post01-zhiheng-xu

ZHIHENG XU

2017/10/31/

An useful package in data wrangling - “tidyr”

Introduction

In R, there are some basic functions that can help us clean the data. Moreover, from the lecture, we learn how to tidy and reshaping data by using paclage “dplyr”. However, there are some other ways that can help us tidy the data. In this post, I’m going to introdcue package “tidyr” where its goal is to help us creat tidy data.

Important properties

There are three important properties about tidy data (http://tidyr.tidyverse.org/):

- 1.Each variable is in a column
- 2.Each observation is a row
- 3.Each value is a cell

Install the package

```
install.packages("tidyr")
```

Getting Started

```
library(tidyr)
```

Major Operations

There are four fundamental operations in tidyr (https://rpubs.com/bradleyboehmke/data_wrangling):

- gather():takes multiple columns, and gathers them into key value pairs: it makes “wide” data longer
- spread(): takes two columns(key&value) and spreads in to multiple columns, it makes “long” data wider
- sepearte(): spilts a single column into multiple columns
- unite():combine multiple columns into a single column

In this post, I will show you how to use these four major tidyr operations. ####gather() Suppose we have a messy data frame such as:

```
messy <- data.frame(
  year = c(2014,2015,2016,2017),
  quarter1 = c(101,145,167,176),
  quarter2 = c(156,189,197,199),
  quarter3 =c(134,123,147,189),
  quarter4 = c(142,169,193,156))
messy
```

```
##   year quarter1 quarter2 quarter3 quarter4
## 1 2014      101      156      134      142
## 2 2015      145      189      123      169
## 3 2016      167      197      147      193
## 4 2017      176      199      189      156
```

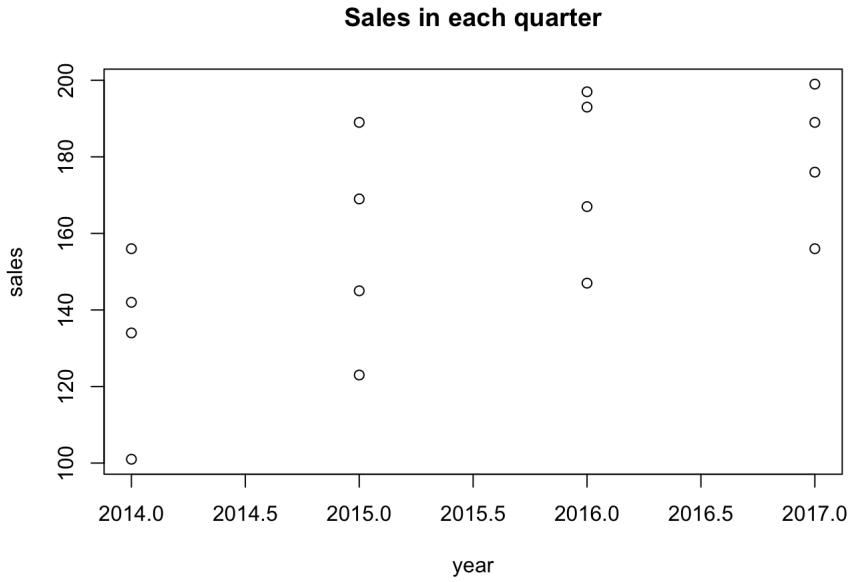
Quarter1,quarter2, quarter3, and quarter 4 indicats the sales in each quarter in a year. However,this data looks wide since each quarter represents a variable. To combine them into one variable, we can gather each quarter into one column variable and gather its sales in another column variable.

```
clean <- gather(messy, quarter, sales,quarter1:quarter4)
clean
```

```
##    year  quarter sales
## 1  2014 quarter1   101
## 2  2015 quarter1   145
## 3  2016 quarter1   167
## 4  2017 quarter1   176
## 5  2014 quarter2   156
## 6  2015 quarter2   189
## 7  2016 quarter2   197
## 8  2017 quarter2   199
## 9  2014 quarter3   134
## 10 2015 quarter3   123
## 11 2016 quarter3   147
## 12 2017 quarter3   189
## 13 2014 quarter4   142
## 14 2015 quarter4   169
## 15 2016 quarter4   193
## 16 2017 quarter4   156
```

And we can plot a nice plot that shows the sales in each year based on our new dataset.

```
plot(clean$year,clean$sales,xlab = "year", ylab = "sales",main = "Sales in each quarter")
```



spread()

Suppose now we already have our cleaned data from previous question, but now a financial analyst want to looking at the sales for each quarter in each year to prepare his financial report. Therefore, we can use spread() now to restructure our dataset.

```
restructure <- spread(clean, key = quarter, value = sales)
restructure
```

```
##   year quarter1 quarter2 quarter3 quarter4
## 1 2014      101      156      134      142
## 2 2015      145      189      123      169
## 3 2016      167      197      147      193
## 4 2017      176      199      189      156
```

seperate()

Suppose now we have a dataset that includes the total sales for each quarter in each year. But now we want to see the quantity and single price in each quarter.

```
total <- data.frame( year = c(2014,2015,2016,2017,2014,2015,2016,2017,2014,2015,2016,2017,2014,2015,2016,2017),
                     quarter = c("quarter1","quarter1","quarter1","quarter1",
                                 "quarter2","quarter2","quarter2","quarter2",
                                 "quarter3","quarter3","quarter3","quarter3",
                                 "quarter4","quarter4","quarter4","quarter4"),
                     total_sales = c("11*15","12*14","12*16","14*19",
                                     "15*17","11*15","14*15","17*19",
                                     "18*17","17*15","14*15","19*15",
                                     "11*15","18*15","17*17","16*15"
                                   ))
total
```

```
##    year  quarter total_sales
## 1  2014 quarter1       11*15
## 2  2015 quarter1       12*14
## 3  2016 quarter1       12*16
## 4  2017 quarter1       14*19
## 5  2014 quarter2       15*17
## 6  2015 quarter2       11*15
## 7  2016 quarter2       14*15
## 8  2017 quarter2       17*19
## 9  2014 quarter3       18*17
## 10 2015 quarter3       17*15
## 11 2016 quarter3       14*15
## 12 2017 quarter3       19*15
## 13 2014 quarter4       11*15
## 14 2015 quarter4       18*15
## 15 2016 quarter4       17*17
## 16 2017 quarter4       16*15
```

But now we want to see the quantity and single price in each quarter.

```
each <- total %>% separate(total_sales, into = c("quantity", "single_price"))
each
```

```
##    year  quarter quantity single_price
## 1  2014 quarter1       11           15
## 2  2015 quarter1       12           14
## 3  2016 quarter1       12           16
## 4  2017 quarter1       14           19
## 5  2014 quarter2       15           17
## 6  2015 quarter2       11           15
## 7  2016 quarter2       14           15
## 8  2017 quarter2       17           19
## 9  2014 quarter3       18           17
## 10 2015 quarter3       17           15
## 11 2016 quarter3       14           15
## 12 2017 quarter3       19           15
## 13 2014 quarter4       11           15
## 14 2015 quarter4       18           15
## 15 2016 quarter4       17           17
## 16 2017 quarter4       16           15
```

unite()

unite() is actually the inverse of separate().By using unite(), we can combine multiple columns into one clolumn. For example, suppose we have a dataset that represent year as two components “century” and “year”.

```
seperate <- data.frame( century = c(rep(20,16)),
                        year =c(14,15,16,17,14,15,16,17,14,15,16,17,14,15,16,17),
                    quarter = c("quarter1","quarter1","quarter1","quarter1",
                                "quarter2","quarter2","quarter2","quarter2",
                                "quarter3","quarter3","quarter3","quarter3",
                                "quarter4","quarter4","quarter4","quarter4"),
                    total_sales = c("11*15","12*14","12*16","14*19",
                                    "15*17","11*15","14*15","17*19",
                                    "18*17","17*15","14*15","19*15",
                                    "11*15","18*15","17*17","16*15"
                                ))
seperate
```

```
##    century year  quarter total_sales
## 1       20   14 quarter1       11*15
## 2       20   15 quarter1       12*14
## 3       20   16 quarter1       12*16
## 4       20   17 quarter1       14*19
## 5       20   14 quarter2       15*17
## 6       20   15 quarter2       11*15
## 7       20   16 quarter2       14*15
## 8       20   17 quarter2       17*19
## 9       20   14 quarter3       18*17
## 10      20   15 quarter3       17*15
## 11      20   16 quarter3       14*15
## 12      20   17 quarter3       19*15
## 13      20   14 quarter4       11*15
## 14      20   15 quarter4       18*15
## 15      20   16 quarter4       17*17
## 16      20   17 quarter4       16*15
```

Now by using unite(), we can combine century and year into a single column.

```
single <- unite(seperate, "calendar_year", c("century","year"),sep = "")
single
```

```
##    calendar_year  quarter total_sales
## 1           2014 quarter1       11*15
## 2           2015 quarter1       12*14
## 3           2016 quarter1       12*16
## 4           2017 quarter1       14*19
## 5           2014 quarter2       15*17
## 6           2015 quarter2       11*15
## 7           2016 quarter2       14*15
## 8           2017 quarter2       17*19
## 9           2014 quarter3       18*17
## 10          2015 quarter3       17*15
## 11          2016 quarter3       14*15
## 12          2017 quarter3       19*15
## 13          2014 quarter4       11*15
## 14          2015 quarter4       18*15
## 15          2016 quarter4       17*17
## 16          2017 quarter4       16*15
```

References

- Pacakge ‘tidyr’, https://cran.r-project.org/web/packages/tidyr/tidyr.pdf
- Easily Tidy data with ‘spread()’ and ‘gather()’ Fucntions http://tidyr.tidyverse.org/
- How to reshape data in R: tidyr vs reshape2 http://www.milanor.net/blog/reshape-data-r-tidyr-vs-reshape2/

- 12 Tidy data - R for data sciencehttp://r4ds.had.co.nz/tidy-data.html
- Data manipulation with tidyr and dplyr https://sesync-ci.github.io/data-manipulation-in-R-lesson/2016/07/26/
- Data Procsessing with dplyr and tidyr https://rpubs.com/bradleyboehmke/data_wrangling
- R: Tidyr Package Intro, data wrangling https://www.youtube.com/watch?v=mAOvzjqQcO0