



Reshaping data

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

The goal is tidy data

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	1	498	12	1207129125	1207129024	2282	A									
2	2	120	11	10711944	102129024	2301	Y									
3	3	1813	10	10711944	102129024	2301	Y									
4	4	12	12	1207129125	1207129024	2282	B									
5	5	2731	10	10711944	102129024	2301	X									
6	6	181	12	1207129125	1207129024	2282	B									
7	7	1813	10	10711944	102129024	2301	B									
8	8	181	12	1207129125	1207129024	2282	B									
9	9	72	12	1207129125	1207129024	2282	C									
10	10	340	11	10711944	102129024	2301	B									
11	11	498	11	10711944	102129024	2301	B									
12	12	287	12	1207129125	1207129024	2282	A									
13	13	523	10	10711944	102129024	2301	Y									
14	14	232	12	1207129125	1207129024	2282	C									
15	15	344	10	1207129125	1207129024	2282	B									
16	16	181	12	1207129125	1207129024	2282	B									
17	17	1813	10	10711944	102129024	2301	B									
18	18	474	12	1207129125	1207129024	2282	A									
19	19	41	10	10711944	102129024	2301	C									
20	20	287	12	1207129125	1207129024	2282	B									
21	21	413	10	10711944	102129024	2301	B									
22	22	181	12	1207129125	1207129024	2282	B									
23	23	344	10	1207129125	1207129024	2282	B									
24	24	181	12	1207129125	1207129024	2282	B									
25	25	287	12	1207129125	1207129024	2282	B									
26	26	498	12	1207129125	1207129024	2282	B									
27	27	91	10	10711944	102129024	2301	C									
28	28	32	10	10711944	102129024	2301	C									
29	29	24	12	1207129125	1207129024	2282	B									
30	30	181	10	10711944	102129024	2301	B									
31	31	91	10	10711944	102129024	2301	C									
32	32	313	12	1207129125	1207129024	2282	B									
33	33	381	10	10711944	102129024	2301	B									
34	34	181	12	1207129125	1207129024	2282	B									
35	35	24	12	1207129125	1207129024	2282	B									
36	36	72	12	1207129125	1207129024	2282	B									
37	37	381	10	10711944	102129024	2301	B									
38	38	247	12	1207129125	1207129024	2282	B									
39	39	237	12	1207129125	1207129024	2282	B									
40	40	181	10	10711944	102129024	2301	B									
41	41	344	10	1207129125	1207129024	2282	A									
42	42	273	12	1207129125	1207129024	2282	A									

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  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