Campaign Analysis in R

Powell Sheagren

2022-09-22

R For Campaign Data Analysis

Introduction & Background

Overview: When I first started volunteering for Bluebonnet, a lot of the tasks I was doing for campaigns were run of the mill targeting and IDing for which I used my most fluent language at the time, R. Compared to Python, R is a language designed for statistical analysis that can be used more generally and, although it isn't as popular and generally applicable I wanted to at least introduce it for those who were familiar with programming with other languages as it has some convenient features specific to it. I'll be explaining some of the functionality and structure of R as I walk through these campaign projects.

Intended Audience: Python programmers who would like to see how things translate over in other languages

Key Learning Objectives:

- R Syntax and stock packages
- Simple donor ID process
- Parsing and summarizing voter data
- Visualizations in ggplot

Importing Packages

R Overview

R is a open source statistical programming language which has been around since the 1990s and has had a continuous community since then. Its primary file types are .r and .rmd (or R markdown) the former being an executable file and the latter being a markdown file with code chunks to work in. There are other more modern versions of the Markdown format like Quarto which can be converted back and forth with .ipynb files. These files are most often worked in the RStudio which is a text editor specifically designed for the language and has built in help pages and package support.

The file I'm referencing with be a .rmd included in a github attached here. Most files start off with a markdown heading and formatting before code blocks are added. These markdown documents are helpful since, like python notebook files, they can be rendered into html or a pdf when needed. I traditionally add my preferred packages in the first code block as can be seen below

The description of each of the packages are included and I will reference them as they are used. Next I wanted to show off some of the generic syntax differences between R and Python. They are not wildly different as they use the same concepts but there are enough to get you confused when going back and forth between the two.

```
# Assigning variables with <- (= does still work)
var <- 16
char <- "hello world"
```

```r
# Printing is pretty much the same
print(var)
```

```
## [1] 16
```

```r
print(char)
```

```
## [1] "hello world"
```

```r
# instead of indents and spacing R uses brackets and parenthesis
while (var < 20){
  var <- var + 1 # It is still indendent but it does not have to be
}

# Same with If/else statements although elif is replaced with else if
if (char == "goodbye"){
  print(char)
} else if(char == "hello world"){
  print(char)
} else{}
```

```
## [1] "hello world"
```

```r
# lastly its the same for for loops but you don't need range()
for (i in 1:10){
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

```r
## You can still do classic time checking
start_time <- Sys.time()
end_time <- Sys.time()
print(end_time - start_time)
```

```
## Time difference of 0.0004379749 secs
```

```r
## Lists are also slightly different using a c() instead of []
practice_list <- c(1,2,3,4,"junk",TRUE)
print(practice_list)
```

```
## [1] "1"    "2"    "3"    "4"    "junk" "TRUE"
```

```r
## IMPORTANT: Indexing in R starts at 1 so
practice_list[1] # will print 2 and not 1
```

```
## [1] "1"
```

With that In mind, I'll switch my focus to more data analysis specific tasks and the process it takes to set them up!

Donor ID

For this project the goal is to compare the names of folks who are on record of donating to races with names of known democrats in our district to see who has donated the most and can be targeted through their address. While this isn't the only way to target donors it is a use case for R analysis.

Our first job on this project is to import the data and do some stock data cleaning to the values. R has a few eccentricities when it comes to data from spreadsheets so I'll be adding some of them in as code.

```
## Importing FEC Data on contributions
texas_contributions <- read.csv("Texas_contributions.csv") ## Importing the contributions
# it is a big file and may take some time to down

## Data often has rownames which count as a column and can mess up indexing
rownames(texas_contributions) <- NULL

## the original column names were not helpful so I'm replacing them with my own
colnames_prep <- c("ROW","NAME", "CITY", "STATE", "ZIP_CODE", "TRANSACTION_AMT")
colnames(texas_contributions) <- colnames_prep

## The colnames() and rownames() functions can be used to replace those values,
### with colnames being the most useful in my experience

## We can also print out information like this to briefly view it
head(texas_contributions) ## this prints out the first 6 rows like it does in Python
```

```
##   ROW                     NAME         CITY STATE  ZIP_CODE TRANSACTION_AMT
## 1   1      RUDY, DEBORAH MRS. LAKE JACKSON    TX     78749            1500
## 2   2          PRYWES, JOSHUA      HOUSTON    TX 752114515            1500
## 3   3              LAVIN, DAN     PEARLAND    TX 760347502            1500
## 4   4 WILLIAMS WALKER, TRACEY FLOWER MOUND    TX 773385059              10
## 5   5                L, LAURA       AUSTIN    TX     77089               5
## 6   6          BOLDEN, HATTIE       FRESNO    TX 773042246             125
```

```
colnames(texas_contributions) # you also don't need a print statement in R markdown
```

```
## [1] "ROW"             "NAME"            "CITY"            "STATE"
## [5] "ZIP_CODE"        "TRANSACTION_AMT"
```

```
## to bring up indexing again,
texas_contributions[1,3] # will give you the 1st row and 3rd column
```

```
## [1] "LAKE JACKSON"
```

```
texas_contributions[1:10,2] # will give you the second columns and the 1st through 10th rows
```

```
##  [1] "RUDY, DEBORAH MRS."      "PRYWES, JOSHUA"
##  [3] "LAVIN, DAN"              "WILLIAMS WALKER, TRACEY"
##  [5] "L, LAURA"                "BOLDEN, HATTIE"
##  [7] "PARKS, MARTHA"           "SHELTON, GEORGE"
##  [9] "MOSS, WILLIAM"           "SHELTON, GEORGE"
```

I would also like to mention that all data in this repository has been scrambled so that names do not align with the correct donation amounts or locations. I will show you the process as I do it to explain more about R but if you are following along no need to duplicate it

```
# Reading in data
Van_data <- read.csv("132nd_pro_dem_voters.csv")
```

```
?sample # used to pull up info on the scramble function in RStudio
```

```
## starting httpd help server ... done
```

```
# I want to randomly re-order the data in the spreadsheet
Scrambler <- function(column){ ## Functions are formatted slightly differently in R with {}
  scrambled <- sample(column,length(column), replace = FALSE) # length instead of len
  return(scrambled) # return with parenthesis
}

## Scrambling each column of interest,
### you can select columns by name with a $ after the variable name
Van_data$Voter.File.VANID <- Scrambler(Van_data$Voter.File.VANID)
Van_data$Address <- Scrambler(Van_data$Address)
Van_data$City <- Scrambler(Van_data$City)
Van_data$Zip5 <- Scrambler(Van_data$Zip5)
Van_data$Zip4 <- Scrambler(Van_data$Zip4)

## Lets do it to Texas Contributions for good measure
texas_contributions$STATE <- Scrambler(texas_contributions$STATE)
texas_contributions$CITY <- Scrambler(texas_contributions$CITY)
texas_contributions$ZIP_CODE <- Scrambler(texas_contributions$ZIP_CODE)

## With that we can move to analysis
```

So now we have two datasets, one with the contributors and their amounts another with names and locations. The obvious next step is joining these two and then doing some analysis on the results!

Examining the data

So we have two different data sets with what should be the same information but there are some differences between the structure of the two files. Ideally we could just merge the two data frames together but unfortunately it is not that simple. Lets look at the data to see this:

```
library(tidyverse) ## I've already loaded it in but I'll be using some tidyverse functions here
 ## so this is a reminder

## %>% is a piping operator used to push one value into another equation
texas_contributions %>% select(NAME) %>% head() # One column Capitalize
```

```
##                       NAME
## 1      RUDY, DEBORAH MRS.
## 2          PRYWES, JOSHUA
## 3              LAVIN, DAN
## 4 WILLIAMS WALKER, TRACEY
## 5                L, LAURA
## 6          BOLDEN, HATTIE
```

```
Van_data %>% select(c(LastName, FirstName, MiddleName, Suffix)) %>% head() # Four columns first letter
```

```
##            LastName FirstName MiddleName Suffix
## 1          Michieli   Aneline   Michieli
## 2               Ray    Robert        Lee     II
## 3             Jones   Kameron     Tyrell
## 4          Redmond   Destiny     Noelle
## 5 Figueroa-Saettone    Silvia  Valentina
```

```
## 6   Chacon Gonzalez     Maria   Karolina
```

So to recap, the NAME column from the Texas contributions dataset has only one column with a full name arranged last, first honorific while each of those is its own column or missing in the VAN dataset. At this point to merge them we can either try to turn the Texas data into the VAN data or vice versa. I chose the latter option in this problem, converting the VAN data into the other format but if you are interested, try doing this in reverse on your own (hint: the separate() function can help you split it up to start). With that being the case. we'll start taking the VAN data columns and then rearranging them. The main challenge in this is that we don't know if peoples names are exactly matches i.e. does someone have a mrs. in one and a ms. in another or a middle name in one and not in the other. In order to solve this we are going to throw a for loop at it and do all of the options to try and get hits between the two. Here is how we do it.

```
## cleaning step

good_names <- c()

## arranging names
for(i in 1:4){ # Larger for loop for middle name options
  if( i == 1){ ## No middle name
    Van_prep <- Van_data %>% mutate(Name = paste(toupper(LastName),toupper(FirstName),sep = ", "))
      ## this uses the paste function to merge two strings
      ## toupper() also makes everything uppercase
      ## Mutate adds a new column to the dataset and the <- assigns it to Van_prep
  } else if(i == 2){ ## Middle name
    Van_prep <- Van_data %>% mutate(Name = paste(paste(toupper(LastName),toupper(FirstName),sep = ", ")
  } else if(i == 3){ ## Middle Initial with dot
    Van_prep <- Van_data %>% mutate(Name = paste(paste(toupper(LastName),toupper(FirstName),sep = ", ")
  } else if(i == 4){ # middle initial
    Van_prep <- Van_data %>% mutate(Name = paste(paste(toupper(LastName),toupper(FirstName),sep = ", ")
  }
  for(g in 1:11){ # Honorific round, trying each combination on name to match
    if(g == 1){ # No honorific
      Van_prep <- Van_prep %>% mutate(NAME = Name)
    } else if(g == 2){ ## adding MR.
      Van_prep <- Van_prep %>% mutate(NAME = paste(Name,"MR.",sep = " "))
    } else if(g == 3){ ## adding MRS.
      Van_prep <- Van_prep %>% mutate(NAME = paste(Name,"MRS.",sep = " "))
    } else if(g == 4){ ## adding MS.
      Van_prep <- Van_prep %>% mutate(NAME = paste(Name,"MS.",sep = " "))
    } else if(g == 6){ ## adding DR.
      Van_prep <- Van_prep %>% mutate(NAME = paste(Name,"DR.",sep = " "))
    } else if(g == 7){ ## adding M.D.
      Van_prep <- Van_prep %>% mutate(NAME = paste(Name,"M.D.",sep = " "))
    } else if(g == 8){ ## adding MR
      Van_prep <- Van_prep %>% mutate(NAME = paste(Name,"MR",sep = " "))
    } else if(g == 9){ ## adding MRS
      Van_prep <- Van_prep %>% mutate(NAME = paste(Name,"MRS",sep = " "))
    } else if(g == 10){ ## adding MS.
      Van_prep <- Van_prep %>% mutate(NAME = paste(Name,"MS",sep = " "))
    } else if(g == 11){ ## Adding DR
      Van_prep <- Van_prep %>% mutate(NAME = paste(Name,"DR",sep = " "))
    }
    good <- inner_join(x = Van_prep, y = texas_contributions, by = "NAME")
      ## Inner join to find all names in each format which allign with each other
    saved <- good ##saving all of the contributions
```

```
    good <- good[!duplicated(good$NAME), ] # Saving all of the names with no duplicated
    if(i == 1 & g == 1){ ## this is creating a new dataframe for the responses
      good_names <- good
      saving <- saved
    } else{ ## appending to that dataframe
      good_names <- rbind(good_names,good)
      saving <- rbind(saving,saved)
    }
  }
}
```

```
## Warning in inner_join(x = Van_prep, y = texas_contributions, by = "NAME"): Detected an unexpected ma
## i Row 140 of `x` matches multiple rows in `y`.
## i Row 84985 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
## Warning in inner_join(x = Van_prep, y = texas_contributions, by = "NAME"): Detected an unexpected ma
## i Row 609 of `x` matches multiple rows in `y`.
## i Row 7995 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
## Warning in inner_join(x = Van_prep, y = texas_contributions, by = "NAME"): Detected an unexpected ma
## i Row 5553 of `x` matches multiple rows in `y`.
## i Row 89932 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
## Resulting transactions from our district and Names of people to contact
head(saving)
```

```
##     X Voter.File.VANID                    Address    City State  Zip5 Zip4
## 1  51          2659575    10503 Three Rivers Way  Cypress    TX 77450 3616
## 2  81         41256522        22122 Kerryblue Dr     Katy    TX 77450 3309
## 3 140         37362144      16151 Lower Pecos St     Katy    TX 77433 1711
## 4 140         37362144      16151 Lower Pecos St     Katy    TX 77433 1711
## 5 172         42858131 600 Park Grove Dr Apt 238  Houston    TX 77095 7696
## 6 172         42858131 600 Park Grove Dr Apt 238  Houston    TX 77095 7696
##   LastName FirstName MiddleName Suffix            Name            NAME    ROW
## 1   Taylor      Mary    Annette           TAYLOR, MARY    TAYLOR, MARY 239921
## 2  Johnson    Donald      Keith        JOHNSON, DONALD JOHNSON, DONALD 150765
## 3    Reyes     Jesus    Alberto           REYES, JESUS    REYES, JESUS 136276
## 4    Reyes     Jesus    Alberto           REYES, JESUS    REYES, JESUS 210967
## 5    Davis   Barbara                    DAVIS, BARBARA  DAVIS, BARBARA 225057
## 6    Davis   Barbara                    DAVIS, BARBARA  DAVIS, BARBARA 238641
##          CITY STATE  ZIP_CODE TRANSACTION_AMT
## 1     GARLAND    TX 760345017             100
## 2     HOUSTON    TX 750025331             500
## 3 SAN ANTONIO    TX 760874416              25
## 4     HOUSTON    TX 773794678              25
## 5     HOUSTON    TX     77043              25
## 6     HOUSTON    TX     77642               2
```

```
head(good_names)
```

```
##     X Voter.File.VANID                    Address    City State  Zip5 Zip4
## 1  51          2659575    10503 Three Rivers Way  Cypress    TX 77450 3616
## 2  81         41256522        22122 Kerryblue Dr     Katy    TX 77450 3309
## 3 140         37362144      16151 Lower Pecos St     Katy    TX 77433 1711
## 5 172         42858131 600 Park Grove Dr Apt 238  Houston    TX 77095 7696
## 7 186          8232212             1906 Boren Dr  Cypress    TX 77433 5250
## 9 206          6492079  24227 Schivener House Ln     Katy    TX 77433 1450
##    LastName FirstName MiddleName Suffix               Name               NAME
## 1    Taylor      Mary    Annette              TAYLOR, MARY       TAYLOR, MARY
## 2   Johnson    Donald      Keith           JOHNSON, DONALD    JOHNSON, DONALD
## 3     Reyes     Jesus    Alberto              REYES, JESUS       REYES, JESUS
## 5     Davis   Barbara                       DAVIS, BARBARA     DAVIS, BARBARA
## 7 Hernandez   Eduardo       Luis        HERNANDEZ, EDUARDO HERNANDEZ, EDUARDO
## 9  Williams    Donald       Ward          WILLIAMS, DONALD   WILLIAMS, DONALD
##      ROW        CITY STATE  ZIP_CODE TRANSACTION_AMT
## 1 239921     GARLAND    TX 760345017             100
## 2 150765     HOUSTON    TX 750025331             500
## 3 136276 SAN ANTONIO    TX 760874416              25
## 5 225057     HOUSTON    TX     77043              25
## 7  78139   NEW CANEY    TX     75701              15
## 9  81577      IRVING    TX 750506686             100
```

So after all that work we are left with a dataset of names and transactions which we can use to find who has made donations in the district and how much! These names can also be taken back to VAN and compared with phone numbers to have an outreach short list. All of these things are now possible with this dataset.

A follow up question then would be how does this compare to a similar workflow in python. However this is another thing I'll leave to you if you are interested.

As one last show of R's capabilities, say we want to rank the total contributions of people from highest to lowest. R also has a group by function which can be piped into summaries as follows:

```
## Using group_by and summarise to generate a unique table
ranked_donations <- saving %>% group_by(Name) %>% summarise(Total_donations = sum(TRANSACTION_AMT))
## Ranking them with the order() function inside the indexing
ranked_donations <- ranked_donations[order(ranked_donations$Total_donations, decreasing = TRUE), ]

head(ranked_donations)
```

```
## # A tibble: 6 x 2
##   Name            Total_donations
##   <chr>                     <int>
## 1 LEE, DAVID               357000
## 2 SMITH, ROBERT            303792
## 3 GARCIA, DAVID             13953
## 4 EDWARDS, JAMES            11800
## 5 JOHNSON, ERIC             10896
## 6 JOHNSON, ROBERT            9344
```

We can see that in this example David Lee is highrolling with $357,000 donated, so he and others on this list would be good people for the campaign and candidate to call. As a reminder though this data is scrambled so please do not actually call anyone on this list.

Conclusion

R may not be peoples first choice when it comes to programming but it has a strong community and more use cases then people would expect. This is only the tip of the iceberg when it comes to the language and I

hope this can act as a jumping off point for people to mess around with it themselves and bring it into their Bluebonnet Projects!