tidyr package

Transposing, Splitting and Concatenating

Contents

Introduction

- The way package **`reshape2`** is used to reshape data, **`tidyr`** is also one of the options to do the same.
- **`tidyr`** is developed by Hadley Wickham which provides four main functions: **`gather(), spread(), separate(), unite()`** which are similar to functions in package reshape2. We will see the difference and similarity between the two packages while performing the following tasks:
  - Convert data from long format to wide & vice versa
  - Split one character column into multiple columns
  - Concatenate multiple columns into one column

Data Snapshot

stud_data consist student names & their marks scored in Math's, Economics & Statistics. It has 5 rows & 5 columns.

Variables

Observations

Student_ID	Names	Maths	Economics	Statistics
1	Rohan	67	56	
2	John	89	88	79

Columns	Description	Type	Measurement	Possible values
Student_ID	Student ID	character	-	-
Names	Student names	character	-	-
Maths	Marks scored in Maths	numeric	-	positive values
Economics	Marks scored in Economics	numeric	-	positive values
Statistics	Marks scored in Statistics	numeric	-	positive values

gather()

- `gather()` converts data from wide format to long format. It take multiple columns and collapses into key-value pairs.

  key is the name of new "key" column (made from names of data columns)

  value is the name of new "value" column

Student_ID	Names	Maths	Economics	Statistics
1	Rohan	67	56	
2	John	89	88	79
3	Anisha	69		88
4	Agatha	79	92	89
5	Ashima	77	67	89

Student_ID	Names	Subjects	Marks
1	Rohan	Maths	67
2	John	Maths	89
3	Anisha	Maths	69
4	Agatha	Maths	79
5	Ashima	Maths	77
1	Rohan	Economics	56
2	John	Economics	88
3	Anisha	Economics	NA
4	Agatha	Economics	92
5	Ashima	Economics	67
1	Rohan	Statistics	NA
2	John	Statistics	79
3	Anisha	Statistics	88
4	Agatha	Statistics	89
5	Ashima	Statistics	89

gather()

```
# Install and load package tidyr
# Import stud_data data
# Convert the stud_data format to long format
```

```
install.packages("tidyr")
library(tidyr)
stud_data<-read.csv("stud_data.csv",header=TRUE)

gather(stud_data)
```

*

Wide Format data: each column represents a different variable.
Long Format data: one column contains all the possible variables, another column contains their respective values
Detailed discussion of these two formats is done in our tutorial for package **reshape2**

gather()

```
gather(stud_data)
```

```
          key  value
1  Student_ID      1
2  Student_ID      2
3  Student_ID      3
4  Student_ID      4
5  Student_ID      5
6       Names  Rohan
7       Names   John
8       Names Anisha
9       Names Agatha
10      Names Ashima
11      Maths     67
12      Maths     89
13      Maths     69
14      Maths     79
15      Maths     77
16  Economics     56
17  Economics     88
18  Economics   <NA>
19  Economics     92
20  Economics     67
21 Statistics   <NA>
22 Statistics     79
23 Statistics     88
24 Statistics     89
25 Statistics     89
Warning message:
attributes are not identical across measure variables;
they will be dropped
```

In our tutorial for Package **reshape2**, we have seen how **melt()** function works. **gather()** is similar to it**.**

- **gather()** converts data from wide to long format with a warning by treating all columns as key while **melt()** treats **Names** as id variables (**Id columns are the columns that contain the identifier of the observation that is represented as a row in our data set**).
- Indeed, if we don't specify any id variables to **melt()**, then it will use the factor or character columns as id variables whereas **gather()** requires mentioning the columns that needs to be treated as key-value pairs.

gather() with key:value Pair

```
# Here, we are adding new variables Subjects and Marks.
longformat<-gather(stud_data,Subjects,Marks,Maths,Economics,Statistics)
longformat
```

```
# Output
   Student_ID  Names   Subjects Marks
1           1  Rohan      Maths    67
2           2   John      Maths    89
3           3 Anisha      Maths    69
4           4 Agatha      Maths    79
5           5 Ashima      Maths    77
6           1  Rohan  Economics    56
7           2   John  Economics    88
8           3 Anisha  Economics    NA
9           4 Agatha  Economics    92
10          5 Ashima  Economics    67
11          1  Rohan Statistics    NA
12          2   John Statistics    79
13          3 Anisha Statistics    88
14          4 Agatha Statistics    89
15          5 Ashima Statistics    89
```

In the syntax of **gather()**, after the data is mentioned:

- 1st mentioned variable is key
- 2nd mentioned variable is value
- and the subsequent variables are the ones' to be gathered.

This command can be alternatively written as:
longformat<-gather(stud_data, Subjects,Marks, Maths:Statistics)
Note that this data has 2 missing values
To remove rows from output where the value column is **NA**, include **na.rm=TRUE.**

* **gather()** cannot handle matrices or arrays, while **melt()** can.

spread()

- **`spread()`** converts data from long format to wide format.
- The **`spread()`** function spreads a key-value pair across multiple columns. It's a complement of **`gather()`**.

Student_ID	Names	Subjects	Marks
1	Rohan	Maths	67
2	John	Maths	89
3	Anisha	Maths	69
4	Agatha	Maths	79
5	Ashima	Maths	77
1	Rohan	Economics	56
2	John	Economics	88
3	Anisha	Economics	NA
4	Agatha	Economics	92
5	Ashima	Economics	67
1	Rohan	Statistics	NA
2	John	Statistics	79
3	Anisha	Statistics	88
4	Agatha	Statistics	89
5	Ashima	Statistics	89

Student_ID	Names	Maths	Economics	Statistics
1	Rohan	67	56	
2	John	89	88	79
3	Anisha	69		88
4	Agatha	79	92	89
5	Ashima	77	67	89

spread() with fill

```
# Convert the longformat data from last example to wide format
spread(longformat,Subjects,Marks,fill=0)
# Output
```

	Student_ID	Names	Economics	Maths	Statistics
1	1	Rohan	56	67	0
2	2	John	88	89	79
3	3	Anisha	0	69	88
4	4	Agatha	92	79	89
5	5	Ashima	67	77	89

- **spread()** two columns(key-value pair)are spread into multiple columns, making 'long' data wider.
- **fill=** is used to replace NA's with the value provided to it.
- **Note** that there are **two types of missingness** in the input: explicit missing values (i.e. NA), and implicit missing rows that simply aren't present. Both types of missing value will be replaced by fill.

* **dcast()** from the package reshape2 performs the same task as **spread().** Also, **dcast()** allows to perform aggregation which is not possible in package **tidyr**

separate()

separate() splits a single character column into multiple columns.

```
# Create a data frame empdata with columns as empid, location,
# address and date.
# Split the column date in 3 columns
```

```
empid<-c(101,102,103,104)
location<-c("Mumbai","Delhi","Delhi","Mumbai")
address<-c("4/Churchgate","12/Rohini","8/Pitampura", "21/Andheri")
date<-c("2016-10-09","2010-11-01","2009-09-23","1990-02-30")
empdata<-data.frame(empid,location,address,date)

empdata
```

employee data:
columns: **empid**(Employee ID),**Location**, **address**(sector and area), **date**(Date of joining)

```
# Output
```

	empid	location	address	date
1	101	Mumbai	4/Churchgate	2016-10-09
2	102	Delhi	12/Rohini	2010-11-01
3	103	Delhi	8/Pitampura	2009-09-23
4	104	Mumbai	21/Andheri	1990-02-30

separate() with into and convert

```
sep_date<-separate(empdata,date,into=c("Year","Month","Date"))
sep_date
```

```
# Output
  empid location      address Year Month Date
1   101   Mumbai 4/Churchgate 2016    10   09
2   102    Delhi    12/Rohini 2010    11   01
3   103    Delhi  8/Pitampura 2009    09   23
4   104   Mumbai   21/Andheri 1990    02   30
```

into= takes names of new columns to create character vector.

```
class(sep_date$Year)
```

```
[1] "character"
```

By default, new columns created will be of the type of original column. Here, since **date** is of type character, columns **Year, Month** and **Date** will of the same type.

```
sep_date<-separate(empdata,date,into=c("Year","Month","Date"),
convert=TRUE)
class(sep_date$Year)
```

```
[1] "integer"
```

convert=TRUE will run **type.convert** with **as.is=TRUE** on new columns. This is useful if the component columns are integer, numeric or logical.

separate() with sep

```
# Split the column address in 2 columns
```

```
sep_address<-separate(empdata,address,into=c("sector","area"),sep="/",
convert=TRUE)
sep_address
```

```
# Output
```

	empid	location	sector	area	date
1	101	Mumbai	4	Churchgate	2016-10-09
2	102	Delhi	12	Rohini	2010-11-01
3	103	Delhi	8	Pitampura	2009-09-23
4	104	Mumbai	21	Andheri	1990-02-30

- Here, we have split the column **address** into two new columns: **sector** and **area** with separator as '/'.
- **sep=** is used to specify separator between columns. The default value is a regular expression that matches any sequence of non-alphanumeric values. Here, separator is assumed as '-'.

* **colsplit()** from package **reshape2** give the same result as you get with **separate()**. Difference is **colsplit()** works only on a single column, we need to use **cbind()** to combine split columns with original data while **separate()** performs all the operations at once reducing the possibility of making mistakes.

unite() with sep

unite() is a complement of **separate()**. It unites multiple columns into single column.

```
# Unite the 3 date columns into one
unite_date<-unite(sep_date,date,c(Year,Month,Date),sep="/")
unite_date
```

unite() takes the dataframe, name of the column to add, vector of columns to combine and a separator to use between the values as arguments

```
# Output
  empid location      address       date
1   101   Mumbai 4/Churchgate 2016/10/09
2   102    Delhi    12/Rohini 2010/11/01
3   103    Delhi  8/Pitampura 2009/09/23
4   104   Mumbai   21/Andheri 1990/02/30
```

tidyr vs reshape2

- As we have seen tidyr and reshape2 functions perform similar operations.
- reshape2 functions can do aggregation which is not possible with tidyr.
- tidyr is designed specifically for tidying data, while reshape2 is designed with a wider purpose of reshaping and aggregating.
- Therefore, we use **`gather()`** and **`separate()`** functions from tidyr to quickly tidy our data and **`dcast()`** function from reshape2 to aggregate them.

Quick Recap

In this session, we learnt how to tidy our data using tidyr functions and what is the difference between package tidyr and reshape2. Here is the quick recap:

tidyr functions

- **`gather()`**: converts wide data to longer format. It is similar to the **`melt()`** function from **`reshape2`** but can handle only dataframes.
- **`spread()`**: converts long data to wider format. It is similar to the **`dcast()`** function from **`reshape2`**.
- **`separate()`**: splits one column into two or more columns. it is similar to **`colsplit()`** function from **`reshape2.`**
- **`unite()`**: combines two or more columns into a single column.