Data Manipulation and Organization through the Package Tidyr

Lyne Cha

October 28, 2017

Intoduction

Major Theme: Data Manipulation

Specific Topic: Tidyr

So far in class, we have learned a couple of different methods and packages to manipulate and extract data. For exmaple. we learned about dplyr, ggpplot2, and readr to name a few.These all are great and useful tools to manipulate data and each has their own specfic purpose. As I was researching about data manipulation, I came across the package Tidyr which for a lack of better words, makes your data look tidy. I felt like this package would be the perfect thing to explore for this post since along with the package dplyr, it can help sort data in a very intuitive and simple manner.

Purpose of Tidyr

More often than not, raw data will not be organized in a friendly manner to manipulate through R. The point of using Tidyr is to to "tidy" up the data or make it easier to work with through R. There are 3 general rules to consider to see if a data is "tidy." They are:

1. Each variable in the data set is placed in its own column
2. Each observation is placed in its own row
3. Each value is placed in its own cell

Tidy data works well with R because R is a vectorized programming language. Data structures in R are built from vectors and R's operations are optimized to work with vectors. Tidy data takes advantage of both of these traits. (Garrett Grolemund)

Getting Started with Tidyr

Before we can get started we need to obviously download the package Tidyr and load it to be able to use it. Let's download and load the package Tiydr now.

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.4.2
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr
```

```
## Warning: package 'tidyr' was built under R version 3.4.2
```

```
## Warning: package 'readr' was built under R version 3.4.2
```

```
## Warning: package 'purrr' was built under R version 3.4.2
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## filter(): dplyr, stats
## lag():    dplyr, stats
```

```
library(readr)
library(dplyr)
```

Basic Functions of Tidyr

Now I will introduce the 3 basic functions of Tidyr which are gather(), spread(), and separate().

gather()

gather() takes multiple columns, and gathers them into key-value pairs: it makes "wide" data longer.

A key-Value pair contains two parts: The key that explains what the information describes and a value that represents the actual information. For example, for the code,

password <- 123456

Password is the key and 123456 is the value.

```r
# create dummy dataset

names <- c('Westbrook','Kyrie','Lebron','Thomas','Simmons','Cousins','Durant')
weight <- c(200,193,250,185,195,270,240)
age <- c(28,24,32,27,27,26,28)

#create data frame

dat1<- data.frame(names, weight, age)
dat1
```

```
##       names weight age
## 1 Westbrook    200  28
## 2     Kyrie    193  24
## 3    Lebron    250  32
## 4    Thomas    185  27
## 5   Simmons    195  27
## 6   Cousins    270  26
## 7    Durant    240  28
```

```r
# use the gather function

dat2 <- dat1 %>% gather(Key, Value, weight:age)
dat2
```

```
##        names    Key Value
## 1  Westbrook weight   200
## 2      Kyrie weight   193
## 3     Lebron weight   250
## 4     Thomas weight   185
## 5    Simmons weight   195
## 6    Cousins weight   270
## 7     Durant weight   240
## 8  Westbrook    age    28
## 9      Kyrie    age    24
## 10    Lebron    age    32
## 11    Thomas    age    27
## 12   Simmons    age    27
## 13   Cousins    age    26
## 14    Durant    age    28
```

As you can see, the data table has become “longer” since there are more rows. The table now lists based on the key-value pairs and lists each entry twice once based on key and another based on value.

Spread()

spread() does reverse of gather. It takes a key:value pair and converts it into separate columns.

```r
# use the spread function

dat3 <- dat2 %>% spread(Key,Value)
dat3
```

```
##       names age weight
## 1   Cousins  26    270
## 2    Durant  28    240
## 3     Kyrie  24    193
## 4    Lebron  32    250
## 5   Simmons  27    195
## 6    Thomas  27    185
## 7 Westbrook  28    200
```

As you can see from the table, it reverses the function of gather

Separate

separate() turns a single character column into multiple columns by splitting the values of the column wherever a separator character appears.

To use separate() pass separate the name of a data frame to reshape and the name of a column to separate. Also give separate() an into argument, which should be a vector of character strings to use as new column names. separate() will return a copy of the data frame with the column removed. The previous values of the column will be split across several columns, one for each name in into.

If you wish to use a specific character to separate a column, you can pass the character to the sep argument of separate().

Separate is commonly used to separte columns that store variables of dates so that month, day, and year are separate.

For this example we will create another dummy dataset where splitting columns makes more sense

```r
# create dummy dataset

temperature <- c(60,70,75,68,78,82,62)
date <- c("2017/06/22", "2017/06/23","2017/06/24","2017/06/25","2017/06/26","2017/06/27","2017/06/28")

# create dataframe

dat4 <- data.frame(temperature, date)

dat4
```

```
##   temperature       date
## 1          60 2017/06/22
## 2          70 2017/06/23
## 3          75 2017/06/24
## 4          68 2017/06/25
## 5          78 2017/06/26
## 6          82 2017/06/27
## 7          62 2017/06/28
```

As you can see in the date column, the year, month, and day are together in one column.

By using the separate function, we are able to easily separate the year, month and day into 3 different columns.

```r
dat5 <- dat4 %>% separate(date, into= c("year","month","day"), sep = "/")
dat5
```

```
##   temperature year month day
## 1          60 2017    06  22
## 2          70 2017    06  23
## 3          75 2017    06  24
## 4          68 2017    06  25
## 5          78 2017    06  26
## 6          82 2017    06  27
## 7          62 2017    06  28
```

As you can see, this data table is a lot easier to read and easier to manipulate in R than the previous data table. By dividing the date into 3 separate columns we are able to manipulate each variable separately.

Key Takeways

Data manipulation is an integral part of data science and data manipulation starts with data cleaning. Tidyr package is a powerful tool that can help clean up raw data to make it more presentable and easier to manipulate in R. I think it would be useful to learn how to use Tidyr and especially the 3 basic functions I have highlighted in my post.

References:

1.http://garrettgman.github.io/tidying/

2.https://www.analyticsvidhya.com/blog/2015/12/faster-data-manipulation-7-packages/

3.https://blog.rstudio.com/2014/07/22/introducing-tidyr/

4.https://www.rdocumentation.org/packages/tidyr/versions/0.7.2/topics/separate

5.https://cran.r-project.org/web/packages/tidyr/README.html

6.https://rpubs.com/bradleyboehmke/data_wrangling

7.https://datascienceplus.com/data-manipulation-with-tidyr/

8.http://tidyr.tidyverse.org/