

Introduction to Reshaping Data with R

Revolution Analytics





- 1 Long versus wide data
- 2 reshape: long to wide reshape
- 3 reshape: wide to long reshape





Overview

In this session we cover data reshaping

- Understand long vs wide data
- Use reshaping tools to convert from wide to tall





Outline

- 1 Long versus wide data
- 2 reshape: long to wide reshape
- 3 reshape: wide to long reshape





Long- versus wide-formatted data

Occasionally we may want to change data between “long” and “wide” formats.

- Wide-formatted data sets contain single records for each individual, with time-varying variables having multiple columns for each time.
- Long-formatted data has multiple rows for each individual, with only one observation per row.

Many functions in R require you to have your data in long format, e.g. `ANOVA()`. Several other software packages require wide format.





Outline

- 1 Long versus wide data
- 2 reshape: long to wide reshape
- 3 reshape: wide to long reshape





The MovieLens data

We'll use the MovieLens 100k dataset from 943 users on 1682 movies.

```
movie.data <- read.table("../data/MovieLens.data")
names(movie.data) <- c("user.id", "item.id", "rating", "timestamp")
movie.data = movie.data[order(movie.data$user.id, movie.data$item.id),
]
```



View the Data

```
head(movie.data)
```

##	user.id	item.id	rating	timestamp
## 32237	1	1	5	874965758
## 23172	1	2	3	876893171
## 83308	1	3	4	878542960
## 62632	1	4	3	876893119
## 47639	1	5	3	889751712
## 5534	1	6	5	887431973

Currently the data has a “long” format, meaning that there are multiple records for each individual corresponding to different times.





reshape to convert from long to wide

Let's say that `item.id` (rather than `timestamp`) is our 'time' variable. We can use this column to convert the data set between long and wide formats:

```
movie.data$timestamp <- NULL
df.wide <- reshape(movie.data, idvar = "user.id", timevar = "item.id",
  direction = "wide")
dim(df.wide)
```

```
## [1] 943 1683
```





View the Wide Data

```
dim(df.wide)
```

```
## [1] 943 1683
```

```
df.wide[1:3, 1:5]
```

```
##      user.id rating.1 rating.2 rating.3 rating.4
## 32237      1      5      3      4      3
## 26185      2      4      NA      NA      NA
## 37189      3     NA     NA     NA     NA
```



Exercise: reshape long to wide

- Reshape the Indometh conc (concentration) variable from long format to wide format, with `idvar = Subject`.

Indometh is a dataset characterizes the pharmacokinetics of Indomethacin.

```
str(Indometh)
```

```
## Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame': 66 obs. of 3 variables:
## $ Subject: Ord.factor w/ 6 levels "1"<"4"<"2"<"5"<...: 1 1 1 1 1 1 1 1 1 1 1 ...
## $ time : num 0.25 0.5 0.75 1 1.25 2 3 4 5 6 ...
## $ conc : num 1.5 0.94 0.78 0.48 0.37 0.19 0.12 0.11 0.08 0.07 ...
## - attr(*, "formula")=Class 'formula' length 3 conc ~ time | Subject
## .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
## - attr(*, "labels")=List of 2
```

...



Outline

- 1 Long versus wide data
- 2 reshape: long to wide reshape
- 3 reshape: wide to long reshape





What if we want to go the other direction?

- same command: `reshape`
- same first argument: input data set
- use `direction = "long"`





Additional arguments for wide to long:

varying a vector of variable names that define the observations that you want to put into long format (the variables that contain the scores).

timevar the name of the variable in the new long-format dataset that will take on the character string names of the variable in varying.

times the values the “score” variable will have, specified as a vector of variable names. This is frequently the same as varying. This is important for the score values.

v.names the name we wish to give the variable containing the values taken by timevar in the new long-format dataset





Example:

Convert the cars data set from wide format to long format.

```
cars.long <- reshape(cars, direction = "long", varying = c("speed",  
  "dist"), timevar = "measuretype", times = c("speed", "dist"),  
  v.names = "measureval")  
head(cars.long)
```

##	measuretype	measureval	id
## 1.speed	speed	4	1
## 2.speed	speed	4	2
## 3.speed	speed	7	3
## 4.speed	speed	7	4
## 5.speed	speed	8	5
## 6.speed	speed	9	6

The id variable is taken from the row number.



Exercise 2: wide to long

- Reshape the mtcars data set from wide format to long format (Hint: Use `names(mtcars)` to specify variables that need to be moved to long format).
- ADVANCED: Look at `help(reshape)` and try to get the “id” variable in your output to correspond to the car name.
- ADVANCED: Reshape the following data from wide to long. Instead of having a separate column for each score for read, write, math, science, and socst. Create a new variable `ContentArea` which will be a character string taking on these subject names. Create a new variable `score` which will take on the value of each subject.





Exercise data set

```
hsb2 <- read.csv("http://www.ats.ucla.edu/stat/r/faq/hsb2.csv")
str(hsb2)
```

```
## 'data.frame':    200 obs. of  11 variables:
## $ id      : int  70 121 86 141 172 113 50 11 84 48 ...
## $ female  : int  0 1 0 0 0 0 0 0 0 0 ...
## $ race    : int  4 4 4 4 4 4 3 1 4 3 ...
## $ ses     : int  1 2 3 3 2 2 2 2 2 2 ...
## $ schtyp  : int  1 1 1 1 1 1 1 1 1 1 ...
## $ prog    : int  1 3 1 3 2 2 1 2 1 2 ...
...

```





Exercise Help

Your finished product should look like this. Note two columns introduced: subj and score.

##	id	female	race	ses	scht	typ	prog	subj	score
## 1	70	0	4	1	1		1	read	57
## 2	121	1	4	2	1		3	read	68
## 3	86	0	4	3	1		1	read	44
## 4	141	0	4	3	1		3	read	63
## 5	172	0	4	2	1		2	read	47
## 6	113	0	4	2	1		2	read	44





Exercise Help 2

In addition to `data = hsb2` and `direction = "long"`, you will need to specify the following arguments to reshape:

pertaining to `subj` column:

varying a vector of variable names that define the metric.

timevar the name of the new variable which will take on the character string names of the variable in `varying`.





Exercise Help 3

pertaining to score column:

- times** the values this variable will have, specified as a vector of variable names. In this case the same as varying – same vector of variable names that define the metric.
- v.names** the name we wish to give the variable containing the values taken by timevar



Thank you

Revolution Analytics is the leading commercial provider of software and support for the popular open source R statistics language.

www.revolutionanalytics.com

1.855.GET.REVO

Twitter: @RevolutionR

