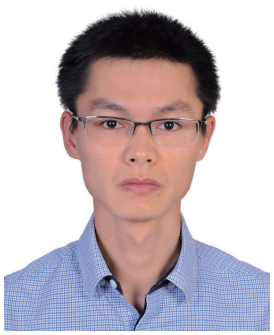
Reshaping and aggregating data: an introduction to reshape package

Zhongheng Zhang

Department of Critical Care Medicine, Jinhua Municipal Central Hospital, Jinhua Hospital of Zhejiang University, Jinhua 321000, China
Correspondence to: Zhongheng Zhang, MMed. 351#, Mingyue Road, Jinhua 321000, China. Email: zh_zhang1984@hotmail.com.

Author's introduction: Zhongheng Zhang, MMed. Department of Critical Care Medicine, Jinhua Municipal Central Hospital, Jinhua Hospital of Zhejiang University. Dr. Zhongheng Zhang is a fellow physician of the Jinhua Municipal Central Hospital. He graduated from School of Medicine, Zhejiang University in 2009, receiving Master Degree. He has published more than 35 academic papers (science citation indexed) that have been cited for over 200 times. He has been appointed as reviewer for 10 journals, including *Journal of Cardiovascular Medicine*, *Hemodialysis International*, *Journal of Translational Medicine*, *Critical Care*, *International Journal of Clinical Practice*, *Journal of Critical Care*. His major research interests include hemodynamic monitoring in sepsis and septic shock, delirium, and outcome study for critically ill patients. He is experienced in data management and statistical analysis by using R and STATA, big data exploration, systematic review and meta-analysis.



Zhongheng Zhang, MMed.

Abstract: It is common that data format extracted from clinical database does not meet the purpose of statistical analysis. In clinical research, variables are frequently measured repeatedly over the follow-up period. Such data can be displayed either in wide or long format. Transformation between these 2 forms can be challenging by hand. Fortunately, there are sophisticated packages in R environment. Data frame should firstly be melted and then casted to format that you want. Aggregation over unique combination of id variables is also allowable. Additionally, the article also introduces 2 functions colsplit() and funstofun() that can be useful in some circumstances.

Keywords: Aggregating; reshape package; R

Submitted Dec 20, 2015. Accepted for publication Jan 09, 2016.
doi: 10.3978/j.issn.2305-5839.2016.01.33
View this article at: http://dx.doi.org/10.3978/j.issn.2305-5839.2016.01.33

Introduction

In clinical studies involving data management of electronic medical record, variables are frequently grouped by one or more other variables (1). For example, when you follow up stroke patients for their quality of life (measured by some certain scores), the scores are recorded at each visit. In other words, these scores are nested within each patient. The display of such data can adopt long or wide format. The former listed each visit score longitudinally in a column and a variable denoting patient identification is mandatory. One patient can take up several rows in long format. One the other hand, the wide format displays 1 patient per row, and each visit score is listed consecutively in a single row. Both formats have their advantages and disadvantages depending on the purpose of analysis. These are a simple example illustrating the importance of data shape during analysis, and some are far more complex. This article introduces a powerful R package named "reshape", which is able to handle varieties of data format (2).

Working example

Suppose we have 3 patients, and each of them has blood partial pressure of oxygen measured on daily basis for 3 days. Blood oxygen can be measured from arterial line (*PaO2*) and central venous line (*PcvO2*).

```
> id<-rep(1:3,each=3)
> time<-rep(1:3,3)
> PaO2<-round(rnorm(9,mean=70,sd=10))
> PcvO2<-round(rnorm(9,mean=40,sd=8))
> data<-data.frame(id,time,PaO2,PcvO2)
> data
```

	id	time	PaO2	PcvO2
1	1	1	78	42
2	1	2	86	36
3	1	3	72	39
4	2	1	89	37
5	2	2	81	47
6	2	3	66	36
7	3	1	65	30
8	3	2	60	46
9	3	3	88	30

Oxygen content is significantly lower in central vein than that in artery, which is consistent with the common sense that arterial oxygenation is much greater than venous oxygenation.

Melting a dataset

The melt() function converts a wide format into long format. A number of variables listed in columns can be stacked into a single column. As in our example, both venous and arterial partial pressures can be stacked in a single column after application of melt(). This function requires an *id* variable and variables of interests to be stacked. The generic form of melt() function is like this:

```
melt(data, id.vars, measure.vars,
     variable_name = "variable",
    na.rm = !preserve.na, preserve.na = TRUE, ...)
```

If either *id.vars* or *measure.vars* is specified, the function takes the remainder variable in the data frame belong to the other. If neither is specified, the function assumes the character and factor variable as the *id.vars*, and the remainders are *measure.vars*. To avoid confusion, you'd better specify both of them. The "variable_name" argument specified the name of the new variable that will be created to store stacked variables. Now let's take a close look at how melt() works by using our working example.

```
> data.melt<-melt(data, id=(c("id", "time")),measure.va
rs=(c("PaO2","PcvO2")),variable_name="PO2")
> data.melt
```

	id	time	PO2	value
1	1	1	PaO2	78
2	1	2	PaO2	86
3	1	3	PaO2	72
4	2	1	PaO2	89
5	2	2	PaO2	81
6	2	3	PaO2	66
7	3	1	PaO2	65
8	3	2	PaO2	60
9	3	3	PaO2	88
10	1	1	PcvO2	42
11	1	2	PcvO2	36
12	1	3	PcvO2	39
13	2	1	PcvO2	37

14	2	2	PcvO2	47
15	2	3	PcvO2	36
16	3	1	PcvO2	30
17	3	2	PcvO2	46
18	3	3	PcvO2	30

Both *id* and *time* are *id.vars* and thus they remain unchanged. PaO2 and PcvO2 are stacked into a single column and a new variable called "PO2" is added to distinguish between arterial and venous oxygen. This melted format is not only useful for statistical analysis but also helpful in reshaping and aggregating data.

Casting a data frame

The cast() function contained in reshape package works on melted dataset. It transforms long data into wide format and can aggregate variable within any combinations of id variables. The generic form of cast() function takes the following form:

```
cast(data, formula = ... ~ variable, fun.aggregate=NULL,
...,
      margins=FALSE, subset=TRUE, df=FALSE,
      fill=NULL, add.missing=FALSE,
      value = guess_value(data))
```

the argument data is a melted dataset. The cast formula has the format:

```
x_variable + x_2 ~ y_variable + y_2 ~ z_variable ~ ... |
list_variable + ...
```

The x variables in combination define the rows, and y variables in combination defines the columns. If a set of x variables does not uniquely identify a row, the *fun.aggregate* argument should be given. Then the aggregate function can be applied to rows identified by a certain combination of x variables. List variables and z variables are usually not required. We don't have dataset of that complex! Next, I will show how to cast the melted data in different formats.

```
> cast(data.melt,id~PO2,mean)
```

	id	PaO2	PcvO2
1	1	78.66667	39.00000
2	2	78.66667	40.00000
3	3	71.00000	35.33333

The rows are defined by the *id* variable on the left side of the formula, and columns are defined by *PO2* variable. There are 2 levels contained in the *PO2*, thus we obtain 2 columns. Because the *id* variable does not uniquely identify a row, function mean is applied to vectors identified by *id* variable.

```
> cast(data.melt,time~PO2,mean)
```

	time	PaO2	PcvO2
1	1	77.33333	36.33333
2	2	75.66667	43.00000
3	3	75.33333	35.00000

When *id* is replaced by *time*, the row represents mean value across *id* variable.

```
> cast(data.melt,id+time~PO2)
```

	id	time	PaO2	PcvO2
1	1	1	78	42
2	1	2	86	36
3	1	3	72	39
4	2	1	89	37
5	2	2	81	47
6	2	3	66	36
7	3	1	65	30
8	3	2	60	46
9	3	3	88	30

Variables *id+time* on the left side of the formula define the rows. Each unique combination of id and time defines a row. Columns are in line with the levels of *PO2* variable. Because the 3 variables have identified a unique row, aggregating function is no longer applicable.

```
> cast(data.melt, id ~ time+PO2, subset=time < 3 & id
< 3)
```

	id	1_PaO2	1_PcvO2	2_PaO2	2_PcvO2
1	1	78	42	86	36
2	2	89	37	81	47

Data can be subset before reshaping. In the above

example, we restrict subset of data with *time*<3 and *id*<3.

```
> cast(data.melt,id~time~PO2)
, , PO2 = PaO2

   time
id       1       2       3
1       78      86      72
2       89      81      66
3       65      60      88

, , PO2 = PcvO2

   time
id       1       2       3
1       42      36      39
2       37      47      36
3       30      46      30
```

When a *z* variable is applied, we can see that the melted data is split into 2 datasets by variable *PO2*. Each item can be directly called by using the following code:

```
> cast(data.melt, id~time| PO2)$PaO2
        id       1       2       3
1        1      78      86      72
2        2      89      81      66
3        3      65      60      88
```

Row and column margins can be calculated with following code.

```
> cast(data.melt, time ~ PO2, mean, margins=c("grand_
row", "grand_col"))
    time    PaO2        PcvO2       (all)
1   1       77.33333    36.33333    56.83333
2   2       75.66667    43.00000    59.33333
3   3       75.33333    35.00000    55.16667
4   (all)   76.11111    38.11111    57.11111
```

Split character vector into multiple columns

The function colsplit() in reshape package is to split character vector into multiple columns on certain expression. This can be helpful in handling a list of variable names. Suppose that we have 3 laboratory items measured on consecutive 3 days. Their variable names can be denoted by: lac_1, lac_2, lac_3, wbc_1, wbc_2, wbc_3, hb_1, hb_2 and hb_3. In analysis, you want to list laboratory values in a column, and the type of value and measurement days are denoted by separate variables.

```
> data.split<-data.frame(lac_1=2.3, lac_2=3.4,
lac_3=4.5, wbc_1=12, wbc_2=11, wbc_3=6, hb_1=60,
hb_2=77, hb_3=89)
> variable.name<-colsplit(names(data.
split),"_",c("lab","days"))
> data.reshape<-cbind(variable.name,t(data.split))
> row.names(data.reshape)<-NULL
> names(data.reshape)[3]<-"value"
> data.reshape
    lab     days    value
1   lac     1       2.3
2   lac     2       3.4
3   lac     3       4.5
4   wbc     1       12.0
5   wbc     2       11.0
6   wbc     3       6.0
7   hb      1       60.0
8   hb      2       77.0
9   hb      3       89.0
```

The first line creates a data frame that can be encountered in practice. All variable names are composed of type of measurement and day, and the latter two are separated by "_". In such case, colsplit() can be used to separate the variable names into 2 columns, with each representing the type of laboratory measurement and the day. Next, values of measurements are added by cbind() function. The following lines rename the variable names to make them easy to understand.

Producing baseline characteristics of cohort automatically

In big-data clinical study, there are numerous covariates under consideration. The first step is usually to take a look at these variables one by one. Suppose you want to have a look at the mean, median, range, and standard deviation of a variable. The traditional way is to execute functions one by one. Alternatively, you can combine these functions into

a single one.

```
> round(funstofun(mean, median,min,max,sd)
(data$PaO2),1)
mean    median    min    max    sd
76.1    78.0      60.0   89.0   10.8
> round(funstofun(mean, median,min,max,sd)
(data$PcvO2),1)
mean    median    min    max    sd
38.1    37.0      30.0   47.0   6.1
```

Sometimes, the *summary()* function contained in the base R package can fulfill the task of general description of variables. However, the output parameters are fixed. The *funstofun()* function overcomes this limitation that functions can be flexibly adapted to the needs of specific purpose.

Summary

The article provides a gentle introduction to data reconstruction and aggregating. It is common that the format of data output from case report form (CRF) does not meet the purpose of statistical analysis. In clinical research, variables are frequently measured repeatedly over the follow-up period. Such data can be displayed either in wide or long format. Transformation between these 2 forms can be challenging by hand. Fortunately, there are sophisticated packages in R environment. Data frame should firstly be melted and then casted to format that you want. Aggregation over unique combination of id variable is also allowable. Additionally, the article also introduces 2 functions colsplit() and funstofun() that may be useful in some situations.

Cite this article as: Zhang Z. Reshaping and aggregating data: an introduction to reshape package. Ann Transl Med 2016;4(4):78. doi: 10.3978/j.issn.2305-5839.2016.01.33

Acknowledgements

None.

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Twisk JW. Applied longitudinal data analysis for epidemiology: a practical guide. Second edition. Cambridge, England: Cambridge University Press, 2013:321.
2. Wickham H. Reshaping data with the reshape package. Journal of Statistical Software 2007;21:1-20.

