Data Manipulation and Transformation

Hamid Abdulsalam

Data Manipulation Using tidyr

Data

The data file for this session has been provided in the data folder, it's named *mbta*. It is a data on passengers boarding and alighting at all stations on all lines of the Massachusetts Bay Transportation Authority (MBTA) commuter rail system

Package

For this lesson, we will be using the tidyverse package. Tidyverse is a collection of essential R packages for data science created by Hadley Wickham.

The following packages are included in the core tidyverse: ggplot2, dplyr, tidyr, readr, purrr , tibble, stringr and forcats.

You can install the tidyverse package by running the following code

```
## install.packages("tidyverse")
```

Loading the Data

```
library(readxl)
library(tidyverse)
dta<-read_excel("data/mbta.xlsx",skip = 1,
                range = cell_cols(2:60))
dta[1:4,]
```

```
## # A tibble: 4 x 59
##   mode  `2007-01` `2007-02` `2007-03` `2007-04` `2007-05
##   <chr> <chr>     <chr>          <dbl> <chr>     <chr>
## 1 All ~ NA        NA              1188. NA        NA
## 2 Boat  4         3.6               40  4.3       4.9
## 3 Bus   335.819   338.675          340. 352.162   354.367
## 4 Comm~ 142.2     138.5            138. 139.5     139
## # ... with 51 more variables: `2007-08` <chr>, `2007-09`
## #   `2007-10` <chr>, `2007-11` <chr>, `2007-12` <dbl>,
```

Combining the Years

The **gather** function in tidyr package helps in gathering multiple columns and collapses the columns into key-value pairs as seen below.

```
dta_1 <- dta %>% gather(`2007-01`:`2011-10`,
                key = "year", value = "passengers")
dta_1[1:4,]
```

```
## # A tibble: 4 x 3
##   mode                year    passengers
##   <chr>               <chr>   <chr>
## 1 All Modes by Qtr    2007-01 NA
## 2 Boat                2007-01 4
## 3 Bus                 2007-01 335.819
## 4 Commuter Rail       2007-01 142.2
```

Separating Year in "Year" and "Month"

The **separate** function in tidyr turns a single character column into multiple columns as seen below

```
dta_2 <- dta_1 %>% separate(year,
                 into = c("year", "month"))
dta_2[1:4,]
```

```
## # A tibble: 4 x 4
##   mode                year  month passengers
##   <chr>               <chr> <chr> <chr>
## 1 All Modes by Qtr    2007  01    NA
## 2 Boat                2007  01    4
## 3 Bus                 2007  01    335.819
## 4 Commuter Rail       2007  01    142.2
```

Spread function

The spread function helps in spreading a key-value pair across multiple columns

```
dta_3 <- dta_2 %>% spread(mode, passengers)
dta_3[1:4,]
```

```
## # A tibble: 4 x 13
##   year  month `All Modes by Q~ Boat  Bus   `Commuter Rai
##   <chr> <chr> <chr>           <chr> <chr> <chr>
## 1 2007  01    NA              4     335.~ 142.2
## 2 2007  02    NA              3.6   338.~ 138.5
## 3 2007  03    1187.653        40    339.~ 137.7
## 4 2007  04    NA              4.3   352.~ 139.5
## # ... with 6 more variables: `Light Rail` <chr>, `Pct Ch
## #   Bus` <chr>, RIDE <chr>, TOTAL <chr>, `Trackless Trol
```

Extracting the needed columns

We may be interested in certain columns, we can apply our knowledge of subsetting to select the needed columns for our analysis

```
dta_4 <- dta_3%>% .[,c(1:2,6:8)]
dta_4[1:4,]
```

```
## # A tibble: 4 x 5
##   year  month `Commuter Rail` `Heavy Rail` `Light Rail`
##   <chr> <chr> <chr>           <chr>        <chr>
## 1 2007  01    142.2           435.294      227.231
## 2 2007  02    138.5           448.271      240.262
## 3 2007  03    137.7           458.583      241.444
## 4 2007  04    139.5           472.201      255.557
```

Using the Gather Function

After successful selecting the columns we are interested in, then we gather columns into a single column using the gather function as seen below.

```
dta_5 <- dta_4 %>% gather(`Commuter Rail`:`Light Rail`,
key="rail_type", value = passengers)
dta_5[1:4,]
```

```
## # A tibble: 4 x 4
##   year  month rail_type     passengers
##   <chr> <chr> <chr>         <chr>
## 1 2007  01    Commuter Rail 142.2
## 2 2007  02    Commuter Rail 138.5
## 3 2007  03    Commuter Rail 137.7
## 4 2007  04    Commuter Rail 139.5
```

Data Transformation with Dplyr

The Data

For data transformation, we will be using hflights data , the dataset contains all flights departing from Houston airports IAH (George Bush Intercontinental) and HOU (Houston Hobby)

```
# install.packages("hflights")
library(hflights)
data(hflights)
hflights[1:4,1:4]
```

```
##      Year Month DayofMonth DayOfWeek
## 5424 2011     1          1         6
## 5425 2011     1          2         7
## 5426 2011     1          3         1
## 5427 2011     1          4         2
```

Some functions in dplyr

- filter(): Extracting rows by the rows values
- arrange(): arranging rows
- select(): selecting columns
- mutate(): creating new variables from existing variables
- summarize(): Obtaining summary statistics of variables
- group_by(): convert existing tables into a grouped table

In dplyr functions,the first argument is always data frame and the returned value is always a data frame as well.

filter()

The filter() is used to choose rows/cases where conditions are true, basically used in subsetting a dataframe based on their row values. Let's select all flights of February, 2011.

```
data("hflights")
f1<-filter(hflights, Year == 2011, Month == 2)
f1[1:4, 1:4]
```

```
##   Year Month DayofMonth DayOfWeek
## 1 2011     2          1         2
## 2 2011     2          2         3
## 3 2011     2          3         4
## 4 2011     2          4         5
```

filter()

Let's select all flights that departed from Las Vegas and Boston ("BOS" "LAS")

```
f2<-filter(hflights, Dest %in% c("BOS" ,"LAS"))
f2[1:4,12:15]
```

```
##   ArrDelay DepDelay Origin Dest
## 1        3        0    IAH  LAS
## 2        4        0    IAH  BOS
## 3       13       11    IAH  LAS
## 4       -5       -3    IAH  BOS
```

Using between in the filter()function

We can also use between to specify the particular range of values we are interested in. It takes the following form between(x, left, right) which is equivalent to x >= left & x <= right. Let's filter all flights that covered distance between 224 and 944 miles

```
f3<-filter(hflights, between(Distance, 224,944))
f3[1:6,13:16]
```

```
##   DepDelay Origin Dest Distance
## 1        0    IAH  DFW      224
## 2        1    IAH  DFW      224
## 3       -8    IAH  DFW      224
## 4        3    IAH  DFW      224
## 5        5    IAH  DFW      224
## 6       -1    IAH  DFW      224
```

Excercise

Find all flights that a. Departed in April, 2011 b. Operated by AA and WN (Hint: Use the UniqueCarrier variable)

arrange()

arrange() function is used to order a dataframe by a set of columns. Let's arrange the flights data by Year, Month

```
arr<-arrange(hflights, Year, Month)
arr[1:6, 1:6]
```

```
##   Year Month DayofMonth DayOfWeek DepTime ArrTime
## 1 2011     1          1         6    1400    1500
## 2 2011     1          2         7    1401    1501
## 3 2011     1          3         1    1352    1502
## 4 2011     1          4         2    1403    1513
## 5 2011     1          5         3    1405    1507
## 6 2011     1          6         4    1359    1503
```

arrange()

By dafault arrange() sorts values in ascending order. Use desc() to re-order by a column in descending order.

```
arr1<-arrange(hflights, desc(DepTime))
arr1[1:6, 1:6]
```

```
##   Year Month DayofMonth DayOfWeek DepTime ArrTime
## 1 2011     5         24         2    2400     144
## 2 2011     4          8         5    2359     455
## 3 2011     5         20         5    2359     130
## 4 2011     5         20         5    2359      56
## 5 2011     6         22         3    2359     113
## 6 2011     6         10         5    2359      40
```

select()

select() can be used to extract variables from a dataframe.

```
sel<-select(hflights, Year, Month, FlightNum, AirTime)
sel[1:4,]
```

```
##      Year Month FlightNum AirTime
## 5424 2011     1       428      40
## 5425 2011     1       428      45
## 5426 2011     1       428      48
## 5427 2011     1       428      39
```

Helper Functions for Select

select() has various helper functions: + everything(): selects all variables. + starts_with("def"): matches names that begin with "def". + ends_with("xyz"): matches names that end with "xyz". + contains("ijk"): matches names that contain "ijk"

Select()

We can select variables that starts with Dep and Arr

```
sel_2<-select(hflights, starts_with("Dep"),
              starts_with("Arr"))
sel_2[1:4,]
```

```
##      DepTime DepDelay ArrTime ArrDelay
## 5424    1400        0    1500      -10
## 5425    1401        1    1501       -9
## 5426    1352       -8    1502       -8
## 5427    1403        3    1513        3
```

mutate()

mutate() adds new variables using the existing ones, it also preserves existing variables.

```
m<-hflights %>%
select(ends_with("Delay"), Distance, AirTime) %>%
mutate(time_gain = ArrDelay - DepDelay,
speed = Distance / AirTime * 60
)
m[1:4,1:6]
```

```
##   ArrDelay DepDelay Distance AirTime time_gain    speed
## 1      -10        0      224      40       -10 336.0000
## 2       -9        1      224      45       -10 298.6667
## 3       -8       -8      224      48         0 280.0000
## 4        3        3      224      39         0 344.6154
```

summarize()

summarize() function creates one or more scalar variables summarizing the variables of an existing table.

```
summarise(hflights, Delay = sum(DepDelay, na.rm = TRUE))
```

```
##     Delay
## 1 2121251
```

summarize() with group_by()

summarize() with group_by() will result in one row in the output for each group. Let's summarize average delay by carrier

```
hflights %>%
group_by(UniqueCarrier) %>%
summarise(delay = mean(DepDelay, na.rm = TRUE))
```

```
## `summarise()` ungrouping output (override with `.groups`
```

```
## # A tibble: 15 x 2
##    UniqueCarrier delay
##    <chr>         <dbl>
##  1 AA             6.39
##  2 AS             3.71
##  3 B6            13.3
##  4 CO             9.26
```

Count with summarize()

For aggregations it is generally a good idea to include a count n(). For example, let's group the Carrier based on the total number of departure delay

```
hflights %>%
group_by(UniqueCarrier) %>%
summarise(DepDelay =n())
```

```
## `summarise()` ungrouping output (override with `.groups`
```

```
## # A tibble: 15 x 2
##    UniqueCarrier DepDelay
##    <chr>            <int>
##  1 AA                3244
##  2 AS                 365
##  3 B6                 695
```

Other Useful functions for Summarize()

- Measures of location: mean(x), sum(x), median(x).
- Measures of spread: sd(x), IQR(x), mad(x).
- Measures of rank: min(x), quantile(x, 0.25), max(x).
- Measures of position: first(x), nth(x, 2), last(x).
- Counts: n().

END