---

title: "Lecture 7: Reading and tidying data" author: "Marc Kaufmann" date: "10/28/2019" output: html_document

HouseKeeping

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(haven)
```

Assignment 7

Exercise 1: Make sure that you create a *new* .Rmd file for your assignment. Read chapter 8 in R4DS on Workflow: projects and set up the folder for lecture 7 as a project. Make sure you open your assignment file as a new project. That way, when you try to read files in this folder, it should automatically find them.

Solution: Completed

Exercise 2: Make sure you know how to load the test files. You do not have to prove this, so if you think you are fine, simply say "All done". If not, write down where you get stuck, and explain why you expected your commands to work, or why you think there should be no error.

Solution: All Done

Exercise 3: I added the hotels-europe dataset to the lecture7 folder. Make sure that you can read in the *clean* data from the csv file. Lookup `read_dta` from the *haven* package (you have to load it first to get the help text). Use this to load the clean hotels data from the .dta files, which is the Stata native format. Compare the two datasets that you loaded, and show a few ways to test whether the two have identical data.

```
getwd()
```

```
## [1] "C:/Users/faaez/OneDrive - Central European University/Current Courses/Data Coding 1 - Data Manag
```

```
hotels_csv <- read_csv("hotels-europe_price.csv")
```

```
## Parsed with column specification:
## cols(
##   hotel_id = col_double(),
##   price = col_double(),
##   offer = col_double(),
##   offer_cat = col_character(),
##   year = col_double(),
##   month = col_double(),
##   weekend = col_double(),
##   holiday = col_double(),
##   nnights = col_double(),
##   scarce_room = col_double()
## )
```

```
hotels_dta <-read_dta("hotels-europe_price.dta")

setdiff(hotels_csv, hotels_dta) # shows no rows that are different
```

```
## Warning: Column `hotel_id` has different attributes on LHS and RHS of join
## Warning: Column `price` has different attributes on LHS and RHS of join
## Warning: Column `scarce_room` has different attributes on LHS and RHS of
## join
## Warning: Column `offer` has different attributes on LHS and RHS of join
## Warning: Column `offer_cat` has different attributes on LHS and RHS of join
## Warning: Column `year` has different attributes on LHS and RHS of join
## Warning: Column `month` has different attributes on LHS and RHS of join
## Warning: Column `weekend` has different attributes on LHS and RHS of join
## Warning: Column `holiday` has different attributes on LHS and RHS of join
## Warning: Column `nnights` has different attributes on LHS and RHS of join
## # A tibble: 0 x 10
## # ... with 10 variables: hotel_id <dbl>, price <dbl>, scarce_room <dbl>,
## #   offer <dbl>, offer_cat <chr>, year <dbl>, month <dbl>, weekend <dbl>,
## #   holiday <dbl>, nnights <dbl>
```

```
all_equal(hotels_csv, hotels_dta) #true
```

```
## Warning: Column `hotel_id` has different attributes on LHS and RHS of join
## Warning: Column `price` has different attributes on LHS and RHS of join
## Warning: Column `offer` has different attributes on LHS and RHS of join
## Warning: Column `offer_cat` has different attributes on LHS and RHS of join
## Warning: Column `year` has different attributes on LHS and RHS of join
## Warning: Column `month` has different attributes on LHS and RHS of join
## Warning: Column `weekend` has different attributes on LHS and RHS of join
## Warning: Column `holiday` has different attributes on LHS and RHS of join
## Warning: Column `nnights` has different attributes on LHS and RHS of join
## Warning: Column `scarce_room` has different attributes on LHS and RHS of
## join
## [1] TRUE
```

```
assertthat::are_equal(hotels_csv, hotels_dta) #true
```

```
## Warning: Column `hotel_id` has different attributes on LHS and RHS of join
## Warning: Column `price` has different attributes on LHS and RHS of join
## Warning: Column `offer` has different attributes on LHS and RHS of join
## Warning: Column `offer_cat` has different attributes on LHS and RHS of join
## Warning: Column `year` has different attributes on LHS and RHS of join
## Warning: Column `month` has different attributes on LHS and RHS of join
```

```
## Warning: Column `weekend` has different attributes on LHS and RHS of join

## Warning: Column `holiday` has different attributes on LHS and RHS of join

## Warning: Column `nnights` has different attributes on LHS and RHS of join

## Warning: Column `scarce_room` has different attributes on LHS and RHS of
## join

## [1] TRUE
```

Exercise 4: Pick a small subset of the hotels-europe dataset that you loaded in exercise 3 of 200 lines. Write it to a file. Now screw up some lines in the file in such a way that the normal `read_csv()` will either fail, or not yet load the dataset in the proper way. Make sure to specify:

```
df_temp <- hotels_csv %>% head(200)
write_csv(df_temp, 'messed_up_data.csv')

df_messy <- read_csv('messed_up_data.csv')
```

```
## Parsed with column specification:
## cols(
##   hotel_id = col_double(),
##   price = col_double(),
##   offer = col_double(),
##   offer_cat = col_character(),
##   year = col_double(),
##   month = col_double(),
##   weekend = col_double(),
##   holiday = col_double(),
##   nnights = col_double(),
##   scarce_room = col_double()
## )
```

1. Which lines you changed

Ans: I changes lines 13, 25, and 50.

2. Why this will lead either to a direct error or a parsing problem

Ans Line 13: It will lead to parsing errors. In line 13 I added a lot of commas instead of the data - this raises a warning that thuis line has 13 columns instead of the 10 it expected.

Line 25: In price I added a really big integer and I also appended another row with this row. The big Int in price messes up the whole price column as it causes R to read that column in standard format. Also the extra columns causes R to create 10 extra columns in the whole dataframe and put NA in all other rows for them.

Line 50: I deleted the values in some of the columns which causes a parsing error that R detected 8 columns instead of 10.

3. How you would notice the error, diagnose the problem, and fix it

Ans: After loading the dataframe I have a few default checks to make sure that the dtaframe loaded correctly, first one of these is to check the number of NA values in each column:

```
colSums(is.na(df_messy))
```

```
##    hotel_id       price       offer   offer_cat        year       month
##           0           0           0           0           0           0
##     weekend     holiday     nnights scarce_room
##           0           0           0           0
```

The high number of NA values in columns x11:x19 would make me observe the problem and drop the extra columns.

```
df_messy <- df_messy %>% select(hotel_id:scarce_room)

colSums(is.na(df_messy))
```

```
##    hotel_id       price       offer   offer_cat        year       month
##           0           0           0           0           0           0
##     weekend     holiday     nnights scarce_room
##           0           0           0           0
```

When viewing the data the problem with price can be observed with checking if a few extreme values are messing up the column and deleting them one by one or all at once and then do a subset to remove rows with an values as well.

```
view(subset(df_messy, price != max(df_messy$price, na.rm = TRUE)))
```

We had many examples of parsing problems, so take your inspiration from there or google for issues other people had.

Parsing Failures: Warning: 3 parsing failures. row col expected actual file 13 – 10 columns 36 columns 'messed_up_data.csv' 21 – 10 columns 19 columns 'messed_up_data.csv' 50 – 10 columns 8 columns 'messed_up_data.csv'

Exercise 5: Do the final case study 12.6 in R4DS.

```
who <- tidyr::who

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

```
## # A tibble: 76,046 x 6
##    country     iso2  iso3   year key         cases
##    <chr>       <chr> <chr> <int> <chr>       <int>
##  1 Afghanistan AF    AFG    1997 new_sp_m014     0
##  2 Afghanistan AF    AFG    1998 new_sp_m014    30
##  3 Afghanistan AF    AFG    1999 new_sp_m014     8
##  4 Afghanistan AF    AFG    2000 new_sp_m014    52
##  5 Afghanistan AF    AFG    2001 new_sp_m014   129
##  6 Afghanistan AF    AFG    2002 new_sp_m014    90
##  7 Afghanistan AF    AFG    2003 new_sp_m014   127
##  8 Afghanistan AF    AFG    2004 new_sp_m014   139
##  9 Afghanistan AF    AFG    2005 new_sp_m014   151
## 10 Afghanistan AF    AFG    2006 new_sp_m014   193
## # ... with 76,036 more rows
```

```
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
```

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

```
## # A tibble: 76,046 x 6
##    country     iso2  iso3   year key         cases
##    <chr>       <chr> <chr> <int> <chr>       <int>
##  1 Afghanistan AF    AFG    1997 new_sp_m014     0
##  2 Afghanistan AF    AFG    1998 new_sp_m014    30
##  3 Afghanistan AF    AFG    1999 new_sp_m014     8
##  4 Afghanistan AF    AFG    2000 new_sp_m014    52
##  5 Afghanistan AF    AFG    2001 new_sp_m014   129
##  6 Afghanistan AF    AFG    2002 new_sp_m014    90
##  7 Afghanistan AF    AFG    2003 new_sp_m014   127
##  8 Afghanistan AF    AFG    2004 new_sp_m014   139
##  9 Afghanistan AF    AFG    2005 new_sp_m014   151
## 10 Afghanistan AF    AFG    2006 new_sp_m014   193
## # ... with 76,036 more rows
```

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

```
## # A tibble: 76,046 x 8
##    country     iso2  iso3   year new   type  sexage cases
##    <chr>       <chr> <chr> <int> <chr> <chr> <chr>  <int>
##  1 Afghanistan AF    AFG    1997 new   sp    m014       0
##  2 Afghanistan AF    AFG    1998 new   sp    m014      30
##  3 Afghanistan AF    AFG    1999 new   sp    m014       8
##  4 Afghanistan AF    AFG    2000 new   sp    m014      52
##  5 Afghanistan AF    AFG    2001 new   sp    m014     129
##  6 Afghanistan AF    AFG    2002 new   sp    m014      90
##  7 Afghanistan AF    AFG    2003 new   sp    m014     127
##  8 Afghanistan AF    AFG    2004 new   sp    m014     139
##  9 Afghanistan AF    AFG    2005 new   sp    m014     151
## 10 Afghanistan AF    AFG    2006 new   sp    m014     193
## # ... with 76,036 more rows
```

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

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

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

who4
```

```
## # A tibble: 76,046 x 5
```

```
##    country      year type  sexage cases
##    <chr>       <int> <chr> <chr>  <int>
##  1 Afghanistan  1997 sp    m014       0
##  2 Afghanistan  1998 sp    m014      30
##  3 Afghanistan  1999 sp    m014       8
##  4 Afghanistan  2000 sp    m014      52
##  5 Afghanistan  2001 sp    m014     129
##  6 Afghanistan  2002 sp    m014      90
##  7 Afghanistan  2003 sp    m014     127
##  8 Afghanistan  2004 sp    m014     139
##  9 Afghanistan  2005 sp    m014     151
## 10 Afghanistan  2006 sp    m014     193
## # ... with 76,036 more rows
```

Exercise 6: Take the following dataset and tidy it:

1. First by hand: Describe what you have to do manually to tidy it. This will make it clear to you what the arguments to `gather()` are and why they are what they are.
   - Make sure you realize which data you have to repeat across rows for this to work

Ans: To manually tidy this data we will have to make a new time column and add the corresponding answer number value to it for each row and another column answer that wil record the value of the answer. As a consequence, we will have to record repeated value of name and age with each corresponding value of time.

2. By coding it with `gather()`

Here is the data, where answer1 etc are all the same type of question at different times (implicit in the name of the answer, that is the owner of the database would answer1 differs from answer2).

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

df %>% gather('answer1', 'answer2', 'answer3', 'answer4', key = 'time', value = 'answer')
```

```
## # A tibble: 8 x 4
##   name    age time    answer
##   <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
```