Data Integration

Hazim Fitri

Contents

Data Integration **1**

Import data in R 1

Native data set 1

Excel file (.xlsx) 2

Integration of data with different attributes. 2

Data integration based on inconsistent attribute names with some mismatched attribute values 3

Customize data attribute 4

Edit manually 4

Remove redundant data 6

Export from R 7

Save .RData file 7

Save .csv file 7

Save .txt file 7

Data Integration

Import data in R

Native data set

Using native data from R, we can call the data by using `data()`. For example:-

```
data(iris)
```

If we wish to understand further regarding the data, we can put `?` in front of out datset. For example:-

```
?iris
```

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

Excel file (.xlsx)

In order to load .xlsx file, first we need to load library openxlsx. Then, we will be able to use function `read.xlsx()`. To see all the default value, run `?read.xlsx`

```
library(openxlsx)
```

```
## Warning: package 'openxlsx' was built under R version 4.4.2
```

```
big_mart = read.xlsx('./Data/Big Mart Dataset.xlsx', sheet = 1, startRow = 1)
```

Integration of data with different attributes.

```
mydata1 = read.table('./Data/mydata1.txt')
mydata2 = read.csv('./Data/mydata2.csv')
print(colnames(mydata1))
```

```
## [1] "ID_Person" "income"    "debt"      "child"     "car"       "saving"
```

```
print(colnames(mydata2))
```

```
## [1] "X"         "ID_Person" "Ethnic"    "Age"       "house"
```

Both data has different column names. Thus, in order to integrate both data, we can use `cbind()` function which will combine both column into one data frame.

```
mydata3 = cbind(mydata1, mydata2)
colnames(mydata3)
```

```
##  [1] "ID_Person" "income"    "debt"      "child"     "car"       "saving"
##  [7] "X"         "ID_Person" "Ethnic"    "Age"       "house"
```

If there's any unique identifier, we can use `merge()` function and it'll automatically merge combine given datasets and merge column with the same name. In this example both dataframe has column 'ID_Person'

```
mydata4 = merge(mydata1, mydata2)
mydata4
```

```
##    ID_Person   income       debt child car   saving  X Ethnic Age house
## 1         A1 37418.41 -14380.808     3   0 3185.266  1      M  20     0
## 2         A2 36773.00 -13074.371     5   2 5944.121 10      C  23     0
## 3         A3 33065.11 -11590.271     3   2 1522.749  5      M  32     1
## 4         A4 21967.69  -5754.715     2   0 3370.326  8      I  28     0
## 5         B1 28495.61  -6944.777     7   2 2763.576  4      I  34     2
## 6         B2 24000.85  -7728.856     2   3 3103.766  9      M  30     0
## 7         B3 37149.13  -4609.790     5   2 3279.630  3      C  28     0
## 8         B4 33728.65  -7519.012    10   4 3930.530 11      M  45     4
## 9         C1 22022.83  -9009.379     5   1 4149.976 12      C  22     1
## 10        C2 36628.36  -6748.113     4   0 3087.376  6      M  40     0
## 11        C3 27148.07 -10415.858     6   2 3026.048  7      I  27     1
## 12        C4 25516.84  -7640.781     8   0 2831.357  2      M  25     2
```

However, if the column name is different for both data sets, we can specify by using parameter `by.x` and `by.y`

```
load('./Data/mydata4.RData')
colnames(mydata4)
```

```
## [1] "ID"     "year"   "income" "debt"   "child"  "car"    "saving"
```

```
mydata5 = read.csv('./Data/mydata5.csv')
colnames(mydata5)
```

```
## [1] "X"        "IDPerson" "Ethnic"   "Age"      "house"
```

```
mydata6 = merge(mydata4, mydata5, by.x='ID', by.y='IDPerson')
mydata6
```

```
##    ID year   income       debt child car   saving  X Ethnic Age house
## 1  A1 2000 26571.57  -7241.077    11   2 2089.584  5      M  32     0
## 2  A2 2001 33704.83  -5943.100     6   4 3760.962  1      M  20     1
## 3  A3 2000 39032.54  -7839.178     6   0 3743.885  7      I  27     0
## 4  A4 2003 22384.77  -9563.551     5   1 2993.336 10      C  23     3
## 5  B1 2003 33299.36  -6907.920     3   1 3446.248  2      M  25     0
## 6  B2 2000 30189.28  -7952.130     4   3 2580.604  3      C  28     1
## 7  B3 2002 44179.24  -7543.476     4   2 2463.710 12      C  22     0
## 8  B4 2002 26664.21 -10509.667     4   2 3434.909 11      M  45     0
## 9  C1 2003 35301.69  -5320.218     5   0 4541.238  4      I  34     1
## 10 C2 2001 34473.69  -9814.277     6   0 3731.690  9      M  30     0
## 11 C3 2002 31305.11  -5287.739     2   6 3383.971  6      M  40     1
```

Data integration based on inconsistent attribute names with some mismatched attribute values

It will remove all the row without matching value.

```
mydata7 = mydata5[1:10, ]
mydata8 = merge(mydata4, mydata7, by.x='ID', by.y='IDPerson')
mydata8
```

```
##   ID year   income      debt child car   saving  X Ethnic Age house
## 1 A1 2000 26571.57 -7241.077    11   2 2089.584  5      M  32     0
## 2 A2 2001 33704.83 -5943.100     6   4 3760.962  1      M  20     1
## 3 A3 2000 39032.54 -7839.178     6   0 3743.885  7      I  27     0
## 4 A4 2003 22384.77 -9563.551     5   1 2993.336 10      C  23     3
## 5 B1 2003 33299.36 -6907.920     3   1 3446.248  2      M  25     0
## 6 B2 2000 30189.28 -7952.130     4   3 2580.604  3      C  28     1
## 7 C1 2003 35301.69 -5320.218     5   0 4541.238  4      I  34     1
## 8 C2 2001 34473.69 -9814.277     6   0 3731.690  9      M  30     0
## 9 C3 2002 31305.11 -5287.739     2   6 3383.971  6      M  40     1
```

However, we can still retain the unmatch data by adding argument `all=T`

```r
mydata9 = merge(mydata4, mydata7, by.x='ID', by.y='IDPerson', all=T)
colnames(mydata9)
```

```
##  [1] "ID"     "year"   "income" "debt"   "child"  "car"    "saving" "X"
##  [9] "Ethnic" "Age"    "house"
```

Alternative way is we can rename the column before merge

```r
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 4.4.2
```

```r
mydata10 = rename(mydata9, c('ID'='Nombor ID', 'house'='Number of House'))
```

Customize data attribute

Edit manually

If we wish to customize our data, the most basic way to do it is by using `edit()` function to manually edit the value one by one.

```r
mydata11 = edit(mydata10)
```

Edit data

```r
datam1 = read.csv('./Data/dataM1.csv')
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.4.2
```

```
## Warning: package 'readr' was built under R version 4.4.2
```

```
## Warning: package 'forcats' was built under R version 4.4.2
```

```
## Warning: package 'lubridate' was built under R version 4.4.2
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::arrange()   masks plyr::arrange()
## x purrr::compact()   masks plyr::compact()
## x dplyr::count()     masks plyr::count()
## x dplyr::desc()      masks plyr::desc()
## x dplyr::failwith()  masks plyr::failwith()
## x dplyr::filter()    masks stats::filter()
```

```
## x dplyr::id()        masks plyr::id()
## x dplyr::lag()       masks stats::lag()
## x dplyr::mutate()    masks plyr::mutate()
## x dplyr::rename()    masks plyr::rename()
## x dplyr::summarise() masks plyr::summarise()
## x dplyr::summarize() masks plyr::summarize()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
city_name = function(city) {
  city = tolower(city) #convert all letter to small letter
  city = trimws(city) #trim all white space
  city = gsub('+', '', city) #replace with 1 space
  city = tools::toTitleCase(city) #change format to title case
  return(city)
}

a = tolower(datam1$City)
b = trimws(a)
c = ?gsub('+', '', b)
d = tools::toTitleCase(c)
```

Edit inconsistent data

(E.g., shortform)

```
datam2 = read.csv('./Data/dataM2.csv')
head(datam2)
```

```
##   X ID    Name        City
## 1 1  1   Alice          NY
## 2 2  2     Bob Los Angeles
## 3 3  3 Charlie         CHI
## 4 4  4   David    New York
## 5 5  5     Eva          LA
## 6 6  6   Frank     Chicago
```

First, look at the unique data of city column

```
unique(datam2$City)
```

```
## [1] "NY"          "Los Angeles" "CHI"         "New York"    "LA"
## [6] "Chicago"
```

Map all short form to the full name. Then, create a function to convert the short form to the full name.

```
city_map = list('NY'='New York', 'CHI'='Chicago', 'LA'='Los Angeles')
std_city = function(city){
  if(city%in%names(city_map)) {
    return(city_map[[city]])
  } else {
    return(city)
  }
```

```
}

datam2$City = sapply(datam2$City, std_city)

unique(datam2$City)
```

```
## [1] "New York"    "Los Angeles" "Chicago"
```

Remove redundant data

```
datam3 = read.csv('./Data/dataM3.csv',sep=';')
datam3
```

```
##     id    name age gender income
## 1  101   Alice  63      F  78109
## 2  102     Bob  60      M  63755
## 3  103 Charlie  33      F  70577
## 4  104   David  53      M  44231
## 5  105     Eva  36      F  55489
## 6  102     Bob  60      M  63755
## 7  107     Mat  33      M  30148
## 8  108     Ali  59      M  59670
## 9  105     Eva  36      F  55489
## 10 109     Bob  40      M  53308
```

```
library(dplyr)
duplicated(datam3) #check for duplicated data
```

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

```
distinct(datam3) #return only unique/distinct data
```

```
##    id    name age gender income
## 1 101   Alice  63      F  78109
## 2 102     Bob  60      M  63755
## 3 103 Charlie  33      F  70577
## 4 104   David  53      M  44231
## 5 105     Eva  36      F  55489
## 6 107     Mat  33      M  30148
## 7 108     Ali  59      M  59670
## 8 109     Bob  40      M  53308
```

```
datam3 %>% distinct(datam3$id, .keep_all=T) #.keep_all is to keep all column
```

```
##    id    name age gender income datam3$id
## 1 101   Alice  63      F  78109       101
## 2 102     Bob  60      M  63755       102
## 3 103 Charlie  33      F  70577       103
```

```
## 4 104   David  53      M  44231       104
## 5 105     Eva  36      F  55489       105
## 6 107     Mat  33      M  30148       107
## 7 108     Ali  59      M  59670       108
## 8 109     Bob  40      M  53308       109
```

Export from R

In order to export file from R, first thing that we need to know is where the will be exported to. To see the save filepath, we can use `getwd()` to see the directory

```
getwd()
```

```
## [1] "D:/Data-Mining/03. Data Integration"
```

If we wish to change the save location, we can redefine our location by using setwd() function and put the path as argument

```
getwd()
```

```
## [1] "D:/Data-Mining/03. Data Integration"
```

Save .RData file

Now, we can proceed to save our very first file in our local storage. To save R file we can simply use:

```
save(datam3, file='data_M3.RData')
```

Save .csv file

```
write.csv(datam3, file='data_M3.csv')
```

Save .txt file

```
write.table(datam3, file='data_M3.txt', sep='\t')
```