236)
-2	-0.175	-0.484*	0.107	-0.556*
	(0.252)	(0.267)	(0.231)	(0.331)
-1	-0.154	-0.545**	-0.099	-1.102***
	(0.269)	(0.265)	(0.241)	(0.307)
0	-0.191	-0.812***	-0.167	-0.718**
	(0.240)	(0.218)	(0.230)	(0.283)
1	-0.259	-0.283	-0.205	-0.397*
	(0.251)	(0.298)	(0.220)	(0.232)
2	-0.367	-0.330	-0.326	-0.505**
	(0.245)	(0.324)	(0.243)	(0.224)
3	-0.409*	-0.747**	-0.262	-0.452
	(0.223)	(0.295)	(0.247)	(0.279)
4	-0.582**	-0.434*	-0.319	-0.347
	(0.233)	(0.263)	(0.263)	(0.248)
5	-0.561**	-0.385	-0.184	-0.433
	(0.230)	(0.339)	(0.247)	(0.290)
6	-0.601**	-0.038	-0.286	-0.551
	(0.251)	(0.371)	(0.268)	(0.356)
7	-0.507**	-0.613**	-0.438*	-0.154
	(0.251)	(0.291)	(0.260)	(0.287)
8	-0.297	-0.335	-0.626**	-0.198
	(0.385)	(0.339)	(0.263)	(0.252)
9	-0.629**	0.137	-0.415	-0.231
	(0.256)	(0.692)	(0.265)	(0.239)
10	-0.755***	0.463	-0.574**	-0.551
	(0.251)	(0.832)	(0.264)	(0.350)
11	-0.619**	0.369	-0.528**	-0.092
	(0.254)	(0.742)	(0.254)	(0.288)
12	-0.727***	0.250	-0.654**	-0.198
	(0.233)	(0.507)	(0.265)	(0.291)
13	-0.661**	-0.657*	-0.427	0.074
	(0.261)	(0.385)	(0.270)	(0.272)
14	-0.669**	-0.594	-0.690***	0.107
	(0.273)	(0.371)	(0.255)	(0.266)
15	-0.632**	0.112	-0.424	0.138
	(0.300)	(0.385)	(0.296)	(0.301)
16	-0.699**	-0.271	-0.563**	0.055
	(0.310)	(0.370)	(0.277)	(0.311)
17	-0.588**	0.205	-0.489*	-0.208
	(0.300)	(0.327)	(0.291)	(0.293)
18	-0.335	-0.303	-0.657**	0.563*
	(0.371)	(0.351)	(0.298)	(0.339)
19	-0.444	-0.088	-0.709**	0.293
	(0.328)	(0.355)	(0.297)	(0.368)
20	-0.449	0.194	-0.607**	0.332
	(0.299)	(0.321)	(0.284)	(0.344)
21+	-0.925***	-0.069	-0.954***	0.161
	(0.270)	(0.330)	(0.294)	(0.293)

Standard errors in parentheses clustered at the model level.
*Significance codes: ***: 0.01, **: 0.05, *: 0.1.*