How can I “collapse” my data in R?

Authored by

stats writer

June 30, 2024

"Collapsing data in R refers to the process of summarizing or aggregating a large dataset into a smaller and more manageable form. This can be achieved by using functions such as aggregate, tapply, or dplyr's group_by and summarize. By collapsing data, users can gain a better understanding of their data and identify patterns or trends. It is a useful technique for data exploration and analysis in R."

How can I "collapse" my data in R? | R FAQ

**Users of Stata are likely familiar with the concept of collapsing data:
reducing the number of observations in your data, keeping one observation for
each value or combination of values from one or more variables and calculating
some summary of the other variables at this level. For examples of this in Stata,
see our Stata Learning Module on collapse.**

**This data management step can also be done in R using the `summaryBy()`
command in the `"doBy"` package. At the end we included one example using function `collap()` from new package `"collapse"`. Let's consider the dataset hsb2. We
can start by collapsing the data by prog and calculating the mean of
socst. The default summary calculation is the mean, as**

indicated by the
output.

```
library(doBy)
hsb2 <-
read.table("https://stats.idre.ucla.edu/wp-content/uploads/2016/02/hsb2-1.csv", header=T, sep=",")
attach(hsb2)
summaryBy(socst ~ prog, data=hsb2)

prog socst.mean
1 1 50.60000
2 2 56.69524
3 3 45.02000
```

We can easily make our collapse more complex, creating one observation for each combination of prog and female and ses, calculating both the mean and standard deviation of several variables, and saving this as a new object.

```
collapse1 <- summaryBy(socst + math ~ prog + ses + female, FUN=c(mean,sd), data=hsb2)
collapse1
```

```
prog ses female socst.mean math.mean socst.sd math.sd
1 1 1 0 47.57143 46.71429 6.502747 8.118175
2 1 1 1 49.00000 48.22222 5.894913 7.546154
3 1 2 0 50.50000 51.10000 8.959787 9.267026
4 1 2 1 52.10000 51.10000 9.938142 6.026792
5 1 3 0 57.25000 54.00000 17.500000 4.320494
6 1 3 1 49.60000 50.40000 10.714476 7.700649
7 2 1 0 46.75000 50.00000 11.926860 6.683313
8 2 1 1 55.13333 55.00000 10.098562 10.281745
9 2 2 0 55.54545 58.13636 8.985318 10.086745
10 2 2 1 55.40909 54.50000 8.511388 7.255540
11 2 3 0 59.33333 57.42857 7.799573 7.743200
12 2 3 1 59.61905 59.42857 9.058014 8.152125
13 3 1 0 32.50000 46.75000 4.725816 5.251984
14 3 1 1 38.25000 42.25000 8.189715 3.918819
15 3 2 0 44.00000 48.20000 11.464230 9.540889
16 3 2 1 51.00000 46.06250 8.755950 7.540723
17 3 3 0 50.25000 42.25000 6.946222 3.947573
18 3 3 1 46.00000 55.66667 10.000000 10.503968
```

If we wish to summarize our data in a way that does not already exist as a function, we can write the function and then pass this to

summaryBy.

Below, we write a quick function halfmean and then apply it to math

and socst within each combination of prog, ses, and female.

```
halfmean <- function(x) return(mean(x)/2)
collapse2 <- summaryBy(socst + math ~ prog + ses + female, FUN=halfmean, data=hsb2)
collapse2
prog ses female socst.halfmean math.halfmean
1 1 1 0 23.78571 23.35714
2 1 1 1 24.50000 24.11111
3 1 2 0 25.25000 25.55000
4 1 2 1 26.05000 25.55000
5 1 3 0 28.62500 27.00000
6 1 3 1 24.80000 25.20000
7 2 1 0 23.37500 25.00000
8 2 1 1 27.56667 27.50000
9 2 2 0 27.77273 29.06818
10 2 2 1 27.70455 27.25000
11 2 3 0 29.66667 28.71429
12 2 3 1 29.80952 29.71429
13 3 1 0 16.25000 23.37500
```

14 3 1 1 19.12500 21.12500
15 3 2 0 22.00000 24.10000
16 3 2 1 25.50000 23.03125
17 3 3 0 25.12500 21.12500
18 3 3 1 23.00000 27.83333

We can also use function `collap()` in the new R package `"collapse"` by Sebastian Krantz which is made compatible with `collapse`'s Fast Statistical Functions, allowing for extremely fast conventional and weighted aggregation.

```
library(collapse)
collapse3 <- collap(hsb2, socst + math ~ prog + ses + female, FUN = list(fmean, fsd))
collapse3
female ses prog fmean.math fsd.math fmean.socst fsd.socst
1 0 1 1 46.71429 8.118175 47.57143 6.502747
2 1 1 1 48.22222 7.546154 49.00000 5.894913
3 0 2 1 51.10000 9.267026 50.50000 8.959787
4 1 2 1 51.10000 6.026792 52.10000 9.938142
5 0 3 1 54.00000 4.320494 57.25000 17.500000
6 1 3 1 50.40000 7.700649 49.60000 10.714476
```

7 0 1 2 50.00000 6.683313 46.75000 11.926860

8 1 1 2 55.00000 10.281745 55.13333 10.098562

9 0 2 2 58.13636 10.086745 55.54545 8.985318

10 1 2 2 54.50000 7.255540 55.40909 8.511388

11 0 3 2 57.42857 7.743200 59.33333 7.799573

12 1 3 2 59.42857 8.152125 59.61905 9.058014

13 0 1 3 46.75000 5.251984 32.50000 4.725816

14 1 1 3 42.25000 3.918819 38.25000 8.189715

15 0 2 3 48.20000 9.540889 44.00000 11.464230

16 1 2 3 46.06250 7.540723 51.00000 8.755950

17 0 3 3 42.25000 3.947573 50.25000 6.946222

18 1 3 3 55.66667 10.503968 46.00000 10.000000