Week-4: Code-along

Ho Zhi Yi

2023-09-02

II. Code to edit and execute using the Code-along.Rmd file

A. Data Wrangling

1. Loading packages (Slide #16)

```
# Load package tidyverse
library(tidyverse)
```

2. Loading data-set (Slide #16)

```
# Read data from the hotels.csv file and assign it to a variable named, "hotels"
hotels <- read.csv("hotels.csv")
hotels
```

3. List names of the variables in the data-set (Slide #19)

```
# Enter code here
names(hotels)
```

4. Glimpse of contents of the data-set (Slide #20)

```
# Enter code here
glimpse(hotels)
```

B. Choosing rows or columns

5. Select a single column (Slide #24)

```
# Enter code here
select(hotels, lead_time)
```

6. Select multiple columns (Slide #25)

```
# Enter code here
select(hotels, lead_time, agent, market_segment)
```

7. Arrange entries of a column (Slide #28)

```
# Enter code here
arrange(hotels, lead_time)
```

8. Arrange entries of a column in the descending order (Slide #30)

```
# Enter code here
arrange(hotels, desc(lead_time))
```

9. Select columns and arrange the entries of a column (Slide #31)

```
# Enter code here
arrange(
 select(hotels, lead_time),
 desc(lead_time)
)
```

10. Select columns and arrange the entries of a column using the pipe operator (Slide #37)

```
# Enter code here
hotels %>%
 select(lead_time) %>%
 arrange(desc(lead_time))
```

11. Pick rows matching a condition (Slide #44)

```
# Enter code here
hotels %>%
 filter(children >= 1) %>%
 select(hotel, children)
```

12. Pick rows matching multiple conditions (Slide #46)

```
# Enter code here
hotels %>%
 filter(children >= 1,hotel == "City Hotel") %>%
 select(hotel, children)
```

13. Non-conditional selection of rows: sequence of indices (Slide #49)

```
# Enter code here
hotels %>% slice(1:5)
```

14. Non-conditional selection of rows: non-consecutive/specific indices (Slide #50)

```
# Enter code here
hotels %>%
 slice(1,3,5)
```

15. Pick unique rows using distinct() (Slide #52)

```
# Enter code here
hotels %>% distinct(hotel)
```

C. Creating new columns

16. Creating a single column with mutate() (Slide #56)

```
# Enter code here
hotels %>%
 mutate(little_ones = children + babies) %>%
 select(hotel, little_ones,children,babies)
```

17. Creating multiple columns with mutate() (Slide #58)

```
# Enter code here
hotels %>%
 mutate(little_ones = children + babies,
 average_little_ones = mean(little_ones)) %>%
 select(hotel, little_ones,children,babies, average_little_ones)
```

D. More operations with examples

18. count() to get frequencies (Slide #60)

```
# Enter code here
hotels %>%
 count(market_segment)
```

19. count() to get frequencies with sorting of count (Slide #61)

```
# Enter code here
hotels %>%
 count(market_segment, sort = TRUE) # <-- decreasing order of counts
```

20. count() multiple variables (Slide #62)

```
# Enter code here
hotels %>%
 count(hotel, market_segment)
```

21. summarise() for summary statistics (Slide #63)

```
# Enter code here
# mean average daily rate for all bookings
hotels %>%
 summarise(mean_adr = mean(adr))
```

22. summarise() by using group_by to find mean (Slide #64)

```
# Enter code here
# mean average daily rate for all booking at city and resort hotels
hotels %>%
 group_by(hotel) %>%
 summarise(mean_adr = mean(adr))
```

23. summarise() by using group_by to get count (Slide #65)

```
# Enter code here
hotels %>%
 group_by(hotel) %>%
 summarise(count = n())
```

24. summarise() for multiple summary statistics (Slide #67)

```
# Enter code here
hotels %>%
 summarise(
 min_adr = min(adr),
 mean_adr = mean(adr),
 median_adr = median(adr),
 max_adr = max(adr)
 )
```

25. select(), slice() and arrange() (Slide #68)

```
# Enter code here
hotels %>%
 select(hotel, lead_time) %>%
 slice(1:5) %>%
 arrange(lead_time)
```

26. select(), arrange() and slice() (Slide #69)

```
# Enter code here
hotels %>%
 select(hotel, lead_time) %>%
 arrange(lead_time) %>%
 slice(1:5)
```

27. filter() to select rows based on conditions (Slide #73)

```
# Enter code here
hotels %>%
 filter(
 adults == 0,
 children >= 1
 ) %>%
 select(adults, babies, children)
```

28. filter() to select rows based on complicated conditions (Slide #74)

```
# Enter code here
hotels %>%
 filter( adults == 1,
 children >= 1 | babies >=1) %>% # | means OR
 select(adults, babies, children)
```

29. count() and arrange() (Slide #76)

```
# Enter code here
hotels %>%
 count(market_segment) %>%
 arrange(desc(n)) # <-- decreasing order of counts
```

30. mutate(), select() and arrange() (Slide #77)

```
# Enter code here
hotels %>%
 mutate(little_ones = children + babies) %>% # <---
 select(children, babies, little_ones) %>%
 arrange(desc(little_ones))
```

31. mutate(), filter() and select() (Slide #78)

```
# Enter code here
hotels %>%
 mutate(little_ones = children + babies) %>%
 filter(
 little_ones >= 1,
 hotel == "Resort Hotel"
 ) %>%
 select(hotel, little_ones)

hotels %>%
 mutate(little_ones = children + babies) %>%
 filter(
 little_ones >= 1,
 hotel == "City Hotel"
 ) %>%
 select(hotel, little_ones)
```