How can I perform data binning in R? Can you provide some examples?

Authored by

stats writer

July 1, 2024

Data binning is a statistical technique used to organize and categorize a large set of data into smaller groups or "bins" based on specific criteria. This process can be easily performed in R, a popular programming language used for data analysis and statistical computing. To bin data in R, you can use the "cut" or "qcut" functions, which allow you to specify the number of bins and the range of values for each bin. Additionally, there are various packages available in R, such as "dplyr" and "ggplot2," that provide more advanced binning methods and visualizations. Some examples of data binning in R include grouping age data into age ranges, categorizing income data into income brackets, or dividing test scores into letter grades. By binning data, you can gain a better understanding of patterns and trends within the data and simplify complex data sets for analysis.

Perform Data Binning in R (With Examples)

You can use one of the following two methods to perform data binning in R:

Method 1: Use cut() Function

```
library(dplyr)

#perform binning with custom breaks
df %>% mutate(new_bin = cut(variable_name, breaks=c(0, 10, 20, 30)))

#perform binning with specific number of bins
df %>% mutate(new_bin = cut(variable_name, breaks=3))
```

Method 2: Use ntile() Function

```
library(dplyr)

#perform binning with specific number of bins
df %>% mutate(new_bin = ntile(variable_name, n=3))
```

The following examples show how to use each method in practice with the following data frame:

```
#create data frame
df <- data.frame(points=c(4, 4, 7, 8, 12, 13, 15, 18, 22, 23, 23, 25),
assists=c(2, 5, 4, 7, 7, 8, 5, 4, 5, 11, 13, 8),
rebounds=c(7, 7, 4, 6, 3, 8, 9, 9, 12, 11, 8, 9))

#view head of data frame
head(df)

points assists rebounds
1 4 2 7
2 4 5 7
3 7 4 4
4 8 7 6
5 12 7 3
6 13 8 8
```

Example 1: Perform Data Binning with cut() Function

The following code shows how to perform data binning on the points variable using the cut() function with specific break marks:

```
library(dplyr)

#perform data binning on points variable
df %>% mutate(points_bin = cut(points, breaks=c(0, 10, 20, 30)))
```

```
points assists rebounds points_bin
1 4 2 7 (0,10]
2 4 5 7 (0,10]
3 7 4 4 (0,10]
4 8 7 6 (0,10]
5 12 7 3 (10,20]
6 13 8 8 (10,20]
7 15 5 9 (10,20]
8 18 4 9 (10,20]
9 22 5 12 (20,30]
10 23 11 11 (20,30]
11 23 13 8 (20,30]
12 25 8 9 (20,30]
```

Notice that each row of the data frame has been placed in one of three bins based on the value in the points column.

We could also specify the number of breaks to use to create bins of equal width that range from the minimum value to the maximum value of the points column:

```
library(dplyr)

#perform data binning on points variable
df %>% mutate(points_bin = cut(points, breaks=3))

points assists rebounds points_bin
1 4 2 7 (3.98,11]
2 4 5 7 (3.98,11]
3 7 4 4 (3.98,11]
4 8 7 6 (3.98,11]
5 12 7 3 (11,18]
6 13 8 8 (11,18]
7 15 5 9 (11,18]
8 18 4 9 (11,18]
9 22 5 12 (18,25]
10 23 11 11 (18,25]
11 23 13 8 (18,25]
```

12 25 8 9 (18,25]

Example 2: Perform Data Binning with ntile() Function

The following code shows how to perform data binning on the points variable using the ntile() function with a specific number of resulting bins:

```
library(dplyr)

#perform data binning on points variable
df %>% mutate(points_bin = ntile(points, n=3))
```

```
points assists rebounds points_bin
1 4 2 7 1
2 4 5 7 1
3 7 4 4 1
4 8 7 6 1
5 12 7 3 2
6 13 8 8 2
7 15 5 9 2
8 18 4 9 2
9 22 5 12 3
10 23 11 11 3
11 23 13 8 3
```

12 25 8 9 3

Notice that each row has been assigned a bin from 1 to 3 based on the value of the points column.

Additional Resources

The following tutorials explain how to perform other common tasks in R: