

Package reshape2

Converting Data from Wide to Long Format

Contents

1. Understanding the difference between Long Format and Wide Format
2. Introduction to reshape2
3. Reshaping data using cast and melt functions

Long format vs Wide format

Some data have wide format and some have long; for different analysis you may choose one of the either.

In **Long Format** data,
one column contains all the
possible variables, another
column contains their
respective values. For example:

| ID | Names | Subject | variable | value |
|----|---------|------------|-------------|-------|
| 1 | Rohit | Maths | I_semester | 78 |
| 2 | John | Physics | I_semester | 56 |
| 3 | Vivek | Statistics | I_semester | 76 |
| 4 | Martina | Physics | I_semester | 89 |
| 5 | Agatha | Statistics | I_semester | 67 |
| 1 | Rohit | Maths | II_semester | 88 |
| 2 | John | Physics | II_semester | 59 |
| 3 | Vivek | Statistics | II_semester | 81 |
| 4 | Martina | Physics | II_semester | 73 |
| 5 | Agatha | Statistics | II_semester | 80 |

In **Wide Format** data,
each column represents a
different variable. For example:

| ID | Names | Subject | I_semester | II_semester |
|----|---------|------------|------------|-------------|
| 1 | Rohit | Maths | 78 | 88 |
| 2 | John | Physics | 56 | 59 |
| 3 | Vivek | Statistics | 76 | 81 |
| 4 | Martina | Physics | 89 | 73 |
| 5 | Agatha | Statistics | 67 | 80 |

Long format vs Wide format

- Long data is mainly used in data visualization (plotting graphs) and statistical modeling. Also it is easy to perform operations on subsets of data.
- Wide data is used in the calculation of growth, for example, we have a company data with list of companies and their revenue for 4 Quarters. Our data is in wide format with 5 columns: Company, Quarter1, Quarter2, Quarter3, Quarter4. Since our data is in a wide format it is easy to calculate Quarter wise growth.
- In practical scenario, while wide format is more readable, long format is easier to analyze. Therefore, it is useful to know how to convert between the two.

Introduction to reshape2

- We know that data comes in many forms. Hence, we are required to reshape it according to our need.
- The process of reshaping data in R is tedious. R base functions for 'aggregation' reduces and rearranges the data into smaller forms, but with reduction in amount of information. Also, these functions don't solve the problem of converting data from wide to long format & vice versa. The package **reshape2** overcomes these problems.
- **reshape2** is an R package written and maintained by Hadley Wickham.
- **reshape2** makes it easy to convert data from long to wide formats & vice versa.

Package reshape2 has two key functions:

melt – converts data from wide format to long format

cast – converts data from long format to wide format

Data Snapshot

stud_data consists 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 |

Tasks to be Performed

stud_data is originally in a wide format. We will perform following activities using package reshape2:

- Convert data format from wide to long & Changing the names of new columns that will be created in the conversion process.
- Convert data format from long to wide. While doing this conversion perform subsetting and aggregation.

melt()

- `melt()` converts data from wide format to long format. It's a kind of restructuring where categorical variables are 'melted' into unique observations.
- Here variables are divided into 2 groups: identifier and measured variables.
- Identifier variables(id) identify the unit or column that measurements takes place on.

```
# Install and load package reshape2  
# Import stud_data  
# Convert wide data into long
```

```
install.packages("reshape2")  
library(reshape2)  
stud_data<-read.csv("stud_data.csv",header=TRUE)
```


melt()

```
melt(stud_data)
```

#Output

Using Names as id variables

| | Names | variable | value |
|----|--------|------------|-------|
| 1 | Rohan | Student_ID | 1 |
| 2 | John | Student_ID | 2 |
| 3 | Anisha | Student_ID | 3 |
| 4 | Agatha | Student_ID | 4 |
| 5 | Ashima | Student_ID | 5 |
| 6 | Rohan | Maths | 67 |
| 7 | John | Maths | 89 |
| 8 | Anisha | Maths | 69 |
| 9 | Agatha | Maths | 79 |
| 10 | Ashima | Maths | 77 |
| 11 | Rohan | Economics | 56 |
| 12 | John | Economics | 88 |
| 13 | Anisha | Economics | NA |
| 14 | Agatha | Economics | 92 |
| 15 | Ashima | Economics | 67 |
| 16 | Rohan | Statistics | NA |
| 17 | John | Statistics | 79 |
| 18 | Anisha | Statistics | 88 |
| 19 | Agatha | Statistics | 89 |
| 20 | Ashima | Statistics | 89 |

- ❑ By default, **melt()** assumes factor and character variables are id variables, and all others are measured. Here, **Names** is assumed as id variable and rest of the columns as measured variables.
- ❑ Column “**variable**” contains measured variables and Column “**value**” contains their respective values.
- ❑ All measured variables must be of same type, (e.g. numeric, factor, date) because melted data is stored in a R data frame, and the column **value** can assume only one type.
- ❑ Note that **stud_data** has 2 missing values. To remove missing values from the melted data set, include **na.rm=TRUE**.

melt() with id.vars

ID variables can be separately specified

Change the ID variables to 'Student_ID' and 'Names'

```
melt(stud_data, id.vars=c('Student_ID', 'Names'))
```

#Output

| | Student_ID | Names | variable | value |
|----|------------|--------|------------|-------|
| 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 |

With **id.vars=** we can give multiple id variables & we can specify them in a list of variables enclosed in **c()**. If we only specify id variables, by default non id variables will be taken as measured variables.

See the next example for setting measured variables.

melt() with measure.vars

Specify measured variables as Maths and Economics

```
melt(stud_data, id.vars=c('Student_ID', 'Names'), measure.vars=c('Maths',  
'Economics'))
```

#Output

| | Student_ID | Names | variable | value |
|----|------------|--------|-----------|-------|
| 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 |

With **measure.vars=** one can specify a vector of measured variables.

melt() with variable.name and value.name

What if we want to control the column names in our long format data? We can do that in just one step:

```
# Change the names of new columns created after melting stud_data
```

```
long_format<-melt(stud_data,id.vars=c('Student_ID','Names'),  
variable.name='Subjects', value.name='Marks')  
long_format
```

#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 |

Using **variable.name=** and **value.name=** arguments, we can change the names of the columns “**variable**” and “**value**”.

cast function

- **cast** function converts data from long format to wide format. It is a complement of **melt()** function. There are two types of **cast** functions: **dcast()** and **acast()**

dcast() – returns a dataframe as the output

acast() – returns a vector/matrix/array as the output

The basic arguments of *cast is the data in long format and a formula of the form:

x1 + x2 ~ y1 + y2

id variables are specified on the left and measured variables on the right.

The order of the variables matter, the first varies slowest, and the last fastest.

- Since dataframe objects are the most common, we will explore the **dcast()** function.

Let's take the **long-format** data object which we created in the previous example and cast it into some different wide formats.

dcast() with Formula

Reshape **long_format** object into wide format

```
dcast(long_format, Student_ID+Names~Subjects)
```

#Output

Using Marks as value column: use value.var to override.

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

dcast() uses a formula to describe the shape of the data.

There are different ways in which you can dcast data.

See the below commands :

```
dcast(long_format, Names+Student_ID~Subjects)
```

```
dcast(long_format, Student_ID~Subjects)
```

```
dcast(long_format, Names ~ Subjects)
```

dcast() with subset

You can subset variables while converting data from long to wide format.

```
# Subsetting  
# Reshape long_format object into wide format displaying columns:  
# Student_ID, Names and Maths column
```

```
install.packages("plyr")  
library(plyr)  
dcast(long_format, Student_ID+Names~Subjects,  
subset=.(Subjects=="Maths"))
```

#Output

Using Marks as value column: use value.var to override.

| | Student_ID | Names | Maths |
|---|------------|--------|-------|
| 1 | 1 | Rohan | 67 |
| 2 | 2 | John | 89 |
| 3 | 3 | Anisha | 69 |
| 4 | 4 | Agatha | 79 |
| 5 | 5 | Ashima | 77 |

Here, we have sub setted **Maths** from the variable **Subjects**.

- ☐ **subset=** is used to subset data using **.()**
- ☐ package **plyr** is needed to access **'.'**

dcast() with fun.aggregate

One confusing “mistake” you might make is casting a dataset in which there is more than one value per data cell. For example, this time we won’t include any measured variable.

```
dcast(long_format, Student_ID+Names ~ .)
```

#Output

Using Marks as value column: use value.var to override.
Aggregation function missing: defaulting to length

| | Student_ID | Names | . |
|---|------------|--------|---|
| 1 | 1 | Rohan | 3 |
| 2 | 2 | John | 3 |
| 3 | 3 | Anisha | 3 |
| 4 | 4 | Agatha | 3 |
| 5 | 5 | Ashima | 3 |

- ☐ “.” represents no variable.
- ☐ Notice a warning message: Aggregation function missing: defaulting to length
- ☐ If you look at the output, the cells are filled with the number of data rows for each **Student_ID-Names** combination.
- ☐ The numbers we’re seeing are the number of **Subjects** for each **Student_ID-Names** combination .

So, when you cast your data and there are multiple values per cell, you also need to tell **dcast** how to aggregate the data. See the next example for using aggregation function.

dcast() with fun.aggregate

Calculate total marks for all students

Aggregation

```
dcast(long_format, Student_ID+Names ~ ., fun.aggregate=sum, na.rm=TRUE)
```

Using Marks as value column: use value.var to override.

| | Student_ID | Names | . |
|---|------------|--------|-----|
| 1 | 1 | Rohan | 123 |
| 2 | 2 | John | 256 |
| 3 | 3 | Anisha | 157 |
| 4 | 4 | Agatha | 260 |
| 5 | 5 | Ashima | 233 |

- ❑ **fun.aggregate** is used to specify the aggregation function
- ❑ **na.rm=TRUE** is to remove NA values.
- ❑ In base R, **aggregate()** also has similar formula where variables on left hand side are continuous and on right hand side are categorical.

In this case, an aggregation function reduces multiple values to a single one.

Quick Recap

In this session, we learnt how to change the data from long to wide format and vice-versa. Here is the quick recap:

Long vs Wide Format

- In Long Format data, one column contains all the possible variables, another column contains their respective values.
- In Wide Format data, each column represents a different variable.

melt()

- melt() converts data from wide format to long format.
- Usage: `melt(data, id.vars, measure.vars, variable.name="variable", na.rm=FALSE, value.name="value")`

cast()

- Use **acast** or **dcast** depending on whether you want vector/matrix/array output or data frame output.
- Usage: `dcast(data, formula, fun.aggregate=NULL, subset = NULL)`