E-Content on Data Manipulation and Visualization through Statistical Software R (Open Source Software)

Dr Gurpreet Singh Tuteja and Dr Dhiraj Kumar Singh
Department of Mathematics
Zakir Husain Delhi College
(University of Delhi)
Jawaharlal Nehru Marg, Delhi - 110002

Day 3: July 29, 2020

1 Session 3

1.1 dplyr Package

Today, we will be learning grammar of manipulations.It provides a consistent set of verbs that help you solve the most common data manipulation challenges:

mutate() adds new variables that are functions of existing variables

select() picks variables based on their names.

filter() picks cases based on their values.

summarise() reduces multiple values down to a single summary.

arrange() changes the ordering of the rows.

%>% the "pipe" operator is used to connect multiple verb actions together into a pipeline, also can be seen as a concept of subset in set theory. The shortcut key for getting this operator is CTRL+SHIFT+M.

The easiest way to get dplyr is to install the whole tidyverse:
install.packages("tidyverse")

Alternatively, install just dplyr:
install.packages("dplyr")
Common dplyr Function Properties All of the above functions we have a few common characteristics. In particular,

1. The first argument is a data frame

2. The subsequent arguments describe what to do with the data frame specified in the first argument, and you can refer to columns in the data frame directly without using the $ operator (just use the column names).

3. The return result of a function is a new data frame.

4. Data frames must be properly formatted In short, there should be one observation per row, and each column should represent a feature or characteristic of that observation.

1.2 Filter

The filter() function subsets the rows with multiple conditions on different criteria.
Lets begin,consider the built-in data file known as 'iris'. The structure can be observed using :
str(iris)

```
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

The complete records and variables can be viewed using : View(iris)

Filter

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa

Showing 1 to 13 of 150 entries, 5 total columns

The names of the variables or fields or columns can be seen using:
names(iris) The output is :
names(iris)
"Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"

Following command segregates all species which are labelled 'setosa'.

setosa ← filter(iris,Species=="setosa")

This seggragates all species which are labelled 'setosa' and further whose Sepal.Length is greater than 5.5 and Sepal.Width is greater than 4.

iris % > % filter(Species=="setosa") % > % filter(Sepal.Length>5.5) % > % filter(Sepal.Width>4)

Some of the data files are built-in and can be used any moment but some files need to be installed. Consider a data file nycflights13 which need be installed.

install.packages("nycflights13")

Now, this file need be loaded using the command which you have used for loading the package dplyr.

library(nycflights13)
str(flights)

```
tibble [336,776 x 19] (S3: tbl_df/tbl/data.frame)
 $ year          : int [1:336776] 2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
 $ month         : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
 $ day           : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
 $ dep_time      : int [1:336776] 517 533 542 544 554 554 555 557 557 558 ...
 $ sched_dep_time: int [1:336776] 515 529 540 545 600 558 600 600 600 600 ...
 $ dep_delay     : num [1:336776] 2 4 2 -1 -6 -4 -5 -3 -3 -2 ...
 $ arr_time      : int [1:336776] 830 850 923 1004 812 740 913 709 838 753 ...
 $ sched_arr_time: int [1:336776] 819 830 850 1022 837 728 854 723 846 745 ...
 $ arr_delay     : num [1:336776] 11 20 33 -18 -25 12 19 -14 -8 8 ...
 $ carrier       : chr [1:336776] "UA" "UA" "AA" "B6" ...
 $ flight        : int [1:336776] 1545 1714 1141 725 461 1696 507 5708 79 301 ...
 $ tailnum       : chr [1:336776] "N14228" "N24211" "N619AA" "N804JB" ...
 $ origin        : chr [1:336776] "EWR" "LGA" "JFK" "JFK" ...
 $ dest          : chr [1:336776] "IAH" "IAH" "MIA" "BQN" ...
 $ air_time      : num [1:336776] 227 227 160 183 116 150 158 53 140 138 ...
 $ distance      : num [1:336776] 1400 1416 1089 1576 762 ...
 $ hour          : num [1:336776] 5 5 5 5 6 5 6 6 6 6 ...
 $ minute        : num [1:336776] 15 29 40 45 0 58 0 0 0 0 ...
 $ time_hour     : POSIXct[1:336776], format: "2013-01-01 05:00:00" "2013-01-01 05:00:00" "2013-01-0
1 05:00:00" ...
```

View(flights)

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier
1	2013	7	27	NA	106	NA	NA	245	NA	US
2	2013	1	2	458	500	-2	703	650	13	US
3	2013	1	3	458	500	-2	650	650	0	US
4	2013	1	4	456	500	-4	631	650	-19	US
5	2013	1	5	458	500	-2	640	650	-10	US
6	2013	1	6	458	500	-2	718	650	28	US
7	2013	1	7	454	500	-6	637	648	-11	US
8	2013	1	8	454	500	-6	625	648	-23	US
9	2013	1	9	457	500	-3	647	648	-1	US
10	2013	1	10	450	500	-10	634	648	-14	US
11	2013	1	11	453	500	7	643	648	5	US

Showing 1 to 11 of 336,776 entries, 19 total columns

View(airports)

	faa	name	lat	lon	alt	tz	dst	tzone
1	04G	Lansdowne Airport	41.13047	-80.61958	1044	-5	A	America/New_York
2	06A	Moton Field Municipal Airport	32.46057	-85.68003	264	-6	A	America/Chicago
3	06C	Schaumburg Regional	41.98934	-88.10124	801	-6	A	America/Chicago
4	06N	Randall Airport	41.43191	-74.39156	523	-5	A	America/New_York
5	09J	Jekyll Island Airport	31.07447	-81.42778	11	-5	A	America/New_York
6	0A9	Elizabethton Municipal Airport	36.37122	-82.17342	1593	-5	A	America/New_York
7	0G6	Williams County Airport	41.46731	-84.50678	730	-5	A	America/New_York
8	0G7	Finger Lakes Regional Airport	42.88356	-76.78123	492	-5	A	America/New_York
9	0P2	Shoestring Aviation Airfield	39.79482	-76.64719	1000	-5	U	America/New_York
10	0S9	Jefferson County Intl	48.05381	-122.81064	108	-8	A	America/Los_Angeles
11	0W3	Harford County Airport	39.56684	-76.20240	409	-5	A	America/New_York

Showing 1 to 12 of 1,458 entries, 8 total columns

Head Function in R: returns the first n rows of a matrix or data frame in R
Tail Function in R: returns the last n rows of a matrix or data frame in R
head(flights)

```
> head(flights)
# A tibble: 6 x 19
   year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier
  <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>
1  2013     1     1      517            515         2      830            819        11 UA
2  2013     1     1      533            529         4      850            830        20 UA
3  2013     1     1      542            540         2      923            850        33 AA
4  2013     1     1      544            545        -1     1004           1022       -18 B6
5  2013     1     1      554            600        -6      812            837       -25 DL
6  2013     1     1      554            558        -4      740            728        12 UA
# ... with 9 more variables: flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
#   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

by default it displays a tibble of top 6 rows.

Similarly, tail function displays bottom 6 rows
tail(flights)

```
> tail(flights)
# A tibble: 6 x 19
   year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier
  <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>
1  2013     9    30       NA           1842        NA       NA           2019        NA EV
2  2013     9    30       NA           1455        NA       NA           1634        NA 9E
3  2013     9    30       NA           2200        NA       NA           2312        NA 9E
4  2013     9    30       NA           1210        NA       NA           1330        NA MQ
5  2013     9    30       NA           1159        NA       NA           1344        NA MQ
6  2013     9    30       NA            840        NA       NA           1020        NA MQ
# ... with 9 more variables: flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
#   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

sample_n(x,n) function is used to take random sample specimens from a data frame, where
x: Data Frame
n: size/number of items to select
sample_n(flights,5)

We can omit dataframe parameter, using piping command:
iris % > % sample_n(5)

Let us practice the filter command and see how we can use it for practical purposes:

This will filter all flights on first day of January.
jan1 ← filter(flights,month==1,day==1)
View(jan1)

Filter

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier	fligh
1	2013	1	1	517	515	2	830	819	11	UA	
2	2013	1	1	533	529	4	850	830	20	UA	
3	2013	1	1	542	540	2	923	850	33	AA	
4	2013	1	1	544	545	-1	1004	1022	-18	B6	
5	2013	1	1	554	600	-6	812	837	-25	DL	
6	2013	1	1	554	558	-4	740	728	12	UA	
7	2013	1	1	555	600	-5	913	854	19	B6	
8	2013	1	1	557	600	-3	709	723	-14	EV	
9	2013	1	1	557	600	-3	838	846	-8	B6	
10	2013	1	1	558	600	-2	753	745	8	AA	
11	2013	1	1	558	600	-2	849	851	-2	B6	

Showing 1 to 11 of 842 entries, 19 total columns

Following will segregate all flights flown in November and December.
nov_dec ← filter(flights,month View(nov_dec)

Filter

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier	fligh
1	2013	11	1	5	2359	6	352	345	7	B6	
2	2013	11	1	35	2250	105	123	2356	87	B6	
3	2013	11	1	455	500	-5	641	651	-10	US	
4	2013	11	1	539	545	-6	856	827	29	UA	
5	2013	11	1	542	545	-3	831	855	-24	AA	
6	2013	11	1	549	600	-11	912	923	-11	UA	
7	2013	11	1	550	600	-10	705	659	6	US	
8	2013	11	1	554	600	-6	659	701	-2	US	
9	2013	11	1	554	600	-6	826	827	-1	DL	
10	2013	11	1	554	600	-6	749	751	-2	DL	
11	2013	11	1	555	600	-5	847	854	-7	B6	

Showing 1 to 11 of 55,403 entries, 19 total columns

In case, you wish to see flights in November or December then
View(flights All flights that had an arrival delay of 2 or more hours
filter(flights,arr_delay¿=120) % > % nrow()

Count all those flights whose carrier is prefixed by "DL"
filter(flights,carrier=="DL") % > % nrow()
The output is:
[1]48110

Displays first ten flights:
slice(flights,1:10)

displays all records between January and March:
View(filter(flights,between(month,1,3)))

Displays all records where departure time is NA
filter(flights,is.na(dep_time)) % > % View()

1.3 Select()

Records with only those fields which are prefixed by 'Sepal'
select(iris,starts_with"Sepal"))

Records with only those fields which ends with 'Sepal'
select(iris,ends_with"Width"))

1.4 Mutate and Transmute

The mutate() function is a function for creating new variables. For the use of mutate() function, you need to specify following three things:

1. The name of the dataframe you want to modify.
2. The name of the new variable that you'll create.
3. The value you will assign to the new variable.

a ← mutate(iris,$Sepal = (Sepal.Length + Sepal.Width)/2$)

Filter

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species	Sepal
1	5.1	3.5	1.4	0.2	setosa	4.30
2	4.9	3.0	1.4	0.2	setosa	3.95
3	4.7	3.2	1.3	0.2	setosa	3.95
4	4.6	3.1	1.5	0.2	setosa	3.85
5	5.0	3.6	1.4	0.2	setosa	4.30
6	5.4	3.9	1.7	0.4	setosa	4.65
7	4.6	3.4	1.4	0.3	setosa	4.00
8	5.0	3.4	1.5	0.2	setosa	4.20
9	4.4	2.9	1.4	0.2	setosa	3.65
10	4.9	3.1	1.5	0.1	setosa	4.00
11	5.4	3.7	1.5	0.2	setosa	4.55

Showing 1 to 12 of 150 entries, 6 total columns

transmute()
The function mutate() compute and add new variables into a data table or dataframe. It preserves existing variables while transmute() compute new columns and drops existing variables.
transmute(iris,Sepal.Length,Sepal=(Sepal.Length+Sepal.Width)/2) % > % View()

	Sepal.Length	Sepal
1	5.1	4.30
2	4.9	3.95
3	4.7	3.95
4	4.6	3.85
5	5.0	4.30
6	5.4	4.65
7	4.6	4.00
8	5.0	4.20
9	4.4	3.65
10	4.9	4.00
11	5.4	4.55

Showing 1 to 12 of 150 entries, 2 total columns

Let us practice the above commands:
mutate(flights,speed=distance/air_time *60) transmute(flights,flight,tailnum,speed=distance/air_time*60)

1.5 Group_by) and Summarize()

You can create subtotals by combining the group_by() function and the summarise() function. Let's start with an example, where we compute the mean delaytime.
summarize(flights, delayflight, mean(dep_delay,na.rm=TRUE))
Consider the following example where we group the flights by destination and then summarize the mean delay at a particular destination.
by_dest ← group_by(flights,dest)
delay ← summarize(by_dest,
count=n(),
delay=mean(arr_delay,na.rm=TRUE))
View(delay)

Consider another example where we group the iris data with respect to the Species and then find mean of sepal length.
iris % > % group_by(Species) % > % summarize(mean(Sepal.Length)) arrange(iris,-desc(Sepal.Length),Sepal.Width) % > % View()

Species ‘mean(Sepal.Length)‘
1 setosa 5.01
2 versicolor 5.94
3 virginica 6.59
¿

1.6 Four different types of Join

Finally, we learn here different types of Join.
There are four types of join: join: full_join, left_join, right_join, inner_join
Consider following two dataframes:
data1 ¡- data.frame(ID=1:2,X1=c(”a1”,”a2”))
data2 ¡- data.frame(ID=2:3,X2=c(”b1”,”b2”))

Full or Outer Join To keep all rows from both data frames.

Natural or Inner Join To keep only rows that match from the data frames.

Left outer or Left Join To include all the rows of your data frame x and only those from y that match.

Right outer or Right Join To include all the rows of your data frame y and only those from x that match.

full_join(data1,data2) % > % View()
left_join(data1,data2) % > % View()
right_join(data1,data2) % > % View()
inner_join(data1,data2) % > % View().

That finishes the session 3. The link for Quiz is already available in the message box, kindly complete it in next 15 mins.