



Reshaping data

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

The goal is tidy data

ID	X	Y	C	D	E	F	G	H	I
2	3	408	11	1207119912	1207120013	2821 A			
1	2	195	11	1207119911	1207120012	2821 B			
4	5	571	11	1207119911	1207120012	2821 C			
3	6	22	11	1207119912	1207120013	2821 D			
6	5	273	11	1207119912	1207120013	2821 E			
7	9	192	11	1207119912	1207120013	2821 F			
8	9	309	11	1207119911	1207120012	2821 G			
10	9	72	11	1207119911	1207120012	2821 H			
11	46	266	11	1207119911	1207120012	2821 I			
12	11	245	11	1207119911	1207120012	2821 J			
13	24	255	11	1207119911	1207120012	2821 K			
14	13	255	11	1207119911	1207120012	2821 L			
15	5	235	11	1207119911	1207120012	2821 M			
16	16	161	11	1207119912	1207120013	2821 N			
17	16	161	11	1207119912	1207120013	2821 O			
18	15	173	11	1207119912	1207120013	2821 P			
19	18	472	11	1207119912	1207120013	2821 Q			
20	16	41	11	1207119912	1207120013	2821 R			
21	20	282	11	1207119914	1207120012	2821 S			
22	21	218	11	1207119915	1207120012	2821 T			
23	15	25	11	1207119915	1207120012	2821 U			
24	23	362	11	1207119912	1207120013	2824 V			
25	24	224	11	1207119912	1207120013	2824 W			
26	25	260	11	1207119913	1207120012	2824 X			
27	26	409	11	1207119913	1207120012	2824 Y			
28	28	51	11	1207119913	1207120012	2825 Z			
29	29	22	11	1207119913	1207120012	2825 C			
30	23	24	11	1207119913	1207120012	2825 D			
31	30	503	11	1207119913	1207120012	2825 E			
32	24	11	11	1207119912	1207120013	2821 A			
33	24	313	11	1207119912	1207120013	2821 B			
34	15	365	11	1207119912	1207120013	1619 Y			
35	23	503	11	1207119912	1207120013	1619 Z			
36	39	29	11	1207119912	1207120013	1646 Z			
37	37	26	11	1207119912	1207120013	1647 Y			
38	27	267	11	1207119912	1207120013	1677 Z			
39	38	257	11	1207119912	1207120013	1692 Z			
40	38	157	11	1207119912	1207120013	1693 Z			
41	40	342	11	1207119912	1207120013	1695 A			
42	41	290	11	1207119912	1207120013	1697 Z			

1. Each variable forms a column
2. Each observation forms a row
3. Each table/file stores data about one kind of observation (e.g. people/hospitals).

<http://vita.had.co.nz/papers/tidy-data.pdf>

[Leek, Taub, and Pineda 2011 PLoS One](#)

Start with reshaping

```
library(reshape2)  
head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

Melting data frames

```
mtcars$carname <- rownames(mtcars)
carMelt <- melt(mtcars,id=c("carname","gear","cyl"),measure.vars=c("mpg","hp"))
head(carMelt,n=3)
```

	carname	gear	cyl	variable	value
1	Mazda RX4	4	6	mpg	21.0
2	Mazda RX4 Wag	4	6	mpg	21.0
3	Datsun 710	4	4	mpg	22.8

```
tail(carMelt,n=3)
```

	carname	gear	cyl	variable	value
62	Ferrari Dino	5	6	hp	175
63	Maserati Bora	5	8	hp	335
64	Volvo 142E	4	4	hp	109

<http://www.statmethods.net/management/reshape.html>

Casting data frames

```
cylData <- dcast(carMelt, cyl ~ variable)
cylData
```

```
 cyl mpg hp
1   4 11 11
2   6  7  7
3   8 14 14
```

```
cylData <- dcast(carMelt, cyl ~ variable,mean)
cylData
```

```
 cyl    mpg      hp
1   4 26.66  82.64
2   6 19.74 122.29
3   8 15.10 209.21
```

<http://www.statmethods.net/management/reshape.html>

Averaging values

```
head(InsectSprays)
```

```
count spray
1    10     A
2     7     A
3    20     A
4    14     A
5    14     A
6    12     A
```

```
tapply(InsectSprays$count, InsectSprays$spray, sum)
```

```
A   B   C   D   E   F
174 184  25  59  42 200
```

<http://www.r-bloggers.com/a-quick-primer-on-split-apply-combine-problems/>

Another way - split

```
spIns = split(InsectSprays$count, InsectSprays$spray)  
spIns
```

```
$A  
[1] 10 7 20 14 14 12 10 23 17 20 14 13  
  
$B  
[1] 11 17 21 11 16 14 17 17 19 21 7 13  
  
$C  
[1] 0 1 7 2 3 1 2 1 3 0 1 4  
  
$D  
[1] 3 5 12 6 4 3 5 5 5 2 4  
  
$E  
[1] 3 5 3 5 3 6 1 1 3 2 6 4  
  
$F  
[1] 11 9 15 22 15 16 13 10 26 26 24 13
```

Another way - apply

```
sprCount = lapply(spIns,sum)  
sprCount
```

```
$A  
[1] 174
```

```
$B  
[1] 184
```

```
$C  
[1] 25
```

```
$D  
[1] 59
```

```
$E  
[1] 42
```

```
$F  
[1] 200
```

Another way - combine

```
unlist(sprCount)
```

A	B	C	D	E	F
174	184	25	59	42	200

```
sapply(spIns,sum)
```

A	B	C	D	E	F
174	184	25	59	42	200

Another way - plyr package

```
ddply(InsectSprays,.(spray),summarize,sum=sum(count))
```

```
spray sum
1      A 174
2      B 184
3      C  25
4      D  59
5      E  42
6      F 200
```

Creating a new variable

```
spraySums <- ddply(InsectSprays,.spray,summarize,sum=ave(count,FUN=sum))  
dim(spraySums)
```

```
[1] 72 2
```

```
head(spraySums)
```

```
spray sum  
1 A 174  
2 A 174  
3 A 174  
4 A 174  
5 A 174  
6 A 174
```

More information

- A tutorial from the developer of plyr - <http://plyr.had.co.nz/09-user/>
- A nice reshape tutorial <http://www.slideshare.net/jeffreybreen/reshaping-data-in-r>
- A good plyr primer - <http://www.r-bloggers.com/a-quick-primer-on-split-apply-combine-problems/>
- See also the functions
 - acast - for casting as multi-dimensional arrays
 - arrange - for faster reordering without using order() commands
 - mutate - adding new variables