The Role of "Live" in Livestreaming Markets: Evidence Using Orthogonal Random Forest

Ziwei Cong, Jia Liu, Puneet Manchanda[1]

October 3, 2021

For the most recent version, please go to:
https://ziweicong.github.io/assets/pdf/jmp.pdf

[1]Ziwei Cong is a Ph.D. candidate in Marketing at Hong Kong University of Science and Technology. Email: zcongaa@connect.ust.hk. Jia Liu is Assistant Professor of Marketing, Hong Kong University of Science and Technology. Email: jialiu@ust.hk. Puneet Manchanda is Isadore and Leon Winkelman Professor, Stephen M. Ross School of Business, University of Michigan. Email: pmanchan@umich.edu. The paper is part of the first author's dissertation. The authors would like to thank ..... (please add whoever give you individual comments on the paper), and participants at the 2020 AIML Conference and the 2021 Marketing Science Conference for feeback.

Abstract

The common belief about the growing medium of livestreaming is that its value lies in its "live" component. We study this belief by quantifying how the response of aggregate demand to price changes before, on the day of, and after the livestream. We leverage our unique access to rich datasets from the largest livestreaming platform for knowledge goods in China that allows consumers to purchase the recorded version of the livestream. We apply our data in a generalized Orthogonal Random Forest algorithm that can estimate heterogeneous treatment effects in the presence of high-dimensional confounders whose relationships with the treatment policy (i.e., price) and outcome of interests (i.e., demand) are complex but partially known. We find significant temporal dynamics in the price elasticity of demand over the entire product life-cycle. Specifically, demand gradually becomes less price sensitive over time to the livestreaming day and is inelastic on the livestreaming day. Over the post-livestream period, demand is still sensitive to price, but much less than the pre-livestream period. We further provide evidence for the mechanisms driving our main results. We find that consumers value the opportunity of real-time interaction with content creators, the quality level and the quality uncertainty level of the content, when making purchase decisions.

Keywords: Livestreaming, Pricing, User Generated Content, Heterogeneous Treatment Effect, Random Forest, Double Machine Learning, Semi-parametric Deep Neural Networks

1 Introduction

Livestreaming is the practice of streaming online content for consumption in real time. China currently is the dominant market for livestreaming, at $16.3 billion (Statista, 2020), with the US the next biggest, estimated to be $11 billion (Thomas and Palmer, 2021). Compared with recorded content, the key benefits of livestreaming are its immediacy, authenticity, and the ability of the audience to interact with the creator or streamer during the stream (Hu et al., 2017; John, 2020; Zhang et al., 2020; Lin et al., 2021).[1] These attributes have made it being widely adopted in various domains in recent years, resulting in a strong growth. For instance, Twitch, the leading U.S. livestreaming platform for gaming, has seen over 300% monthly audience growth from 2015 to 2018 (Iqbal, 2021). In addition to e-commerce giants Amazon and Alibaba, many retailers and brands (e.g., Walmart, Macy's, Avon, LG, Nordstrom, and Petco) have also embraced livestreaming shopping events (Larson, 2021; Wheless, 2021). Therefore, it becomes more and more important for marketers to understand the economic impact of live content or events.

Given the novelty of livestreaming, there is, however, no academic work that compares consumer purchase behavior of live-streamed content versus recorded content. This paper aims to fill in this significant research gap using data from one of the largest livestreaming platforms in China, Zhihu Live. We leverage the fact that the platform also makes the recorded version of live content available for purchase at the same price, allowing us to contrast consumer response to the content before, on the day of, and after the livestream. Specifically, we examine demand and also estimate price elasticity of demand over the entire product life-cycle.

Zhihu Live targets creators who want to monetize their expertise or establish their own business via livestreaming. The creators on Zhihu Live can launch paid live events (denoted as "Live" in the rest of the paper) on specific topics, such as cooking, culture, business, etc. A Live is usually listed on the market for purchase 2-3 weeks prior to the scheduled live day. During the livestream,

[1] The "Creator Economy" is built around highly motivated, creative, and skilled individuals that have started their own brand, business, or community utilizing a digital platform to share their work (Bakhtiari, 2021; Henderson, 2021; The Economist, 2021). We will use the word "creator" and "streamer" interchangeably in this paper.

the creator gives a one to two hour live talk, interacting with the customers in a virtual chat room. After the livestream concludes, its recorded version is available for purchase on Zhihu at the same price, but without the opportunity of creator-audience real-time interaction. Zhihu Live follows the most common pricing model in livestreaming markets, pay-per-view model, that is also used by Facebook, Vimeo and OnlyFans. Each creator in Zhihu Live market sets a common price for both her/his live and recorded content with consumers paying this fixed price one-time.[2]

Estimating the price elasticity of demand using the rich observational data in this context - where the "product" is a custom, one-off experiential event - is challenging for two major reasons. First, prices and demand are likely to be confounded with a large number of factors (e.g., content quality and creator credibility) in potentially complex and partially known ways. This is, the number of covariates and potential variables formed by different ways of interacting and transforming the covariates is high-dimensional. This make the number of model parameters large relative to the sample size. Second, the computed price elasticity is likely to be heterogeneous, varying both temporally and cross-sectionally. In terms of our research question, the temporal variation (i.e., how it varies relative to the livestreaming day) is particularly important.

In order to address the above challenges, we adopt a framework that allows us to estimate the price elasticity of demand non-parametrically in the presence of high-dimensional confounders whose relationships with price are complex but partially known. Specifically, our framework generalizes Orthogonal Random Forest (ORF) (Oprescu et al. (2018)) via the use of Semi-parametric Deep Neural Networks (SDNNs) to estimate the nuisance functions (i.e., functions of all confounding variables). SDNNs are denoted as semi-parametric as they allow partitioning of the confounders into two sets. The first set consists of confounders that are assumed to have a known relationship with the key treatment and outcome variable (price and demand in our case) while the second set consists of confounders that are assumed to have an (unknown) relationship with the treatment and outcome. The use of SDNNs to estimate the nuisance functions is particularly

[2]In livestreaming, pay-per-view is considered to be a "direct" pricing model mechanism, along with subscriptions, and pay-what-you-want (PWYW) or tipping (Lu et al., 2021). For more on the PWYW model of pricing in livestreaming, we refer the reader to Lu et al. (2021) and Lin et al. (2021). In contrast, some platforms follow an "indirect" pricing model, revenue is generated via ad and/or product placement during the livestream.

suited to marketing and economics settings, e.g., when panel data are used. This is because the method allows for both "fixed effects" while accommodating flexible non-linear combinations of the remaining confounders.

We find that live (recorded) content generated about 70% (30%) of sales in this market. Our estimation results show considerable dynamics in price sensitivity relative to the livestreaming day. Specifically, demand gradually becomes less price sensitive approaching the livestreaming day and is inelastic on the livestreaming day. Throughout post-stream period, demand becomes price sensitive again, but on average is less sensitive than in the pre-livestream period. The pricing literature often assumes that high price sensitivity co-occurs with low utility (i.e., low valuation or low willingness to pay) and low price sensitivity co-occurs with high utility (i.e., high valuation or high willingness to pay). Therefore, our finding that demand is inelastic only on the livestreaming day confirms that consumers in general value live format. In addition, there should be factors (other than "live") that make post-livestream demand for recorded version less price sensitive than pre-livestream demand. We have also conducted a few robustness checks, showing that our main findings are not driven by consumer self-selection issues.

We then explore the possible mechanisms underlying our research findings and provide a series of empirical evidence. First, we show that the inelastic demand on the livestreaming day can be attributed to the availability of immediate real-time interaction with creators. Second, we find that demand for recorded version is less price-sensitive when its consumer rating is higher. This suggests that quality concern is another major driver of the temporal dynamics. We further show confirming evidence that consumers are more likely to rely on information cues (such as Live popularity, own prior knowledge, creator online reputation, and Live topic category) to resolve their quality uncertainty when making purchase for a live version rather than for a recorded version. Our findings suggest that quality uncertainty reduction may explain the reduced price sensitivity of post-livestream demand relatively to pre-livestream demand. That is because consumers are much better at assessing the quality of the recorded version when consumer ratings become available, leading to a lower price sensitivity over post-livestream period.

Our paper makes several contributions. First, it is one of the first studies investigating the growing phenomenon of livestreaming, focusing on how price responsiveness of demand changes with temporal distance to the livestreaming day. Our study provides timely and relevant implications for livestreaming pay-per-view service, which is gaining popularity across many platforms especially during the pandemic. Second, our findings that post-livestream demand accounts for about 30% of market sales and exhibits relatively less sensitive price elasticity are contrary to the conventional belief that livestreaming content has no much residual value once the event is over (Vosgerau et al., 2006; Barber, 2017). Vosgerau et al. (2006) show that consumers prefer watching live broadcasts, even when tape-delayed broadcasts provide the same content. That is because events (such as soccer match) lose indeterminacy and hence excitement, if the broadcast is not live. Our paper complements this literature by showing that when content is learning-oriented, recorded version has value because of its reduced quality uncertainty. Therefore, our research findings suggest that it is beneficial for content creators and platforms to provide both live and recorded content, and there is opportunity for dynamic pricing and promotions. In doing so, efforts to reduce quality uncertainty is another key consideration. Finally, our methodological approach extends ORF by introducing SDNNs for nuisance estimation. Our extension makes ORF more suitable for panel data, which is common in economics and marketing problems. We believe that this framework is applicable in other real-world settings that involve highly-differentiated goods and high-dimensional covariates, such as peer-to-peer markets and gig-economy platforms.

The rest of paper is organized as follows. We first describes our empirical context and data, followed by descriptive analysis. Next, we introduce the model setup, ORF algorithm, and how we allow SDNNs for nuisance estimation. Then, we report the main estimation results based on our proposed method, and we explore their underlying mechanisms. Finally, we conclude with a discussion.

2 Research Context and Data

2.1 Zhihu Live

Zhihu is the earliest and largest knowledge sharing platform in China, starting as a Q&A community (similar to Quora.com) in 2011.[3] By December 2020, Zhihu had 43.1 million cumulative content creators, who had contributed 315.3 million questions and answers (Hu et al., 2017). The company went public in the U.S. in March 2021. Each user on Zhihu has a profile page, which contains some personal information provided by the user and a summary of the user's past contribution activities along with all the contributed content. Please refer to Figures A1 and A2 in Appendix A for illustrative examples.

Zhihu launched Zhihu Live on May 14, 2016, targeting creators who want to monetize their expertise via livestreaming content to a paying audience. By December 2020, Zhihu Live had nearly 10,000 cumulative paid Lives, with 5,000 creators, 6 million paying users, resulting in a revenue of approximately $12 million in 2020 (Fergus, 2020; CIW Team, 2021). To become a Live creator, a user needs to provide Zhihu with basic personal information, such as real name, educational background, and the proof of expertise in certain area(s). Upon approval, the user is allowed to open his/her own Lives. During our study period, the Live revenue is split between the creator and the platform 70/30 (%).

Figure 1 illustrates the life-cycle of a typical Live on Zhihu. A creator first decides the key product features, including full price (under loose guidance from the platform), topic, content, starting time, and seat limit (the maximum number of viewers who are allowed to interact with the creators during the livestream). Once Zhihu approves of the event, the Live is listed on the platform for purchase, typically 2-3 weeks prior to the scheduled starting time. Consumers become aware of upcoming Lives through the Live market webpage. Note that once a Live enters the market,

[3] Users on Zhihu can voluntarily seek and share information (such as knowledge, expertise, and customized solutions) in a variety of domains mainly through posting questions and answers. A typical question consists of a title and several topic tags. A question usually receives multiple answers, which by default are ranked by the total number of votes each answer has received. In addition to content-related activities, users can organically follow any other users.

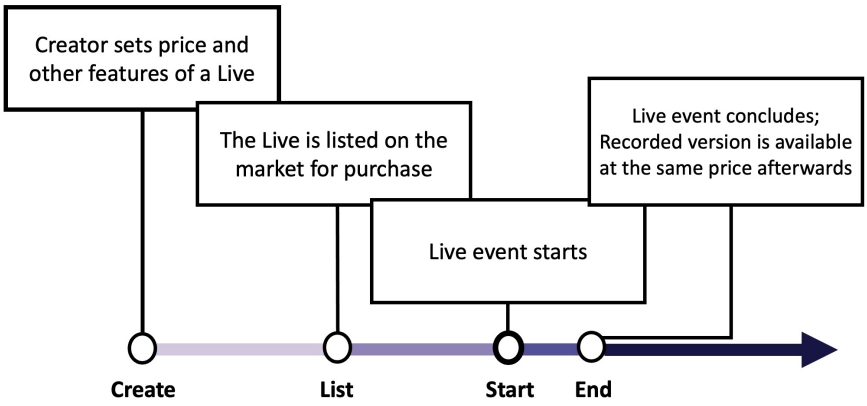


Figure 1: Life-Cyle of a Live

the creator can no longer change its full price, which means that strategic price adjustment by the creator over time in reaction to realized demand conditions is not allowed.[4] The seat limit of a Live controls for the number of viewers who are allowed to raise questions and leave comments during the livestream. So consumers who purchase the Live after all seats are filled up can only view the event in real-time. The number of cumulative sales is explicitly displayed on the page for each Live, but the number of remaining seats is not. Consumers will be informed seat availability when clicking into the "Purchase This Event" button. Because seat limit is set to be 500 by most creators, consumers can infer the number of remaining seats roughly.

During the livestream, the creator will give a 1-2 hour live talk on the prepared content. The creator can interact with his/her viewers via text, voice, or picture messages in a virtual chat room. An illustrative example is provided in Figure A3 of Appendix A. After the livestream concludes, consumers are allowed to provide feedback, i.e., a rating on a 5-point scale along with text comments. The recording of the livestream is then made available for purchase at the same price, but without the opportunity of creator-consumer real-time interaction. Consumers can explore all recorded Lives via a subpage of the Live market website. The fact that the platform also makes live content available in recorded form at the same price allows us to compare consumer purchase

[4]In a few cases, we observe that the platform lowered the price by running platform-level promotions. These were often run over weekends and public holidays. We discuss this in detail in Section 2.2. Except this type of price variations, Zhihu seldom use other (non-price) marketing mix elements to market Live events, except for the use of in-site banner advertising for Lives opened by celebrities. But we control for this type of adverting in our models.

behavior before, on the day of, and after the livestream.

2.2 Data

Zhihu provided us two main data sets for our study purpose. The first comprises transaction-level records on the platform from May 2016 to June 2017, involving 1,519 creators and 3,491 unique Lives. Each transaction record includes buyer ID, purchase time, unit price paid, and rich characteristics of the purchased Live. The second data set contains complete and detailed historical activities of all individual users in the free Q&A community, i.e., the main Zhihu platform. We observe user profile, registration time, content contribution, content subscription, and social activities. We also know different types of individual feedback that a user received for each answer he/she has contributed. These observed creator characteristics are related to creators' reputation in the free Q&A community. This reputation is expected to be an influential factor for creators to set price of their Lives and also for the demand of their Lives (Shapiro, 1982; Varian, 1999; Wang and Nicolau, 2017).

Our empirical study focuses on 2,705 Lives that were launched from November 2016 to June 2017. During this period, the Zhihu Live market was relatively stable with no major changes in the operating policy. When estimating price elasticity of demand over Live life-cycle, we aggregate individual transactions to obtain the daily unit sales of each Live. As about 90% of total sales transactions by volume in the Zhihu Live market occur from the day of Live listing to 30 days post Live airing, we restrict our analysis to this window. This results in a total of 125,028 Live-day observations.

The internal data provided by Zhihu allows us to observe hundreds of characteristics of all parties (Lives, creators, consumers, and platform) that may affect price decision and daily demand. We organize these covariates into five categories and summarize them in Table 1. The first category is Live characteristics. Examples of fixed variables are Live topic category, seat limit, when the event was listed and aired. Examples of time-varying variables are cumulative demand and advertising/promotion by the platform. These variables are expected to account for a substantial

Table 1: Full List of Control Variables

Live-Specific Characteristics

- Fixed variables: The num. of days from listing to livestream; Seat limit; Rating; Whether the Live is advertised by the platform; Whether livestream is held on public holiday; *Topic indicators*; *livestream week indicators*; *livestream day-of-week indicators*
- Time-varying variables: The cum. sales of a Live; Whether seats are still available; Percentage discount; *Temporal distance relative to livestream*

Creator-Specific Characteristics

- Social network: The cum. num. of followers/followees
- Content contribution in the Q&A community: The cum. num. of answers/questions/articles; The cum. num. of up-votes/down-votes/thank/unhelpful that own posted answers received
- Past Live experience: The cum. num. of hosted Lives; The average sales of past Lives; The average rating of past Lives; The average num. of reviews of past Lives; The average num. of "likes" of past Lives; The num. of days since the first Live
- Profile: The num. of days since registered Zhihu; Whether is a off-line celebrity; Whether is invited to open Live; Whether is an "Excellent Contributor" certified by Zhihu

Market-Level Factors

- Market size/growth: The num. of days since launching Zhihu Live; The cum. num. of (unique) consumers/creators/Lives on the market
- Competition: The cum. num. of available Lives from the same topic; The num. of running Lives from the same topic on the same day; The average price of available Lives from the same topic; The average price of running Lives from the same topic on the same day; The average follower size of other Live' creators from the same topic; The average follower size of other creators who running Lives in the same topic on the same day
- Policy change: Whether Live rating function was launched (November 02, 2016); Whether detailed reviews were displayed to public (April 24, 2017)

Platform-Level Factors

- User base: The cum. num. of users; The daily num. of newly registered users
- Voluntary content contribution: The cum. num. of questions/answers; The daily number of newly posted questions/answers

Transaction-Day Specific Factors

- Dummy indicators for day-distance relative to live streaming
- Seasonality: *Transaction week indicators*; *Transaction day-of-week indicators*

proportion of demand signals used for pricing decisions (Gibbs et al., 2018). The second category is creator characteristics. Examples are creator's degree centrality in the social network on Zhihu, creator past content contribution related activities on both the Q&A platform and the Live market.

These variables can capture very well a creator's reputation, expertise, and experience that are expected to be driving pricing decisions and influencing demand (Shapiro, 1982; Varian, 1999; Wang and Nicolau, 2017).

The third category captures the condition of the Live market, including the growth of the entire Live market, category-level competition, and few changes in market design. These are likely to influence daily demand and creators' pricing decisions. The fourth category captures the conditions of the Q&A platform, including the total/new user base and the total/new volume of contributed questions and answers. These may potentially influence the growth of the Live market and hence the potential demand of individual Lives. The last category captures seasonal and temporal shocks on each transaction day. We include dummy indicators for calendar week, day of the week, and the temporal distance to the livestreaming day which can capture systematic variations in demand within a Live life-cycle.

3 Descriptive Analysis

To illustrate our data patterns, this section first presents some summary statistics of creators and Lives, following by basic regression analysis. Due to the endogeneity concern, our research findings in this section are best interpreted as a precursor to the main results based on ORF algorithm.

3.1 Summary Statistics

Table 2 reports some summary statistics of the 1,330 unique creators observed in our data. The large standard deviations of these variables suggest that activity levels in the Q&A community vary widely across creators.[5] Table 3 presents the summary statistics of all the Lives hosted by these creators over our study period. An average Live is priced at 22 RMB (about $ 3), allows 498 seats, and is listed on the market for sales 17 days prior to the livestream. While the standard deviation of "Seat Limit" is large, we find it is set to 500 for most Lives. Hence, there is very limited

[5]Because 619 creators held more than one Live over our study period, we obtain the characteristics for each creator at the time of each of his/her Live, and then compute the average across Lives hosted by the same creator.

Table 2: Creator Summary Statistics

				Correlation													
	Variable	Mean	Std.	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14
V1	Num. Lives	2	3	1													
V2	Live Market Lifetime (days)	35	62	*0.637*	1												
V3	Q&A Platform Lifetime (days)	976	671	*0.144*	*0.231*	1											
V4	Num. Followers	18,513	64,302	*0.255*	*0.353*	*0.281*	1										
V5	Num. Followees	140	345	*0.090*	*0.121*	*0.282*	*0.136*	1									
V6	Num. Articles	19	58	*0.331*	*0.234*	*0.195*	*0.382*	*0.171*	1								
V7	Num. Questions	9	44	*0.100*	0.037	*0.215*	*0.256*	*0.191*	*0.410*	1							
V8	Num. Answers	171	392	*0.180*	*0.177*	*0.307*	*0.363*	*0.320*	*0.349*	*0.366*	1						
V9	Num. Upvotes per Ans.	148	476	*0.088*	*0.159*	0.042	*0.226*	*0.046*	0.029	-0.001	-0.013	1					
V10	Num. Downvotes per Ans.	4	11	*0.120*	*0.166*	*0.122*	*0.241*	*0.081*	*0.078*	*0.055*	*0.048*	*0.625*	1				
V11	Num. Thanks per Ans.	45	175	*0.087*	*0.155*	*0.052*	*0.171*	0.031	0.030	-0.005	-0.023	*0.901*	*0.598*	1			
V12	Num. Unhelpfuls per Ans.	5	16	*0.126*	*0.203*	*0.118*	*0.251*	*0.061*	*0.056*	0.025	0.015	*0.835*	*0.820*	*0.882*	1		
V13	Excellent Contributor	0.147	0.354	*0.322*	*0.476*	*0.272*	*0.279*	*0.149*	*0.135*	*0.109*	*0.234*	*0.083*	*0.080*	*0.083*	*0.143*	1	
V14	Offline celebrity	0.028	0.165	0.011	-0.021	*-0.185*	*0.066*	*-0.064*	-0.014	-0.029	*-0.059*	*0.106*	-0.004	0.024	0.004	-0.044	1

Note: The last two rows are dummy indicators for whether a creator is recognized as "excellent contributor" by the platform, and for whether being a offline celebrity (which are identified manually based on our own research). The correlations that are in italic format are statistically significant at $p < 0.1$.

variation in seat limit. In contrast, prices vary widely across Lives, ranging from 0.99 RMB to 399 RMB, as shown in Figure 2. We find that there exists some within-Live price variation across transactions for about 10% of Lives due to platform promotions. But, virtually promotions are applied over the entire Live life-cycle, making them equivalent to a permanent price cut. Therefore, when estimating price elasticity of demand, our identification primarily leverages across-Live price variation.

Table 3: Live Summary Statistics

				Correlation					
	Variable	Mean	Std.	V1	V2	V3	V4	V5	V6
V1	Num. Days before Livestream	17	13	1					
V2	Seat Limit	498	139	*0.074*	1				
V3	Price	22	22	*0.123*	-0.025	1			
V4	Total Sales	639	2,311	0.004	-0.021	0.018	1		
V5	Pre-livestream Sales	499	2,084	0.007	-0.018	*0.035*	*0.990*	1	
V6	30-day Post-livestream Sales	75	195	-0.015	-0.017	*-0.012*	*0.557*	*0.448*	1

Note: The correlations in italics are statistically significant at $p < 0.1$.

The total sales of an average Live by the end of our study period is 639, with 499 sales over the pre-livestream period and 75 sales within the first 30-day of the post-livestream period. Overall, we find that 72% of sales transactions occurred prior to the airing of the Live. Thus, a surprisingly high 28% of sales is realized after the livestream. Figure 3 reports the average daily sales across all Lives over Live life-cycle, along with the 95% confidence intervals. We can see that it gradually increases throughout the pre-livestream period, peaks on the livestreaming day (with about 60 tickets sold per day), and drops immediately afterwards. This pattern confirms significant dynamics in demand over Live life-cycle in this market.

Recall that when the cumulative sales of a Live reach the seat limit, buyers will no longer have the opportunity to interact with the creator during the livestream. A buyer is informed about whether a seat is available during her purchase process. If consumers value live interaction with content creators, we expect that demands will drop significantly whether seats are no longer available. In our data, there are 583 Lives whose seats were filled up before the livestreaming day.

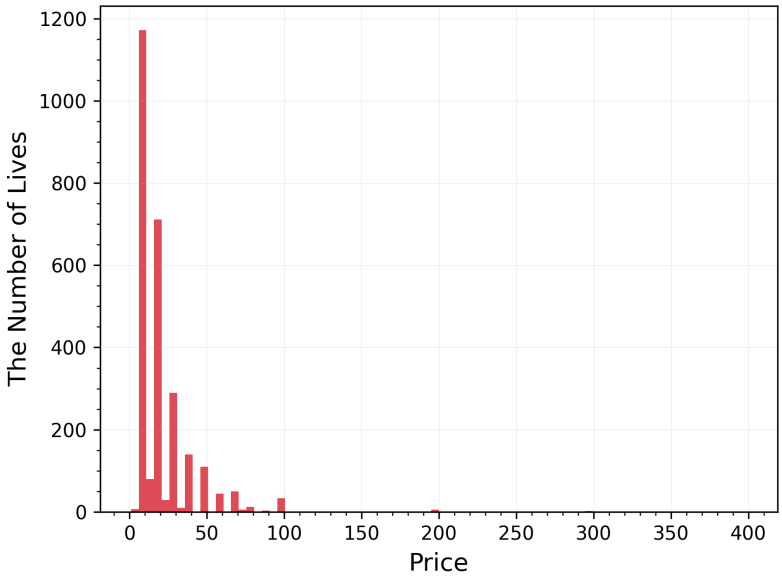


Figure 2: Distribution of Price

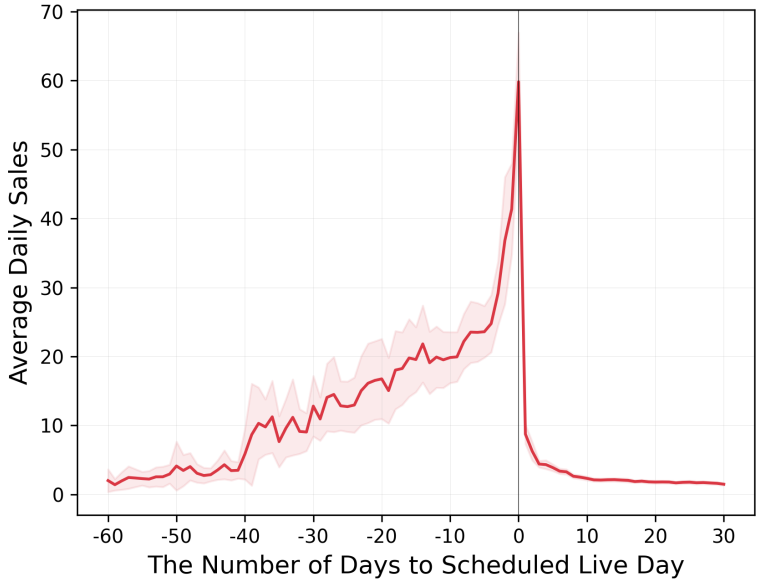


Figure 3: Average Daily Sales over Time to Scheduled Live Day

Figure 4 shows the average hourly sales of these Lives from 5-hour before to 5-hour after seats were filled up, along with the 95% confidence intervals. One can see that the positive time trend of

hourly sales becomes significantly negative immediately after the seats are filled up. This confirms our hypothesis that the opportunity of interacting with creator during livestream is important to consumers. We will provide more details on how this also influences price elasticity of demand in later section on mechanisms.

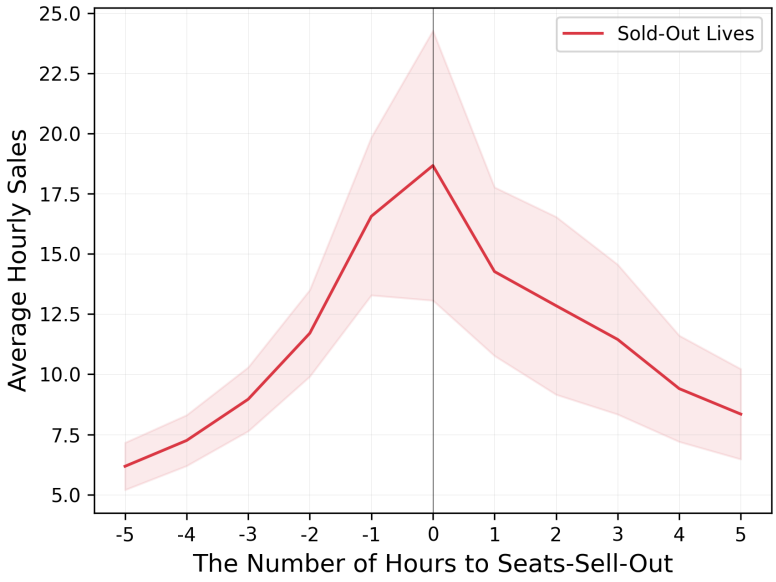


Figure 4: Average Hourly Sales over Time to Seats-Sell-Out

Note: This figure is based on 538 Lives whose seats are sold out before the scheduled live day. The shaded region depicts 95% confidence interval across all selected Lives at each temporal hour distance.

3.2 Price Regression

We adopt a cross-sectional regression analysis to understand the factors that may influence the prices set by creators. Specifically, we regress the log-transformed price p_{ij} of Live j hosted by creator i on a subset of the covariates listed in Table 1. All the continuous independent variables are standardized and hence their coefficients are comparable. The estimation results are presented in Table 4, where the full model is built up gradually. In the full model (Model (4)), the estimated coefficients for Live-specific characteristics suggest that creators tend to set a lower price if the Live is to be aired on holidays ($p < 0.1$) or has fewer days on the market prior to airing ($p < 0.01$).

Table 4: Regression of Log Seller's Full Price

Model	(1)	(2)	(3)	(4)
Live-Specific Features and Seasonality on the Livestreaming Day				
Seat Limit	−0.004 (0.011)	−0.002 (0.011)	−0.001 (0.011)	−0.001 (0.011)
Days before Livestream	0.096 (0.011)***	0.099 (0.011)***	0.065 (0.011)***	0.070 (0.011)***
Advertised		−0.087 (0.041)**	−0.066 (0.040)	−0.128 (0.042)***
Topic Fixed Effects		Yes	Yes	Yes
Holiday	−0.130 (0.061)**	−0.130 (0.061)**	−0.134 (0.059)**	−0.121 (0.058)**
Live Day of Week Fixed Effects	Yes	Yes	Yes	Yes
Live Week Fixed Effects	Yes	Yes	Yes	Yes
Creator Past Experience and Reputation in Live Market				
Live Market Lifetime			−0.049 (0.020)**	−0.033 (0.021)
Invited Creator			0.112 (0.050)**	0.134 (0.050)***
Num. Past Lives			0.131 (0.017)***	0.075 (0.018)***
Average Past Pre-Livestream Demand			−0.008 (0.011)	−0.019 (0.011)*
Average Past Live Rating			0.034 (0.014)**	0.046 (0.014)***
Creator Experience and Reputation in Q&A Community				
Q&A Platform Lifetime				−0.015 (0.012)
Offline Celebrity				0.379 (0.077)***
Excellent Contributor				0.023 (0.033)
Num. Followers				0.036 (0.016)**
Num. Followees				−0.016 (0.012)
Num. Articles				0.061 (0.016)***
Num. Questions				0.039 (0.017)**
Num. Answers				−0.029 (0.015)*
Num. Upvotes per Ans.				−0.063 (0.031)**
Num. Downvotes per Ans.				0.028 (0.023)
Thank Num. per Ans.				0.059 (0.038)
Num. Unhelpfuls per Ans.				−0.028 (0.044)
Num. Obs.	2,705	2,705	2,705	2,705
R^2	0.085	0.122	0.164	0.194

Note: Creator characteristics on the Live market and the Q&A community is computed by the starting time of each Live held by the creator (except for *InvitedCreator*, *Celebrity* and *ExcellentContributor*, which are fixed over time). $^{***}p < 0.01$, $^{**}p < 0.05$, $^{*}p < 0.1$.

Intuitively, we find that several factors that can indicate creators' levels of experience or reputation are positively associated with the price of the current Live ($p < 0.01$): the number of past Lives, the average rating past Lives, the number of contributed questions and answers, and the number of followers in the Q&A platform. Similarly, we find that offline celebrities or invited creators are more likely to set high price.

In summary, these findings confirm that creators consider timing, their experience and reputation in both the paid market and the free Q&A community when setting prices. This is further supported by the improvement in R^2 across different model specifications. We can see that Live specific features and seasonality together accounts for about 12.2% of the variation in prices (Model (2)), while creator-specific variables together account for another 7.2% of the variation (Model (4)). Note that the full model explains only 19.4% of the price variation. This is significantly lower than the existing studies, e.g., 80% for cruise industry data (Joo et al. (2020)) and 40-50% for Airbnb data (Li et al. (2016), Gibbs et al. (2018)). This further confirms that a large number of factors may influence price decision in a complex way in our study context. This thereby calls for the use of more flexible functional form to model these relationships, when addressing the endogeneity issue.

3.3 Demand Regression

We now consider a cross-sectional analysis of total demand in the pre-livestream and post-livestream periods, respectively.[6] Our dependent variable is the log-transformed total sales of a Live j in each time period. The independent variables include the log-transformed full price of Live j and a subset of covariates shown in Table 1. So the coefficient of price can be interpreted as elasticity, though it is most likely biased due to endogeneity. The other continuous independent variables are all standardized. For post-livestream demand, we additionally control for rating score and log-transformed total pre-livestream demand.

The estimation results are shown in Table 5. We highlight two key findings. First, we find

[6]We also replicate the demand regression using Live-day panel data and the results are consistent with cross-sectional analysis. Details are available from the authors upon request.

Table 5: Regression of Log Aggregate Demand

Dep. Var.	Log Pre-Livestream Demand	Log Post-Livestream Demand
Model	(1)	(2)
Live-Specific Features and Seasonality on Live Session Day		
Log Price	−0.718 (0.043)***	−0.039 (0.031)
Seat Limit	−0.007 (0.023)	−0.029 (0.016)*
Days before Live	0.200 (0.025)***	−0.134 (0.017)***
Advertised	1.237 (0.093)***	−0.396 (0.066)***
Topic Fixed Effects	Yes	Yes
Log Pre-Livestream Demand		0.805 (0.014)***
Rating		0.167 (0.027)***
Holiday	−0.366 (0.128)***	−0.105 (0.088)
Live Day of Week Fixed Effects	Yes	Yes
Live Week Fixed Effects	Yes	Yes
Creator Past Experience and Reputation in Live Market		
Live Market Lifetime	0.051 (0.046)	−0.106 (0.032)***
Invited Creator	0.108 (0.110)	0.117 (0.075)
Num. Past Lives	−0.256 (0.039)***	0.081 (0.027)***
Average Past Pre-Livestream Demand	0.118 (0.025)***	−0.011 (0.017)
Average Past Live Rating	0.066 (0.014)**	0.027 (0.021)
Creator Experience and Reputation in Q&A Community		
Q&A Platform Lifetime	0.079 (0.027)***	0.043 (0.019)**
Celebrity	0.859 (0.168)***	−0.459 (0.116)***
Excellent Contributor	−0.104 (0.072)	−0.077 (0.049)
Num. Followers	0.183 (0.035)***	0.016 (0.024)
Num. Followees	0.024 (0.025)	−0.010 (0.017)
Num. Articles	0.129 (0.035)***	−0.006 (0.024)
Num. Questions	−0.200 (0.036)***	0.014 (0.025)
Num. Answers	0.062 (0.032)*	−0.019 (0.022)
Num. Upvotes per Ans	0.247 (0.068)***	0.207 (0.047)***
Num. Downvotes per Ans	0.153 (0.051)***	0.027 (0.035)
Num. Thanks per Ans	0.033 (0.084)	−0.039 (0.057)
Num. Unhelpful per Ans	−0.277 (0.097)***	−0.138 (0.067)**
Num. Obs.	2,705	2,705
R^2	0.483	0.716

Note: Creator characteristics on the Live market and the Q&A community is computed by the starting time of each Live held by the creator (except for *InvitedCreator*, *Celebrity* and *ExcellentContributor*, which are fixed over time). $^{***}p < 0.01$, $^{**}p < 0.05$, $^{*}p < 0.1$.

that the price elasticity of pre-livestream demand (Model (1)) is significantly negative ($p < 0.01$), whereas the price elasticity of post-livestream demand (Model (2)) is insignificant. This suggests that post-livestream demand is less price-elastic than pre-livesream demand. Second, as expected, we find that the creator's past experience and reputation appear to have much larger effects on the

pre-livestream demand than on the post demand. That is possibly because consumers face with smaller quality uncertainty over post-livestream period when consumer ratings become available, and hence are less likely to leverage such information to make purchase decisions for recorded versions. We will test these mechanisms more formally in our section of mechanisms.

4 Modeling Approach

To address high-dimensional confounders in our causal identification, in this section we describe the ORF algorithm and introduce our methodological approach. For the ease of illustration, we use our empirical context to setup the notation, but it can be easily generalized to all other problems.

4.1 The General Problem

Let $Y_{it} \in \mathcal{R}$ denote the total number sales of Live i on day t, which is represented as a function of its treatment (i.e., price) on day t, denoted by T_{it}. All the observations $D = \{Z_{it} = (Y_{it}, T_{it}, W_{it}, X_{it})\}$ are drawn i.i.d. from some underlying distribution, where $W_{it} \in \mathcal{R}^p$ represents a multitude of control/confounding variables and $X_{it} \in \mathcal{R}^d$ is the feature vector that captures the heterogeneity in price elasticity. The outcome and treatment are assumed to follow a general specification as in Oprescu et al. (2018):

$$\begin{aligned} Y_{it} &= \theta_0(X_{it})T_i + f_0(X_{it}, W_{it}) + \varepsilon_{it} \\ T_{it} &= g_0(X_{it}, W_{it}) + \eta_{it} \end{aligned} \tag{1}$$

where ε_{it} and η_{it} represent the unobserved noise such that $\mathbb{E}[\varepsilon_{it}|T_{it}, X_{it}, W_{it}] = 0$ and $\mathbb{E}[\eta_{it}|X_{it}, W_{it}] = 0$, and θ_0 represents the treatment effect function. Note that the full set of confounding factors consists of W and X. We differentiate between the target feature and other controls using notation X as we are interested in how treatment effect evolves along the target feature X. The second equation keeps track of the confounding, namely the dependence of the treatment variable on controls. Thus, the confounding factors affect the treatment variable T via the nuisance function g_0 and influence the outcome Y via the nuisance function f_0 (Chernozhukov et al., 2018). Our

goal is to estimate the conditional average treatment effect (CATE), $\theta_0(x)$, conditioned on targeted feature x.

4.2 Methodological Background

We now briefly review two streams of research, in the intersection of causal inference and machine learning, that have been proposed to estimate (heterogeneous) treatment effects with observational data. The first is a two-stage estimation method, double machine learning (DML), introduced in Chernozhukov et al. (2017, 2018). DML first orthogonalizes out the effect of high-dimensional confounding factors using standard machine learning algorithms, e.g., lasso, deep neural networks (DNNs) or random forests. Then, it estimates the effect of the lower dimensional treatment variable, typically parametrically, by running a low-dimensional linear regression between the residualized treatment and residualized outcome. A challenge with this approach is that it handles heterogeneity in the treatment effect in a fairly limited manner, typically via pre-specified parametric distributions (Chernozhukov et al., 2017).

The second stream is to estimate causal heterogeneous treatment effects based on random forests. The most prevalent algorithm is the Generalized Random Forest (GRF) proposed by Athey et al. (2019). GRF assigns each observation a similarity weight derived from the fraction of trees in which an observation appears in the same leaf as the target feature point. It then solves the local estimation equation using the weighted set of "neighbors" at the particular value of target feature. GRF (and related methods) allow for fully flexible non-parametric estimation of heterogeneity in the treatment effect (Athey and Imbens, 2016; Wager and Athey, 2018; Athey et al., 2019). However, these methods tend to perform poorly in the presence of high-dimensional or "highly complex" controls arising from the required regularization bias - see Chernozhukov et al. (2018) for details.

Orthogonal Random Forest (ORF) leverages the benefits of both DML and GRF. It allows non-parametric estimation of the target parameter while permitting more complex nuisance functions with high-dimensional parameterizations (Oprescu et al., 2018). At a high level, ORF incorporates

DML into GRF by orthogonalizing the effect from high-dimensional controls via (local) two-stage estimation. In another words, ORF follows the residualization approach similar to DML to create an orthogonal moment for identifying $\theta_0(x)$ at each target point x. Importantly, ORF estimates are shown to be asymptotically normal and hence have theoretical properties that render bootstrap based confidence intervals asymptotically valid.

To illustrate the basic idea of ORF, we define a function $q_0(X,W) = \mathbf{E}[Y|X,W]$ for the problem introduced in Section 4.1. We define the residuals as $\widetilde{Y} = Y - q_0(X,W)$ and $\widetilde{T} = T - g_0(X,W)$. Then, one can simplify the equation as $\widetilde{Y} = \theta_0(X)\widetilde{T} + \varepsilon$, which leads to $\mathbf{E}[\widetilde{Y}|X,\widetilde{T}] = \theta_0(X)\widetilde{T}$. This relationship suggests that one can obtain an estimate of $\theta_0(x)$ by regressing $\widetilde{Y}$ on $\widetilde{T}$ locally around $X = x$. ORF achieves such an estimation, following the approach of GRF. Specifically, using a forest-based kernel learner, ORF first assigns a set of similarity weights between each sample and the target point x, and then computes $\hat{\theta}$ via kernel regression with the set of weights. We summarize all the essential steps of the ORF algorithm in Appendix B, along with the pseudo code.

The key causal identification requirement underlying all these methods (DML, GRF and ORF) is the unconfoundedness assumption. This states that all the variables (often called *covariates*) affecting both the treatment T and the outcome Y are observed and can be controlled for. That is, all the confounding variables must be known and be included in (X,W) in the Equation System (1). The Zhihu Live context offers us an ideal setting to apply ORF. Having access to Zhihu's internal data sets enables us to observe rich characteristics of all parties that may affect price and demand, as shown in Table 1. This helps us get as close as possible to the unconfoundedness assumption for causal identification.

The unconfoundedness assumption also suggests that the specifications of the nuisance functions (e.g., g_0 and f_0 in the Equation System (1)) are very important in making sure that all the confounding factors are properly controlled for. The existing literature primarily uses lasso to model the nuisance functions. Clearly, lasso helps maintain interpretability, but assumes linear relationship between model parameters and outcome/treatment, which could be quite restrictive in settings like ours. In comparison, purely nonparametric methods (e.g., random forests and DNNs)

could be used to improve flexibility and fitting, but they are not interpretable and can also lead to over-fitting issues. In our study context, price and demand are confounded with high-dimensional covariates in complex and partially known ways. For example, creator and time fixed effects are typically assumed to have linear relationships with treatment/outcome according to econometric convention, while other covariates are likely to have unknown functional forms. This motivates us to propose a semi-parametric approach to specify the nuisance function in the next section.

4.3 SDNNs for Nuisance Estimation

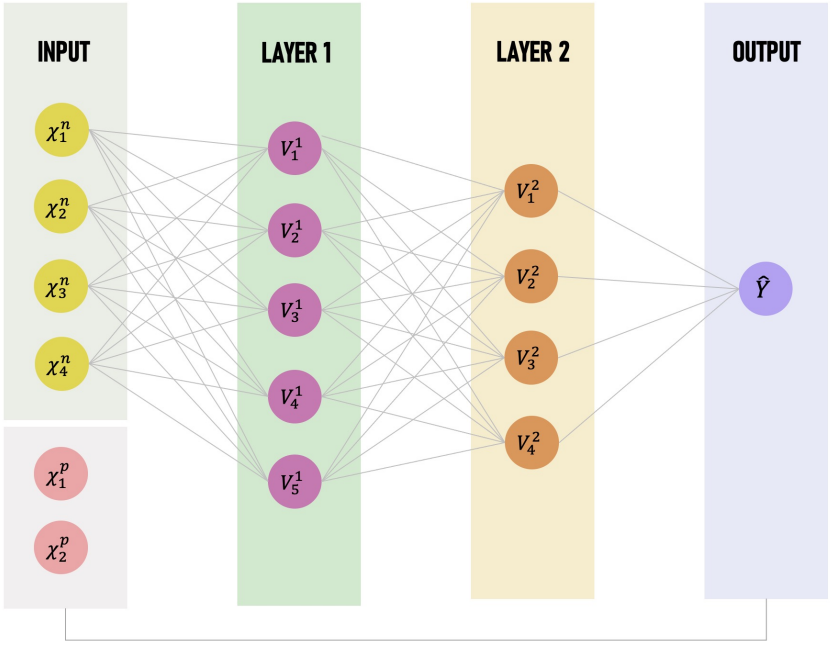


Figure 5: Semi-Parametric Neural Network Model

Note: The upper part of input is the nonparametric component, which will go through nonlinear transformation. The lower part of input is the parametric component, which has linear relationship with the target variable.

SDNNs combine a parametric form for some known covariates (usually based on prior knowledge about data-generating process or standard econometric techniques) with flexible nonparametric forms for the reminder of the covariates. The basic idea of SDNNs is illustrated by a network diagram in Figure 5. SDNNs allow partitioning of the covariates into two sets. The first set consists of covariates that are assumed to have a known relationship with the treatment and outcome

while the second set consists of covariates that are assumed to have an unknown relationship with the treatment and outcome. SDNNs are thus well-suited to settings where treatment and outcome are confounded with many observable factors in complex but partially known ways. For example, in panel data setting, our proposed approach can deal with clustering via fixed effects, while accommodating non-linear combinations of the remaining covariates in a totally flexible manner (Crane-Droesch, 2017; Athey and Wager, 2019). To the best of our knowledge, this paper is the first to adopt SDNNs in ORF to estimate the nuisance functions.

We operationalize SDNNs using a deep feedforward neural network architecture with multiple inputs. For ease of notation, let $\chi = (X, W)$ represent all the observed covariates. We form two sets based on χ, denoted as χ^p and χ^n, which enter the parametric and nonparametric components of the nuisance functions respectively. Then, we can re-write the nuisance functions as

$$q_0(\chi) = q_{01}(\chi^p) + q_{02}(\chi^n), \;\; g_0(\chi) = g_{01}(\chi^p) + g_{02}(\chi^n)$$

where, the first (second) function in each equation represents the known parametric (nonparametric) component. The SDNNs for estimating the nuisance function q is represented in below:

$$\begin{aligned} Y_{it} &= q_{01}(\chi^p_{it}) + V^1_{it}\Gamma^1 + \varepsilon_{it} \\ V^1_{it} &= \sigma(\gamma^2 + V^2_{it}\Gamma^2), V^2_{it} = \sigma(\gamma^3 + V^3_{it}\Gamma^3), \ldots \quad \ldots, V^L_{it} = \sigma(\gamma^L + \chi^n_{it}\Gamma^L) \end{aligned} \tag{2}$$

where, V^l are nodes at the lth layer, Γ^L are weights that map the data to the outcome via the intermediate nodes, and $\Gamma^{2:L}$ are weight matrices of dimensions equal to the number of nodes of the lth layer and the next layer up. The activation function $\sigma(.)$ accommodates the nonlinear functional mappings by converting the input signal (from the previous layer) to an output signal of the current layer. Common choices are sigmoids, hyperbolic tangent, and ReLU. The number of layers and the number of nodes per layer are predetermined hyperparameters. A similar SDNNs can be designed for estimating g, when replacing the outcome by T_{it}. Because the top layer is a linear model in χ^p and the derived variables V^1_{it}, covariates in the parametric component χ^p have

a linear relationship with the treatment and outcome. We train the SDNNs by minimizing a loss function which is defined as weighted sum-of-square errors between the true value and model estimates. See Appendix B for the underlying mathematics and see Appendix C for a detailed instruction of how to enable SDNNs in ORF using Python.

5 Price Sensitivity and Live Life-Cycle

We now apply ORF to understand how the price elasticity of demand changes over Live life-cycle. When estimating nuisance functions using SDNNs, we pass 1,458 dummy indicators into the parametric component. They are creator fixed effects (to control for unobserved creator characteristics), Live topic category fixed effects, and various time fixed effects (including transaction/livestream week, transaction/livestream day-of-week, and temporal distance to livestream). We pass the other 242 covariates shown in Table 1 into nonparametric component to allow for flexible interactions among them. As our objective in this section is to document the price elasticity variation relative to the temporal distance to the Live, we consider 29 test points within the window of 14 days before and after the scheduled live day, i.e., $d \in \{-14, -13, ..., 0, ...13, 14\}$. The hyperparmeter in the neural network is suitably chosen to maximize predictive performance of both the price and demand equations. Details are provided in Appendix D.

5.1 Main Results

Figure 6 reports the estimated price elasticities over the Live life-cycle, along with the 95% confidence intervals obtained via 100 bootstrap samples.[7] Over the pre-livestream period, the estimated price elasticity is significantly negative, but gradually becomes less negative over time. On the livestreaming day, the price elasticity actually reaches zero. The post-livestream demand is still price elastic, but it is much less sensitive than the pre-livestream demand. The difference between them can be as large as 80%.

[7]Figure 6 shows a small magnitude of fluctuation over the pre-livestream period. This may be driven by the fact that the number of unique Lives involve at each temporal distance are different over pre-stream period.

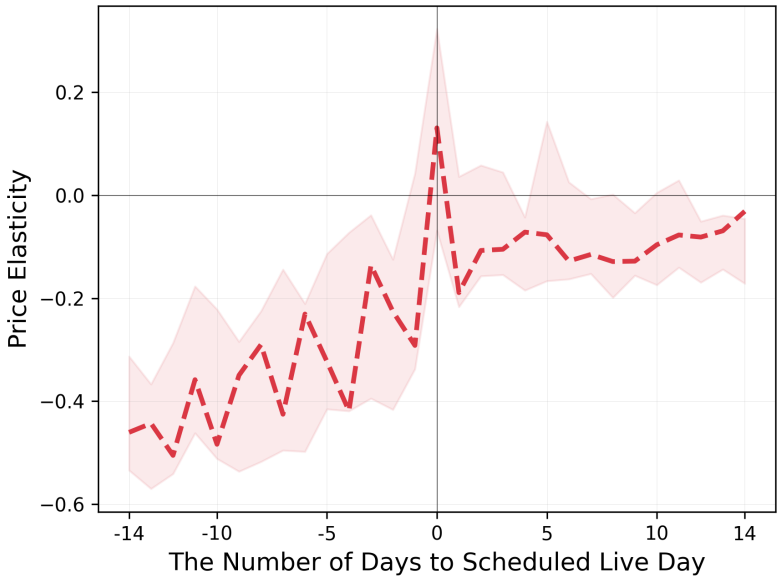


Figure 6: Price Elasticity over Live Life-Cycle

Note: The shaded region depicts 95% confidence interval via 100 bootstrap resamples.

Given that a significant proportion of sales happened on the livestreaming day, our research findings confirm that the live content format is highly valued by consumers in this market. In addition, our research findings suggest that the recorded version is also an important content strategy as it accounts for about 30% of sales and its demand is actually less price elastic than the demand for the live version. This is contrary to the common belief that livestreaming content does not have much residual value once the event is over (Vosgerau et al., 2006; Barber, 2017). Take all together, the overall temporal dynamics in price elasticity suggests the opportunities for platforms and creators to implement dynamic pricing, promotion, and targeting over event life-cycle. However, meaningful policy implications cannot be derived without a further understanding of mechanisms driving the results. Therefore, we will conduct a series of analysis in the next section to achieve this research objective.

Note that the magnitude of price elasticity is small in most cases, between -1 and 0, suggesting relatively inelastic demand in the Zhihu Live market. A natural question to ask is why creators do not raise prices. This is most likely due to the fact that there are norms that are typically established

for live events in a given market and event providers do not usually set prices outside these norms. This has been documented in previous research for many types of live events, such as concerts and performing arts (e.g., Felton, 1992; Pompe et al., 2018). Based on our survey, we find that the price points for events hosted on Zhihu Live were typically higher than those of comparable events on four competing platforms in China over the same time period. This suggests that there is little room for creators on Zhihu Live to increase their prices. Detailed statistics are presented in Appendix E.

5.2 Robustness Checks

In this section, we conduct a series of analysis to rule out possible alternative explanations, demonstrating the robustness of our main research findings shown in Figure 6.

5.2.1 Differences in Consumer Characteristics

One possibility is that the temporal dynamics in price elasticity of demand may be explained by different types of consumers arriving to purchase at different time. To investigate, we examine differences on observables by zooming in on three major types of consumer characteristics on the platform that are highly associated with consumers' levels of experience, engagement, and reputation: consumer lifetime, followee count and follower count on the Q&A community.[8] We regress each dependent variable at the time of purchase on Live fixed effects and transaction calendar-day fixed effects. Our variables of interests are all the fixed effects of temporal distances to livestreaming day, showing us how consumer characteristics vary over Live life-cycle. We consider transactions within 14-day before and after the livestreaming day.

The estimated fixed effects of temporal distances are displayed in Figure A4 of Appendix E. We find that the estimates are quite noisy and their differences are mostly insignificant (or very small in magnitude if significant). Therefore, it seems unlikely that differences across these observables

[8] We use consumer characteristics on the Q&A platform rather than those on the Live market because about 80% consumers only purchased once in our data period, leading to very sparse information about each consumer on Live market.

could account for about 80% change in consumer price elasticity from pre-period to post-period. Therefore, we conclude that differences in observed consumer characteristics are unlikely to be the main driver of the temporal dynamics in price plasticity over Live life-cycle.

5.2.2 Partitioned Consumers based on Purchase Preferences

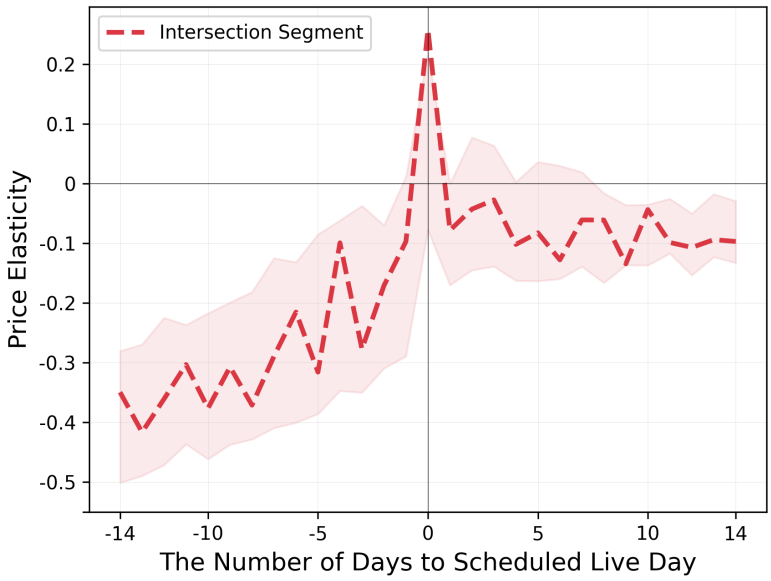
Another possibility is that consumers vary systematically on their preferences for consuming live versus recorded content. For example, consumers who purchase before or on the live day are likely to be those who value live interaction and would like to pay more for livestream. Likewise, consumers who always purchase after the event is held tend to be those who prefer watching recorded content and hence might be willing to pay as much or more for the recorded version than the live version. To examine whether our main results depend on such differences in consumer preferences, we partition all consumers into three segments: pre-loyal (consumers who have only purchased before day of livestream), post-loyal (consumers who have only purchased access to recorded versions of Lives), and intersection (consumers who have purchased Lives in both periods). Table 6 compares the three consumer segments along a few dimensions. The intersection segment is the most active one, including 20% of all the consumers and generating 64% of all the transactions in the market.

Table 6: The Three Consumer Segments

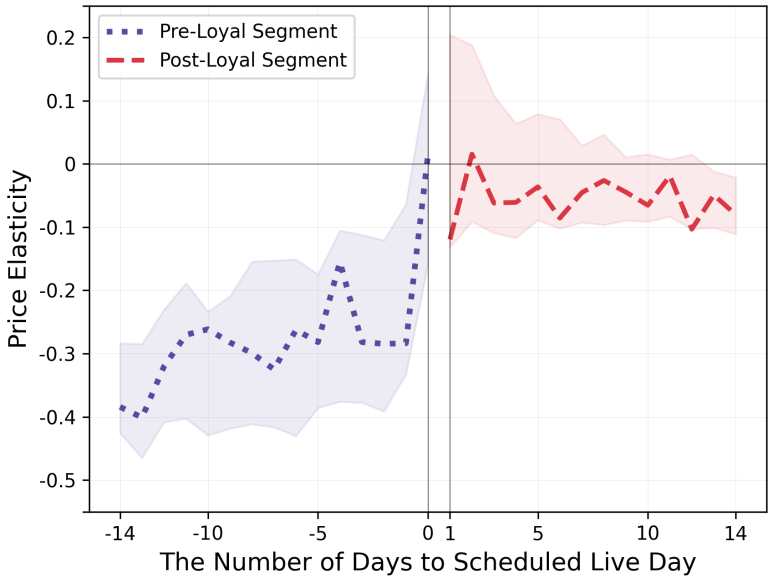
	Intersection	Pre-loyal	Post-loyal
Num. Consumers	181,233	537,938	183,766
% Consumers	20	59	21
% Transactions	64	26	10
Num. pre-purchases	7.66 (272.96)	1.80 (21.20)	0.00 (0.00)
Num. post-purchases	2.99 (11.95)	0.00 (0.00)	1.27 (2.24)

Note: The last two rows report the average (pre-stream and post-stream) purchases per consumer within each consumer segment, with the standard deviation in parentheses.

For each Live-day observation, we recompute the daily demand within each consumer segment. Then, for each consumer segment, we repeat our main analysis by applying ORF algorithm. But



(a) Price Elasticity of the Intersection Segment



(b) Price Elasticity of the Pre-loyal and Post-loyal Segment

Figure 7: Price Elasticity over Live Life-Cycle across Consumer Segments

Note: Pre-loyal (post-loyal) segment consists of consumers who have only purchased Lives over pre-livestream (post-livestream) period; intersection segment consists of consumers who have purchased over both periods. The shaded region depicts 95% confidence interval via 100 bootstrap resamples.

the difference is that the price elasticity of pre-loyal (post-loyal) segment can only be estimated for $d \in [-14, 0]$ ($d \in (0, 14]$). The estimation results are shown in Figure 7. One can see that the temporal patterns for consumers in the intersection segment is very similar to these shown in Figure 6. The pre-livestream (post-livestream) temporal patterns are also replicated in pre-loyal (post-loyal) segment. More importantly, we can see that the price elasticity over pre-period does not differ systematically between the pre-loyal segment and the intersection segment. The same is also true for the post-period price elasticity between post-loyal segment and the intersection segment. In conclusion, we do not have evidence that the temporal dynamics in price elasticity can be driven by consumer self-selection that based on preference for live versus recorded version.

5.2.3 Excluding Extreme Consumers

Table 6 shows that the standard deviation of the average number of purchases per consumer is large, indicating the existence of extreme consumers in this market. Hence, it is possibility that a small group of heavy consumers is driving the main estimation results. We therefore order consumers according to their total number of purchases and re-estimate the price elasticities after dropping the top 1% of consumers. The estimation results are shown in Figure A5 of Appendix E. We find that our main results are robust to the exclusion of these extreme consumers. In addition, we replicate this procedure for the three consumer segments defined based on purchase preferences, respectively. We also find that the estimated temporal patterns are robust across all segments.

6 Mechanisms

So far we have shown systematic dynamics in the price elasticity of demand over Live life-cycle. In this section, we explore the potential mechanisms that may drive our main results. We focus on two important features of live events: the opportunity of real-time interaction with content creator and consumer perception of content quality.

6.1 Real-Time Interaction Matters

We start by focusing on the inelastic demand on the livestreaming day. Its most intuitive mechanism is that consumers place very high value on the opportunity of immediate real-time interaction with content creators. Because consumers who purchased after seats are filled up can no longer interact with creators during livestream, this market setup enables us to examine how the opportunity of real-time interaction with content creators may influence price elasticity of demand.

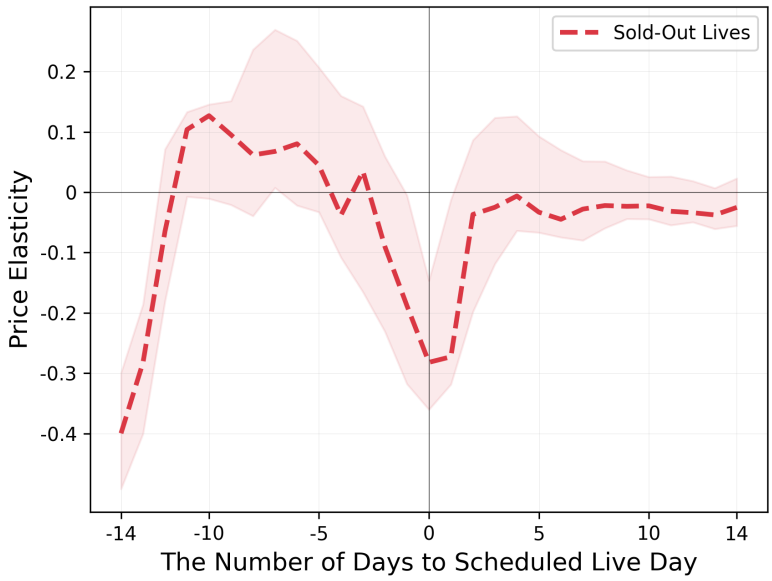


Figure 8: Price Elasticity over Live Life-Cycle among Filled-up Lives

Note: The shaded region depicts 95% confidence interval via 100 bootstrap resamples.

In this analysis, we focus on 317 Lives for which seats were filled up within one week prior to the livestreaming day.[9] If our proposed mechanism is at play, the price elasticity of demand for these Lives on the livestreaming day will be significantly negative. To test this prediction, we repeat our main analysis among these Lives. The ORF estimation results are displayed in Figure 8. Consistent with our expectation, there is a sharp drop in price elasticity, leading to a significantly negative price elasticity only on the livestreaming day. This result confirms that the

[9]The one week cutoff is conservative as a longer cutoff would make our result even stronger. We find that our results are robust if we use the data on all 583 Lives that were filled up before the day of airing.

inelastic demand on the livestreaming day is driven by the immediate "live" feature, as the seats of most Lives in the market can not be filled up.

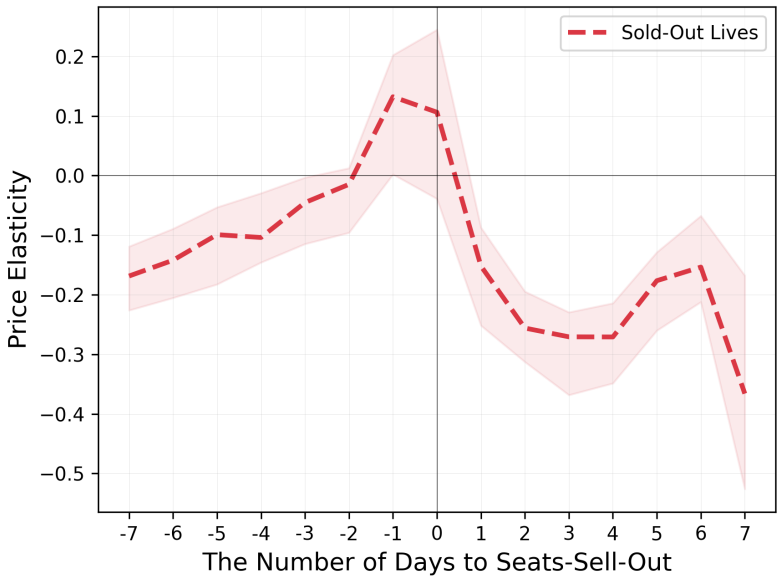


Figure 9: Price Elasticity Before and After Seats are Filled Up

Note: The shaded region depicts 95% confidence interval via 100 bootstrap resamples.

The above finding motivates us to investigate whether the opportunity of real-time interaction with creators can affect price elasticity of demand even when the livestream occurs a few days after the purchase (rather than on the day of purchase)? To investigate, we repeat the ORF estimation across these 317 Lives while using the number of days relatively to the day that a Live's seats are filled up as our target feature. The estimation results are displayed in Figure 9. Intuitively, we find an increasing trend before seats are filled up. And the elasticity reaches zero around two days before the filled-up day. In comparison, there is a sharp decreasing trend immediately after the filled-up day. In combination, these results suggest that the opportunity of real-time interacting with creators during livestream is an important consideration for consumers when purchasing Lives, regardless whether the Lives will be aired on the day of purchase or a few days later after purchase.

6.2 Quality Matters

In Zhihu Live market, consumers not only entertain through the interactive experience, but also benefit from content consumption (e.g., gain useful knowledge and skills in an interactive way). This means that content quality could be another important factor that influences consumer willingness to pay. If this is true, then demand should be less price sensitive for Lives with a higher quality (Zeithaml, 1988; Kirmani and Wright, 1989; Isik et al., 2004; Okada, 2010; Tsui, 2012). In our study context, consumer ratings should be the most important measure of Live-specific quality. In this market, ratings are provided (on a 5 point scale) by consumers who have already purchased and consumed a Live event. An average rating score, based on all the individual ratings, is displayed on each Live's webpage for future consumers to make purchase decision. Figure 10 shows the distribution of average rating scores across all the Lives in our data. We can see that about 95% of Lives have a rating between 3 and 5.

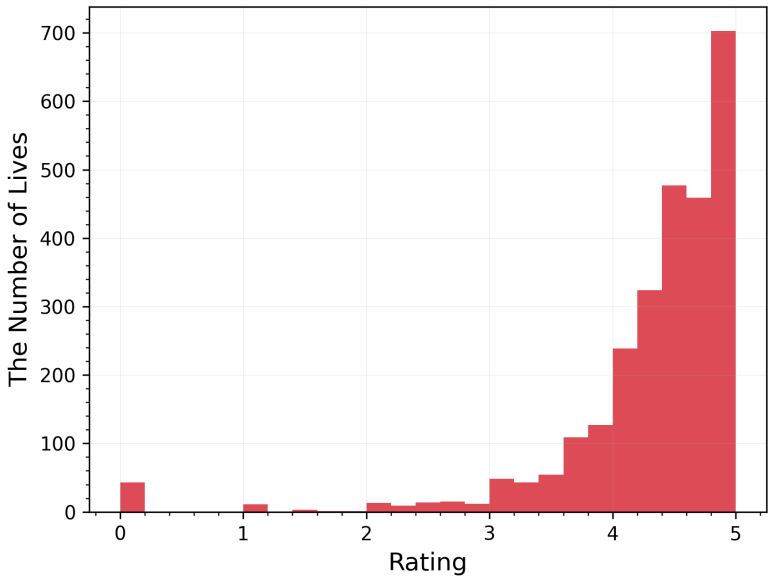


Figure 10: Distribution of Rating

To test our hypothesis, we apply ORF while using rating score as the target feature. As the rating of a Live is only available after its livestream ends, we examine the impact of rating scores

on price elasticity of post-livestream demand only. We use six test points evenly distributed over the interval $[3,5]$. The estimation results are shown in Figure 11. Consistent with our expectation, we find that demand is significantly less price sensitive when the rating is higher. For example, price elasticity is on average 65% lower with a rating of 5 than with a rating of 3. In summary, our research findings suggest that consumers have much higher willingness-to-pay for higher quality, confirming that quality matters in Zhihu Live market.

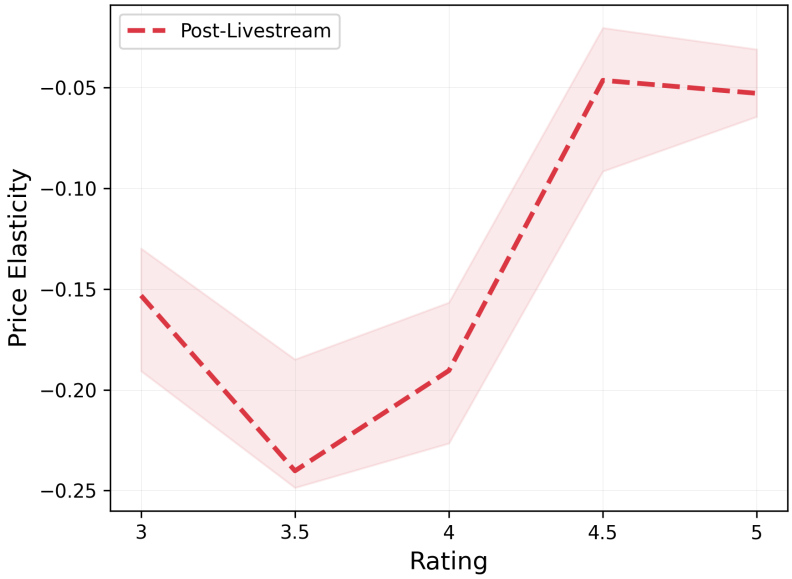


Figure 11: Price Elasticity and Rating Score

Note: The shaded region depicts 95% confidence interval via 100 bootstrap resamples.

6.3 Quality Uncertainty Reduction through Social Proof

We now turn to mechanism that may explain our main findings that demand becomes much less price-sensitive as approaching the livestreaming day. The experiential nature of knowledge goods enables very limited information for consumers to assess the quality of a Live event, given the fact that most creators host only one Live in the market. This implies that consumers face with considerable quality uncertainty at the time of purchase. The prior literature show that consumers typically use informations cues to lower uncertainty (Urbany et al., 1989). One popular way to infer

quality is through popularity information, e.g., the number of consumers who have purchased the same product (Cai et al., 2009; Tucker and Zhang, 2011). This implies that as more information cues about a Live, such as cumulative sales and consumer ratings, become available over Live life-cycle, consumer face lower quality uncertainty and hence become less price-sensitive. If our hypothesis is true, we expect to see a significantly positive association between price elasticity and cumulative demand. In addition, if uncertainty reduces after the livestream concludes, we expect that the popularity information has a smaller association with price elasticity.

To investigate, we apply ORF algorithm while setting the (log-transformed) cumulative demand of a Live as our target feature for each time period separately. Figure 12 reports the estimation results with seven testing points. Consistent with the pattern in our main results, we find that, on average, price elasticity of demand is much stronger (i.e., more negative) over the pre-livestream period. More importantly, over the pre-livestream period, price elasticity (in absolute value) decreases as the cumulative demand increases, suggesting that the social proof embedded in popularity information plays an important role over pre-livestream period. In contrast, over the post-livestream period, the price elasticity of demand for the recorded version is almost inelastic to cumulative demand. These results provide suggestive evidence that consumers only rely on popularity information to make their purchase decision over the pre-livestream period i.e., when the quality uncertainty is high.

One may argue that cumulative demand just acts as a proxy for the temporal distance to livestreaming day, so the pattern in Figure 12 may not be driven by the social proof. To test this, we apply ORF algorithm using all the pre-livestream observations, while setting the cumulated demand and the temporal distance as our target features simultaneously. We consider a grid of $7 \times 14 = 98$ test pairs. The results are shown in Figure 13. We find that the pattern in Figure 12 can be replicated at every temporal distance (the x-axis). This suggests that the social proof mechanism is present regardless of the temporal distance to airing. Finally, we want to emphasize that the social proof mechanism not only explains the less price-sensitive demand over post-livestream period, but also the temporal dynamics in price elasticity over the pre-livestream period.

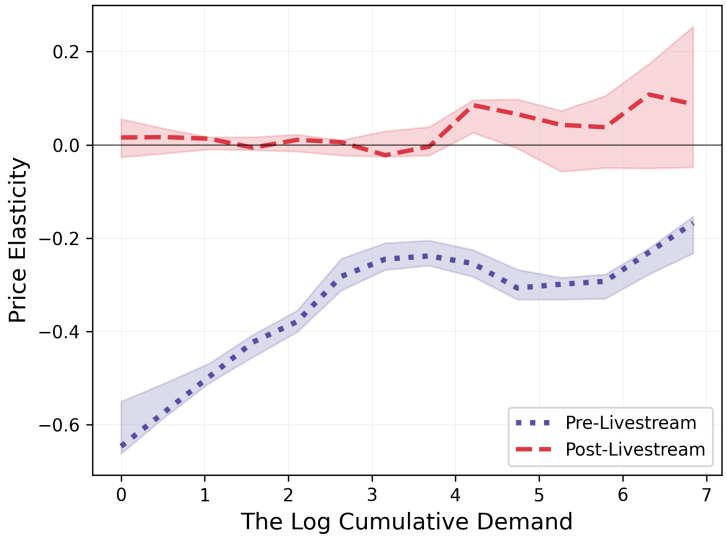


Figure 12: Price Elasticity and Cumulative Demand

Note: The shaded region depicts 95% confidence interval via 100 bootstrap resamples.

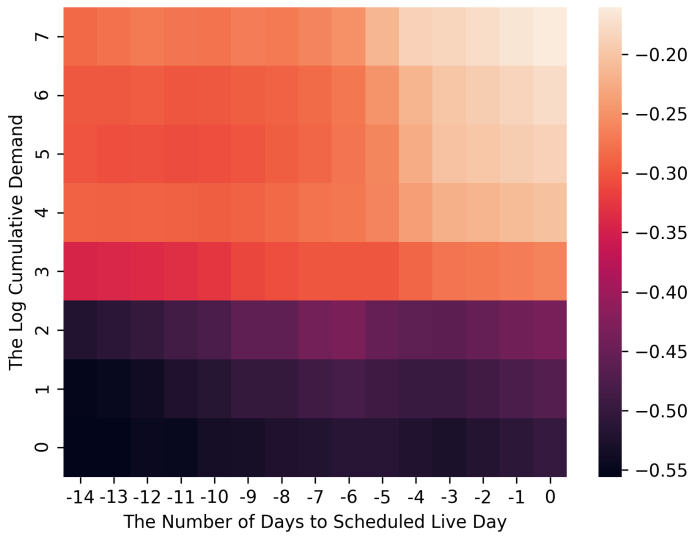


Figure 13: How Price Elasticity Varies over Live Life-Cycle and by Cumulative Demand

6.4 Consumer Reliance on Other Information Cues

We now provide more empirical evidence on the mechanism of quality uncertainty reduction that can also explain the reduced price sensitivity of demand for recorded version relative to that for the live version. The basic idea is that quality uncertainty for the recorded version are reduced as consumer ratings and reviews become available. If our hypothesis is true, we expect to see a significantly smaller impact of the same information cues on price elasticity of demand for recorded version than for live version. In this study, we consider three relatively stable information cues over a Live life-cycle: consumer prior knowledge, creator reputation, and content category.

Creator Credibility. Previous literature on reputation building suggests that consumers often rely on seller credibility to infer product quality, and tend to have higher willingness-to-pay if the seller has high credibility (Tellis, 1988; Bolton, 1989; Bijmolt et al., 2005; Hitsch et al., 2019). On Zhihu, even if a consumer has never followed a creator, the consumer may still be able to reduce quality uncertainty of a Live by examining the creator's reputation in the free Q&A community upon purchase. In particular, a creator's total number of followers is the most important indicator of his/her expertise and reputation in the Q&A community (Toubia and Stephen, 2013; Goes et al., 2014). If a creator's number of followers is considered when consumers make purchase decision, we expect that consumers will be less price-sensitive to Lives hosted by creators with larger number of followers, especially over pre-livestream period.

To test this hypothesis, we apply ORF algorithm for pre-livestream and post-livestream periods, respectively, while setting the normalized number of followers that creators have as our target feature. Our procedure can be described as follows. We first compute the cumulative follower number of each creator i by the time of his/her Live j that belong to topic k, denoted as F_{ij_k}. Because there is very large heterogeneity in readership across topics on Zhihu, the number of followers is meaningful indicator of creator credibility only among creators who host Lives within the same topic category.[10] Thus, for each topic k we obtain the maximum number of followers

[10]For example, creator A who is active in popular topics that suit mainstream tastes, such as Information Technology, is more likely to have larger follower size than the equally-successful creator B who is active in topics with a narrower

across all the creators who have hosted at least one Live on topic k, denoted as M_k. We divide each creator's follower size by the maximum in his/her topic category, to control for inherent differences in popularity across topics. That is, our target feature is $F_{ij_k}/M_k \in [0,1]$, which we call *follower share*.

The ORF estimation results with 10 evenly distributed test points are shown in Figure 14. One can see a clearly increasing trend in both periods, confirming that consumers are less price sensitive if creators have higher reputation on a given topic. In addition, the changes in the magnitudes of price elasticities across follower share are significantly sharper over pre-livestream period than over post-livestream period. This is also consistent with our earlier argument that consumers are more likely to use information cues to infer Live quality when Live-specific consumer rating is not available. When comparing the magnitudes of the two lines, we find that consumers are overall less price sensitive over post-livestream period than over pre-livestream period when the follower share is below 0.6, which is consistent with our main results. As the follower share reaches one, which describes highly established creators, demand become price inelastic or even exhibits slightly positive elasticity. These results imply that a creator's number of followers is a key piece information that consumers use to determine Live quality, especially over the pre-livestream period.

Consumer Prior Knowledge. In addition to creator reputation, which is public information, consumers can infer the quality of a Live from their own prior knowledge about the content creator. One way for consumers to know more about a creator is through following the creator's content contribution activities in the free Q&A community. Following this logic, we split all the transactions into two groups based on whether each consumer had followed the creator before making purchase. We find that about 20% of consumers had done so before purchasing, and these consumers generated about 17% of all transactions. We label these consumers as the "high prior knowledge" group, while others are labeled as the "low prior knowledge" group. For each group, we aggregate all the transactions at Live-day level, and repeat our main analysis. Figure 15 presents the ORF estimation results. First, the overall temporal dynamics shown in Figure 6 are replicated in

appeal, such as Humanities.

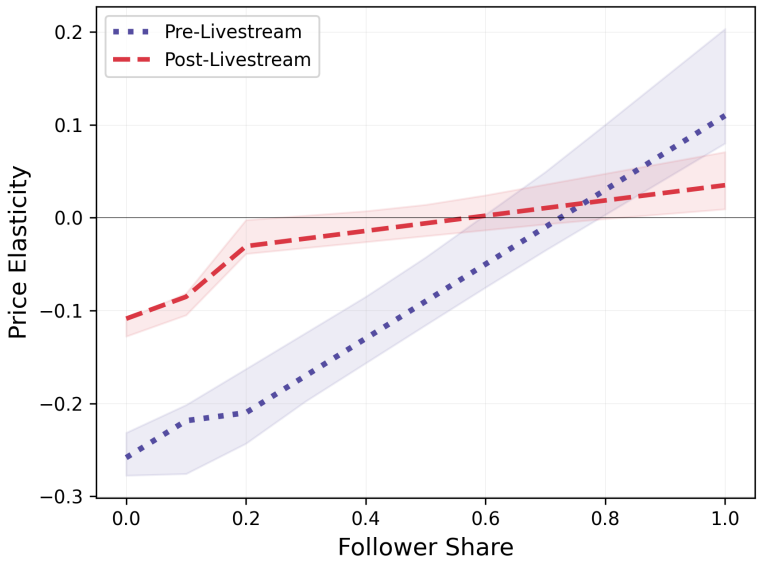


Figure 14: Price Elasticity and Creator Reputation in the Q&A Community

Note: The shaded region depicts 95% confidence interval via 100 bootstrap resamples.

both consumer groups. Second, the group with high prior knowledge is significantly less price sensitive than the other group at all temporal distances, except for a few days around the livestreaming day. This observation confirms our hypothesis that prior knowledge about a creator can influence consumer perceived quality uncertainty and hence price sensitivity. Third, the differences in price elasticity between these two consumer groups gradually become smaller over Live life-cycle. This pattern implies that consumers are much less likely to rely on their prior knowledge to resolve uncertainty over the post-livestream period. This is also consistent with our previous results on rating and social proof.

Live Topic Category. The existing literature suggests that one way consumers mitigate uncertainty about a purchase decision is through the use of a heuristic that rests on product category characteristics (Bettman, 1974; Scattone, 1995). For example, a product might be perceived to be of lower quality when there is higher quality variance in the corresponding category. In the context of Zhihu Live, consumers are likely to rely on the topic category of a Live to mitigate uncertainty. Each Live can belong to one of 17 topic categories in the Zhihu Live market. Using common

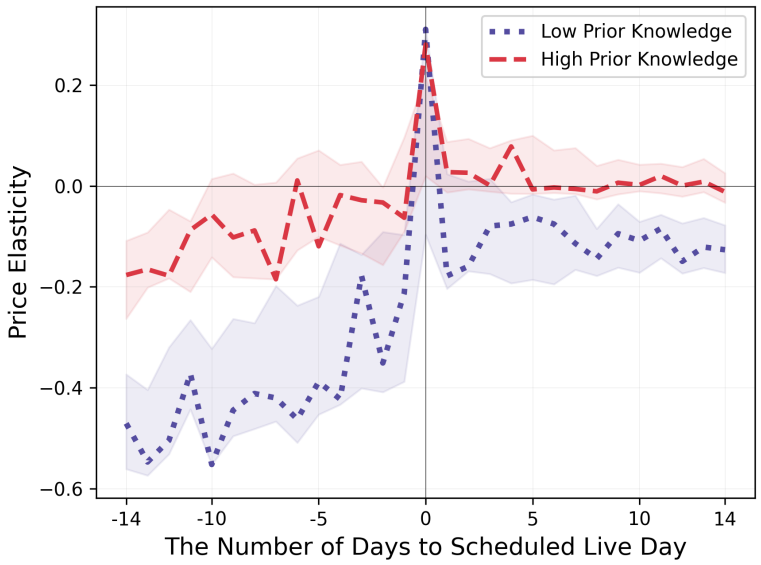


Figure 15: Price Elasticity and Consumer Prior Knowledge

Note: The shaded region depicts 95% confidence interval via 100 bootstrap resamples.

perceptions of these categories, we classify eight of these as "*hard*," as they usually have more objective content than the remaining nine, which we classify as "*soft*." The *hard* categories are Law, Business, Econ&Fin, Healthcare, Architecture&Interior Design, Art Appreciation&History, Science&Technology and Sports. The *soft* categories are Internet, Psychology, Education, Travel, Life Style, Food & Cuisine, Career, Reading & Writing, Music/Game/Movie.[11] The literature suggests that soft categories generally have higher (perceived) quality uncertainty than hard categories (Levitt, 1981; Bebko, 2000; Castelo et al., 2019; Dai et al., 2020). We also see this in our data as the variance in rating across Lives in the soft categories is significantly higher than that in the hard categories ($p < 0.05$).

We apply ORF to estimate how the price elasticity of the pre- and post-livestream demand varies across the 17 topic categories, respectively. Figure 16 reports the estimation results, with the eight hard topics on the left of the figure and the nine soft topics on the right. First, consistent with our previous results, demand is significantly less price-sensitive over the post-livestream

[11] We do not claim that this classification is based on formal criteria. However, as our objective is to provide evidence for the uncertainty reduction mechanism, misclassification of one topic here or there is unlikely to matter.

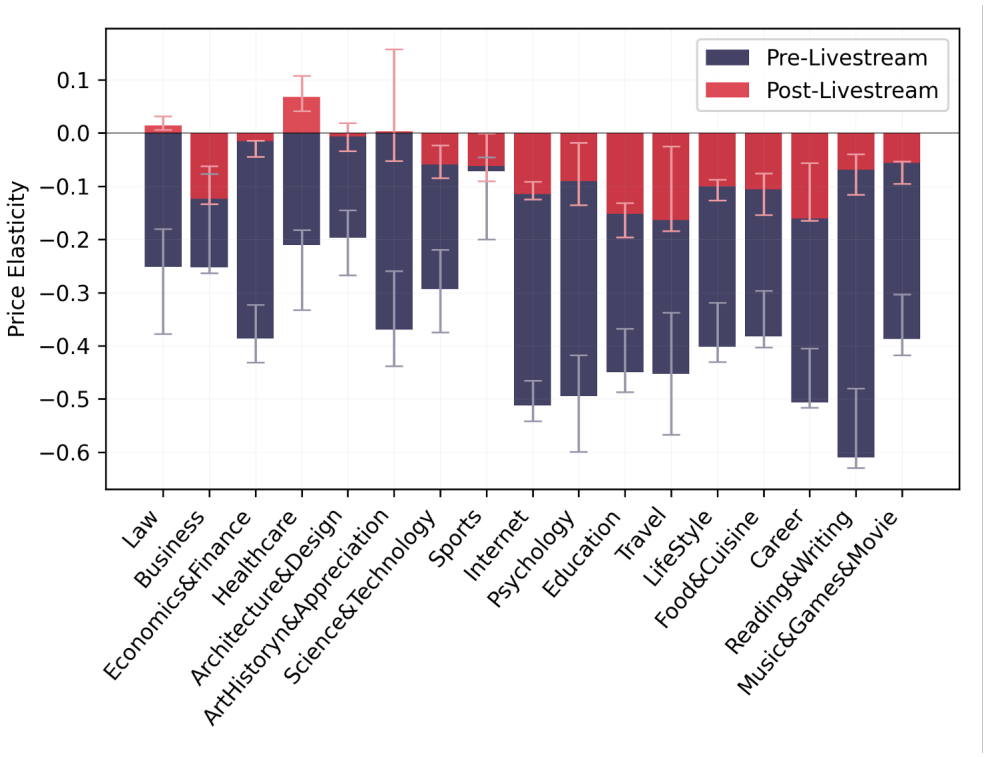


Figure 16: Price Elasticity and Live Topic Category

Note: The error bars depict 95% confidence interval via 100 bootstrap resamples.

period than over the pre-livestream period across all topics. Second, demand is much more price-sensitive for the soft topics than for the hard topics, over both pre- and post-livestream periods. This confirms with our assertion that uncertainty level differs across topics and that it affects consumer price sensitivity. Lastly, by comparing the magnitudes of the price elasticities between hard and soft topical categories within each time period, we find that their differences are significantly higher over pre-livestream period than over post-livestream period ($p < 0.05$). This suggests that consumers are more likely to rely on category characteristics (as heuristics) for pre-livestream purchases than for post-livestream purchases. This sharp difference provides confirming evidence on our proposed quality uncertainty reduction mechanism.

7 Conclusion and Discussion

This paper represents one of the very early studies on the fast growing phenomenon of livestreaming. While there is widespread belief that the value of livestreaming lies in its live part, there is virtually no research that has examined this. In this paper, using data from a large livestreaming platform (Zhihu Live), we build a modeling framework to estimate this value over the life-cycle of the livestream. By leveraging the fact that Zhihu Live also provides access to recorded, non-live, content at the same price as for the live content, we document the dynamics in price responsiveness of demand over temporal distance to the scheduled livestreaming day and after. We find that demand becomes less price-sensitive gradually over time to the scheduled live day and is inelastic on the day of livestream. Over the post-livestream period, demand is still price elastic, but surprisingly, it is much less sensitive than the pre-livestream demand.

We further provide correlationally consistent evidence for two potential factors that drive our main results. First, we show that consumers value the opportunity of real-time interaction with creators highly. This lowers their price sensitivity dramatically, leading to inelastic demand on the day of livestream. Second, we show that the reduced price sensitivity of post-livestream period can be partially attributed to quality uncertainty reduction. It turns out that consumers are much better at assessing the quality of the content after the event, leading to an on average, lower price sensitivity for the recorded (non-live) content.

Our research findings have important managerial implications. First, the results confirm that some of the value of livestream lies in the real-time consumer-creator interaction. Second and more surprisingly, the results show that live content has value beyond the live part. In other words, there is considerable value in the content that is not centered in its live attribute. Currently, many livestreaming platforms (e.g., Instagram Live, YouNow) only support live broadcasts and do not offer build-in recording functions; some platforms (e.g., Facebook Live, Periscope) offer built-in recording and replay functions, but generally devote very few resources towards managing recorded versions of live events. Our research findings are likely to be valuable to both streamer-

s/creators and platforms in the sense that they encourage them to provide both live and recorded content to enhance revenue and provide value to consumers.

In addition, marketing efforts to reduce uncertainty about upcoming events could also be beneficial for all parties (consumers, creators, and platforms). Specifically, platforms can improve their online rating system and other operating policies (e.g., filtering and search functions) to reduce information asymmetry. Creators can benefit from providing detailed descriptions of upcoming events and building reputation in the Q&A community. More broadly, our findings on the value of livestreaming content over its entire life-cycle can be used by platforms and creators to implement dynamic pricing, promotion, and targeting. For example, it could be beneficial for platforms and creators to target consumers who have lower uncertainty (e.g., followers of creators) in the early stage of pre-livestream period, while target those who are higher quality concerns in the post-livestream period. Also, price promotion appears to be more profitable early in the pre-livestream period, when demand is relatively more sensitive to price.

Methodologically, we demonstrate a novel approach for causal inference using observational data. In particular, we generalize ORF via the use of SDNNs for nuisance estimation. The generalized ORF is well suited to panel data because it can handle clustering effect via fixed effects, while allowing for nonlinear transformation of other covariates. We believe that this framework is applicable in other real-world settings with highly-differentiated goods and high-dimensional covariates, such as peer-to-peer markets and gig-economy platforms.

We hope this paper will stimulate further empirical study of livestreaming markets, especially as our study has some limitations. First, we use data from a single livestreaming platform where creators provide paid live events. Replicating the findings using data from different platforms will be important to confirm the generalizability of the findings. Second, we explore two potential factors that drive the temporal dynamics in price elasticity based on consumer purchase behavior. However, it will be important to understand the observed pattern from consumer psychology point of view and actual consumption pattern (e.g., watching behavior). Further research could explore this type of factors if individual consumption data is available. Third, there is no intertemporal

price variation in the price of a Live in our data. Thus, future research can examine the impact of more complex pricing strategies. Fourth, our research focuses on the effect of price while controlling for other (non-price) marketing mix elements, e.g, advertising. However, an important area of research is the effectiveness of these elements and their joint impact with price over the life-cycle of livestreaming.

References

Athey, S. and G. Imbens (2016). Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences 113*(27), 7353–7360.

Athey, S., J. Tibshirani, S. Wager, et al. (2019). Generalized random forests. *The Annals of Statistics 47*(2), 1148–1178.

Athey, S. and S. Wager (2019). Estimating treatment effects with causal forests: An application. *arXiv preprint arXiv:1902.07409*.

Bakhtiari, K. (2021, April). The creator economy, NFTs and marketing. https://www.forbes.com/sites/kianbakhtiari/2021/04/18/the-creator-economy-nfts-and-marketing/?sh=6d9687821204. Accessed on 6 June, 2021.

Barber, N. (2017, August). Prepare to support video livestreaming for customer experiences. https://www.forrester.com/report/Prepare+To+Support+Video+Livestreaming+For+Customer+Experiences/RES135382. Accessed on Aug 14, 2021.

Bebko, C. P. (2000). Service intangibility and its impact on consumer expectations of service quality. *Journal of services marketing*.

Bettman, J. R. (1974). Relationship of information-processing attitude structures to private brand purchasing behavior. *Journal of Applied psychology 59*(1), 79.

Bijmolt, T. H., H. J. Van Heerde, and R. G. Pieters (2005). New empirical generalizations on the determinants of price elasticity. *Journal of marketing research 42*(2), 141–156.

Bolton, R. N. (1989). The relationship between market characteristics and promotional price elasticities. *Marketing Science 8*(2), 153–169.

Buzanakova, A. and E. Ozhegov (2016). Demand for performing arts: the effect of unobserved quality on price elasticity. *Higher School of Economics Research Paper No. WP BRP 156*.

Cai, H., Y. Chen, and H. Fang (2009). Observational learning: Evidence from a randomized natural field experiment. *American Economic Review 99*(3), 864–82.

Castelo, N., M. W. Bos, and D. R. Lehmann (2019). Task-dependent algorithm aversion. *Journal of Marketing Research 56*(5), 809–825.

Chernozhukov, V., D. Chetverikov, M. Demirer, E. Duflo, C. Hansen, and W. Newey (2017). Double/debiased/neyman machine learning of treatment effects. *American Economic Review 107*(5), 261–65.

Chernozhukov, V., D. Chetverikov, M. Demirer, E. Duflo, C. Hansen, W. Newey, and J. Robins (2018). Double/debiased machine learning for treatment and structural parameters: Double/debiased machine learning. *The Econometrics Journal 21*(1).

Chernozhukov, V., M. Goldman, V. Semenova, and M. Taddy (2017). Orthogonal machine learning for demand estimation: High dimensional causal inference in dynamic panels. *arXiv preprint arXiv:1712.09988*.

CIW Team (2021, March). Profile of zhihu: China's Quora, a top content community. https://www.chinainternetwatch.com/31732/zhihu-profile/. Accessed on June 7, 2021.

Crane-Droesch, A. (2017). Semiparametric panel data models using neural networks. *arXiv preprint arXiv:1702.06512*.

Dai, H., C. Chan, and C. Mogilner (2020). People rely less on consumer reviews for experiential than material purchases. *Journal of Consumer Research 46*(6), 1052–1075.

Davis, J. and S. B. Heller (2017). Using causal forests to predict treatment heterogeneity: An application to summer jobs. *American Economic Review 107*(5), 546–50.

Felton, M. V. (1992). On the assumed inelasticity of demand for the performing arts. *Journal of Cultural Economics 16*(1), 1–12.

Fergus (2020, March). The growth of paid knowledge indusy in China 2020. https://www.iimedia.cn/c1020/70056.html. Accessed on June 7, 2020.

Gibbs, C., D. Guttentag, U. Gretzel, J. Morton, and A. Goodwill (2018). Pricing in the sharing

economy: a hedonic pricing model applied to airbnb listings. *Journal of Travel & Tourism Marketing 35*(1), 46–56.

Goes, P. B., M. Lin, and C.-m. Au Yeung (2014). "popularity effect" in user-generated content: Evidence from online product reviews. *Information Systems Research 25*(2), 222–238.

Henderson, G. (2021, May). What is the creator economy? https://www.digitalmarketing.org/blog/what-is-the-creator-economy. Accessed on 20 May, 2021.

Hitsch, G. J., A. Hortacsu, and X. Lin (2019). Prices and promotions in us retail markets: Evidence from big data. Technical report, National Bureau of Economic Research.

Hu, M., M. Zhang, and Y. Wang (2017). Why do audiences choose to keep watching on live video streaming platforms? an explanation of dual identification framework. *Computers in Human Behavior 75*, 594–606.

Iqbal, M. (2021, March). Twitch revenue and usage statistics. https://www.businessofapps.com/data/twitch-statistics/#:~:text=Trafficcontinuedtogrow,with,1.4millioninQ12020. Accessed on May 19, 2021.

Isik, M. et al. (2004). Does uncertainty affect the divergence between wtp and wta measures? *Economics Bulletin 4*(1), 1–7.

John, A. (2020, November). 5 powerful reasons why live streaming is so important for enterprises. https://hackernoon.com/5-powerful-reasons-why-live-streaming-is-so-important-for-enterprises-ta123wts. Accessed on June 7, 2021.

Joo, M., D. K. Gauri, and K. C. Wilbur (2020). Temporal distance and price responsiveness: Empirical investigation of the cruise industry. *Management Science 66*(11), 5362–5388.

Kirmani, A. and P. Wright (1989). Money talks: Perceived advertising expense and expected product quality. *Journal of consumer research 16*(3), 344–353.

Lanzetta, J. T. (1963). Information acquisition in decision making. *Motivation and social interaction-cognitive determinants*, 239–265.

Larson, K. (2021, March). Retailers embrace livestreaming, market expected to reach $11 billion in 2021. https://www.forbes.com/sites/kristinlarson/2021/03/27/

retailers-embrace-livestreaming-market-expected-to-reach-11-billion-in-2021/?sh=7ad6fae52fde. Accessed on May 19, 2021.

Levitt, T. (1981). Marketing intangible products and product intangibles. *Cornell Hotel and Restaurant Administration Quarterly 22*(2), 37–44.

Li, J., A. Moreno, and D. Zhang (2016). Pros vs joes: Agent pricing behavior in the sharing economy. *Ross School of Business Paper* (1298).

Lin, Y., D. Yao, and X. Chen (2021). Express: Happiness begets money: Emotion and engagement in live streaming. *Journal of Marketing Research*, 00222437211002477.

Lu, S., D. Yao, X. Chen, and R. Grewal (2021). Do larger audiences generate greater revenues under pay what you want? evidence from a live streaming platform. *Marketing Science (forthcoming)*.

Okada, E. M. (2010). Uncertainty, risk aversion, and wta vs. wtp. *Marketing Science 29*(1), 75–84.

Oprescu, M., V. Syrgkanis, and Z. S. Wu (2018). Orthogonal random forest for causal inference. *arXiv preprint arXiv:1806.03467*.

Pompe, J., L. Tamburri, and J. Munn (2018). Subscription ticket sales for symphony orchestras: Are flexible subscription tickets sustainable? *Managerial and Decision Economics 39*(1), 71–78.

Scattone, J. (1995). Factors influencing consumer perceptions, attitudes, and consideration of sbs. In *AMA Summer Marketing Educators' Conference Proceedings*. Chicago: American Marketing Association.

Shapiro, C. (1982). Consumer information, product quality, and seller reputation. *The Bell Journal of Economics*, 20–35.

Statista (2020). Market size of online live streaming in China in 2018 and 2019 with a forecast for 2020. *Available online at https://www.statista.com/statistics/874591/china-online-live-streaming-market-size/#statisticContainer*.

Tellis, G. J. (1988). The price elasticity of selective demand: A meta-analysis of econometric models of sales. *Journal of marketing research 25*(4), 331–341.

The Economist (2021, May). The new rules of the "creator economy". https://www.economist.

com/briefing/2021/05/08/the-new-rules-of-the-creator-economy. Accessed on 7 June, 2021.

Thomas, L. and A. Palmer (2021, May). U.S. retailers scramble to crack the code on livestream shopping events. https://www.cnbc.com/2021/05/03/retailers-from-bloomingdales-to-petco-test-livestreaming-to-win-sales.html. Accessed on May 19, 2021.

Toubia, O. and A. T. Stephen (2013). Intrinsic vs. image-related utility in social media: Why do people contribute content to twitter? *Marketing Science 32*(3), 368–392.

Tsui, H.-C. (2012). Advertising, quality, and willingness-to-pay: Experimental examination of signaling theory. *Journal of Economic Psychology 33*(6), 1193–1203.

Tucker, C. and J. Zhang (2011). How does popularity information affect choices? a field experiment. *Management Science 57*(5), 828–842.

Urbany, J. E., P. R. Dickson, and W. L. Wilkie (1989). Buyer uncertainty and information search. *Journal of consumer research 16*(2), 208–215.

Varian, H. R. (1999). *Markets for information goods*, Volume 99. Citeseer.

Vosgerau, J., K. Wertenbroch, and Z. Carmon (2006). Indeterminacy and live television. *Journal of Consumer Research 32*(4), 487–495.

Wager, S. and S. Athey (2018). Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association 113*(523), 1228–1242.

Wang, D. and J. L. Nicolau (2017). Price determinants of sharing economy based accommodation rental: A study of listings from 33 cities on airbnb. com. *International Journal of Hospitality Management 62*, 120–131.

Wheless, E. (2021). Lg kicks off series of live stream shopping events produced in-house. *Available online at http://ec2-34-225-151-148.compute-1.amazonaws.com/retail/lg-kicks-off-series-of-live-stream-shopping-events-produced-in-house/*.

Zeithaml, V. A. (1988). Consumer perceptions of price, quality, and value: a means-end model and synthesis of evidence. *Journal of marketing 52*(3), 2–22.

Zhang, Q., W. Wang, and Y. Chen (2020). Frontiers: In-consumption social listening with moment-

to-moment unstructured data: The case of movie appreciation and live comments. *Marketing Science 39*(2), 285–295.

Zieba, M. (2009). Full-income and price elasticities of demand for german public theatre. *Journal of Cultural Economics 33*(2), 85–108.

A Examples of Zhihu Interfaces

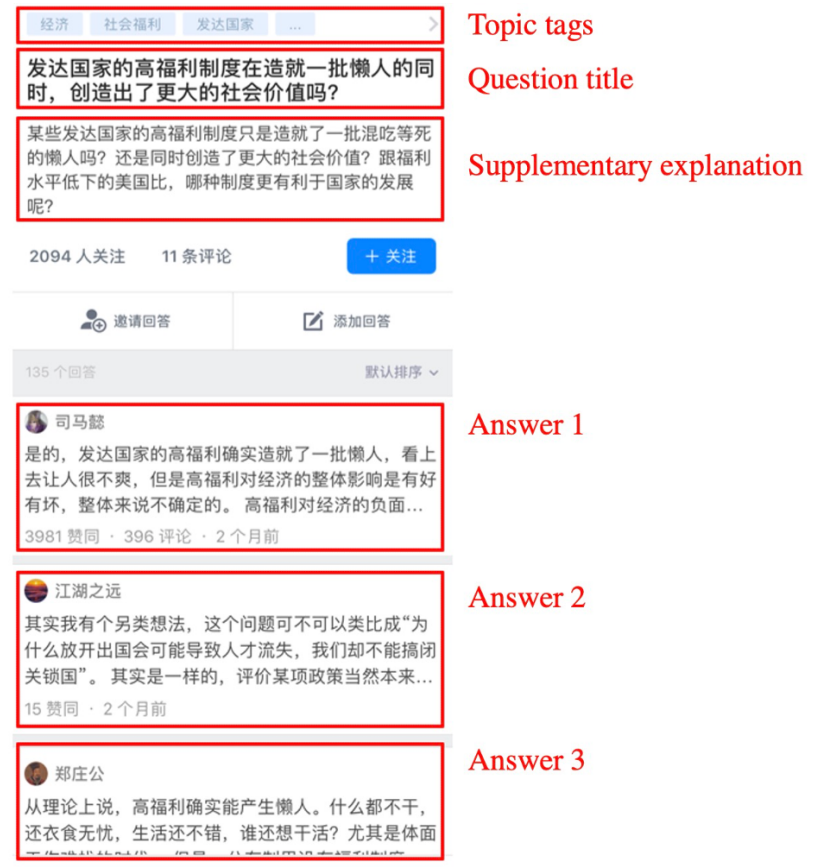


Figure A1: Example of a Question and Its Answers

Figure A2: Example of User Profile Page

Figure A3: Example of a Live Session

Note: This figure illustrates the interface of a Live. During the livestream, the creator will give a real-time talk via voice/text/picture messages. The creator can also address questions raised by consumers. Notice that only consumers who purchased the Live before seats are filled up are able to raise questions and leave comments during the livestream.

B Details of ORF Algorithm

In this Appendix, Algorithm 1 describes how ORF uses the forest-based kernel learner to perform kernel estimation, Algorithms 2 and 3 describe how to train the forest-based kernel learner. For the purpose of illustration, we simplify the problem set-up described in Section 4.1 by changing subscript of all variables/data from *it* to *i* throughout this section.

Algorithm 1 Orthogonal Random Forests

Input: data set D of size N, target feature vector x, base tree learner $\mathcal{L}_T$, the number of trees B, subsamples size s, and minimum observation per leaf r

Randomly partition D into two data sets D_1 and D_2 of equal size
procedure NUISANCE ESTIMATION
  for each tree $b \in 1, ..., B$ **do**
    Randomly sub-sample s observations from D_1 without replacement to form S_b
    Randomly partition S_b into two datasets of even size: S_b^1 and S_b^1
    Apply learner $\mathcal{L}_T$ to build a tree T_b using $Z \in S_b^1$ and $X \in S_b^2$ ▷ See Algorithm 2
    Obtain the leaf $L_b(x) \subseteq \mathcal{X}$ that contains the target feature x
    for each observation $i \in D_1$ **do**
      Compute the tree weight: $\omega_{ib} = \frac{1[(X_i \in L_b(x)) \wedge (Z_i \in S_b^2)]}{|L_b(x) \cap S_b^2|}$
  for each observation i in D_1 **do**
    Compute the importance weight: $\omega_i = \frac{1}{B} \sum_{b=1}^{B} \omega_{ib}$
  Estimate $\hat{q}$ and $\hat{g}$ using observations in D_1 and weights ω ▷ See Appendix C

procedure TARGET ESTIMATION
  for each tree $d \in 1, ..., B$ **do**
    Randomly sub-sample s observations from D_2 without replacement to form J_b
    Randomly partition J_b into two datasets of even size: J_d^1 and J_d^2
    Apply learner $\mathcal{L}_T$ to build a tree T_d using $Z \in J_d^1$ and $X \in J_d^2$ ▷ See Algorithm 2
    Obtain the leaf $L_d(x) \subseteq \mathcal{X}$ that contains the target feature x
    for each observation $i \in D_2$ **do**
      Compute the tree weight: $a_{id} = \frac{1[(X_i \in L_d(x)) \wedge (Z_i \in J_d^2)]}{|L_d(x) \cap J_d^2|}$
  for each observation i in D_2 **do**
    Compute the importance weight: $a_i = \frac{1}{B} \sum_{d=1}^{B} a_{id}$
    Compute the residuals: $\widetilde{Y}_i = Y_i - \hat{q}(X_i, W_i)$, and $\widetilde{T}_i = T_i - \hat{g}(X_i, W_i)$
  Estimate $\hat{\theta}$ by solving the kernel residualized regression problem

$$\hat{\theta} = \arg\min_{\theta} \sum_{i: Z_i \in D_2} a_i (\theta \widetilde{T}_i - \widetilde{Y}_i)^2 \tag{3}$$

Output: The conditional average treatment effect $\hat{\theta}(x)$ at x

Algorithm 2 Gradient Tree (Forest-based Kernel Learner)

Input: tree splitting sample S_1, estimation sample S_2, minimum leaf size r, minimum ratio at each split ρ, maximum number of splits (tree depth) τ, number of randomly selected candidate features m at each split, and number of proposed split points k

Initialize the tree with root node P_0 covering the whole feature space, and the Queue $Q = \{P_0\}$
Initialize the observations belong to the current node P_0 as $S^{P_0} = \{S_1, S_2\}$
Initialize the accumulated number of splits $accSplits = 0$
while $NotNull(P \leftarrow Q)$ & $accSplits \leq \tau$ **do**
    Perform the second-stage estimation of $\hat{\theta}_P$:
        Estimate nuisance function $\hat{q}_P$ and $\hat{g}_P$ for observations in S_1^p ▷ See Appendix C
        Calculate the residuals: $\widetilde{Y}_i = Y_i - \hat{q}_P(X_i, W_i)$, $\widetilde{T}_i = T_i - \hat{g}_P(X_i, W_i)$
        Solve the optimization problem: $\hat{\theta}_P = \arg\min_{\theta} \frac{1}{|S_1^P|} \sum_{Z_i \in S_1^P} \frac{1}{2}(\theta \widetilde{T}_i - \widetilde{Y}_i)^2$
    Compute the empirical Hessian: $A_P = -\sum_{i \in S_1^P} \widetilde{T}_i^2 / |S_1^P|$
    Generate k split points $Cand_P$ based on S_1^P ▷ See Algorithm 3
    for each candidate split point $C \in Cand_p$ with children nodes C_1 and C_2 **do**
        Obtain the remaining subsamples that belong to the children nodes, denoted as

$$S^{C_1} = \{S_1^{C_1}, S_2^{C_1}\}, \quad S^{C_2} = \{S_1^{C_2}, S_2^{C_2}\}$$

        if $min\{|S^{C_1}|, |S^{C_2}|\} \geq r$ & $min\{|S_1^{C_1}|/|S_1|, S_1^{C_2}/|S_1|, S_2^{C_1}/|S_2|, S_2^{C_2}/|S_2|\} \geq \rho$ **then**
            Compute the approximated estimates at each children node $j \in \{1,2\}$:

$$\tilde{\theta}_{C_j} = \hat{\theta}_P - \sum_{i \in C_j} A_P^{-1}(\widetilde{Y}_i - \hat{\theta}_P \widetilde{T}_i)\widetilde{T}_i$$

            Compute the proxy heterogeneity score at node C:

$$\tilde{\Delta}(C_1, C_2) = \sum_{j=1}^{2} \frac{1}{|C_j|}(\tilde{\theta}_{C_j} - \hat{\theta}_P)^2 = \sum_{j=1}^{2} \frac{1}{|C_j|}(\sum_{i \in C} A_P^{-1}(\widetilde{Y}_i - \hat{\theta}_P \widetilde{T}_i)\widetilde{T}_i)^2$$

        else
            Remove candidate C from the set $Cand_p$
    if $NotNull(Cand_p)$ **then**
        Select $C^* \in Cand_p$ that maximizes the proxy heterogeneity score
        Update the sample S^{S_j} as these belong to the children node C_j^* for $j \in \{1,2\}$
        Update $\{C_1^*, C_2^*\} \leftarrow Q$, and $accSplits = accSplits + 1$

Output: tree T_{p_0} with root node P_0

Algorithm 3 Proposal Splitting Points

Input: d-dimensional feature space data X that are relevant at node P, number of selected features m, and number of proposal split points k

1: Randomly pick m columns of X, denoted as Z
2: **for** each feature $j = 1, 2, ..., m$ **do**
3: Sort the possible value of Z_j, denoted as $V_j = \{v_1, v_2, ..., v_N\}$
4: Add $\min\{k, |V_j|\}$ randomly selected values from V_j to $Cand_p$
5: candidate split points $Cand_p$ at node P

B.1 Random Forest as a Weights Learner

We first describe the forest-based kernel learner, whose pseudo code is provided in Algorithm 2. This procedure relies on a few forest regularity conditions required for convergence. Please refer to Oprescu et al. (2018) for details. The random forest algorithm proceeds over B iterations. For each iteration b, it first randomly subsamples a subset S_b without replacement from the full dataset, which contains N observations $D = \{Z_i = (T_i, Y_i, W_i, X_i)\}$. The tree learner $\mathcal{L}_T$ then randomly partitions S_b into two subsets of equal sizes S_b^1, S_b^2. The learner uses S_b^1 to grow the tree (i.e., place splits in the tree), and uses the feature X_i in S_b^2 for stopping rules and balance maintenance. To ensure honesty of the tree (Athey et al., 2019), $\mathcal{L}_T$ does not select the splits using the controls and outcomes in S_b^2, which will be used for kernel estimation later. That is, in order to reduce prediction bias, forests use only the first half of the randomly split subsmaple for splitting, while the second half for populating the tree's leaf nodes: each new example is "pushed down" the tree and added to the leaf in which it falls. This is a key difference from classic random forests in which a single subsample is used both to choose a tree's splits and for the leaf node examples used in predictions.

Similar to GRF, the tree learner starts with a root node that contains the entire feature vector space, iteratively grows the tree by greedily selecting the splits that maximize certain splitting criterion, until certain stopping condition is met. However, the key modification to GRF's tree learner is the incorporation of orthogonal nuisance estimation (i.e., DML) in tree splitting criterion. At each internal node P, $\mathcal{L}_T$ will perform the following two-stage estimation for $\hat{\theta}_P$ over the set of observations in S_b^1 that reach P (denoted as $P \cap S_b^1$). First, the algorithm estimates the nuisance functions q_0 and g_0, using any machine learning methods, such as lasso, random forest, and DNNs.

Second, the algorithm uses the estimated nuisance functions, denoted as $\hat{q}_P$ and $\hat{g}_P$, to residualize the outcome and treatment for each observation that reaches P:

$$\widetilde{Y}_i = Y_i - \hat{q}_P(X_i, W_i), \widetilde{T}_i = T_i - \hat{g}_P(X_i, W_i) \tag{4}$$

The residualization step partials out the confounding effects from X_i and W_i in each observation, so these residuals represent exogenous variation that can be used to identify a valid causal effect in a second stage regression (Chernozhukov et al., 2017). Then, the second-stage estimation is obtained by regressing the residualized $\widetilde{Y}$ on the residualized $\widetilde{X}$:

$$\hat{\theta}_P = \arg\min_{\theta} \frac{1}{|P \cap \mathcal{S}_b^1|} \sum_{Z_i \in P \cap S_b^1} \frac{1}{2}(\theta \widetilde{T}_i - \widetilde{Y}_i)^2 \tag{5}$$

Given the estimate $\hat{\theta}_P$ at parent node P, we would like to split the parent node into two children C_1 and C_2. Similar to GRF, we seek the split that maximize heterogeneity in the treatment effect estimates across children nodes. Namely, if we perform the same two-stage estimation separately at each child, the new estimates $\hat{\theta}_{C_1}$ and $\hat{\theta}_{C_2}$ will maximize the following heterogeneity score:

$$\Delta(C_1, C_2) = \sum_{j=1}^{2} \frac{1}{|C_j|}(\hat{\theta}_{C_j} - \hat{\theta}_P)^2 \tag{6}$$

However, performing the two-stage estimation of $\hat{\theta}_{C_1}$ and $\hat{\theta}_{C_2}$ for all possible splits is too computationally expensive. Instead, we will approximate these estimates by taking a Newton Step from a parent node estimate $\hat{\theta}_P$. Let us first write the loss function of the second-stage residualized least square regression, $\mathcal{L}_{res}(\theta, Z_i) = \frac{1}{2}(\theta \widetilde{T}_i - \widetilde{Y}_i)^2$. Then for each observation Z_i that belongs to node P, the gradient is $\nabla_\theta \mathcal{L}_{res}(\hat{\theta}_P, Z_i) = (\widetilde{Y}_i - \hat{\theta}_P \widetilde{T}_i)\widetilde{T}_i$. The empirical Hessian over the observations in parent node P,

$$A_P = \nabla^2_\theta\left(\frac{\sum_{i \in P \cap S_b^1} \mathcal{L}_{res}(\hat{\theta}_P, Z_i)}{|P \cap S_b^1|}\right) = -\frac{\sum_{i \in P \cap S_b^1} \widetilde{T}_i^2}{|P \cap S_b^1|} \tag{7}$$

Then, for any child node C given by a candidate split, we could derive a proxy estimate $\tilde{\theta}_C$ by

taking a Newton step with data in node C,

$$\tilde{\theta}_C = \hat{\theta}_P - \sum_{i \in C} A_P^{-1}(\widetilde{Y}_i - \hat{\theta}_P \widetilde{T}_i)\widetilde{T}_i \tag{8}$$

Given the estimates $\tilde{\theta}_C$, we compute the proxy heterogeneity score:

$$\tilde{\Delta}(C_1, C_2) = \sum_{j=1}^{2} \frac{1}{|C_j|}(\tilde{\theta}_{C_j} - \hat{\theta}_P)^2 \tag{9}$$

We grow the tree by greedily selecting the splits that maximize this proxy heterogeneity score, until certain stopping condition is met. After the tree is grown, the set of observations in S_b^2 is "pushed down" to determine which leaf it falls in, while the learner ensures that the leaf contains at least r observations in S_b^2. For each tree indexed $b \in [B]$, let $L_b(x)$ as the set of training example falling in the same "leaf" as the test point x. Then, the tree weight assigned to each observation i is computed as

$$K(X_i, x, (S_b^1, S_b^2)) = \frac{1[(X_i \in L_b(x)) \wedge (Z_i \in \mathcal{S}_b^2)]}{|L_b(x) \cap S_b^2|} \tag{10}$$

Finally, we assign the importance weight for each observation by averaging over all the B trees:

$$K(X_i, x) = \frac{1}{B} \sum_{b=1}^{B} K(X_i, x, (S_b^1, S_b^2)) \tag{11}$$

For each observation, the importance weight measures the similarity between its feature vector(s) X_i and the target feature point x. It will be used for the next kernel two-stage estimation.

B.2 Weighted Two-Stage Estimation

We now describe how ORF uses the forest-based kernel learner to perform kernel estimation, which is summarized in Algorithm 1. The algorithm first partitions the input dataset D into D_1 and D_2, and runs the forest-based weight learner on D_1 and D_2 to derive two forests and two sets of similarity weights. We use $\{\omega_{ib}\}_{b=1}^{B}$ and ω_i to denote the tree weights and importance weights over the observations in D_1, and use $\{a_{id}\}_{d=1}^{B}$ and a_i to denote the tree weights and importance weights over the observations in D_2. We will utilize these weights to perform a kernel two-stage

estimation.

In the first stage, we pass the set of weights ω_i to observations in D_1 to estimate the local nuisance functions q_0 and g_0 at $X = x$, denoted as $\hat{q}$ and $\hat{g}$. As suggested in Chernozhukov et al. (2017), a broad array of sophisticated machine learning methods can be applied here, such as lasso, ridge, DNNs, random forests, and various hybrids and ensembles of these methods. Then, following the cross-fitting approach in DML, we compute the residualized $\widetilde{Y}_i$ and $\widetilde{T}_i$ for each observation $Z_i = (T_i, Y_i, W_i, X_i) \in D_2$: $\widetilde{Y}_i = Y_i - \hat{q}(X_i, W_i)$, and $\widetilde{T}_i = T_i - \hat{g}(X_i, W_i)$. We can obtain the estimates of $\theta_0(x)$ by performing a kernel linear regression over observations in D_2:

$$\hat{\theta} = \arg\min_{\theta} \sum_{i:Z_i \in D_2} a_i (\theta \widetilde{T}_i - \widetilde{Y}_i)^2 \tag{12}$$

Note that GRF proposed in Athey et al. (2019) also recommends such a residual on residual regression approach in their empirical study, which is referred to as "local centering," albeit without theoretical analysis. The key difference between "local centering" and ORF is that ORF residualizes locally around test point x, as opposed to performing an overall residualization step and then calling the GRF algorithm on the residuals. Oprescu et al. (2018) have shown that ORF can better minimize non-local mean squared error and improve the final performance of the treatment effect estimates. The reason is that ORF requires that the nuisance estimator achieve a good estimation rate only around the target point x and residualizing locally seems more appropriate than running a global nuisance estimation, which would typically minimize a non-local mean squared error. while it does add some extra computational cost as a separate first stage model needs to be fitted for each target test point.

B.3 SDNNs for Nuisance Estimation

Here we describe the underlying math when estimating the nuisance functions using SDNNs. Basically, we train the SDNNs by minimizing a loss function which is defined as weighted sum-of-square errors between the true value and model estimates. The weight a_i for each observation in D_1 is obtained by running the same tree learner over B random subsamples (without replacement).

To mitigate overfitting, we employ weight decay using $\mathcal{L}_2$ regularization, a widely-used technique for regularizing neural network models. Therefore, we can estimate the nuisance functions by minimizing the loss function of our SDNNs defined in Equations (2):

$$\hat{q} = \arg\min_{q} \sum_{i \in D_1} a_i (Y_{it} - q_1(\chi^p) - V_{it}^1 \Gamma^1)^2 + \lambda_q ||q||^2 \tag{13}$$

$$\hat{g} = \arg\min_{g} \sum_{i \in D_1} a_i (T_{it} - g_1(\chi^p) - V_{it}^1 \Gamma^1)^2 + \lambda_g ||g||^2 \tag{14}$$

where, $\lambda \geq 0$ are the regularization parameters. Larger values tend to shrink the parameters in the parametric and nonparametric components toward zero.

C Enabling SDNNs in ORF

A team of researchers at Microsoft Research developed a Python package, *EconML*, for applying machine learning and causal inference method.[1] *EconML* contains a module for implementing ORF, built on the *sklearn* library for performing nuisance estimation. *sklearn* provides a wide range of parametric and nonparametric methods, such as lasso and standard neural networks architecture, but it doesn't allow both simultaneously. That is, one can implement ORF using package *EconML*, only when the nuisance function is assumed to be either completely parametric or having standard neural network architecture. Therefore, to enable flexible nonparametric and parametric form of nuisance function simultaneously in ORF, we contribute to the main ORF library a PyTorch extension. In this section, we first describe step-by-step how one can build a SDNNs in Python, followed by how to enable semi-parametric estimation of the nuisance functions for ORF in Python.

C.1 Building a SDNNs in Python

Among many libraries that can be used for deep learning, we adopt PyTorch, primarily developed by Facebook's AI Research lab, that offers much flexibility to construct neural network models using built-in functions and classes. We recommend installing PyTorch with Python 3.6 or

[1] See https://econml.azurewebsites.net/spec/spec.html for detailed documentation.

greater. One will also need to install Skorch, a Python-based library for PyTorch that provides full Skicit-Learn (sklearn) compatibility.[2] The clean sklearn interface offered by Skorch could largely simplify the model training and evaluation process.

Our SDNNs is customized specifically to estimate the nuisance function g_0 and q_0 that capture the confounding effects of covariates on price and demand, respectively. It has two sets of input variables, χ^p and χ^n, which enter the parametric and nonparametric components of the model respectively. This requires a multiple-input feedforward neural network that takes χ^n and χ^p in separate branches and let only χ^n go through the nonlinear operations. The full code for building and training this SDNNs is provided in below. We explain each block in the following sections. For researchers and practitioners, it is easy to extend this semi-parametric framework and customize their own DNNs by harnessing the full power of PyTorch.

[2]For installation of PyTorch, see https://pytorch.org/get-started/locally/. For installation of Skorch, see https://skorch.readthedocs.io/en/latest/index.html. See Skorch API documentation for a complete list of arguments and methods. Skicit-Learn is an open source Python library that implements a wide range of machine learning using a unified interface. It is by far the easiest and most widely-used tools for machine learning applications.

```
import numpy as np
from torch import nn
import torch.nn.functional as F

from skorch import NeuralNetRegressor

X = torch.from_numpy(X).float()
Y = torch.from_numpy(Y).float()

class MyModule(torch.nn.Module):
    def __init__(self):
        super(RegressorModule, self).__init__()
        self.dense0 = nn.Linear(260, 100)
        self.dense1 = nn.Linear(100, 100)
        self.output = nn.Linear(100 +  1458, 1)

    def forward(self, X, **Kwargs):
        X1, X2 = X[:, :260], X[:, 260:]
        X1 = F.relu(self.dense0(X1), inplace=True)
        X1 = F.relu(self.dense1(X1), inplace=True)
        X = torch.cat([X1, X2], 1)
        X = self.output(X)
        return X

net = NeuralNetRegressor(
    module=MyModule,
    criterion = torch.nn.MSELoss,
    optimizer = torch.optim.Adam,
    lr = 0.001
    # Shuffle training data on each epoch
    iterator_train__shuffle=True,
)
net.fit(X, Y)
y_pred = net.predict(X)
```

C.1.1 Data Preparation

Before building a model, we need to convert the input(s) and target(s) to PyTorch variables. Pytorch uses tensor to store and operate on n-dimensional rectangular arrays of numbers, which are similar to NumPy arrays but can also be operated on GPU and provides automatic differentiation that efficiently computes the gradient w.r.t. some parameters. Suppose the input data `X` and target `Y` are in Numpy arrays, we could convert them to PyTorch tensors using the following code:

```
X = torch.from_numpy(X).float().to(device)
```

```
Y = torch.from_numpy(Y).float().to(device)
```

The command `.to(device)` send data to the chosen device (CPU or GPU).

C.1.2 Defining a SDNNs

We now illustrate how to implement a SDNNs defined in Section 4.3 using PyTorch API. We first create a class for our own neural network `MyModule` by subclassing `torch.nn.Module`. The `torch.nn.Module` is the base class in PyTorch that contains the building blocks needed to create all sorts of neural network architectures. Through class inheritance, we are able to use all of the functionality of the `nn.Module` base class, but still need to define the hyperparameters (e.g., the number of layers and the number of nodes in each layer) for our own network.[3] The actual code used to define our model looks like this:

```
import torch.nn as nn
import torch.nn.functional as F
class MyModule(nn.Module):
    def __init__(self):
        super(MyModule, self).__init__()
        self.dense0 =nn.Linear(260, 100)
        self.dense1 =nn.Linear(100, 100)
        self.output =nn.Linear(100 + 1458, 1)
```

In the first line of the class initialization (`def __init__(self):`), we have the required Python `super()` function, which created an instance of the base `nn.Module` class. The following three lines are where we create our fully connected layers, which are represented by the `nn.Linear` object, with the first argument being the number of input nodes in layer l and the second argument being the number of outputs. The first layer `dense0` takes 260 inputs (the dimensionality and the shape of input data must match) and produces 100 outputs, followed by a 100-dim to 100-dim hidden layer `dense1`. These two layers consist a branch that will be used to operate over the set of input variables χ^n. Then we have the top layer, which accepts the derived 100 nodes from the hidden layer `dense1` along with another set of input variables (corresponds to χ^p), and outputs a

[3] Note that it is mandatory to subclass `torch.nn.Module` when you create a class for your own network. The name of your customized class can be anything.

single value - the final predication.[4]

After setting up the "skeleton" of our network architecture, the next step is to define how the data flows through the network and performs the computation. We do this by defining a `forward()` method in our class. For the `forward()` method, we supply the input data `X`, a Tensor contains all of our input variables, as the primary arguments. In the first line of the `forward()` class, we split `X` into `X1` and `X2`, corresponding to χ^n and χ^p respectively in Equation (2). We then feed `X1` into our first layer `dense0` and apply the activation function ReLU to the nodes in this layer using `F.relu()`. It is important to note that each layer itself is simply a linear combination of inputs and nonlinearity is added via the activation function. The transformed value is reassigned to `X1` and continuously fed to the next layer `dense1`, followed by ReLU activation. The derived nodes from `dense1` will then be combined with the auxiliary input `X2` at the `output` layer to obtain our final prediction. Note that, without nonlinear activation attached, this top layer is simply a linear model in `X2` and the derived nodes. The auxiliary input `X2` thus falls into the linear component of the model.

```
def forward(self, X, **Kwargs):
    X1, X2 = X[ :, :260], X[ :, 260: ]
    X1 = F.relu(self.dense0(X1), inplace=True)
    X1 = F.relu(self.dense1(X1), inplace=True)
    X = torch.cat([X1, X2], 1)
    X = self.output(X)

    return X
```

C.1.3 Training a SDNNs

Now it is time to train the neural network. The original training process in PyTorch requires a fit loop, which typically generates a lot boilerplate code and is incompatible with ORF package. To ease the training process, we leverage the sklearn compatibility provided by Skorch. Skorch offers two main classes, `NeuralNetRegressor` and `NeuralNetClassifier` (for regression and classi-

[4]The dimension of input in the first layer is 260, which equals the number of covariates that assume to have nonlinear relationships with treatment/outcome in Table 1 plus the 1-d target feature (e.g., the number of days to scheduled live day). The input in the top layer is the vector of 1,458 creator, category, and time fixed effects.

fication tasks respectively) that wrap the PyTorch `Module` while providing an training/evaluation interface similar as sklearn. Since our analysis focuses on regression task, we pass our Pytorch model to `NeuralNetRegressor`, in conjunction with a PyTorch (-compatible) criterion such as the loss function and optimizer. A sample code is provided in below:

```
net = NeuralNetRegressor(
      module=MyModule,
      criterion=torch.nn.MSELoss,
      optimizer=torch.optim.Adam,
      lr=0.001, ... ,
)
net.fit(X, y)

y_pred = net.predict(X_test)
```

We can see that `NeuralNetRegressor` wraps the PyTorch `Module` in an sklearn interface. The argument `modeule` is where we pass our PyTorch `Module`. The argument `criterion` specifies the loss function and is set to PyTorch `MSELoss` by default when you use the `NeuralNetRegressor`. More arguments, such as the optimizer and learning rate, etc., can be specified as well. After defining relevant arguments and methods to the wrapper, we could call `fit()` and `predict()`, as with an sklearn estimator. To evaluate the neural network model, we could call `score()` method that returns the coefficient of determination R^2 of the prediction for regressors.

C.2 Execution

To accommodate and streamline customized neural network usage in ORF, we contribute to the main ORF library a PyTorch extension. Our extension module addresses two issues. First, since the main ORF module generally takes Numpy Arrays while neural networks need to be feed with PyTorch Tensors, we perform format transformation when data flowing through the network. Second, although the Skorch wrapper offers `fit` and `predict` that looks like sklearn API, it is not fully sklearn-compatible in terms of inputs and outputs dimensionality/shape. We hence reshape input/output flows and apply dimensionality check at each key algorithmic block to guarantee full sklean compatibility of the customized neural network. Note that Skorch `fit()` does not support passing `sample_weight` directly as arguments to `fit` calls. Instead, we can the `sample_weight`

with `X` as a dictionary to the `forward` method and then define your own loss function in Skorch.

The ORF package provides a helper class `WeightedModelWrapper` that enables sample weights functionality, by wrapping any class that supports `fit` and `predict`. Below is an example code of applying the wrapper on SDNNs:

```
model_T_final=WeightedModelWrapper(
              model_instance=NeuralNetRegressor(MyModule,
              lr=0.001,
              optimizer=torch.optim.Adam,
              optimizer__weight_decay=0.0001),
              sample_type="sampled")
```

The first argument `model_instance` is where we pass our Skorch (-wrapped) model that requires weights. The second argument `sample_type` specifies method for adding weights to the model. For nonlinear model (e.g., neural network and random forest), this argument should be set to `sampled` method, which samples the training set according to the normalized weights and creates a dataset larger than the original. Set this argument to `weighted` for linear regression models where the weights can be incorporated in the matrix multiplication.

After enabling sample weights functionality for our SDNNs, we could now run the full ORF algorithm using the example code below. Since our treatment (i.e., price) is continuous, we call the class `ContinuousTreatmentOrthoForest` to perform ORF estimator. In the first two lines of the class, we specify hyperparameters that controls the tree-growing procedure for the forest kernel learner, in line with description in Section B.1.[5] The following two arguments `model_T` and `model_Y` (for treatment and outcome respectively) are where we pass our semi-parametric DNN to the random forest kernel learner. Note that `model_T` and `model_Y` are used to perform first-stage estimation when placing splits in the tree and no weights will be involved at this tree-growing

[5]The argument `n_trees` controls how many trees are grown in the random forest. Generally, obtaining high-quality confidence intervals requires growing a large number of trees. The `min_leaf_size` relates to the minimum size a leaf node is allowed to have. Given this parameter, if a node reaches too small of a size during splitting, it will not be split further. The maximum number of splits are specified in the `max_depth` argument. The `subsample_ratio` is a number in range $(0, 1]$ that controls the fraction of samples should be used in growing each tree. As noted in the "honesty" criterion, the fractional subsample will be further split into halves.

stage. The next two arguments `model_T_final` and `model_Y_final` are where we pass the semi-parametric DNN that supports sample weights derived from the random forest kernel leaner. After defining relevant arguments and method, we could call `fit()` to start training. The final treatment effects at every test points can be produced by calling `const_marginal_effect(X_test)`. One can generate confidence intervals to wrap the derived estimates using bootstrap, which has been proven in theory to be asymptotically valid (Oprescu et al., 2018).

```
from econml.ortho_forest import ContinuousTreatmentOrthoForest
from econml.ortho_forest import WeightedModelWrapper

# Specify hyperparameters
est = ContinuousTreatmentOrthoForest(

      n_trees=500, min_leaf_size=100,
      max_depth=20, subsample_ratio=0.5,

      model_T=NeuralNetRegressor(MyModule,
      optimizer=torch.optim.Adam,
      lr=0.001, optimizer__weight_decay=0.0001),

      model_Y=NeuralNetRegressor(MyModule,
      optimizer=torch.optim.Adam,
      lr=0.001, optimizer__weight_decay=0.0001),

      model_T_final=WeightedModelWrapper(NeuralNetRegressor(MyModule,
      optimizer=torch.optim.Adam, lr=0.001, optimizer__weight_decay=0.0001),
      sample_type="sampled")

      model_Y_final=WeightedModelWrapper(NeuralNetRegressor(MyModule,
      optimizer=torch.optim.Adam, lr=0.001, optimizer__weight_decay=0.0001),
      sample_type="sampled")
)

est.fit(Y, T, X, W)

# Specify test points on the target feature X (time distance to live day)
X_test = np.linspace(-14, 14, 29).reshape(-1, 1)
treatment_effects = est.const_marginal_effect(X_test)
```

D Applying Generalized ORF in Zhihu Live Data

D.1 Parameterization

The random forest kernal learner requires us to select three tunning parameters: the number of trees, the minimum number of observations in each leaf, and the subsample size. In the absence of formal criteria to guide our choices, we used a large number of trees because more trees reduce the Monte Carlo error introduced by subsampling. We found moving from 100 to 500 improved the stability of estimates across samples, but moving from 500 to above does not improve stability significantly. Increasing the minimum number of observations in each leaf trades off bias and variance; bigger leaves make results more consistent across different samples but predict less heterogeneity. Smaller subsamples reduce dependency across trees but increase the variance of each estimate, though we find that increasing subsamples made little difference in our application (Davis and Heller, 2017).

Note that there is no formal rule to determine the number of test points. We choose these test points in order to obtain a reasonable tradeoff between granularity (of the estimates) and computational load.

D.2 Specifying Nuisance Functions Using Different Methods

To illustrate the potential gain of using SDNNs to specify nuisance functions relatively to the existing common approaches (i.e., lasso and standard DNNs), we compare their performance in fitting observed (log-transformed) price and demand along a few dimensions, respectively. The lasso includes all the covariates in Table 1 linearly in the nuisance functions. The DNNs is implemented with the MLPregressor provided in sklearn library, which has a similar architecture to the SDNNs but without the parametric component. We randomly split all the Live-day observations into a in-sample training data set (80%) and a out-sample testing data set (20%).

Table A1 reports the performance results. One can see that the neural neural network-based methods achieve much higher prediction accuracy than lasso in both demand and price equation. The out-sample R^2 and MSE both improve by at least 27%. This suggests that, as we expected,

Table A1: Fitting Comparison across Methods

		In-Sample		Out-Sample		
	Model	R^2	MSE	R^2	MSE	Time(ms)
Demand Equation	Lasso	0.694	0.500	0.689	0.502	567
	DNNs	0.807	0.315	0.818	0.294	292
	SDNNs	**0.755**	**0.400**	**0.786**	**0.355**	**211**
Price Equation	Lasso	0.783	0.093	0.784	0.092	517
	DNNs	0.997	0.002	0.994	0.002	528
	SDNNs	**0.978**	**0.010**	**0.980**	**0.006**	**391**

Note: The average value of log price and log daily sales is 2.89 and 1.50, respectively.

linear functional form cannot fully recover the complex relationships between the large number of covariates and price/demand. Within the two neural network models, DNN achieves slightly higher performance than SDNNs, but it is prone to overfitting (as shown by the discrepancy between the in-sample and out-sample scores of the price equation) and involves higher computational costs (as shown by the time cost on the test set). This is because the DNN naively dump all covariates (including a large number of vector for capturing individual and time fixed effects) into the neural network blackbox, which accommodate non-linear combinations of all covariates in a total flexible manner. In contrast, the SDNN can better balance the trade-offs between prediction accuracy and over-fitting, while is computationally more efficient.

E Supplementary Analysis for Main Results

E.1 Price Comparison across Comparable Livestreaming Markets

To the best of our knowledge, Zhihu Live is the largest livestreaming platform of its type during our study period. Its competing platforms were quite small during. We identify four relatively similar platforms in China that may compete with Zhihu Live. In summer 2016, we surveyed the prices of events on these five platforms, by randomly scarping 100 paid events on each platform. The descriptive statistics of these prices are shown in Table A2. We find that the price points for events hosted on Zhihu Live are typically higher than those of comparable events on competing platforms.

Table A2: Price Comparison

Platform	Mean	Median	Std. Dev.
Zhihu Live	**12.81**	**13.82**	**9.50**
Himalaya FM	5.95	4.06	5.56
Dedao	3.52	2.68	2.58
Lychee Live	12.87	4.95	29.43
Qianliao Live	3.36	2.91	3.01

Note: To make sure that prices are comparable across platforms, we anchored the unit to RMB/30min.

E.2 Robustness Checks

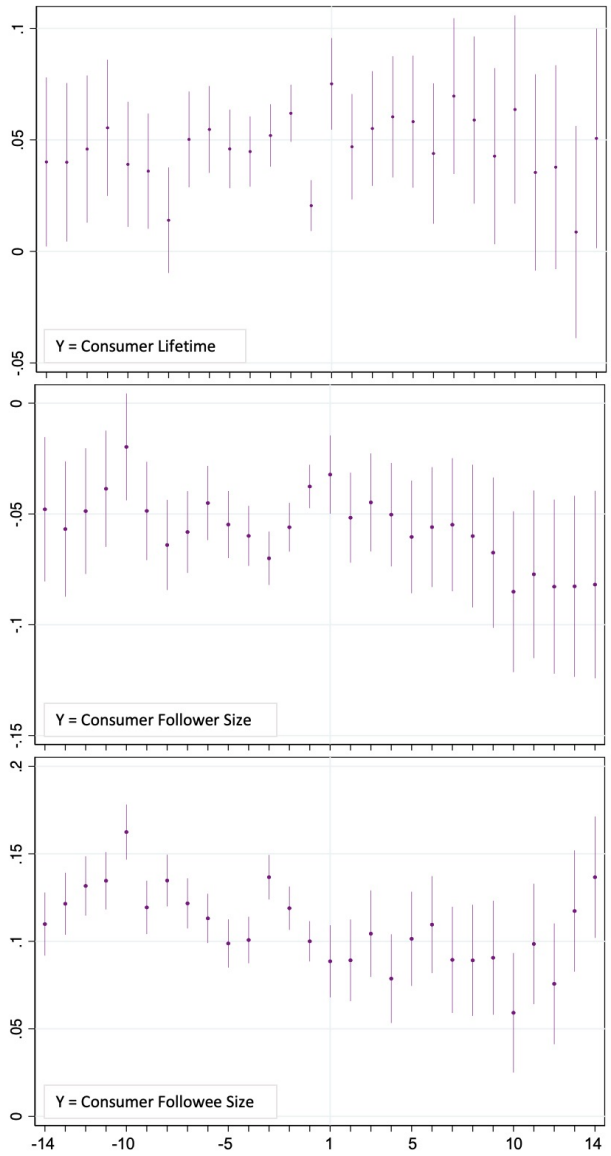


Figure A4: Temporal Distance Fixed Effects in Consumer Characteristics Regressions

Note: The figures depict the point estimates in the vector of temporal distance fixed effects and the associated 95% confidence intervals. For each regression, the unit of analysis is individual transaction and the number of observations 1,221,103.

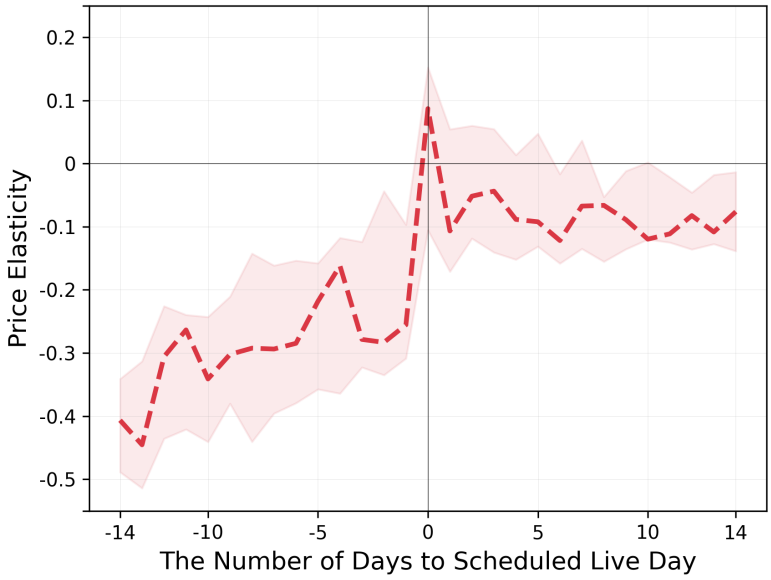


Figure A5: Price Elasticity over Time to Scheduled Live Day (Excluding Extreme Consumers)

Note: The shaded region depicts 95% confidence interval via 100 bootstrap resamples.