7.7: Reshaping a Data Frame

One of the most annoying tasks that you need to undertake on a regular basis is that of reshaping a data frame. Framed in the most general way, reshaping the data means taking the data in whatever format it's given to you, and converting it to the format you need it. Of course, if we're going to characterise the problem that broadly, then about half of this chapter can probably be thought of as a kind of reshaping. So we're going to have to narrow things down a little bit. To that end, I'll talk about a few different tools that you can use for a few different tasks. In particular, I'll discuss a couple of easy to use (but limited) functions that I've included in the `lsr` package. In future versions of the book I plan to expand this discussion to include some of the more powerful tools that are available in R, but I haven't had the time to do so yet.

7.7.1 Long form and wide form data

The most common format in which you might obtain data is as a "case by variable" layout, commonly known as the ***wide form*** of the data.

```
load("./rbook-master/data/repeated.Rdata")
who()
```

```
##    -- Name --          -- Class --    -- Size --
##    age                 numeric        11
##    age.breaks          numeric        4
##    age.group           factor         11
##    age.group2          factor         11
##    age.group3          factor         11
##    age.labels          character      3
##    cake.1              numeric        5
##    cake.2              numeric        5
##    cake.df             data.frame     5 x 2
##    cake.mat1           matrix         5 x 2
##    cake.mat2           matrix         2 x 5
##    cakes               matrix         4 x 5
##    cakes.flipped       matrix         5 x 4
##    choice              data.frame     4 x 10
##    df                  data.frame     4 x 1
##    drugs               data.frame     10 x 8
##    fac                 factor         3
##    fibonacci           numeric        6
##    garden              data.frame     5 x 3
##    is.MP.speaking      logical        5
##    itng                data.frame     10 x 2
##    itng.table          table          3 x 4
##    likert.centred      numeric        10
##    likert.raw          numeric        10
##    makka.pakka         character      4
##    numbers             numeric        3
##    opinion.dir         numeric        10
##    opinion.strength    numeric        10
##    some.data           numeric        18
##    speaker             character      10
##    speech.by.char      list           3
##    text                character      3
##    tombliboo           character      2
##    upsy.daisy          character      4
##    utterance           character      10
```

To get a sense of what I'm talking about, consider an experiment in which we are interested in the different effects that alcohol and and caffeine have on people's working memory capacity (WMC) and reaction times (RT). We recruit 10 participants, and measure their WMC and RT under three different conditions: a "no drug" condition, in which they are not under the influence of either caffeine or alcohol, a "caffeine" condition, in which they are under the inflence of caffeine, and an "alcohol" condition, in which… well, you can probably guess. Ideally, I suppose, there would be a fourth condition in which both drugs are administered, but for the sake of simplicity let's ignore that. The `drugs` data frame gives you a sense of what kind of data you might observe in an experiment like this:

```
drugs
```

```
##    id gender WMC_alcohol WMC_caffeine WMC_no.drug RT_alcohol RT_caffeine
## 1   1 female         3.7          3.7         3.9        488         236
## 2   2 female         6.4          7.3         7.9        607         376
## 3   3 female         4.6          7.4         7.3        643         226
## 4   4   male         6.4          7.8         8.2        684         206
## 5   5 female         4.9          5.2         7.0        593         262
## 6   6   male         5.4          6.6         7.2        492         230
## 7   7   male         7.9          7.9         8.9        690         259
## 8   8   male         4.1          5.9         4.5        486         230
## 9   9 female         5.2          6.2         7.2        686         273
## 10 10 female         6.2          7.4         7.8        645         240
##    RT_no.drug
## 1         371
## 2         349
## 3         412
## 4         252
## 5         439
## 6         464
## 7         327
## 8         305
## 9         327
## 10        498
```

This is a data set in "wide form", in which each participant corresponds to a single row. We have two variables that are characteristics of the subject (i.e., their `id` number and their `gender`) and six variables that refer to one of the two measured variables (WMC or RT) in one of the three testing conditions (alcohol, caffeine or no drug). Because all of the testing conditions (i.e., the three drug types) are applied to all participants, drug type is an example of a ***within-subject factor***.

7.7.2 Reshaping data using `wideToLong()`

The "wide form" of this data set is useful for some situations: it is often very useful to have each row correspond to a single subject. However, it is not the only way in which you might want to organise this data. For instance, you might want to have a separate row for each "testing occasion". That is, "participant 1 under the influence of alcohol" would be one row, and "participant 1 under the influence of caffeine" would be another row. This way of organising the data is generally referred to as the ***long form*** of the data. It's not too difficult to switch between wide and long form, and I'll explain how it works in a moment; for now, let's just have a look at what the long form of this data set looks like:

```
drugs.2 <- wideToLong( data = drugs, within = "drug" )
head(drugs.2)
```

```
##   id gender    drug WMC  RT
## 1  1 female alcohol 3.7 488
## 2  2 female alcohol 6.4 607
## 3  3 female alcohol 4.6 643
## 4  4   male alcohol 6.4 684
## 5  5 female alcohol 4.9 593
## 6  6   male alcohol 5.4 492
```

The `drugs.2` data frame that we just created has 30 rows: each of the 10 participants appears in three separate rows, one corresponding to each of the three testing conditions. And instead of having a variable like `WMC_caffeine` that indicates that we were measuring "WMC" in the "caffeine" condition, this information is now recorded in two separate variables, one called `drug` and another called `WMC`. Obviously, the long and wide forms of the data contain the same information, but they

represent quite different ways of organising that information. Sometimes you find yourself needing to analyse data in wide form, and sometimes you find that you need long form. So it's really useful to know how to switch between the two.

In the example I gave above, I used a function called `wideToLong()` to do the transformation. The `wideToLong()` function is part of the `lsr` package. The key to understanding this function is that it relies on the *variable names* to do all the work. Notice that the variable names in the `drugs` data frame follow a very clear scheme. Whenever you have a variable with a name like `WMC_caffeine` you know that the variable being measured is "WMC", and that the specific condition in which it is being measured is the "caffeine" condition. Similarly, you know that `RT_no.drug` refers to the "RT" variable measured in the "no drug" condition. The measured variable comes first (e.g., `WMC`), followed by a separator character (in this case the separator is an underscore, `_`), and then the name of the condition in which it is being measured (e.g., `caffeine`). There are two different prefixes (i.e, the strings before the separator, `WMC`, `RT`) which means that there are two separate variables being measured. There are three different suffixes (i.e., the strings after the separtator, `caffeine`, `alcohol`, `no.drug`) meaning that there are three different levels of the within-subject factor. Finally, notice that the separator string (i.e., `_`) does not appear anywhere in two of the variables (`id`, `gender`), indicating that these are ***between-subject*** variables, namely variables that do not vary within participant (e.g., a person's `gender` is the same regardless of whether they're under the influence of alcohol, caffeine etc).

Because of the fact that the variable naming scheme here is so informative, it's quite possible to reshape the data frame without any additional input from the user. For example, in this particular case, you could just type the following:

```
wideToLong( drugs )
```

```
##    id gender   within WMC  RT
## 1   1 female  alcohol 3.7 488
## 2   2 female  alcohol 6.4 607
## 3   3 female  alcohol 4.6 643
## 4   4   male  alcohol 6.4 684
## 5   5 female  alcohol 4.9 593
## 6   6   male  alcohol 5.4 492
## 7   7   male  alcohol 7.9 690
## 8   8   male  alcohol 4.1 486
## 9   9 female  alcohol 5.2 686
## 10 10 female  alcohol 6.2 645
## 11  1 female caffeine 3.7 236
## 12  2 female caffeine 7.3 376
## 13  3 female caffeine 7.4 226
## 14  4   male caffeine 7.8 206
## 15  5 female caffeine 5.2 262
## 16  6   male caffeine 6.6 230
## 17  7   male caffeine 7.9 259
## 18  8   male caffeine 5.9 230
## 19  9 female caffeine 6.2 273
## 20 10 female caffeine 7.4 240
## 21  1 female  no.drug 3.9 371
## 22  2 female  no.drug 7.9 349
## 23  3 female  no.drug 7.3 412
## 24  4   male  no.drug 8.2 252
## 25  5 female  no.drug 7.0 439
## 26  6   male  no.drug 7.2 464
## 27  7   male  no.drug 8.9 327
## 28  8   male  no.drug 4.5 305
## 29  9 female  no.drug 7.2 327
## 30 10 female  no.drug 7.8 498
```

This is pretty good, actually. The only think it has gotten wrong here is that it doesn't know what name to assign to the within-subject factor, so instaed of calling it something sensible like `drug`, it has use the unimaginative name `within`. If you want to ensure that the `wideToLong()` function applies a sensible name, you have to specify the `within` argument, which is just a character string that specifies the name of the within-subject factor. So when I used this command earlier,

```
drugs.2 <- wideToLong( data = drugs, within = "drug" )
```

all I was doing was telling R to use `drug` as the name of the within subject factor.

Now, as I was hinting earlier, the `wideToLong()` function is very inflexible. It *requires* that the variable names all follow this naming scheme that I outlined earlier. If you don't follow this naming scheme it won't work.[120] The only flexibility that I've included here is that you can change the separator character by specifying the `sep` argument. For instance, if you were using variable names of the form `WMC/caffeine`, for instance, you could specify that `sep="/"`, using a command like this

```
drugs.2 <- wideToLong( data = drugs, within = "drug", sep = "/" )
```

and it would still work.

7.7.3 Reshaping data using `longToWide()`

To convert data from long form to wide form, the `lsr` package also includes a function called `longToWide()`. Recall from earlier that the long form of the data (i.e., the `drugs.2` data frame) contains variables named `id`, `gender`, `drug`, `WMC` and `RT`. In order to convert from long form to wide form, all you need to do is indicate which of these variables are measured separately for each condition (i.e., `WMC` and `RT`), and which variable is the within-subject factor that specifies the condition (i.e., `drug`). You do this via a two-sided formula, in which the measured variables are on the left hand side, and the within-subject factor is on the ritght hand side. In this case, the formula would be `WMC + RT ~ drug`. So the command that we would use might look like this:

```
longToWide( data=drugs.2, formula= WMC+RT ~ drug )
```

```
##    id gender WMC_alcohol RT_alcohol WMC_caffeine RT_caffeine WMC_no.drug
## 1   1 female         3.7        488          3.7         236         3.9
## 2   2 female         6.4        607          7.3         376         7.9
## 3   3 female         4.6        643          7.4         226         7.3
## 4   4   male         6.4        684          7.8         206         8.2
## 5   5 female         4.9        593          5.2         262         7.0
## 6   6   male         5.4        492          6.6         230         7.2
## 7   7   male         7.9        690          7.9         259         8.9
## 8   8   male         4.1        486          5.9         230         4.5
## 9   9 female         5.2        686          6.2         273         7.2
## 10 10 female         6.2        645          7.4         240         7.8
##    RT_no.drug
## 1         371
## 2         349
## 3         412
## 4         252
## 5         439
## 6         464
## 7         327
## 8         305
## 9         327
## 10        498
```

or, if we chose to omit argument names, we could simplify it to this:

```
longToWide( drugs.2, WMC+RT ~ drug )
```

```
##    id gender WMC_alcohol RT_alcohol WMC_caffeine RT_caffeine WMC_no.drug
## 1   1 female         3.7        488          3.7         236         3.9
## 2   2 female         6.4        607          7.3         376         7.9
## 3   3 female         4.6        643          7.4         226         7.3
## 4   4   male         6.4        684          7.8         206         8.2
## 5   5 female         4.9        593          5.2         262         7.0
## 6   6   male         5.4        492          6.6         230         7.2
## 7   7   male         7.9        690          7.9         259         8.9
## 8   8   male         4.1        486          5.9         230         4.5
## 9   9 female         5.2        686          6.2         273         7.2
## 10 10 female         6.2        645          7.4         240         7.8
##    RT_no.drug
## 1         371
## 2         349
## 3         412
## 4         252
## 5         439
## 6         464
## 7         327
## 8         305
## 9         327
## 10        498
```

Note that, just like the `wideToLong()` function, the `longToWide()` function allows you to override the default separator character. For instance, if the command I used had been

```
longToWide( drugs.2, WMC+RT ~ drug, sep="/" )
```

```
##    id gender WMC/alcohol RT/alcohol WMC/caffeine RT/caffeine WMC/no.drug
## 1   1 female         3.7        488          3.7         236         3.9
## 2   2 female         6.4        607          7.3         376         7.9
## 3   3 female         4.6        643          7.4         226         7.3
## 4   4   male         6.4        684          7.8         206         8.2
## 5   5 female         4.9        593          5.2         262         7.0
## 6   6   male         5.4        492          6.6         230         7.2
## 7   7   male         7.9        690          7.9         259         8.9
## 8   8   male         4.1        486          5.9         230         4.5
## 9   9 female         5.2        686          6.2         273         7.2
## 10 10 female         6.2        645          7.4         240         7.8
##    RT/no.drug
## 1         371
## 2         349
## 3         412
## 4         252
## 5         439
## 6         464
## 7         327
## 8         305
## 9         327
## 10        498
```

the output would contain variables with names like `RT/alcohol` instead of `RT_alcohol` .

7.7.4 Reshaping with multiple within-subject factors

As I mentioned above, the `wideToLong()` and `longToWide()` functions are quite limited in terms of what they can do. However, they do handle a broader range of situations than the one outlined above. Consider the following, fairly simple psychological experiment. I'm interested in the effects of practice on some simple decision making problem. It doesn't really matter what the problem is, other than to note that I'm interested in two distinct outcome variables. Firstly, I care about people's accuracy, measured by the proportion of decisions that people make correctly, denoted PC. Secondly, I care about people's speed, measured by the mean response time taken to make those decisions, denoted MRT. That's standard in psychological experiments: the speed-accuracy trade-off is pretty ubiquitous, so we generally need to care about both variables.

To look at the effects of practice over the long term, I test each participant on two days, `day1` and `day2` , where for the sake of argument I'll assume that `day1` and `day2` are about a week apart. To look at the effects of practice over the short term, the testing during each day is broken into two "blocks", `block1` and `block2` , which are about 20 minutes apart. This isn't the world's most complicated experiment, but it's still a fair bit more complicated than the last one. This time around we have two within-subject factors (i.e., `day` and `block`) and we have two measured variables for each condition (i.e., `PC` and `MRT`). The `choice` data frame shows what the wide form of this kind of data might look like:

```
choice
```

```
##   id gender MRT/block1/day1 MRT/block1/day2 MRT/block2/day1
## 1  1   male             415             400             455
## 2  2   male             500             490             532
## 3  3 female             478             468             499
## 4  4 female             550             502             602
##   MRT/block2/day2 PC/block1/day1 PC/block1/day2 PC/block2/day1
## 1             450             79             88             82
## 2             518             83             92             86
## 3             474             91             98             90
## 4             588             75             89             78
##   PC/block2/day2
## 1             93
## 2             97
## 3            100
## 4             95
```

Notice that this time around we have variable names of the form `MRT/block1/day2`. As before, the first part of the name refers to the measured variable (response time), but there are now two suffixes, one indicating that the testing took place in block 1, and the other indicating that it took place on day 2. And just to complicate matters, it uses `/` as the separator character rather than `_`. Even so, reshaping this data set is pretty easy. The command to do it is,

```
choice.2 <- wideToLong( choice, within=c("block","day"), sep="/" )
```

which is pretty much the exact same command we used last time. The only difference here is that, because there are two within-subject factors, the `within` argument is a vector that contains two names. When we look at the long form data frame that this creates, we get this:

```
choice.2
```

```
##    id gender MRT  PC  block  day
## 1   1   male 415  79 block1 day1
## 2   2   male 500  83 block1 day1
## 3   3 female 478  91 block1 day1
## 4   4 female 550  75 block1 day1
## 5   1   male 400  88 block1 day2
## 6   2   male 490  92 block1 day2
## 7   3 female 468  98 block1 day2
## 8   4 female 502  89 block1 day2
## 9   1   male 455  82 block2 day1
## 10  2   male 532  86 block2 day1
## 11  3 female 499  90 block2 day1
## 12  4 female 602  78 block2 day1
## 13  1   male 450  93 block2 day2
## 14  2   male 518  97 block2 day2
## 15  3 female 474 100 block2 day2
## 16  4 female 588  95 block2 day2
```

In this long form data frame we have two between-subject variables (`id` and `gender`), two variables that define our within-subject manipulations (`block` and `day`), and two more contain the measurements we took (`MRT` and `PC`).

To convert this back to wide form is equally straightforward. We use the `longToWide()` function, but this time around we need to alter the formula in order to tell it that we have two within-subject factors. The command is now

```
longToWide( choice.2, MRT+PC ~ block+day, sep="/" )
```

```
##   id gender MRT/block1/day1 PC/block1/day1 MRT/block1/day2 PC/block1/day2
## 1  1   male             415             79             400             88
## 2  2   male             500             83             490             92
## 3  3 female             478             91             468             98
## 4  4 female             550             75             502             89
##   MRT/block2/day1 PC/block2/day1 MRT/block2/day2 PC/block2/day2
## 1             455             82             450             93
## 2             532             86             518             97
## 3             499             90             474            100
## 4             602             78             588             95
```

and this produces a wide form data set containing the same variables as the original `choice` data frame.

7.7.5 What other options are there?

The advantage to the approach described in the previous section is that it solves a quite specific problem (but a commonly encountered one) with a minimum of fuss. The disadvantage is that the tools are quite limited in scope. They allow you to switch your data back and forth between two different formats that are very common in everyday data analysis. However, there a number of other tools that you can use if need be. Just within the core packages distributed with R there is the `reshape()` function, as well as the `stack()` and `unstack()` functions, all of which can be useful under certain circumstances. And there are of course thousands of packages on CRAN that you can use to help you with different tasks. One popular package for this purpose is the `reshape` package, written by Hadley Wickham (**???** for details see Wickham2007). There are two key functions in this package, called `melt()` and `cast()` that are pretty useful for solving a lot of reshaping problems. In a future version of this book I intend to discuss `melt()` and `cast()` in a fair amount of detail.
