# Homework

*Dylan Rose*

## Introduction and warmup

Setup packages and load in the farmers_market csv file of data.

```
options(width = 30)
library(tidyverse)
library(stringr)
library(readxl)
library(magrittr)

# Check to see if the file's available and if so load it, otherwise print an
# error string
if (!exists("./farmers_market.csv")) {
    farmers_market_data <- read_csv("farmers_market.csv")
} else {
    sprintf("Data's not where it should be!")
}
```

## Warmup

To get the city and state formatted and printed into a single column where the elements are separated by a comma, you can do this:

```
city_state_single_column <- str_c(farmers_market_data$city,
    farmers_market_data$State,
    sep = ",")
head(city_state_single_column)
```

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

```
# Just check that the number of
# rows is the same as the
# original tibble so you could
# it add it back as a new
# column if needed
if (length(city_state_single_column) ==
    nrow(farmers_market_data)) {
    sprintf("The number of rows is the same, so we can proceed safely.")
} else {
    sprintf("Something's gone wront!")
}
```

```
## [1] "The number of rows is the same, so we can proceed safely."
```

## Question 1: Cleanup the Facebook and Twitter column to let them contain only the facebook username or twitter handle.

Because the url format for these columns should be specified by the company, we can begin by checking for a few apparent regularities in each of the variables.

Based on the example provided in the homework, it looks like Facebook related links have a substring "/pages/" before the chunk we want. Let's see if it's there for all of them:

```
sum(str_detect(farmers_market_data$Facebook,
    "/pages/"), na.rm = TRUE)
```

```
## [1] 716
```

That we even have to do an na.rm=TRUE means that this data is pretty sparse and messy: it looks like many of the markets don't have a Facebook page listed at all. In looking at the raw data for those that do, it seems like there are three main patterns: markets that just have a name, markets that have a "flat" facebook address (no pages subdirectory in the link), and those that are listed as pages.

We can start straightforwardly by getting indices for those markets that don't have anything listed or those that are just the name of the market for some reason:

```
# Get everything lowercase to
# start
facebook_data <- str_trim(str_to_lower(farmers_market_data$Facebook))

# strip everything up to .com
# or .com/pages/
just_facebook_name_data <- str_replace_all(facebook_data,
    "(https://)|(www.)|(http://)",
    "")
# Get rid of page specific
# preambles, weird replacements
# for 'pages',etc
just_facebook_name_data <- str_replace_all(just_facebook_name_data,
    "(facebook.com/)|(facebook.com/pages/)(facebook.com/#!)|(pages/)|(#!/)",
    "")
# Drop everything that isn't a
# letter
just_facebook_name_data <- str_replace_all(just_facebook_name_data,
    "[^[:alpha:]]", "")
```

Now for the twitter data:

```
twitter_data <- str_trim(str_to_lower(farmers_market_data$Twitter))

# In this case just drop
# everything up to '.com/'
just_twitter_name_data <- str_replace_all(twitter_data,
    "(http://(.*?)twitter.com/)|(https://(.*?)twitter.com/)|(twitter.com/)",
    "")

# Drop out a few types of
# remaining oddballs: 'n/a', a
# few places where 'http/s'
# strings are still left
just_twitter_name_data <- str_replace_all(just_twitter_name_data,
```

```
    "(https//)|(n/a)|(/)|(www)",
    "")

# Last, do anything not
# alphanumeric
just_twitter_name_data <- str_replace_all(just_twitter_name_data,
    "[^[:alnum:]]", "")
```

## Question 2: clean up the city and street column. Remove state and county names from the city column and consolidate addres spellings to be more consistent.

```
# Start with the city data
city_data <- str_trim(str_to_title(farmers_market_data$city))

# Drop anything with a comma
# followed by strings, which
# seems to be how most of the
# cities included with
# statename errors end up
# happening in this data
city_data <- str_replace_all(city_data,
    "\\,.*?$", "")
```

Now for the street data, which is a lot more laborious:

```
# Load it and fix the
# capitalization
street_data <- str_to_title(farmers_market_data$street)

# First clean up street
# references
street_data <- str_replace_all(street_data,
    "([Ss]treet)|(St[\\.r])", "St.")
# Now blvd, avenue, etc
street_data <- str_replace_all(street_data,
    "([Aa]ve+.*?$)|([Aa]venue+.*?$)",
    "Ave.")
street_data <- str_replace_all(street_data,
    "([bB]lvd[\\..*?]+.*?$)|([bB]oulevard+.*?$)",
    "Blvd.")
street_data <- str_replace_all(street_data,
    "([pP]ark+.*?$)", "Prk.")
street_data <- str_replace_all(street_data,
    "([rR]oad+.*?$)|([rR]d+.*?$)",
    "Rd.")
street_data <- str_replace_all(street_data,
    "^+[Aa]nd+$", "\\&")
```

## Question 3: create a new data frame (tibble) that explains the online presence of each state's farmer's markets. I.e. what percentage have a facebook account? A twitter? Or either of the accounts?

We can do this in the following way:

```
# Set this variable up as a
# factor
farmers_market_data$State <- as.factor(farmers_market_data$State)
# Then group and compute the
# percentages
group_by(farmers_market_data, State) %>%
    summarise(percent_twitter = sum(!is.na(Twitter))/n(),
        percent_facebook = sum(!is.na(Facebook))/n(),
        percent_either = sum(!is.na(Facebook),
            !is.na(Twitter))/sum(n(),
            n()))
```

```
## # A tibble: 53 x 4
##                    State
##                   <fctr>
##  1              Alabama
##  2               Alaska
##  3              Arizona
##  4             Arkansas
##  5           California
##  6             Colorado
##  7          Connecticut
##  8             Delaware
##  9 District of Columbia
## 10              Florida
## # ... with 43 more rows, and
## #   3 more variables:
## #   percent_twitter <dbl>,
## #   percent_facebook <dbl>,
## #   percent_either <dbl>
```

## Question 4: 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 the top of the graph will have the one with the highest number of markets.

forcats::fct_recode seems to really be focused as a "manual" factor level editor, so to make this work we'll first have to find a few (say, no more than ten for the sake of convenience) level names that are too long:

```
farmers_market_data$MarketName <- factor(farmers_market_data$MarketName)
farmers_market_levels <- levels(factor(farmers_market_data$MarketName))
farmers_market_levels_ordered <- farmers_market_levels[order(nchar(farmers_market_levels),
```

```
    farmers_market_levels, decreasing = TRUE)]
farmers_market_levels_ordered[1:10]
```

```
##  [1] "Torrance Certified Farmers' Market -Tuesday and saturday- 2200 Crenshaw Blvd Torrance,Ca.90501
##  [2] "North San Diego (Sikes Adobe) Certified Farmers Market, at the Sikes Adobe Historic Farmstead"
##  [3] "Faith in Place Winter Farmers Market - Our Saviour's Lutheran Church, Arlington Heights"
##  [4] "Faith in Place Winter Farmers Market - Euclid Avenue United Methodist Church, Oak Park"
##  [5] "Faith in Place Winter Farmers Market - Augustana Lutheran Church of Hyde Park, Chicago"
##  [6] "Berkshire Harmony Downtown Pittsfield Farmers Markets, Music, and Creative Arts Fair"
##  [7] "Running Springs Area Chamber of Commerce Certified Farmers Market and Artisan Faire"
##  [8] "Faith in Place Winter Farmers Market - Greenstone United Methodist Church - Pullman"
##  [9] "Harvest for Health Farmers' Market at NewYork-Presbyterian/Hudson Valley Hospital"
## [10] "Winlock Saturday Market - Featuring Local Farmers & Artisans from our community."
```

With these top ten offenders in hand, we can straightforwardly change their levels using the required function:

```
# create a list of new
# replacement
farmers_market_data$MarketName <- forcats::fct_recode(farmers_market_data$MarketName,
    TorrMarket = farmers_market_levels_ordered[1],
    NorSand = farmers_market_levels_ordered[2],
    FaithPlaceOur = farmers_market_levels_ordered[3],
    FaithPlaceEuc = farmers_market_levels_ordered[4],
    FaithPlaceAug = farmers_market_levels_ordered[5],
    BerkHarm = farmers_market_levels_ordered[6],
    RunSpring = farmers_market_levels_ordered[7],
    FaithPlaceGrn = farmers_market_levels_ordered[8],
    HarvForHel = farmers_market_levels_ordered[9],
    Winlock = farmers_market_levels_ordered[10])
```

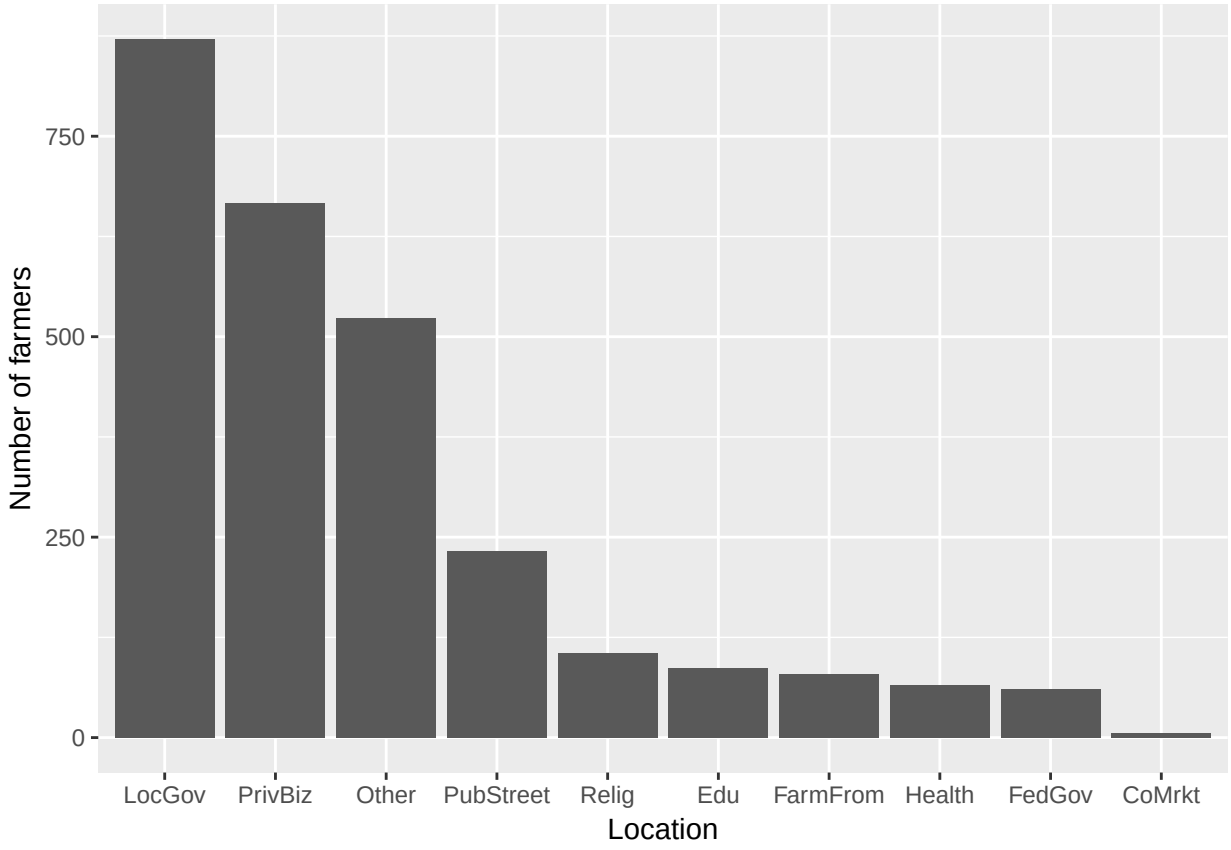
Now set up the data for the plotting question:

```
farmers_by_location_type <- group_by(farmers_market_data,
    Location) %>% summarise(number_of_farmers = n())
# Drop rows that have NAs
# (don't list a location type),
# as that swamps the rest of
# the data visually
farmers_by_location_type <- arrange(farmers_by_location_type[complete.cases(farmers_by_location_type),
    ], desc(number_of_farmers))
```

I wasn't sure whether these "Locations" were the names that we're supposed recode into something tidier than what's there originally, but we should probably do that anyway (and then show the plot):

```
farmers_by_location_type$Location <- factor(farmers_by_location_type$Location)
farmers_by_location_type$Location <- forcats::fct_recode(farmers_by_location_type$Location,
    LocGov = "Local government building grounds",
    PrivBiz = "Private business parking lot",
    PubStreet = "Closed-off public street",
    Relig = "Faith-based institution (e.g., church, mosque, synagogue, temple)",
    Edu = "Educational institution",
    FarmFrom = "On a farm from: a barn, a greenhouse, a tent, a stand, etc",
    Health = "Healthcare Institution",
    FedGov = "Federal/State government building grounds",
    CoMrkt = "Co-located with wholesale market facility")
ggplot(farmers_by_location_type,
```

```
    aes(x = reorder(Location, -number_of_farmers),
        y = number_of_farmers)) +
    geom_bar(stat = "identity") +
    xlab("Location") + ylab("Number of farmers")
```




## Question 5: 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.)

Note that rather than the standard "xlsx" library, I was forced to use a tidyverse specific alternative "readxl", as there's apparently some issue with MacOS and the "rJava" package that is a dependency of "xlsx" that is all kinds of garment-rending and headache inducing trouble to solve.

```
# Check to see if the file's
# available and if so load it,
# otherwise print an error
# string
if (!exists("./kyfprojects.xls")) {
    kyfprojects <- read_excel("kyfprojects.xls")
} else {
    sprintf("Data's not where it should be!")
}
```

First thing: change the column names to something more useable:

```
replacement_names <- str_replace_all(str_to_lower(colnames(kyfprojects)),
    "\\s", "_")
colnames(kyfprojects) <- replacement_names
```

Just inspecting the "head" of the data frame shows us that while they've been coerced to strings, the project titles often contain a badly escaped/specified set of characters. Let's get them down to just letters:

```
kyfprojects$project_title <- kyfprojects$project_title %>%
    str_replace_all(., "[^[:alnum:]]$",
        "")
```

We can also check that the program name and its abbreviation line up

```
program_name <- kyfprojects$program_name
program_abbreviation <- kyfprojects$program_abbreviation

# Strip the program name of
# non-alphabet characters
no_non_alpha_name <- str_replace_all(program_name,
    "[^[:alnum:]]", "")
# Now get just the capital
# letters mushed together
no_non_only_caps_name <- str_extract_all(no_non_alpha_name,
    "[A-Z]")
# we'll use a for loop for this
output_names <- c()
for (each_element in 1:length(no_non_only_caps_name)) {
    this_abbreviation <- paste(no_non_only_caps_name[each_element][[1]],
        collapse = "")
    output_names <- c(output_names,
        this_abbreviation)

}

# Now check the number of
# matches between the two
sum(str_count(program_abbreviation,
    output_names)) == length(program_abbreviation)
```

```
## [1] FALSE
```

As we get a "FALSE" return value here, it looks like the program abbreviation doesn't always smoothly or directly match up to the program name. For example, one program is named "Business & Industry Loan Guaranteed", but it's program abbreviation is "B and I". You could get around the mismatch between spelling out "and" and using a "&" character with regex, I think, but the mapping between the name and the abbreviation still seems to drop some info (nothing included in the abbreviation about "Loan Guarantee", for example).

Finally, we can double check to make sure that there weren't any projects that were double booked for a particular program in a particular year, for a particular state

```
# Gotta set these three as
# factors first
kyfprojects$year <- factor(kyfprojects$year)
kyfprojects$program_name <- factor(kyfprojects$program_name)
kyfprojects$project_title <- factor(kyfprojects$project_title)
```

```r
# Now group by year,program
# name, and state
number_of_distinct_programs <- group_by(kyfprojects,
    year, program_name, state) %>%
    summarise(number_of_programs = n(),
        distinct_programs = n_distinct(project_title)) %>%
    mutate(total_equals_distinct = number_of_programs ==
        distinct_programs, size_difference = number_of_programs -
        distinct_programs)

# Grab just the ones where the
# logical comparison column is
# false
multiple_entries_per_year_state_program <- subset(number_of_distinct_programs,
    total_equals_distinct == FALSE)
print.data.frame(head(multiple_entries_per_year_state_program,
    n = 5))
```

```
##   year
## 1 2009
## 2 2009
## 3 2009
## 4 2009
## 5 2009
##                          program_name
## 1 Business & Industry Loan Guarantee
## 2 Business & Industry Loan Guarantee
## 3    Rural Business Enterprise Grant
## 4         Specialty Crop Block Grant
## 5         Specialty Crop Block Grant
##   state number_of_programs
## 1    ND                 2
## 2    NY                 2
## 3    CA                 3
## 4    IL                22
## 5    MS                 3
##   distinct_programs
## 1                 1
## 2                 1
## 3                 2
## 4                21
## 5                 2
##   total_equals_distinct
## 1                 FALSE
## 2                 FALSE
## 3                 FALSE
## 4                 FALSE
## 5                 FALSE
##   size_difference
## 1               1
## 2               1
## 3               1
## 4               1
## 5               1
```

So it looks like even with a pretty refined set of grouping conditions, there still appear to be duplicate project titles in the data when there probably shouldn't be. It looks like by and large, the differences are small: on average 1.51 (sd 0.8) duplicates per year/program/state. This means one of two things, either of which would probably have to be checked further in the data to be parsible:

1. Certain project or funding types are carried over between years and are re-labeled as such in the way that my sort hasn't picked up on. We could check this by looking at whether the funding types differ for cases that overlap. It's also possible that some of these projects have the same names but are treated as distinct because they happen in different towns.
2. Because the errors are small and irregular, they're just double bookings of the kind that can easily happen when working manually with spreadsheet data, and so should probably get cleaned up out of the data.