The Role of “Live” in Livestreaming Market: Evidence Using Orthogonal Random Forest

Ziwei Cong, Jia Liu, Puneet Manchanda*

June 28, 2021

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

Abstract

This paper examines price elasticity of demand for paid live content in a large livestreaming platform, where the recorded version of each live event is also available for purchase after being aired. We conduct causal estimation using a generalized Orthogonal Random Forest (ORF) framework which can deliver heterogeneous treatment effects in the presence of high-dimensional confounders whose relationships with the treatment policy (i.e., price) are complex but partially known. Our main result shows significant temporal dynamics in price responsiveness of demand over temporal distance to the scheduled live day and after. Specifically, demand gradually becomes less price sensitive over time to the scheduled live day and is inelastic on the live day. Over the post-stream period, demand is still price elastic, but much less sensitive than the pre-period. We then provide correlational evidence for the mechanisms driving the results. First, we attribute the inelastic demand on the live day to the possibility of real-time interaction with the creator during the livestreaming. Second, uncertainty reduction plays a role for the gradually decreasing trend in price elasticity over the pre-period and the much less sensitive demand over the post-period.

Keywords: Livestreaming, Pricing, Knowledge Goods, Heterogeneous Treatment Effect, Random Forest, Double Machine Learning, Semi-parametric Deep Neural Networks

*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 participants at the 2020 AIML Conference and the 2021 Marketing Science Conference for feeback.

1 Introduction

Livestreaming is the practice of streaming online content for consumption in real time. This is in contrast to typical influencer marketing content that is available in recorded form. 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 led to it being widely adopted in various domains (e.g., gaming, education and e-commerce) in recent years, resulting in strong growth. For instance, Twitch, the leading US livestreaming platform (focused on 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). China is the the dominant market for livestreaming currently (at $16.3 billion (Statista, 2020)) with the US the next biggest (estimated to be $11 billion by the end of 2021 (Thomas and Palmer, 2021)). Not surprisingly, given the novelty of livestreaming, there is virtually no academic work that examines how consumers value the "live" component of livestreaming (relative to recorded content).

In this paper, we use data from one of the largest livestreaming platforms in China (Zhihu Live) to shed light on how consumers value live (and recorded) content. We exploit the fact that the platform also makes live content available in recorded form, allowing us to contrast valuation of the content before, on the day of, and after the livestream. We summarize the value of both the live and recorded content via the use of the estimated price elasticity of demand. To the best of our knowledge, our paper is the first attempt to empirically examine consumer valuation of live content over the entire content life-cycle.

Zhihu Live targets creators who want to monetize their expertise or establish their own business via livestreaming. The creators on Zhihu Live typically provide content (denoted as "Live" in the

[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.

rest of the paper) on “serious,” i.e., non-entertainment, topics such as history, 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, the creator gives a one to two hour live talk, interacting only with the customers who paid for the event before it sold out via written, voice, or picture messages in a virtual chat room.[2] After the Live 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 (used by Facebook, Vimeo, OnlyFans etc.) livestreaming pay-per-view (PPV) pricing model. Under this model, each creator sets a common price for both her/his live and recorded content (with some loose guidance from the platform) with the consumer paying this fixed price one-time.[3] This variation in pricing across different Lives (including variation in creator, audience and platform characteristics) enables us to summarize the value of the live versus the recorded content via the price elasticity of demand.

However, 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 are likely to be confounded with a large number of factors (e.g., content quality, creator credibility) in potentially complex and partially unknown ways. This results in a high-dimensional setting where the number of variables could be 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 (how it varies relative to the day of Live) is particularly important.

In order to address the above challenges, we develop 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 unknown. Specifically, our framework generalizes Orthogonal Random Forest (ORF) (Oprescu et al. (2018)) via the use of

[2]Customers who bought a seat after the event sells out can still watch it in real-time, but cannot interact with the creator.

[3]In livestreaming, PPV 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.

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 linear 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 a (partially unknown) non-linear relationship with the treatment and outcome. The use of SDNNs to estimate the nuisance functions is particularly 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.

The results show considerable dynamics in price sensitivity relative to the scheduled live day. Specifically, demand gradually becomes less price sensitive approaching the live day, is inelastic on the live day and becomes price sensitive again throughout the post-period (though, on average, less sensitive than in the pre-livestream period). This pattern suggests that while the live part of the content is valuable (as can be seen by the inelastic demand on the day of live), there is considerable value in the content that is not centered in its live attribute. We then provide suggestive evidence for the mechanisms that drive these results via a series of analyses. First, we attribute the inelastic demand on the scheduled live day to the availability of immediate real-time interaction with the creator. Second, the results conclude that the best fitting mechanism explaining the less sensitive post-period demand is 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.

This paper makes several contributions. First, it is one of the early studies investigating the growing phenomenon of livestreaming, focusing on how much and why consumers value live (and recorded) content. Second, we find that while a live part of a livestream is non-replicable, its has value beyond the live part. This is contrary to popular belief that livestreaming content has no residual value once the event is over (Gomes, 2018; John, 2020). To the best of our knowledge, this has not been shown before. Third, these results are likely to be valuable to both streamers/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.[4] In addition, marketing efforts to reduce uncertainty about upcoming events could also be beneficial for all parties (streamers, platform, consumers). Finally, our methodological approach shows how ORF can be generalized via the use of SDNNs for nuisance estimation. We believe that this framework is likely to be very useful in real-world marketing and economics settings, especially with high-dimensional covariates that impact the treatment in complex and partially unknown ways.

The rest of paper is organized as follows. Section 2 describes our empirical context and data. Section 3 conducts descriptive regression analysis. In section 4, we introduce the model setup, ORF algorithm, and how we allow SDNNs for nuisance estimation. In Section 5, we apply our proposed generalized ORF to obtain our main findings on the temporal dynamics in price sensitivity. In Section 6, we explore reasons that explain the main findings. Section 7 concludes with a discussion.

2 Research Setting and Data

2.1 Research Setting: Zhihu Live

Our setting is the Chinese platform Zhihu Live. Zhihu Live is the livestreaming service of Zhihu.com, launched on May 14, 2016.[5] By Dec 2020, Zhihu.com had 43.1 million cumulative content creators, who had contributed 315.3 million questions and answers (Hu et al., 2017). Zhihu.com went public in the US 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 the Appendix C for

[4]As such, many livestreaming platforms (e.g., Instagram Live, YouNow) only support live broadcasts and do not offer build-in recording functions. Other platforms (e.g., Facebook Live, Periscope), while offering built-in recording and replay functions, generally devote very few resources towards managing recorded versions of live events.

[5]Zhihu.com is the earliest and largest knowledge sharing platform in China, starting as a Q&A community (similar to Quora.com) in 2011. Users on Zhihu.com 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.

illustrative examples.

Zhihu Live targets Zhihu.com creators who want to monetize their expertise via livestreaming content to a paying audience. By Dec 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.

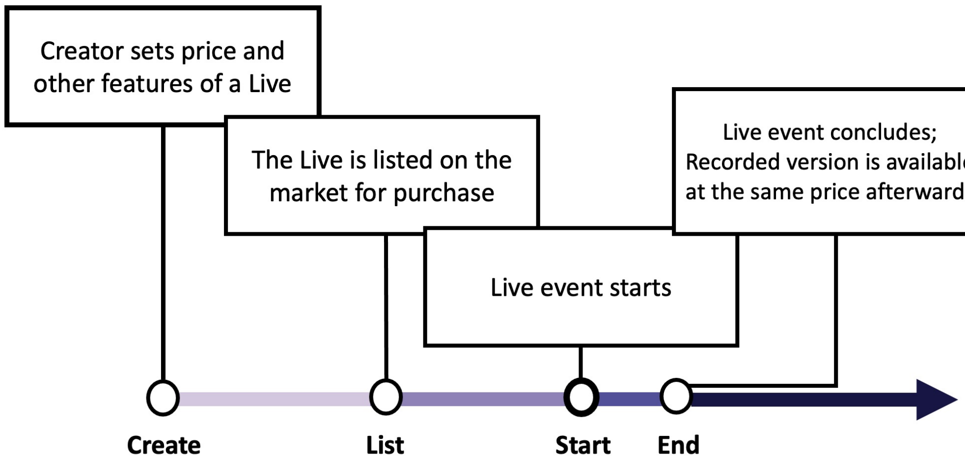

Creator sets price and other features of a Live
The Live is listed on the market for purchase
Live event starts
Live event concludes; Recorded version is available at the same price afterwards
Create
List
Start
End

Figure 1: Life-Cyle of a Live

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 consumers who are allowed to interact with the creators in real-time during the Live). During our study period, the Live revenue is split between the creator and the platform 70:30. 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. During this period, consumers become aware of upcoming Lives through the webpage of Live market, which has a section that displays news about upcoming Lives. Note that once a Live enters the market, 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.[6]

During the livestream, the creator will give a 1-2 hour live talk on the prepared content. During this time period, the creator can interact with his/her paying audience via text, voice, or picture messages in a virtual chat room. An illustrative example is provided in Figure A3 of Appendix C. Note that only consumers who purchase the Live *before* seats sell out can raise questions and leave comments during the Live. Other consumers can still view the event in real-time but cannot interact with the creator. After the live concludes, consumers can provide feedback (a rating on a 5-point scale along with free text comments). The recording of the live is then made available for purchase (viewing) at the same (full) 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 contrast price elasticity of demand 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 the characteristics of the purchased Live (such as creator ID, topical category, full price, seat limit, posting time, starting time, talk duration, and rating if available). 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 (i.e., question, answer, and article) contribution, content (i.e., topic and question) subscription, and social activities (i.e., following others or being followed). We also know different types of individual feedback (e.g., upvote, downvote, thank, unhelpful) that a user received for each answer he/she has contributed. These observed creator characteristics are related to a creator's reputation in the free Q&A community.

[6]In a few cases, we observe that the platform lowers the price, by running platform-level promotions. These are typically run over weekends and public holidays. We discuss this in detail in Section 2.2. In addition to this, 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.

This reputation is expected to be an important factor for setting the full price of the creator's Live and also for influencing the realized demand for the Live (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 platform was relatively stable with no major changes in the operating policy. When estimating price elasticity of demand over Live life-cycle using ORF, 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.

2.3 Covariates

The internal data provided by Zhihu allows us to observe rich characteristics of all parties (creators, products, consumers, platform) that may affect full price and realized demand. This helps us to get close to the unconfoundedness assumption for causal identification. In total, there are 1,700 covariates, among which 1,458 variables are fixed effects (for creator, Live category, and seasonality) and the remaining are observed characteristics. We categorize these covariates into four categories: Live, creator, Live market, and Q&A platform characteristics. Table 1 summarizes all covariates.

The characteristics of Lives include seat limit, the number of days from listing to the scheduled live day, rating (if available), whether the Live was advertised by the platform, seasonality of the livestream day, and creator and topic fixed effects. We also have time-varying Live characteristics observed for each calendar (transaction) day, including the cumulative sales of the Live, whether seats are still available, percentage of discount, the vector of day-distance relative to Live fixed effects, seasonality of the calendar (transaction) day. Prior literature suggests that characteristics of products or services are likely to account for a substantial proportion of demand signals used for pricing decisions (Gibbs et al., 2018).

Creators are mainly characterized by their past activities on the Live market and the Q&A

Table 1: Full List of Control Variables

Live-Specific Characteristics

- Fixed variables: *Creator indicators*; *Topic indicators*; Whether the Live belongs to "hard" topics; The num. of days from listing to livestream; Seats limit; Rating (if available); Whether the Live is advertised by the platform; Whether Live is held on public holiday; *Live week indicators*; *Live day-of-week indicators*
- Time-varying variables: The cum. sales of a Live; Whether seats are still available; Percentage discount; Dummy indicators for day-distance relative to live streaming

Creator-Specific Characteristics

- Content contribution in free Q&A community: The cum. num. of answers/questions/articles
- Feedback from the community: The cum. num. of up-votes/down-votes/thank/unhelpful on own posted answers
- Social network: The cum. num. of followers/followees
- Lifetime: The num. of days since registered Zhihu
- Profile: Whether is a off-line celebrity; Whether is invited to open Live; Whether is an "Excellent Contributor" certified by Zhihu
- 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

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 running Lives' creators from the same topic on the same day
- Policy change: Whether Live rating function was launched (November 02, 2016); Whether detailed reviews was 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
- Seasonality: *Calendar (transaction) week indicators*; *Calendar (transaction) day-of-week indicators*

Note: As will be discussed in Section 4, when estimating nuisance functions using SDNNs, we pass the variables that are in italic format into linear component to control for creator, category, and time fixed effects, while pass all other covariates into nonlinear component to allow for flexible interactions among them. Notice that we let the vector of Live category indicators enters both components.

platform. Overall, these activities capture a creator's reputation that is expected to be an important factor for setting the full price of a Live and for influencing the realized demand (Shapiro, 1982; Varian, 1999; Wang and Nicolau, 2017). For creator activity on the Live market, we have the cumulative number of Lives hosted by the creator, the average sales/rating/number of reviews of the creator's past Lives, and the number of days since the creator's first Live. We also have rich descriptor variables that capture creator past activities in the Q&A platform, including creator lifetime in the community, content contribution, feedback from the community, social network, and profile information, etc. All creator characteristics (except for the profile information) is computed by the starting time of each Live hosted by the creator.

Conditions on the Live market are also likely to influence demand and creators' pricing decisions. Therefore, we control for the growth of the entire Live market and category-level competition. In addition, we also control for shocks due to changes in market design.

Lastly, we include variables that characterize conditions on the Q&A platform whose growth may potentially influence the demand and prices on the Live market. We mainly consider the growth in user base and the trend in content contribution.

3 Descriptive Analysis

In this section, we present the key summary statistics of creator characteristics and variations in Live price and demand. We then use basic regression analysis to illustrate the factors that may influence prices set by creators as well as demand. Note that the regression analysis here is best interpreted as a precursor to the ORF estimation. Although it can provide us some useful descriptive insights, it is not free of endogeneity concerns as it does not fully control for the confounders.

3.1 Summary Statistics

Table 2 reports some descriptive 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.[7] 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 actual airing. While the standard deviation of "Seats Limit" seem to be quite large in Table 3, we find that the majority of the Lives set it to be 500. Hence, there is very limited variation in this variable. Prices vary widely across Lives, ranging from 0.99 RMB to 399 RMB, as shown in Figure 2. In addition, we find that there exists some within-Live price variation across transactions for about 10% of Lives due to platform promotions. However, virtually all these promotions typically applied over the duration of the Live, making them equivalent to a permanent price cut. Therefore, when estimating price elasticity of demand at each target feature, our identification primarily leverages across-Live price variation because within-Live price variation is close to zero.

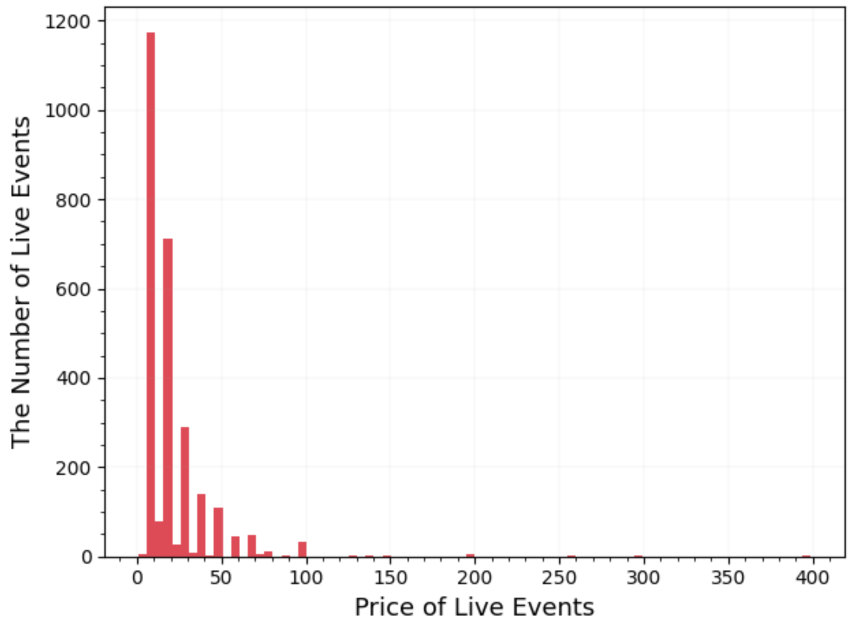


Figure 2: Distribution of Price

Note: This figure depicts the distribution of price (unit: RMB) across 2,705 Lives.

The average total sales (seats) per Live by the end of our study period is 639. We find that

[7]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

	Variable	Mean	Std.	Correlation													
				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 $p < 0.1$.

72% of sales transactions occurred prior to the airing of the Live. Thus, a surprisingly high 28% of demand is realized after the livestream. Figure 3 reports the average daily sales across all Lives, along with the 95% confidence intervals. We can see that it gradually increased throughout the pre-livestream period, peaked at the day of live (with out 60 transactions sold per day), and dropped immediately after the live. This pattern suggests a large association between demand and temporal distance to the live day.

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. The cumulative demand of a Live is public information, though the seat limit is not observable. However, a buyer is informed about whether a seat is available during her purchase process. Thus, we expect that seat availability will have a significant impact on demand. In our data, there are 583 Lives whose seats were filled up before the scheduled Live day. Figure 4 shows the average hourly sales of these Lives from 5-hour before to 5-hour after seats were sold-out, along with the 95% confidence intervals. One can see that the hourly sales drops immediately after the seats are filled up, confirming that the opportunity to interact with creator in real-time during livestream is valuable to consumers. We will provide more details on the value of live interaction based on our estimates in Section 5.

Table 3: Live Summary Statistics

				Correlation					
	Variable	Mean	Std.	V1	V2	V3	V4	V5	V6
V1	Days before Live	17	13	1					
V2	Seats 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 variable, Days before Live, is the number of days from listing to the scheduled live day.

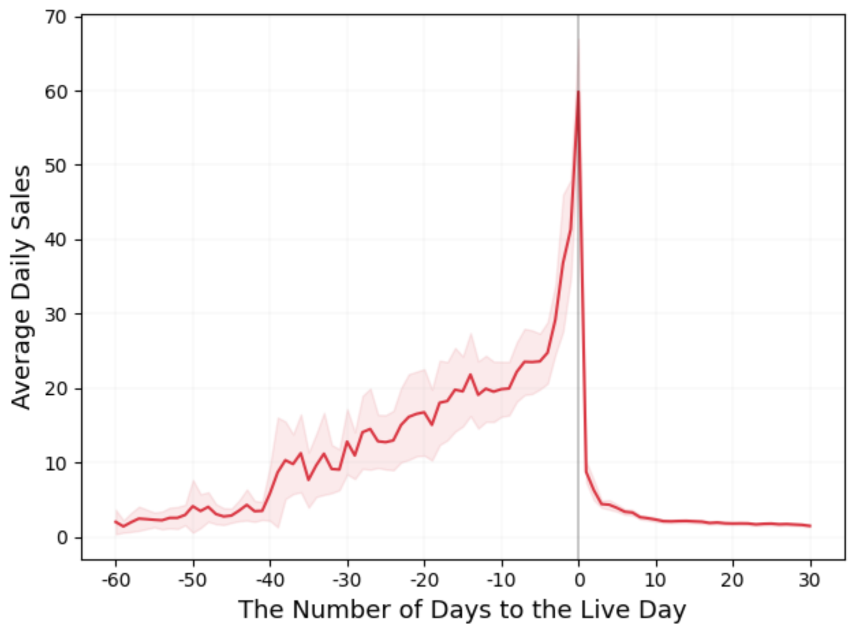


Figure 3: Average Daily Sales over Time to Scheduled Live Day

Note: The shaded areas denote the 95% confidence interval that is based on the daily sales all Lives at each temporal distance.

3.2 Descriptive Price Regression

We adopt a cross-sectional regression analysis to understand the factors that may influence the (full) price set by the creator. 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.[8] All the cumulative count variables are computed based on user i's activities by the starting of Live j. 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 ($p < 0.01$). The number of past Lives held and the average past Live rating is positively associated with the price of the current Live ($p < 0.01$), suggesting that

[8]We do not use covariates that vary by calendar day in this cross-sectional regression. We also do not control for creator and calendar day fixed effects, as this introduces too many variables for a regression with merely 2,705 observations.

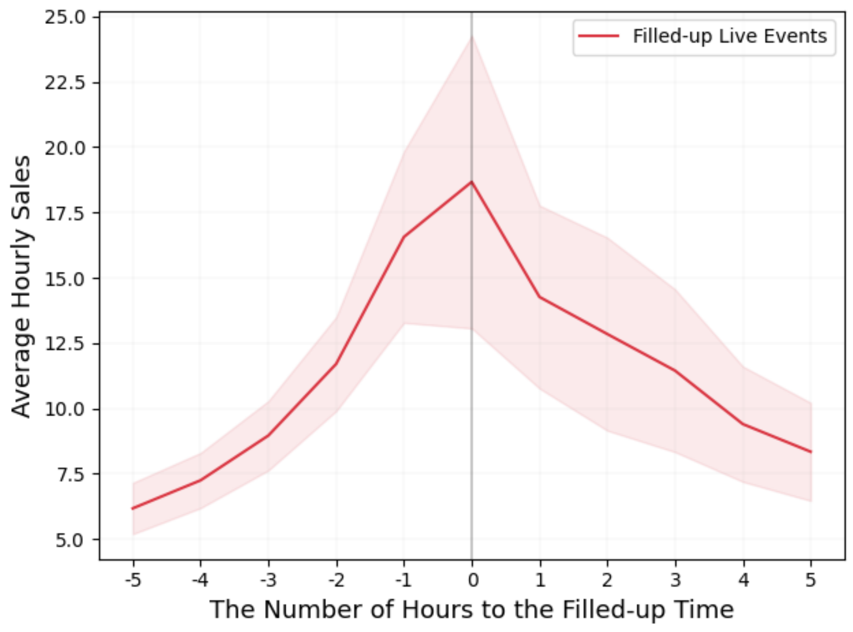


Figure 4: Average Hourly Sales over Time to Seats Sell Out

Note: This figure is based on 538 Lives whose seats are sold out before the scheduled live day. The x-axis indicates the number of hours to when seats sell out. The shaded region depicts 95% confidence interval that is based on the hourly sales of all selected Lives at each temporal hour distance.

more experienced or established creators tend to set higher prices. Similarly, we find that creator content contribution level and follower size in the Q&A platform are also positively associated with price. Lastly and not surprisingly, offline celebrities or invited creators are more likely to set higher price. Taken together, these findings suggest that creators consider timing and their experience and reputation in both the paid market and the free Q&A platform. This can be further confirmed by the improvement in R^2 across the different specifications. We can see that Live specific features and seasonality together accounts for about 8.5% of the variation in price across Lives (Model (1)), while including topic fixed effects accounts for another 3.7% of the variatoin variation (Model (2)). Creator-specific features together account for about 7.2% of the variation (Model (3) and (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 suggests that there is a potentially complex set of relationships between price and large number of observed covariates in our context, which requires the use of more flexible functional form to model these relationships.

Table 4: Regression of Log Seller's Full Price

Model	(1)	(2)	(3)	(4)
Live-Specific Features and Seasonality on the Livestreaming Day				
Seats Limit	−0.004 (0.011)	−0.002 (0.011)	−0.001 (0.011)	−0.001 (0.011)
Days before Live	0.096 (0.011)***	0.099 (0.011)***	0.065 (0.011)***	0.070 (0.011)***
Advertising		−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: *Advertising* is an indicator variable of whether the Live is advertised by the platform. 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). $^{***}p < 0.01$, $^{**}p < 0.05$, $^{*}p < 0.1$.

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)
Seats Limit	-0.007 (0.023)	-0.029 (0.016)*
Days before Live	0.200 (0.025)***	-0.134 (0.017)***
Advertising	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: *Advertising* is an indicator variable of whether the Live is advertised by the platform. 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). For post-livestream demand regression, we additionally control for rating score and log aggregate demand pre-livestream, which are realized after the livestream concludes. $^{***}p < 0.01$, $^{**}p < 0.05$, $^{*}p < 0.1$.

3.3 Descriptive Demand Regression

We now consider a cross-section analysis of total demand in the pre-livestream and post-livestream periods, respectively.[9] Our dependent variable is the log-transformed total sales of a Live j hosted by creator i from the time it was listed on the market to 30 days after it was aired. The independent variables include the log-transformed full price of Live j and a subset of covariates listed in Table 1. So the coefficient of price can be interpreted as the elasticity (though it is most likely biased as this regression does not control for all confounders). All other continuous independent variables are standardized. The results for the pre-livestream (Model (1)) and post-livestream (Model (2)) periods are shown in Table 5. We highlight two key findings. First, we find that the estimated pre-livestream price elasticity is significantly negative ($p < 0.01$), whereas the post-livestream price elasticity is insignificant. This suggests that demand is less price-elastic post-Live than pre-livesream. Second, as expected, we find that the creator's past experience and reputation appear to have a bigger impact on the pre-livestream demand than on the post-period demand.

4 Modeling Approach

While regression analysis above provides us some useful descriptive insights, there are two challenges arise in our context that are not well addressed by typical regression approaches. First, price and sales may be confounded with the large number of covariates in complex and partially known ways, which means the number of covariates and potential variables formed by different ways of interacting and transforming the covariates could be large relative to sample size. Second, price elasticity of demand for Live is likely to be heterogeneous, varying both temporally and cross-sectionally. In terms of our research question, the temporal variation (how it varies relative to the day of livestream) is particularly important. We therefore turn to the ORF analysis to estimate the causal impact of price on Live demand. We will first describe the general model setup and then the

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

ORF approach.

4.1 The General Problem

Let $Y_{it} \in \mathcal{R}$ represents the daily sales for each Live as a function of T_{it}, the treament/policy of interest, i.e., price. Given N observations $D = \{Z_{it} = (Y_{it}, T_{it}, W_{it}, X_{it})\}_{i=1}^{N}$ 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 the estimated price elasticity. In terms of our research question, the temporal distance relative to the scheduled live day is the key target feature in our main analysis. 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. That is, 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

In order to do this, we look at methods that combine causal inference and machine learning. There are two types of methods that have been proposed to estimate (heterogeneous) treatment effects with observational data.

The first proposes a two-stage estimation method (Chernozhukov et al., 2017, 2018). This method, called double machine learning (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 (e.g., Chernozhukov et al., 2017).

The second builds on random forests to estimate causal heterogeneous treatment effects. 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. To create an orthogonal moment for identifying $\theta_0(x)$, ORF follows the residualization approach similar to Chernozhukov et al. (2018). For the problem introduced in Section 4.1, we define a function $q_0(X,W) = \mathbf{E}[Y|X,W]$, and consider the residuals $\widetilde{Y} = Y - q_0(X,W)$ and $\widetilde{T} = T - g_0(X,W) = \eta$. 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 (Athey et al., 2019). Specifically, using a forest-based kernel learner, ORF first assigns a set of similarity weights between each sample and the test point x, and then computes $\hat{\theta}$ via kernel regression with the set of weights. ORF estimates are shown to be asymptotically normal and hence have theoretical properties that render bootstrap based confidence intervals asymptotically valid. We summarize all the essential steps of the ORF algorithm into pseudo code in Appendix A, along with a detailed explanation. 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.

4.3 Extending ORF Using SDNNs

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. This means all the confounding variables must be known and be included in (X,W) in the Equation System (1). This further suggests that the specification of the nuisance function (e.g., g_0 and f_0 in the Equation System (1)) in these causal inference methods are very important in making sure all the confounding factors are properly controlled for. In the conventional ORF, the nuisance function is estimated using lasso. This maintains interpretability but relies on linearity of the functional form. In contrast, purely nonparametric methods (e.g., random forests and/or DNNs) could be used, but while those could provide better fit, they compromise interpretability.

In our setting, we have a combination of known and unknown relationships between the treatment and outcome (price and demand) and sets of covariates. Specifically, the fixed effects of creator, Live category, and seasonality are typically assumed to have linear relationships with treatment/outcome according to econometric convention, while other covariates that capture creator, Live, market, and platform characteristics are likely to have unknown and complex functional forms. Thus we propose a semi-parametric approach, the SDNN, to specify the nuisance function.

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. Specifically, SDNNs allow partitioning of the covariates into two sets. The first set consists of covariates that are assumed to have a linear relationship with the treatment and outcome while the second set consists of covariates that are assumed to have a non-linear 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, the use of SDNNs to estimate nuisance functions is particularly suited to panel data (as in our setting) because the method 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). The basic idea of SDNNs is illustrated by a network diagram in Figure 5. To the best of our knowledge, ours is the first paper to use SDNNs in ORF to estimate the nuisance function.

We operationalize this using a deep feedforward neural network architecture with multiple inputs. For ease of notation, let $\chi = (X, W)$ represent all the control variables. We form two sets of variables 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 of a nuisance function. The SDNNs for estimating the nuisance function q is represented in below:

$$\begin{aligned} &Y_{it} = q_{01}(\chi_{it}^p) + V_{it}^1\Gamma^1 + \varepsilon_{it} \\ &V_{it}^1 = \sigma(\gamma^2 + V_{it}^2\Gamma^2), V_{it}^2 = \sigma(\gamma^3 + V_{it}^3\Gamma^3), \ldots \quad \ldots, V_{it}^L = \sigma(\gamma^L + \chi_{it}^n\Gamma^L) \end{aligned} \tag{2}$$

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 (the 1th) layer is a linear model in the χ^p and derived variables V_{it}^1, covariates in the parametric component χ^p thus have a linear relationship with the treatment and outcome.

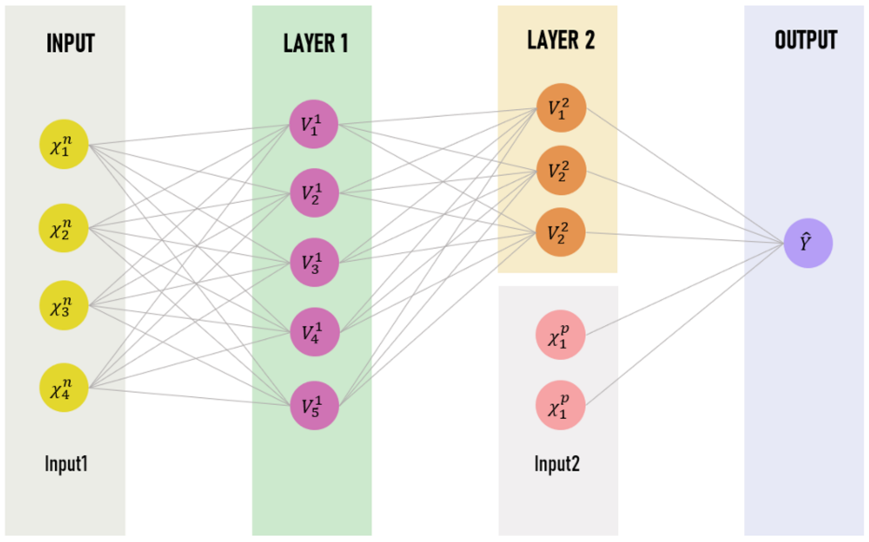


Figure 5: Panel Data Neural Network Model

Note: Nonlinear component will enter the model through input1 and the linear component (e.g., the vector of creator fixed effects) will enter through input2.

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 (as Algorithm 2) 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):

$$\hat{q} = \arg\min_{q} \sum_{i \in D_1} a_i (Y_{it} - q_1(\chi^p) - V_{it}^1 \Gamma^1)^2 + \lambda_q ||q||^2 \tag{3}$$

$$\hat{g} = \arg\min_{g} \sum_{i \in D_1} a_i (T_{it} - g_1(\chi^p) - V_{it}^1 \Gamma^1)^2 + \lambda_g ||g||^2 \tag{4}$$

where, $\lambda \geq 0$ are the regularization parameters. Larger values tend to shrink the parameters in the parametric and nonparametric components toward zero. See Appendix B for a detailed instruction of how to enable SDNNs in ORF using Python.

5 Price Sensitivity and Temporal Distance to Live

5.1 Empirical Approach

We now apply ORF to understand how the price elasticity of demand changes over Live life-cycle. Before estimating ORF, we need to specify a set of test points on the target feature whose impact (on price elasticity) is being estimated. As our objective is to document the price elasticity variation relative to the temporal distance to the Live, we choose each day, d, as our test points. Specifically, we consider 29 test points within the window of 14 days before and after the scheduled live day, i.e., $d \in \{-14, -13, ..., 0, ...13, 14\}$.[10]

Next we turn to our design of the SDNNs for the nuisance function estimation. We divide the covariates into two sets. The first set consists of indicator variables that control for creator, Live category and time. The remaining covariates (see Table 1) get included in the nonlinear component to allow for flexible interactions among them. The hyperparmeter in the neural network is suitably chosen to maximize predictive performance of both the price and demand equations. Details on how the SDNNs are trained are provided in Appendix D.

[10]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.

5.2 Main Results

Figure 6 displays the price elasticities over the Live life-cycle. The shaded region is the 95% confidence interval obtained via 100 bootstrap samples. The estimated price elasticity is about -0.5 around 14-day before the scheduled Live day, becomes less negative approaching the scheduled Live day, reaches around zero on the scheduled live day. Over the post-livestream period, demand is still price elastic at around -0.1, but surprisingly, it is much less sensitive than the pre-livestream demand. The difference between pre-livestream and post-livestream price elasticities can be as large as 80%. As can be seen from the dashed line in Figure 6, there is a small magnitude of fluctuation in the temporal pattern estimates. This is due to the randomness created by subsampling and sample spliting in each iteration process of random forest.[11]

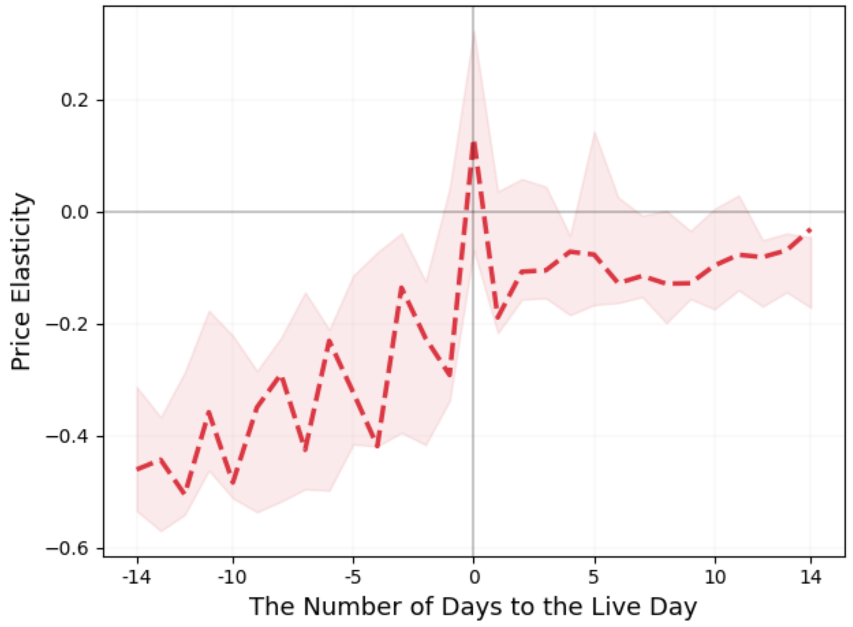


Figure 6: Price Elasticity over Time to Scheduled Live Day

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

The pattern above suggests that the live part of the content is valuable, but there is also considerable value in the content that is not centered in it live attribute. This finding has important

[11]The slightly wider confidence interval over pre-livestream period compared to post-livestream period is because Lives are listed on the market for purchase at difference temporal distance.

implications for platforms and creators to implement dynamic pricing and promotion over the event life-cycle. In section 6, we conduct a series of analysis to better understand mechanisms driving the results.

It is important to 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 then 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 (e.g., concerts, performing arts) (e.g., Zieba, 2009; Felton, 1992; Pompe et al., 2018; Buzanakova and Ozhegov, 2016). In order to check that this also holds in our setting, we surveyed the prices of live content on Zhihu and competing platforms in China during 2016-2017, by randomly scarping about 100 paid content/events from each platform. As shown in Table 6, the price points for Lives hosted on Zhihu are typically higher than those of comparable events on competing platforms. This suggests that there is little room for creators on Zhihu Live to increase their prices.

Table 6: Competitor Price

	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: The survey was conducted by randomly scarping about 100 paid content/events that were streamed during 2016-2017 on each platform. To make sure that prices are comparable across platforms, we anchored the unit to RMB/30min.

5.3 Robustness Checks

A possible explanation for the main result shown in Figure 6 is that the temporal pattern is driven by different types of consumers purchasing at different temporal distance to the day of Live. These differences across consumers could be based on observable or unobservable characteristics.

5.3.1 Differences on Observables

It is likely that consumers with different engagement levels on Zhihu - an observable - arrive to purchase at different temporal distance. For example, consumers who have higher engagement and loyalty to Zhihu may have a higher tendency to arrive on or before livestreaming starts than others. We therefore examine differences on observables by zeroing in on three important consumer characteristics that capture user engagement on the platform: consumer lifetime, follower count and followee count on the Q&A community at the time of purchase. 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 data if we focus only on the Live market. We regress each of the three consumer characteristics on Live characteristics (e.g., unit price, cumulative demand) and the fixed effects of temporal distance, calendar (transaction) day and Live category using transaction-level within 14-day before and after the scheduled live day.[12] The point estimates in the temporal distance fixed effects provide an estimate of how consumers who bought a given Live differed over temporal distance to the day of livestream.

The findings are reported in Figure A4 of Appendix E. We can see that the estimates are quite noisy and the differences in estimates are mostly insignificant (or very small in magnitude if significant). Therefore, it seems extremely unlikely that differences across these observables could account for the more than 70% decrease in consumer price elasticity from pre-period to post-period.

A different observable that could account for the temporal dynamics in price elasticity could

[12]We use transaction-level data instead of panel data because if a Live has zero transaction on a day, then consumer characteristics will be missing value for that day.

be purchase activity. Specifically, it could be that a small group of heavy purchasers is driving the main finding. 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. As can be seen from the figure, our main results are robust to the exclusion of these extreme consumers.

5.3.2 Differences on Unobservables

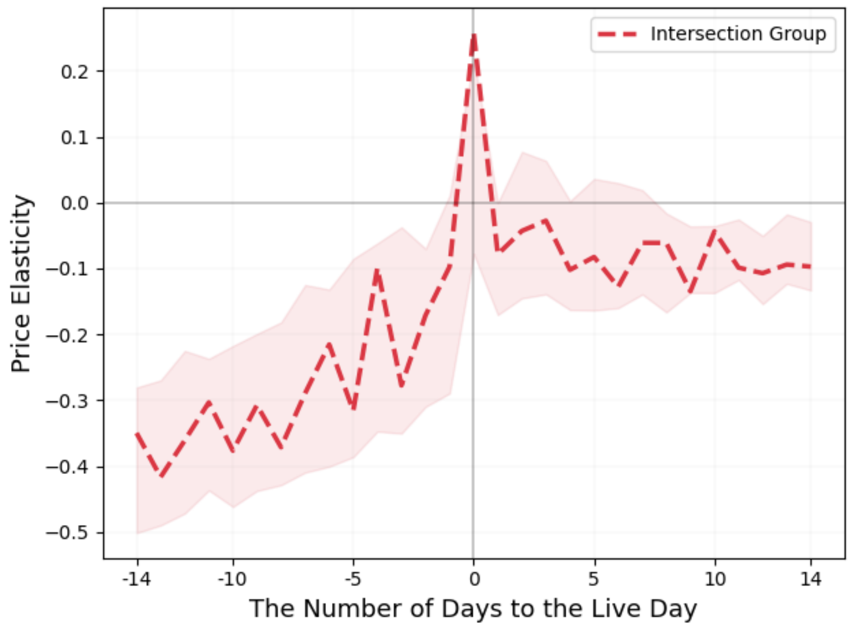
It is possible that consumers vary systematically on their preference - an unobservable - for consuming live versus recorded content. We therefore partition all consumers into three segments: pre-loyal (consumers who have only purchased before day of Live), post-loyal (consumers who have only purchased access to recorded versions of Lives i.e., after day of Live), and intersection (consumers who have purchased Lives in both periods). Table 7 shows the details of each segment. As can be seen from the table, the intersection segment comprising 20% of consumers is the most active, accounting for 64% of all transactions. For each Live-day observation, we recompute the daily demand within each consumer segment. We then replicate the ORF estimation for each consumer segment separately. For the intersection segment, we estimate price elasticity as before for $d \in [-14, 14]$. We estimate the price elasticity for $d \in [-14, 0]$ for the pre-loyal segment and over $d \in (0, 14]$ for post-loyal segment.

Table 7: Consumer Segment Characteristics

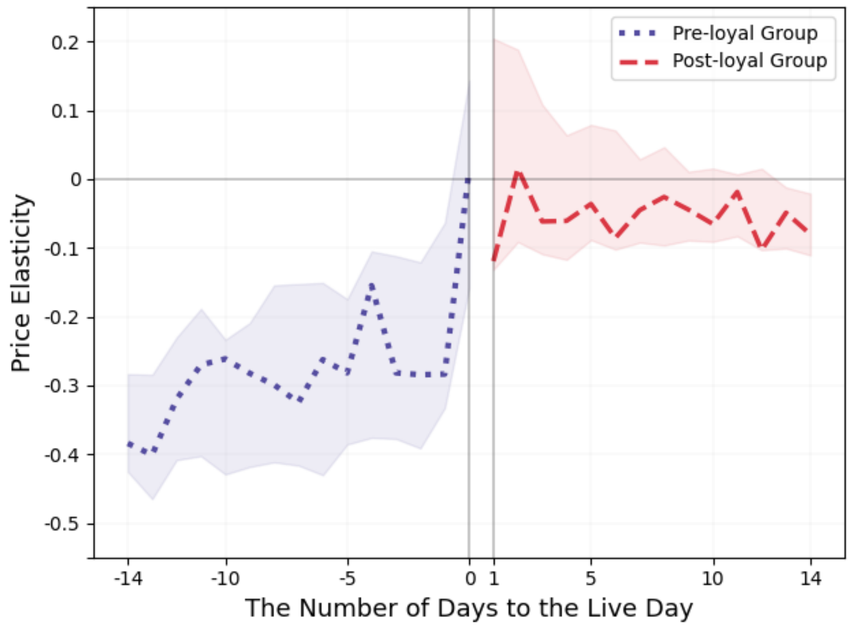
	Intersection	Pre-loyal	Post-loyal
Num. Consumers	181,233	537,938	183,766
% Consumers	20	59	21
% Transactions	64	26	10
Num. pre-livestream purchases	7.66 (272.96)	1.80 (21.20)	0.00 (0.00)
Num. post-livestream purchases	2.99 (11.95)	0.00 (0.00)	1.27 (2.24)

Note: The last two rows report the average purchases per consumer within each consumer segment with the standard deviation in parentheses.

The estimation results are shown in Figure 7. As can be seen from this figure, the temporal



(a) Price Elasticity of the Intersection Segment



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

Figure 7: Time-varying Price Elasticity 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.

pattern for the consumers in the intersection is very similar to the temporal pattern on our main result (Figure 6).[13] This provides strong evidence that our main results are not driven by systematic differences between consumers who buy only in the pre-livestream period and those who buy only in the post-livestream period. In addition, the pre-livestream temporal patterns are replicated among consumers in pre-loyal segment, and the post-livestream temporal patterns are also replicated among consumers in post-loyal segment. As a result, we can conclude that the temporal variation in the price elasticity of demand is not driven by selection on unobservables among consumers.[14]

6 Mechanisms

Our main result shows systematic dynamics in the price elasticity of demand in terms of the period before, on the day of and after the Live. As can be seen from Section 5.3, these dynamics are not caused by selection across consumers. In this section, we try and examine the potential mechanisms driving the underlying temporal patterns.

6.1 The Value of Live Interaction

We start by focusing on the inelastic demand on the day of Live. The most likely mechanism for this, given the setting, is that arises from the consumer placing a high value on live interaction with the creator. Imagine that a consumer is deciding which Live to purchase. If the consumer purchases a Live that will be aired on the same day, s/he will be able to enjoy the live interaction immediately. This is much more attractive compared with purchasing Lives that will be aired in the future.

[13]Note that the slightly wider confidence bands in Figure 7 compared to Figure 6 are due to increased data sparsity after data partition into different consumer types.

[14]Table 7 shows that the standard deviation of the average number of purchases per consumer is large, indicating the existence of extreme consumers within each segment. This is especially true for the intersection segment. Therefore, for each consumer segment, we 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 of each segment are robust to the exclusion of extreme consumers. The results are available upon request.

To test this possible mechanism, we focus on 317 Lives for which seats were filled up within one week before the scheduled live day.[15] For these Lives, buyers who purchased on the scheduled live day could not interact with the creator in real-time during the livestream. If our proposed mechanism is at play, the price elasticity on the scheduled live day will be significantly negative for these lives (as live interaction is no longer possible). To test this prediction, we replicate our main analysis using these Lives. The results are displayed in Figure 8. Consistent with our expectation, there is a sharp drop in price elasiticity on the scheduled live day, leading to a significantly negative price elasticity. This strongly suggests that the inelastic demand on scheduled live day in our main result, Figure 6, is driven by the value inherent in real-time interaction with the creator.

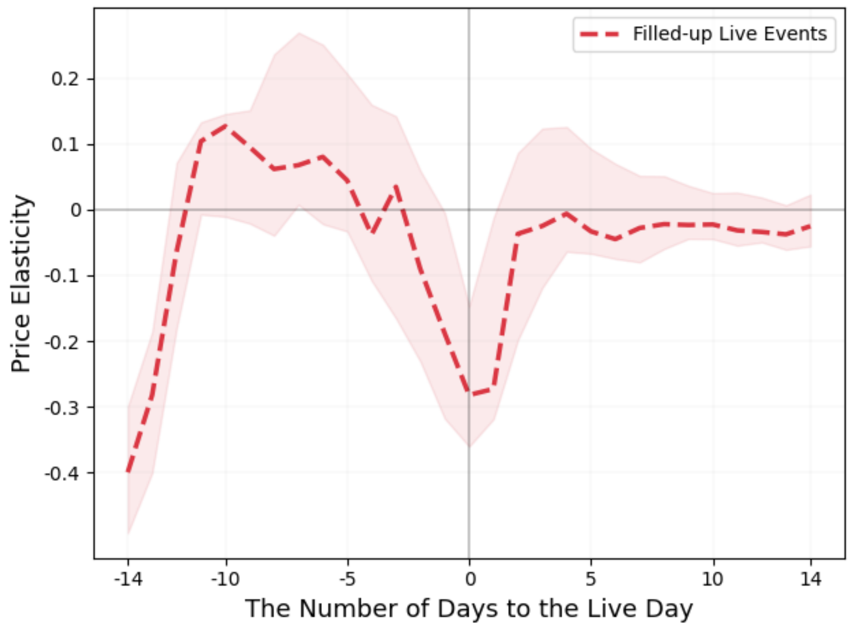


Figure 8: Price Elasticity over Time to Scheduled Live Day among Filled-up Lives

Note: The seats are filled up when cumulative sales reach the seat limit set by the creator. The shaded region depicts 95% confidence interval via 100 bootstrap resamples.

In addition, we find that price elasticity in Figure 8 is essentially zero for $d \in [-10, -1]$, i.e., during the period that cumulative sales get close to the seat limit. This suggests that consumers

[15]The one week cutoff is conservative as a longer cutoff would make our result even stronger. In analyses available from the authors on request, we find that our results are robust if we use the data on all 583 Lives that were sold out before the day of airing.

value the opportunity of interacting with creator in general, even when the livestream will occur a few days after the purchase. In order to test this more formally, we examine how the price elasticity of demand changes on a daily basis relative to the day that the seat limit was reached. As can be seen from Figure 9, the price elasticity shows an increasing trend, reaching zero, before the seat limit was reached. In contrast, the trend becomes a decreasing one once the seat limit is reached. Take together, these results show that consumers place a high value on interacting with the creator during the livestream.

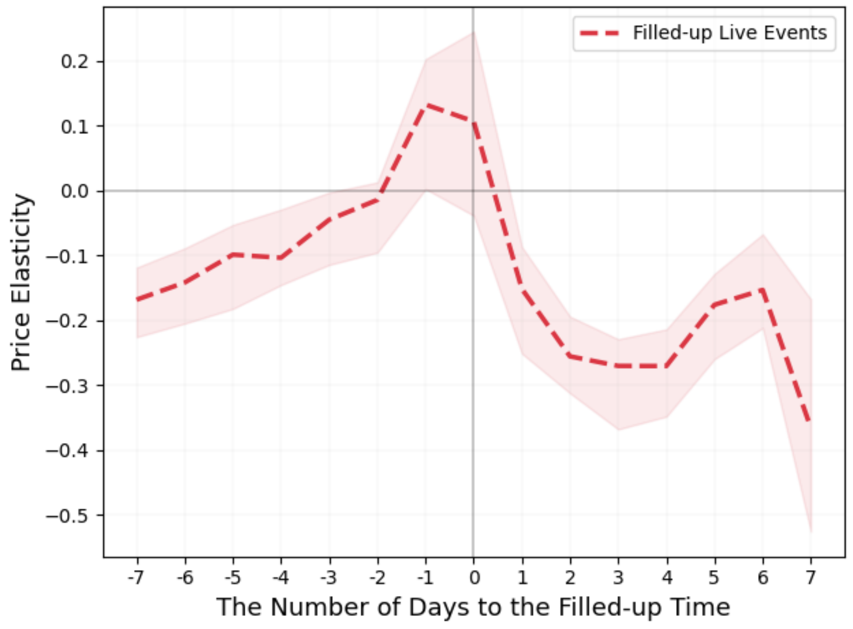


Figure 9: Price Elasticity Before and After Seats Sell Out

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

6.2 The Importance of Live Quality

We now turn to the mechanism driving the less price-sensitive demand over the post-livestream period. In the Zhihu Live market, creators typically provide content on serious (non-entertainment) topics. Given that this requires a reasonable level of in-depth knowledge and expertise, most creators usually offer only one Live. Thus, from the consumer perspective, there is considerable uncertainty about the quality of each Live. This uncertainty is higher over the pre-livestream period than over the post-livestream period because consumer ratings and reviews of the live become available when the recorded live event is released. Therefore, the reduction in uncertainty may potentially explain why post-livestream demand is much less price-sensitive than the pre-livestream demand.

In order to consider this as the driving mechanism, we need to first establish that live quality impacts consumer willingness-to-pay on the Zhihu Live market. We measure the quality of a Live using its rating, which is provided by consumers who have purchased and consumed the Live. As noted above, the rating is only available after the livestream ends. We therefore focus on post-livestream transactions to examine the impact of quality on willingness-to-pay. Figure 10 shows the distribution of rating score across all the Lives in our data. We can see that about 95% of Lives have a rating between 3 and 5 (on a 5 point scale).

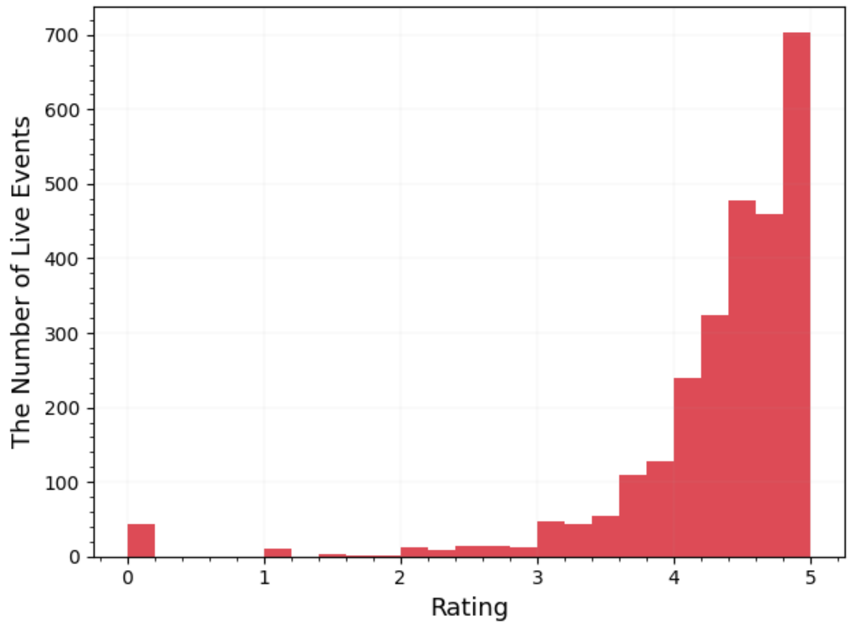


Figure 10: Distribution of Rating

Note: This figure depicts the distribution of rating scores across 2,705 Lives. For 43 Lives whose rating score is missing value, we impute rating score with zero.

If Live quality does matter, then demand should be less price sensitive for Lives with a higher rating (Zeithaml, 1988; Kirmani and Wright, 1989; Isik et al., 2004; Okada, 2010; Tsui, 2012). To investigate this, we treat rating score as the target feature in ORF. Given the sparsity of Lives with a rating below 3, we use six support 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 less price sensitive for Lives with higher rating score. In particular, a Live rated 5 has a price elasticity (on average) at least 65% lower than one rated 3. This suggests that consumers have much higher willingness-to-pay for high-quality content, confirming that quality matters in Zhihu Live market.

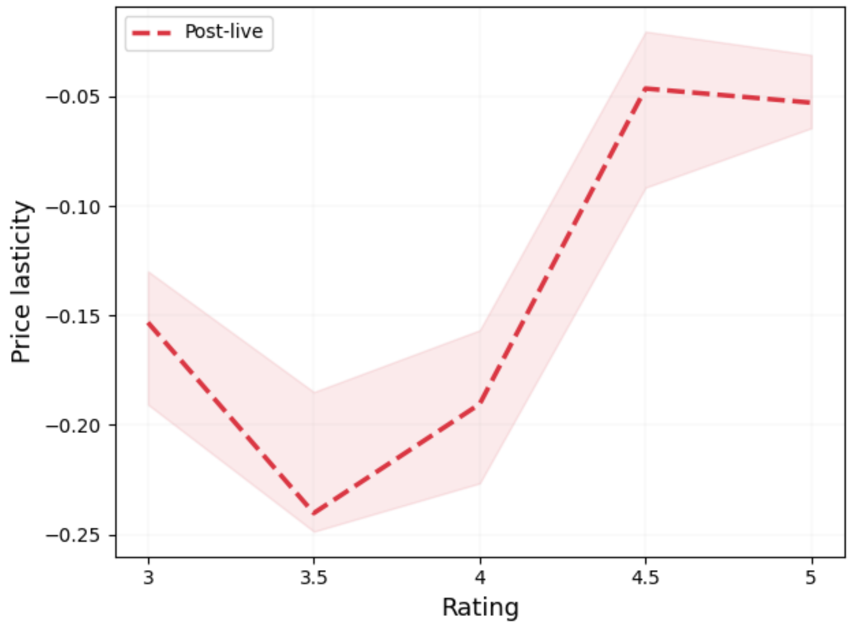


Figure 11: Price Elasticity and Rating Score

Note: This figure reports the changes in price elasticity across rating score over the range [3, 5]. The shaded region depicts 95% confidence interval via 100 bootstrap resamples.

6.3 Quality Uncertainty Reduction

Given that Live quality impacts the willingness-to-pay, we investigate this in detail by leveraging findings in the prior literature that consumers typically use informations cues to lower uncertainty (Lanzetta, 1963; Urbany et al., 1989). In our context, if uncertainty about Live quality indeed reduces after the live event concludes, we would expect to see reduced impact of information cues on purchase decision over the post-livestream period.

We test this by estimating the impact of information cues on price elasticity of demand over the pre-period and post-period separately. In particular, we consider four information cues on the Zhihu Live market: social proof, consumer prior knowledge of a creator, creator reputation, and content category. Note that it is indeed possible that consumers use other information cues to reduce uncertainty. However, we focus on these four as we have access to these data.

6.3.1 Social Proof

Literature on observational learning has shown that consumers could infer quality from popularity information (e.g., how much existing consumers have purchased the product) (Cai et al., 2009; Tucker and Zhang, 2011). In the Zhihu Live market, the key product-specific popularity information is the cumulative sales of a Live, which is displayed prominently on the Live page over its entire life cycle. If consumers rely on popularity information to resolve uncertainty, we expect to see a negative association between consumer price elasticity and cumulative demand. In addition, if uncertainty reduces after the live event concludes, we expect that the popularity information plays a less important role for post-livestream purchases.

To investigate the role of popularity information, we apply ORF algorithm to understand how price elasticity changes with the cumulative demand of a Live within pre- and post-periods, respectively. Our target feature in ORF is the log-transformed cumulative demand of a Live by each day. Figure 12 reports the estimation results with 7 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 for pre-livestream purchases. 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 uncertainty about the live quality is high.

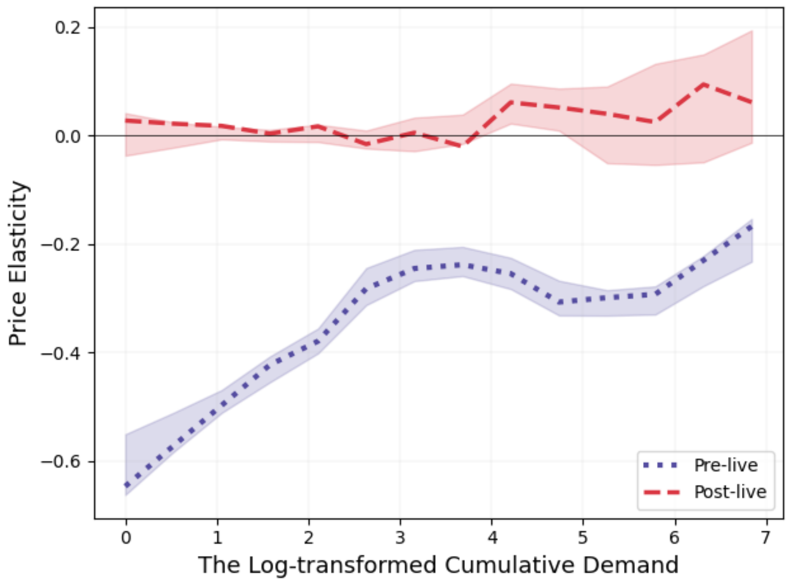


Figure 12: Price Elasticity and Cumulative Demand

Note: Each line shows the changes in price elasticity with the log-transformed cumulative demand within a period. The shaded region depicts 95% confidence interval via 100 bootstrap resamples.

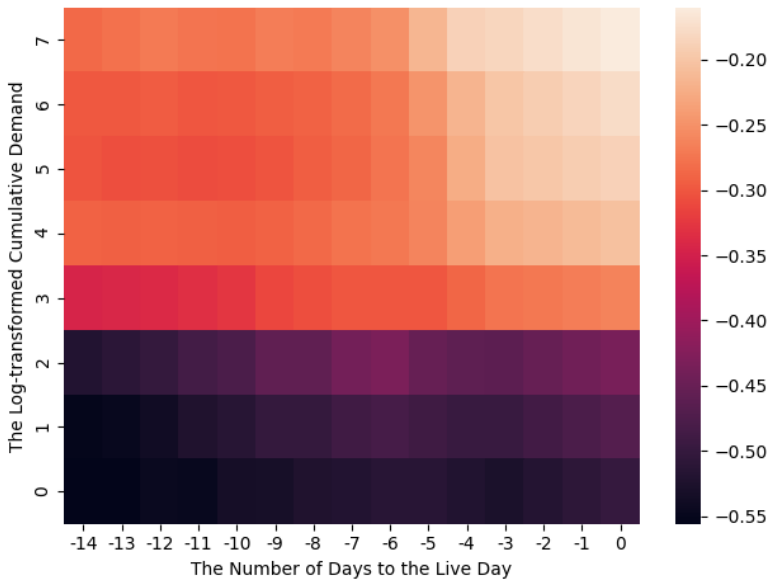


Figure 13: Price Elasticity over Time to The Live Day and Cumulative Demand

Note: This figure depicts the changes in price elasticity with the log-transformed cumulative demand and the temporal distance to the scheduled live day within the pre-livestream period.

It is possible that cumulative demand is just acting as a proxy for the temporal distance to scheduled live day, especially over pre-livestream period. In that case, the pattern in Figure 12 could just reflect demand as the day of live gets closer. In order to test this, we carry out the ORF estimation again (using all the pre-livestream observations). However, we now set both cumulated demand and temporal distance to day of live as target features using a grid of $98 = 7 \times 14$ test pairs. The results are shown in over Figure 13. We find that our main findings 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 note that the social proof mechanism not only explains the less price-sensitive demand over post-livestream period, but also sheds light on the decreasing price elasticity (in absolute value) as the scheduled live day gets closer in our main result (Figure 6).

6.3.2 Live Topic Category

Existing literature suggests that one way consumers mitigate uncertainty about a purchase decision is through the use of a heuristic that uses 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 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 (i.e., Internet, Psychology, Education, Travel, Life Style, Food & Cuisine, Career, Reading & Writing, Music/Game/Movie.[16] The literature suggests that soft categories generally have higher (perceived) quality uncertainty than hard categories (Levitt, 1981;

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

Bebko, 2000; Castelo et al., 2019; Dai et al., 2020). We also see this in our data as the the variance in rating across Lives in the soft categories is significantly higher than that in the hard categories ($p < 0.05$).

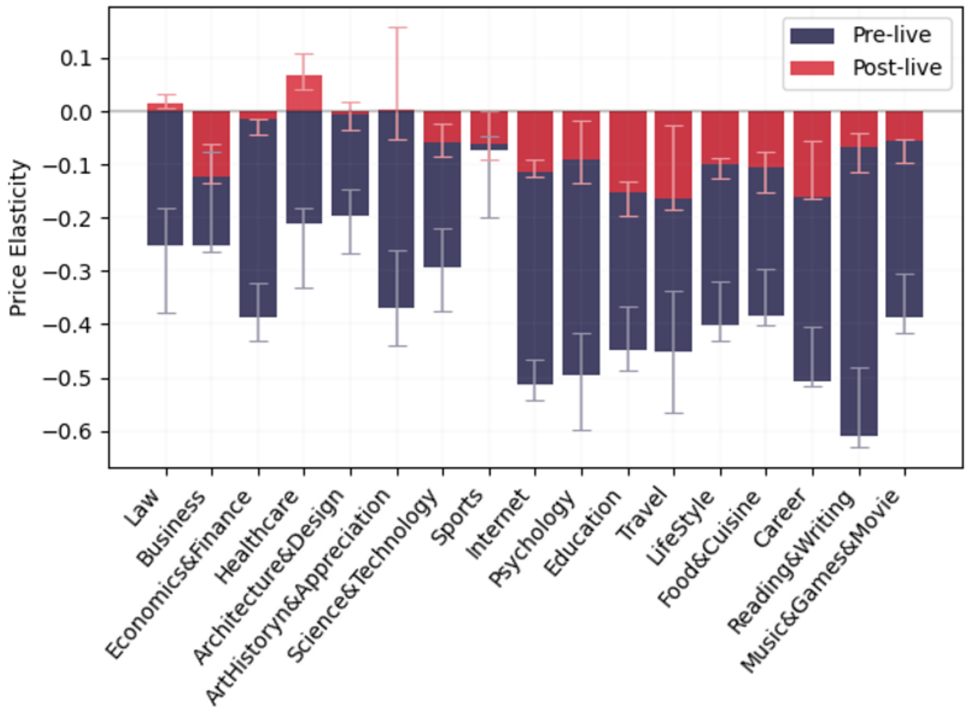


Figure 14: Price Elasticity and Live Topic Category

Note: We categorize the eight topics that start from the left as *hard* and the remaining nine topics as *soft*. The error bars depict 95% confidence interval via 100 bootstrap resamples.

We apply ORF to estimate how the price elasticity of demand variables across the 17 topic categories for the pre- and post-period. Figure 14 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 main results, demand is significantly less price-sensitive over the post-livestream 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 is consistent 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 rely more on category characteristics (as heuristics) for pre-livestream purchases, while relying less on category characteristics for post-livestream purchases. This sharp difference provides further evidence the quality uncertainty reduction mechanism drives the difference in the price elasticity results pre- and post-livestream.

6.3.3 Consumer Prior Knowledge on Creator

Consumers can also infer the quality of a Live from their prior knowledge about the creator. In other words, all else being equal, the more the knowledge they have about the creator, the lower the uncertainty. One way consumers can obtain this knowledge is to follow the creator on the broader Zhihu (free) Q&A community. By doing so, the consumer gets access to the creator's (free) content on her personalized content-feed. This will help the consumer determine better the expertise level of the creator, leading to lower uncertainty about the quality of the creator's Live.

Following this logic, we split all the individual transactions into two groups based on whether each consumer has followed a creator before she purchase the creator's live. About 20% of consumers have followed the creators before purchasing the live. These consumers account for about 17% of all transactions. We label them as the group with relatively high prior knowledge, and label the remaining consumers as the group as consumers with relatively lower prior knowledge. Then, for each group we aggregate all the transactions at Live-day level and repeat ORF algorithm with temporal distance to scheduled live day as our target feature.

Figure 15 presents the results for both consumer groups. First, the overall temporal dynamics in both consumer groups are consistent with the main results shown in Figure 6, with slight differences on the absolute magnitudes of the price elasticities. Second, the high prior knowledge group is significantly less price sensitive than the low prior knowledge group at all temporal distances, except for a few days around the scheduled live day. This observation confirms our hypothesis that prior knowledge about a creator can influence consumer perceived quality uncertainty and hence price sensitivity. Third, going from the pre-livestream to the post-livestream periods, the difference in price elasticity between the two consumer groups is reduced significantly, suggesting that consumers are much less likely to rely on their prior knowledge on creator to resolve uncertainty over the post-livestream period.

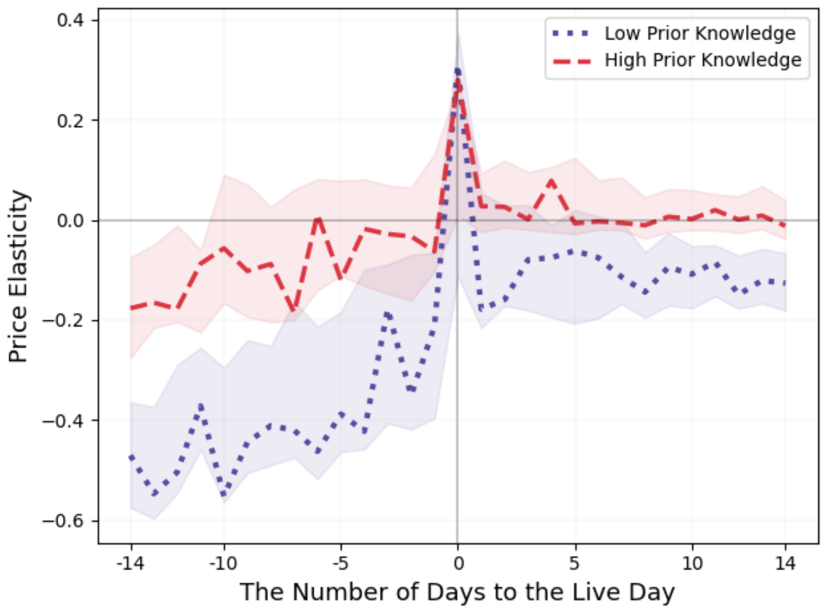


Figure 15: Price Elasticity and Consumer Prior Knowledge on Creator

Note: We categorize consumers who have followed the creator before purchase into the high prior knowledge group, and those who have not into the low prior knowledge group. The shaded region depicts 95% confidence interval via 100 bootstrap resamples.

6.3.4 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). In the context of Zhihu, even if a consumer has never followed a creator, the consumer may still be able to resolve quality uncertainty by examining the creator's reputation in the free Q&A community. In particular, the total number of followers is the most important indicator of a user's reputation and expertise level on Zhihu, as followers have to be gained through persistent contribution of high-quality content in the free Q&A community, as is common on many other UGC platforms (Toubia and Stephen, 2013; Goes et al., 2014). If consumers use the number of followers that a creator has gained to resolve uncertainty, we expect that consumers are less price-sensitive to Lives hosted by creators with a larger number of followers. In addition, if uncertainty reduces after live event concludes, we expect to see a lower impact of follower size on post-period demand.

To test this hypothesis, we apply ORF algorithm to understand how consumer price elasticities vary across creator follower size over the pre-livestream and post-livestream periods. Our procedure can be described as follows. We first compute the cumulative follower size 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.[17] Thus, for each topic k we obtain the maximum number of followers 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 in ORF algorithm 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 16. One can see a clearly increasing trend in both periods, confirming that consumers are less price sensitive if the creator has a 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 other sources of information to infer Live quality when word-of-mouth on the same Live is not available over pre-livestream period. 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 provide suggestive

[17]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 appeal, such as Humanities.

evidence that follower size is a key piece information that consumers can use to lower quality uncertainty about the Live quality, especially over the pre-livestream period.

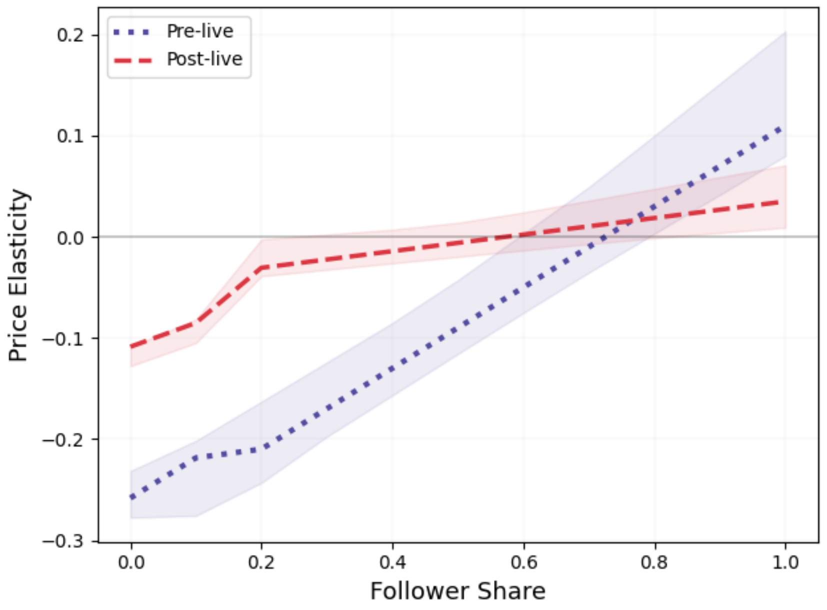


Figure 16: Price Elasticity and Creator Reputation in the Q&A Community

Note: Each line shows the changes in price elasticity with *follower share* within a period. The shaded region depicts 95% confidence interval via 100 bootstrap resamples.

In conclusion, we find strong correlational evidence for two key mechanisms that drive our main results. First, consumers value live interaction with the creator highly. This lowers their price sensitivity dramatically, leading to inelastic demand on the day of Live. Second, Live quality matters to consumers, impacting their willingness-to-pay significantly. Consumers also seem to use at least four information cues - social proof, consumer prior knowledge of a creator, creator reputation, and topic of Live - to lower their quality uncertaintly, showing a higher willingness-to-pay whenever quality uncertainty is lower.

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 live. 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 for Live events over temporal distance to the scheduled live 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 then provide correlationally consistent evidence for the mechanisms driving these results. First, we attribute the inelastic demand on the day of livestream to the possibility of real-time interaction with the creator during the livestreaming. Second, uncertainty reduction (and not selection across consumer types) plays a role for the gradually decreasing trend in price elasticity over the pre-livestream period and the much less price-sensitive demand over the post-livestream period.

These findings have important managerial implications. First, the results confirm that some of the value of livestreaming lies in the real-time consumer-speaker 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. These findings are likely to be valuable to both streamers/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 beneficial for all parties (consumers, creators, and platforms). 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 advertising.

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 personalized advertising and recommendations.

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 livestreaming content on non-entertainment topics. It is indeed possible that the value of livestreaming may show up differently on entertainment focused platforms. Second, 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. Third, 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.

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.

Gomes, M. (2018, April). 5 reasons why live streaming is so important. https://manycam.com/blog/why-live-streaming-is-so-important/. Acccessed on June 7, 2021.

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.

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 Details of ORF Algorithm

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 $r$
 1: Randomly partition $D$ into two data sets $D_1$ and $D_2$ of equal size
 2: procedure NUISANCE ESTIMATION
 3:     for each tree $b \in 1,...,B$ do
 4:         Randomly sub-sample $s$ observations from $D_1$ without replacement to form $S_b$
 5:         Randomly partition $S_b$ into two datasets of even size: $S_b^1$ and $S_b^1$
 6:         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
 7:         Obtain the leaf $L_b(x) \subseteq \mathcal{X}$ that contains the target feature $x$
 8:         for each observation $i \in D_1$ do
 9:             Compute the tree weight: $\omega_{ib} = \frac{1[(X_i \in L_b(x)) \wedge (Z_i \in S_b^2)]}{|L_b(x) \cap S_b^2|}$
10:     for each observation $i$ in $D_1$ do
11:         Compute the importance weight: $\omega_i = \frac{1}{B}\sum_{b=1}^{B} \omega_{ib}$
12:     Estimate $\hat{q}$ and $\hat{g}$ using observations in $D_1$ and weights $\omega$      ▷ See Appendix B

13: procedure TARGET ESTIMATION
14:     for each tree $d \in 1,...,B$ do
15:         Randomly sub-sample $s$ observations from $D_2$ without replacement to form $J_b$
16:         Randomly partition $J_b$ into two datasets of even size: $J_d^1$ and $J_d^2$
17:         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
18:         Obtain the leaf $L_d(x) \subseteq \mathcal{X}$ that contains the target feature $x$
19:         for each observation $i \in D_2$ do
20:             Compute the tree weight: $a_{id} = \frac{1[(X_i \in L_d(x)) \wedge (Z_i \in J_d^2)]}{|L_d(x) \cap J_d^2|}$
21:     for each observation $i$ in $D_2$ do
22:         Compute the importance weight: $a_i = \frac{1}{B}\sum_{d=1}^{B} a_{id}$
23:         Compute the residuals: $\widetilde{Y}_i = Y_i - \hat{q}(X_i, W_i)$, and $\widetilde{T}_i = T_i - \hat{g}(X_i, W_i)$
24:     Estimate $\hat{\theta}$ by solving the kernel residualized regression problem

            $\hat{\theta} = \arg\min_{\theta} \sum_{i:Z_i \in D_2} a_i(\theta \widetilde{T}_i - \widetilde{Y}_i)^2$      (5)

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 $\rho$, maximum number of splits (tree depth) $\tau$, number of randomly selected candidate
    features $m$ at each split, and number of proposed split points $k$
 1: Initialize the tree with root node $P_0$ covering the whole feature space, and the Queue $Q = \{P_0\}$
 2: Initialize the observations belong to the current node $P_0$ as $S^{P_0} = \{S_1, S_2\}$
 3: Initialize the accumulated number of splits $accSplits = 0$
 4: while $NotNull(P \leftarrow Q)$ & $accSplits \leq \tau$ do
 5:     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 B
          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: $\hat{\theta}_P = \arg\min_{\theta} \frac{1}{|S_1^P|} \sum_{Z_i \in S_1^P} \frac{1}{2}(\theta\widetilde{T}_i - \widetilde{Y}_i)^2$
 6:     Compute the empirical Hessian: $A_P = -\sum_{i \in S_1^P} \widetilde{T}_i^2 / |S_1^P|$
 7:     Generate $k$ split points $Cand_P$ based on $S_1^P$        ▷ See Algorithm 3
 8:     for each candidate split point $C \in Cand_p$ with children nodes $C_1$ and $C_2$ do
 9:         Obtain the remaining subsamples that belong to the children nodes, denoted as
                $S^{C_1} = \{S_1^{C_1}, S_2^{C_1}\}, \quad S^{C_2} = \{S_1^{C_2}, S_2^{C_2}\}$
10:         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
11:             Compute the approximated estimates at each children node $j \in \{1, 2\}$:
                    $\tilde{\theta}_{C_j} = \hat{\theta}_P - \sum_{i \in C_j} A_P^{-1}(\widetilde{Y}_i - \hat{\theta}_P \widetilde{T}_i)\widetilde{T}_i$
12:             Compute the proxy heterogeneity score at node $C$:
                    $\tilde{\Delta}(C_1, C_2) = \sum_{j=1}^{2} \frac{1}{|C_j|}(\tilde{\theta}_{C_j} - \hat{\theta}_P)^2 = \sum_{j=1}^{2} \frac{1}{|C_j|}(\sum_{i \in C} A_P^{-1}(\widetilde{Y}_i - \hat{\theta}_P \widetilde{T}_i)\widetilde{T}_i)^2$
13:         else
14:             Remove candidate $C$ from the set $Cand_p$
15:     if $NotNull(Cand_p)$ then
16:         Select $C^* \in Cand_p$ that maximizes the proxy heterogeneity score
17:         Update the sample $S^{S_j}$ as these belong to the children node $C_j^*$ for $j \in \{1, 2\}$
18:         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 $P$
```

A.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 $D = \{Z_i = (T_i, Y_i, W_i, X_i)\}_i^n$. 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:

$$\widetilde{Y}_i = Y_i - \hat{q}_P(X_i, W_i), \widetilde{T}_i = T_i - \hat{g}_P(X_i, W_i) \tag{6}$$

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 $\widetilde{X}$:

$$\hat{\theta}_P = \arg\min_{\theta} \frac{1}{|P \cap S_b^1|} \sum_{Z_i \in P \cap S_b^1} \frac{1}{2}(\theta \widetilde{T}_i - \widetilde{Y}_i)^2 \tag{7}$$

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:

$$\Delta(C_1, C_2) = \sum_{j=1}^{2} \frac{1}{|C_j|} (\hat{\theta}_{C_j} - \hat{\theta}_P)^2 \tag{8}$$

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,

$$A_P = \nabla_\theta^2 \left(\frac{\sum_{i \in P \cap S_b^1} \mathcal{L}_{res}(\hat{\theta}_P, Z_i)}{|P \cap S_b^1|} \right) = -\frac{\sum_{i \in P \cap S_b^1} \widetilde{T}_i^2}{|P \cap S_b^1|} \tag{9}$$

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,

$$\tilde{\theta}_C = \hat{\theta}_P - \sum_{i \in C} A_P^{-1} (\widetilde{Y}_i - \hat{\theta}_P \widetilde{T}_i) \widetilde{T}_i \tag{10}$$

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

$$\tilde{\Delta}(C_1, C_2) = \sum_{j=1}^{2} \frac{1}{|C_j|} (\tilde{\theta}_{C_j} - \hat{\theta}_P)^2 \tag{11}$$

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

$$K(X_i, x, (S_b^1, S_b^2)) = \frac{1[(X_i \in L_b(x)) \wedge (Z_i \in S_b^2)]}{|L_b(x) \cap S_b^2|} \tag{12}$$

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

$$K(X_i, x) = \frac{1}{B} \sum_{b=1}^{B} K(X_i, x, (S_b^1, S_b^2)) \tag{13}$$

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.

A.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:

$$\hat{\theta} = \arg\min_{\theta} \sum_{i:Z_i \in D_2} a_i(\theta \widetilde{T}_i - \widetilde{Y}_i)^2 \tag{14}$$

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 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

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

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.

B.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 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.

```
import numpy as np
from torch import nn
```

[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 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)
```

B.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).

B.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 single value - the final predication.[4]

[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.

[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

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
```

B.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 classification 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

day). The input in the top layer is the vector of 1,458 creator, category, and time fixed effects.

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.

B.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.

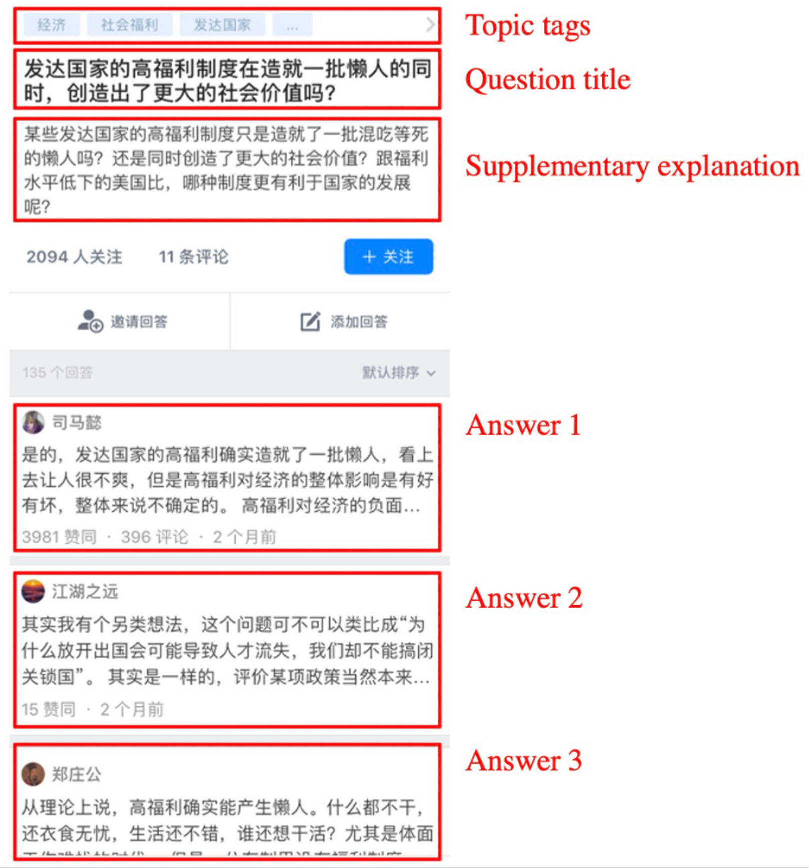
After enabling sample weights functionality for our SDNNs, we could now run the full ORF algorithm using the example code below. Since our treatment (i.e., price) is continuous, we call the class `ContinuousTreatmentOrthoForest` to perform ORF estimator. In the first two lines of the class, we specify hyperparameters that controls the tree-growing procedure for the forest kernel learner, in line with description in Section A.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 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

[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.

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)
```

C Examples of Zhihu Interfaces



经济 社会福利 发达国家 ...

发达国家的高福利制度在造就一批懒人的同时，创造出了更大的社会价值吗？

某些发达国家的高福利制度只是造就了一批混吃等死的懒人吗？还是同时创造了更大的社会价值？跟福利水平低下的美国比，哪种制度更有利于国家的发展呢？

2094 人关注 11 条评论

135 个回答

司马懿

是的，发达国家的高福利确实造就了一批懒人，看上去让人很不爽，但是高福利对经济的整体影响是有好有坏，整体来说不确定的。 高福利对经济的负面...

3981 赞同 · 396 评论 · 2 个月前

江湖之远

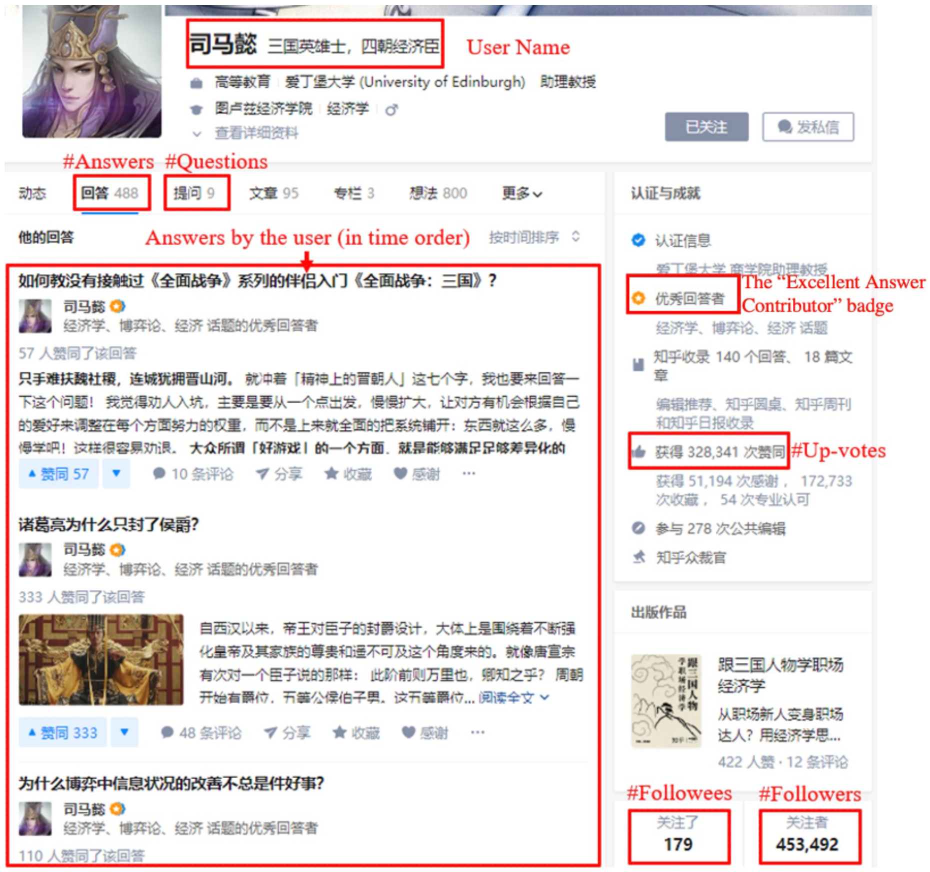
其实我有个另类想法，这个问题可不可以类比成“为什么放开出国会可能导致人才流失，我们却不能搞闭关锁国”。 其实是一样的，评价某项政策当然本来...

15 赞同 · 2 个月前

郑庄公

从理论上说，高福利确实能产生懒人。什么都不干，还衣食无忧，生活还不错，谁还想干活？尤其是体面

Figure A1: Example of a Question and Its Answers


User Name
#Answers
#Questions
Answers by the user (in time order)
The “Excellent Answer Contributor” badge
#Up-votes
#Followees
#Followers

司马懿 三国英雄士，四朝经济臣

高等教育 | 爱丁堡大学 (University of Edinburgh) 助理教授

图卢兹经济学院 | 经济学 | ♂

查看详细资料

已关注 发私信

动态 回答 488 提问 9 文章 95 专栏 3 想法 800 更多

他的回答 按时间排序

如何教没有接触过《全面战争》系列的伴侣入门《全面战争：三国》？

司马懿
经济学、博弈论、经济 话题的优秀回答者

57 人赞同了该回答

只手难扶魏社稷，连城犹拥晋山河。 就冲着「精神上的晋朝人」这七个字，我也要来回答一下这个问题！ 我觉得劝人入坑，主要是要从一个点出发，慢慢扩大，让对方有机会根据自己的爱好来调整在每个方面努力的权重，而不是上来就全面的把系统铺开：东西就这么多，慢慢学吧！这样很容易劝退。 **大众所谓「好游戏」的一个方面，就是能够满足足够差异化的**

▲ 赞同 57 ▼ 10 条评论 分享 收藏 感谢 ···

诸葛亮为什么只封了侯爵?

司马懿
经济学、博弈论、经济 话题的优秀回答者

333 人赞同了该回答

自西汉以来，帝王对臣子的封爵设计，大体上是围绕着不断强化皇帝及其家族的尊贵和遥不可及这个角度来的。就像唐宣宗有次对一个臣子说的那样： 此阶前则万里也，卿知之乎？ 周朝开始有爵位，五等公侯伯子男。这五等爵位... 阅读全文

▲ 赞同 333 ▼ 48 条评论 分享 收藏 感谢 ···

为什么博弈中信息状况的改善不总是件好事?

司马懿
经济学、博弈论、经济 话题的优秀回答者

110 人赞同了该回答

认证与成就

认证信息
爱丁堡大学 商学院助理教授

优秀回答者
经济学、博弈论、经济 话题

知乎收录 140 个回答、 18 篇文章
编辑推荐、知乎圆桌、知乎周刊和知乎日报收录

获得 328,341 次赞同
获得 51,194 次感谢， 172,733 次收藏， 54 次专业认可

参与 278 次公共编辑

知乎众裁官

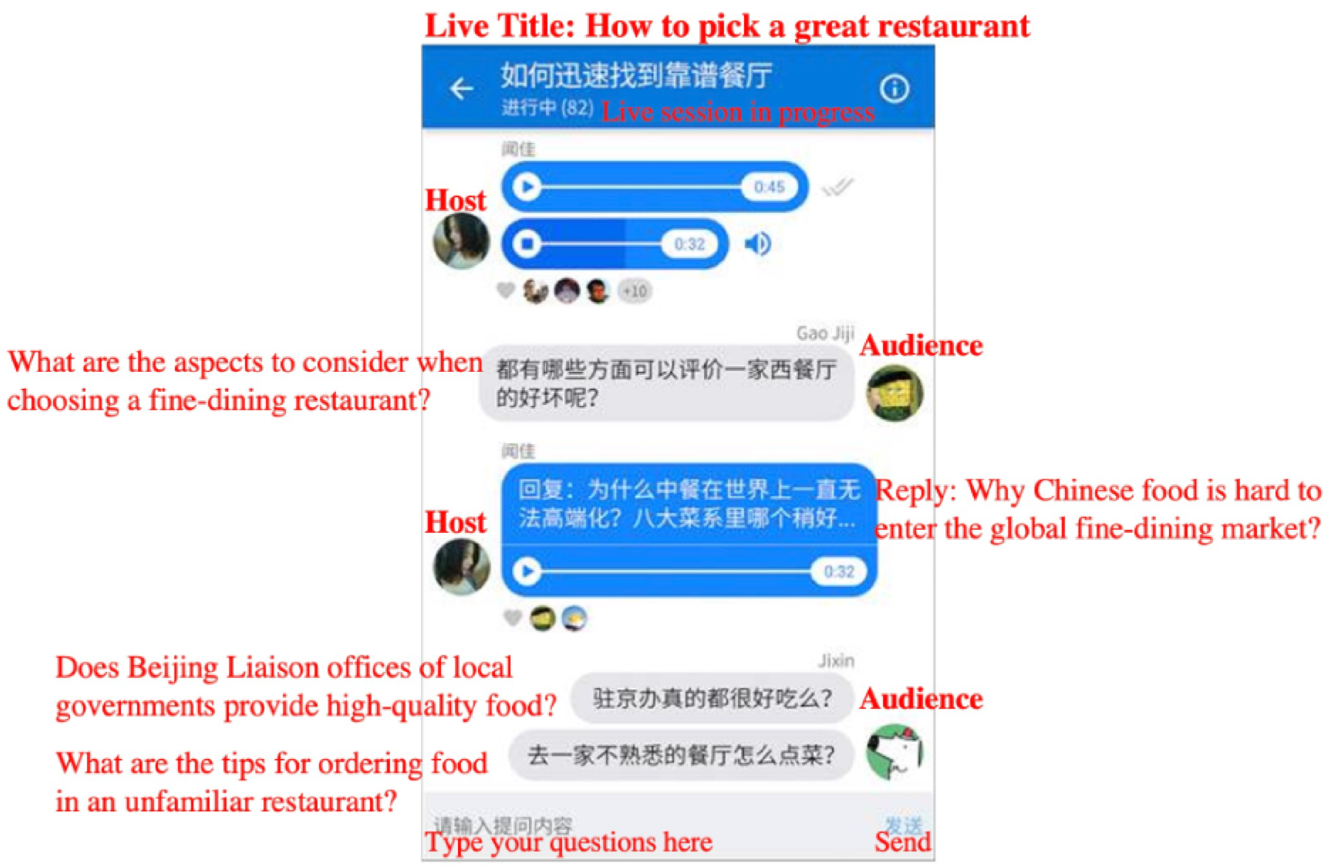
出版作品

跟三国人物学职场经济学
从职场新人变身职场达人？用经济学思...
422 人赞 · 12 条评论

关注了 179

关注者 453,492

Figure A2: Example of User Profile Page



如何迅速找到靠谱餐厅

都有哪些方面可以评价一家西餐厅的好坏呢?

What are the aspects to consider when choosing a fine-dining restaurant?

回复：为什么中餐在世界上一直无法高端化? 八大菜系里哪个稍好...

Reply: Why Chinese food is hard to enter the global fine-dining market?

驻京办真的都很好吃么?

Does Beijing Liaison offices of local governments provide high-quality food?

去一家不熟悉的餐厅怎么点菜?

What are the tips for ordering food in an unfamiliar restaurant?

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.

D Applying Generalized ORF for 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).

D.2 Performance Comparison between Nuisance Functions using SDNNs and Benchmark Methods

We apply the SDNN defined in Appendix B to estimate nuisance functions for log price and log demand, respectively. We compare the performance of SDNN with two benchmark models: lasso and standard DNNs. Table A1 reports related statistics.

Table A1: Fitting Comparison Between the SDNN and Benchmark Methods

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

Note: The training(testing) set contains 80%(20%) Live-day observations. The DNN is implemented with the MLPregressor provided in sklearn library, which has a similar architecture to the SDNN but without the parametric component. The average value of log price and log daily sales is 2.89 and 1.50, respectively.

One can see that the SDNN and DNN achieve much higher prediction accuracy than lasso in both demand and price equation. The difference in out-sample R^2 and MSE between the two types of model could be as large as 27%. This suggests that, as we expected, 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 Robustness Checks

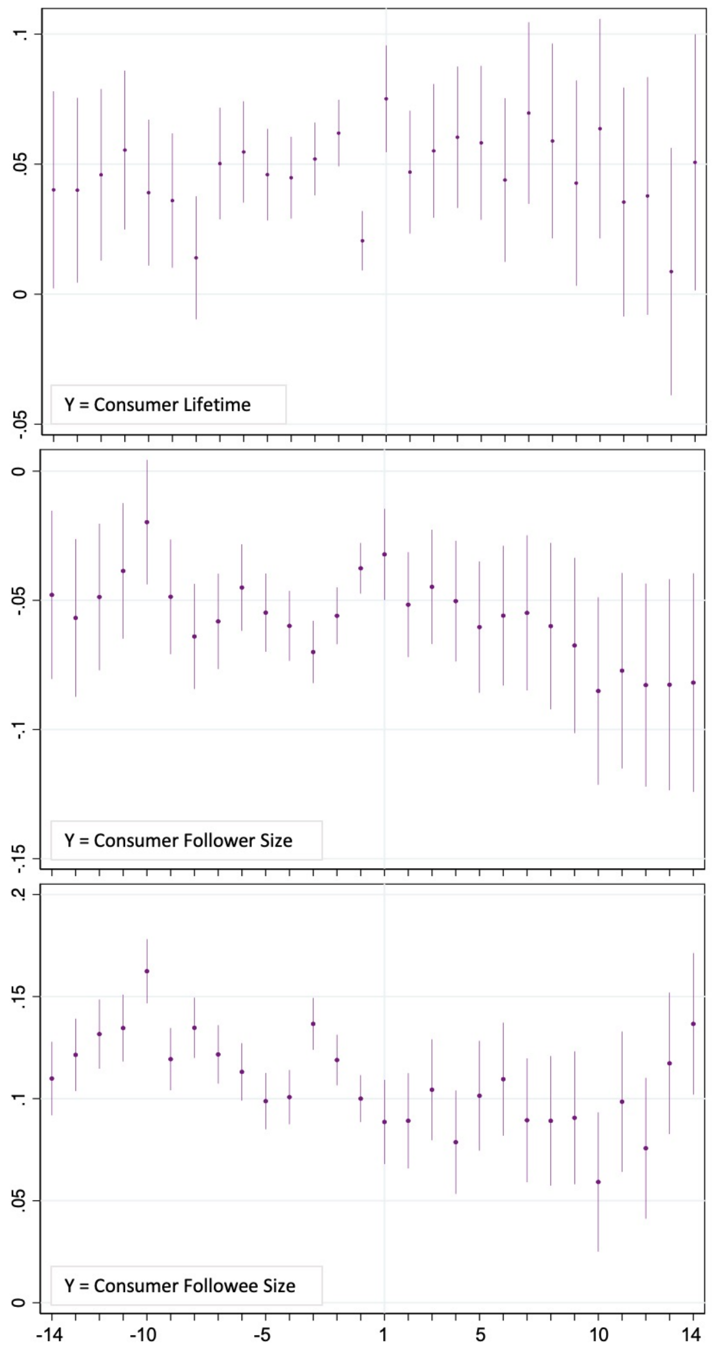


Figure A4: Temporal Distance Fixed Effects in Consumer Characteristics Regressions

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.

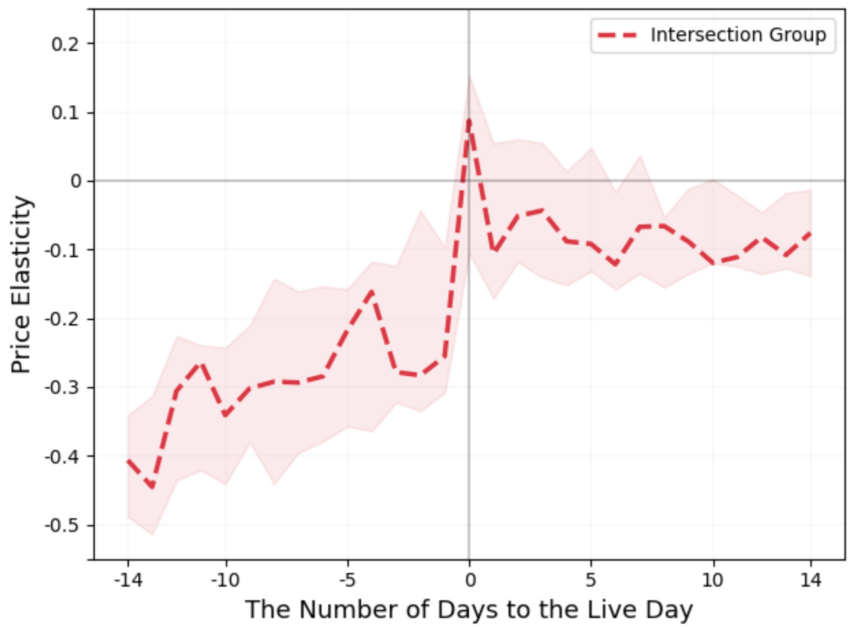


Figure A5: Price Elasticity over Time to Scheduled Live Day (Excluding Extreme consumers)

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