Introduction to R programming: Data preprocessing

Bernard Silenou[1] and Henrik Schanze[1]

[1]Department of Epidemiology, Helmholtz Centre for Infection Research, Braunschweig, Germany

November 28, 2023

Contents

Welcome to this R course on preparing data for analysis (data preprocessing). Before proceeding with this course we recommend that you to get familiar with R-Studio (or what ever IDE you are using) and the content covered in the first chapter of the course titled "Basic R".

1 Loading packages and data into R

Goals

- Install and load packages
- Import data into R

1.1 Loading packages

Before loading a package to your current R session, the package needs to already be installed to your computer. Use the command `install.packages` to install a package and `library` to load a package. The RStudio IDE provides an option to search and install packages.

```
install.packages("MASS")
library(MASS)
```

The command `install.packages` would install the needed package from the default R repository called CRAN. If the package that you wish to install in not on CRAN, you would need to search for the repository hosting the package, download the tar.gz file before installing it.

1.2 Loading data

We are starting with loading the data we want to work with into R. Data could be stored in different kinds of formats. For the majority of common formats there are simple solutions to import that data.
As an example we want to use a `.csv` file which stores data about movies including the name, genre, rating and a lot more. We can import the file into R using the function `read.csv` and give it the name `dataMovies`.

```
setwd("~/Introduction-to-R-programming/lecture_notebooks")
dataMovies = read.csv("./data/movies.csv")
```

`.` in the file path represents the current working directory and can be printed using `getwd()` command.

2 Cleaning and transforming data

Goals

After reading this section, you should be able to do th following:

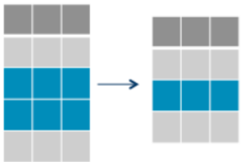
- Deduplicate a data
- Manipulate character or numeric variables
- Subset a data
- Transform variables in a data

- Convert data from wide to long formats and back
- Sort a data
- Do a single imputation

At this point, your data should be successfully loaded to your R session. Since raw data often has much noise or missing values, it is essential to be processed thoroughly and carefully before fitting a model to it.

This section showcase some of the steps (identifying outliers, error records and missing values, duplicates records, etc.) in transforming raw data into informative data for analysis.

2.1 Detecting duplicate records



Duplicate data is in reference to all or a particular variable, often the ID

Example: Identify and remove rows having the same actors `name` in `dataMovies`

```
# checking if there are duplicates
dataName = data.frame(table(dataMovies$name))
duplicateNames = dataName[dataName$Freq >1,]
dim(duplicateNames)

## [1] 149   2

View(dataMovies[dataMovies$name == "Anna",])

# Deleting rows with duplicate names at random
library(dplyr)
dataMoviesDistinctName = dplyr::distinct(dataMovies, name, .keep_all = TRUE)

# Deleting rows with duplicate year and score
dataMoviesDistinctYearScore = dplyr::distinct(dataMovies, year, score, .keep_all = TRUE)

# Deleting rows with duplicates for all the columns
dataMoviesDistinctAll = dplyr::distinct(dataMovies, .keep_all = TRUE)
nrow(dataMoviesDistinctAll) # The same as dataMovies

## [1] 7668
```

2.2 Adding and deleting colums in a dataframe

Example: Compute new variables, for for the total score and another for how old a movie is.

```
dataMovies$age  = dataMovies$yearCurrent - dataMovies$year
dataMovies$scoreTotal  = dataMovies$score + dataMovies$scoreSecond
head(dataMovies, 3)[, c(1,20:ncol(dataMovies))]
```

```
##                                              name age scoreTotal
## 1                                     The Shining  43   18.80959
## 2                                 The Blue Lagoon  43   11.00959
## 3 Star Wars: Episode V - The Empire Strikes Back  43   19.70959
```

Using the the command **within** reduces typing effort and leads to clean code.

```
dataMovies = within(
data = dataMovies,
expr = {
age = yearCurrent - year
scoreTotal  = score + scoreSecond
movies21Century = year > 2000
}
)
```

Quick exercise:

1. Extract the data call dataMoviesNegativeScore for movies with negative second score (scoreSecond) from dataMovies. Which movie has the smallest score and which has the maximum score.

2.3 Cleaning character variables

Data values can be recorded in a way that R does not understand, for example, a question that requires a TRUE or FALSE response may have been recorded as *Y* or *N*, or *Yes* or *No*.

Example:

Replace the character variable *drugUse* with the logical value *TRUE* or *FALSE*.

```
characterToLogical <- function(x){
n = length(x)
y = rep(NA, n)
y[x == "Yes"] = TRUE
y[x == "No"] <- FALSE
return(y)
}
dataMovies$drugUseLogical = (characterToLogical(dataMovies$drugUse))
```

Have a look at the sex of the director **sexDirector** and code males with **Male** and famales with **Female**.

```
table(dataMovies$sexDirector, useNA = "always") # always, to display missing values
```

```
##
##      f Female      M   male   Male   <NA>
##   1255   1311   1243   1349   1289   1221
```

```
replacing_enums <- function(x){
n = length(x)
y = rep(NA, n)
y[x %in% c("f", "Female")] = "Female" # %in% to check for multiple options
y[x %in% c("M", "male", "Male")] = "Male"
return(y)
}
dataMovies$sexDirectorClean = (replacing_enums(dataMovies$sexDirector))
```

Quick exercise: Create a dummy (indicator) variable call drugUseDummy for drugUse. Code Yes with 1 and No with 0. What is the data type of drugUseDummy?

2.4 Cleaning numeric variables

Data values can be recorded with *errors*. For example, age of 114 yrs for a person. This age value may not be an error but doesn't belong to the population that we are interested in (*outlier*)

We want to identify and either correct the age of the person (if its an error record) or drop the person from the study (outlier).

Example: Assume the required age is between 10 and 40 yrs.

Are there records having an `age` that is not between 10 and 40 yrs? If Yes, how many?

Create a new data set having only records with `age` between 10 and 40 yrs

Replace all records having age below 10 years with 10yrs and those above 40yrs with 40yrs

```
table(is.na(dataMovies$age)) # checking for missing data

##
## FALSE
##  7668

summary(dataMovies$age) # Summary of age

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3.00   13.00   23.00   22.59   32.00   43.00

# Filtering records with age less than 10 or grater than 40
dataMoviesAgeNot10to40 = dataMovies[(dataMovies$age < 10 | dataMovies$age > 40), ]
#Counting number of records
nrow(dataMoviesAgeNot10to40) # or dim(dataMoviesAgeNot10to40)[1]

## [1] 1556

# Alternately, we can negate the condition to filter for age between 10 to 40
dataMoviesAge10to40 = dataMovies[!(dataMovies$age < 10 | dataMovies$age > 40), ]
# OR
dataMoviesAge10to40 = dataMovies[(dataMovies$age >= 10 & dataMovies$age <= 40), ]

# Replacing numeric variables
# Copying age to a new column called ageImputed
# It is not advisable to manipulate existing variables
dataMovies$ageImputed = dataMovies$age
# Replacing ages less than 10 with 10
dataMovies[dataMovies$ageImputed < 10, ] = 10
# Replacing ages greater than 40 with 40
dataMovies[dataMovies$ageImputed > 40, ] = 40
```

Homework: Write a function that compute the median `score` and assign its value to all records having missing values for `score`.

2.5 Sorting

Sorting involves *arranging* data into some *meaningful order* to make it easier to understand or analyse.

Example: Sort `dataMovies` using the following variables: (a) `year` , (a) `rating` , (a) `score` and `votes`.

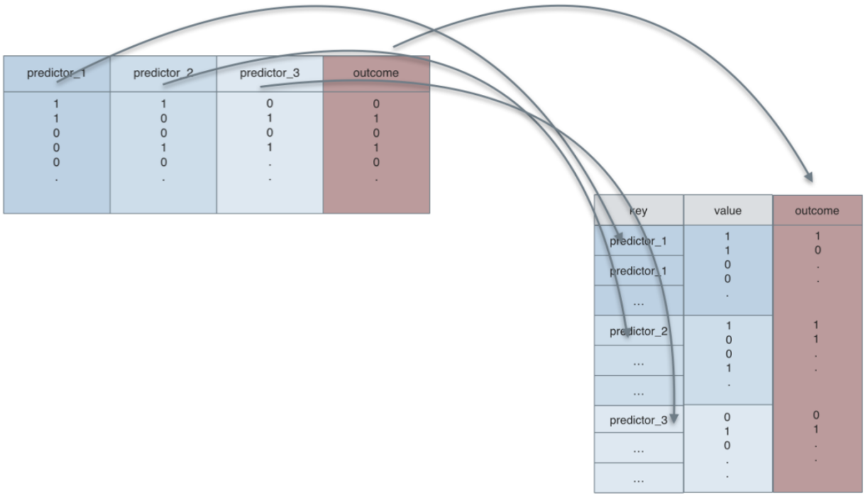
```
# Sorting by year (integer)
dataMoviesSortYear =  dataMovies[order(dataMovies$year), ]
# Use vector[order(vector)] # to sort a vector

# Sorting by rating (character)
dataMoviesSortRating =  dataMovies[order(dataMovies$rating, decreasing = T), ]

# Sorting by score (numeric) and rating (character), Order matters!
dataMoviesSortScoreRating =  dataMovies[order(dataMovies$score, dataMovies$rating), ]
# OR using the "with" command
dataMoviesSortScoreRatingWith = dataMovies[with(dataMovies, order(score, rating)), ]
```

Quick exercise:

(a) Sort `dataMovies` using variables `score` and `writer`.

(b) Would you have the same results if you sort by: (i) `writer` and `score`, (ii) First by `score` only and then by `writer` only?

2.6 Converting between wide and long form



Wide format: a single row for every data point and multiple columns for the varables or predictors.

Long format: for each data point there are many rows as the number of variables.

The `melt` function in the `reshape2` package converts from wide to long. The `dcast` function does the opposite.

All the variables or predictors should be of the *same data type*

Example: Convert `dataMovies` data to long format and back to wide format.

```
library(reshape2)
# Converting from wide to long format
dataMoviesScoresGross = dataMovies[, colnames(dataMovies) %in%
                                    c("name", "score", "scoreSecond",  "gross")]
dataMoviesScoresGrossLong = melt(data = dataMoviesScoresGross, id.vars = "name")
head(dataMoviesScoresGrossLong,3)

##   name variable value
## 1   40    score    40
## 2   40    score    40
## 3   40    score    40

tail(dataMoviesScoresGrossLong, 3 )

##       name    variable value
## 23002   10 scoreSecond    10
## 23003   10 scoreSecond    10
## 23004   10 scoreSecond    10

# Converting from long to wide format
#deer_wide_again <- reshape2::dcast(deer_long, SkullID ~ variable)
dataMoviesScoresGrossWide = reshape2::dcast(data = dataMoviesScoresGrossLong,
                                            value.var="value", mean,
                                            formula = name ~ variable)
```

2.7 Detecting missing values

Missing data, or missing values, occur when no data value is stored for a variable in an observation. Missing data are coded as `NA` in R.

Example 1:
How many movies have missing data for the directors' sex (`sexDirectorClean`)?
How many movies have complete data for all the variables (columns)?

```
# Counting the number of categories, including NA
table(dataMovies$sexDirectorClean, useNA = "always")

##
##     10     40 Female   Male   <NA>
##   1225    331   2059   3088    965

# Extracting data with complete cases across the entire columns
has_all_measurements = complete.cases(dataMovies)
dataMoviesComplete = dataMovies[has_all_measurements, ]
# dataMovies[has_all_measurements, ] is same as dataMovies[has_all_measurements==TRUE, ]
# dataMoviesComplete = na.omit(dataMovies) # does the same job
```

```
# Extracting data with at least one incomplete case across the entire columns
dataMoviesIncomplete = dataMovies[has_all_measurements == FALSE, ]
```

How can we address missing values in our analysis? A statistical method called Multiple Imputation (MI) can be used. MI is beyond the scope of this course. However, the mice package is a good references to look at if needed.

Example 2: How many movies have missing `score`. Remember that `score` is a numeric vector. Create a data set called `dataMoviesIncompleteScore` having only records with missing scores.

```
# Creating a logical vector for score status (missing or available)
is_score_missing = is.na(dataMovies$score)
table(is_score_missing)

## is_score_missing
## FALSE
##  7668

# Alternative solution
# summary(dataMovies)
# summary(dataMovies£score)

# Creating data with missing scores
dataMoviesIncompleteScore = dataMovies[is_score_missing,]
dataMoviesCompleteScore = dataMovies[!is_score_missing,]
```

Quick exercise: Create a data excluding missing values for `gross` or `sexDirector`. Are there records with missing values for `budget` and `gross`?

```
condition = with(data = dataMovies, expr = (is.na(gross) | is.na(sexDirector)) )
dataMoviesMissingGrosOrSexDirector = dataMovies[!condition,]
```

Quick Exercise: A few cells in the column rating of `dataMovies` are empty. Count the number of empty cells and replace them with `NA`, the rightful notation.