Assignment 7

Veronika Palotai

11/2/2019

Exercise 1

All done although setting up an existing directory as a project was a bit different from creating a completely new one. The necessary step were 'File' -> 'New Project' -> 'Existing Directory'.

Exercise 2

All done.

Exercise 3

Import Libraries

```
library(ggplot2)
library(tidyverse)
library(scales)
```

- Import from csv

Setting the Path

```
dir <-  "D:/Egyetem/CEU/Coding_1/R-Coding/"
```

Location Folders

```
data_in <- paste0(dir,"da_data_repo/hotels-europe/clean/")
```

Loading Datasets

```
hotels_prices_csv <- read_csv(paste0(data_in,"hotels-europe_price.csv"))
hotels_features_csv <- read_csv(paste0(data_in,"hotels-europe_features.csv"))
```

- Import from .dta

Import 'haven' library and look up `read_dta`

```
library(haven)
?read_dta
```

Reading .dta files

```
hotels_prices_dta <- read_dta(paste0(data_in,"hotels-europe_price.dta"), encoding = NULL)
hotels_features_dta <- read_dta(paste0(data_in,"hotels-europe_features.dta"), encoding = NULL)
```

- Comparing contents

First, let's look at the contents of the tables.

```
View(hotels_features_dta)
View(hotels_features_csv)
View(hotels_prices_dta)
View(hotels_prices_csv)
```

We can see that the order of the fields and the records are different. Let's check the length of each table to see whether they contain equal number of observations.

```
hotels_features_csv %>%
  summarise(
    nr_of_records = n()
  )
```

```
## # A tibble: 1 x 1
##   nr_of_records
##           <int>
## 1         22902
```

```
hotels_features_dta %>%
  summarise(
    nr_of_records = n()
  )
```

```
## # A tibble: 1 x 1
##   nr_of_records
##           <int>
## 1         22902
```

```
hotels_prices_csv %>%
  summarise(
    nr_of_records = n()
  )
```

```
## # A tibble: 1 x 1
##   nr_of_records
##           <int>
## 1        148021
```

```
hotels_prices_dta %>%
  summarise(
    nr_of_records = n()
  )
```

```
## # A tibble: 1 x 1
##   nr_of_records
##           <int>
## 1        148021
```

As we can see, they contain the same number of records. Let's go into the details a bit and check the number of records for each city in the features table and each year in the prices table.

```
hotels_features_csv %>%
  group_by(city) %>%
  summarise(
    nr_of_records = n()
  )
```

```
## # A tibble: 47 x 2
##    city       nr_of_records
##    <chr>              <int>
##  1 Amsterdam            428
##  2 Athens               369
##  3 Barcelona            835
##  4 Belgrade             111
##  5 Berlin               579
##  6 Birmingham           185
##  7 Bratislava           104
##  8 Brussels             333
##  9 Bucharest            132
## 10 Budapest             340
## # ... with 37 more rows
```

```
hotels_features_dta %>%
  group_by(city) %>%
  summarise(
    nr_of_records = n()
  )
```

```
## # A tibble: 46 x 2
##    city       nr_of_records
##    <chr>              <int>
##  1 Amsterdam            429
##  2 Athens               369
##  3 Barcelona            835
##  4 Belgrade             111
##  5 Berlin               579
##  6 Birmingham           185
##  7 Bratislava           104
##  8 Brussels             333
##  9 Bucharest            132
## 10 Budapest             340
## # ... with 36 more rows
```

```
hotels_prices_csv %>%
  group_by(year) %>%
  summarise(
    nr_of_records = n()
  )
```

```
## # A tibble: 2 x 2
##    year nr_of_records
##   <dbl>         <int>
## 1  2017         62235
## 2  2018         85786
```

```
hotels_prices_dta %>%
  group_by(year) %>%
  summarise(
    nr_of_records = n()
  )
```

```
## # A tibble: 2 x 2
##    year nr_of_records
##   <dbl>         <int>
## 1  2017         62235
## 2  2018         85786
```

Most of the time they are the same in both tables, however, we can see a difference in case of 'Amsterdam' which might be due to a reading error. Since the overall number of records are the same, there has to be at least one other city with a difference.

Exercise 4

Picking the first 200 lines of the prices table

```
(hotels_prices_csv_small <- hotels_prices_csv %>%
  slice(1:200))
```

```
## # A tibble: 200 x 10
##    hotel_id price offer offer_cat  year month weekend holiday nnights
##       <dbl> <dbl> <dbl> <chr>     <dbl> <dbl>   <dbl>   <dbl>   <dbl>
##  1        1   172     0 0% no of~  2017    11       1       0       1
##  2        1   122     1 15-50% o~  2018     1       1       0       1
##  3        1   122     1 15-50% o~  2017    12       0       1       1
##  4        1   552     1 1-15% of~  2017    12       0       1       4
##  5        1   122     1 15-50% o~  2018     2       1       0       1
##  6        1   114     1 15-50% o~  2017    11       0       0       1
##  7        2   119     0 0% no of~  2017    11       0       0       1
##  8        2   119     0 0% no of~  2017    12       0       1       1
##  9        2   547     0 0% no of~  2017    12       0       1       4
## 10        3   118     1 15-50% o~  2017    12       0       1       1
## # ... with 190 more rows, and 1 more variable: scarce_room <dbl>
```

Writing to file

```
write.csv(hotels_prices_csv_small, file = "hotels-europe-small.csv", row.names = FALSE)
```

- Lines changed: I deleted the field names
- It lead to the problem that the contents of the first row were interpreted as column names
- Fixing:

```
hotels_small <- read.csv("hotels-europe-small.csv", header = FALSE)
# names columns as V1, V2, V3, ...
```

Exercise 5

12.6 Case Study

Load the dataset:

```r
who <- tidyr::who
```

Gather together all the columns from `new_sp_m014` to `newrel_f65` because they are values not variables:

```r
who1 <- who %>%
  gather(new_sp_m014:newrel_f65, key = "key", value = "cases", na.rm = TRUE)
```

Some hint on the structure of the values in the new key column by counting them:

```r
who1 %>%
  count(key)
```

```
## # A tibble: 56 x 2
##    key              n
##    <chr>        <int>
##  1 new_ep_f014   1032
##  2 new_ep_f1524  1021
##  3 new_ep_f2534  1021
##  4 new_ep_f3544  1021
##  5 new_ep_f4554  1017
##  6 new_ep_f5564  1017
##  7 new_ep_f65    1014
##  8 new_ep_m014   1038
##  9 new_ep_m1524  1026
## 10 new_ep_m2534  1020
## # ... with 46 more rows
```

Minor fix to the format of the column names:

```r
who2 <- who1 %>%
  mutate(key = stringr::str_replace(key, "newrel", "new_rel"))
```

Separate the values in each code with two passes of `separate()`. The first pass will split the codes at each underscore.

```r
who3 <- who2 %>%
  separate(key, c("new", "type", "sexage"), sep = "_")
```

Drop the 'new' column because it's constant in this dataset. Let's also drop 'iso2' and 'iso3' since they're redundant.

```r
who3 %>%
  count(new)
```

```
## # A tibble: 1 x 2
##   new       n
##   <chr> <int>
## 1 new   76046
```

```
who4 <- who3 %>%
  select(-new, -iso2, -iso3)
```

Separate ‘sexage’ into ‘sex’ and ‘age’ by splitting after the first character:

```
who5 <- who4 %>%
  separate(sexage, c("sex", "age"), sep = 1)
```

The ‘who’ dataset is now tidy, however, we could have done all these steps at the same time, like this:

```
who %>%
  gather(key, value, new_sp_m014:newrel_f65, na.rm = TRUE) %>%
  mutate(key = stringr::str_replace(key, "newrel", "new_rel")) %>%
  separate(key, c("new", "var", "sexage")) %>%
  select(-new, -iso2, -iso3) %>%
  separate(sexage, c("sex", "age"), sep = 1)
```

```
## # A tibble: 76,046 x 6
##    country      year var   sex   age   value
##    <chr>       <int> <chr> <chr> <chr> <int>
##  1 Afghanistan  1997 sp    m     014       0
##  2 Afghanistan  1998 sp    m     014      30
##  3 Afghanistan  1999 sp    m     014       8
##  4 Afghanistan  2000 sp    m     014      52
##  5 Afghanistan  2001 sp    m     014     129
##  6 Afghanistan  2002 sp    m     014      90
##  7 Afghanistan  2003 sp    m     014     127
##  8 Afghanistan  2004 sp    m     014     139
##  9 Afghanistan  2005 sp    m     014     151
## 10 Afghanistan  2006 sp    m     014     193
## # ... with 76,036 more rows
```

Exercise 6

The dataset:

```
(df <- tibble(name = c("A123", "B456"), age = c(30, 60), answer1 = c(0, 1), answer2 = c(1,1), answer3 =
```

```
## # A tibble: 2 x 6
##   name    age answer1 answer2 answer3 answer4
##   <chr> <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 A123     30       0       1       1       0
## 2 B456     60       1       1       0       0
```

1. Cleaning by hand - description only

What needs to be done is to gather the columns ‘answers1’ to ‘answers4’ and create an ‘answer_nr’ (key) and an ‘answer_value’ (value) column for the answers and their corresponding values pertaining to a specific age group.

2. Cleaning by coding it with `gather()`

```
tidy_df <- df %>%
  gather(answer1:answer4, key="answer_nr", value="answer_value")
tidy_df
```

```
## # A tibble: 8 x 4
##   name    age answer_nr answer_value
##   <chr> <dbl> <chr>            <dbl>
## 1 A123     30 answer1              0
## 2 B456     60 answer1              1
## 3 A123     30 answer2              1
## 4 B456     60 answer2              1
## 5 A123     30 answer3              1
## 6 B456     60 answer3              0
## 7 A123     30 answer4              0
## 8 B456     60 answer4              0
```