Data manipulation with dplyr

Grace He

December 1, 2017

Introduction

dplyr is a powerful R-package to transform and manipulate tabular data. It was written by Hadley Wickham, who is also the author of some other extremely useful R packages such as ggplot2. The package "dplyr" contains many functions that perform data manipulation operations such as selecting specific columns, sorting data, and aggregating data. tidyr, another package written by the same author, is also a great tool for data manipulation. The first motivation of this post is to give a summary of key dplyr and tidyr functions and show the reader some quick demonstrations. The post is also motivated by the important roles played by dplyr and tidyr in the data preparation process – the functions in these packages process faster than base R functions, and their syntaxes are also easier to undertsand and more consistent. Mastering these two packages can greatly improve your effciency when manipulating data. This post will start with some basic functions that we are faimilar with, and extends to some new material that we have not learned in class or labs. We're using the built-in dataset `iris` in the dplyr section of the post; for the tidyr section, we are going to create our new data frame `grade`.

Before we start with the actual data manipulation, please install the following packages and load them into R. The version of R studio and the packages are also shown below.

```
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(tidyr)
```

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

```
sessionInfo()
```

```
## R version 3.4.1 (2017-06-30)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: OS X El Capitan 10.11.5
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] tidyr_0.7.2 dplyr_0.7.3
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.12     digest_0.6.12    rprojroot_1.2    assertthat_0.2.0
##  [5] R6_2.2.2         backports_1.1.0  magrittr_1.5     evaluate_0.10.1
##  [9] rlang_0.1.2      stringi_1.1.5    bindrcpp_0.2     rmarkdown_1.6
## [13] tools_3.4.1      stringr_1.2.0    glue_1.1.1       purrr_0.2.4
## [17] yaml_2.1.14      compiler_3.4.1   pkgconfig_2.0.1  htmltools_0.3.6
## [21] bindr_0.1        knitr_1.17       tibble_1.3.4
```

dplyr functions

`filter()` function in the dplyr package allows us to select a subset of rows with matching logical conditions. The first argument is the name of the data frame, and the second argument is the filtering expression evaluated in the context of our data frame; if you want, you can add more filtering expressions after the second argument. Here are a couple examples:

```
# we are selecting rows of species setosa only
filter(iris, Species == "setosa")
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1           5.1         3.5          1.4         0.2  setosa
## 2           4.9         3.0          1.4         0.2  setosa
## 3           4.7         3.2          1.3         0.2  setosa
## 4           4.6         3.1          1.5         0.2  setosa
## 5           5.0         3.6          1.4         0.2  setosa
## 6           5.4         3.9          1.7         0.4  setosa
## 7           4.6         3.4          1.4         0.3  setosa
## 8           5.0         3.4          1.5         0.2  setosa
## 9           4.4         2.9          1.4         0.2  setosa
## 10          4.9         3.1          1.5         0.1  setosa
## 11          5.4         3.7          1.5         0.2  setosa
## 12          4.8         3.4          1.6         0.2  setosa
## 13          4.8         3.0          1.4         0.1  setosa
## 14          4.3         3.0          1.1         0.1  setosa
## 15          5.8         4.0          1.2         0.2  setosa
## 16          5.7         4.4          1.5         0.4  setosa
## 17          5.4         3.9          1.3         0.4  setosa
## 18          5.1         3.5          1.4         0.3  setosa
## 19          5.7         3.8          1.7         0.3  setosa
## 20          5.1         3.8          1.5         0.3  setosa
## 21          5.4         3.4          1.7         0.2  setosa
## 22          5.1         3.7          1.5         0.4  setosa
## 23          4.6         3.6          1.0         0.2  setosa
## 24          5.1         3.3          1.7         0.5  setosa
## 25          4.8         3.4          1.9         0.2  setosa
## 26          5.0         3.0          1.6         0.2  setosa
## 27          5.0         3.4          1.6         0.4  setosa
## 28          5.2         3.5          1.5         0.2  setosa
## 29          5.2         3.4          1.4         0.2  setosa
## 30          4.7         3.2          1.6         0.2  setosa
## 31          4.8         3.1          1.6         0.2  setosa
## 32          5.4         3.4          1.5         0.4  setosa
## 33          5.2         4.1          1.5         0.1  setosa
## 34          5.5         4.2          1.4         0.2  setosa
## 35          4.9         3.1          1.5         0.2  setosa
## 36          5.0         3.2          1.2         0.2  setosa
## 37          5.5         3.5          1.3         0.2  setosa
## 38          4.9         3.6          1.4         0.1  setosa
## 39          4.4         3.0          1.3         0.2  setosa
## 40          5.1         3.4          1.5         0.2  setosa
## 41          5.0         3.5          1.3         0.3  setosa
## 42          4.5         2.3          1.3         0.3  setosa
## 43          4.4         3.2          1.3         0.2  setosa
## 44          5.0         3.5          1.6         0.6  setosa
## 45          5.1         3.8          1.9         0.4  setosa
## 46          4.8         3.0          1.4         0.3  setosa
## 47          5.1         3.8          1.6         0.2  setosa
## 48          4.6         3.2          1.4         0.2  setosa
## 49          5.3         3.7          1.5         0.2  setosa
## 50          5.0         3.3          1.4         0.2  setosa
```

```r
# We are selecting rows of species setosa that have sepal width greater than 3
filter(iris, Species == "setosa", Sepal.Width >= 3)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1           5.1         3.5          1.4         0.2  setosa
## 2           4.9         3.0          1.4         0.2  setosa
## 3           4.7         3.2          1.3         0.2  setosa
## 4           4.6         3.1          1.5         0.2  setosa
## 5           5.0         3.6          1.4         0.2  setosa
## 6           5.4         3.9          1.7         0.4  setosa
## 7           4.6         3.4          1.4         0.3  setosa
## 8           5.0         3.4          1.5         0.2  setosa
## 9           4.9         3.1          1.5         0.1  setosa
## 10          5.4         3.7          1.5         0.2  setosa
## 11          4.8         3.4          1.6         0.2  setosa
## 12          4.8         3.0          1.4         0.1  setosa
## 13          4.3         3.0          1.1         0.1  setosa
## 14          5.8         4.0          1.2         0.2  setosa
## 15          5.7         4.4          1.5         0.4  setosa
## 16          5.4         3.9          1.3         0.4  setosa
## 17          5.1         3.5          1.4         0.3  setosa
## 18          5.7         3.8          1.7         0.3  setosa
## 19          5.1         3.8          1.5         0.3  setosa
## 20          5.4         3.4          1.7         0.2  setosa
## 21          5.1         3.7          1.5         0.4  setosa
## 22          4.6         3.6          1.0         0.2  setosa
## 23          5.1         3.3          1.7         0.5  setosa
## 24          4.8         3.4          1.9         0.2  setosa
## 25          5.0         3.0          1.6         0.2  setosa
## 26          5.0         3.4          1.6         0.4  setosa
## 27          5.2         3.5          1.5         0.2  setosa
## 28          5.2         3.4          1.4         0.2  setosa
## 29          4.7         3.2          1.6         0.2  setosa
## 30          4.8         3.1          1.6         0.2  setosa
## 31          5.4         3.4          1.5         0.4  setosa
## 32          5.2         4.1          1.5         0.1  setosa
## 33          5.5         4.2          1.4         0.2  setosa
## 34          4.9         3.1          1.5         0.2  setosa
## 35          5.0         3.2          1.2         0.2  setosa
## 36          5.5         3.5          1.3         0.2  setosa
## 37          4.9         3.6          1.4         0.1  setosa
## 38          4.4         3.0          1.3         0.2  setosa
## 39          5.1         3.4          1.5         0.2  setosa
## 40          5.0         3.5          1.3         0.3  setosa
## 41          4.4         3.2          1.3         0.2  setosa
## 42          5.0         3.5          1.6         0.6  setosa
## 43          5.1         3.8          1.9         0.4  setosa
## 44          4.8         3.0          1.4         0.3  setosa
## 45          5.1         3.8          1.6         0.2  setosa
## 46          4.6         3.2          1.4         0.2  setosa
## 47          5.3         3.7          1.5         0.2  setosa
## 48          5.0         3.3          1.4         0.2  setosa
```

The function `slice()` also helps us to select a subset of rows, but by position instead. Same as the filter function, you will have the name of the data frame as your first argument, and you specify the desired row positions for the second argument. Here is an example:

```
# Subsetting the first five rows of the data frame
slice(iris, 1:5)
```

```
## # A tibble: 5 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl>  <fctr>
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

`arrange()` reorders rows of the data frame. It takes the name of the data frame as its first argument, and the column name to order by as the second argument. If more than one column names are provided, each column will be used to break ties in the values of previous columns. And we can use `desc()` to sort in descending order.

```
# reordering the data frame by descending petal length
arrange(iris, desc(Petal.Length))
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1            7.7         2.6          6.9         2.3  virginica
## 2            7.7         3.8          6.7         2.2  virginica
## 3            7.7         2.8          6.7         2.0  virginica
## 4            7.6         3.0          6.6         2.1  virginica
## 5            7.9         3.8          6.4         2.0  virginica
## 6            7.3         2.9          6.3         1.8  virginica
## 7            7.2         3.6          6.1         2.5  virginica
## 8            7.4         2.8          6.1         1.9  virginica
## 9            7.7         3.0          6.1         2.3  virginica
```

```
## 10          6.3         3.3          6.0         2.5  virginica
## 11          7.2         3.2          6.0         1.8  virginica
## 12          7.1         3.0          5.9         2.1  virginica
## 13          6.8         3.2          5.9         2.3  virginica
## 14          6.5         3.0          5.8         2.2  virginica
## 15          6.7         2.5          5.8         1.8  virginica
## 16          7.2         3.0          5.8         1.6  virginica
## 17          6.9         3.2          5.7         2.3  virginica
## 18          6.7         3.3          5.7         2.1  virginica
## 19          6.7         3.3          5.7         2.5  virginica
## 20          6.3         2.9          5.6         1.8  virginica
## 21          6.4         2.8          5.6         2.1  virginica
## 22          6.4         2.8          5.6         2.2  virginica
## 23          6.1         2.6          5.6         1.4  virginica
## 24          6.3         3.4          5.6         2.4  virginica
## 25          6.7         3.1          5.6         2.4  virginica
## 26          6.8         3.0          5.5         2.1  virginica
## 27          6.5         3.0          5.5         1.8  virginica
## 28          6.4         3.1          5.5         1.8  virginica
## 29          6.9         3.1          5.4         2.1  virginica
## 30          6.2         3.4          5.4         2.3  virginica
## 31          6.4         2.7          5.3         1.9  virginica
## 32          6.4         3.2          5.3         2.3  virginica
## 33          6.7         3.0          5.2         2.3  virginica
## 34          6.5         3.0          5.2         2.0  virginica
## 35          6.0         2.7          5.1         1.6 versicolor
## 36          5.8         2.7          5.1         1.9  virginica
## 37          6.5         3.2          5.1         2.0  virginica
## 38          5.8         2.8          5.1         2.4  virginica
## 39          6.3         2.8          5.1         1.5  virginica
## 40          6.9         3.1          5.1         2.3  virginica
## 41          5.8         2.7          5.1         1.9  virginica
## 42          5.9         3.0          5.1         1.8  virginica
## 43          6.7         3.0          5.0         1.7 versicolor
## 44          5.7         2.5          5.0         2.0  virginica
## 45          6.0         2.2          5.0         1.5  virginica
## 46          6.3         2.5          5.0         1.9  virginica
## 47          6.9         3.1          4.9         1.5 versicolor
## 48          6.3         2.5          4.9         1.5 versicolor
## 49          5.6         2.8          4.9         2.0  virginica
## 50          6.3         2.7          4.9         1.8  virginica
## 51          6.1         3.0          4.9         1.8  virginica
## 52          5.9         3.2          4.8         1.8 versicolor
## 53          6.8         2.8          4.8         1.4 versicolor
## 54          6.2         2.8          4.8         1.8  virginica
## 55          6.0         3.0          4.8         1.8  virginica
## 56          7.0         3.2          4.7         1.4 versicolor
## 57          6.3         3.3          4.7         1.6 versicolor
## 58          6.1         2.9          4.7         1.4 versicolor
## 59          6.1         2.8          4.7         1.2 versicolor
## 60          6.7         3.1          4.7         1.5 versicolor
## 61          6.5         2.8          4.6         1.5 versicolor
## 62          6.6         2.9          4.6         1.3 versicolor
## 63          6.1         3.0          4.6         1.4 versicolor
## 64          6.4         3.2          4.5         1.5 versicolor
## 65          5.7         2.8          4.5         1.3 versicolor
## 66          5.6         3.0          4.5         1.5 versicolor
## 67          6.2         2.2          4.5         1.5 versicolor
## 68          6.0         2.9          4.5         1.5 versicolor
## 69          5.4         3.0          4.5         1.5 versicolor
## 70          6.0         3.4          4.5         1.6 versicolor
## 71          4.9         2.5          4.5         1.7  virginica
## 72          6.7         3.1          4.4         1.4 versicolor
## 73          6.6         3.0          4.4         1.4 versicolor
## 74          6.3         2.3          4.4         1.3 versicolor
## 75          5.5         2.6          4.4         1.2 versicolor
## 76          6.4         2.9          4.3         1.3 versicolor
## 77          6.2         2.9          4.3         1.3 versicolor
## 78          5.9         3.0          4.2         1.5 versicolor
## 79          5.6         2.7          4.2         1.3 versicolor
## 80          5.7         3.0          4.2         1.2 versicolor
## 81          5.7         2.9          4.2         1.3 versicolor
## 82          5.8         2.7          4.1         1.0 versicolor
## 83          5.6         3.0          4.1         1.3 versicolor
## 84          5.7         2.8          4.1         1.3 versicolor
## 85          5.5         2.3          4.0         1.3 versicolor
## 86          6.0         2.2          4.0         1.0 versicolor
## 87          6.1         2.8          4.0         1.3 versicolor
## 88          5.5         2.5          4.0         1.3 versicolor
## 89          5.8         2.6          4.0         1.2 versicolor
## 90          5.2         2.7          3.9         1.4 versicolor
## 91          5.6         2.5          3.9         1.1 versicolor
## 92          5.8         2.7          3.9         1.2 versicolor
## 93          5.5         2.4          3.8         1.1 versicolor
## 94          5.5         2.4          3.7         1.0 versicolor
## 95          5.6         2.9          3.6         1.3 versicolor
```

```
## 96           5.0         2.0          3.5         1.0 versicolor
## 97           5.7         2.6          3.5         1.0 versicolor
## 98           4.9         2.4          3.3         1.0 versicolor
## 99           5.0         2.3          3.3         1.0 versicolor
## 100          5.1         2.5          3.0         1.1 versicolor
## 101          4.8         3.4          1.9         0.2     setosa
## 102          5.1         3.8          1.9         0.4     setosa
## 103          5.4         3.9          1.7         0.4     setosa
## 104          5.7         3.8          1.7         0.3     setosa
## 105          5.4         3.4          1.7         0.2     setosa
## 106          5.1         3.3          1.7         0.5     setosa
## 107          4.8         3.4          1.6         0.2     setosa
## 108          5.0         3.0          1.6         0.2     setosa
## 109          5.0         3.4          1.6         0.4     setosa
## 110          4.7         3.2          1.6         0.2     setosa
## 111          4.8         3.1          1.6         0.2     setosa
## 112          5.0         3.5          1.6         0.6     setosa
## 113          5.1         3.8          1.6         0.2     setosa
## 114          4.6         3.1          1.5         0.2     setosa
## 115          5.0         3.4          1.5         0.2     setosa
## 116          4.9         3.1          1.5         0.1     setosa
## 117          5.4         3.7          1.5         0.2     setosa
## 118          5.7         4.4          1.5         0.4     setosa
## 119          5.1         3.8          1.5         0.3     setosa
## 120          5.1         3.7          1.5         0.4     setosa
## 121          5.2         3.5          1.5         0.2     setosa
## 122          5.4         3.4          1.5         0.4     setosa
## 123          5.2         4.1          1.5         0.1     setosa
## 124          4.9         3.1          1.5         0.2     setosa
## 125          5.1         3.4          1.5         0.2     setosa
## 126          5.3         3.7          1.5         0.2     setosa
## 127          5.1         3.5          1.4         0.2     setosa
## 128          4.9         3.0          1.4         0.2     setosa
## 129          5.0         3.6          1.4         0.2     setosa
## 130          4.6         3.4          1.4         0.3     setosa
## 131          4.4         2.9          1.4         0.2     setosa
## 132          4.8         3.0          1.4         0.1     setosa
## 133          5.1         3.5          1.4         0.3     setosa
## 134          5.2         3.4          1.4         0.2     setosa
## 135          5.5         4.2          1.4         0.2     setosa
## 136          4.9         3.6          1.4         0.1     setosa
## 137          4.8         3.0          1.4         0.3     setosa
## 138          4.6         3.2          1.4         0.2     setosa
## 139          5.0         3.3          1.4         0.2     setosa
## 140          4.7         3.2          1.3         0.2     setosa
## 141          5.4         3.9          1.3         0.4     setosa
## 142          5.5         3.5          1.3         0.2     setosa
## 143          4.4         3.0          1.3         0.2     setosa
## 144          5.0         3.5          1.3         0.3     setosa
## 145          4.5         2.3          1.3         0.3     setosa
## 146          4.4         3.2          1.3         0.2     setosa
## 147          5.8         4.0          1.2         0.2     setosa
## 148          5.0         3.2          1.2         0.2     setosa
## 149          4.3         3.0          1.1         0.1     setosa
## 150          4.6         3.6          1.0         0.2     setosa
```

`summarise()` is a function that transforms a data frame to a single or several values. Here is an example to make it easier to understand what the verb does:

```
# average sepal length
summarise(iris, average_sepallength = mean(Sepal.Length))
```

```
##   average_sepallength
## 1            5.843333
```

```
# Calculating minimum, median, and average sepal length
summarise(iris, min = min(Sepal.Length),
                average = mean(Sepal.Length),
                median = median(Sepal.Length))
```

```
##   min  average median
## 1 4.3 5.843333    5.8
```

The function `summarise()` becomes especialy useful when we combine it with the `group_by()` function, which breaks a dataset down into groups and then the function such as mean and median will be applied to different groups. It is probably easier to understand it with an example:

```
# average and median sepal length grouped by species
summarise(
  group_by(iris, Species),
  avg_sepallength = mean(Sepal.Length),
  median_sepallength = median(Sepal.Length)
)
```

```
## # A tibble: 3 x 3
##      Species avg_sepallength median_sepallength
##       <fctr>           <dbl>              <dbl>
## 1     setosa           5.006                5.0
## 2 versicolor           5.936                5.9
## 3  virginica           6.588                6.5
```

The above functions are those we have learned in lecture and labs. There are also some other useful functions in the dplyr package that we haven't learned yet. For example, we can use the function `sample_n` to sleect random rows from a data frame. The first argument of this function is the name of the data frame, and the second argument is the number of rows to select.

```
# select four rows from iris randomly
sample_n(iris, 4)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 99           5.1         2.5          3.0         1.1 versicolor
## 118          7.7         3.8          6.7         2.2  virginica
## 67           5.6         3.0          4.5         1.5 versicolor
## 137          6.3         3.4          5.6         2.4  virginica
```

We can also randomly select rows using `sample_frac` function, which returns a percentage of rows. The second argument of this function is a fraction between 0 to 1 instead of a fixed number.

```
# randomly selecting 10% of rows of iris
sample_frac(iris, 0.1)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 56           5.7         2.8          4.5         1.3 versicolor
## 120          6.0         2.2          5.0         1.5  virginica
## 122          5.6         2.8          4.9         2.0  virginica
## 23           4.6         3.6          1.0         0.2     setosa
## 90           5.5         2.5          4.0         1.3 versicolor
## 3            4.7         3.2          1.3         0.2     setosa
## 86           6.0         3.4          4.5         1.6 versicolor
## 59           6.6         2.9          4.6         1.3 versicolor
## 82           5.5         2.4          3.7         1.0 versicolor
## 112          6.4         2.7          5.3         1.9  virginica
## 111          6.5         3.2          5.1         2.0  virginica
## 131          7.4         2.8          6.1         1.9  virginica
## 77           6.8         2.8          4.8         1.4 versicolor
## 16           5.7         4.4          1.5         0.4     setosa
## 146          6.7         3.0          5.2         2.3  virginica
```

tidyr

It is important to have tidy data before one proceed to data analysis. The package tidyr offers a few functions that can help us to convert raw daya into tidy data. And for this section of the post, let's create a new data frame `grade`, which contains data about scores of three different homeworks assignments and one quiz received by six students. You can create the exact same data frame using the code chunk below:

```
grade <- data.frame(
  student = c("1", "2", "3", "4", "5", "6"),
  hw.1 = c("80", "70", "85", "100", "60", "90"),
  hw.2 = c("70", "70", "100", "95", "80", "68"),
  hw.3 = c("50", "100", "90", "78", "66", "82"),
  qz.1 = c("100", "92", "76", "63", "87", "89")
)
grade
```

```
##   student hw.1 hw.2 hw.3 qz.1
## 1       1   80   70   50  100
## 2       2   70   70  100   92
## 3       3   85  100   90   76
## 4       4  100   95   78   63
## 5       5   60   80   66   87
## 6       6   90   68   82   89
```

gather(): make wide data long

Sometimes when we have multiple common attribtes as several columns, the verb `gather()` can reformat the data and collect these attributes as a single variable. It takes multiple columns and turns them into key-value pairs. The complete function looks like:
`gather(df, newVar1, newVar2, vector1, vector2, vector3)`

For example, for the `grade` data frame, we have columns `hw.1`, `hw.2`, `hw.3`, and `qz.1` giving us grade information regarding four assignments. If we want to collapse these four columns into one and create a new key-value pair of all assignments and their corresponding scores, we can use `gather()`:

```
# reformating columns hw.1, hw.2, and hw.3 into a single column:
grade_gather <- gather(grade, assignments, score, hw.1, hw.2, hw.3, qz.1)
```

```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```

```
grade_gather
```

```
##    student assignments score
## 1        1        hw.1    80
## 2        2        hw.1    70
## 3        3        hw.1    85
## 4        4        hw.1   100
## 5        5        hw.1    60
## 6        6        hw.1    90
## 7        1        hw.2    70
## 8        2        hw.2    70
## 9        3        hw.2   100
## 10       4        hw.2    95
## 11       5        hw.2    80
## 12       6        hw.2    68
## 13       1        hw.3    50
## 14       2        hw.3   100
## 15       3        hw.3    90
## 16       4        hw.3    78
## 17       5        hw.3    66
## 18       6        hw.3    82
## 19       1        qz.1   100
## 20       2        qz.1    92
## 21       3        qz.1    76
## 22       4        qz.1    63
## 23       5        qz.1    87
## 24       6        qz.1    89
```

spread(): make wide data long

`spread()` is the complement function of `gather()`. And it does the opposite job – transforming long formatted data into wide formatted data. The function spreads the key-value pair (all assignments and their corresponding scores) across three columns, and notice that after applying `spread()`, our data frame looks the same as our original data frame.

```
# Exapanding the column "hw" into three columns
grade_spread <- spread(grade_gather, assignments, score)
grade_spread
```

```
##   student hw.1 hw.2 hw.3 qz.1
## 1       1   80   70   50  100
## 2       2   70   70  100   92
## 3       3   85  100   90   76
## 4       4  100   95   78   63
## 5       5   60   80   66   87
## 6       6   90   68   82   89
```

separate(): Splitting a single variable inside a column into several variables.

Sometimes our data frame has a column that contains several variable, and we often may not care about the more specific groups these variables belong to, but in some situations we may want to separate them into several columns for later data anaysis. For example, the data frame `grade_gather` that we just created using function `gather()` has a column called `assignments` that has values `hw.1`, `hw.2`, `hw.3`, and `qz.1`. `hw` and `qz` are actually grouping variables, and if we want to separate the column `assignments` and transform it into two different columns regarding the type and number of the assignments, we can use `separate()` function. The documentation functions is as following:

`separate(df, col, into, sep = "")`

Arguments:

data: name of data frame

col: column name representng current variable

into: names of columns representing new variables

sep: how to separate current variable

```
# separate assignment column into two columns in respect to the type and number of the assignment
grade_separate <- separate(grade_gather, col = assignments, into = c("Type of Assignment", "# of Assignment"))
grade_separate
```

```
##    student Type of Assignment # of Assignment score
## 1        1                 hw               1    80
## 2        2                 hw               1    70
## 3        3                 hw               1    85
## 4        4                 hw               1   100
## 5        5                 hw               1    60
## 6        6                 hw               1    90
## 7        1                 hw               2    70
## 8        2                 hw               2    70
## 9        3                 hw               2   100
## 10       4                 hw               2    95
## 11       5                 hw               2    80
## 12       6                 hw               2    68
## 13       1                 hw               3    50
## 14       2                 hw               3   100
## 15       3                 hw               3    90
## 16       4                 hw               3    78
## 17       5                 hw               3    66
## 18       6                 hw               3    82
## 19       1                 qz               1   100
## 20       2                 qz               1    92
## 21       3                 qz               1    76
## 22       4                 qz               1    63
## 23       5                 qz               1    87
## 24       6                 qz               1    89
```

unite(): Merging two variables into one

`unite()` is the complement function of `separate()`, and it does the opposite of `separate()`. The function takes several variables and merge them into one. For example, if we feel the information about type and numbering of the assignments are not the kind of information we care about, we can merge these two columns into one using `unite()`. The documentation of the function is as following:

```
unite(df, col, ..., sep = "")
```

Arguments:

df: name of the data frame

col: column anme of the new column that contains merged variables

...: names of columns we want to merge

sep: separator to use between merged values

```
# Merging columns "Type of Assignment" and "# of Assignment" into one column
grade_unite <- unite(grade_separate, Assignment, "Type of Assignment", "# of Assignment", sep = ".")
grade_unite
```

```
##    student Assignment score
## 1        1       hw.1    80
## 2        2       hw.1    70
## 3        3       hw.1    85
## 4        4       hw.1   100
## 5        5       hw.1    60
## 6        6       hw.1    90
## 7        1       hw.2    70
## 8        2       hw.2    70
## 9        3       hw.2   100
## 10       4       hw.2    95
## 11       5       hw.2    80
## 12       6       hw.2    68
## 13       1       hw.3    50
## 14       2       hw.3   100
## 15       3       hw.3    90
## 16       4       hw.3    78
## 17       5       hw.3    66
## 18       6       hw.3    82
## 19       1       qz.1   100
## 20       2       qz.1    92
## 21       3       qz.1    76
## 22       4       qz.1    63
## 23       5       qz.1    87
## 24       6       qz.1    89
```

Conclusion

To summarize, the purpose of this post was to give the reader a quick overview of the important functions in dplyr and tidyr package. We saw some demonstrations of the functions and how we can use them to manipulate, aggregate, and transform the data frame in different ways. Although the data frames that we worked with in this post are rather simple compared to datasets in the real-world, which are far more complex, one can still apply these functions in the same logic. The take-away: functions in dyplyr and tidyr can help us work more efficiently with raw datasets during the data preparation process, which is crucial in most situtaions before performing any data analysis.

References:

- http://genomicsclass.github.io/book/pages/dplyr_tutorial.html
- https://www.r-bloggers.com/data-manipulation-with-dplyr/
- https://spark.rstudio.com/dplyr/
- http://www.listendata.com/2016/08/dplyr-tutorial.html
- https://rpubs.com/bradleyboehmke/data_wrangling
- http://www.jvcasillas.com/tidyr_tutorial/
- http://jules32.github.io/2016-07-12-Oxford/dplyr_tidyr/
- http://dplyr.tidyverse.org