ID Analytics

Data analysis exercise

Cristian Heredia

ID ANALYTICS

"You have applied for the role of Data Scientist at ID Analytics. To move forward, we ask you to perform a data analysis exercise. The purpose of this exercise is to provide an example of how you organize, perform, and communicate your work. The exercise itself is not meant to be difficult or particularly time consuming. We value our time, and we value your time. Feel free to explain what you would do if you had more time. Please include all of the source code for your analysis in your report, and please describe all of the steps. We should be able to reproduce your analysis with little work using the content of your report."

April 2017

Contents

1. Ordered list of fields

1.1 Concatenate nested field names

The following operations were compiled with R-Studio. My general approach was to work with a small subset of the data, e.g., ten entries. Once I was able to answer the prompt to the best of my abilities, I scaled up to the full 150,000 observations. The R file is attached as IDA_exercise.R. This document was created with LaTeX.

The original JSON data file was assigned to the data frame *large_data*. Immediately after reading in the file NULL and empty spaces were replaced with NA for downstream operations. The PLYR package was used to unpacked the nested *name* fields. These unpacked fields were then named,

"name.first","name.last", "name.middle", "name."

The original *name* column was deleted from *large_data*, and the above previously nested fields were added to *large_data*. I applied a similar method to the address fields. For a neater solution, I would have preferred code that concatenates field names from the original JSON file. However, for brevity, I chose to hand code the column names.

The final column list in alphabetical order:

"address", "address.city", "address.state", "address.street", "address.zip", "dob", "email", "id", "name", "name.first", "name.last", "name.middle", "phone", "record_date", "ssn"

R Stream in json file, convert NULL to NA

```
> json_file <- stream_in(file("ida_wrangling_exercise_data.2017-02-13.jsonl.gz"))
> json_file$address[json_file$address=="NULL"] <- "NA"
> json_file$name[json_file$name=="NULL"] <- "NA"
> large_data <- json_file
```

This will unpack nested name lists and return strings for names

```
> nested_names_large <- ldply (large_data$name, data.frame)
> nested_names_large <- data.frame(lapply(nested_names_large, as.character),
  stringsAsFactors=FALSE)
> colnames(nested_names_large) <- c("name.first","name.last", "name.middle",
  "name")
> large_data$name <- NULL
> large_data <- cbind(large_data, nested_names_large)
```

Concatenate address fields

```
> nested_address_large <- ldply (large_data$address, data.frame)
> nested_address_large <- data.frame(lapply(nested_address_large, as.character),
  stringsAsFactors=FALSE)
> colnames(nested_address_large) <- c("address.street","address.city", "address.state",
  "address.zip","address")
> large_data$address <- NULL
> large_data <- cbind(large_data, nested_address_large)
```

Arrange columns in alphabetical order

```
> large_data <- large_data[ , order(names(large_data)) ]
> names_list <- colnames(large_data)
> print(names_list)
```

2. Analyzing individual fields

2.1 What percentage of the records contain the field?

To find the percentage of a field the number of occurrences was calculated, then divided by the number of rows. The percent() function was used to neatly present the data. The data is saved in the field_percentage data frame, and presented in Table 2.1.

Table 2.1: Individual field occurrences

Field name	Percentage
address	59.20%
address.city	40.80%
address.state	40.80%
address.street	40.80%
address.zip	40.80%
name	30%
name.fist	70%
name.last	70%
name.middle	29.10%

Fields with entries in every row were omitted from Table 2.1.

R Determine number of instances for field, divide by number of rows, then print

```
> a <- percent(sum(complete.cases(large_data$address))/nrow(large_data))
> a.city <- percent(sum(complete.cases(large_data$address.city))/nrow(large_data))
> a.state <- percent(sum(complete.cases(large_data$address.state))/nrow(large_data))
> a.street <- percent(sum(complete.cases(large_data$address.street))/nrow(large_data))
> a.zip <- percent(sum(complete.cases(large_data$address.zip))/nrow(large_data))
> n.first <- percent(sum(complete.cases(large_data$name.first))/nrow(large_data))
> n.last <- percent(sum(complete.cases(large_data$name.last))/nrow(large_data))
> n.middle <- percent(sum(complete.cases(large_data$name.middle))/nrow(large_data))
> field_percentage <- c(a, a.city,a.state,a.street,a.zip,n,n.first,n.last,n.middle)
> names(field_percentage) <- c("address","address.city","address.state","address.street","address.zip","name",
> field_percentage <- data.frame(field_percentage)
> print(field_percentage)
```

2.2 What are the five most common values of the field?

A table function was used to analyze the individual fields. The tables were then sorted by the five most common elements. These individually sorted tables were then displayed as bar plots. Given more time I would have preferred a more elegant solution that stepped through each column of the large_data data frame and sorted the common elements.

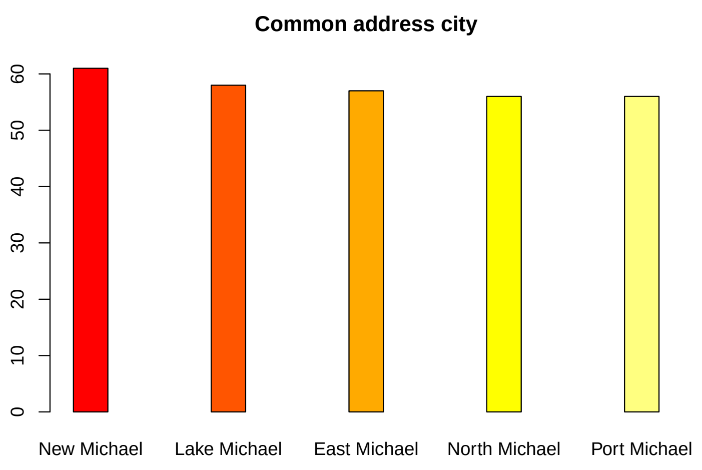

Figure 2.1: Five most common cities in address.city field. New Michael: 61; Lake Michael: 58; East Michael: 57; North Michael:56; Port Michael: 56

The following code was repeated for each field

R Field: address.city

```
> ac <- sort(table(large_data$address.city),decreasing=TRUE)[1:5]
> barplot(ac, main = "Common address city", space=3,col=heat.colors(5))
```

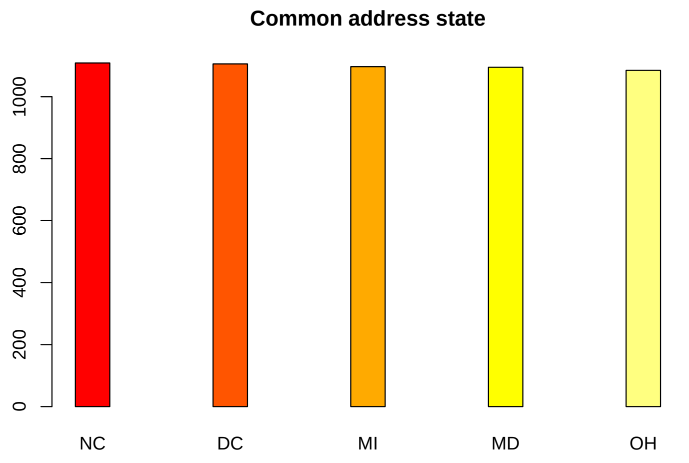

Figure 2.2: Five most common states in address.city field. NC: 1109; DC: 1106; MI: 1097; MD: 1095; OH: 1085

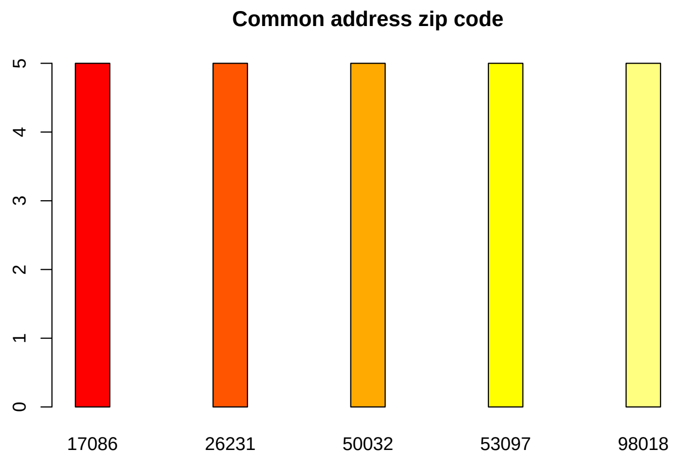

Figure 2.3: Five most common zip codes in address.city field. Each common zip code appeared five times.

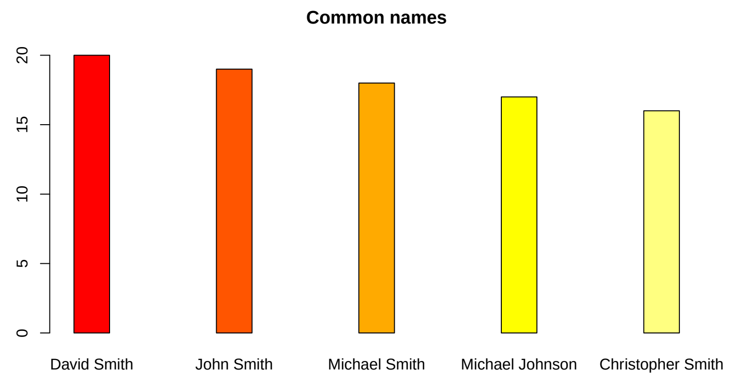

Figure 2.4: Five most common names in name field.

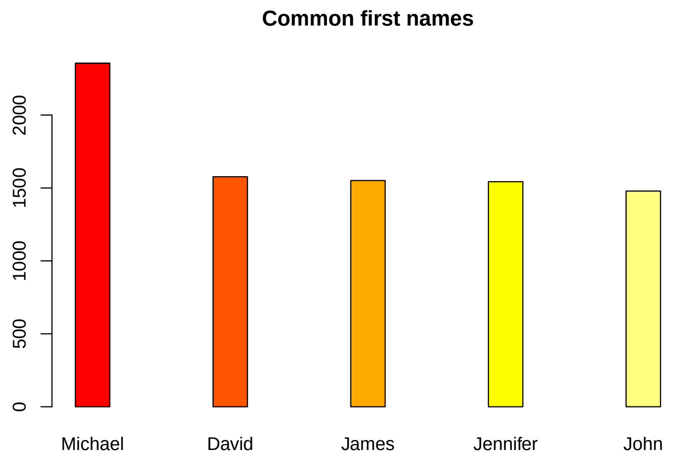

Figure 2.5: Five most common names in name.first field.

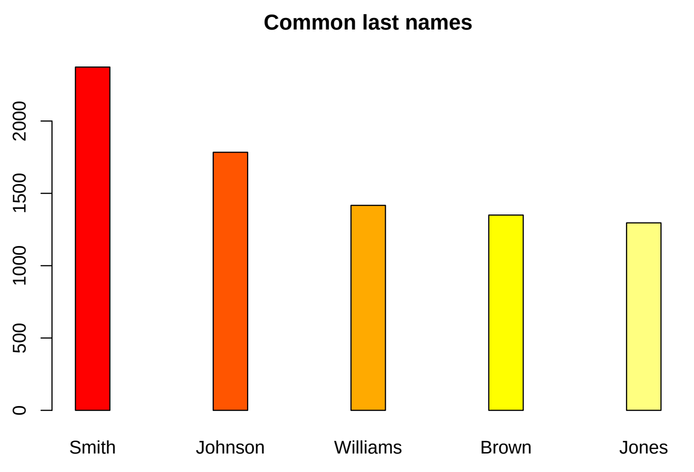

Figure 2.6: Five most common names in name.last field.

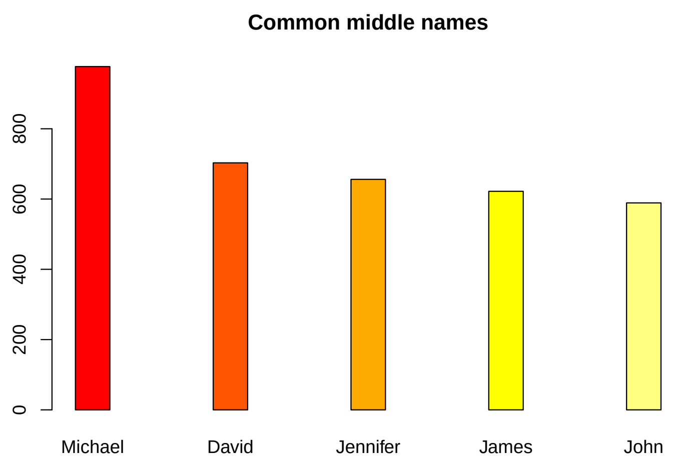

Figure 2.7: Five most common names in name.middle field.

3. Unique names

3.1 How many distinct first names appear in this data set?

To determine the number of unique first names I looked in the name.first column. From that column, I excluded any NA values, which made up the majority of unique values. Then the unique() function was used to process all unique names. Finally, the length() function yielded the total number of unique names.

"Number of unique first names: 690"

R How many unique first names in small set, excluding NA values

```
> paste("Number of unique first names:", length(unique(na.exclude(large_data$name.first))))
```

4. Unique street names

4.1 How many unique street names?

To find the number of unique street names I looked in the address.street field. Within that field, the street numbers were stripped using strsplit(), row by row. The results were bound to another data frame with two fields: Street Number and Street Name.

Once street names were isolated, the unique() function was executed on the column, as used in 3.1. I suspect there is a more efficient method of isolating strings. Additionally, with more time I would have included the general address field. Therefore this isn't a complete list of street names, only street names from the address.street column.

"Unique street names: 56858"

R Strips numbers from address.street

```
> street_names <- lapply(strsplit(large_data$address.street, "(?<=
  d)
  b ", perl=T), function(x) if (length(x)<2) c("", x) else x)
> street_names <- do.call(rbind, street_names)
> colnames(street_names) <- c("Street Number", "Street Name")
```

R Find Unique names in street_names, excluding NA

```
> street_names<- data.frame(street_names)
> street_names <- data.frame(lapply(street_names, as.character), stringsAsFac-
  tors=FALSE)
> paste("Unique street names:",length(unique(na.exclude(street_names$Street.Name))))
> barplot(sort(table(street_names$Street.Name),decreasing=TRUE)[1:5], main =
  "Five most common street names", space=3, col=heat.colors(5) )
```

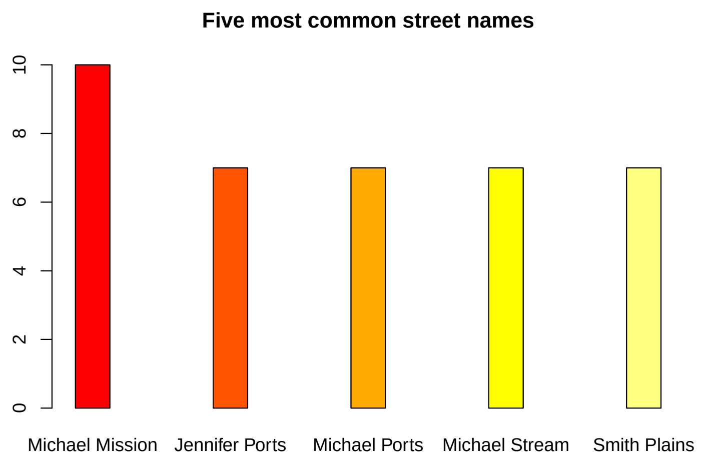

Figure 4.1: Five most common street names

5. Five most common area codes

To find the five most common area codes in the phone number field I began by stripping all special characters. The special characters were stripped with the gsub() function. Then I attempted to strip the "1 " in front of some of the numbers, again with gsub(). Invoking gsub() again I removed all whitespace. Finally, with substring(), I isolated the first three numbers, presumably the area code. As a first time user of R, stripping strings is definitely an area for improvement. Regardless, the results are plotted below:

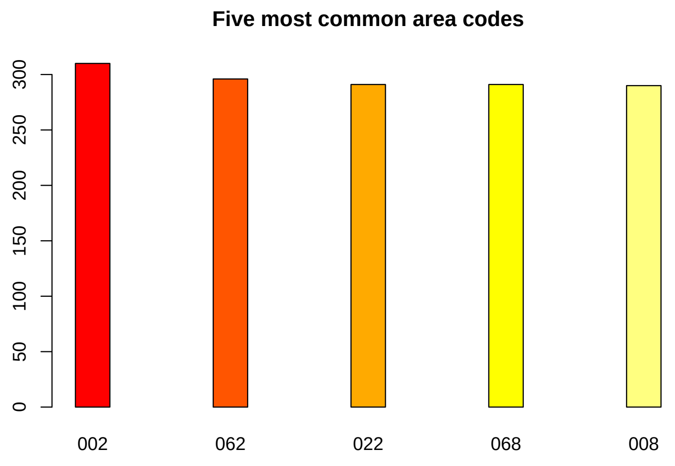

Figure 5.1: Five most common area codes

R Approach for finding US area codes

```
> area_codes <- large_data$phone
> area_codes <- gsub("[[:punct:]]", " ", area_codes)
> area_codes <- gsub("1 ", " ", area_codes)
> area_codes <- gsub(" ", "", area_codes, fixed = TRUE)
> area_codes <- substring(area_codes, 1, 3)
> barplot(sort(table(area_codes),decreasing=TRUE)[1:5], main = "Five most com-
  mon area codes", space=3, col=heat.colors(5) )
```