Introduction to R: **Aggregation**

Day 2, Part C





In this lecture

- 1. Aggregation functions
- 2. Aggregating with dcast()
- 3. Aggregation vs reshape with dcast()

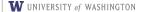




Aggregating (or collapsing) data refers to taking multiple rows (observations) and combining them into a single row according to some function (e.g., sum, or mean, or median, etc.).

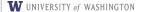
There are a number of different functions in base R that are explicitly for aggregating, e.g.:

- apply()
- tapply()
- aggregate()



Because this is such a common task, and potentially a quite complicated one, there are also a number of other packages that provide additional functions.

- dcast() in the reshape2 library can be used to aggregate, in addition to being used to reshape. It's one of the easiest functions to work with for simple aggregations, so this is what we will do today.
- We won't do more complicated aggregations at this point, but when you
 get there, the functions in the plyr and dplyr libraries are very useful
 when working with data frames.
- The data.table library is also useful for aggregations this library defines
 a whole new data type and syntax, so there's a fairly steep learning curve,
 but it is very speedy for big data, so worth the effort.



In all the examples using dcast() so far, we were careful to include all identifying variables in the casting formula such that these variables uniquely identified one single data value.



In all the examples using dcast() so far, we were careful to include all identifying variables in the casting formula such that these variables uniquely identified one single data value.

If you don't include all identifying variables, something different happens:

```
> head(long, 2)
 cntv vear sex variable value
1 King 2010 1
                  рор 965486
2 King 2010 2 pop 971999
> dcast(long, cntv + sex ~ variable, value,var = "value")
Aggregation function missing: defaulting to length
     cnty sex pop deaths
1 King 1 4
    King 2 4
3 Pierce 1 4
  Pierce 2 4
5 Snohomish 1 4
6 Snohomish 2 4
```

The variable that we didn't include (year) has been dropped. The four values present for each combination of cnty, sex, and variable (one for each of the four years) have been aggregated, in this case using the default function, length(), since no aggregation function was specified.





We can aggregate using a wide range of functions:

```
> dcast(long, cnty + sex ~ variable, value.var = "value", fun.aggregate = sum)
      cntv sex
                 pop deaths
    King 1 3971435 24155
    King 2 3989386 24464
   Pierce 1 1600737 11904
   Pierce 2 1629950 11241
5 Snohomish 1 1460946 9319
6 Snohomish 2 1455889 9374
> dcast(long, cnty + sex ~ variable, value.var = "value", fun.aggregate = mean)
      cnty sex
                 pop deaths
    King 1 992858.8 6038.75
  King 2 997346.5 6116.00
  Pierce 1 400184.2 2976.00
   Pierce 2 407487.5 2810.25
5 Snohomish 1 365236.5 2329.75
6 Snohomish 2 363972.2 2343.50
> dcast(long, cnty + sex ~ variable, value.var="value", fun.aggregate=quantile, p=0.25)
      cntv sex
                   pop deaths
   King 1 978914.8 5951.50
    King 2 983941.2 6058.50
   Pierce 1 396580.0 2931.25
   Pierce 2 404838.8 2756.75
5 Snohomish 1 360971.0 2284.00
6 Snohomish 2 359777.0 2324.00
```





Aggregation

And by changing the casting formula, we can change what is aggregated over and how the final data are shaped:

```
> dcast(long, cnty ~ variable, value.var = "value", fun.aggregate = sum)
       cnty
             pop deaths
     King 7960821 48619
    Pierce 3230687 23145
3 Snohomish 2916835 18693
> dcast(long, variable ~ cnty, value.var = "value", fun.aggregate = sum)
  variable King Pierce Snohomish
      pop 7960821 3230687 2916835
  deaths 48619 23145
                              18693
> dcast(long, . ~ variable, value.var = "value", fun.aggregate = sum)
        pop deaths
1 . 14108343 90457
> dcast(long, variable ~ ., value.var = "value", fun.aggregate = sum)
  variable
      pop 14108343
2 deaths 90457
```





Reshaping vs aggregating

Remember, dcast() can do two (very) different things, depending on how it's used:

- reshape your data
- aggregate (collapse) your data

If the variables in the casting formula uniquely identify each data value, dcast() will **reshape**.

If not, dcast() will aggregate (and possibly reshape at the same time).

