Reshaping Data

This tutorial introduces the tidyr package for reshaping data in R. tidyr is written by Hadley Wickham and can be installed from CRAN using the install.packages() function. The GitHub page for the project is found here: https://github.com/hadley/tidyr.

This tutorial will cover the following topics about the tidyr package:

* [Reshaping with gather()](#gather)
* [Reshaping with spread()](#spread)
* [How to recognize long and wide formats](#recognize)

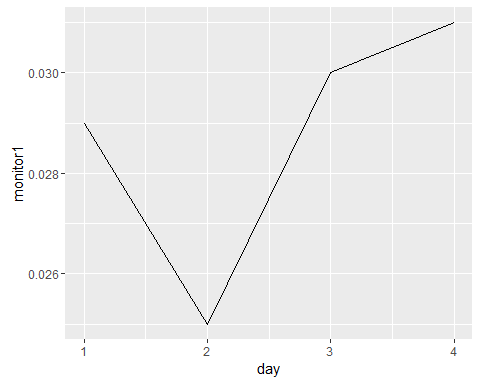
## Reshaping with gather()

Often we have data in a format that is not convenient for a certain purpose. Below is an example of a data frame that is in a wide format. This data frame contains made up data for 8 hour maximum values in parts per million from 4 different monitors over 4 consecutive days.

ozone\_wide <- read.table(header=T, text='  
 day monitor1 monitor2 monitor3 monitor4   
 1 0.029 0.040 0.010 0.032  
 2 0.025 0.050 0.015 0.041  
 3 0.030 0.045 0.012 0.040  
 4 0.031 0.052 0.019 0.041  
 ')

Now suppose we wanted to plot this data frame using the ggplot2 package. In the wide format, we can only plot one column at a time.

library(ggplot2)  
  
ggplot(ozone\_wide, aes(day, monitor1)) + geom\_line()



We could easily plot all of the values from each monitor in a ggplot2 graph if all of the values were stacked in one column. Again, right now the data frame is *wide*. We want the values to be all in one column, a *long* format. We want to gather the monitor columns in, so we use the tidyr function called gather().

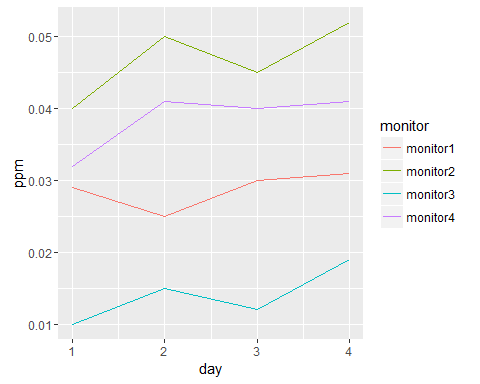
library(tidyr)  
ozone\_long <- gather(ozone\_wide, key = "monitor", value = "ppm", monitor1:monitor4)  
ozone\_long

## day monitor ppm  
## 1 1 monitor1 0.029  
## 2 2 monitor1 0.025  
## 3 3 monitor1 0.030  
## 4 4 monitor1 0.031  
## 5 1 monitor2 0.040  
## 6 2 monitor2 0.050  
## 7 3 monitor2 0.045  
## 8 4 monitor2 0.052  
## 9 1 monitor3 0.010  
## 10 2 monitor3 0.015  
## 11 3 monitor3 0.012  
## 12 4 monitor3 0.019  
## 13 1 monitor4 0.032  
## 14 2 monitor4 0.041  
## 15 3 monitor4 0.040  
## 16 4 monitor4 0.041

The first parameter in gather() is the data frame (just like every function in dplyr). The second parameter is the name we want to give to the column that will contain all of the monitor column names in the ozone\_wide data frame. The second parameter is the name we want to give to the column that will have all of the numeric values. The last parameter specifies what columns we want to gather. The colon means that we want to gather the "monitor1" column, the "monitor4" column, and all of the columns in between.

Now we can easily plot all of the values.

ggplot(ozone\_long, aes(day, ppm, color = monitor)) + geom\_line()



## Reshaping with spread()

Of course, if you have a data frame in a long format you may actually want it in a wide format. In this case, you have all of your values stacked in one long column and you want to spread those values out. We use the spread() function from tidyr to do this. In this example we simply take the long data frame we just created and reshape it back into a wide format.

spread(ozone\_long, key = "monitor", value = "ppm")

## day monitor1 monitor2 monitor3 monitor4  
## 1 1 0.029 0.040 0.010 0.032  
## 2 2 0.025 0.050 0.015 0.041  
## 3 3 0.030 0.045 0.012 0.040  
## 4 4 0.031 0.052 0.019 0.041

The parameters are the same as gather(), except we don't have to supply the columns we want to create--that information is contained within the key.

## How to recognize long and wide formats

It may be difficult to figure out what shape your data is in. The best way to figure out the shape of a data frame is to determine if there is a natural key or not. A natural key is a column in your data frame that uniquely represents each row in the data frame and does not repeat. If your data frame has a column that is a natural key, then your data is in a wide format. If your data frame doesn't have a natural key, it's in a long format.

Let's look at ozone\_wide.

ozone\_wide

## day monitor1 monitor2 monitor3 monitor4  
## 1 1 0.029 0.040 0.010 0.032  
## 2 2 0.025 0.050 0.015 0.041  
## 3 3 0.030 0.045 0.012 0.040  
## 4 4 0.031 0.052 0.019 0.041

"day" is a column that uniquely identifies each row of the data frame and doesn't repeat. There is a single day 1, a single day 2, etc.

ozone\_long doesn't have a column that is a natural key.

ozone\_long

## day monitor ppm  
## 1 1 monitor1 0.029  
## 2 2 monitor1 0.025  
## 3 3 monitor1 0.030  
## 4 4 monitor1 0.031  
## 5 1 monitor2 0.040  
## 6 2 monitor2 0.050  
## 7 3 monitor2 0.045  
## 8 4 monitor2 0.052  
## 9 1 monitor3 0.010  
## 10 2 monitor3 0.015  
## 11 3 monitor3 0.012  
## 12 4 monitor3 0.019  
## 13 1 monitor4 0.032  
## 14 2 monitor4 0.041  
## 15 3 monitor4 0.040  
## 16 4 monitor4 0.041

Each day is repeated several times.

Wide formats are easy to identify and easy to reshape to a long format. Long formats are more difficult to identify and more difficult to reshape. You must recognize that there is no key, and you must figure out which column *should* be the key.

## Exercises

Exercises for this tutorial can be found here: http://rpubs.com/NateByers/ReshapingExercises.