homework iv

Vivek Panchal,Rohit Kunjilikattil

2019-10-3

Introduction

In this assignment, we clean the NYC311 data to make sure it conforms to the tidy data standards. For this we use various tidyr functions like **spearate** and **filter**. We transform the dataset into tidy data. Further we select another dataset, the nyc_payroll_information dataset, and clean that dataset so that it too follows the standards of tidy data.

Tidyverse is a r package that helps in making data tidy. Tidy data is defined as data which follows three basic rules : each variable must have its own column, each observation must have its own row and finally each value must have its own cell.One of the advantages of tidy data is it's easier to work with data that always has a consistent structure. Further, R's vectorised nature makes it particularly easy to transform tidy data.

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
install.packages('tidyverse', repos = "http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/b8/hxw52n9d3wz9th1p6y6sb8zh0000gn/T//RtmppaYGq8/downloaded_packages
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(plyr)
```

```
## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:purrr':
##
##     compact
```

```
library(dplyr)
```

```
nyc311_a = fread("311_Service_Requests_from_2010_to_Present.csv")
```

Cleaning Data

Here, we first try to clean data as much as possible so that we do not have to waste our processing power on data that we don't want or on bad data. As the complaint type is one of the most important columns, we try to make it as uniform as possible. We go about doing this by removing any unwanted spaces, replacing hyphens with slashes, etc. We then filter the zipcodes to only have values of length 5.

```
nyc311_a$Complaint.Type = tolower(nyc311_a$Complaint.Type)
nyc311_a$Complaint.Type = gsub('s$', '', nyc311_a$Complaint.Type)
nyc311_a$Incident.Zip = gsub('-[[:digit:]]{4}$', '',nyc311_a$Incident.Zip)
nyc311_a$Complaint.Type = gsub('paint - plaster', 'paint/plaster', nyc311_a$Complaint.Type)
nyc311_a$Complaint.Type = gsub('general construction', 'construction', nyc311_a$Complaint.Type)
nyc311_a$Complaint.Type = gsub('nonconst', 'construction', nyc311_a$Complaint.Type)
nyc311_a$Complaint.Type = gsub('street sign - [[:alpha:]]+', 'street sign', nyc311_a$Complaint.Type)
nyc311_a$Complaint.Type = gsub('fire alarm - .+','fire alarm', nyc311_a$Complaint.Type)
nyc311_clean2 = nyc311_a
```

Separating Columns

Here, we see multiple columns that are of class character that have date and time values together in one column. So we use the **separate** function from tidyr library to separate such columns into separate columns with date and time stored separately.

```
nyc311_clean2 <- nyc311_clean2 %>%
  separate(`Created Date`, into = c("Created Date", "Created Time"), sep = 10)

nyc311_clean2 <- nyc311_clean2 %>%
  separate(`Closed Date`, into = c("Closed Date", "Closed Time"), sep = 10)

nyc311_clean2 <- nyc311_clean2 %>%
  separate(`Due Date`, into = c("Due Date", "Due Time"), sep = 10)

nyc311_clean2 <- nyc311_clean2 %>%
  separate(`Resolution Action Updated Date`, into = c("Resolution Action Updated Date", "Resolution Acti
```

Removing unwanted columns

Further we remove rows which have 'Unspecified' listed for Boroughs. Finally as all of this data is from New York City, we remove the city column as it is redundant.

```
library(data.table)
nyc311_clean2 <- data.table(nyc311_clean2)

nyc311_clean2 <- nyc311_clean2[,c("City"):=NULL]
nyc311_clean2 <- nyc311_clean2 %>% filter(!str_detect(Borough, "Unspecified"))
```

Reading a dataset from a url

Here we use r to read and download the csv file directly from the url. The saves us the trouble of sharing the csv file along with the rmd file. This way is much more efficient. We are using NYC payroll data. This dataset contians the salary, pay rate and total compensation of every New York City employees from year 2014 till 2017.This dataset provides columns for fiscal year, agency they work for, borough they are working in.This will provide us with an insight into who gets paid how much and for what. The source dataset that we are extracting from url already follows the rules of tidydata. Hence, we do not use tidyr functions to further clean the data.

```
nyc_payroll <- fread("https://data.cityofnewyork.us/api/views/k397-673e/rows.csv?accessType=DOWNLOAD")
```

There are 3 types of pay basis i.e Per hour, Per day and Per annum. We are trying to explore this by finding the total count of such employees working in New York city.

```
nyc_payroll %>% dplyr::group_by(`Pay Basis`) %>% dplyr::summarise(n = n())
```

```
## # A tibble: 4 x 2
##   `Pay Basis`           n
##   <chr>             <int>
## 1 per Annum       1988530
## 2 per Day          720694
## 3 per Hour         608106
## 4 Prorated Annual   16038
```

Conclusion

In this assignment, we learnt the definition of tidy data and implemented steps to convert normal or 'messy' data to tidy data. Tidy data is data that conforms to the basic rule that each variable, observation and value must have its own column, row and cell respectively. Further, we introduced a new dataset using R's capability to directly read data from a URL. We choose the Nyc Payroll Dataset. This dataset can shows the disparities in payroll based on Borough, gender and job types. We can use this dataset in conjunction with the NYC311 data to find out which agencies spend what on salaries. Then we can map out whether their is a correlation between high salaries and better performance as in whether departments that have higher salaries are better and faster at resolving complaints.