Data Wrangling (Data Preprocessing)

Code

Practical assessment 1

Wong Yi Wei

Setup

Hide

```
library(kableExtra)
library(magrittr)
library(readr)
library(xlsx)
library(readxl)
library(foreign)
library(rvest)
library(knitr)
library(dplyr)
library(lubridate)
library(priceR)
library(wakefield)
```

Student names, numbers and percentage of contributions

Group information

Student name	Student number	Percentage of contribution
Wong Yi Wei	s3966890	50%
Adeyinka Kayode Freeman	s3960988	50%

Data Description

The data selected for this Assessment in Data Wrangling was downloaded from Kaggle. (https://www.kaggle.com/datasets/ahsan81/hotel-reservations-classification-dataset?resource=download (https://www.kaggle.com/datasets/ahsan81/hotel-reservations-classification-dataset?resource=download)) (Raza 2023). The Data was scraped and put together by Ahsan Raza to initiate the conversation and research on how the following impacts online hotel reservations and revenue:

- Hotel reservation procedure
- The channels for reservation and booking
- Period booked (Year/Month/day)
- Cancellations

The dataset has 36,275 unique records with 19 columns. The data dictionary of the data set is as explained below (Raza 2023):

- Booking_ID: This captures the unique identifier for each row in the dataset.

- no_of_adults: This signifies the number of adults in the reservation.
- no_of_children: This signifies the number of childrens in the reservation.
- no_of_weekend_nights: This signifies the number of weekend nights (Saturday or Sunday) the guest stayed or booked to stay at the hotel.
- no_of_week_nights: This signifies the number of weeknights (Monday to Friday) the guest stayed or booked to stay at the hotel.
- type_of_meal_plan: This signifies the type of meal plan booked by the customer (type includes meal plan 1, meal plan 2, meal plan 3, and not selected).
- required_car_parking_space: This signifies if the customers require a car parking space? (0 - No, 1- Yes).
- room_type_reserved: This signifies the type of room reserved by the customer. The levels are from Room types 1 to 7.
- lead_time: This signifies the number of days between the date of booking and the arrival date.
- arrival_year: The year of the arrival date of the guest.
- arrival_month: The month of arrival month of the guest.
- arrival_date: This arrival day of the guest.
- market_segment_type: This signifies the market segment designation. (It is segmented into: - offline, online, corporate, aviation, and complementary).
- repeated_guest: This signifies if the customer a repeated guest? (0 - No, 1- Yes).
- no_of_previous_cancellations: This signifies the number of previous bookings that were cancelled by the customer prior to the current booking.
- no_of_previous_booking_not_cancelled: This signifies the number of previous bookings not cancelled by the customer prior to the current booking.
- avg_price_per_room: This signifies the average price per day of the reservation; prices of the rooms are dynamic (in euros).
- no_of_special_requests: This signifies total number of special requests made by the customer (e.g., high floor, view from the room, etc.)
- booking_status: This signifies whether or not the booking made by the guest was cancelled or not.

Read/Import Data

```
[1] "C:/Users/ethan/OneDrive - RMIT University/Data Wrangling/Practical Assessment 1"
```

Hide

```
hotel<-read_csv(file="C:/Users/ethan/OneDrive - RMIT University/Data Wrangling/Practical Assessment 1/Hotel Reservations.csv")
```

```
Rows: 36275 Columns: 19── Column specification ────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────────────────────────
Delimiter: ","
chr  (5): Booking_ID, type_of_meal_plan, room_type_reserved, market_segment_type, booking_status
dbl (14): no_of_adults, no_of_children, no_of_weekend_nights, no_of_week_nights, required_car_parking_spac
e, lead_time, arrival_year, arrival_month, arrival_date, repeated_guest,...
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.
```

Hide

```
head(hotel,3)
```

Booking_ID <chr>	no_of_adults <dbl>	no_of_children <dbl>	no_of_weekend_nights <dbl>	no_of_week_nights <dbl>	type_of_meal_pla <chr>
INN00001	2	0	1	2	Meal Plan 1
INN00002	2	0	2	3	Not Selected
INN00003	1	0	2	1	Meal Plan 1

3 rows | 1-6 of 19 columns

- The recommended data types of variable in the data set is explained below:
    - Booking ID is the ID of each booking, and should be a character variable as it contains alphanumeric values
    - no_of_adults, no_of_children, no_of_wee end_nights, no_of_week_nights should be a numerical data as it contains numbers.
    - type_of_meal_plan should be a factor variable with levels of meal plan 1, meal plan 2, meal plan 3, and not selected.
    - required_car_parking_space should be a factor variable, 0 as no, and 1 as yes.
    - room_type_reserved should be a factor variable with levels from Room Types 1 to 7.
    - lead_time should be a numerical variable as it only contains numbers.
    - arrival_year, arrival_month, and arrival_date can be in a new column with the data type being a date variable.
    - market_segment_type should be a factor variable: offline, online, corporate, aviation, and complementary.
    - repeated_guest should be a factor variable, 0 as no, and 1 as yes.
    - no_of_previous_cancellations should be a numerical variable as it contains numbers only.
    - no_of_previous_bookings_not_cancelled should be a numerical variable as it contains numbers only.
    - avg_price_per_room should be a numerical variable as it is the average price per day in euros, however, currency format could also be added into the data.
    - no_of_special_requests should be a numerical value as it contains numbers only.
    - booking_status should be a factor variable with levels of not cancelled and cancelled.
- Firstly, we import the data using the read_csv function and labeled the data as hotel.
- We then print out the first 3 contents of the data by using the head() function with a argument of 3.

Inspect and Understand

Hide

```
str(hotel)
```

```
spc_tbl_ [36,275 × 19] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ Booking_ID                          : chr [1:36275] "INN00001" "INN00002" "INN00003" "INN00004" ...
 $ no_of_adults                        : num [1:36275] 2 2 1 2 2 2 2 2 3 2 ...
 $ no_of_children                      : num [1:36275] 0 0 0 0 0 0 0 0 0 0 ...
 $ no_of_weekend_nights                : num [1:36275] 1 2 2 0 1 0 1 1 0 0 ...
 $ no_of_week_nights                   : num [1:36275] 2 3 1 2 1 2 3 3 4 5 ...
 $ type_of_meal_plan                   : chr [1:36275] "Meal Plan 1" "Not Selected" "Meal Plan 1" "Meal Pla
n 1" ...
 $ required_car_parking_space          : num [1:36275] 0 0 0 0 0 0 0 0 0 0 ...
 $ room_type_reserved                  : chr [1:36275] "Room_Type 1" "Room_Type 1" "Room_Type 1" "Room_Type
1" ...
 $ lead_time                           : num [1:36275] 224 5 1 211 48 346 34 83 121 44 ...
 $ arrival_year                        : num [1:36275] 2017 2018 2018 2018 2018 ...
 $ arrival_month                       : num [1:36275] 10 11 2 5 4 9 10 12 7 10 ...
 $ arrival_date                        : num [1:36275] 2 6 28 20 11 13 15 26 6 18 ...
 $ market_segment_type                 : chr [1:36275] "Offline" "Online" "Online" "Online" ...
 $ repeated_guest                      : num [1:36275] 0 0 0 0 0 0 0 0 0 0 ...
 $ no_of_previous_cancellations        : num [1:36275] 0 0 0 0 0 0 0 0 0 0 ...
 $ no_of_previous_bookings_not_canceled: num [1:36275] 0 0 0 0 0 0 0 0 0 0 ...
 $ avg_price_per_room                  : num [1:36275] 65 106.7 60 100 94.5 ...
 $ no_of_special_requests              : num [1:36275] 0 1 0 0 0 1 1 1 1 3 ...
 $ booking_status                      : chr [1:36275] "Not_Canceled" "Not_Canceled" "Canceled" "Canceled"
...
 - attr(*, "spec")=
  .. cols(
  ..   Booking_ID = col_character(),
  ..   no_of_adults = col_double(),
  ..   no_of_children = col_double(),
  ..   no_of_weekend_nights = col_double(),
  ..   no_of_week_nights = col_double(),
  ..   type_of_meal_plan = col_character(),
  ..   required_car_parking_space = col_double(),
  ..   room_type_reserved = col_character(),
  ..   lead_time = col_double(),
  ..   arrival_year = col_double(),
  ..   arrival_month = col_double(),
  ..   arrival_date = col_double(),
  ..   market_segment_type = col_character(),
  ..   repeated_guest = col_double(),
  ..   no_of_previous_cancellations = col_double(),
  ..   no_of_previous_bookings_not_canceled = col_double(),
  ..   avg_price_per_room = col_double(),
  ..   no_of_special_requests = col_double(),
  ..   booking_status = col_character()
  .. )
 - attr(*, "problems")=<externalptr>
```

Hide

```
dim(hotel)
```

```
[1] 36275    19
```

Hide

```
hotel$type_of_meal_plan<-as.factor(hotel$type_of_meal_plan)

hotel$required_car_parking_space<-factor(hotel$required_car_parking_space,levels=c("0","1"),labels=(c("N
o","Yes")))

hotel$room_type_reserved<-factor(hotel$room_type_reserved,levels=c("Room_Type 1","Room_Type 2","Room_Type
3","Room_Type 4","Room_Type 5","Room_Type 6","Room_Type 7"),labels=c("Type 1","Type 2","Type 3","Type 4","T
ype 5","Type 6","Type 7"))

hotel$arrival<-as.Date(with(hotel, paste(arrival_year,arrival_month,arrival_date,sep="-")),"%Y-%m-%d")
hotel<-hotel %>% relocate(arrival, .before=market_segment_type)

hotel$market_segment_type<-as.factor(hotel$market_segment_type)

hotel$repeated_guest<-factor(hotel$repeated_guest,levels=c("0","1"),labels=c("No","Yes"))

hotel$avg_price_per_room<-format_currency(hotel$avg_price_per_room,"€",2)

hotel$booking_status<-factor(hotel$booking_status,levels = c("Not_Canceled", "Canceled"),labels = c("Not Ca
ncelled","Cancelled"))

colnames(hotel)<-c("Booking ID","No of Adults","No of Children","No of Weekend Nights","No of Week Night
s","Meal Plan Type","Parking","Room Type","Lead Time","Year","Month","Day","Arrival Date","Segment Type","R
epeated Guest","No of Previous Cancellations","No of Previous Non-Cancellations","Average Price per Roo
m","No of Special Requests","Booking Status")

str(hotel)
```

```
tibble [36,275 × 20] (S3: tbl_df/tbl/data.frame)
 $ Booking ID                       : chr [1:36275] "INN00001" "INN00002" "INN00003" "INN00004" ...
 $ No of Adults                     : num [1:36275] 2 2 1 2 2 2 2 2 3 2 ...
 $ No of Children                   : num [1:36275] 0 0 0 0 0 0 0 0 0 0 ...
 $ No of Weekend Nights             : num [1:36275] 1 2 2 0 1 0 1 1 0 0 ...
 $ No of Week Nights                : num [1:36275] 2 3 1 2 1 2 3 3 4 5 ...
 $ Meal Plan Type                   : Factor w/ 4 levels "Meal Plan 1",..: 1 4 1 1 4 2 1 1 1 1 ...
 $ Parking                          : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ Room Type                        : Factor w/ 7 levels "Type 1","Type 2",..: 1 1 1 1 1 1 1 4 1 4 ...
 $ Lead Time                        : num [1:36275] 224 5 1 211 48 346 34 83 121 44 ...
 $ Year                             : num [1:36275] 2017 2018 2018 2018 2018 ...
 $ Month                            : num [1:36275] 10 11 2 5 4 9 10 12 7 10 ...
 $ Day                              : num [1:36275] 2 6 28 20 11 13 15 26 6 18 ...
 $ Arrival Date                     : Date[1:36275], format: "2017-10-02" "2018-11-06" "2018-02-28" "2018-05
-20" ...
 $ Segment Type                     : Factor w/ 5 levels "Aviation","Complementary",..: 4 5 5 5 5 5 5 5 4 5
...
 $ Repeated Guest                   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ No of Previous Cancellations     : num [1:36275] 0 0 0 0 0 0 0 0 0 0 ...
 $ No of Previous Non-Cancellations: num [1:36275] 0 0 0 0 0 0 0 0 0 0 ...
 $ Average Price per Room           : chr [1:36275] "€65.00" "€106.68" "€60.00" "€100.00" ...
 $ No of Special Requests           : num [1:36275] 0 1 0 0 0 1 1 1 1 3 ...
 $ Booking Status                   : Factor w/ 2 levels "Not Cancelled",..: 1 1 2 2 2 2 1 1 1 1 ...
```

- Firstly, we use the str() function to check the data set structure and the variable data types, as well as to check which variable data types should be converted.
- By using the dim() function, we can double check the number of observations and the number of variables in the data set, which is 36275 observations and 19 variables.
- type_of_meal_plan is shown in the str() function as a character, therefore we can convert this data type into a factor data by using the as.factor() function as we do not need to change the labels of the data.
- In room_type_reserved, the str() function shows us that it is a numerical data. However, we have mentioned that this data is a factor variable with 0 as no, and 1 as yes. Therefore, we used the factor() function and labeled the data appropriately.
- Next, we have created a new variable called Arrival and used the as.Date() and paste() function for easier reading of the date of the booking. Furthermore, we have moved the newly created variable right after arrival_date.
- In market_segment_type, since there is no need to relabel the data, we can use the as.factor() function to convert the data.
- In repeated_guest, similarly to room_type_reserved, we used the factor() function and applied the appropriate labels to the data where 0 is no, and 1 is yes.
- For avg_price_per_room, we used the format_currency() function from the priceR package to label the data with a euro symbol in front of it, with 2 decimal points as it is a currency.
- Next, for the booking_status, we used the factor() function and relabeled the data as necessary.
- By using the colnames() function, we renamed the column names to a more appropriate name.
- Lastly, we can use the str() function to check if the changes were made for the data type. Looking at the ouput of the function, we can proceed with the next task as all the appropriate conversions and renaming were successfuly made.

Subsetting

Hide

```
hoteldf<-head(hotel,10)
hoteldf
```

Booking ID <chr>	No of Adults <dbl>	No of Children <dbl>	No of Weekend Nights <dbl>	No of Week Nights <dbl>	Meal Plan Type <fctr>
INN00001	2	0	1	2	Meal Plan 1
INN00002	2	0	2	3	Not Selected
INN00003	1	0	2	1	Meal Plan 1
INN00004	2	0	0	2	Meal Plan 1
INN00005	2	0	1	1	Not Selected
INN00006	2	0	0	2	Meal Plan 2
INN00007	2	0	1	3	Meal Plan 1
INN00008	2	0	1	3	Meal Plan 1
INN00009	3	0	0	4	Meal Plan 1
INN00010	2	0	0	5	Meal Plan 1

1-10 of 10 rows | 1-6 of 20 columns

Hide

```
hotelmat<-as.matrix(hoteldf)
str(hotelmat)
```

```
 chr [1:10, 1:20] "INN00001" "INN00002" "INN00003" "INN00004" "INN00005" "INN00006" "INN00007" "INN00008"
"INN00009" "INN00010" "2" "2" "1" "2" "2" "2" "2" "2" "3" "2" "0" "0" ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:20] "Booking ID" "No of Adults" "No of Children" "No of Weekend Nights" ...
```

- Firstly to subset the hotel data, we assign the first 10 observations by using the head() function to a new data named hoteldf. After this we can use hoteldf to view the contents of the data we have created.
- To convert the data to a matrix, we use the as.matrix() function to convert it, and we named it as hotelmat.
- Lastly, by using the str() function on hotelmat, we can see the structure of the data. Here we noticed that all the data types are in characters. The reason for this is that is because since a matrix is a collection of elements in a data arranged in a 2 dimensional layout, R specifies that all the elements of the matrix must be of the same class. Since the data here contains alphanumeric letters, R has automatically applied a conversion of all data in the matrix to be a character data type (Data Wrangling (Preprocessing) n.d.).

Create a new Data Frame

Hide

```
df1<-data.frame("VisitorID"=1:10,
                "Satisfaction"=r_sample_factor(x=c("Very Satisfied","Satisfied",
                "Not Satisfied"),n=10),
                stringsAsFactors = TRUE)
str(df1)
```

```
'data.frame':	10 obs. of  2 variables:
 $ VisitorID   : int  1 2 3 4 5 6 7 8 9 10
 $ Satisfaction: Factor w/ 3 levels "Very Satisfied",..: 1 1 1 2 2 1 1 2 3 1
  ..- attr(*, "varname")= chr "Factor"
```

Hide

```
levels(df1$Satisfaction)
```

```
[1] "Very Satisfied" "Satisfied"      "Not Satisfied"
```

Hide

```
no_of_guests<-round(runif(10,min=1,max=5),digits=0)

df2<-cbind(df1,no_of_guests)

colnames(df2)<-c("VisitorID","Satisfaction Level","Number of Guests")

df2
```

VisitorID <int>	Satisfaction Level <S3: variable>	Number of Guests <dbl>
1	Very Satisfied	5
2	Very Satisfied	3
3	Very Satisfied	1
4	Satisfied	4
5	Satisfied	4
6	Very Satisfied	5
7	Very Satisfied	4
8	Satisfied	2
9	Not Satisfied	3
10	Very Satisfied	3

1-10 of 10 rows

- Firstly, we will create a new data frame named df1, and we will input the data here by using the data.frame() function. The first column, VisitorID can be inputted by using a default 1:10 value. The second column, Satisfaction can be randomly generated by using the r_sample_factor() function from the wakefield package. In the Satisfaction column, we have specified the observations to only contain Very Satisfied, Satisfied, and Not Satisfied as a ordinal data, and to only include 10 observations. Furthermore, we can write down the stringsAsFactors = True argument from the data.frame function to specify that the character vectors are to be converted into factors.

- Next, we can use the str() function to view the structure of the data frame. Here we can see that the structure for VisitorID is a integer, and Satisfaction is a Factor with 3 Levels. Besides that, we also used the levels() function to view the levels of the variable. From here we can see that the 3 levels of Very Satisfied, Satisfied, and Not Satisfied.
- Before we add a new column into the data, we have to create a new vector to be inputted into the data frame. Here we created a new vector, no_of_guests, and used the runif() function to randomly generate 10 observations with a minimum number of 1 and a maximum number of 5. Since using the runif() function only would return values with decimal points, we used the round() function to remove the decimal numbers as numbers of guests should be whole numbers only.
- To add the newly created vector into the data frame, we can use the cbind() function and assign into df2 and add the new vector into the previously created data frame df1. By using the cbind() function, we have binded the new columns into the previously created data frame.
- Lastly, we used the colnames() function to change the name of the columns for easier understanding of the columns.

Reference List

Raza A (2023) *Hotel Reservations Dataset* [data set], Kaggle website, accessed 11 March 2023. https://www.kaggle.com/datasets/ahsan81/hotel-reservations-classification-dataset?resource=download (https://www.kaggle.com/datasets/ahsan81/hotel-reservations-classification-dataset?resource=download)

Data Wrangling (Preprocessing) (n.d.) *Module 3*, Data Wrangling (Preprocessing) website, accessed 12 March 2023, http://rare-phoenix-161610.appspot.com/secured/Module_03.html (http://rare-phoenix-161610.appspot.com/secured/Module_03.html)