cleaning_strings

October 19, 2020

Cleaning String Data in R

1 1. What is Data Cleaning?

- No uniform definition "data cleaning"
- Roughly speaking, refers to exploring the idiosyncrasies of a data set, and then addressing them in a principled manner so as to allow for data analysis

1.1 1.1 Examples of Data Cleaning

- Recoding `"NULL"`, `" "`, `""`, to be `NA`
- Eliminating duplicate entries
- Ensure numeric data is being treated as numerics (e.g., `"2" + 2 != 4`)
- Treating dates or timestamps as `Date` or `POSIXct` data type

2 2. Cleaning Strings

- Parsing/cleaning/extracting info from strings is extremely common
- Parsing timestamp strings is a great example

2.1 2.1 Errors in our `officer_cnt`

```
In [1]: # Load necessary packages and arrests data
        library(stringr)
        library(dplyr)

        arrests_df <- read.csv("./data/pvd_arrests_2020-10-03.csv")
```

```
Attaching package: dplyr


The following objects are masked from package:stats:

    filter, lag
```

```
The following objects are masked from package:base:

    intersect, setdiff, setequal, union
```

```
In [2]: count_names <- function(names_str) {
            # This function should return the number of names in
            # the string `names_str` that we pass to the function.

            name_vec <- unlist(str_split(names_str, ", "))
            k <- length(name_vec)

            return(k)
        }
```

2.1.1 2.1.1 Inconsistencies in arresting_officers Column

```
In [3]: head(arrests_df$arresting_officers, 10)
```

1. ’ YGonzalez, LTaveras’ 2. ’ NManfredi’ 3. ’ MPlace, JPerez, ASantos’ 4. ’ MPlace, JPerez, ASantos’ 5. ’ MPlace, JPerez, ASantos’ 6. ’ MPlace, JPerez, ASantos’ 7. ’ MPlace, JPerez, ASantos’ 8. ’ CVingi, SCooney’ 9. ’ CVingi, SCooney’ 10. ’ CVingi, SCooney’

```
In [4]: tail(arrests_df$arresting_officers, 10)
```

1. ’Lopez, Vincent/ Schneider, Alex/ Vargas, Guillermo’ 2. ’Lopez, Vincent/ Schneider, Alex/ Vargas, Guillermo’ 3. ’Lopez, Vincent/ Schneider, Alex/ Vargas, Guillermo’ 4. ’Lopez, Vincent/ Schneider, Alex/ Vargas, Guillermo’ 5. ’Maycock, Michael’ 6. ’San Lucas, Luis’ 7. ’Lopes, Joseph’ 8. ’Lopes, Joseph’ 9. ’Heaton, Robert’ 10. ’Newton, Frank/ Chin, Rosemarie’

2.2 2.2 Addressing the Inconsistency

- Use different criteria for counting names with full-name format
  - Define function to identify full-name vs. first-initial format
  - Note: first-inital format always starts with two capital letters

```
In [5]: LETTERS                   # This is a built-in object in R
```

1. ’A’ 2. ’B’ 3. ’C’ 4. ’D’ 5. ’E’ 6. ’F’ 7. ’G’ 8. ’H’ 9. ’I’ 10. ’J’ 11. ’K’ 12. ’L’ 13. ’M’ 14. ’N’ 15. ’O’ 16. ’P’ 17. ’Q’ 18. ’R’ 19. ’S’ 20. ’T’ 21. ’U’ 22. ’V’ 23. ’W’ 24. ’X’ 25. ’Y’ 26. ’Z’

```
In [6]: "B" %in% LETTERS
```

TRUE

2.3 2.3 Identifying Full-Name Format

- If the first two characters are uppercase, it's full-name format

```
In [7]: is_uppercase <- function(chr) {
            res <- chr %in% LETTERS
            return(res)
        }

        has_full_names <- function(names_str) {
            char1 <- substr(names_str, 1, 1)
            char2 <- substr(names_str, 2, 2)

            res <- !(is_uppercase(char1) && is_uppercase(char2))
            return(res)
        }
```

2.3.1 2.3.1 Testing our Functions

```
In [8]: is_uppercase("a")                                 # false
        is_uppercase("b")                                 # false
        has_full_names("NManfredi")                       # Not full name
        has_full_names("MPlace, JPerez, ASantos")         # Not full name

        is_uppercase("A")
        is_uppercase("B")
        has_full_names("Newton, Frank")
        has_full_names("Newton, Frank/ Chin, Rosemarie")
```

```
FALSE
FALSE
FALSE
FALSE
TRUE
TRUE
TRUE
TRUE
```

2.4 2.4 Fixing our count_names() Function

```
In [9]: old_count_names <- function(names_str) {
            name_vec <- unlist(str_split(names_str, ", "))
            k <- length(name_vec)

            return(k)
        }
```

```
In [10]: count_names <- function(names_str) {
             names_str_trm <- str_trim(names_str)      # remove whitespace
             if (has_full_names(names_str_trm)) {
                 split_char <- "/ "
             } else {
                 split_char <- ", "
             }

             name_vec <- unlist(str_split(names_str_trm, split_char))
             k <- length(name_vec)

             return(k)
         }
```

2.4.1 2.4.1 Testing New count_names()

```
In [11]: old_count_names("YGonzalez, LTaveras") == 2
         old_count_names("Newton, Frank/ Chin, Rosemarie") == 2     # function is wrong
         count_names("YGonzalez, LTaveras") == 2
         count_names("Newton, Frank/ Chin, Rosemarie") == 2
```

```
TRUE
FALSE
TRUE
TRUE
```

2.5 2.5 Re-Counting Officers

- Let's compare how the "old" (i.e., incorrect) method did relative to our new count_names()

```
In [12]: count_officers <- function(col, old = FALSE) {

             n <- length(col)   # get the length of our input column
             cnts <- rep(0, n)  # allocate vector of zeros to populate with counts

             for (i in 1:n) {
                 if (old) {
                     cnts[i] <- old_count_names(col[i])
                 } else {
                     cnts[i] <- count_names(col[i])
                 }
             }
             return(cnts)
         }
```

```
In [13]: arrests_df$old_officer_cnt <- count_officers(arrests_df$arresting_officers, old = TRUE

         arrests_df$officer_cnt <- count_officers(arrests_df$arresting_officers)
```

```
In [14]: head(arrests_df)
```

A data.frame: 6 Œ 20

	arrest_date	year	month	gender	race	ethnicity	year_of
	<chr>	<int>	<int>	<chr>	<chr>	<chr>	<int>
1	2019-08-24T02:23:00.0	2019	8	Male	White	NonHispanic	1981
2	2019-08-24T02:02:00.0	2019	8				1994
3	2019-08-24T02:02:00.0	2019	8	Female	Black	NonHispanic	1984
4	2019-08-24T02:02:00.0	2019	8	Female	Black	NonHispanic	1984
5	2019-08-24T02:02:00.0	2019	8	Female	Black	Unknown	2001
6	2019-08-24T02:02:00.0	2019	8	Female	Black	Unknown	2001

```
In [15]: tail(arrests_df, 12)
```

A data.frame: 12 Œ 20

	arrest_date	year	month	gender	race	ethnicity	ye
	<chr>	<int>	<int>	<chr>	<chr>	<chr>	<i
8744	2020-09-25T15:14:00.0	2020	9	Male	White	NonHispanic	194
8745	2020-09-25T14:36:00.0	2020	9	Male	White	Hispanic	199
8746	2020-09-25T14:36:00.0	2020	9	Male	White	Hispanic	199
8747	2020-09-25T14:36:00.0	2020	9	Male	White	Hispanic	199
8748	2020-09-25T14:36:00.0	2020	9	Male	White	Hispanic	199
8749	2020-09-25T14:36:00.0	2020	9	Male	White	Hispanic	199
8750	2020-09-25T09:45:00.0	2020	9	Male	White	NonHispanic	197
8751	2020-09-25T09:11:00.0	2020	9	Male	Black	NonHispanic	199
8752	2020-09-25T00:00:00.0	2020	9	Female	Black	Hispanic	198
8753	2020-09-25T00:00:00.0	2020	9	Male	Black	Hispanic	198
8754	2020-09-12T20:03:00.0	2020	9	Male	NULL	NULL	199
8755	2020-08-27T07:10:00.0	2020	8	Male	White	NonHispanic	196

2.6 2.6 How Many Errors?

```
In [16]: sum(arrests_df$old_officer_cnt != arrests_df$officer_cnt)
```

```
4197
```

```
In [17]: nrow(arrests_df)
```

```
8755
```