

Lab 2: Data preparation and data cleaning

Benjamin Lacroix

February 4, 2020

1 Loading data

Download the file 'stateDataLong.csv' from the Moodle page and put in your working directory for this lab.

Open a new Script in RStudio to type all the commands of this lab and run them.

Load the file in R using read.csv()

```
stateDataLong <- read.csv("stateDataLong.csv")
```

Task:

- Have a look at the data using the command str(), head() and summary().

It seems to be a mixture of socioeconomic and geographical data about US states. This data format is sometimes called the 'long format' where each row contains one data value with some sort of indicator.

As seen last week we would prefer for analysis 'wide format'. To do that we can use the command spread() from the library 'tidyr':

```
#instal the library tidyr  
install.packages("tidyr")
```

```
#load the library  
library(tidyr)  
# use the spread command  
stateDataWide <- spread(stateDataLong, indicator, value)
```

Task:

- Have a look at the data using the command str(), head() and summary().
- Can you tell how many missing values there are in each column?

2 Missing values

2.1 Detecting missing values

The command `complete.cases()` allows you to identify the rows that have at least one missing value

```
complete.cases(stateDataWide)

## [1] TRUE FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE TRUE
## [12] FALSE FALSE FALSE TRUE TRUE FALSE FALSE TRUE FALSE TRUE TRUE
## [23] FALSE TRUE FALSE FALSE FALSE TRUE TRUE FALSE FALSE TRUE TRUE
## [34] TRUE TRUE TRUE FALSE TRUE FALSE TRUE FALSE TRUE FALSE TRUE
## [45] TRUE FALSE TRUE TRUE TRUE FALSE
```

You can apply the `sum()` function to count the number of incomplete cases:

```
# number of complete cases
sum(complete.cases(stateDataWide))

## [1] 25
```

You can obtain use the 'not' sign `!` with boolean variables to flip them

```
# cases with missing values
!complete.cases(stateDataWide)

## [1] FALSE TRUE TRUE TRUE TRUE FALSE TRUE FALSE TRUE TRUE FALSE
## [12] TRUE TRUE TRUE FALSE FALSE TRUE TRUE FALSE TRUE FALSE FALSE
## [23] TRUE FALSE TRUE TRUE TRUE FALSE FALSE TRUE TRUE FALSE FALSE
## [34] FALSE FALSE FALSE TRUE FALSE TRUE FALSE TRUE FALSE TRUE FALSE
## [45] FALSE TRUE FALSE FALSE FALSE TRUE

# number of cases with missing values
sum(!complete.cases(stateDataWide))

## [1] 25
```

Display complete cases from the data frame:

```
stateDataWide[complete.cases(stateDataWide),]
```

Task:

- Display all the INCOMPLETE cases

2.2 Imputation

Let's have a look at ways of replacing missing values. The library `tidyr` has a function called `replace_na()` which automatically replaces missing values with a given value. Let say for instance we want to replace the missing values for illiteracy by 0:

```
replace_na(stateDataWide$Illiteracy,0)

## [1] 2.1 0.0 0.0 1.9 1.1 0.7 1.1 0.9 1.3 2.0 1.9 0.6 0.9 0.7 0.5 0.6 1.6
## [18] 2.8 0.7 0.9 1.1 0.9 0.6 2.4 0.0 0.0 0.6 0.5 0.7 1.1 0.0 1.4 1.8 0.8
## [35] 0.8 1.1 0.6 1.0 0.0 2.3 0.5 1.7 2.2 0.6 0.6 1.4 0.6 1.4 0.7 0.6
```

Another (and probably smarter) option would be to replace those missing values with the mean of that column:

```
stateDataWide$Illiteracy <- replace_na(stateDataWide$Illiteracy,mean(stateDataWide$Illiteracy))
```

Task:

- Replace the missing values in the Income column with the minimum value of this column (use function `min()`), and replace the missing values in the Murder column with the median value of this column (use function `median()`)

3 Data transformation

The complete data is actually available in R:

```
stateDataComplete <- data.frame(state.x77)
```

Let's create a new feature, the area in square kilometer:

```
stateDataComplete[, "Area.km2"] <- stateDataComplete[, "Area"] * 1.609^2
```

Task:

- Create a new feature called `population.density`. It corresponds to the number of inhabitant per square kilometer.

The `dplyr` library provides a number of helpful functions for this sort of transformation (look up `mutate` or `transform`)

```
# install dplyr
install.packages("dplyr")
```

```
#load dplyr
library(dplyr)
```

Here we combine mutate (creates a new column from existing ones) with the case_when() construct (assigns a value according to the first true statement.

```
stateDataComplete <- mutate(stateDataComplete,
  size = case_when(Area > 100000 ~ "huge", Area > 50000 ~ "medium", TRUE ~ "small"))
```

Convert the column 'size' to an ordered factor:

```
stateDataComplete$size <- factor(stateDataComplete$size,
  levels = c("small", "medium", "large"),
  ordered = TRUE)
```

Task:

- The HS.Grad is a percentage, change the values in this column to be proportions (between 0 and 1) instead of percentages (between 0 and 100)
- Add a column using mutate() and case_when() that has the following categories:
 - "Dangerous" when Murder_i > 10
 - "Unhealthy" when not "Dangerous" but Life.Exp_i < 70
 - "Fine" otherwise.

4 Dates and Time

Just a little exercise about dates and times. **Task :**

- Convert the following strings into POSIXct objects.
- Print those objects back in the same format of your choice (using the function format()).

```
d1 <- "March 29 1985 12:00:00"
d2 <- "Jun 30, 1985 15:33:03"
d3 <- "Jun 30, 1985 09:00:00 pm"
d4 <- "04/29/1985 12:31:01.543"
d5 <- "29-10-1985 12:31:01"
d6 <- 1104295502
```

If your output is NA, it means you did something wrong.

Tip: you can bind the conversion specification in the help of the function format.POSIXct:

```
?format.POSIXct
```