Lab2

R Markdown

```r
# Load the required libraries
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.0     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflicts to become er
```

```r
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.0.0 --
## v broom        1.0.3     v rsample      1.1.1
## v dials        1.2.0     v tune         1.1.1
## v infer        1.0.4     v workflows    1.1.3
## v modeldata    1.1.0     v workflowsets 1.0.1
## v parsnip      1.1.0     v yardstick    1.2.0
## v recipes      1.0.5
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```r
library(grf)
library(plotrix)
```

```
##
## Attaching package: 'plotrix'
##
## The following object is masked from 'package:scales':
##
##     rescale
```

```r
#Get Airbnb data
airbnb <- read_csv('data/airbnb-project-msba-sampled-10k.csv')
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
```

```
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 153995 Columns: 100
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (51): listing_url, state, city, name, summary, space, description, pict...
## dbl  (32): id, high_booking, host_id, latitude, longitude, accommodates, bat...
## lgl  (13): host_is_superhost, is_location_exact, requires_license, host_has_...
## date  (4): date, host_since, first_review, last_review
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
unique_values <- airbnb%>%
  distinct(id)

set.seed(3.14159)
sampled_values <- unique_values %>%
  pull() %>%
  sample(5000)

sampled_data <- airbnb%>%
  filter(id %in% sampled_values)
```

```
#Coding all observations after April, 2016 as 1 and 0 otherwise
df<-sampled_data%>%
mutate(treatment = ifelse(date>="2019-11-01",1,0))%>%
relocate(treatment,date)

df$price = as.numeric(gsub("\\$", "", df$price))
```

```
## Warning: NAs introduced by coercion
```

```
df1<-df%>%
  select(id,bedrooms,accommodates,minimum_nights,review_scores_rating,room_type,state,city,price,treatme

df1=drop_na(df1)
```

```
# Isolate the "treatment" as a matrix => Treatment is the deployment of the crime detection algorithm
safety <- as.matrix(df1$treatment)
```

```
# Isolate the outcome as a matrix => Outcome is the nightly price
book <- as.matrix(df1$high_booking)
```

```
# Isolate the subject (id) as a matrix
id <- as.matrix(df1$id)
```

```
# Use model.matrix to create a predictor matrix from the training data
X <- model.matrix(lm(high_booking ~ -1+bedrooms+accommodates
                     +minimum_nights+review_scores_rating+factor(room_type)+
                       factor(state)+factor(city)+price, data = df1))
```

```
#Estimating Causal Forest
cf <- causal_forest(X, book, safety, num.trees = 5000, seed = 3.14159, clusters = id)
```

```
#Calculate Average Treatment Effect on our outcome variable
average_treatment_effect(cf,target.sample=c("overlap"),method = c("AIPW"))
```

```
##    estimate     std.err
## 0.066363372 0.009793245
```

```r
# Split the data into training and testing
df1=df1[-c(8)]
set.seed(3.14159)
df1.split <- group_initial_split(df1,id)
df1.train <- training(df1.split)
df1.test <- testing(df1.split)
```

```r
# Isolate the "treatment" as a matrix
safety <- as.matrix(df1.train$treatment)
```

```r
# Isolate the outcome as a matrix
book <- as.matrix(df1.train$high_booking)
```

```r
# Isolate the subject (id) as a matrix
id <- as.matrix(df1.train$id)
```

```r
# Use model.matrix to create a predictor matrix from the training data
X2 <- model.matrix(lm(high_booking ~ -1+bedrooms+accommodates
                     +minimum_nights+review_scores_rating+factor(room_type)+
                       factor(state)+price, data = df1.train))
```

```r
#Estimating Causal Forest
cf_split <- causal_forest(X2, book, safety, num.trees = 5000, seed = 3.14159, clusters = id)
```

```r
# Use model.matrix to create a predictor matrix from the testing data
X2.test <- model.matrix(lm(high_booking ~ -1+bedrooms+accommodates
                     +minimum_nights+review_scores_rating+factor(room_type)+
                       factor(state)+price, data = df1.test))
```

```r
df1.test <-
  predict(cf_split, X2.test) %>%
  bind_cols(df1.test)

df1.test%>%
  summarise(ATE = mean(predictions),se=std.error(predict(cf_split, X2.test)))
```

```
##          ATE           se
## 1 0.06164293 0.0005811336
```

```r
#Heterogeneity of ATE for Airbnbs with few or many bedrooms
average_treatment_effect(cf,target.sample=c("overlap"),method = c("AIPW"),subset=X[,1]>1)
```

```
##   estimate    std.err
## 0.04607106 0.01415550
```

```r
#Heterogeneity of ATE for Airbnbs that accommodate more or less people
average_treatment_effect(cf,target.sample=c("overlap"),method = c("AIPW"),subset=(X[,2]>10 | X[,2]<5))
```

```
##   estimate    std.err
## 0.07193539 0.01194974
```

```r
#Heterogeneity of ATE for Airbnbs which have Low or high minimum nights requirements
average_treatment_effect(cf,target.sample=c("overlap"),method = c("AIPW"),subset=(X[,3]>466 | X[,3]<233
```

```
##    estimate     std.err
## 0.066197817 0.009802044
```

```r
#Heterogeneity of ATE for Higher or lower rated Airbnbs
average_treatment_effect(cf,target.sample=c("overlap"),method = c("AIPW"),subset=(X[,4]>66 | X[,4]<33))
```

```
##    estimate     std.err
## 0.066122149 0.009815698
```

```r
#Heterogeneity of ATE Whether Airbnbs are the entire house or not
average_treatment_effect(cf,target.sample=c("overlap"),method = c("AIPW"),subset=(X[,5]==1))
```

```
##   estimate    std.err
## 0.06659807 0.01116819
```

```r
#Heterogeneity of ATE For three states and cities in different parts of the country
#State CO
average_treatment_effect(cf,target.sample=c("overlap"),method = c("AIPW"),subset=(X[,9]==1))
```

```
##  estimate   std.err
## 0.1157297 0.0418297
```

```r
#State DC
average_treatment_effect(cf,target.sample=c("overlap"),method = c("AIPW"),subset=(X[,10]==1))
```

```
##    estimate     std.err
## -0.01467874  0.04727981
```

```r
#State FL
average_treatment_effect(cf,target.sample=c("overlap"),method = c("AIPW"),subset=(X[,11]==1))
```

```
##   estimate    std.err
## 0.02845460 0.02040599
```

```r
#City Austin
average_treatment_effect(cf,target.sample=c("overlap"),method = c("AIPW"),subset=(X[,27]==1))
```

```
##   estimate    std.err
## 0.09007115 0.05856985
```

```r
#City Boston
average_treatment_effect(cf,target.sample=c("overlap"),method = c("AIPW"),subset=(X[,28]==1))
```

```
##    estimate     std.err
## -0.13527132  0.08590961
```

```r
#City Chicago
average_treatment_effect(cf,target.sample=c("overlap"),method = c("AIPW"),subset=(X[,31]==1))
```

```
##   estimate    std.err
## 0.03434872 0.04926678
```

```r
#Heterogeneity of ATE for One source of heterogeneity you think is interesting to check(price>500)
average_treatment_effect(cf,target.sample=c("overlap"),method = c("AIPW"),subset=(X[,54]>500))
```

```
##   estimate    std.err
## 0.07174725 0.04804818
```