Joining Columns and Rows in a Data Frame

We can join multiple vectors to create a data frame using the **cbind()**function. Also we can merge two data frames using **rbind()** function.

# Create vector objects.

city <- c("Tampa","Seattle","Hartford","Denver")

state <- c("FL","WA","CT","CO")

zipcode <- c(33602,98104,06161,80294)

# Combine above three vectors into one data frame.

addresses <- cbind(city,state,zipcode)

# Print a header.

cat("# # # # The First data frame\n")

# Print the data frame.

print(addresses)

# Create another data frame with similar columns

new.address <- data.frame(

city = c("Lowry","Charlotte"),

state = c("CO","FL"),

zipcode = c("80230","33949"),

stringsAsFactors = FALSE

)

# Print a header.

cat("# # # The Second data frame\n")

# Print the data frame.

print(new.address)

# Combine rows form both the data frames.

all.addresses <- rbind(addresses,new.address)

# Print a header.

cat("# # # The combined data frame\n")

# Print the result.

print(all.addresses)

When we execute the above code, it produces the following result −

# # # # The First data frame

city state zipcode

[1,] "Tampa" "FL" "33602"

[2,] "Seattle" "WA" "98104"

[3,] "Hartford" "CT" "6161"

[4,] "Denver" "CO" "80294"

# # # The Second data frame

city state zipcode

1 Lowry CO 80230

2 Charlotte FL 33949

# # # The combined data frame

city state zipcode

1 Tampa FL 33602

2 Seattle WA 98104

3 Hartford CT 6161

4 Denver CO 80294

5 Lowry CO 80230

6 Charlotte FL 33949

Merging Data Frames

We can merge two data frames by using the **merge()** function. The data frames must have same column names on which the merging happens.

In the example below, we consider the data sets about Diabetes in Pima Indian Women available in the library names "MASS". we merge the two data sets based on the values of blood pressure("bp") and body mass index("bmi"). On choosing these two columns for merging, the records where values of these two variables match in both data sets are combined together to form a single data frame.

library(MASS)

merged.Pima <- merge(x = Pima.te, y = Pima.tr,

by.x = c("bp", "bmi"),

by.y = c("bp", "bmi")

)

print(merged.Pima)

nrow(merged.Pima)

When we execute the above code, it produces the following result −

bp bmi npreg.x glu.x skin.x ped.x age.x type.x npreg.y glu.y skin.y ped.y

1 60 33.8 1 117 23 0.466 27 No 2 125 20 0.088

2 64 29.7 2 75 24 0.370 33 No 2 100 23 0.368

3 64 31.2 5 189 33 0.583 29 Yes 3 158 13 0.295

4 64 33.2 4 117 27 0.230 24 No 1 96 27 0.289

5 66 38.1 3 115 39 0.150 28 No 1 114 36 0.289

6 68 38.5 2 100 25 0.324 26 No 7 129 49 0.439

7 70 27.4 1 116 28 0.204 21 No 0 124 20 0.254

8 70 33.1 4 91 32 0.446 22 No 9 123 44 0.374

9 70 35.4 9 124 33 0.282 34 No 6 134 23 0.542

10 72 25.6 1 157 21 0.123 24 No 4 99 17 0.294

11 72 37.7 5 95 33 0.370 27 No 6 103 32 0.324

12 74 25.9 9 134 33 0.460 81 No 8 126 38 0.162

13 74 25.9 1 95 21 0.673 36 No 8 126 38 0.162

14 78 27.6 5 88 30 0.258 37 No 6 125 31 0.565

15 78 27.6 10 122 31 0.512 45 No 6 125 31 0.565

16 78 39.4 2 112 50 0.175 24 No 4 112 40 0.236

17 88 34.5 1 117 24 0.403 40 Yes 4 127 11 0.598

age.y type.y

1 31 No

2 21 No

3 24 No

4 21 No

5 21 No

6 43 Yes

7 36 Yes

8 40 No

9 29 Yes

10 28 No

11 55 No

12 39 No

13 39 No

14 49 Yes

15 49 Yes

16 38 No

17 28 No

[1] 17

Melting and Casting

One of the most interesting aspects of R programming is about changing the shape of the data in multiple steps to get a desired shape. The functions used to do this are called **melt()** and **cast()**.

We consider the dataset called ships present in the library called "MASS".

library(MASS)

print(ships)

When we execute the above code, it produces the following result −

type year period service incidents

1 A 60 60 127 0

2 A 60 75 63 0

3 A 65 60 1095 3

4 A 65 75 1095 4

5 A 70 60 1512 6

.............

.............

8 A 75 75 2244 11

9 B 60 60 44882 39

10 B 60 75 17176 29

11 B 65 60 28609 58

............

............

17 C 60 60 1179 1

18 C 60 75 552 1

19 C 65 60 781 0

............

............

Melt the Data

Now we melt the data to organize it, converting all columns other than type and year into multiple rows.

molten.ships <- melt(ships, id = c("type","year"))

print(molten.ships)

When we execute the above code, it produces the following result −

type year variable value

1 A 60 period 60

2 A 60 period 75

3 A 65 period 60

4 A 65 period 75

............

............

9 B 60 period 60

10 B 60 period 75

11 B 65 period 60

12 B 65 period 75

13 B 70 period 60

...........

...........

41 A 60 service 127

42 A 60 service 63

43 A 65 service 1095

...........

...........

70 D 70 service 1208

71 D 75 service 0

72 D 75 service 2051

73 E 60 service 45

74 E 60 service 0

75 E 65 service 789

...........

...........

101 C 70 incidents 6

102 C 70 incidents 2

103 C 75 incidents 0

104 C 75 incidents 1

105 D 60 incidents 0

106 D 60 incidents 0

...........

...........

Cast the Molten Data

We can cast the molten data into a new form where the aggregate of each type of ship for each year is created. It is done using the **cast()** function.

recasted.ship <- cast(molten.ships, type+year~variable,sum)

print(recasted.ship)

When we execute the above code, it produces the following result −

type year period service incidents

1 A 60 135 190 0

2 A 65 135 2190 7

3 A 70 135 4865 24

4 A 75 135 2244 11

5 B 60 135 62058 68

6 B 65 135 48979 111

7 B 70 135 20163 56

8 B 75 135 7117 18

9 C 60 135 1731 2

10 C 65 135 1457 1

11 C 70 135 2731 8

12 C 75 135 274 1

13 D 60 135 356 0

14 D 65 135 480 0

15 D 70 135 1557 13

16 D 75 135 2051 4

17 E 60 135 45 0

18 E 65 135 1226 14

19 E 70 135 3318 17

20 E 75 135 542 1