# Week-4: Code-along

Wong Wei Qi

2023-09-03

## 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)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✔ dplyr     1.1.2     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.3     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── 
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force
all conflicts to become errors
```

#### 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")
```

#### 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)

```
glimpse(hotels)
# Gives you a glimpse of the data set
```

## B. Choosing rows or columns

### 5. Select a single column (Slide #24)

```
hotels %>% select(hotel)
```

### 6. Select multiple columns (Slide #25)

```
select(hotels, hotel, lead_time)
```

### 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)

```
arrange(hotels, lead_time, desc = TRUE)
```

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

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

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

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

# Write each function on different lines
```

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

```
hotels %>%
  filter(children >= 1) %>%
  select(hotel, children) %>%
  arrange(desc(children))
```

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

```
hotels %>%
  filter(children >= 1, hotel == "City Hotel") %>%
  select(hotel, children) %>%
  arrange(desc(children))
```

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

```
hotels %>%
  slice(1:5)
```

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

```
hotels %>%
  slice(1,3,5)
```

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

```
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)

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

## D. More operations with examples

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

```
hotels %>%
  count(market_segment)
```

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

```
hotels %>%
  count(market_segment, sort = TRUE)
```

```
##   market_segment     n
## 1      Online TA 56477
## 2  Offline TA/TO 24219
## 3         Groups 19811
## 4         Direct 12606
## 5      Corporate  5295
## 6  Complementary   743
## 7       Aviation   237
## 8      Undefined     2
```

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

```
hotels %>%
  count(market_segment,hotel, sort = TRUE)
```

```
##    market_segment        hotel     n
## 1       Online TA   City Hotel 38748
## 2       Online TA Resort Hotel 17729
## 3   Offline TA/TO   City Hotel 16747
## 4          Groups   City Hotel 13975
## 5   Offline TA/TO Resort Hotel  7472
## 6          Direct Resort Hotel  6513
## 7          Direct   City Hotel  6093
## 8          Groups Resort Hotel  5836
## 9       Corporate   City Hotel  2986
## 10      Corporate Resort Hotel  2309
## 11  Complementary   City Hotel   542
## 12       Aviation   City Hotel   237
## 13  Complementary Resort Hotel   201
## 14      Undefined   City Hotel     2
```

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

```
hotels %>%
  summarise(mean_adr = mean(adr))

##   mean_adr
## 1 101.8311
```

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

```
# Enter code here
hotels %>%
  group_by(hotel) %>%
  summarise(mean_adr = mean(adr))

## # A tibble: 2 × 2
##   hotel        mean_adr
##   <chr>           <dbl>
## 1 City Hotel      105.
## 2 Resort Hotel     95.0
```

### 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(desc(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(hotel == "City Hotel")
```

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

```
hotels %>%
  filter(adults == 1, children >= 1 | babies >= 1) %>%
  select(adults,babies,children)
```

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

```
hotels %>%
count(market_segment) %>%
arrange(desc(n))
```

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

```
hotels %>%
  mutate(little_ones = children + babies) %>%
  select(children, babies, little_ones) %>%
  arrange(desc(little_ones))
```

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

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