DA5020 Homework 4: Strings and Factors

2019-10-01

```
# Clear plots
#if(!is.null(dev.list())) dev.off()
# Clear console
#cat("\014")
# Clean workspace
#rm(list=ls())
```

Github

https://github.com/ajb7/R-workbooks/tree/master/workbook4 (https://github.com/ajb7/R-workbooks/tree/master/workbook4)

Preparation

Download US Farmers Markert Directory (https://www.ams.usda.gov/local-food-directories/farmersmarkets) data from the website of USDA (click on “Export to Excel”). Rename the file as *farmers_market.csv*.

Download the Know Your Farmer, Know Your Food Projects (https://catalog.data.gov/dataset/know-your-farmer-know-your-food-projects) dataset and name it as *kyfprojects.xls*. Put it into the same folder.

Read the data:

```
# Installing dplyr library and importing dataset
# using read.csv to read csv data and read_excel to read data from xls file
#install.packages("dplyr")
library("dplyr")
library("readxl")
setwd("D:/UNIVERSITY/Assignments/DA 5020/Assegments/4/")
usda <- read.csv("_farmers_market.csv_.csv", header = T, sep = ",")
kyf <- read_excel("_kyfprojects.xls_.xls")
```

Warm Up

This dataset stores city and state in different columns, what if you want to print out city and state in the format “City, State”?

Answer: One way of achieving this is, we concatenate “city”, “,” and “State” columns from our dataset

into a new variable. We can use “paste” command or “str_c” which concatenates multiple strings.

```
library("stringr")
state_city <- str_c(usda$city, ', ',  usda$State)
head(state_city, 5)
```

```
## [1] "Danville, Vermont"        "Parma , Ohio"
## [3] "Six Mile, South Carolina" "Lamar , Missouri"
## [5] "New York, New York"
```

Questions

Q1. (20 points) Cleanup the `Facebook` and `Twitter` column to let them contain only the facebook username or twitter handle name. I.e., replace “https://www.facebook.com/pages/Cameron-Park-Farmers-Market/97634216535?ref=hl (https://www.facebook.com/pages/Cameron-Park-Farmers-Market/97634216535?ref=hl)” with “Cameron-Park-Farmers-Market”, “https://twitter.com/FarmMarket125th (https://twitter.com/FarmMarket125th)” with “FarmMarket125th”, and “@21acres” with “21acres”.

Steps:

1. First we convert all value which are blank in Twitter and Facebook column into NA
2. We create 2 functions to parse twitter and facebook values
3. “parse_twitter” function:
    a. Checks if value is not NA
    b. Because in Twitter, account names usually start after “@” or after “.com/”, our regular expression looks for any character or number that start after “@” or after “.com/”
    c. We get all such values where regular expression matches, split the value by “@” and “.com/”
    d. Finally return the split value, which is the twitter username.
4. “parse_fb” function:
    a. For all values that are not NA, function matches regular expression with the values
    b. Regular expression looks for any character or number after “.com/” or “.com/pages” in each row of “Facebook” column
    c. For all such matches, function then splits these values by “.com/” and “.com/pages” and finally returns the username or pagename
5. Finally, we call both of these functions for each row in the dataset, by passing “twitter” and “facebook” column value for each row

```
#Make Twitter, Facebook values NA, where it is blank
usda$Twitter[usda$Twitter == ""] <- NA
usda$Facebook[usda$Facebook == ""] <- NA

#Function to parse "twitter" value
#Regex looks for pattern with "@" or ".com/"
parse_twitter <- function(twitter){
  tid <- ''
  if (!is.na(twitter)){
    re <- '@[a-zA-Z0-9_]([^ ]+)|.com/[a-zA-Z0-9_]([^ ]+)'
    tword <- str_extract(twitter, re)
    tsplit <- str_split(tword, "@")
    if(!is.na(tsplit[[1]][1])){
      if (length(tsplit[[1]]) > 1){
        tid <- sapply(tsplit, "[[", 2)
      }else{
        tsplit2 <- str_split(tsplit, ".com/")
        tword <- sapply(tsplit2, "[[", 2)
        tsplit_reg <- str_split(tword, "\\?")
        tid <- sapply(tsplit_reg, "[[", 1)
      }

    }else{
      tid <- twitter
    }

  }

  return(tid)

}

#Function to parse "facebook" value
#Regex looks for pattern with ".com/" or ".com/pages/""
parse_fb <- function(fb){
  tid <- ""
  if (!is.na(fb)){
    re <- '.com/[a-zA-Z0-9_]([^ ]+)|.com/pages/[a-zA-Z0-9_]([^ ]+)'
    tword <- str_extract(fb, re)
    tsplit <- str_split(fb, ".com/|.com/pages")

    if(!is.na(tsplit[[1]][1])){
      if (length(tsplit[[1]]) > 1){
        tid <- sapply(tsplit, "[[", 2)
      }else{
        tid <- fb
      }
```

```
    }

  }

  return(tid)

}

#For each row in usda dataset, call parse_twitter and parse_fb, with respective column
s as argument
for (row in 1:nrow(usda)) {
  usda[row, 'twitter_uname'] <- parse_twitter(usda[row, 'Twitter'])
}

for (row in 1:nrow(usda)) {
  usda[row, 'fb_uname'] <- parse_fb(usda[row, 'Facebook'])
}
```

Q2. (20 points) Clean up the `city` and `street` column. Remove state and county names from the `city` column and consolidate address spellings to be more consistent (e.g. “St.”, “ST.”, “Street” all become “St”; “and” changes to “&”, etc…).

Answer: We perform following clean up on City and Street columns:

1. We replace blank values with NA
2. We check if value in in Camel Case, ex: New York etc.
3. Remove all words after ‘comma’, so that City column only has city names
4. Replace “and” with “&” between words
5. Replace “St.”, “st.”, “ST.”, “street”, “street” with “St” for consistency
6. Replace Rd. Rd rd road with Rd
7. Replace Av. Ave. Avenue avenue with Ave

```
# City and Street:
usda$city[usda$city == ""] <- NA #If value is blank, then replace with NA
usda$street[usda$street == ""] <- NA #If value is blank, then replace with NA

clean_str <- function(str_cl){
  str_cl <- ""
  if(!is.na(str_cl)){
    str_cl <- as.character(str_cl)
    str_cl <- str_trim(str_cl)  #Remove whitespace trailing and leading
    str_cl <- str_to_title(str_cl) #Convert case to camel case
    str_cl <- gsub("(.*),.*", "\\1", str_cl) #remove any word after comma
    str_cl <- gsub(" and ", " & ", str_cl)   #replace " and " between words with " & "
    str_cl <- gsub("St\\. |ST\\. |street |st\\. |Street ", "St", str_cl)  #replace st.
ST, Street with Street
    str_cl <- gsub("Av\\. |Ave\\. |Av |street |Avenue |avenue ", "Ave", str_cl)  #repl
ace Av. Ave. Avenue avenue with Ave
    str_cl <- gsub("Rd\\. |Rd |rd |road", "Rd", str_cl)  #replace Rd. Rd rd road with
Rd
  }
  return(str_cl)
}

for (row in 1:nrow(usda)) {
  usda[row, 'city'] <- clean_str(usda[row, 'city'])
}

for (row in 1:nrow(usda)) {
  usda[row, 'street'] <- clean_str(usda[row, 'street'])
}
```

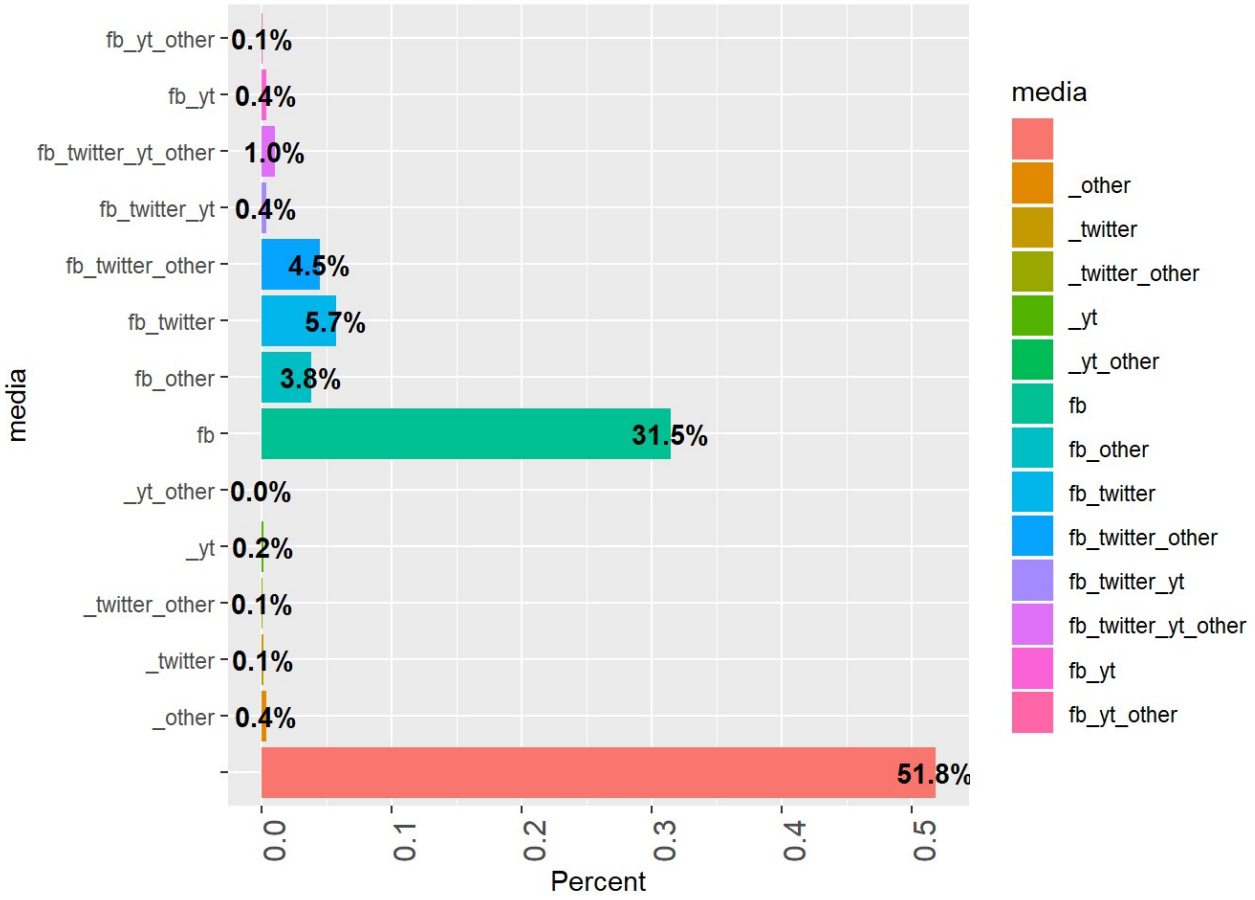
Q3. (20 points) Create a new data frame (tibble) that explains the online presence of each state's farmers market. I.e., how many percentages of them have a facebook account? A twitter account? Or either of the accounts? (Hint: use the `is.na()` function)

Steps:

1. First we set all the values in Twitter, Facebook, Youtube and Other Media column as "NA" if its blank
2. We create a new dataframe 'social_df' which has 1 columnn 'media' for each row in 'usda' dataset
3. 'Media' column will have value:
    a. "" ie; blank, if it is NA
    b. "fb" if facebook column in usda has value
    c. "_twitter" if twitter column in usda has value
    d. "_yt" if youtube column in usda has value
    e. "_other" if OtherMedia column in usda has value
    f. A combination of above values, if farmer has multiple social media

4. We create a function which:
    a. checks if Facebook, Twitter, Youtube or OtherMedia value is NA for a given row
    b. Appends ‘media’ column based on ‘Facebook’, ‘Twitter’, ‘Youtube’, ‘OtherMedia’ values for each row
5. We plot graph to show percent values in social_df for each media type.

```
#install.packages("ggplot2")
library("ggplot2")

usda$Youtube[usda$Youtube == ""] <- NA
usda$OtherMedia[usda$OtherMedia == ""] <- NA

get_social <- function(usda){
  media <- ""
  if(!is.na(usda['Facebook'])){
    media <- str_c(media, "fb")
  }
  if(!is.na(usda['Twitter'])){
    media <- str_c(media, "_twitter")
  }
  if(!is.na(usda['Youtube'])){
    media <- str_c(media, "_yt")
  }
  if(!is.na(usda['OtherMedia'])){
    media <- str_c(media, "_other")
  }

  return(media)
}

social_df <- usda
social_df$media <- apply(usda, 1, get_social)

ggplot(social_df, aes(x= media)) +
    geom_bar(aes(y = (..count..)/sum(..count..), fill = media), stat="count") +
    geom_text(aes( label = scales::percent((..count..)/sum(..count..)),
                   y= (..count..)/sum(..count..), fontface = "bold"), stat= "count") +
    coord_flip() +
    labs(y = "Percent", fill="media") +
    theme(plot.title = element_text(hjust = 0.5), axis.text.x=element_text(angle=90, h
just=1, size=12))
```



We observe that 51% of farmers have no media presence, while 31% have facebook only, while 42% have facebook atleast. Around 5% have both facebook and twitter and seem to be highly active.

Q4. (20 points)

```
Some of the farmer market names are quite long. Can you make them shorter by using the
`forcats::fct_recode` function? Create a plot that demonstrates the number of farmers
markets per location type. The locations should be ordered in descending order where t
he top of the graph will have the one with the highest number of markets.
```

Answer. We see that there are 96 rows where Market name has words more than 60. We replace such Market names with shorter version, i.e; replacing them with first 60 words only.

To make ‘market name’ shorter, we use substr function to grab the first 60 words of ‘market names’ which are greater than 60 words

```
length(usda$MarketName[str_count(usda$MarketName)>60])
```

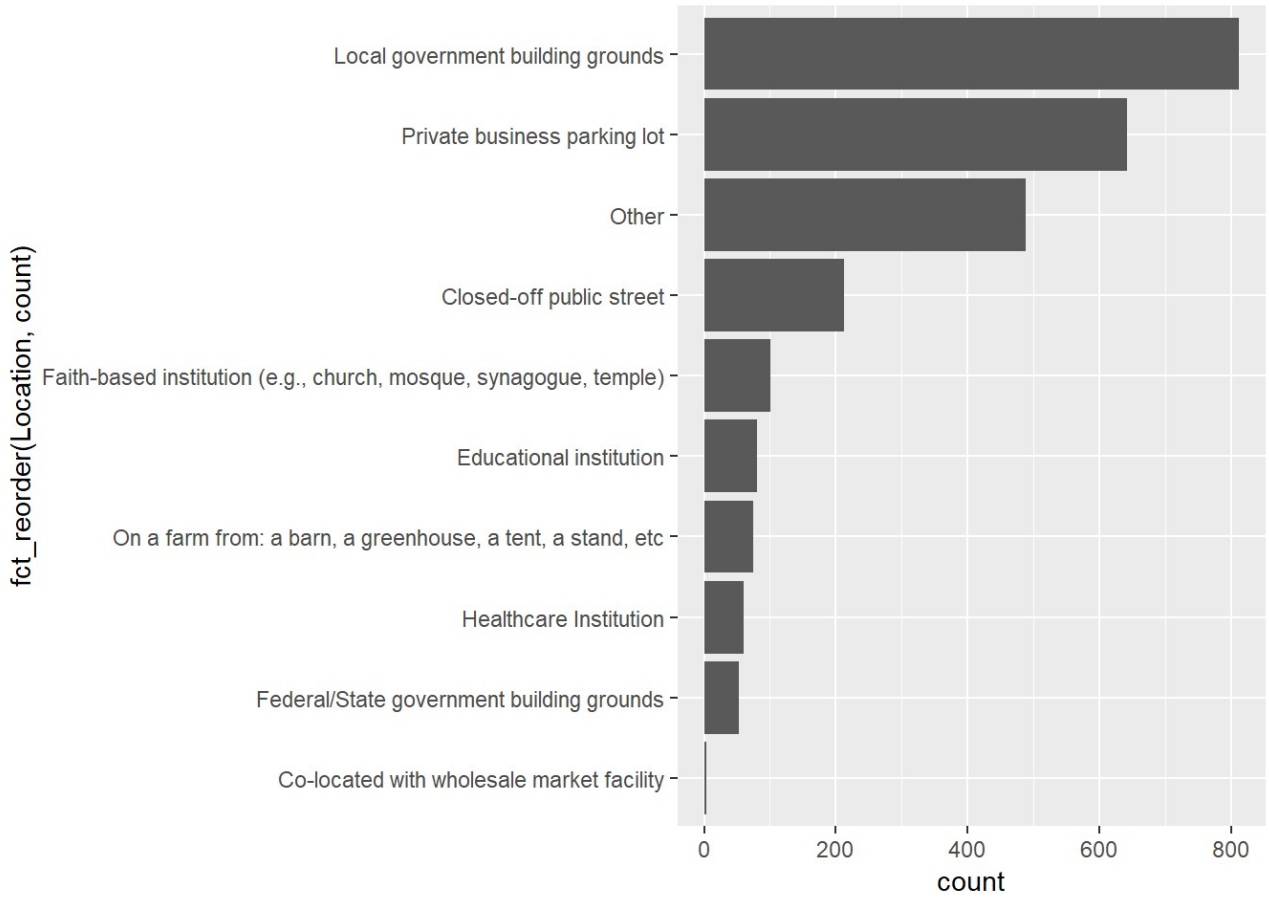
```
## [1] 96
```

```
usda$MarketName[str_count(usda$MarketName)>60] <- substr(usda$MarketName[str_count(usd
a$MarketName)>60], 1, 59)
```

```
## Warning in `[<-.factor`(`*tmp*`, str_count(usda$MarketName) > 60, value =
## structure(c(1L, : invalid factor level, NA generated
```

Next, we want to show number of farmer's markets per location type. To do that, we: 1. filter for all the rows, for which 'Location' column contains some value 2. next, we group all the rows by location type 3. We count the number of farmer markets in each location type group 4. we arrange them in decreasing order of count 5. Finally, we plot the above collected data in graph. We use 'fct_reorder' to reorder the factor level of 'Location' column by 'count' computed in step 4. We flip the cordinates to show the results.

```
library("forcats")
market_per_loc <- usda %>% filter(Location != "") %>% group_by(Location) %>% summarize
(count=n()) %>% arrange(desc(count))


#ggplot(data = market_per_loc, aes(x=fct_reorder(Location, count), y=count)) + geom_po
int(stat='identity') + coord_flip()
ggplot(data = market_per_loc, aes(x=fct_reorder(Location, count), y=count)) + geom_bar
(stat='identity') + coord_flip()
```



We observe that 'Local government building grounds' has most number of farmer markets followed by private business parking lot.

Q5. (20 points) Write code to sanity check the `kyfprojects` data. For example, does `Program Abbreviation` always match `Program Name` for all the rows? (Try thinking of your own rules, too.)

Answer: We perform following checks for sanity on this dataset:

1. Check if ZIP codes are less than 5 digits: Usually, ZIP codes in the US are 5 digits, however, few rows in the dataset have more than 5 digit in their ZIPs, which could be manual error while entry. We trim such ZIP and keep only first 5 digits
2. Check if Year field is not less than zero: As Year is suppose to be a positive integer always, we check in our dataset if there exists any row with less than zero or blank year. We observe that there is only 1 row with Year value is NA.
3. Check if Funding Amount is less than zero: Funding amount is another value that cannot be less than zero, hence we check rows which have any less than zero value for that column. No such data exists in our dataset.
4. Check if abbreviations are proper: “Program name” and “Program abbreviations” must be idetentifiable, hence we get all the rows, where “program abbreviation” value does not match with the expected abbreviation of “Program name”. We convert and replace the “Program abbreviation” value with the abbreviation of “Program name”, for all such rows.

```
#ZIp cannot be bigger than 5 digits
length(kyf$Zip[str_count(kyf$Zip)>5 & !is.na(kyf$Zip)])
```

```
## [1] 11
```

```
kyf$Zip[!is.na(kyf$Zip) & str_count(kyf$Zip)>5] <- substr(kyf$Zip[!is.na(kyf$Zip) & st
r_count(kyf$Zip)>5], 0, 5)
#check year
print(str_c("Number of rows with NA or negative year: ", length(kyf$Year[is.na(kyf$Yea
r) | kyf$Year == "" | kyf$Year < 0])))
```

```
## [1] "Number of rows with NA or negative year: 0"
```

```
#State cannot be blank
print(str_c("Number of rows with NA or blank state: ", length(kyf$State[is.na(kyf$Stat
e) | kyf$State == ""])))
```

```
## [1] "Number of rows with NA or blank state: 0"
```

```
#funding amount cannot be negative
print(str_c("Number of rows with negative funding amount: ", length(kyf$`Funding Amoun
t ($)`[is.na(kyf$`Funding Amount ($)`) | kyf$`Funding Amount ($)` < 0])))
```

```
## [1] "Number of rows with negative funding amount: 1"
```

```
#Check program abbreviations
print(str_c("Number of rows with non-matching abbreviations: ", length(kyf$`Program Na
me`[kyf$`Program Abbreviation` != abbreviate(kyf$`Program Name`)])))
```

```
## [1] "Number of rows with non-matching abbreviations: 491"
```

```
kyf$`Program Abbreviation`[kyf$`Program Abbreviation` != abbreviate(kyf$`Program Name
`)] <- abbreviate(kyf$`Program Name`[kyf$`Program Abbreviation` != abbreviate(kyf$`Pro
gram Name`)])
```

We converted 491 rows in the dataset into proper abbreviations. Moreover,11 ZIP codes had digits more than 5, we converted them to 5 digit codes.