

Data Merging:

MERGING DATA

Merging data:

When combining separate dataframes, (in the R programming language), into a single dataframe, using the `cbind()` function usually requires use of the “`Match()`” function. To simulate the database joining functionality in SQL, the “`Merge()`” function in R accomplishes dataframe merging with the following protocols;

- “Inner Join” where the left table has matching rows from one, or more, key variables from the right table.
- “Outer Join” where all the rows from both tables are joined.
- “Left Join” where all rows from the left table, and any rows with matching keys from the right table are returned.
- “Right Join” where all rows from the right table, and any rows with matching keys from the left table are returned.

EXERCISES

- Practice the exercises explained in the class room (Follow the lecture slides) first and solve the exercises given below later:

Exercise 1

Create the dataframes to merge:

```
buildings <- data.frame(location=c(1, 2, 3), name=c("building1", "building2",  
"building3"))
```

```
data <- data.frame(survey=c(1,1,1,2,2,2), location=c(1,2,3,2,3,1),  
efficiency=c(51,64,70,71,80,58))
```

The dataframes, buildings and data have a common key variable called, “location”. Use the merge() function to merge the two dataframes by “location”, into a new dataframe, “buildingStats”.

Exercise 2

Give the dataframes different key variable names:

```
buildings <- data.frame(location=c(1, 2, 3), name=c("building1", "building2",  
"building3"))
```

```
data <- data.frame(survey=c(1,1,1,2,2,2), LocationID=c(1,2,3,2,3,1),  
efficiency=c(51,64,70,71,80,58))
```

The dataframes, buildings and data now have corresponding variables called, location, and LocationID. Use the merge() function to merge the columns of the two dataframes by the corresponding variables.

Exercise 3

Inner Join:

The R merge() function automatically joins the frames by common variable names. In that case, demonstrate how you would perform the merge in Exercise 1 without specifying the key variable.

Exercise 4

Outer Join:

Merge the two dataframes from Exercise 1. Use the “all=” parameter in the merge() function to return all records from both tables. Also, merge with the key variable, “location”.

Exercise 5

Left Join:

Merge the two dataframes from Exercise 1, and return all rows from the left table. Specify the matching key from Exercise 1.

Exercise 6

Right Join:

Merge the two dataframes from Exercise 1, and return all rows from the right table. Use the matching key from Exercise 1 to return matching rows from the left table.

Exercise 7

Cross Join:

Merge the two dataframes from Exercise 1, into a “Cross Join” with each row of “buildings” matched to each row of “data”. What new column names are created in “buildingStats”?

Exercise 8

Merging Dataframe rows:

To join two data frames (datasets) vertically, use the rbind function. The two data frames must have the same variables, but they do not have to be in the same order.

Merge the rows of the following two dataframes:

```
buildings <- data.frame(location=c(1, 2, 3), name=c("building1", "building2", "building3"))
```

```
buildings2 <- data.frame(location=c(5, 4, 6), name=c("building5", "building4", "building6"))
```

Also, specify a new dataframe, “allBuidings”.

- Apply different join operations on the tables given below. Write the expected outputs and compare them with the outputs obtained by R commands

Super Heroes

Name	Alignment	Gender	Publisher
Magneto	bad	male	Marvel
Storm	good	female	Marvel
Mystique	bad	female	Marvel
Batman	good	male	DC
Joker	bad	male	DC
Catwoman	bad	female	DC
Hellboy	good	male	Dark Horse Comics

Publishers

publisher	yr_founded
DC	1934
Marvel	1939
Image	1992

Data Import and Export:

EXPORT DATA

Exporting data:

There are plenty of functions for writing data to files. Some of them are:

- write.table
- write.csv
- cat
- writeLines
- dump
- dput
- save
- serialize

EXERCISES

Consider the data set “airquality”.

Read first 6 lines into a new data frame “aq” (aq <- head(airquality))

Practice the following exercises on the following data.

```
> aq <- head(airquality)
> aq
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
```

Note:

After creating files, open and see the files to comply with the required output.

.csv files can be opened either by excel or notepad

You can observe the delimiters when you open the file using notepad.

Cat command:

- Write a command to export (store/save) data into the file cat_test1.txt (Use only two arguments). After creating file check the output.
- Write a command to export data into the file cat_test2.txt. Use separator as comma
- Write a command to export data into the file cat_test3.txt. Use separator as semi-colon
- Write a command to export data into the file cat_test3.txt. Use separator as tab (use \t to insert tab)
- Can a separator be any string that you want to insert as delim?? Experiment it.
- What are Delimiters? What is the need of delimiters?
- What is the difference between over writing and appending?
- Write a command to append the same data into the file cat_test1.txt. After creating file check the output.
- Write a command to append the same data into the file cat_test2.txt. Use separator as comma
- Write a command to append the same data into the file cat_test3.txt. Use separator as semi-colon
- Write a command to append the same data into the file cat_test3.txt. Use separator as tab (use \t to insert tab)
- Can you store the data into .csv files using cat command?
- Write a command to export the data to cat_test1.csv.
- Write a command to export the data using various separators such as semi-colon and tab and store them in new .csv files. Open newly created files using both excel and notepad. Observe the difference.
- What are other arguments you can use while exporting data to a file using cat function. (**Hint:** Use ?cat command to learn about other arguments and experiment on them)

write.table command:

Repeat the above exercises using write.table command. Here, file extension is .txt as write.table stores data into text files.

write.csv command:

Repeat the above exercises using write.csv command. Here, file extension is .csv as write.csv stores data into comma separated files.

Other commands:

Practice the below commands using the examples given in lecture slides

- writeLines, readLines
- dump, source
- dput, dget
- save, load
- serialize, unserialize
- Scan

Importing data:

There are a few principal functions reading data into R.

- read.table, read.csv, for reading tabular data
- readLines, for reading lines of a text file
- source, for reading in R code files (inverse of dump)
- dget, for reading in R code files (inverse of dput)
- load, for reading in saved workspaces
- unserialize, for reading single R objects in binary form
- read.delim(), for reading data separated by tab

EXERCISES

Note:

`read.csv` is identical to `read.table` except that the default separator is a comma

There are variants in the functions in `read.csv` and `read.delim` such as `read.csv2` and `read.delim2`.

read.table command:

- Write a command to read the data from `cat_test1.txt`. Display the output.
- Write a command to read the data from `cat_test2.txt`. Display the output. Modify your command to get the output as given below

```
Ozone Solar.R Wind Temp Month Day
1    41    190  7.4   67     5    1
2    36    118  8.0   72     5    2
3    12    149 12.6   74     5    3
4    18    313 11.5   62     5    4
5    NA     NA 14.3   56     5    5
6    28     NA 14.9   66     5    6
```

- Write a command to read the data from remaining files and display the output as in the above figure.
- Write a command to read and display the data without quotes on strings
- Write a command to read the data without column names
- Write a command to read the data without row names

Read.csv command:

Repeat the above exercises using `read.csv` command

Data Cleaning and Summarizing with “dplyr” package:

The dplyr package is one of the most powerful and popular package in R. This package was written by the most popular R programmer Hadley Wickham who has written many useful R packages such as ggplot2, tidyr etc. This post includes several examples and tips of how to use dplyr package for cleaning and transforming data. It's a complete tutorial on data manipulation and data wrangling with R.

What is dplyr?

The dplyr is a powerful R-package to manipulate, clean and summarize unstructured data. In short, it makes data exploration and data manipulation easy and fast in R.

What's special about dplyr?

The package "dplyr" comprises many functions that perform mostly used data manipulation operations such as applying filter, selecting specific columns, sorting data, adding or deleting columns and aggregating data.

Another most important advantage of this package is that it's very easy to learn and use dplyr functions. Also easy to recall these functions. For example, filter() is used to filter rows.

dplyr vs. Base R Functions

dplyr functions process faster than base R functions. It is because dplyr functions were written in a computationally efficient manner. They are also more stable in the syntax and better supports data frames than vectors.

SQL Queries vs. dplyr

People have been utilizing SQL for analyzing data for decades. Every modern data analysis software such as Python, R, SAS etc supports SQL commands. But SQL was never designed to perform data analysis. It was rather designed for querying and managing data. There are many data

analysis operations where SQL fails or makes simple things difficult. For example, calculating median for multiple variables, converting wide format data to long format etc. Whereas, dplyr package was designed to do data analysis.

The names of dplyr functions are similar to SQL commands such as select() for selecting variables, group_by() - group data by grouping variable, join() - joining two data sets. Also includes inner_join() and left_join(). It also supports sub queries for which SQL was popular for.

How to install and load dplyr package

To install the dplyr package, type the following command.

```
install.packages("dplyr")
```

To load dplyr package, type the command below

```
library(dplyr)
```

dplyr Function	Description	Equivalent SQL
select()	Selecting columns (variables)	SELECT
filter()	Filter (subset) rows.	WHERE
group_by()	Group the data	GROUP BY
summarise()	Summarise (or aggregate) data	-
arrange()	Sort the data	ORDER BY
join()	Joining data frames (tables)	JOIN
mutate()	Creating New Variables	COLUMN ALIAS

ALIAS

Data : Income Data by States

In this tutorial, we are using the following data which contains income generated by states from year 2002 to 2015. Note : This data do not contain actual income figures of the states.

This dataset contains 51 observations (rows) and 16 variables (columns).

The snapshot of first 6 rows of the dataset is shown below.

Index	State	Y2002	Y2003	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009
1	A	Alabama	1296530	1317711	1118631	1492583	1107408	1440134	1945229
2	A	Alaska	1170302	1960378	1818085	1447852	1861639	1465841	1551826
3	A	Arizona	1742027	1968140	1377583	1782199	1102568	1109382	1752886
4	A	Arkansas	1485531	1994927	1119299	1947979	1669191	1801213	1188104
5	C	California	1685349	1675807	1889570	1480280	1735069	1812546	1487315
6	C	Colorado	1343824	1878473	1886149	1236697	1871471	1814218	1875146
		Y2010	Y2011	Y2012	Y2013	Y2014	Y2015		
1		1237582	1440756	1186741	1852841	1558906	1916661		
2		1629616	1230866	1512804	1985302	1580394	1979143		
3		1300521	1130709	1907284	1363279	1525866	1647724		
4		1669295	1928238	1216675	1591896	1360959	1329341		
5		1624509	1639670	1921845	1156536	1388461	1644607		
6		1913275	1665877	1491604	1178355	1383978	1330736		

Download the Dataset

How to load Data

Submit the following code. Change the file path in the code below.

```
mydata = read.csv("C:\\Users\\Deepanshu\\Documents\\sampledata.csv")
```

Example 1 : Selecting Random N Rows

The `sample_n` function selects random rows from a data frame (or table).

The second parameter of the function tells R the number of rows to select.

```
sample_n(mydata,3)
```

```
Index State Y2002 Y2003 Y2004 Y2005 Y2006 Y2007 Y2008 Y2009
```

```
2 A Alaska 1170302 1960378 1818085 1447852 1861639 1465841 1551826 1436541
```

```
8 D Delaware 1330403 1268673 1706751 1403759 1441351 1300836 1762096 1553585
```

```
33 N New York 1395149 1611371 1170675 1446810 1426941 1463171 1732098 1426216
```

```
Y2010 Y2011 Y2012 Y2013 Y2014 Y2015
```

```
2 1629616 1230866 1512804 1985302 1580394 1979143
```

```
8 1370984 1318669 1984027 1671279 1803169 1627508
```

```
33 1604531 1683687 1500089 1718837 1619033 1367705
```