Hotel Overbooking Strategy

Joseph Despres
Rishabh Sareen

October 5, 2021

Abstract Overbooking is used in many businesses such as airlines, cruise ships and even manufacturing. This project develops a profitable overbooking strategy using publicly available hotel cancellation data. The objective is to optimize the number of reservations given the uncertainty of a guest's arrival subject to a penalty. Overbooking is optimized by finding when expected profit of one additional reservation no longer justifies the expected cost of the penalty. This project shows, a hotel can increase arrivals and reduce empty rooms by over half if they are willing to pay modest overbooking penalties.

Work done in partial fulfillment of the requirements of Michigan State University MTH 843; advised by Dr. Peiru Wu, Michigan State University.

Table of Contents

Nomenclature

p	Probability of a guest arriving to a booking
A	Quantity of guests arriving to a reservation
b	Number of bookings made by the hotel
F	Binomial CDF
E	Expectation operator
π	Profit per guest arrived
L	Loss function of penalty paid when arrivals exceed capacity
k	Hotel capacity
$\hat{p}$	Estimated probability
β_i	Regression coefficient
x_i	Covariate
b^*	Maximum bookings given capacity and estimated probability of arrival
C_f	Estimated fraction of hotel capacity

Introduction

Overselling or overbooking is the practice of committing to sell more of a product or service than the ability to fulfill. Airlines, oversell tickets and offer compensation in the event excess passengers arrive (CFR 250). Manufactures often use similar algorithms to predict deferred or cancelled orders (Leahy 1). This permits them to accept more orders and collect additional revenue. However, this is a delicate task due to high costs incurred in the event of customer arrivals exceeding capacities.

The goal of this project is to construct a profitable overbooking strategy. Publicly available hotel booking data allows for this strategy to be validated with a simulation. This strategy is constructed by making some assumptions about overbooking penalties, setting an arbitrary hotel capacity, and estimating the uncertainty of each arrival. Then this strategy is tested by resampling bookings until expected capacity is reached.

If the probability of a guest arriving to a hotel reservation is not one, then a hotel is certain to have empty rooms. This is where the opportunity presents itself. Implemented properly, this offers a hotel the opportunity to collect additional revenue without incurring the costs of increasing capacity. A profitable hotel overbooking strategy can be modified, repurposed, and increase revenues in industries where clients tend to cancel preordered products or services.

This report is outlined in several sections. The Data Section contains a description of the hotel cancellation data used. The Analysis Section explains the underlying mathematics. The Methods Section contains the process of developing the overbooking strategy and the validation simulation. The Results section contains a discussion the results and a discussion of the applications and limitations. The Conclusion contains a summary of the results.

Data

This overbooking strategy is validated using a hotel booking demand data set (Antonio 1) in the public domain. These data are all reservations and cancellations from a hotel in Lisbon from 1st of July 2015 to 31st of August 2017. There are 119,930 observations with details of the client, reservations, and cancellations. The variables of interest are how many adults, children, and babies are included in a reservation, days until reservation, length of stay, and whether a deposit was made or if the booking is in a group or an individual reservation. These variables were selected because in addition to being statistically significant predictors of cancellations, it is the information a booking agent would have access to at the time of a booking.

Basic descriptive statistics for the numeric variables can be found in Table 1. Find that, on average, 63% of bookings arrive. Children are rarely staying at this hotel and babies are rarely included in reservations. The average lead time is 104 days however the median is only 69 days

indicating that the number of days customers book in advanced is right skewed. The high standard deviation suggests that there is significant variation in the lead time. Average stay length is approximately three and a half days, with a slight right skew and high standard deviation.

Table 1 Descriptive statistics on hotel arrivals and cancellations.

	Is canceled	Not canceled	Adults	Children	Babies	Lead time	Stay length
Minimum	0	0	0	0	0	0	0
Mean	0.37	0.63	1.86	0.1	0.01	104.01	3.43
Median	0	1	2	0	0	69	3
Standard Deviation	0.48	0.48	0.58	0.4	0.1	106.86	2.56

The categorical variables used to predict cancellations are the type of consumer and if a deposit was collected. The variable counts can be found in Table 2. Transient is a single guest unaffiliated with an organized group. This is the most common occurrence. Transient party is where an organized group has multiple bookings. Contract is when a booking has a formal business relationship with the hotel. Deposits are asked infrequently, and close inspection shows that deposits are taken in the event a customer is requesting flexible booking dates.

Table 2 Categorical variable counts.

Customer type	Deposit	N
Transient	No Deposit	76,684
Transient-Party	No Deposit	23,858
Transient	Deposit	12,929
Contract	No Deposit	3,530
Transient-Party	Deposit	1,266
Group	No Deposit	569
Contract	Deposit	546

This dataset is used to validate this overbooking strategy. This project proceeds on the assumption the booking data are broadly representative, and there is no selection mechanism biasing the underlying data. Therefore, this project will proceed on the assumption that the hotel is not employing any overbooking strategy or selecting customers for the likelihood of arrival. An additional assumption is that these data are representative of the typical arrival rates for this hotel.

Analysis

A guest making a hotel reservation is not certain to arrive. This uncertainty is defined as the probability of arrival p. A hotel will make many reservations and wonder how many arrivals A to expect given the number of bookings b. The number of arrivals A or fewer guests arriving follows a binomial distribution with a cumulative density function

$$\mathrm{F}(A|b,p) = \sum_{i=1}^{b} \binom{b}{i} p^i (1-p)^{b-i}, \quad A \leq b, \quad 0 \leq p \leq 1.$$

This function gives the probability of A or fewer guests arriving given the number of bookings, and probability of arrival. With this distribution, the total profit a hotel can expect while booking no more than capacity is

$$E[\text{Total Profit}] = \pi b F(A|b,p), \; A \leq b, \; 0 \leq p \leq 1, \; \pi > 0.$$

Where π denotes profit per arrival. Multiply profit, number of bookings, and sum probability of each arrival to get expected profit.

Booking over capacity allows fewer empty rooms, at the cost of risking arrivals exceeding capacity. The penalties associated with overbooking should be carefully considered. This strategy assumes the cost of overbooking is providing each additional customer accommodations at a cost of four times profit $L = 4\pi$. Now relax the assumption that bookings b is less than or equal to capacity k and obtain

$$E_b[\text{Total Profit}] = \pi b F(k|b,p) - L(b-k)\big(1 - F(k|b,p)\big), \; 0 \leq p \leq 1, \; \pi > 0.$$

This model estimates profit for a given number of bookings, accounting for the possibility of arrivals exceeding capacity. To determine the maximum number of bookings by finding the amount of booing's where $E_b[\text{Total Profit}] > E_{b+1}[\text{Total Profit}]$. Determine the number of bookings that maximize profit by solving this equation for b. Use an algorithm iterating over integer values of b starting at capacity and stopping when

$$E_b[\text{Total Profit}] > E_{b+1}[\text{Total Profit}].$$

See the source code in Table B.1 in Appendix B for the implementation in R (R Core Team 2020).

The hotel booking data (Antonio 1) shows, on average, 64% of guests arrive. Assume a 100-room hotel, the cost of overbooking is four times profit, and every guest has the same probability of arrival, maximize profit by booking 129 guests. The binomial distribution assumes that each p is the same for every trial. Assuming all customers have the same p, exposes the hotel to the inevitable event of booking too many customers highly likely to arrive and booking too few customers unlikely to arrive. The profitability of this strategy will depend on the ability to

accurately estimate p for each reservation. Therefore, this strategy requires an estimate for p denoted by $\hat{p}$.

The regression results can be found in Table A.1 in Appendix A. This supports the claim that each guest has a different probability of arrival. Fit a linear regression model to the log odds of arrival

$$\log\left[\frac{\hat{p}}{1-\hat{p}}\right] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ \ldots \ + \beta_n x_n.$$

Exponentiate both sides and solve for $\hat{p}$

$$\hat{p} = \frac{exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ \ldots \ + \beta_n x_n)}{1 + exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ \ldots \ + \beta_n x_n)}.$$

In addition to estimating $\hat{p}$, the values of β_i estimate the effect of variable x_i.

Methods

It has been shown that different guests have different probabilities of arrival. This overbooking strategy begins by estimating the probability a given guest will arrive. The closer $\hat{p}$ is to the true value of p, the more profitable this strategy will be. To validate this method, split the booking data into a training dataset and a testing data set. Rather than randomly selecting data, separate the data by the first 80% chronologically. Then fit and validate a predictive model. A logistic regression model estimates the probability arrival to be

$$\begin{aligned}\log\left[\frac{\hat{p}}{1-\hat{p}}\right] = \beta_0 &+ \beta_1(\text{customer type}) + \beta_4(\text{adults}) + \beta_5(\text{children}) + \beta_6(\text{babies}) \\ &+ \beta_7(\text{lead time}) + \beta_8(\text{stay length}) + \beta_9(\text{deposit}).\end{aligned}$$

The estimated coefficients β_i can be found in Appendix A on Table A.1. These regression coefficients are all statistically significant. This supports the claim that the probability of arrival p is not the same for each guest. Groups on average tend to have a higher arrival rate than individual short-term guests. The more adults in a single booking, the lower the expected arrival rate. Children seem to increase the odds of arrival until accounting for factors such as group, lead time, and stay length. Guests not making a deposit have a far higher arrival rate than those making non-refundable deposits. Closer inspection shows that non-refundable deposits are only asked of guests requesting flexible booking or have a history of cancelling reservations.

Use $\hat{p}$ to obtain the estimated fraction of capacity of the booking. Let b^* be the optimal number of bookings b where $E_{b+1}[\text{Total Profit}] - E_b[\text{Total Profit}] = 0$ given $\hat{p}$. Figure 1 shows that b^* goes to infinity as the probability of arrival approaches 0. In a commercial setting, this should be restricted to avoid the possibility of a catastrophic loss. This strategy restricts b^* to no more than 10 times capacity.

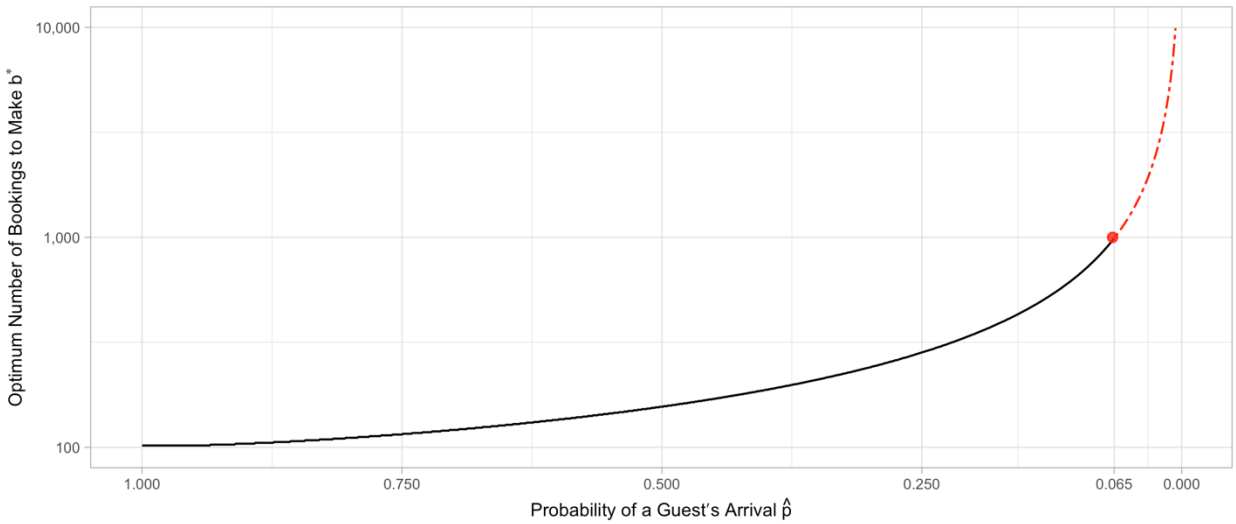

Figure 1 Relationship between estimated probability of arrival and optimum bookings.

After obtaining b^*, find the expected fraction of capacity the guest consumes C_f by taking the inverse of b^*

$$C_f = \frac{1}{b^*}.$$

Assign each booking an expected fraction of capacity. Then accept bookings until the total fraction of capacity exceeds 0.99.

$$\sum_i C_{fi} > 0.99$$

The validation simulation randomly samples bookings, with replacement, on each date until estimated capacity is reached. For the details of the sampling function see the source code in Appendix B Table B.2.

This simulation assumes a hotel has a capacity of 100 rooms. Then to study the results of $\hat{p}$ being the same sample as if $\hat{p} = 0.64$ and book 129 guests every day. This involves sampling with replacement until there are 129 guests in a day. Then sample on the b^* associated with each booking and sample for each day until $\sum_i C_{fi} > 0.99$. This results in an overbooking simulation of 162 days, where each day has bookings at capacity, overbooked assuming p is constant at 64%, and overbooking based on the estimate $\hat{p}$. Then is experiment is repeated 2,500 times.

Results and Discussion

The simulation results are found in Table 3. Booking at capacity expects 59 arrivals, the lowest average arrivals. Booking with a constant $\hat{p} = 0.64$ is not a viable option as it way too frequently overbooks and incurs an expected per day penalty of 4.25. The proposed strategy reduces expected number of empty rooms by 60% and increases number of arrivals by 30% from

59 to 84. This comes at modest expected per day penalty of 0.75. The opportunity to collect, an average, the revenue for 25 additional rooms is worthy of consideration.

Table 3 Descriptive statistics of simulation results.

	Expected Per day Penalty	Expected Empty Rooms	Expected Arrivals	Std. Dev. Arrivals
Booking at Capacity	0	41.05	58.95	12.87
Overbooking with $\hat{p} = 0.64$	4.25	10.77	93.48	18.02
Proposed Overbooking Strategy	0.75	16.31	84.43	12.55

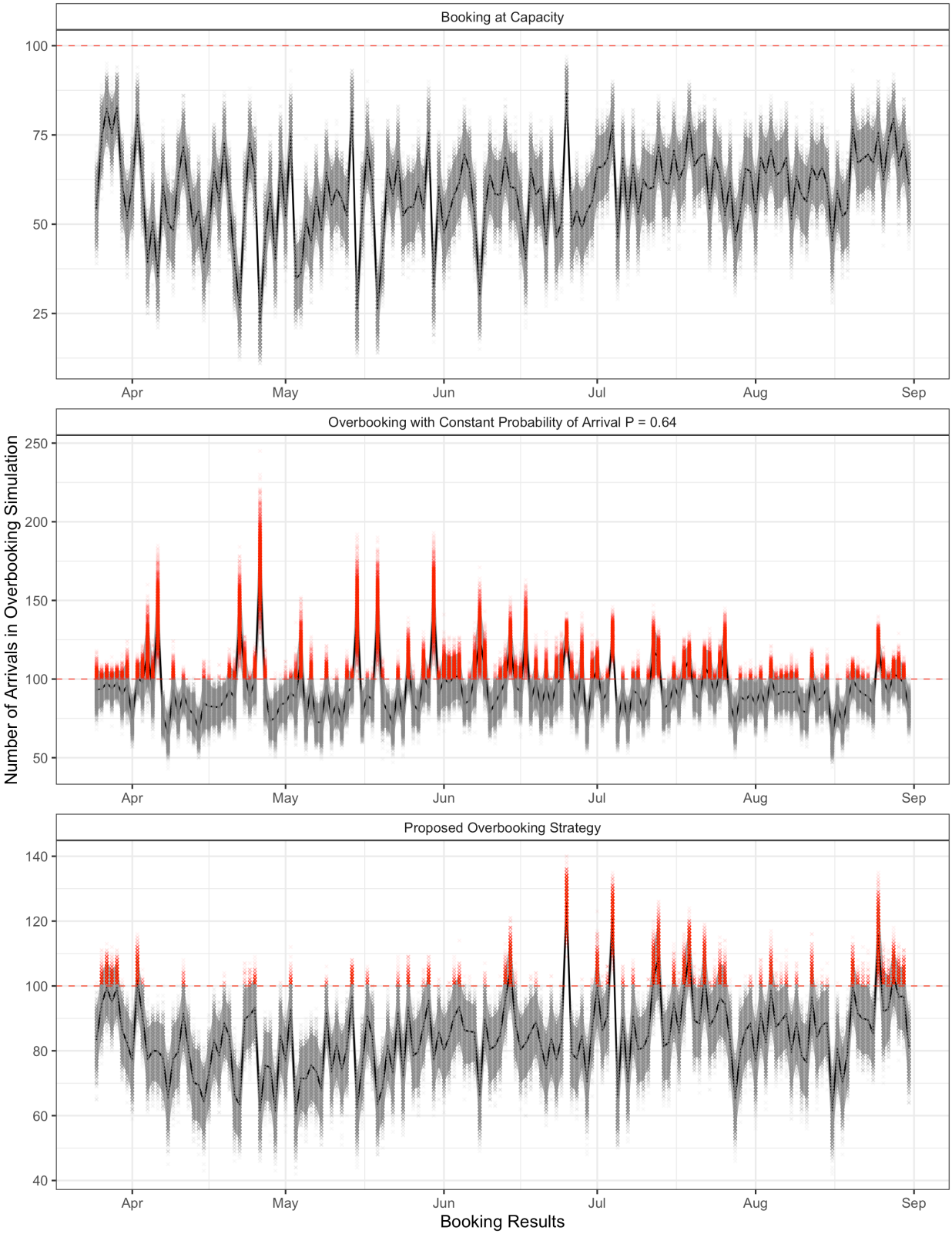

Figure 2 Overbooking strategy performance test.

Figure 2 contains the results of the simulation. First, are resampling bookings at capacity. The black line indicates the average arrivals, solid grey is the 95% confidence interval, the "x" indicates outliers occurring in less than 5% of the trials, and red represents overbooks or anytime capacity is exceeded. Booking at capacity never exceeds 100, overbooking with constant p is often over capacity, and the proposed strategy maintains arrivals consistently close to capacity and only exceeding capacity on rare occasions.

Another problem with assuming the same probability of arrival is the increased standard deviation of arrivals. Any increase in the standard deviation of arrivals further strains the operation of the hotel. The proposed overbooking strategy resulted in a slight decrease in the standard deviation of arrivals. Further explorations could develop methods to lower the standard deviation of arrivals. This would have several benefits including more predictable workloads for staff and general ease of operation.

The most obvious area to improve this strategy would be to attempt other classification engines to get more accurate estimates for p. However, more powerful engines may reduce interpretability and require more data than a booking agent is able to collect at the time of a reservation. Additionally, more powerful classification engines may be more sensitive to changes in the business climate.

Conclusion

This project has shown on overbooking strategy that slightly lowers the standard deviation of arrivals and reduces the expected number of empty rooms. This comes with the cost of a modest overbooking penalty. This method is validated with a simulation and the results are:

- A hotel can expect to have 30% more guests arriving.
- A hotel will experience 60% fewer empty rooms.
- This method comes at an expected cost of 0.75 rooms over capacity per day.

From the above performance results, this strategy can be employed and increase profits.

References

[1] “14 CFR 250.2B -- Carriers to Request Volunteers for Denied Boarding.” *14 CFR 250.2b -- Carriers to Request Volunteers for Denied Boarding.*, <https://www.ecfr.gov/current/title-14/chapter-II/subchapter-A/part-250/section-250.2b>.

[2] Antonio, Nuno, et al. “Hotel Booking Demand Datasets.” *Data in Brief*, Elsevier, 29 Nov. 2018, <https://www.sciencedirect.com/science/article/pii/S2352340918315191#f0010>.

[3] Leahy, John. “Too Many Orders? Yes, Says Consultant. No, Says Ex-Super-Salesman.” *Leeham News and Analysis*, 28 Jan. 2019, <https://leehamnews.com/2019/01/28/too-many-orders-yes-says-consultant-no-says-ex-super-salesman/>.

[4] R Core Team: A language and environment for statistical computing. 2020 R Foundation for Statistical Computing, Vienna, Austria, <https://www.R-project.org/>.

Appendix A

The regression results in Table A.1 show statistically significant relationships between the log odds of a guest arriving and the covariates.

Table A.1 Regression results showing that the probability of arrival is unique to each guest.

	Dependent variable:		
	Log Odds of Arrival		
	(1)	(2)	(3)
Adults	-0.183***	-0.065***	-0.231***
	(0.020)	(0.018)	(0.017)
Children	-0.161***	0.001	0.145***
	(0.025)	(0.025)	(0.024)
Babies	0.730***	0.890***	1.236***
	(0.154)	(0.156)	(0.153)
Lead Time (Days)	-0.004***	-0.007***	
	(0.0001)	(0.0001)	
Stay Length (Days)	-0.076***	-0.030***	
	(0.005)	(0.005)	
Customer Type: Group	1.312***		
	(0.220)		
Customer Type: Transient	0.061		
	(0.054)		
Customer Type: Tranient-Party	0.582***		
	(0.057)		
No Deposit	6.295***		
	(0.187)		
Constant	-4.654***	1.313***	0.764***
	(0.197)	(0.036)	(0.032)
Observations	59,494	59,494	59,494
Log Likelihood	-28,857.340	-36,311.220	-40,138.290
Akaike Inf. Crit.	57,734.680	72,634.440	80,284.570
Note:			*p<0.1; **p<0.05; ***p<0.01

Appendix B

Table B.1 contains the source code for implementing the overbooking strategy. Input phat capacity, loss function, and the minimum probability this is to calculate. After initializing some values, this function takes a given p and estimates the expected units of profit given that probability of arrival while accounting for the possibility of too many guests arriving.

Table B.1 Source code for the function converting estimated probability of arrival to fraction of capacity.

```
capacity_frac ← function (p, rooms=100, loss=4, min_p=0.065) {

  # input phat associated with each booking
  E ← 0; E_m1 ← 0; Fx ← 0; P_overbook ← 0; bookings ← rooms;
  # Initialize values

  if(p > min_p) { # set a maximum overbooking threshold
    while(E_m1 ⩾ E) {
      E ←  bookings * Fx - loss * ((bookings - rooms) * P_overbook)
      Fx ←  pbinom(rooms, bookings, p, lower.tail = TRUE)
      bookings ←  bookings + 1
      P_overbook ←  1 - Fx
      E_m1 ←  bookings * Fx - loss * ((bookings - rooms) * P_overbook)
      }
  } else {
    bookings = rooms * 10
  }
  return(bookings)
}

test_pred ← test %>%
  mutate(p_hat = predict(logistic_fit,
                         newdata = test, type = "response"),
         cap = map_dbl(p_hat, capacity_frac),
         cap_fra = 1/cap)
```

Table B.2 Contains the resampling functions for the three scenarios that are tested. First, a unique estimate for *p*. The input is a single day of a hotel and samples from that until the total expected fraction of capacity exceeds 0.99. In other words, this simulates the process of booking and overbooking hotel reservations until the capacity is reached weighting the guests based on their probability of arrival.

Table B.2 Resampling functions that simulate the overbooking strategy.

```
sample_fn_overbook ← function(data_input, p_threash=0.99) {
  # Input data for a single booking date
  success ← FALSE
  i ← 1
  samp ← data_input[1,]
  while(!success) {
    samp[i,] ← slice_sample(data_input, n = 1) # randomly select booking until
    success ← sum(samp$cap_fra) > p_threash    # expected capacity is reached
    i ← i + 1
  }
  samp ← samp[1:i,]
  # output resampled bookings
  return(samp)
}

sample_capacity ← function(data_input, capacity = 100) {
  # Input data for a single booking date
  output ← slice_sample(data_input, n = capacity, replace = TRUE)
  # Sample to hotel capacity with replacement
  return(output)
}

sample_same_phat ← function(data_input, capacity = 100) {
  # Input data for a single booking date
  output ← slice_sample(data_input, n = 129, replace = TRUE)
  # Sample 129 rooms assuming each guest has the same
  # likelihood of arrival
  return(output)
}
```

Table B.3 Simulation function that repeats the resampling functions for 2500 reps.

```
simulater ← function (x) {
  message(paste0(x))
  test_pred %>%
    group_by(check_in_date) %>%
    nest() %>%
    mutate(
      resampled_overbookings = map(data, sample_fn_overbook),
      resampled_same_phat = map(data, sample_same_phat),
      resampled_capacity = map(data, sample_capacity),

      over_nbookings = map_dbl(resampled_overbookings, ~nrow(.x)),
      over_canceled = map_dbl(resampled_overbookings, ~sum(.x$is_canceled, na.rm = TRUE)),
      over_booking_arrivals = over_nbookings - over_canceled,

      over_same_phat = 129,
      over_canceled_same_p = map_dbl(resampled_same_phat, ~sum(.x$is_canceled, na.rm = TRUE)),
      over_booking_arrivals_phat = over_nbookings - over_canceled,

      booking_n = 100,
      booking_canceled = map_dbl(resampled_capacity, ~sum(.x$is_canceled, na.rm = TRUE)),
      booking_arrivals = booking_n - booking_canceled,

      iter = x
    )   %>%
    select_if(is.numeric)
}

map(1:2.5e3, simulater) %>%
  # run simulation 2500 times
  # note this requires approximately
  # 18 hours of computer time
  bind_rows() %>%
  select_if(is.numeric) %>%
  write.csv(here::here("data", "sim_results_iterations.csv"))
```

Table B.3 Contains the function that implements multiple iterations of this resampling process. This runs the resampling process for each of the testing days 2,500 times. If you wish to recreate these results find the full project source code here.