An introduction to the `reshape2` package

BISC 888–1 Simon Fraser University

Sean C. Anderson
sean@seananderson.ca

October 2, 2013

`reshape2` is an `R` package written by Hadley Wickham that makes it easy to transform data between "wide" and "long" format. It is based around two key functions: `melt` and `cast`:

`melt` takes wide-format data and melts it into long-format data.

`cast` takes long-format data and casts it into wide-format data.

Think of working with metal: if you melt metal, it drips and becomes long. If you cast it into a mold, it becomes wide.

It turns out that you need wide-format data for some types of data analysis and long-format data for others. In reality, you need long-format data much more commonly than wide-format data. For example, `ggplot2` requires long-format data, `plyr` requires long-format data, and most modelling functions (such as `lm()`, `glm()`, and `gam()`) require long-format data. But people often find it easier to record their data in wide format.

1 What makes data wide or long?

Wide data has a column for each variable. For example, this is wide-format data:

```
> head(d)
```

```
     ozone      wind     temp
1 23.61538 11.622581 65.54839
2 29.44444 10.266667 79.10000
3 59.11538  8.941935 83.90323
4 59.96154  8.793548 83.96774
```

And this is long-format data:

```
> melt(d)
```

```
   variable     value
1     ozone 23.615385
2     ozone 29.444444
3     ozone 59.115385
4     ozone 59.961538
5      wind 11.622581
6      wind 10.266667
7      wind  8.941935
8      wind  8.793548
9      temp 65.548387
10     temp 79.100000
11     temp 83.903226
12     temp 83.967742
```

2 Wide- to long-format data: the `melt` function

For this example we'll work with the `airquality` dataset that is built into `R`. First we'll change the column names to lower case to make them easier to work with. Then we'll look at the data:

```
> names(airquality) <- tolower(names(airquality))
> head(airquality)

  ozone solar.r wind temp month day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
```

What happens if we run the function `melt` with all the default argument values?

```
> airquality_long <- melt(airquality)
> head(airquality_long)

  variable value
1    ozone    41
2    ozone    36
3    ozone    12
4    ozone    18
5    ozone    NA
6    ozone    28

> tail(airquality_long)

    variable value
913      day    25
914      day    26
915      day    27
916      day    28
917      day    29
918      day    30
```

By default `melt` has assumed that all columns with numeric values are variables with values. Often this is what you want. Maybe here we want to know the values of `ozone`, `solar.r`, `wind`, and `temp` for each `month` and `day`. We can do that with `melt` by telling it that we want `month` and `day` to be "ID variables":

```
> airquality_long <- melt(airquality, id.vars = c("month", "day"))
> head(airquality_long)
```

```
  month day variable value
1     5   1    ozone    41
2     5   2    ozone    36
3     5   3    ozone    12
4     5   4    ozone    18
5     5   5    ozone    NA
6     5   6    ozone    28
```

What if we wanted to control the column names in our long-format data? `melt` let's us set those too all in one step:

```
> airquality_long <- melt(airquality, id.vars = c("month", "day"),
+   variable.name = "climate_variable", value.name = "climate_value")
> head(airquality_long)
```

```
  month day climate_variable climate_value
1     5   1            ozone            41
2     5   2            ozone            36
3     5   3            ozone            12
4     5   4            ozone            18
5     5   5            ozone            NA
6     5   6            ozone            28
```

That's about all there is to `melt`!

3 Long- to wide-format data: the `cast` functions

Whereas going from wide- to long-format data is pretty straightforward, going from long- to wide-format data can take a bit more thought. It usually involves some head scratching and some trial and error for all but the simplest cases. Let's go through some examples.