Tidyr: Making Data Neater

Arjun Banerjee

December 3, 2017

Introduction:

One of the most importatn steps of data analytics is preparing the data for analysis when it comes to us in less than ideal forms. Packages like `"dplyr"` or `"reshape2"` are very useful in this pursuit. But the great developer Hadley Wickham has developed an even more advanced version of these two packages with the package `"tidtyr"`.

This package has four primary function which can help us make some data more compact when we need it to, and other data more spread out when we need it to. The four functions are `separate()`, `unite()`, `spread()` and `gather()`. In this post we will learn how to use all of these functions.

Getting Started:

```
#For this tutorial we're also going to be using magrittr and dplyr
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 3.4.3
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(magrittr)
```

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

```
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

Now that we have loaded the packages into our session we are going to start looking at the functions. For this post we will be using many of the data sets that already come in R studio

Separate:

The function `separate()` is used when the data strings in a column needs to be separated into mulitple columns. If one of the columns have has a series of strings that you want split up separate can do it.

For this example we're going to use mtcars, a data set that details the results of a series of tests on cars

```
mtcars
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

As you can see each row is labeled with the cars maker and model, but we want to look at stats by maker. TO do this we will use separate. But first we need to get it into workable condition

```
messy = mtcars
messy$Make_and_Model = row.names(mtcars)
#The car labeled Valiant doesn't list a maker and both Plymouth and Chrysler made on so we'll filter it out
messy = messy[messy$Make_and_Model != "Valiant", ]
head(messy, 10)
```

```
##                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C         17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
##                      Make_and_Model
## Mazda RX4                 Mazda RX4
## Mazda RX4 Wag         Mazda RX4 Wag
## Datsun 710               Datsun 710
## Hornet 4 Drive       Hornet 4 Drive
## Hornet Sportabout Hornet Sportabout
## Duster 360               Duster 360
## Merc 240D                 Merc 240D
## Merc 230                   Merc 230
## Merc 280                   Merc 280
## Merc 280C                 Merc 280C
```

Now that we have a column with the Maker and the Model we will separate the columns.The default function has 4 inputs. The first is the data frame you're manipulating. The second is the column with the info that you want to separate. The thrid is a vector with the names of the new columns. The fourth is what the function should use to determine when to separate the data string in that column. We'll do it here

```
#messy will be the data frame we use
#Make_and_Model will be the column we select
new_column_names = c("Maker", "Model")
#The separater will be a space indicating the column should be split at the space

#If there are too many values, like if there is more than one space in the Make and Model column separate will tak
e the first value and inform you at what rows the error occured. This doesn't bother us because we're only doing t
his to get the first word which is the maker
clean_cars = separate(messy, col = Make_and_Model, new_column_names, sep = " ")
```

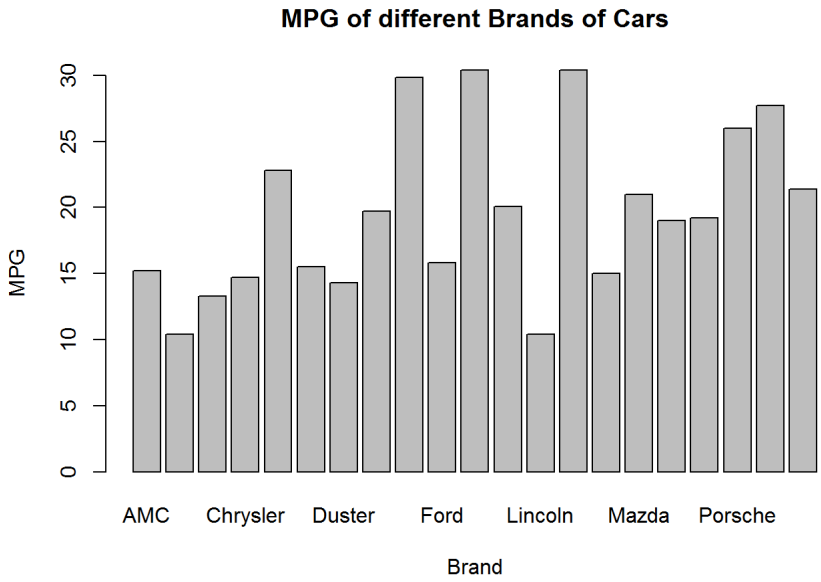
```
## Warning: Too many values at 3 locations: 2, 4, 28
```

```
clean_cars$Maker = factor(clean_cars$Maker)
head(clean_cars, 10)
```

```
##                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C         17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
##                    Maker      Model
## Mazda RX4          Mazda        RX4
## Mazda RX4 Wag      Mazda        RX4
## Datsun 710        Datsun        710
## Hornet 4 Drive    Hornet          4
## Hornet Sportabout Hornet Sportabout
## Duster 360        Duster        360
## Merc 240D           Merc       240D
## Merc 230            Merc        230
## Merc 280            Merc        280
## Merc 280C           Merc       280C
```

Now we have a Column that specifies the Maker of the car and can do whatever we want with that information. We're going to use it to make a chart looking at the average mpg of each Maker's cars.

```
barplt = clean_cars %>% group_by(Maker) %>% summarise(mean_mpg = mean(mpg))
barplot(barplt$mean_mpg, main = "MPG of different Brands of Cars", xlab = "Brand", ylab = "MPG" , names.arg = as.c
haracter(barplt$Maker))
```




```
#The image might be a bit to small but it's not the focus of what we're doing
```

Unite:

The function `unite()` does the opposite of `separate()`. It takes two columns of strings and combines them into one column of the two strings combined.

There isn't really a built in R data frame that could be used for this so we're gonna create a small one ourselves.

```
First_name = c("Alex", "Tom", "Jimmy")
Last_name = c("Smith", "Brady", "Garropolo")
QBR = c(85, 10, 158.3)
unite_example = data.frame(First_name, Last_name, QBR)
unite_example
```

```
##   First_name Last_name   QBR
## 1       Alex     Smith  85.0
## 2        Tom     Brady  10.0
## 3      Jimmy Garropolo 158.3
```

Now that we have the data frame we're going to combine the first name and last name column into one name column. We will use unite to do

this. Unite has many inputs, like separate. The first is the data frame you're using. The second is the name of the NEW Column you're creating (unlike in separate). The third is one of the columns you want to merge. The fourth is the other one. The theoretical fifth one is the third column you want to merge and so on and so forth. The final one is the thing that separates the two newly merged strings. Here's the example

```
#unite_example is the data frame we are going to use
#The name of the New column will be Name. It is not in quotation marks, it's just an object
#We'll put a space in between the names

cleaned_unite_example = unite(unite_example, Name, First_name, Last_name, sep = " ")
cleaned_unite_example
```

```
##              Name   QBR
## 1      Alex Smith  85.0
## 2       Tom Brady  10.0
## 3 Jimmy Garropolo 158.3
```

Now we'll move onto the other two functions which are also complements of each other. `spread()` and `gather()`

Spread:

Spread is used to take the values of one column that denote a category (The values need to be strings), and turn them into labels for columns. We'll se an example

We'll have to create an example data frame.

```
QB = c(rep("Alex Smith", 4), rep("Tom Brady", 4), rep("Jimmy Garropolo", 4))
Week = rep(c("Wk1", "Wk2", "Wk3", "Wk4"), 3)
Rating = c(102, 79, 56, 83, 12, 16, 0, 42, 158.3, 158.3, 158.3, 145.6)
spread_example = data.frame(QB, Week, Rating)
spread_example
```

```
##                QB Week Rating
## 1       Alex Smith  Wk1  102.0
## 2       Alex Smith  Wk2   79.0
## 3       Alex Smith  Wk3   56.0
## 4       Alex Smith  Wk4   83.0
## 5        Tom Brady  Wk1   12.0
## 6        Tom Brady  Wk2   16.0
## 7        Tom Brady  Wk3    0.0
## 8        Tom Brady  Wk4   42.0
## 9  Jimmy Garropolo  Wk1  158.3
## 10 Jimmy Garropolo  Wk2  158.3
## 11 Jimmy Garropolo  Wk3  158.3
## 12 Jimmy Garropolo  Wk4  145.6
```

So if instead of looking at QBs in this ungangly way we can use spread to make the data frame 4 rows, each week with the same value of passer rating in this frame. Spread has 3 mandatory inputs, the first is the data frame you are using, The second is the column from which the new column labels will be taken from. And the thrid is the values which will be used

```
#The Second input is the "key". It will be the guide as to which column that rows Rating value will go into
cleaned_spread_example = spread(spread_example, Week, Rating)
cleaned_spread_example
```

```
##               QB   Wk1   Wk2   Wk3   Wk4
## 1      Alex Smith 102.0  79.0  56.0  83.0
## 2 Jimmy Garropolo 158.3 158.3 158.3 145.6
## 3       Tom Brady  12.0  16.0   0.0  42.0
```

This data looks a lot cleaner and we could use it to see if there is any pattern from week to week among all QBs or their individual patterns over time.

Gather:

Now we will be getting to the final function. Gather does the opposite of the spread. It takes the specified column labels of a data frame and turns them into string values of a new column. To see why this is useful, and to mix in the other functions we will be using the `presidents` data from R Studio. It is a time series of the presidents quarterly approval ratings so some processing will be needed.

```
Year = seq(1945, 1974)
Qtr1 = presidents[seq(1, 117, 4)]
Qtr2 = presidents[seq(2, 118, 4)]
Qtr3 = presidents[seq(3, 119, 4)]
Qtr4 = presidents[seq(4, 120, 4)]
approval = data.frame(Year, Qtr1, Qtr2, Qtr3, Qtr4)
head(approval, 15)
```

```
##    Year Qtr1 Qtr2 Qtr3 Qtr4
## 1  1945   NA   87   82   75
## 2  1946   63   50   43   32
## 3  1947   35   60   54   55
## 4  1948   36   39   NA   NA
## 5  1949   69   57   57   51
## 6  1950   45   37   46   39
## 7  1951   36   24   32   23
## 8  1952   25   32   NA   32
## 9  1953   59   74   75   60
## 10 1954   71   61   71   57
## 11 1955   71   68   79   73
## 12 1956   76   71   67   75
## 13 1957   79   62   63   57
## 14 1958   60   49   48   52
## 15 1959   57   62   61   66
```

Now tha we have the data we will “gather” it into a three column data frame where every year is repeated 4 times with a different Quarter value for the new column Quarter.

Gather has a couple main inputs. The first is the data frame we will be using. The second is the name of the new category column (The column that will have the previous data frame column label). The third is the name of the new value column hich has the old column values. The final ones are the remaining columns that you wish to gather into one new column.

```
#approval is the data frame
#Quarter will be the new name of the category column
#Rating will be the name of the new approval rating column
#The columns selected will be Qtr, Qtr2, Qtr3, Qtr4
clean_approval = gather(approval, Quarter, Rating, Qtr1, Qtr2, Qtr3, Qtr4)
head(clean_approval, 10)
```

```
##    Year Quarter Rating
## 1  1945    Qtr1     NA
## 2  1946    Qtr1     63
## 3  1947    Qtr1     35
## 4  1948    Qtr1     36
## 5  1949    Qtr1     69
## 6  1950    Qtr1     45
## 7  1951    Qtr1     36
## 8  1952    Qtr1     25
## 9  1953    Qtr1     59
## 10 1954    Qtr1     71
```

Now this might seem to have made the data less legible but now we can convert the Quarters into number values using separate and then we can add them to the year and then plot the approval rating over time. Like this

```
separated_approval = separate(clean_approval, col = Quarter, c("Useless", "Quarter_Value"), sep = "r")
head(separated_approval, 10)
```

```
##    Year Useless Quarter_Value Rating
## 1  1945      Qt             1     NA
## 2  1946      Qt             1     63
## 3  1947      Qt             1     35
## 4  1948      Qt             1     36
## 5  1949      Qt             1     69
## 6  1950      Qt             1     45
## 7  1951      Qt             1     36
## 8  1952      Qt             1     25
## 9  1953      Qt             1     59
## 10 1954      Qt             1     71
```

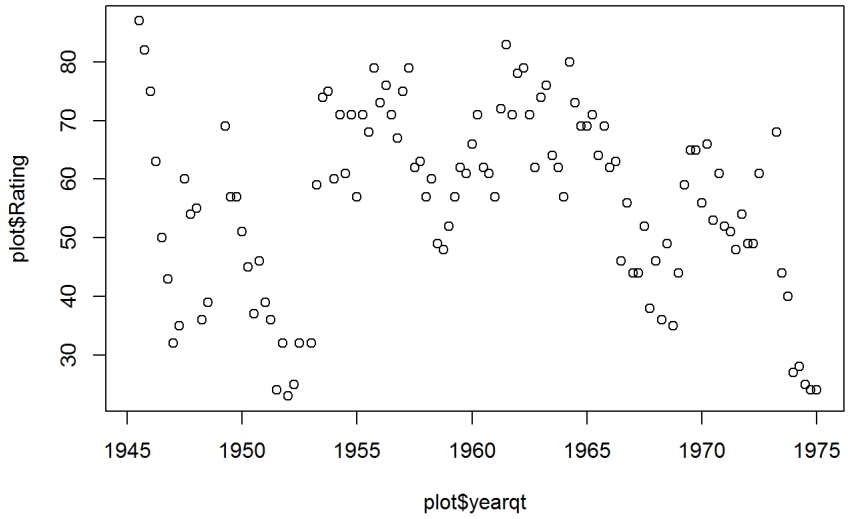
Now we’ll make Quarter_value numbers out of 4, so that each quarter is a fraction of a year, and then add them to the year thus creating a linear timeline that we can plot.

```
plot = separated_approval
plot$Useless = NULL
plot$Quarter_Value = as.numeric(plot$Quarter_Value)/4
plot = mutate(plot, yearqt = Quarter_Value + Year)
head(plot)
```

```
##   Year Quarter_Value Rating  yearqt
## 1 1945          0.25     NA 1945.25
## 2 1946          0.25     63 1946.25
## 3 1947          0.25     35 1947.25
## 4 1948          0.25     36 1948.25
## 5 1949          0.25     69 1949.25
## 6 1950          0.25     45 1950.25
```

Now we can use this new data frame to make a plot

```
plot(plot$yearqt, plot$Rating)
```



Takeaways:

Tidyr makes data neater. It allows us to get information that otherwise would be very difficult to parse out manually.

References:

https://www.youtube.com/watch?v=RbUWwuJeUC8 http://data.library.virginia.edu/a-tidyr-tutorial/ https://www.r-bloggers.com/introducing-tidyr/ https://awesome-r.com/ https://github.com/tidyverse/tidyr/tree/master/R https://rpubs.com/bradleyboehmke/data_wrangling http://www.jvcasillas.com/tidyr_tutorial/