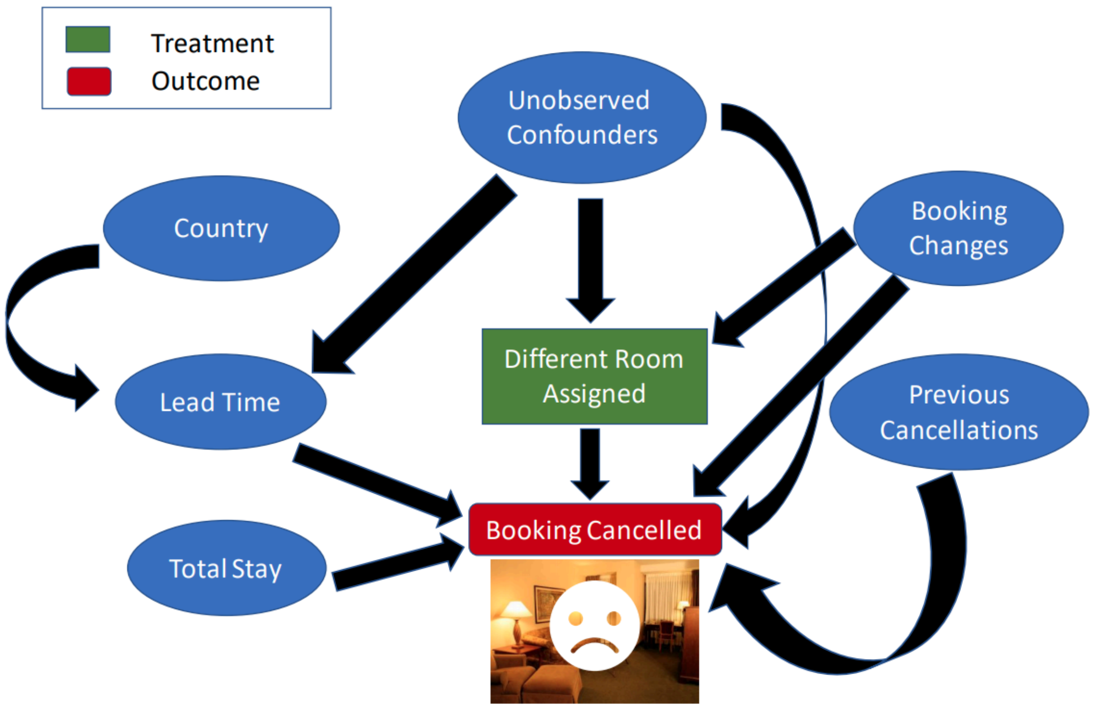
Exploring Causes of Hotel Booking Cancellations



We consider what factors cause a hotel booking to be cancelled. This analysis is based on a hotel bookings dataset from Antonio, Almeida and Nunes (2019). On GitHub, the dataset is available at rfordatascience/tidytuesday.

There can be different reasons for why a booking is cancelled. A customer may have requested something that was not available (e.g., car parking), a customer may have found later that the hotel did not meet their requirements, or a customer may have simply cancelled their entire trip. Some of these like car parking are actionable by the hotel whereas others like trip cancellation are outside the hotel's control. In any case, we would like to better understand which of these factors cause booking cancellations.

The gold standard of finding this out would be to use experiments such as *Randomized Controlled*

Skip to main content

customer is either assigned a car parking or not. However, such an experiment can be too costly and also unethical in some cases (for example, a hotel would start losing its reputation if people learn that its randomly assigning people to different level of service).

Can we somehow answer our query using only observational data or data that has been collected in the past?

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import dowhy
```

Data Description

For a quick glance of the features and their descriptions the reader is referred here.
rfordatascience/tidytuesday

[2]:
```
dataset = pd.read_csv('https://raw.githubusercontent.com/Sid-darthvader/DoWhy-T
dataset.head()
```

[2]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	ar
0	Resort Hotel	0	342	2015	July	27	
1	Resort Hotel	0	737	2015	July	27	
2	Resort Hotel	0	7	2015	July	27	
3	Resort Hotel	0	13	2015	July	27	
4	Resort Hotel	0	14	2015	July	27	

5 rows × 32 columns

[3]:
```
dataset.columns
```

[3]:
```
Index(['hotel', 'is_canceled', 'lead_time', 'arrival_date_year',
       'arrival_date_month', 'arrival_date_week_number',
       'arrival_date_day_of_month', 'stays_in_weekend_nights',
       'stays_in_week_nights', 'adults', 'children', 'babies', 'meal',
       'country', 'market_segment', 'distribution_channel',
       'is_repeated_guest', 'previous_cancellations',
```

Skip to main content

```
        'required_car_parking_spaces', 'total_of_special_requests',
        'reservation_status', 'reservation_status_date'],
       dtype='object')
```

Feature Engineering

Lets create some new and meaningful features so as to reduce the dimensionality of the dataset. - **Total Stay** = stays_in_weekend_nights + stays_in_week_nights - **Guests** = adults + children + babies - **Different_room_assigned** = 1 if reserved_room_type & assigned_room_type are different, 0 otherwise.

[4]:
```
# Total stay in nights
dataset['total_stay'] = dataset['stays_in_week_nights']+dataset['stays_in_weeke
# Total number of guests
dataset['guests'] = dataset['adults']+dataset['children'] +dataset['babies']
# Creating the different_room_assigned feature
dataset['different_room_assigned']=0
slice_indices =dataset['reserved_room_type']!=dataset['assigned_room_type']
dataset.loc[slice_indices,'different_room_assigned']=1
# Deleting older features
dataset = dataset.drop(['stays_in_week_nights','stays_in_weekend_nights','adult
                        ,'reserved_room_type','assigned_room_type'],axis=1)
dataset.columns
```

[4]:
```
Index(['hotel', 'is_canceled', 'lead_time', 'arrival_date_year',
       'arrival_date_month', 'arrival_date_week_number',
       'arrival_date_day_of_month', 'meal', 'country', 'market_segment',
       'distribution_channel', 'is_repeated_guest', 'previous_cancellations',
       'previous_bookings_not_canceled', 'booking_changes', 'deposit_type',
       'agent', 'company', 'days_in_waiting_list', 'customer_type', 'adr',
       'required_car_parking_spaces', 'total_of_special_requests',
       'reservation_status', 'reservation_status_date', 'total_stay', 'guests',
       'different_room_assigned'],
      dtype='object')
```

We also remove other columns that either contain NULL values or have too many unique values (e.g., agent ID). We also impute missing values of the `country` column with the most frequent country. We remove `distribution_channel` since it has a high overlap with `market_segment`.

[5]:
```
dataset.isnull().sum() # Country,Agent,Company contain 488,16340,112593 missing
dataset = dataset.drop(['agent','company'],axis=1)
# Replacing missing countries with most freqently occuring countries
dataset['country']= dataset['country'].fillna(dataset['country'].mode()[0])
```

[6]:
```
dataset = dataset.drop(['reservation_status','reservation_status_date','arrival
dataset = dataset.drop(['arrival_date_year'],axis=1)
dataset = dataset.drop(['distribution_channel'], axis=1)
```

Skip to main content

```
# Replacing 1 by True and 0 by False for the experiment and outcome variables
dataset['different_room_assigned']= dataset['different_room_assigned'].replace(
dataset['different_room_assigned']= dataset['different_room_assigned'].replace(
dataset['is_canceled']= dataset['is_canceled'].replace(1,True)
dataset['is_canceled']= dataset['is_canceled'].replace(0,False)
dataset.dropna(inplace=True)
print(dataset.columns)
dataset.iloc[:, 5:20].head(100)
```

```
Index(['hotel', 'is_canceled', 'lead_time', 'arrival_date_month',
       'arrival_date_week_number', 'meal', 'country', 'market_segment',
       'is_repeated_guest', 'previous_cancellations',
       'previous_bookings_not_canceled', 'booking_changes', 'deposit_type',
       'days_in_waiting_list', 'customer_type', 'adr',
       'required_car_parking_spaces', 'total_of_special_requests',
       'total_stay', 'guests', 'different_room_assigned'],
      dtype='object')
```

[7]:

	meal	country	market_segment	is_repeated_guest	previous_cancellations	previous_bookings_not
0	BB	PRT	Direct	0	0	
1	BB	PRT	Direct	0	0	
2	BB	GBR	Direct	0	0	
3	BB	GBR	Corporate	0	0	
4	BB	GBR	Online TA	0	0	
...	...	...	...	...	...	
95	BB	PRT	Online TA	0	0	
96	BB	PRT	Online TA	0	0	
97	HB	ESP	Offline TA/TO	0	0	
98	BB	PRT	Online TA	0	0	
99	BB	DEU	Direct	0	0	

100 rows × 15 columns

[8]:
```
dataset = dataset[dataset.deposit_type=="No Deposit"]
dataset.groupby(['deposit_type','is_canceled']).count()
```

[8]:

		hotel	lead_time	arrival_date_month	arrival_date_week_number	meal	c
deposit_type	**is_canceled**						
No Deposit	**False**	74947	74947	74947	74947	74947	
	True	29690	29690	29690	29690	29690	

[9]:
```
dataset_copy = dataset.copy(deep=True)
```

Skip to main content

Since the number of number of cancellations and the number of times a different room was assigned is heavily imbalanced, we first choose 1000 observations at random to see that in how many cases do the variables; *'is_cancelled'* & *'different_room_assigned'* attain the same values. This whole process is then repeated 10000 times and the expected count turns out to be near 50% (i.e. the probability of these two variables attaining the same value at random). So statistically speaking, we have no definite conclusion at this stage. Thus assigning rooms different to what a customer had reserved during his booking earlier, may or may not lead to him/her cancelling that booking.

[10]:
```
counts_sum=0
for i in range(1,10000):
        counts_i = 0
        rdf = dataset.sample(1000)
        counts_i = rdf[rdf["is_canceled"]== rdf["different_room_assigned"]].sha
        counts_sum+= counts_i
counts_sum/10000
```

[10]: 588.7789

We now consider the scenario when there were no booking changes and recalculate the expected count.

[11]:
```
# Expected Count when there are no booking changes
counts_sum=0
for i in range(1,10000):
        counts_i = 0
        rdf = dataset[dataset["booking_changes"]==0].sample(1000)
        counts_i = rdf[rdf["is_canceled"]== rdf["different_room_assigned"]].sha
        counts_sum+= counts_i
counts_sum/10000
```

[11]: 572.7711

In the 2nd case, we take the scenario when there were booking changes(>0) and recalculate the expected count.

[12]:
```
# Expected Count when there are booking changes = 66.4%
counts_sum=0
for i in range(1,10000):
        counts_i = 0
        rdf = dataset[dataset["booking_changes"]>0].sample(1000)
        counts_i = rdf[rdf["is_canceled"]== rdf["different_room_assigned"]].sha
        counts_sum+= counts_i
counts_sum/10000
```

[12]: 665.5664

Skip to main content

But is *Booking Changes* the only confounding variable? What if there were some unobserved confounders, regarding which we have no information(feature) present in our dataset. Would we still be able to make the same claims as before?

Using DoWhy to estimate the causal effect

Step-1. Create a Causal Graph

Represent your prior knowledge about the predictive modelling problem as a CI graph using assumptions. Don't worry, you need not specify the full graph at this stage. Even a partial graph would be enough and the rest can be figured out by *DoWhy* ;-)

Here are a list of assumptions that have then been translated into a Causal Diagram:-

- *Market Segment* has 2 levels, "TA" refers to the "Travel Agents" and "TO" means "Tour Operators" so it should affect the Lead Time (which is simply the number of days between booking and arrival).
- *Country* would also play a role in deciding whether a person books early or not (hence more *Lead Time*) and what type of *Meal* a person would prefer.
- *Lead Time* would definitely affected the number of *Days in Waitlist* (There are lesser chances of finding a reservation if you're booking late). Additionally, higher *Lead Times* can also lead to *Cancellations*.
- The number of *Days in Waitlist*, the *Total Stay* in nights and the number of *Guests* might affect whether the booking is cancelled or retained.
- *Previous Booking Retentions* would affect whether a customer is a or not. Additionally, both of these variables would affect whether the booking get *cancelled* or not (Ex- A customer who has retained his past 5 bookings in the past has a higher chance of retaining this one also. Similarly a person who has been cancelling this booking has a higher chance of repeating the same).
- *Booking Changes* would affect whether the customer is assigned a *different room* or not which might also lead to *cancellation*.
- Finally, the number of *Booking Changes* being the only variable affecting *Treatment* and *Outcome* is highly unlikely and its possible that there might be some *Unobsevered Confounders*, regarding which we have no information being captured in our data.

Skip to main content

[13]:
```
import pygraphviz
causal_graph = """digraph {
different_room_assigned[label="Different Room Assigned"];
is_canceled[label="Booking Cancelled"];
booking_changes[label="Booking Changes"];
previous_bookings_not_canceled[label="Previous Booking Retentions"];
days_in_waiting_list[label="Days in Waitlist"];
lead_time[label="Lead Time"];
market_segment[label="Market Segment"];
country[label="Country"];
U[label="Unobserved Confounders",observed="no"];
is_repeated_guest;
total_stay;
guests;
meal;
hotel;
U->{different_room_assigned,required_car_parking_spaces,guests,total_stay,total
market_segment -> lead_time;
lead_time->is_canceled; country -> lead_time;
different_room_assigned -> is_canceled;
country->meal;
lead_time -> days_in_waiting_list;
days_in_waiting_list ->{is_canceled,different_room_assigned};
previous_bookings_not_canceled -> is_canceled;
previous_bookings_not_canceled -> is_repeated_guest;
is_repeated_guest -> {different_room_assigned,is_canceled};
total_stay -> is_canceled;
guests -> is_canceled;
booking_changes -> different_room_assigned; booking_changes -> is_canceled;
hotel -> {different_room_assigned,is_canceled};
required_car_parking_spaces -> is_canceled;
total_of_special_requests -> {booking_changes,is_canceled};
country->{hotel, required_car_parking_spaces,total_of_special_requests};
market_segment->{hotel, required_car_parking_spaces,total_of_special_requests};
}"""
```

Here the *Treatment* is assigning the same type of room reserved by the customer during Booking. *Outcome* would be whether the booking was cancelled or not. *Common Causes* represent the variables that according to us have a causal affect on both *Outcome* and *Treatment*. As per our causal assumptions, the 2 variables satisfying this criteria are *Booking Changes* and the *Unobserved Confounders*. So if we are not specifying the graph explicitly (Not Recommended!), one can also provide these as parameters in the function mentioned below.

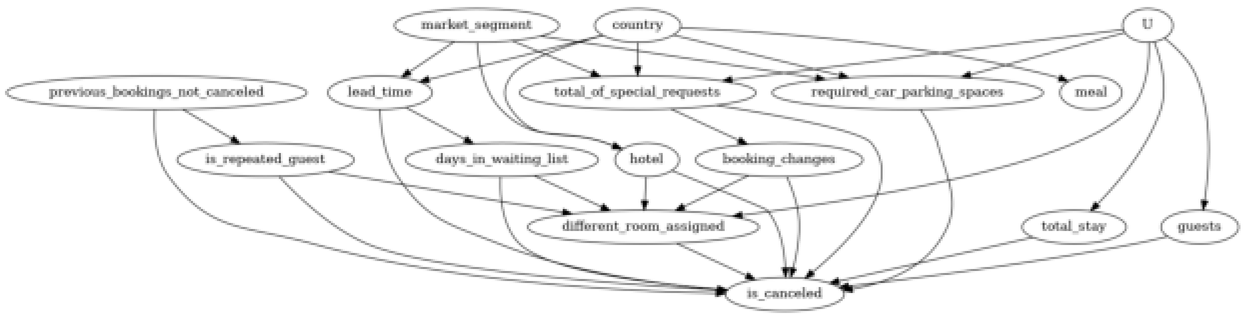
To aid in identification of causal effect, we remove the unobserved confounder node from the graph. (To check, you can use the original graph and run the following code. The `identify_effect` method will find that the effect cannot be identified.)

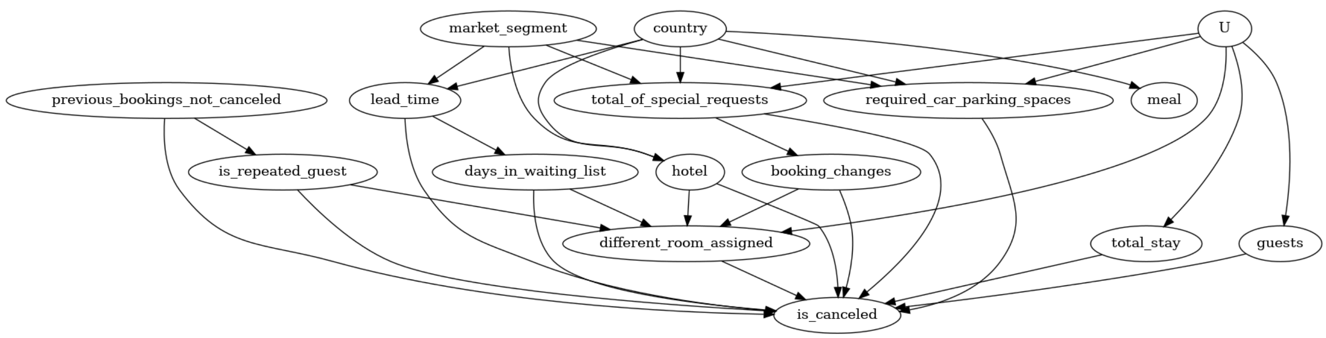
[14]:
```
model= dowhy.CausalModel(
        data = dataset
```

Skip to main content

```
        treatment="different_room_assigned",
        outcome='is_canceled')
model.view_model()
from IPython.display import Image, display
display(Image(filename="causal_model.png"))
```

```
/__w/dowhy/dowhy/dowhy/causal_model.py:583: UserWarning: 1 variables are assume
  warnings.warn(
```





Step-2. Identify the Causal Effect

We say that Treatment causes Outcome if changing Treatment leads to a change in Outcome keeping everything else constant. Thus in this step, by using properties of the causal graph, we identify the causal effect to be estimated

[15]:
```
#Identify the causal effect
identified_estimand = model.identify_effect(proceed_when_unidentifiable=True)
print(identified_estimand)
```

```
Estimand type: EstimandType.NONPARAMETRIC_ATE

### Estimand : 1
Estimand name: backdoor
Estimand expression:
              d
────────────────────────────────(E[is_canceled|days_in_waiting_list,booking_changes,
d[different_room_assigned]

↪
↪ lead_time,total_stay,guests,total_of_special_requests,required_car_parking_s
```

Skip to main content

```
↪
↪ paces,hotel,is_repeated_guest])
↪
Estimand assumption 1, Unconfoundedness: If U→{different_room_assigned} and U→i

### Estimand : 2
Estimand name: iv
No such variable(s) found!

### Estimand : 3
Estimand name: frontdoor
No such variable(s) found!
```

Step-3. Estimate the identified estimand

[16]:
```
estimate = model.estimate_effect(identified_estimand,
                                 method_name="backdoor.propensity_score_weighti
# ATE = Average Treatment Effect
# ATT = Average Treatment Effect on Treated (i.e. those who were assigned a dif
# ATC = Average Treatment Effect on Control (i.e. those who were not assigned a
print(estimate)
```

```
*** Causal Estimate ***

## Identified estimand
Estimand type: EstimandType.NONPARAMETRIC_ATE

### Estimand : 1
Estimand name: backdoor
Estimand expression:
            d
──────────────────────────(E[is_canceled|days_in_waiting_list,booking_changes,
d[different_room_assigned]

↪
↪ lead_time,total_stay,guests,total_of_special_requests,required_car_parking_s
↪


↪
↪ paces,hotel,is_repeated_guest])
↪
Estimand assumption 1, Unconfoundedness: If U→{different_room_assigned} and U→i

## Realized estimand
b: is_canceled~different_room_assigned+days_in_waiting_list+booking_changes+lea
Target units: ate

## Estimate
Mean value: -0.2622285649999514


/github/home/.cache/pypoetry/virtualenvs/dowhy-oN2hW5jr-py3.8/lib/python3.8/sit
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Skip to main content

```
    Please also refer to the documentation for alternative solver options:
        https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
      n_iter_i = _check_optimize_result(
```

The result is surprising. It means that having a different room assigned *decreases* the chances of a cancellation. There's more to unpack here: is this the correct causal effect? Could it be that different rooms are assigned only when the booked room is unavailable, and therefore assigning a different room has a positive effect on the customer (as opposed to not assigning a room)?

There could also be other mechanisms at play. Perhaps assigning a different room only happens at check-in, and the chances of a cancellation once the customer is already at the hotel are low? In that case, the graph is missing a critical variable on *when* these events happen. Does `different_room_assigned` happen mostly on the day of the booking? Knowing that variable can help improve the graph and our analysis.

While the associational analysis earlier indicated a positive correlation between `is_canceled` and `different_room_assigned`, estimating the causal effect using DoWhy presents a different picture. It implies that a decision/policy to reduce the number of `different_room_assigned` at hotels may be counter-productive.

Step-4. Refute results

Note that the causal part does not come from data. It comes from your *assumptions* that lead to *identification*. Data is simply used for statistical *estimation*. Thus it becomes critical to verify whether our assumptions were even correct in the first step or not!

What happens when another common cause exists? What happens when the treatment itself was placebo?

Method-1

Random Common Cause:- *Adds randomly drawn covariates to data and re-runs the analysis to see if the causal estimate changes or not. If our assumption was originally correct then the causal estimate shouldn't change by much.*

[17]:
```
refute1_results=model.refute_estimate(identified_estimand, estimate,
        method_name="random_common_cause")
print(refute1_results)
```

```
New effect:-0.26222856499995134
p value:1.0
```

Method-2

Placebo Treatment Refuter:- *Randomly assigns any covariate as a treatment and re-runs the analysis. If our assumptions were correct then this newly found out estimate should go to 0.*

[18]:
```
refute2_results=model.refute_estimate(identified_estimand, estimate,
        method_name="placebo_treatment_refuter")
print(refute2_results)
```

```
Refute: Use a Placebo Treatment
Estimated effect:-0.2622285649999514
New effect:0.05420218503316096
p value:0.0
```

Method-3

Data Subset Refuter:- *Creates subsets of the data(similar to cross-validation) and checks whether the causal estimates vary across subsets. If our assumptions were correct there shouldn't be much variation.*

[19]:
```
refute3_results=model.refute_estimate(identified_estimand, estimate,
        method_name="data_subset_refuter")
print(refute3_results)
```

```
Refute: Use a subset of data
Estimated effect:-0.2622285649999514
New effect:-0.26239836091544644
p value:0.88
```

We can see that our estimate passes all three refutation tests. This does not prove its correctness, but it increases confidence in the estimate.

‹ Previous Next