Sample exercise: Deal with uncommon NA strings

Hannes Seller

July 28, 2021

Contents

Introduction

This is a sample exercise that illustrates an exercise from the proposed "Comparing Strategies to Deal with Missing Data in R" course. The exercise belongs to "Chapter 1: Get Familiar with the Data" / "Lesson 2: Inconsistent Missing Value Indicators".

The learner is introduced to the concept that missing values might not always be indicated by NA. Ideally, a data set is checked for inconsistent indicators and outliers before further analyses and data manipulation take place.

A few code passages are faded out and left for the learner to complete. These contain functions like `is.na` that the learner should already know and understand, as well as the central `replace_with_na_all` function. The learner is asked to load the package to better memorize where the function comes from. Finally, the learner has to include the `custom_na_strings` to illustrate they can differentiate it from the `common_na_strings`.

Exercise

Context

There are many reasons why data sets have missing values, and equally many ways to indicate them. While missing values are ideally indicated by NA, other indicators might have been chosen, e.g. during quick quantitative interviews where the surveyor did not manage to indicate missing values consistently. The `naniar` package contains a vector of `common_na_strings`, but data sets might introduce indicators that are less common. The `Boston_missing` data set contains the uncommon indicator "is missing". In this exercise we tidy our data set by replacing all indicators with consistent NA values.

Instructions

- Load the `naniar` package to gain access to the `common_na_strings` vector and the `replace_with_na_all` function.
- Use the `is.na` function on the `Boston_missing` data set. The `sum` function counts all `TRUE` values as 1 and adds them together.
- Add "is missing" to the list of `common_na_strings` and save the result as `custom_na_strings`
- Add the `custom_na_strings` as a condition to the `replace_with_na_all` function. The function checks for every element of the data set (`~.x`) if it contains (`%in%`) any element of the `custom_na_strings` and replaces matching values with NA.

Code with blanks

```
# load the naniar package
_______

# Run the following command first to see how many NAs are in the Boston_missing data set
sum(_______(Boston_missing))

# Print the list of "common_na_strings to check which indicators are commonly used
print(common_na_strings)

# Create a vector custom_na_string by adding "is missing" to the vector of common_na_strings
custom_na_strings <- c(common_na_strings, _______)

# Replace all values that contain one of the indicators with NA
Boston_onlyNA <- _______(Boston_missing, condition = ~.x %in% _______)

# Check again how many NAs are in the dataset Boston_onlyNA
sum(_______(Boston_onlyNA))
```

Preparation and solution

```r
## CUSTOM FUNCTIONS ######

create_missing_values <- function(data, missing_rate = 0.05, seed=42){
  # the function returns the input data set with random values replaced by common NA strings
  # the missing_rate indicates the the ratio of values replaced (default: 0.05 or 5%)
  # a seed can be set for a pseudorandom selection (default: 42)

  # count number of rows and columns of data set
  n_rows <- dim(data)[1]
  n_cols <- dim(data)[2]

  # determine total number of missing values as product of n_rows, n_cols, and missing_rate
  n_missing <- as.integer(n_rows * n_cols * missing_rate)

  # set a seed for the sampling of missing value indices
  set.seed(seed)

  # sample n_missing rows and columns indices which indicate which values will be replaced
  # the algorithm does not check for duplicate indices
  # it is likely that a fewer than n_missing values are replaced
  rows_sample <- sample(1:n_rows, n_missing, replace=T)
  columns_sample <- sample(1:n_cols, n_missing, replace=T)

  # iterate n_missing times through the data set
  # override values with matching indices with a common_na_string
  for(i in 1:n_missing){
    data[rows_sample[i], columns_sample[i]] <- sample(c(naniar::common_na_strings, "is missing"), 1, re
  }

  # return the data set with missing values
  return(data)
}

## HERE BEGINS THE DATA PREPARATION PART ######

# load the Boston data set
Boston <- MASS::Boston

# introduce common_na_strings randomly into the data set
Boston_missing <- create_missing_values(Boston)
```

```r
## HERE BEGINS THE SOLUTION PART ######

# load the naniar package
library(naniar)

# Run the following command first to see how many NAs are in the Boston_missing data set
sum(is.na(Boston_missing))
```

```
## [1] 0
```

```r
# Print the list of "common_na_strings to check which indicators are commonly used
print(common_na_strings)
```

```
##  [1] "NA"     "N A"    "N/A"    "#N/A"   "NA "    " NA"    "N /A"   "N / A"
##  [9] " N / A" "N / A " "na"     "n a"    "n/a"    "na "    " na"    "n /a"
## [17] "n / a"  " a / a" "n / a " "NULL"   "null"   ""       "\\?"    "\\*"
## [25] "\\."
```

```r
# Create a vector custom_na_string by adding "is missing" to the vector of common_na_strings
custom_na_strings <- c(common_na_strings, "is missing")

# Replace all values that contain one of the indicators with NA
Boston_onlyNA <- replace_with_na_all(Boston_missing, condition = ~.x %in% custom_na_strings)

# Check again how many NAs are in the dataset Boston_onlyNA
sum(is.na(Boston_onlyNA))
```

```
## [1] 343
```