Basic Data Preprocessing

Code ▾

Phankawee Mukdaprakorn

25 March 2024

Setup

Hide

```
library(kableExtra)
library(magrittr) #to be able to execute pipe-operator
library(readr) #to read .csv file
library(dplyr) #to manipulate data
library(here) #to automate setting the working directory
```

Data Description

This dataset has been collected by Los Angeles Police Department(LAPD) since 2020. It demonstrates the crime incidents in Los Angeles. However, because it is digitalised from hard-copied reports which have been stored for long time, there may be some missing information in some rows. This dataset can be downloaded from data.lacity.org (2024) as a `.csv` file.

The dataset consists of 28 fields but for the purpose of this assignment, it will be subset into 18 fields which are explained below:

- **DR_NO** : This field is an unique number of each incident report called “Division of Records Number”. The first two-digit indicates the year when the incident is reported, following by the area ID in Los Angeles and 5 more digits to make it unique.
- **Date Rptd** : This field shows the date when the incident is reported.
- **DATE OCC** This field demonstrates the date when the incident occurs.
- **AREA NAME** : This field shows the place of the police Stations among 21 area in LAPD which has been reported the incident.
- **Rpt Dist No** : This code indicates the sub-area where the incident occurs.
- **Crm Cd Desc** : This field is the description of the crime code 1 which is considered the most prioritized crime from each incident.
- **Vict Age** : This field shows number of age of the victim in the case.
- **Vict Sex** : This field shows the victim’s gender in the case. F, M, and X stand for female, male, and unknown, respectively.
- **Vict Descent** : This presents the descent of the victim.
- **Premis Cd** : This code indicates the type of the location where the incident occurs.
- **Premis Desc** : This field describes the location where the incident occurs.
- **Weapon Desc** : This is a broad description of the weapon which is used in the crime.
- **Status Desc** : This field shows the status of the incident.
- **Crm Cd 1** This field indicates the primary crime occured in the incident.
- **Crm Cd 2** : This field indicates the less serious crime which happen in the incident compared to Crm Cd1.
- **Crm Cd 3** : This field indicates the less serious crime which happen in the incident compared to Crm Cd1 and Crm Cd2.
- **LOCATION** : This field show the address where the crime incident occur.
- **Cross Street** : This field shows the cross street by the address near the location where the incident takes place.

Read/Import Data

Hide

```
setwd(here())
path <- "https://www.dropbox.com/scl/fi/moekhj57kbo1qy6q5rvdj/Crime_Data_from_2020_to
_Present.csv?rlkey=qg97ery60gfkmy2srco7zq5y3&st=rfu12prs&dl=1"
crime <- read_csv(path, col_names = TRUE, col_select = c("DR_NO","Date Rptd","DATE OC
C","AREA NAME","Rpt Dist No","Crm Cd Desc","Vict Age","Vict Sex","Vict Descent","Prem
is Cd","Premis Desc","Weapon Desc","Status Desc","Crm Cd 1","Crm Cd 2","Crm Cd 3","LO
CATION","Cross Street"))
```

```
Rows: 910707 Columns: 18── Column specification ─────────────────────────────────────
──────────────────────────────────────────────
Delimiter: ","
chr (12): Date Rptd, DATE OCC, AREA NAME, Rpt Dist No, Crm Cd Desc, Vict Sex, Vict De
scent, Premis Desc...
dbl  (6): DR_NO, Vict Age, Premis Cd, Crm Cd 1, Crm Cd 2, Crm Cd 3
ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Putting the file path within the `path` variable first does not only make it more convenience, but also easier to read and understand the process. I read my `.csv` raw data file using `read_csv` from *readr* package. I also set the argument `col_select()` to specify only the fields that I would like to import which are 18 columns that I mentioned above in this report. I, then, named my dataset " **`crime`** " as its related topic.

Inspect and Understand

Hide

```
dim(crime)
```

```
[1] 910707     18
```

Hide

```
colnames(crime)
```

```
 [1] "DR_NO"          "Date Rptd"      "DATE OCC"       "AREA NAME"      "Rpt Dist No"    "Crm
Cd Desc"
 [7] "Vict Age"       "Vict Sex"       "Vict Descent" "Premis Cd"      "Premis Desc"    "Weap
on Desc"
[13] "Status Desc"    "Crm Cd 1"       "Crm Cd 2"       "Crm Cd 3"       "LOCATION"       "Cros
s Street"
```

Hide

```
head(crime,5)
```

DR_NO	Date Rptd	DATE OCC	AREA NAME	Rpt Dist No
<dbl>	<chr>	<chr>	<chr>	<chr>
190326475	03/01/2020 12:00:00 AM	03/01/2020 12:00:00 AM	Wilshire	0784
200106753	02/09/2020 12:00:00 AM	02/08/2020 12:00:00 AM	Central	0182
200320258	11/11/2020 12:00:00 AM	11/04/2020 12:00:00 AM	Southwest	0356
200907217	05/10/2023 12:00:00 AM	03/10/2020 12:00:00 AM	Van Nuys	0964
220614831	08/18/2022 12:00:00 AM	08/17/2020 12:00:00 AM	Hollywood	0666

5 rows | 1-5 of 18 columns

I used `dim()` to see the dimension of my **`crime`** dataset. I also check the column names using `colnames()` to confirm the right columns I have imported. After that, I used `head()` to take a glance at first 5 rows of my dataset.

Hide

```
str(crime)
```

```
tibble [910,707 × 18] (S3: tbl_df/tbl/data.frame)
 $ DR_NO       : num [1:910707] 1.90e+08 2.00e+08 2.00e+08 2.01e+08 2.21e+08 ...
 $ Date Rptd   : chr [1:910707] "03/01/2020 12:00:00 AM" "02/09/2020 12:00:00 AM" "1
1/11/2020 12:00:00 AM" "05/10/2023 12:00:00 AM" ...
 $ DATE OCC    : chr [1:910707] "03/01/2020 12:00:00 AM" "02/08/2020 12:00:00 AM" "1
1/04/2020 12:00:00 AM" "03/10/2020 12:00:00 AM" ...
 $ AREA NAME   : chr [1:910707] "Wilshire" "Central" "Southwest" "Van Nuys" ...
 $ Rpt Dist No : chr [1:910707] "0784" "0182" "0356" "0964" ...
 $ Crm Cd Desc : chr [1:910707] "VEHICLE - STOLEN" "BURGLARY FROM VEHICLE" "BIKE - ST
OLEN" "SHOPLIFTING-GRAND THEFT ($950.01 & OVER)" ...
 $ Vict Age    : num [1:910707] 0 47 19 19 28 41 25 27 24 26 ...
 $ Vict Sex    : chr [1:910707] "M" "M" "X" "M" ...
 $ Vict Descent: chr [1:910707] "O" "O" "X" "O" ...
 $ Premis Cd   : num [1:910707] 101 128 502 405 102 501 502 248 750 502 ...
 $ Premis Desc : chr [1:910707] "STREET" "BUS STOP/LAYOVER (ALSO QUERY 124)" "MULTI-U
NIT DWELLING (APARTMENT, DUPLEX, ETC)" "CLOTHING STORE" ...
 $ Weapon Desc : chr [1:910707] NA NA NA NA ...
 $ Status Desc : chr [1:910707] "Adult Arrest" "Invest Cont" "Invest Cont" "Invest Co
nt" ...
 $ Crm Cd 1    : num [1:910707] 510 330 480 343 354 354 354 354 354 624 ...
 $ Crm Cd 2    : num [1:910707] 998 998 NA NA NA NA NA NA NA NA ...
 $ Crm Cd 3    : num [1:910707] NA NA NA NA NA NA NA NA NA NA ...
 $ LOCATION    : chr [1:910707] "1900 S  LONGWOOD                     AV" "1000 S  FL
OWER                       ST" "1400 W  37TH                         ST" "14000    RI
VERSIDE                    DR" ...
 $ Cross Street: chr [1:910707] NA NA NA NA ...
 - attr(*, "spec")=
  .. cols(
  ..   DR_NO = col_double(),
  ..   `Date Rptd` = col_character(),
  ..   `DATE OCC` = col_character(),
  ..   `TIME OCC` = col_skip(),
  ..   AREA = col_skip(),
  ..   `AREA NAME` = col_character(),
  ..   `Rpt Dist No` = col_character(),
  ..   `Part 1-2` = col_skip(),
  ..   `Crm Cd` = col_skip(),
  ..   `Crm Cd Desc` = col_character(),
  ..   Mocodes = col_skip(),
  ..   `Vict Age` = col_double(),
  ..   `Vict Sex` = col_character(),
  ..   `Vict Descent` = col_character(),
  ..   `Premis Cd` = col_double(),
  ..   `Premis Desc` = col_character(),
  ..   `Weapon Used Cd` = col_skip(),
  ..   `Weapon Desc` = col_character(),
  ..   Status = col_skip(),
  ..   `Status Desc` = col_character(),
  ..   `Crm Cd 1` = col_double(),
  ..   `Crm Cd 2` = col_double(),
  ..   `Crm Cd 3` = col_double(),
  ..   `Crm Cd 4` = col_skip(),
  ..   LOCATION = col_character(),
  ..   `Cross Street` = col_character(),
  ..   LAT = col_skip(),
  ..   LON = col_skip()
  .. )
```

I also used `str()` to see the structure of my data. There are 917,707 records (I have downloaded this dataset on 18/03/2024) and 18 imported columns. And by using `read_csv()`, R will try to guess types of the data while importing. As be seen in the output from `str()`, it can be shown that there are some columns that need to be converted their datatypes.

Type Conversion

Hide

```
crime$DR_NO <- as.integer(crime$DR_NO)
crime$`Date Rptd` <- as.Date(crime$`Date Rptd`,"%m/%d/%Y")
crime$`DATE OCC` <- as.Date(crime$`DATE OCC`, "%m/%d/%Y")
crime$`AREA NAME` <- as.factor(crime$`AREA NAME`)
crime$`Rpt Dist No` <- as.factor(crime$`Rpt Dist No`)
crime$`Crm Cd Desc`<- as.factor(crime$`Crm Cd Desc`)
crime$`Vict Age` <- as.integer(crime$`Vict Age`)
crime$`Vict Sex` <- crime$`Vict Sex` %>% factor(.,
                                                levels = c("M","F","X"),
                                                labels = c("Male","Female","Unknow
n"))
crime$`Vict Descent` <- as.factor(crime$`Vict Descent`)
crime$`Premis Cd` <- as.integer(crime$`Premis Cd`)
crime$`Premis Desc` <- as.factor(crime$`Premis Desc`)
crime$`Weapon Desc` <- as.factor(crime$`Weapon Desc`)
crime$`Status Desc` <- as.factor(crime$`Status Desc`)
crime$`Crm Cd 1` <- as.integer(crime$`Crm Cd 1`)
crime$`Crm Cd 2` <- as.integer(crime$`Crm Cd 2`)
crime$`Crm Cd 3` <- as.integer(crime$`Crm Cd 3`)
```

I converted some columns' datatypes because of the reasons listing below:

- **DR_NO** : This field should be read as an integer because it is the unique number of the incident without the character inside. I used `as.integer()` to convert it.
- **Date Rptd** : Date report column had been read as a character at first. I changed its datatype to be date using `as.Date()` function and also, change the format by setting an argument inside the function to be `%m/%d/%Y` which I applied after reading a blog post from Indigo(2011). I chose to remove the time from the format because all rows are all the same as a default value in the system `12:00:00 AM`.
- **DATE OCC** : For this column, I proceeded the same operation as I did with **Date Rptd**.
- **AREA NAME** : As mentioned earlier in this report, area names indicate only 21 police stations in the area so this field can be considered as a categorical variable using `as.factor()` function.
- **Crm Cd Desc** : This one as well, can be a categorical variable because they are all same descriptions describing the crime codes.
- Vict Age This variable should be read as an integer, not double variable. Thus, I converted it using `as.integer()` function.
- **Vict Sex** : This column should be a categorical variable so I converted it using `factor()` function. In addition, I renamed the value inside from "M","F","X" to be "Male", "Female", and "Unknown", respectively, just to make it easier to understand by setting the argument inside `labels = c("Male","Female","Unknown")`. However, I left the value "H" as N/A since there is no description about it. Unlike "X" is described as "Unknown" in the description from the data source website
- **Vict Descent** : This column shows the descent of the victim so its datatype should be nominal, not just only character. Thus, I used `as.factor()` to convert it.

- **Premis Cd** : This should be read as integer since it is the code without an alphabet inside so I converted it using `as.integer()` .
- **Premis Desc**, **Weapon Desc**, **Status Desc** : They are identically the broad and short descriptions described the code in the system so they should be read as categorical variables. Thus, I used `as.factor()` to convert them.
- **Crm Cd 1**, **Crm Cd 2**, **Crm Cd 3** : They are all a four-digit integer code indicating the type of crime. Thus, they should be read as integer so I used `as.integer()` to convert it.

Hide

```
crime$`AREA NAME` %>% levels()
```

```
 [1] "77th Street" "Central"     "Devonshire"  "Foothill"    "Harbor"      "Hollenbec
k"  "Hollywood"  
 [8] "Mission"     "N Hollywood" "Newton"      "Northeast"   "Olympic"     "Pacific"    
"Rampart"    
[15] "Southeast"   "Southwest"   "Topanga"     "Van Nuys"    "West LA"     "West Vall
ey" "Wilshire"   
```

Hide

```
crime$`Vict Sex` %>% levels()
```

```
[1] "Male"    "Female"  "Unknown"
```

I rechecked some converted fields in my dataset using `levels()` function. However, there is no ordinal variable nor logical variable in my selected dataset.

Subsetting

Hide

```
New_Matrix <- crime %>% slice_head(.,n=10) %>% as.matrix()
class(New_Matrix)
```

```
[1] "matrix" "array" 
```

Hide

```
typeof(New_Matrix)
```

```
[1] "character"
```

I used the `pipe operator` to subset my data as the following steps:

1. I tried to use `slice_head()` function in from *`dplyr`* after finding ways to see the tops of the data apart from using `head()` . After reading the blog post from Moe(2010), I found out many ways of doing it so I chose to try `slice_head()` instead of using `head()` in this case. I put my **`crime`** dataset inside the `slice_head()` function and set `n=10` to get the first 10 rows of my data.
2. After I got my first 10 rows from my dataset using `slice_head()` , I used `as.matrix()` function to convert my dataframe to be matrix and named it **`New_Matrix`** .

3. Then, I used `class()` to make sure that after step 2, **`New_Matrix`** is a matrix. Moreover, I proceeded by using `typeof()` function to check my matrix's datatype.

It is a character because even though at first, there are various datatypes in the dataset, when converting into the matrix, all datas must be the same datatype so `as.matrix()` converted them to be all character before putting them into the matrix named **`"New_Matrix"`** .

Create a new Data Frame

Hide

```
My_df <- data.frame(StudentID = as.integer(round(runif(10)*10000)),
                    Grade = factor(sample(c("A","B","C","D"),size = 10,replace = TRU
E),levels = c("D","C","B","A"), ordered = TRUE))
```

I first created my new data frame by designing it into 2 columns which are:

- **StudentID** : This field indicates 10 random four-digit integers as a student ID of each record. I used `runif()` to generate random numbers between 0 to 1, then time 10,000 and used `round()` to get four-digit numbers. After that, I used `as.integer()` to make sure that those set of random numbers are integer. I adapted the way of using `runif()` to generate random numbers from reading from Cookbook for R website(n.d.)
- **Grade** : For this column, I designed it to represent the grade of each record. I, first, used `sample()` by determining `size=10` and `replace = TRUE` in order to get 10 random grades among A, B, C, D. Secondly, I put that function into `factor()` to factorize those random grades. In this process, I also set `levels = c("D","C","B","A")` and make this dataset an ordinal variable by set `ordered = TRUE` .

I used `data.frame()` with those two vectors to create the new dataframe and store it as **`My_df`** .

Hide

```
class(My_df$StudentID)
```

```
[1] "integer"
```

Hide

```
class(My_df$Grade)
```

```
[1] "ordered" "factor"
```

Hide

```
str(My_df)
```

```
'data.frame':	10 obs. of  2 variables:
 $ StudentID: int  1247 2051 298 3663 8878 301 9213 8945 4831 8510
 $ Grade    : Ord.factor w/ 4 levels "D"<"C"<"B"<"A": 3 4 3 2 2 1 4 1 1 1
```

I tried to recheck the datatypes of both field in my new dataframe using `class()` and `str()` . As be seen from the output, there are four levels in my ordinal variable (**Grade**) which are `"D"<"C"<"B"<"A"` .

Hide

```
weight_vect <- round(runif(10,min = 40,max = 80),digits = 2)
class(weight_vect)
```

```
[1] "numeric"
```

According to the instruction, I created a new numeric vector called `weight_vect` using the similar methodology as when creating `StudentID`, but this time, I set the range of the number between 40-80 just to make it more realistic, assuming that students weigh between 40 to 80 kilograms.

Hide

```
My_df <- cbind(My_df, weight_vect)
My_df
```

StudentID <int>	Grade <ord>	weight_vect <dbl>
1247	B	51.17
2051	A	63.30
298	B	67.90
3663	C	59.94
8878	C	53.41
301	D	59.77
9213	A	42.47
8945	D	47.12
4831	D	57.40
8510	D	75.48

1-10 of 10 rows

I combined the new numeric vector `weight_vec` with **`My_df`** using `cbind()` and get 3 variables in my new dataset **`My_df`**. I printed **`My_df`** to check the result showing that there are 3 variables in the dataset.

References

Bache S, Wickham H (2022). *magrittr: A Forward-Pipe Operator for R*. R package version 2.0.3, https://CRAN.R-project.org/package=magrittr (https://CRAN.R-project.org/package=magrittr).

Data.lacity.org (Department LAP) (2024) *Crime Data from 2020 to Present* [data set], Data.lacity.org website, accessed 18 March 2024. https://data.lacity.org/Public-Safety/Crime-Data-from-2020-to-Present/2nrs-mtv8/about_data (https://data.lacity.org/Public-Safety/Crime-Data-from-2020-to-Present/2nrs-mtv8/about_data)

Generating random numbers, (n.d.) Cookbook for R website, accessed 20 March 2024. http://www.cookbook-r.com/Numbers/Generating_random_numbers/ (http://www.cookbook-r.com/Numbers/Generating_random_numbers/)

Indigo (16 September 2011) ‘Changing date format in R’, *Stackoverflow*, accessed 19 March 2024. https://stackoverflow.com/questions/7439977/changing-date-format-in-r (https://stackoverflow.com/questions/7439977/changing-date-format-in-r)

Moe (19 April 2010) ‘Select first 4 rows of a data.frame in R’, *Stackoverflow*, accessed 19 March 2024. https://stackoverflow.com/questions/2667673/select-first-4-rows-of-a-data-frame-in-r (https://stackoverflow.com/questions/2667673/select-first-4-rows-of-a-data-frame-in-r)

R Core Team (2024). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/ (https://www.R-project.org/).

Taheri S (2024) ‘Module 2 Notes Get: Importing, Scraping and Exporting Data with R’ [Course Website], Royal Mlebourne Institute of Technology, 124 La Trobe St, Melbourne VIC 3000.

Taheri S (2024) ‘Module 3 Notes Understand: Understanding Data and Data Structures’ [Course Website], Royal Mlebourne Institute of Technology, 124 La Trobe St, Melbourne VIC 3000.

Wickham H, François R, Henry L, Müller K, Vaughan D (2023). *dplyr: A Grammar of Data Manipulation*. R package version 1.1.4, https://CRAN.R-project.org/package=dplyr (https://CRAN.R-project.org/package=dplyr).

Wickham H, Hester J, Bryan J (2024). *readr: Read Rectangular Text Data*. R package version 2.1.5, https://CRAN.R-project.org/package=readr (https://CRAN.R-project.org/package=readr).

Zhu H (2024). *kableExtra: Construct Complex Table with ‘kable’ and Pipe Syntax*. R package version 1.4.0, https://CRAN.R-project.org/package=kableExtra (https://CRAN.R-project.org/package=kableExtra).