

Pricing a Short-Term Rental

Estimating Demand, and Turning the Estimate into a Nightly Rate

Mathematics & Physical Sciences Series

Abstract—A 2023 note derived the price that maximizes a vacation rental’s revenue per night: the price at which occupancy falls by one percent for each percent the price rises. It stopped at the practical problem, which is that the demand curve is unknown. This tutorial completes the model. It writes demand as a booking probability per night and per day on sale, which depends on price, on how far ahead the stay is, and on season, weekday and events. It shows why a host’s own history gives the wrong answer when read naively. In a simulated market whose true price sensitivity is \$110, the month-by-month scatter of occupancy against price slopes upward, and a regression with seasonal controls estimates \$226. A random daily price nudge of $\pm 15\%$ costs about 1.0% of profit, and a two-stage fit on the nudged data recovers \$97 $\pm$ 18. A dynamic program then sets each night’s price for each day before the stay. With the experimental estimate it earns 98.7% of the profit available with perfect knowledge of demand. With the biased estimate it earns 74.8%, and the host’s rule of thumb earns 84.3%. One listing learns its own price sensitivity slowly: a $\pm 10\%$ interval takes over ten years of its own data. Profit is flat near the optimum, though, so an estimate within 10% loses under 2%, and a prior taken from comparable listings closes most of the gap in the first year.

Index Terms—Revenue management, dynamic pricing, demand estimation, short-term rentals, discrete-time hazard models, endogeneity, randomized experiments, Thompson sampling.

CONTENTS

I	Introduction	1
I-A	The running example	2
I-B	The program	2
I-C	How to read this tutorial	2
II	The 2023 Model and What It Leaves Open	2
II-A	The demand curve and the optimum	2
II-B	Two refinements	2
II-C	What the model leaves open	3
III	A Demand Model with Time in It	3
III-A	Shoppers and willingness to pay	3
III-B	Season, weekday, events and lead time	3
III-C	The booking curve	3
III-D	The 2023 curve as a special case	3
III-E	Revenue management	3
IV	The Price for Each Day Before the Stay	3
IV-A	The recursion	3
IV-B	The best price in closed form	4
IV-C	The cabin’s price paths	4
IV-D	Floors and the value of an unsold night	4

A tutorial in the *Mathematics & Physical Sciences* series. It continues a 2023 note on the optimal price of a vacation rental, which derived the revenue-maximizing condition and stopped at the question of how to estimate the demand curve. Every number in this volume comes from a simulated market in `examples/strprice.py` beside this document, which the build runs; a test checks the claims the text makes about its output. The market is simulated so that the true demand curve is known and each estimator can be checked against it. No real listing’s data is used. This is a modelling tutorial; it gives no financial advice.

Rev. 5307d5685, built 2026-10-30 08:44 UTC.

V	The Data a Host Has	4
V-A	The at-risk table	4
V-B	Censoring	4
V-C	Several nights per booking	5
V-D	Data about other listings	5
VI	Why the Obvious Estimates Are Wrong	5
VI-A	The host’s rule	5
VI-B	Occupancy against ADR	5
VI-C	The hazard model on the same data	5
VI-D	What the wrong estimate costs	5
VI-E	Three ways out	6
VII	A Price Experiment	6
VII-A	The nudge	6
VII-B	The mistake: fitting as before	6
VII-C	The two-stage estimator	6
VII-D	What the experiment costs	6
VIII	Comparing the Pricing Policies	6
VIII-A	The policies	7
VIII-B	What the comparison shows	7
IX	How Much Data Is Enough	7
IX-A	How close is close enough	7
IX-B	How fast one listing learns	7
IX-C	What follows	7
X	Starting with Little Data	8
X-A	A prior from comparable listings	8
X-B	Pooling	8
X-C	Thompson sampling	8
X-D	The cabin’s first year	8
XI	Checking the Model	8
XI-A	Bookings by lead time	9
XI-B	The check that tests the price	9
XI-C	Testing a pricing policy live	9
XII	Stays, Gaps and Fees	9
XII-A	Cleaning and the length of stay	9
XII-B	Minimum stays and gap nights	9
XII-C	The floor price	9
XIII	A Weekly Routine for One Host	9
XIV	What the Model Leaves Out	10
XV	Conclusion	10
	References	10

I. INTRODUCTION

A HOST who rents a house by the night sets a price for each night of the year, and changes it as the night approaches. Raise the price and fewer guests book; lower it and the house fills with guests who would have paid more. Commercial pricing tools adjust the price automatically, but they do not publish how they decide.

A 2023 note set out to derive the best price from first principles [1]. It wrote revenue per available night as price times occupancy, assumed a fixed demand curve linking the two, and found the price that maximizes the product. Its conclusion

named the open problem: “The practical difficulty here is estimating the demand curve D .” This tutorial picks the work up at that point. It answers three questions:

- 1) What should the demand curve of a rental night look like, once time is taken into account?
- 2) How can a host estimate it from the data a host has, without being misled?
- 3) How does the estimate become a price for each night, on each day before the stay?

A. The running example

The running example is a fictional listing, *the cabin*: a house in a lake town, busiest in July and quietest in January. Each night of the year goes on sale 120 days ahead. The cabin’s market is simulated, so its true demand curve is known. That is the reason for using a simulation: every estimator in the text can be checked against the truth, which is never possible with a real listing. Two nights recur throughout: a Thursday in mid-January with no event, and a Saturday in mid-July with no event. The text calls them the January night and the July night.

The cabin’s numbers were chosen to look like a plausible leisure rental. They describe no real listing. Its demand is 3.0 times as high at the peak of summer as in January, 1.42 times as high on Friday and Saturday nights, and 2.2 times as high on 10 event nights a year. Under the host’s own rule of thumb it fills 87% of its nights at an average rate of \$188.

B. The program

A program beside this document, `examples/strprice.py`, simulates the market, generates the data a host would see, fits every estimator and runs every pricing policy. It uses only the Python standard library and fixed random seeds, so a rerun gives the same numbers. The build runs it and reads its numbers into the text, and a test checks each claim the text makes about them. Changing an assumption in the program changes the numbers in the next build of this document.

C. How to read this tutorial

Section II restates the 2023 note’s model and result. Section III replaces its single demand curve with a model of bookings over time, and Sec. IV derives the price for each day before the stay. Sections V to VII are the core: what data a host has, why the obvious estimates are wrong, and how a price experiment fixes them. Section VIII compares the pricing policies, Sec. IX asks how much data is needed, and Sec. X shows how to start without any. Section XI covers validation, Sec. XII covers stays of several nights and cleaning fees, and Sec. XIII gives a weekly routine a single host can run. Section XIV lists what the model leaves out.

II. THE 2023 MODEL AND WHAT IT LEAVES OPEN

The 2023 note uses two quantities [1]. The *average daily rate* (ADR) x is the mean price of the nights sold, in dollars. The *occupancy* y is the fraction of available nights that sold, between 0 and 1. Their product is the *revenue per available night*, which hotels call RevPAR:

$$R(x, y) = xy. \quad (1)$$

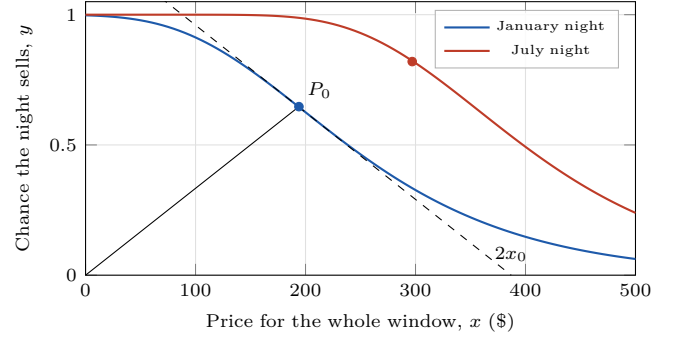


Fig. 1. The 2023 note’s rule on two of the cabin’s nights, each held at one price for its whole 120-day window. The solid line from the origin and the dashed tangent make equal angles with the price axis at the January optimum P_0 (\$194, 65%); the tangent meets the axis at \$387. The July optimum is \$297 at 82%.

A. The demand curve and the optimum

The note assumes that occupancy and price lie on a fixed curve $D(x, y) = 0$. It asks three things of the curve: occupancy is 1 at a price of zero, occupancy is zero above some large price, and occupancy falls as the price rises. Writing the curve as $y(x)$, the note maximizes xy on it with a Lagrange multiplier and finds

$$\frac{y}{x} = -\frac{dy}{dx}. \quad (2)$$

Multiplying both sides by x/y gives the same condition in the economist’s form. The *price elasticity* of occupancy, $\varepsilon = (x/y) dy/dx$, is the percentage change in occupancy for a one percent rise in price. At the optimum it is -1 . Above that price, a one percent rise loses more than one percent of the bookings, and revenue falls. Below it, the opposite holds.

The note also gives the condition a picture. The left side of (2) is the slope of the ray from the origin to the optimum $P_0 = (x_0, y_0)$. The right side is the negative slope of the tangent there. The two lines make equal angles with the price axis, so the origin, P_0 and the point where the tangent meets the axis form an isosceles triangle, with that point at $2x_0$ (Fig. 1).

B. Two refinements

Two small additions make the rule fit a rental. The first is cost. A sold night costs the host money: cleaning, laundry, utilities and the platform’s fee. Call the cost of a sold night c . The host then maximizes the profit $(x - c)y$, and the condition becomes

$$\frac{y}{x - c} = -\frac{dy}{dx}. \quad (3)$$

The tangent now meets the price axis at $2x_0 - c$, and the optimal price rises. With the cabin’s cost of \$30 a night, the January optimum moves from \$194 to \$209. The note mentions a published derivation of this form, which reduces to (2) at $c = 0$ [2].

The second is uniqueness. The note argues that the optimum is unique, but its argument needs one more assumption. Condition (2) can hold at several prices when the elasticity rises and falls again along the curve. It has a single solution when the magnitude of the elasticity grows steadily with price. That holds for straight-line demand and for the model used in this tutorial (Sec. IV).

C. What the model leaves open

The note's model is a single curve, fixed in time. A rental night differs from that picture in three ways.

- **A night is on sale for months.** Guests book at different times before the stay, and the host can change the price on every one of those days.
- **A night is perishable.** A night that has not sold by the date of the stay is worth nothing. Airlines and hotels face the same problem, and the methods built for them are called *revenue management* [3].
- **Every night has its own curve.** A July Saturday and a January Thursday face different numbers of guests.

The fourth gap is the one the note named: the curve has to be estimated. The next section writes a model that covers the three differences, and Sec. V begins the estimation.

III. A DEMAND MODEL WITH TIME IN IT

The model in this section describes one night and one day at a time. It counts the people who look at the night on that day, and gives each of them a chance to book at the price shown. Everything a host can estimate later is a parameter of this model.

A. Shoppers and willingness to pay

Call the night of the stay d and the number of days before it the *lead time* t , so $t = 0$ is the day of the stay itself. On each day, some number of *shoppers* look at night d : people searching the platform for those dates in that town. Their number is random, with mean $\Lambda_d f(t)$. The factor Λ_d is the night's total interest, and $f(t)$ spreads it over the 120 days on sale.

Each shopper has a *willingness to pay*: the highest price at which that shopper would book this house over the alternatives. A shopper books if the price is below it. The model takes the willingness to pay to be exponentially distributed, so a fraction

$$p(x) = e^{-x/\theta} \quad (4)$$

of shoppers would book at price x . The parameter θ , in dollars, is the *price sensitivity* that the rest of this tutorial tries to estimate. A larger θ means guests care less about price. For the cabin $\theta = \$110$: 37% of shoppers would pay \$110 or more.

If shoppers arrive as a Poisson process, the chance that at least one of them books on that day is

$$h(x, t) = 1 - \exp(-\Lambda_d f(t) e^{-x/\theta}). \quad (5)$$

This is the *booking hazard*: the chance that a night still unsold on the morning of day t sells that day. It is the quantity the data can estimate, one night and one day at a time.

B. Season, weekday, events and lead time

Taking logarithms puts every effect in one linear expression. Write

$$\eta = \log \Lambda_d + \log f(t) - \beta x, \quad \beta = 1/\theta, \quad (6)$$

so that $h = 1 - \exp(-e^\eta)$. The cabin's model sets

$$\log \Lambda_d = \mu + a \cos \frac{2\pi d}{365} + b \sin \frac{2\pi d}{365} + w \text{wkd}_d + e \text{event}_d + \nu_d, \quad (7)$$

$$\log f(t) = -t/\tau + \text{const.} \quad (8)$$

The cosine and sine terms describe one seasonal cycle with its peak on any date. wkd_d is 1 on Friday and Saturday nights, and event_d is 1 on a festival or holiday night. The shock ν_d

is everything else that makes one night busier than another: a wedding in town, a conference, a heat wave. Its standard deviation is 0.30, so a typical night is about 35% busier or quieter than its season predicts. The scale $\tau = 30$ days sets how early shoppers look: half of them arrive within the last 21 days.

This form is a discrete-time proportional hazards model, and the link $h = 1 - \exp(-e^\eta)$ is called the *complementary log-log* (cloglog) link [9], [10]. Its practical value is that η is linear in its features. A standard maximum-likelihood fit estimates μ , a , b , w , e , τ and β together.

C. The booking curve

Applying the hazard day after day gives the *booking curve*: the chance a night has been booked by each lead time. Under the best prices of Sec. IV, the July night is 31% likely to be booked 30 days ahead and 88% by the day of the stay. The January night reaches 22% and 63% (Fig. 2, lower panel). Hosts read booking curves to judge whether a night is selling faster or slower than usual. In this model the curve follows from Λ_d , $f(t)$ and the prices.

D. The 2023 curve as a special case

The note's single demand curve is what this model gives when the price stays the same for the whole window. The chance that the night sells at all is then

$$y(x) = 1 - \exp(-K_d e^{-x/\theta}), \quad K_d = \Lambda_d \sum_t f(t), \quad (9)$$

where K_d is the expected number of shoppers over the whole window. Figure 1 plots this curve for the two nights. When few shoppers come, $y \approx K_d e^{-x/\theta}$, its elasticity is $-x/\theta$, and condition (2) gives $x_0 = \theta$. A quiet night's best fixed price is the price sensitivity itself. Busier nights saturate, and their best price is higher: with no cost, the January night ($K = 6.05$) is best at \$194 and the July night ($K = 25.76$) at \$297.

E. Revenue management

A night that perishes on a fixed date, sold to customers who arrive over time at a price the seller can change, is the standard problem of revenue management [3], [5]. Airlines developed most of the field, for seats, from the 1970s on. Two of its results matter here. The value of an unsold unit is the revenue it can still earn, which falls as the date approaches [6]. And the best price for a single unit follows from a short dynamic program [4]. The next section applies both to the cabin.

IV. THE PRICE FOR EACH DAY BEFORE THE STAY

With the hazard (5) known, the best price for every day of a night's window follows from a dynamic program. This section derives it for one night. Nights are treated as independent here; Sec. XII adds stays of several nights.

A. The recursion

Let W_t be the expected profit still to come from the night if it is unsold on the morning t days before the stay. On that day the host shows price x . With probability $h(x, t)$ the night sells and earns $x - c$; otherwise it stays unsold, and tomorrow it is worth W_{t-1} . The host picks the price that maximizes the sum:

$$W_t = \max_x \left[h(x, t) (x - c) + (1 - h(x, t)) W_{t-1} \right], \quad (10)$$

with $W_{-1} = 0$, because a night not sold by its date earns nothing. The program starts at $t = 0$ and works backwards to the first day of sale. The rearranged form

$$W_t = W_{t-1} + \max_x h(x, t) (x - c - W_{t-1}) \quad (11)$$

shows what the price is for. Selling today gives up the chance of selling later, which is worth W_{t-1} . So W_{t-1} is an *opportunity cost*, and it adds to the cost of the night: the host should accept a booking only at a price above $c + W_{t-1}$. This is the same reasoning airlines use to decide whether to sell a cheap seat now or hold it for a later passenger who pays more [6].

B. The best price in closed form

Write $k = \Lambda_d f(t)$ for the day's expected shoppers and $z = ke^{-x/\theta}$ for the expected number who would book at x . Setting the derivative of (10) to zero gives

$$x^* = c + W_{t-1} + \theta q(z), \quad q(z) = \frac{e^z - 1}{z}. \quad (12)$$

The price has three parts: the cost of the night, the opportunity cost of selling it today, and a markup. On a quiet day z is small, $q(z) \approx 1$, and the markup is θ . That is the 2023 result $x_0 = \theta$ again, now with the two costs added. When many shoppers would book, $q(z)$ grows and so does the markup. The right side of (12) falls as x rises, because z falls, so the equation has exactly one solution. The program finds it by Newton's method in a few steps.

C. The cabin's price paths

Figure 2 shows the result for the two nights. The January night opens at \$249 120 days out, is at \$222 a month out, and falls to \$143 on the day. The July night opens at \$372 and ends at \$153. Early in the window W_{t-1} is large, because the night still has many days of shoppers ahead of it. As the date approaches, fewer selling days remain, the opportunity cost shrinks, and the price falls with it. On the last day the price is c plus the markup alone.

When the night opens, its expected profit W is \$109 in January and \$232 in July. These are the values a host should compare with any offer for the night, such as a request for a discount on a long stay (Sec. XII).

D. Floors and the value of an unsold night

Two practical settings fit the recursion directly. A *floor price* is a price the host will not go below. Because the expression maximized in (10) rises and then falls in x , the best price under a floor is simply $\max(x^*, \text{floor})$. A floor costs money only on the days when it binds, which in the cabin's case are the last days before quiet nights.

The second setting is the value of a night that does not sell. The recursion uses $W_{-1} = 0$. A host who would otherwise stay in the house, or who saves wear by leaving it empty, can set W_{-1} to that value. The opportunity cost then never falls below it, and late prices stay higher.

V. THE DATA A HOST HAS

Estimating the model needs one record per night per day on sale. Most of that record is easy to collect. One part is not, and a host who wants to estimate demand later has to start recording it now.

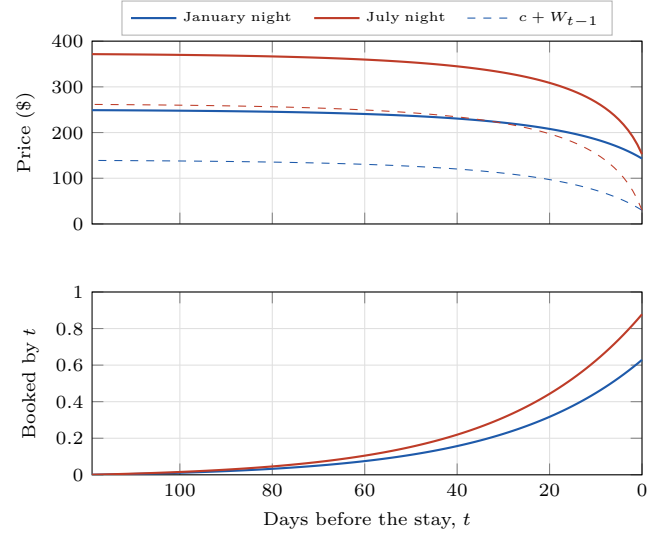


Fig. 2. Top: the best price for each day of the window, from (12), with the floor that the opportunity cost sets (dashed). Time runs left to right towards the stay. Bottom: the booking curve, the chance that the night has been booked by each day, at those prices.

A. The at-risk table

The estimator works on a table with one row for each night and each day on which that night was for sale and still unsold. A row holds:

- the night's features: date, weekday, event flag;
- the lead time t ;
- the price shown that day;
- whether the night was booked that day (1) or not (0).

A night booked 40 days ahead contributes 79 rows with a 0, from day 119 to day 41, and one row with a 1 at day 40. A night that never sells contributes 120 rows, all 0. Statisticians call such a night *at risk* on each day it could still be booked, and the table is a *person-period* table in the survival-analysis literature [10]. Two years of the cabin's history under its host's rule make 60,910 rows.

The bookings come from the platform's reservation export, which lists the booking date, the dates of the stay and the payout. The export has no record of the prices shown on the days when nothing booked: a calendar holds its current prices only. A host who wants to estimate demand therefore needs a daily snapshot of the calendar: for every future night, its price and whether it is open. A small script run once a day, or the price history of a pricing tool, provides it.

Pitfall. Nights the owner blocks for personal use are not unsold nights. They leave the at-risk table from the day they were blocked. Counting them as nights that failed to sell makes demand look lower than it is.

B. Censoring

A booked night hides part of its demand. Once the January night books at day 40, nobody learns how many shoppers would have booked it on days 39 to 0, or at what prices. Its data stop at the booking: they are *censored*. A night that never sells shows the opposite: no shopper on any day was willing to pay the price shown. That sets an upper bound on the shoppers' willingness to pay. It does not measure it.

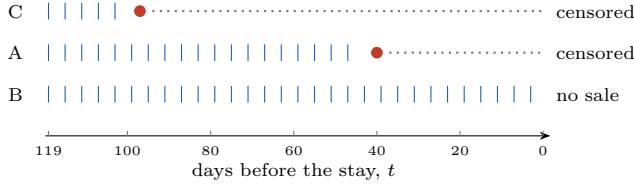


Fig. 3. Rows of the at-risk table for three nights. Each tick is a day on sale with no booking (a 0 row, one tick drawn per four days); a dot is the day of the booking (a 1 row). After a booking the night has no more rows. Night B's rows say only that nobody booked at the prices shown.

The hazard model handles both cases because each row asks only what happened on that day to a night still unsold. The booked night's rows after day 40 never enter the table, so they bias nothing. The unsold night's rows each record a day on which nobody booked at the price shown, and that is exactly what they mean. Figure 3 draws the three cases.

A common shortcut breaks this. Counting the fraction of nights that sold at each price uses each night once, at one price, and loses the days. A night priced at \$300 that sold 90 days ahead and a night priced at \$300 that sold on the day count the same, though they say very different things about demand.

C. Several nights per booking

Most bookings cover several nights. The simplest treatment, used in this tutorial, gives each night of the stay its own row with a 1 at the same lead time. The nights of one booking are then not independent events, which makes the estimated standard errors somewhat too small. Section XII returns to stays of several nights.

D. Data about other listings

A single listing produces a few hundred bookings a year. Comparable listings nearby add much more. Inside Airbnb publishes snapshots of public listing data for many cities: the listings, their reviews, and a calendar that says for each future date whether it is open [17]. A date that changes from open to closed between two snapshots was probably booked, though an owner's block looks the same. Commercial market-data services sell similar figures, with occupancy and rates by date for a town. Such data serve two purposes in this tutorial: a prior for the listing's own parameters (Sec. X), and a signal of how busy the town is on each date. They do not give the listing's own price sensitivity directly, because listings differ in more than price. A larger, newer house costs more and also books better, and the same confusion between price and demand that Sec. VI describes appears across listings.

VI. WHY THE OBVIOUS ESTIMATES ARE WRONG

The first estimate most hosts would try is the one the 2023 model suggests: plot occupancy against ADR and fit a line. This section runs it on two years of the cabin's history, then runs the hazard model on the same data, and shows that both give the wrong answer for the same reason.

A. The host's rule

The cabin's host prices by a rule of thumb. The rule starts from a base rate, raises it for summer, weekends and events, raises it further when the town looks busy, and cuts 15% in

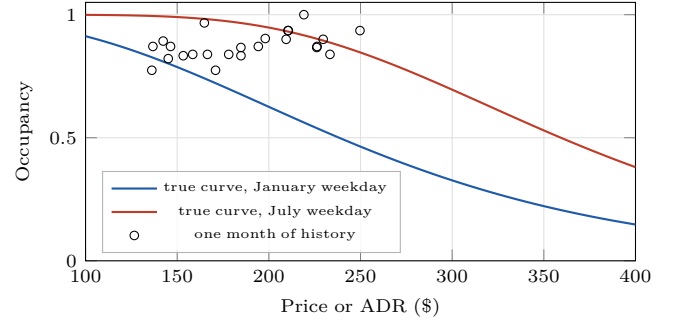


Fig. 4. The cabin's history under its host's rule, one point per month for two years. The points rise to the right, but every true demand curve falls. The host charges more in the months when more guests come, so the points trace the host's pricing rule.

the last week. “The town looks busy” is the host's reading of the shock ν_d : the neighbours' calendars filling up, a festival announced. The host reads it with error, but the reading is correlated with the truth. Under this rule the cabin fills 87% of its nights at an ADR of \$188.

B. Occupancy against ADR

Each month of the two years gives one point: that month's ADR and occupancy. The ADR ranges from \$136 to \$250, and occupancy from 77% to 100%. A straight line through the 24 points rises by +0.7 points of occupancy per \$10 of ADR (Fig. 4). Read as a demand curve, it says that guests book more when the price is higher, and condition (2) has no solution at all.

The line is not a demand curve. Every month has its own curve, and each point lies on a different one. The host raised the price in the months when more guests came, so the points show how the host prices. Economists have known this since 1927 [7]: a scatter of observed prices and quantities traces a demand curve only if the demand stayed put while the price moved.

C. The hazard model on the same data

The hazard model of Sec. III uses every night and every day, and it can include the season and weekday as controls. Fitted with price and lead time alone, it estimates $\theta = \$591$: price barely seems to matter. Adding the seasonal terms, the weekend and the event flag gives $\theta = \$226$ with a standard error of \$40. The true value is \$110, 2.9 standard errors away.

The controls remove the part of the bias that the analyst can see. The host's price also reacts to ν_d , which the analyst cannot. On a night with a positive shock the host charges more, and the night still sells well, because more shoppers came. The fit attributes those extra bookings partly to the higher price, and concludes that guests mind price less than they do. Econometricians call this *endogeneity*: the price is set in response to the same unobserved demand that drives bookings [8]. Its direction here is always the same. A host who raises prices when demand is high makes demand look less sensitive to price than it is.

D. What the wrong estimate costs

A pricer that believes $\theta = \$226$ charges too much. Run through the dynamic program of Sec. IV, it sets an ADR of \$356 and fills 37% of the nights. It earns 74.8% of the profit available with the true demand curve, which is less than the

host's own rule earns (Sec. VIII). A careful-looking model on biased data does worse than a rule of thumb.

E. Three ways out

The bias needs price variation that the host did not set in response to demand. There are three sources of it.

- **Better controls.** If the model includes everything the host reacted to, the price variation that remains is independent of demand. A host who records the signals used, such as the town's occupancy for that date from comparable listings, can add them as features. In practice some of what the host reacts to always goes unrecorded.
- **An instrument.** An *instrument* is a variable that moves the price but affects bookings only through the price [8]. A change in cost that the host passes on, such as a new cleaning contract, is the textbook example. Such changes are rare for one listing.
- **An experiment.** The host can create the variation directly, by moving prices at random. This is the only one of the three that the host fully controls, and the next section develops it.

VII. A PRICE EXPERIMENT

A host can break the link between price and demand by moving prices at random. This section describes the experiment, shows a mistake that makes even experimental data give the wrong answer, and gives the estimator that avoids it.

A. The nudge

The experiment keeps the host's usual price as a plan, and each day multiplies each open night's planned price by e^u . The *nudge* u is drawn at random, uniformly between -15% and $+15\%$, independently for every night and every day. The geometric mean of the price stays at the plan. The arithmetic mean rises by a factor of about $1 + \delta^2/6$ for a nudge of $\pm\delta$, which is 0.4% for $\pm 15\%$. The record for each row of the at-risk table gains one field: the planned price, beside the price actually shown.

Daily nudges, drawn afresh each day, keep the nudge independent of everything else in a row. A nudge fixed for a night's whole window would not be. A night with a high nudge tends to stay unsold longer, so it appears in more rows at short lead times. The nudge would then be correlated with lead time.

B. The mistake: fitting as before

Two years of nudged history give 61,072 rows. Fitted exactly as in Sec. VI, with the seasonal and event controls, they give $\theta = \$245$ with a standard error of $\$45$. The experiment has barely helped.

The reason is that the fit uses all the variation in price, and most of it still comes from the host's plan. The plan reacts to the unobserved shock ν_d as before. A $\pm 15\%$ nudge adds random variation on top, but the confounded variation still dominates the fit.

C. The two-stage estimator

The fix is to make the nudge the only price variation the price coefficient can learn from. It takes two fits.

- 1) **Stage 1.** Fit the hazard model with the logarithm of the planned price as an extra feature. The planned price absorbs everything the host reacted to, including the host's reading

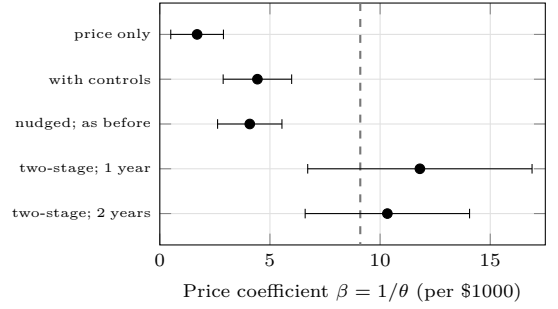


Fig. 5. Estimates of the price coefficient with their 95% intervals. The coefficient is $1/\theta$, so a smaller value means guests care less about price. The first three use the host's own price variation and land far below the truth (dashed). The two-stage estimator, which learns only from the random nudge, covers it.

of ν_d . The price coefficient can then learn only from the difference between the shown and the planned price, which is the random nudge.

- 2) **Stage 2.** Hold the price coefficient at its stage-1 value and refit the remaining coefficients without the planned price.

Stage 2 is needed because stage 1 gets the other coefficients wrong. The planned price rises on event nights, so in stage 1 it absorbs most of the event effect: the fitted event multiplier is 0.9, against a true 2.2. Stage 2 removes the planned price, and the remaining features (season, weekday, events) are not correlated with the unobserved shock. Only the price was. With the price coefficient fixed at an unbiased value, they are estimated without bias. The event multiplier comes out at 1.7, the weekend multiplier at 1.57 against a true 1.42, and the lead-time scale at 30 days against a true 30. The event and weekend values differ from the truth by about their sampling error, with only 20 event nights in two years.

The two-stage estimate of the price sensitivity is $\theta = \$97$, with a standard error of $\$18$. The truth, $\$110$, lies inside its 95% interval. One year of nudged history alone gives $\$85 \pm \19 , also consistent with the truth. Figure 5 compares all the estimates.

D. What the experiment costs

A nudge moves the price away from the best one, so it costs some profit. The cost is small, because profit is flat near the optimum and the loss grows with the square of the nudge. Applied to the best prices of Sec. IV, a nudge of $\pm 5\%$ gives up 0.1% of profit, $\pm 10\%$ gives up 0.4%, $\pm 15\%$ gives up 1.0% and $\pm 20\%$ gives up 1.8%. What a nudge teaches also grows with the square of its width (Sec. IX). A host who relies on the nudge to learn should make it as wide as the guests and the listing's image allow; $\pm 15\%$ to $\pm 20\%$ is a reasonable range.

Recipe 1 (Running the experiment). Each morning, for every open night: compute the planned price by whatever method is in use; draw u uniformly from $[-0.15, 0.15]$; set the price to the plan times e^u , rounded to the dollar; and log the night, the day, the plan and the price shown. Fit with the two-stage estimator. Never let the nudge depend on how the night is selling.

VIII. COMPARING THE PRICING POLICIES

This section runs seven pricing policies on the same 20 simulated years of the cabin's market and compares their profit. Each year has its own event nights and its own shocks, which the policies do not see in advance. Profit is computed as an

TABLE I
SEVEN PRICING POLICIES ON 20 SIMULATED YEARS. PROFIT IS PER AVAILABLE NIGHT, AFTER THE \$30 COST OF EACH SOLD NIGHT.

Policy	ADR	Occ.	Profit	% of ref.
Best fixed price	\$260	66%	\$153	94.1
Host's rule	\$187	87%	\$137	84.3
2023 rule, true curve	\$252	70%	\$156	96.0
Dynamic, biased	\$356	37%	\$121	74.8
Dynamic, experimental	\$229	81%	\$160	98.7
Dynamic, true	\$245	75%	\$162	100.0
Dynamic, with shock	\$246	76%	\$163	100.8

expected value, night by night, under the true demand, so the comparison has no sampling noise from the bookings themselves.

A. The policies

- **Best fixed price:** one price for every night and every day, \$260, chosen with knowledge of the true demand.
- **Host's rule:** the rule of thumb of Sec. VI.
- **The 2023 rule:** one price per night for its whole window, from (3), with the true demand curve.
- **Dynamic, biased:** the program of Sec. IV with the estimate from the host's own history ($\theta = \$226$).
- **Dynamic, experimental:** the same program with the two-stage estimate from two years of nudged history ($\theta = \$97$).
- **Dynamic, true:** the same program with the true demand model. This is the reference for the comparison.
- **Dynamic, with the shock:** the true model, also told each night's shock ν_d in advance. No host can do this; it shows how much the unpredictable part of demand is worth.

B. What the comparison shows

Table I gives the results. Four findings stand out.

The experimental estimate is good enough. The dynamic program with the two-stage estimate earns 98.7% of the reference. That is 17% more than the host's rule, or \$8,523 a year on the cabin's 365 nights.

The biased estimate is worse than no model. With θ estimated from the host's own history, the same program earns 74.8%, below the host's rule. It charges an ADR of \$356 and leaves most nights empty.

Most of the gain comes from the price level. The best single price for the whole year already earns 94.1%. Setting one price per night by the 2023 rule, with the true demand curve, earns 96.0%. Changing the price day by day adds the last 4.2%. The demand estimate matters most because it sets the level of prices; the dynamic program refines it.

The host's rule underprices. It fills 87% of the nights at an ADR of \$187, where the reference fills 75% at \$245. High occupancy feels like success. At the optimum, though, about a quarter of the cabin's nights stay empty. Filling them would need a lower price, which would lose more on the other nights than it gains.

The shock adds 0.8% when the pricer knows it. In this model, knowing each night's surprise in advance is worth little once the season, weekday and events are priced correctly. A real market may have larger shocks than the cabin's; Sec. XIV returns to this.

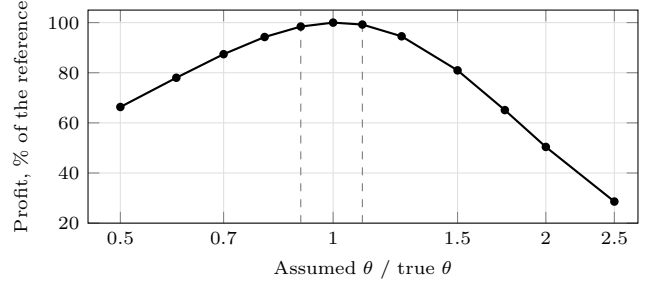


Fig. 6. Profit of the dynamic program when it assumes the wrong price sensitivity. Between the dashed lines ($\pm 10\%$) it loses less than 2%.

IX. HOW MUCH DATA IS ENOUGH

An estimate of θ is never exact. This section asks two questions: how close it has to be, and how long one listing needs to get that close from its own data.

A. How close is close enough

Figure 6 runs the dynamic program with a wrong θ and the true values of everything else. Profit is flat near the truth. A θ 10% too low loses 1.6% of profit, and 10% too high loses 0.8%. At 20% too low the loss is 5.7%, and at 25% too high it is 5.5%. Far from the truth the curve falls steeply: half the true value loses 33.7%, and twice the true value loses 49.6%. The flat top is why the experiment of Sec. VII is cheap, and it sets the target here. An estimate within about $\pm 10\%$ costs under 2% of profit; one within $\pm 25\%$ costs about 5%.

B. How fast one listing learns

The precision of the estimate comes from bookings, and from how much the price moved for reasons the model can separate from demand. The program computes the expected Fisher information, the curvature of the log-likelihood that sets the standard error. It does so for one year of the cabin priced by the dynamic program with a random nudge, and with the planned price as a control, so that only the nudge informs θ . That is what the two-stage estimator of Sec. VII uses.

With a $\pm 10\%$ nudge, one year gives a standard error of \$48.0 on θ . The 95% interval is then $\pm 85\%$ after one year, $\pm 60\%$ after two and $\pm 43\%$ after four (Fig. 7). Reaching $\pm 10\%$ would take about 73 years. The information grows with the square of the nudge's width, and so does its cost. A $\pm 20\%$ nudge gives $\pm 46\%$ after one year and $\pm 23\%$ after four, and costs 1.8% of profit a year against 0.4% for $\pm 10\%$.

A model can learn faster from prices it set itself. Once the prices come from a model that reacts only to features it records, they are not confounded, and their own movements with season and lead time inform θ too. With that variation counted, one year with a $\pm 10\%$ nudge gives a standard error of \$25, or $\pm 45\%$. The gain rests on the model's form for season and lead time being right, so the nudge-only figure is the safer guide.

C. What follows

A single listing does not have enough bookings to measure its own price sensitivity well. With a $\pm 20\%$ nudge it reaches the $\pm 25\%$ band, where pricing loses about 5%, after about four years. The $\pm 10\%$ band takes 21 years. Two consequences follow. A host should not trust an elasticity estimated from

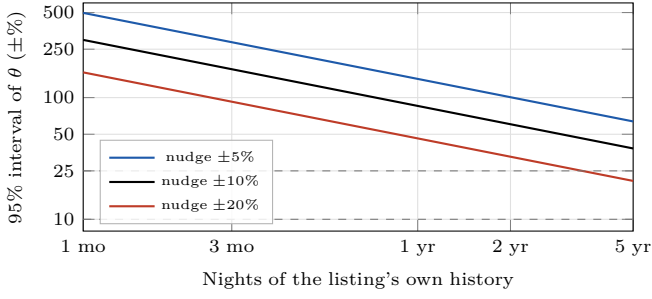


Fig. 7. How precisely one listing pins down its own price sensitivity from its nudges alone, from the expected Fisher information. The interval shrinks with the square root of the history. The dashed lines mark $\pm 25\%$ (about 5% of profit) and $\pm 10\%$ (under 2%).

one or two seasons of a single listing without a check against other evidence. And a host who starts with reasonable outside information about θ gains far more from it than from another year of data. The next section shows how to combine the two.

X. STARTING WITH LITTLE DATA

A new listing has no history, and Sec. IX showed that even an old one learns slowly. The remedy is to start from what similar listings show and let the listing's own data correct it. This section does that with a Bayesian prior and then uses the uncertainty that remains to choose prices.

A. A prior from comparable listings

A *prior* is a probability distribution over the parameters before the listing's own data arrive. Comparable listings in the same town, from Inside Airbnb or a market-data service, suggest values for the seasonal shape, the weekend and event effects and the lead-time scale. A price sensitivity can come from published market studies or from an experiment run across several listings. The cabin's prior puts θ at \$147, which is 33% too high, with a standard deviation of 30%. It is a deliberately imperfect prior, as an outside estimate usually is.

The prior and the data combine by Bayes' rule. The program uses the *Laplace approximation*. It finds the most probable parameters, where the log-likelihood plus the log-prior is largest. It then treats the uncertainty around them as Gaussian, with a covariance taken from the curvature at that point [18]. The estimate then moves from the prior towards the data as bookings accumulate.

B. Pooling

The same idea scales up. A *hierarchical* model estimates many listings together: each listing has its own θ , and the θ s are drawn from a common distribution whose centre and spread are also estimated [18]. A listing with little data gets an estimate close to the group's centre; a listing with much data keeps its own. This is called *partial pooling*. A host with several listings can pool them, and a pricing service that serves thousands of listings pools across all of them. The same structure can pool across a single listing's nights. The cabin's model already does: one θ serves every night, and the seasonal curve shares strength across neighbouring dates.

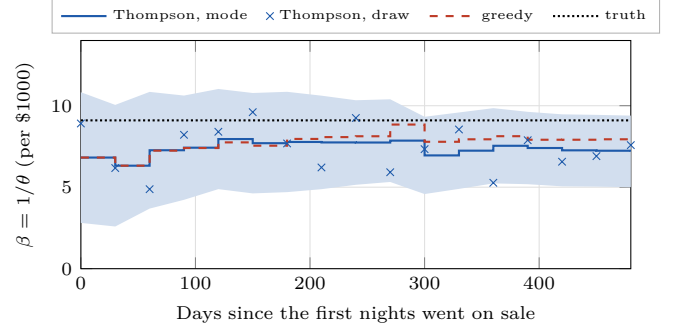


Fig. 8. One new listing's estimate of the price coefficient, refitted monthly. The shaded band is the 95% posterior interval of the Thompson-sampling pricer, and the crosses are the draws it priced with. The estimate starts at the prior and moves towards the truth.

C. Thompson sampling

Once the uncertainty is known, it can choose the prices. In *Thompson sampling*, the pricer draws one set of parameters at random from the current posterior and prices as if that draw were the truth [11], [12]. Where the posterior is wide, the draws differ, prices vary, and the fit learns from the variation. As the posterior narrows, the draws agree and exploration stops by itself. The method has been adapted to revenue management with limited inventory [13]. The simpler alternative, the *greedy* pricer, always uses the most probable parameters. It can stop learning if its prices settle, because the fit can then no longer tell parameter values apart [14], [15].

D. The cabin's first year

The program opens four new listings with the cabin's market and the prior above, and prices the nights of each one's first year. The first nights go on sale 120 days before the year starts, so each run covers 484 days and 16 monthly refits. At the start of each month it refits the model on everything so far, then either draws from the posterior (Thompson) or takes its mode (greedy), and reprices every open night with the dynamic program.

Both pricers do well. Over the first year, Thompson sampling earns 97% of the profit of the reference, and the greedy pricer 99%. A pricer that kept the prior's θ and the true values of everything else would lose 9.6%. After the year, Thompson's estimate of θ averages \$123 (from \$116 to \$139 across the four listings), and the greedy pricer's averages \$122. Both have moved most of the way from the prior's \$147 towards the true \$110. The posterior standard deviation shrinks by a factor of 1.8 over the year (Fig. 8).

In this market the greedy pricer did slightly better, and the reason is specific to it. The dynamic program's prices already move a great deal with season and lead time, and that movement is enough to learn from. Thompson's random draws added variation that was not needed and cost a little profit. A listing whose model prices nights more uniformly would learn less from greedy pricing. There, Thompson sampling or the nudge of Sec. VII keeps the learning going.

XI. CHECKING THE MODEL

A fitted model should be checked on data it has not seen. This section holds out a year of the cabin's nudged history, compares the model's predictions with what happened, and shows which check tests the price sensitivity and which does not.

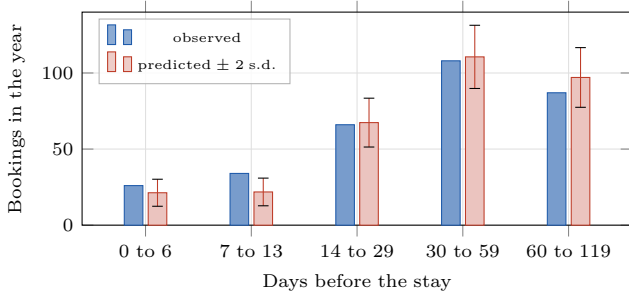


Fig. 9. A held-out year: bookings by lead time against the model's prediction, with two standard deviations around it. Each band's count is a sum of independent daily chances, given the nights on sale.

A. Bookings by lead time

A different year of nudged history had 321 bookings. The model fitted on one year predicts 318. Split by lead time, four of the five bands lie within two standard deviations of the prediction (Fig. 9). The band from 7 to 13 days out had 34 bookings against 22 predicted, 2.7 standard deviations off. With five bands, one such miss is no strong evidence against the model, but a host who saw it again the next year would add a feature for the last two weeks.

B. The check that tests the price

Matching the total bookings says little about θ . A model fitted to a host's own prices predicts well at those prices, whatever its θ : the biased fit of Sec. VI matches its own history too. What tests θ is a prediction at prices the host did not choose. The nudge provides them.

The check splits the held-out rows by nudge: the cheapest third ($u < -5\%$) against the dearest third ($u > +5\%$). Each third has about 10,119 rows. The dearer third booked 86 times and the cheaper 126 times, a ratio of 0.68 with a standard error of 0.10. The two-stage fit predicts 0.65, and the biased fit predicts 0.83: it expects guests to mind the roughly 20% price difference between the two thirds much less than they did. The true model gives 0.71.

One year of nudges separates the two fits by only 1.6 standard errors. The check points the right way, but it is weak for the same reason as Sec. IX: one listing makes few bookings.

C. Testing a pricing policy live

The most direct test of a pricer is to run it and count the money. The host assigns each night at random to the current policy or the new one, and compares the average profit per night. The difficulty is noise. One night's realized profit under the reference policy has a standard deviation of \$104: most nights either sell for a couple of hundred dollars or earn nothing. Detecting a 4% difference between two policies, with the usual 5% significance and 80% power, needs 4,012 nights in each arm, about 11 years of one listing. A single host cannot test small improvements this way. The nudge check and the model-based comparison of Sec. VIII are what remain.

XII. STAYS, GAPS AND FEES

The model so far sells nights one at a time. Real bookings cover several nights, cost a fixed amount to clean, and leave gaps between them. This section adds these with the same tools: a per-night cost and a smaller pool of shoppers.

A. Cleaning and the length of stay

The cabin's cost of \$30 a sold night stands for a cleaning fee of \$90 spread over a typical stay of three nights. The cost of a night actually depends on the length of the stay L : it is \$90 for one night, \$45 a night for two, \$30 for three and \$13 for seven. Equation (12) adds the cost to the price one for one, so the best price per night falls with the length of stay by the same amount. A week-long stay costs \$17 a night less than a three-night stay, and a length-of-stay discount of that size passes the saving on without giving up profit. A larger discount needs another reason, such as a long stay filling quiet nights that would otherwise go empty.

A separate cleaning fee, charged per stay, does the same job from the other side. It makes the guest pay the per-stay cost directly, so the nightly rate can follow (12) with only the per-night costs. Guests compare the total, though, so a fee changes the price they react to. The model should be fitted on the total a guest pays, divided by the nights.

B. Minimum stays and gap nights

A *gap night* is a single open night between two bookings. Only a guest who wants exactly one night can book it. If a fifth of shoppers want one night, the gap night's demand is the usual demand times 20/100. That guest also bears the whole \$90 cleaning cost. Both changes enter the dynamic program directly.

Suppose a gap opens 14 days before the January night. The program prices the gap at \$208, above the usual \$196 for that day, because the cleaning cost is higher. It sells with probability 7%, and its expected profit is \$8, against \$57 for an ordinary night at the same lead time. For the July night the gap's price is \$231 against \$287, it sells with probability 25%, and it is worth \$32 against \$149.

Two practical conclusions follow. A gap night is worth little in any season, so a minimum stay that creates gaps costs less than it seems. And the dynamic program has already set the gap's best price, so a further discount loses money. The gap's value rises only with more shoppers: allowing a one-night stay into the gap alone, or offering the night to the guests on either side as an extension.

C. The floor price

A host's floor price (Sec. IV) usually reflects one of the costs above. Set it to the cost of the shortest stay the calendar allows, so that no booking loses money. A host may set a higher floor to protect the listing's image. The dynamic program shows what that costs: run it with and without the floor and compare W .

XIII. A WEEKLY ROUTINE FOR ONE HOST

The pieces of the previous sections fit into a routine that one host can run with a laptop and a short script. It has a daily part, which is mechanical, and a weekly part, which needs a few minutes of judgement. Figure 10 shows the cycle.

- Recipe 2 (Daily).** 1) Save a snapshot of the calendar: for each night up to 120 days ahead, whether it is open, booked or blocked, and its price.
- 2) Add one row per open night to the at-risk table, with the price shown, the planned price and the lead time. Mark the rows of nights that were booked since yesterday.
- 3) Set each open night's price to its planned price for today times e^u , with u uniform in ± 15 to 20% (Recipe 1).

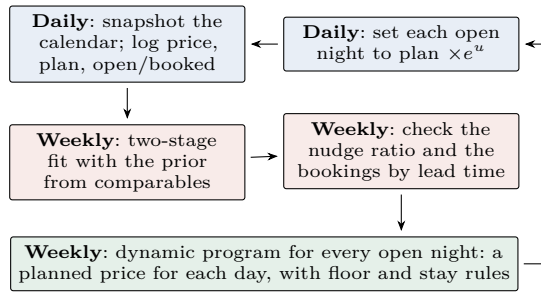


Fig. 10. The weekly routine. Blue steps run every day without judgement; red steps need a look; the green step produces the next week's planned prices.

- Recipe 3** (Weekly). 1) Refit the hazard model with the two-stage estimator, using the comparables' values as the prior. The functions `fit` and `dp_path` in `examples/strprice.py` are a starting point.
- 2) Look at three numbers: the new θ and its standard error; the predicted against the observed bookings by lead-time band; and the nudge ratio of Sec. XI. A θ that jumps by more than its standard error from one week to the next, or a band that misses twice running, needs investigating before the prices change.
- 3) Run the dynamic program for every open night, with the floor and the stay rules of Sec. XII, and store the planned price for each night and each lead time for the coming week.
- 4) Add or update the event flags for the coming months: festivals, holidays, school breaks. The model knows only the events it is told about.

The routine needs the platform, or a channel manager in front of it, to accept a daily price per night. Most do, through an upload or an interface for programs. The daily steps take no judgement and should be automated first. The weekly steps are where the host's knowledge of the town enters the model.

Tip. Start logging before anything else. The at-risk table cannot be rebuilt afterwards, because the prices of unbooked days are lost. Even a host who prices by hand today gains a year of usable history by keeping the daily snapshot.

XIV. WHAT THE MODEL LEAVES OUT

The cabin's market is simple on purpose, so that every estimate can be checked. A real market has effects the simulation does not, and each of them changes some number in this tutorial.

Other hosts react. The model treats the rest of the market as fixed. When neighbours use pricing tools that react to the same signals, their prices move with demand, and the cabin's θ measures the response to its own price with theirs moving too. The experiment still measures this correctly, because the nudge is the cabin's alone.

Price affects visibility. Platforms rank listings in search, and price is likely one input. A nudge may change how many shoppers see the listing as well as how many book. The two-stage estimate then measures both effects together, which is the quantity a pricer needs. A nudge that changes daily may interact with ranking in ways a slower change would not.

Willingness to pay has a shape. The exponential form (4) has a single parameter. Real guests may have a willingness to pay with a different shape, and it may differ between early and late bookers. Both can be tested with the nudge: fit θ separately

for early and late lead times, or compare small and large nudges. More flexible forms cost more data.

Larger surprises. The cabin's nights vary by about 35% beyond what season, weekday and events explain, and knowing each night's surprise in advance was worth only 0.8%. Where demand depends on weather, conferences or a single large venue, the shocks are larger. Market data on how fast the town is booking for each date, used as a feature, then become worth more.

Everything else. The model ignores cancellations, reviews (a full house earns more of them, which help bookings later), taxes, and platform fees that grow with the price. A fee that takes a fraction ϕ of the price leaves the host $(1 - \phi)x$, which in (12) acts like a higher cost $c/(1 - \phi)$.

Commercial tools face the same problem at a larger scale. Airbnb has published the structure of its own price suggestions: a model of the booking probability of each listing and night, followed by a second model trained to suggest prices [16]. The approach in this tutorial has the same two parts, at the scale of one listing.

XV. CONCLUSION

The 2023 note found the price that maximizes revenue per night: the price at which occupancy falls by one percent for each percent the price rises. It stopped at estimating the demand curve. This tutorial completed the model in four steps, each checked on a simulated market whose truth is known.

First, demand is modelled as the chance that an unsold night books on a given day, from the number of shoppers that day and their willingness to pay. The 2023 curve is this model with one price held for the whole window. With exponential willingness to pay, the 2023 optimum on a quiet night is the price sensitivity θ itself.

Second, a dynamic program turns the model into a price for each night and each day. The price is the cost of the night, plus the value of keeping it for a later guest, plus a markup of about θ . It falls as the stay approaches.

Third, the estimate must come from price variation that the host did not set in response to demand. A host's own history gives the wrong answer: the cabin's monthly scatter slopes upward, and a regression with seasonal controls estimates $\theta = \$226$ against a true \$110. A random daily nudge of $\pm 15\%$ costs about 1.0% of profit and, with the two-stage estimator, recovers $\$97 \pm \18 . Priced with that estimate, the cabin earns 98.7% of the best possible profit, 17% more than its host's rule.

Fourth, one listing learns slowly. With a $\pm 20\%$ nudge, its own data take about 21 years to pin θ within 10%. Profit is flat near the optimum, though: 10% off costs under 2%, and a prior from comparable listings that starts 33% off reaches 99% of the best profit in the first year.

The answer to the 2023 note's open question is therefore practical. Do not estimate the demand curve from a scatter of past prices and occupancy. Log the calendar every day, nudge the prices at random, fit the booking hazard with the two-stage estimator, start from what comparable listings show, and let the dynamic program set the prices.

REFERENCES

- [1] "Optimal pricing of short term rentals," *Our adventures in real estate* (blog), 25 June 2023, with a Google Colab notebook. [Online]. Available: <https://re.filmr.us/note/2023/06/25/optimal-pricing-of-short-term-rentals/>

- [2] “How to get the price right,” *Teconomics* (blog), Medium. [Online]. Available: <https://medium.com/teconomics-blog/how-to-get-the-price-right-9fda84a33fe5>
- [3] K. T. Talluri and G. J. van Ryzin, *The Theory and Practice of Revenue Management*. Boston, MA: Kluwer, 2004.
- [4] G. Gallego and G. van Ryzin, “Optimal dynamic pricing of inventories with stochastic demand over finite horizons,” *Management Science*, vol. 40, no. 8, pp. 999–1020, 1994.
- [5] R. L. Phillips, *Pricing and Revenue Optimization*. Stanford, CA: Stanford University Press, 2005.
- [6] K. Littlewood, “Forecasting and control of passenger bookings,” in *Proc. 12th AGIFORS Symposium*, 1972; reprinted in *J. Revenue and Pricing Management*, vol. 4, no. 2, pp. 111–123, 2005.
- [7] E. J. Working, “What do statistical ‘demand curves’ show?” *Quarterly Journal of Economics*, vol. 41, no. 2, pp. 212–235, 1927.
- [8] J. D. Angrist and J.-S. Pischke, *Mostly Harmless Econometrics*. Princeton, NJ: Princeton University Press, 2009.
- [9] R. L. Prentice and L. A. Gloeckler, “Regression analysis of grouped survival data with application to breast cancer data,” *Biometrics*, vol. 34, no. 1, pp. 57–67, 1978.
- [10] P. D. Allison, “Discrete-time methods for the analysis of event histories,” *Sociological Methodology*, vol. 13, pp. 61–98, 1982.
- [11] W. R. Thompson, “On the likelihood that one unknown probability exceeds another in view of the evidence of two samples,” *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.
- [12] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband and Z. Wen, “A tutorial on Thompson sampling,” *Foundations and Trends in Machine Learning*, vol. 11, no. 1, pp. 1–96, 2018.
- [13] K. J. Ferreira, D. Simchi-Levi and H. Wang, “Online network revenue management using Thompson sampling,” *Operations Research*, vol. 66, no. 6, pp. 1586–1602, 2018.
- [14] A. V. den Boer, “Dynamic pricing and learning: Historical origins, current research, and new directions,” *Surveys in Operations Research and Management Science*, vol. 20, no. 1, pp. 1–18, 2015.
- [15] A. V. den Boer and B. Zwart, “Simultaneously learning and optimizing using controlled variance pricing,” *Management Science*, vol. 60, no. 3, pp. 770–783, 2014.
- [16] P. Ye, J. Qian, J. Chen, C.-H. Wu, Y. Zhou, S. De Mars, F. Yang and L. Zhang, “Customized regression model for Airbnb dynamic pricing,” in *Proc. 24th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2018, pp. 932–940.
- [17] Inside Airbnb. [Online]. Available: <http://insideairbnb.com/>
- [18] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari and D. B. Rubin, *Bayesian Data Analysis*, 3rd ed. Boca Raton, FL: CRC Press, 2013.