Data Manipulation using dplyr in R

Bheeni Garg

Sunday, April 19, 2015

Objective

The project aims at simplifying data manipulation using dplyr package by elucidating on the most common data manipulation operations so that one's thoughts are easily translated into code. It introduces the basic tools that dplyr provides and applying them to the data frame. The project also aims at comparing the functionalities of the most common data manipulation verbs with the basic R functions and highlighting the proficiency of the former. One of the best features of dplyr which could not be highlighted in this project is that- it's really fast!

About dplyr

dplyr is a package in R for data manipulation and exploration written by Hadley Wickham who is also the author of the famous ggplot2 package in R. plyr, reshape, testthat are some of the other packages that he authored as well. dplyr is the next iteration of plyr, focussing exclusively on data frames. It is faster, has a more consistent API and should be easier to use. It is built around five verbs or five core functions and also holds one of the most interesting and important features called `chaining` or `pipelining`.

More pointers: dplyr

1. uses five basic verbs: `filter`,`select` (helper functions include `contains`,`starts_with`,`ends_with`,`matches`),`arrange`,`mut`
   `group_by` and `n`).
2. can work with data stored in databases and data tables.
3. results in output which is always a data frame.
4. follows a simple syntax: a verb with original data frame as the first argument followed filtering conditions in the subsequent arguments which are evaluated in the context of the data frame.

Why dplyr?

- **Speed**: dplyr is much, much faster than other, more traditional functions
- **Syntax simplicity and ease of use**: The code is easy to write and read and hence much easier to follow. It is almost intuitive.
- **Function chaining**: this allows a cleaner code and avoids cluttering of the workspace with interim objects.
- **Direct connection to and analysis within external databases**: this permits simpler handling of large data

Getting Started

Load dplyr dplyr will mask a few base functions

```
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
```

Example dataset

The 1950-2014_tor dataset comes from a tornado report that contains a collection of variables recorded between 1950 and 2014. The data set comes from NOAA's National Weather Service field offices. The description of the data is given here.

```
tornado <- read.csv('1950-2014_torn.csv', stringsAsFactors = FALSE)
head(tornado)
```

1. Read-in the dataset

```
##   Tornado_number Year Month Day      Date     Time Time_zone State
## 1              1 1950     1   3  1/3/1950 11:00:00         3    MO
## 2              1 1950     1   3  1/3/1950 11:00:00         3    MO
## 3              1 1950     1   3  1/3/1950 11:10:00         3    IL
## 4              2 1950     1   3  1/3/1950 11:55:00         3    IL
## 5              3 1950     1   3  1/3/1950 16:00:00         3    OH
## 6              4 1950     1  13 1/13/1950  5:25:00         3    AR
##   FIPS_number State_number F_scale Injuries Fatalities Property_loss
## 1          29            1       3        3          0             6
## 2          29            1       3        3          0             6
## 3          17            1       3        0          0             5
## 4          17            2       3        3          0             5
## 5          39            1       1        1          0             4
## 6           5            1       3        1          1             3
##   Crop_loss S_lat  S_lon E_lat  E_lon Length Width ns sn sg  f1 f2 f3 f4
## 1         0 38.77 -90.22 38.83 -90.03    9.5   150  2  0  1   0  0  0  0
## 2         0 38.77 -90.22 38.82 -90.12    6.2   150  2  1  2 189  0  0  0
## 3         0 38.82 -90.12 38.83 -90.03    3.3   100  2  1  2 119  0  0  0
## 4         0 39.10 -89.30 39.12 -89.23    3.6   130  1  1  1 135  0  0  0
## 5         0 40.88 -84.58  0.00   0.00    0.1    10  1  1  1 161  0  0  0
## 6         0 34.40 -94.37  0.00   0.00    0.6    17  1  1  1 113  0  0  0
```

```
tornado <- tbl_df(read.csv("1950-2014_torn.csv", stringsAsFactors = FALSE))
tornado
```

2. Convert to local data frame using tbl_df()

```
## Source: local data frame [59,945 x 28]
##
##    Tornado_number  Year Month   Day      Date     Time Time_zone State
##             (int) (int) (int) (int)     (chr)    (chr)     (int) (chr)
## 1               1  1950     1     3  1/3/1950 11:00:00         3    MO
## 2               1  1950     1     3  1/3/1950 11:00:00         3    MO
## 3               1  1950     1     3  1/3/1950 11:10:00         3    IL
## 4               2  1950     1     3  1/3/1950 11:55:00         3    IL
## 5               3  1950     1     3  1/3/1950 16:00:00         3    OH
## 6               4  1950     1    13 1/13/1950  5:25:00         3    AR
## 7               5  1950     1    25 1/25/1950 19:30:00         3    MO
## 8               6  1950     1    25 1/25/1950 21:00:00         3    IL
## 9               7  1950     1    26 1/26/1950 18:00:00         3    TX
## 10              8  1950     2    11 2/11/1950 13:10:00         3    TX
## ..            ...   ...   ...   ...       ...      ...       ...   ...
## Variables not shown: FIPS_number (int), State_number (int), F_scale (int),
##   Injuries (int), Fatalities (int), Property_loss (int), Crop_loss (int),
##   S_lat (dbl), S_lon (dbl), E_lat (dbl), E_lon (dbl), Length (dbl), Width
##   (int), ns (int), sn (int), sg (int), f1 (int), f2 (int), f3 (int), f4
##   (int)
```

- the function tbl_df creates a 'local data frame'
- it prints out the data, showing the first 10 rows and only shows as many columns as could fit to the screen. The rest of the columns are listed below with some of their details. Try widening the window and re-run the above command. Notice more columns being added to the screen.
- this is a wrapper around a data frame that prevents a data deluge which usually occurs when using a simple data frame, where a lot of data accidentally gets printed to the console, which is barely comprehensible.

```
print(tornado, n= 20)
```

3. To view more rows, specify the number of rows

```
## Source: local data frame [59,945 x 28]
##
##    Tornado_number  Year Month   Day      Date     Time Time_zone State
##             (int) (int) (int) (int)     (chr)    (chr)     (int) (chr)
## 1               1  1950     1     3  1/3/1950 11:00:00         3    MO
## 2               1  1950     1     3  1/3/1950 11:00:00         3    MO
## 3               1  1950     1     3  1/3/1950 11:10:00         3    IL
## 4               2  1950     1     3  1/3/1950 11:55:00         3    IL
## 5               3  1950     1     3  1/3/1950 16:00:00         3    OH
## 6               4  1950     1    13 1/13/1950  5:25:00         3    AR
```

```
## 7               5  1950     1    25 1/25/1950 19:30:00         3    MO
## 8               6  1950     1    25 1/25/1950 21:00:00         3    IL
## 9               7  1950     1    26 1/26/1950 18:00:00         3    TX
## 10              8  1950     2    11 2/11/1950 13:10:00         3    TX
## 11              9  1950     2    11 2/11/1950 13:50:00         3    TX
## 12             10  1950     2    11 2/11/1950 21:00:00         3    TX
## 13             11  1950     2    11 2/11/1950 23:55:00         3    TX
## 14             12  1950     2    12 2/12/1950  0:30:00         3    TX
## 15             13  1950     2    12 2/12/1950  1:15:00         3    TX
## 16             14  1950     2    12 2/12/1950  6:10:00         3    TX
## 17             15  1950     2    12 2/12/1950 11:57:00         3    TX
## 18             18  1950     2    12 2/12/1950 12:00:00         3    TX
## 19             17  1950     2    12 2/12/1950 12:00:00         3    MS
## 20             16  1950     2    12 2/12/1950 12:00:00         3    MS
## ..            ...   ...   ...   ...       ...      ...       ...   ...
## Variables not shown: FIPS_number (int), State_number (int), F_scale (int),
##   Injuries (int), Fatalities (int), Property_loss (int), Crop_loss (int),
##   S_lat (dbl), S_lon (dbl), E_lat (dbl), E_lon (dbl), Length (dbl), Width
##   (int), ns (int), sn (int), sg (int), f1 (int), f2 (int), f3 (int), f4
##   (int)
```

The first verb: filter

- Keep rows matching criteria
- lets us select a set of rows from an original data frame according to some condition

Example 1

```
tornado[tornado$State == "TX" & tornado$Year > 1970,]
```

base R approach to view all tornadoes that occurred in Texas after 1970

```
## Source: local data frame [6,357 x 28]
##
##    Tornado_number  Year Month   Day      Date     Time Time_zone State
##             (int) (int) (int) (int)     (chr)    (chr)     (int) (chr)
## 1              48  1971     2    18 2/18/1971 11:00:00         3    TX
## 2              49  1971     2    18 2/18/1971 11:00:00         3    TX
## 3              53  1971     2    18 2/18/1971 14:55:00         3    TX
## 4              55  1971     2    18 2/18/1971 16:45:00         3    TX
## 5              56  1971     2    18 2/18/1971 17:20:00         3    TX
## 6              57  1971     2    18 2/18/1971 18:00:00         3    TX
## 7              58  1971     2    18 2/18/1971 18:30:00         3    TX
## 8              62  1971     2    21 2/21/1971  7:30:00         3    TX
## 9              63  1971     2    21 2/21/1971  8:15:00         3    TX
## 10             82  1971     2    25 2/25/1971  9:20:00         3    TX
## ..            ...   ...   ...   ...       ...      ...       ...   ...
## Variables not shown: FIPS_number (int), State_number (int), F_scale (int),
##   Injuries (int), Fatalities (int), Property_loss (int), Crop_loss (int),
```

```
##   S_lat (dbl), S_lon (dbl), E_lat (dbl), E_lon (dbl), Length (dbl), Width
##   (int), ns (int), sn (int), sg (int), f1 (int), f2 (int), f3 (int), f4
##   (int)
```

dplyr approach

- dplyr involves simple, self-explanatory syntax whereby the use of '$' is eliminated
- note: you can use comma or ampersand to represent AND condition

```
filter(tornado, State == "TX", Year > 1970)
```

```
## Source: local data frame [6,357 x 28]
##
##    Tornado_number  Year Month   Day      Date     Time Time_zone State
##             (int) (int) (int) (int)     (chr)    (chr)     (int) (chr)
## 1              48  1971     2    18 2/18/1971 11:00:00         3    TX
## 2              49  1971     2    18 2/18/1971 11:00:00         3    TX
## 3              53  1971     2    18 2/18/1971 14:55:00         3    TX
## 4              55  1971     2    18 2/18/1971 16:45:00         3    TX
## 5              56  1971     2    18 2/18/1971 17:20:00         3    TX
## 6              57  1971     2    18 2/18/1971 18:00:00         3    TX
## 7              58  1971     2    18 2/18/1971 18:30:00         3    TX
## 8              62  1971     2    21 2/21/1971  7:30:00         3    TX
## 9              63  1971     2    21 2/21/1971  8:15:00         3    TX
## 10             82  1971     2    25 2/25/1971  9:20:00         3    TX
## ..            ...   ...   ...   ...       ...      ...       ...   ...
## Variables not shown: FIPS_number (int), State_number (int), F_scale (int),
##   Injuries (int), Fatalities (int), Property_loss (int), Crop_loss (int),
##   S_lat (dbl), S_lon (dbl), E_lat (dbl), E_lon (dbl), Length (dbl), Width
##   (int), ns (int), sn (int), sg (int), f1 (int), f2 (int), f3 (int), f4
##   (int)
```

```
filter(tornado, State == "TX" & Year > 1970)
```

```
## Source: local data frame [6,357 x 28]
##
##    Tornado_number  Year Month   Day      Date     Time Time_zone State
##             (int) (int) (int) (int)     (chr)    (chr)     (int) (chr)
## 1              48  1971     2    18 2/18/1971 11:00:00         3    TX
## 2              49  1971     2    18 2/18/1971 11:00:00         3    TX
## 3              53  1971     2    18 2/18/1971 14:55:00         3    TX
## 4              55  1971     2    18 2/18/1971 16:45:00         3    TX
## 5              56  1971     2    18 2/18/1971 17:20:00         3    TX
## 6              57  1971     2    18 2/18/1971 18:00:00         3    TX
## 7              58  1971     2    18 2/18/1971 18:30:00         3    TX
## 8              62  1971     2    21 2/21/1971  7:30:00         3    TX
## 9              63  1971     2    21 2/21/1971  8:15:00         3    TX
## 10             82  1971     2    25 2/25/1971  9:20:00         3    TX
## ..            ...   ...   ...   ...       ...      ...       ...   ...
## Variables not shown: FIPS_number (int), State_number (int), F_scale (int),
##   Injuries (int), Fatalities (int), Property_loss (int), Crop_loss (int),
##   S_lat (dbl), S_lon (dbl), E_lat (dbl), E_lon (dbl), Length (dbl), Width
```

```
##   (int), ns (int), sn (int), sg (int), f1 (int), f2 (int), f3 (int), f4
##   (int)
```

Example 2

```
tornado[tornado$State == "MO" | tornado$State == "MS",] # using pipe for OR condition
```

base R approach to view all tornadoes that occurred in Missouri or Mississippi

```
## Source: local data frame [4,225 x 28]
##
##    Tornado_number  Year Month   Day      Date     Time Time_zone State
##             (int) (int) (int) (int)     (chr)    (chr)     (int) (chr)
## 1               1  1950     1     3  1/3/1950 11:00:00         3    MO
## 2               1  1950     1     3  1/3/1950 11:00:00         3    MO
## 3               5  1950     1    25 1/25/1950 19:30:00         3    MO
## 4              17  1950     2    12 2/12/1950 12:00:00         3    MS
## 5              16  1950     2    12 2/12/1950 12:00:00         3    MS
## 6              28  1950     3     1  3/1/1950  2:30:00         3    MS
## 7              39  1950     3    27 3/27/1950  5:00:00         3    MS
## 8              41  1950     3    27 3/27/1950  7:30:00         3    MS
## 9              43  1950     3    27 3/27/1950  7:45:00         3    MS
## 10             43  1950     3    27 3/27/1950  7:45:00         3    MS
## ..            ...   ...   ...   ...       ...      ...       ...   ...
## Variables not shown: FIPS_number (int), State_number (int), F_scale (int),
##   Injuries (int), Fatalities (int), Property_loss (int), Crop_loss (int),
##   S_lat (dbl), S_lon (dbl), E_lat (dbl), E_lon (dbl), Length (dbl), Width
##   (int), ns (int), sn (int), sg (int), f1 (int), f2 (int), f3 (int), f4
##   (int)
```

```
filter(tornado, State == "MO" | State == "MS")
```

dplyr approach

```
## Source: local data frame [4,225 x 28]
##
##    Tornado_number  Year Month   Day      Date     Time Time_zone State
##             (int) (int) (int) (int)     (chr)    (chr)     (int) (chr)
## 1               1  1950     1     3  1/3/1950 11:00:00         3    MO
## 2               1  1950     1     3  1/3/1950 11:00:00         3    MO
## 3               5  1950     1    25 1/25/1950 19:30:00         3    MO
## 4              17  1950     2    12 2/12/1950 12:00:00         3    MS
## 5              16  1950     2    12 2/12/1950 12:00:00         3    MS
## 6              28  1950     3     1  3/1/1950  2:30:00         3    MS
## 7              39  1950     3    27 3/27/1950  5:00:00         3    MS
## 8              41  1950     3    27 3/27/1950  7:30:00         3    MS
## 9              43  1950     3    27 3/27/1950  7:45:00         3    MS
```

```
## 10             43  1950     3    27 3/27/1950  7:45:00         3    MS
## ..            ...   ...   ...   ...       ...      ...       ...   ...
## Variables not shown: FIPS_number (int), State_number (int), F_scale (int),
##   Injuries (int), Fatalities (int), Property_loss (int), Crop_loss (int),
##   S_lat (dbl), S_lon (dbl), E_lat (dbl), E_lon (dbl), Length (dbl), Width
##   (int), ns (int), sn (int), sg (int), f1 (int), f2 (int), f3 (int), f4
##   (int)
```

```
filter(tornado, State %in% c("MO", "MS"))
```

using the %in% operator

```
## Source: local data frame [4,225 x 28]
##
##    Tornado_number  Year Month   Day      Date     Time Time_zone State
##             (int) (int) (int) (int)     (chr)    (chr)     (int) (chr)
## 1               1  1950     1     3  1/3/1950 11:00:00         3    MO
## 2               1  1950     1     3  1/3/1950 11:00:00         3    MO
## 3               5  1950     1    25 1/25/1950 19:30:00         3    MO
## 4              17  1950     2    12 2/12/1950 12:00:00         3    MS
## 5              16  1950     2    12 2/12/1950 12:00:00         3    MS
## 6              28  1950     3     1  3/1/1950  2:30:00         3    MS
## 7              39  1950     3    27 3/27/1950  5:00:00         3    MS
## 8              41  1950     3    27 3/27/1950  7:30:00         3    MS
## 9              43  1950     3    27 3/27/1950  7:45:00         3    MS
## 10             43  1950     3    27 3/27/1950  7:45:00         3    MS
## ..            ...   ...   ...   ...       ...      ...       ...   ...
## Variables not shown: FIPS_number (int), State_number (int), F_scale (int),
##   Injuries (int), Fatalities (int), Property_loss (int), Crop_loss (int),
##   S_lat (dbl), S_lon (dbl), E_lat (dbl), E_lon (dbl), Length (dbl), Width
##   (int), ns (int), sn (int), sg (int), f1 (int), f2 (int), f3 (int), f4
##   (int)
```

Example 3

view all tornadoes that occurred in Alabama between 1980 and 2000 of intensity(`F-scale`) greater than or equal to 4 *include as many conditions as you like

```
filter(tornado, State == "AL", Year > 1980, Year < 2000, F_scale >= 4)
```

```
## Source: local data frame [4 x 28]
##
##   Tornado_number  Year Month   Day       Date     Time Time_zone State
##            (int) (int) (int) (int)      (chr)    (chr)     (int) (chr)
## 1            816  1989    11    15 11/15/1989 16:30:00         3    AL
## 2             50  1994     3    27  3/27/1994 10:55:00         3    AL
## 3            481  1995     5    18  5/18/1995 16:33:00         3    AL
## 4            833  1998     4     8   4/8/1998 18:42:00         3    AL
## Variables not shown: FIPS_number (int), State_number (int), F_scale (int),
```

```
##   Injuries (int), Fatalities (int), Property_loss (int), Crop_loss (int),
##   S_lat (dbl), S_lon (dbl), E_lat (dbl), E_lon (dbl), Length (dbl), Width
##   (int), ns (int), sn (int), sg (int), f1 (int), f2 (int), f3 (int), f4
##   (int)
```

Second verb : select

- Pick columns by name
- lets us subset the data frame by columns

Example 4

```
tornado[, c("Time_zone","State","Injuries")]
```

base R approach to select `Time_zone`, `State` and `Injuries` column

```
## Source: local data frame [59,945 x 3]
##
##    Time_zone State Injuries
##        (int) (chr)    (int)
## 1          3    MO        3
## 2          3    MO        3
## 3          3    IL        0
## 4          3    IL        3
## 5          3    OH        1
## 6          3    AR        1
## 7          3    MO        5
## 8          3    IL        0
## 9          3    TX        2
## 10         3    TX        0
## ..       ...   ...      ...
```

```
select(tornado, Time_zone, State, Injuries)
```

dplyr approach

```
## Source: local data frame [59,945 x 3]
##
##    Time_zone State Injuries
##        (int) (chr)    (int)
## 1          3    MO        3
## 2          3    MO        3
## 3          3    IL        0
## 4          3    IL        3
## 5          3    OH        1
## 6          3    AR        1
```

```
## 7          3    MO        5
## 8          3    IL        0
## 9          3    TX        2
## 10         3    TX        0
## ..       ...   ...      ...
```

- colon can be used to select multiple contiguous columns and `contains` to match the columns by name
- note: helper functions like `starts_with`,`ends_with` and `matches` (for regular expressions) can also be used to match columns by name

Example 5

```
select(tornado, Date:State, contains ("loss"))
```

select columns from `Date` through `State` and all that contains the word "loss"

```
## Source: local data frame [59,945 x 6]
##
##         Date     Time Time_zone State Property_loss Crop_loss
##        (chr)    (chr)     (int) (chr)         (int)     (int)
## 1   1/3/1950 11:00:00         3    MO             6         0
## 2   1/3/1950 11:00:00         3    MO             6         0
## 3   1/3/1950 11:10:00         3    IL             5         0
## 4   1/3/1950 11:55:00         3    IL             5         0
## 5   1/3/1950 16:00:00         3    OH             4         0
## 6  1/13/1950  5:25:00         3    AR             3         0
## 7  1/25/1950 19:30:00         3    MO             5         0
## 8  1/25/1950 21:00:00         3    IL             5         0
## 9  1/26/1950 18:00:00         3    TX             0         0
## 10 2/11/1950 13:10:00         3    TX             4         0
## ..       ...      ...       ...   ...           ...       ...
```

Nesting functions

When we want to perform multiple operations or use more than one verb in a single line code, for example using `filter` and `select` in the same line:

Example 6

```
select(filter(tornado, State == "KS"), Year, Property_loss, starts_with("f"))
```

view all tornadoes that have occurred in the state of Kansas and select columns `Year`, `Property_loss` and those which start with 'f'

```
## Source: local data frame [3,968 x 9]
##
##     Year Property_loss FIPS_number F_scale Fatalities    f1    f2    f3
##    (int)         (int)       (int)   (int)      (int) (int) (int) (int)
## 1   1950             4          20       1          0    31     0     0
## 2   1950             4          20       1          0   165     0     0
## 3   1950             5          20       4          0     9   145     0
## 4   1950             5          20       3          0    85    13     0
## 5   1950             5          20       2          0   123     0     0
## 6   1950             4          20       2          0    89     0     0
## 7   1950             4          20       2          0   141     0     0
## 8   1950             4          20       1          0    29     0     0
## 9   1950             0          20       1          0   157     0     0
## 10  1950             4          20      -9          0   161     0     0
## ..   ...           ...         ...     ...        ...   ...   ...   ...
## Variables not shown: f4 (int)
```

The above method of wrapping function calls inside each other can get tricky if we want to do many operations at once. This doesn't lead to a particularly elegant code as it becomes difficult to read because the order of the operations is from inside to out, and the arguments are a long way away from the function. To get around this problem, dplyr provides the infix operator (%>%), discussed next.

"Chaining" or "Pipelining"

- Used to perform multiple operations in one line by nesting
- The commands can be written in a natural order using the pipe operator (%>%). The R language allows symbols wrapped in '%' to be defined as functions, the '>' helps imply a chain.
- Pipes take the output from one function and feed it to the first argument of the next function.

Working Example 6 with chaining:

```
tornado %>%
  filter(State == "KS") %>%
  select(Year, Property_loss, starts_with("f"))
```

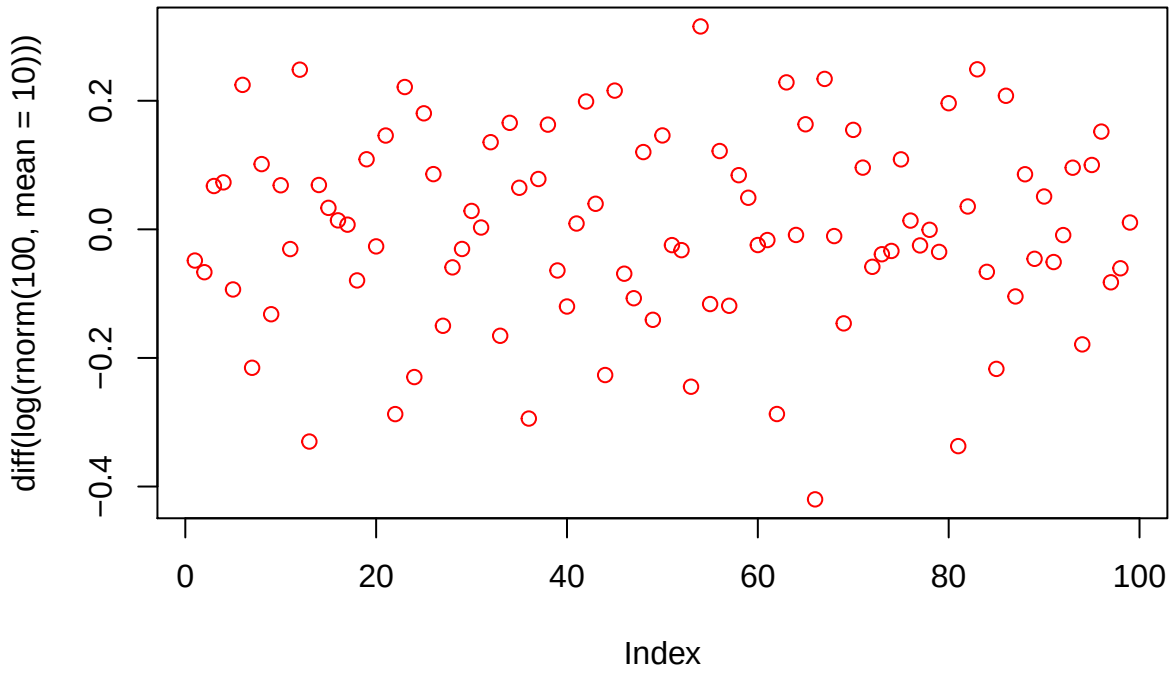
```
## Source: local data frame [3,968 x 9]
##
##     Year Property_loss FIPS_number F_scale Fatalities    f1    f2    f3
##    (int)         (int)       (int)   (int)      (int) (int) (int) (int)
## 1   1950             4          20       1          0    31     0     0
## 2   1950             4          20       1          0   165     0     0
## 3   1950             5          20       4          0     9   145     0
## 4   1950             5          20       3          0    85    13     0
## 5   1950             5          20       2          0   123     0     0
## 6   1950             4          20       2          0    89     0     0
## 7   1950             4          20       2          0   141     0     0
## 8   1950             4          20       1          0    29     0     0
## 9   1950             0          20       1          0   157     0     0
## 10  1950             4          20      -9          0   161     0     0
## ..   ...           ...         ...     ...        ...   ...   ...   ...
## Variables not shown: f4 (int)
```

So, we took the tornado data frame, fed it to filter() to extract rows corresponding to Kansas state and then selected columns we wanted to see.

dplyr lets us write the data manipulation steps in the order we think of them and avoid creating temporary variables in the middle to capture the output. This works because the output from every dplyr function is a data frame and the first argument of every dplyr function is a data frame.

Trivial non-dplyr chaining example Suppose we want to plot the log differences of 100 normally distributed random numbers with mean 10. The traditional code would look like this:

```
plot(diff(log(rnorm(100, mean = 10))), col = "red")
```




```
rnorm(100, mean = 10) %>%
  log %>%
  diff %>%
  plot(col = "red")
```

With infix (%>%) operator, the code can be restructured

- The pipe operator %>% (pronounced as then) is used to chain multiple operations together. So, the above operation can be read as:
- generate 100 normally distributed random numers with mean 10 then take the log of those numbers then
- calculate the difference between the logs taken then
- plot the differences with the color red.
- it can also be noted that chaining increases the readability of code and removes intermittent objects.

Third verb: arrange

- Reorder rows
- allows reordering of rows by one or more columns in ascending(default) or descending order.

Example 7

```
tornado[order(tornado$Length, decreasing = TRUE), c("State", "Length")]
```

base R approach to sort all tornadoes by decreasing order of `Length` and identify the state which experienced the longest tornado

```
## Source: local data frame [59,945 x 2]
##
##    State Length
##    (chr)  (dbl)
## 1     LA  234.7
## 2     GA  217.8
## 3     MS  202.5
## 4     MS  202.1
## 5     MS  198.5
## 6     NE  176.4
## 7     OK  170.5
## 8     KS  169.7
## 9     FL  168.5
## 10    IA  162.0
## ..   ...    ...
```

```
arr <- arrange(tornado, desc(Length)) # assigning the sorted Length to `arr`
select(arr, Length, State)        # select columns `Length` and `State` from data frame `arr`
```

dplyr approach - 1

```
## Source: local data frame [59,945 x 2]
##
##    Length State
##     (dbl) (chr)
## 1   234.7    LA
## 2   217.8    GA
## 3   202.5    MS
## 4   202.1    MS
## 5   198.5    MS
## 6   176.4    NE
## 7   170.5    OK
## 8   169.7    KS
## 9   168.5    FL
## 10  162.0    IA
## ..    ...   ...
```

```
tornado %>%
  arrange(desc(Length)) %>%
  select(State, Length)
```

dplyr approach - 2 : using the pipe operator

```
## Source: local data frame [59,945 x 2]
##
##    State Length
##    (chr)  (dbl)
## 1     LA  234.7
## 2     GA  217.8
## 3     MS  202.5
## 4     MS  202.1
## 5     MS  198.5
## 6     NE  176.4
## 7     OK  170.5
## 8     KS  169.7
## 9     FL  168.5
## 10    IA  162.0
## ..   ...    ...
```

Example 8

```
arrange(tornado, desc(Date), desc(Time))
```

sort the tornado data by date and time and identify the state which had the most recent tornado

```
## Source: local data frame [59,945 x 28]
##
##    Tornado_number  Year Month   Day     Date     Time Time_zone State
##             (int) (int) (int) (int)    (chr)    (chr)     (int) (chr)
## 1          543445  2014     9     9 9/9/2014 21:22:00         3    MO
## 2          542447  2014     9     9 9/9/2014 20:10:00         3    MO
## 3          543444  2014     9     9 9/9/2014 18:40:00         3    MO
## 4          543443  2014     9     9 9/9/2014 18:35:00         3    MO
## 5          543441  2014     9     9 9/9/2014 18:10:00         3    MO
## 6          543440  2014     9     9 9/9/2014 16:50:00         3    MO
## 7          533721  2014     9     9 9/9/2014 15:29:00         3    NE
## 8          411603  2012     9     9 9/9/2012 15:00:00         3    CA
## 9            1050  2010     9     9 9/9/2010 16:20:00         3    MT
## 10           1049  2010     9     9 9/9/2010 16:00:00         3    ND
## ..            ...   ...   ...   ...      ...      ...       ...   ...
## Variables not shown: FIPS_number (int), State_number (int), F_scale (int),
##   Injuries (int), Fatalities (int), Property_loss (int), Crop_loss (int),
##   S_lat (dbl), S_lon (dbl), E_lat (dbl), E_lon (dbl), Length (dbl), Width
##   (int), ns (int), sn (int), sg (int), f1 (int), f2 (int), f3 (int), f4
##   (int)
```

Fourth verb: mutate

- add new variables
- The new variables are functions of the existing variables

Example 9

```
tornado$prop <- tornado$Length / tornado$Width
tornado[, c("Length","Width","prop")]
```

base R approach to create a new variable `prop` which is the ratio of `Length` to `Width`and another variable `Severity` created using the variables `Fatalities` and `Injuries`

```
## Source: local data frame [59,945 x 3]
##
##    Length Width        prop
##     (dbl) (int)       (dbl)
## 1     9.5   150 0.063333333
## 2     6.2   150 0.041333333
## 3     3.3   100 0.033000000
## 4     3.6   130 0.027692308
## 5     0.1    10 0.010000000
## 6     0.6    17 0.035294118
## 7     2.3   300 0.007666667
## 8     0.1   100 0.001000000
## 9     4.7   133 0.035338346
## 10    9.9   400 0.024750000
## ..    ...   ...         ...
```

Example 10

```
tornado$Severity <- 100*tornado$Fatalities + 2*tornado$Injuries
tornado[, c("Fatalities","Injuries","Severity")]
```

base R approach to create a new variable, severity created using the variables `Fatalities` and `Injuries`

```
## Source: local data frame [59,945 x 3]
##
##    Fatalities Injuries Severity
##         (int)    (int)    (dbl)
## 1           0        3        6
## 2           0        3        6
## 3           0        0        0
## 4           0        3        6
## 5           0        1        2
## 6           1        1      102
```

```
## 7           0        5       10
## 8           0        0        0
## 9           0        2        4
## 10          0        0        0
## ..        ...      ...      ...
```

dplyr approach - 1

- note: dplyr prints the new variable but does not store it
- new variable can be saved by assigning it

```
new <- mutate(tornado, prop = Length/Width, Severity = 100*Fatalities + 2*Injuries) # create new variab

select(new, prop, Severity)           # print the new variables prop and Severity by selecting columns
```

```
## Source: local data frame [59,945 x 2]
##
##           prop Severity
##          (dbl)    (dbl)
## 1  0.063333333        6
## 2  0.041333333        6
## 3  0.033000000        0
## 4  0.027692308        6
## 5  0.010000000        2
## 6  0.035294118      102
## 7  0.007666667       10
## 8  0.001000000        0
## 9  0.035338346        4
## 10 0.024750000        0
## ..         ...      ...
```

```
tornado %>%
  select(Length, Width, Fatalities, Injuries) %>%
  mutate(prop = Length/Width, Severity = 100*Fatalities + 2*Injuries)
```

dplyr approach- 2 : using the %>% operator

```
## Source: local data frame [59,945 x 6]
##
##    Length Width Fatalities Injuries        prop Severity
##     (dbl) (int)      (int)    (int)       (dbl)    (dbl)
## 1     9.5   150          0        3 0.063333333        6
## 2     6.2   150          0        3 0.041333333        6
## 3     3.3   100          0        0 0.033000000        0
## 4     3.6   130          0        3 0.027692308        6
## 5     0.1    10          0        1 0.010000000        2
## 6     0.6    17          1        1 0.035294118      102
## 7     2.3   300          0        5 0.007666667       10
## 8     0.1   100          0        0 0.001000000        0
```

```
## 9     4.7   133          0        2 0.035338346        4
## 10    9.9   400          0        0 0.024750000        0
## ..    ...   ...        ...      ...         ...      ...
```

`mutate` does not require selection of variables to create a new variable

```
tornado %>%
  mutate(prop = Length/Width, Severity = 100*Fatalities + 2*Injuries) # creates variables prop and Seve
```

```
## Source: local data frame [59,945 x 30]
##
##    Tornado_number  Year Month   Day      Date     Time Time_zone State
##             (int) (int) (int) (int)     (chr)    (chr)     (int) (chr)
## 1               1  1950     1     3  1/3/1950 11:00:00         3    MO
## 2               1  1950     1     3  1/3/1950 11:00:00         3    MO
## 3               1  1950     1     3  1/3/1950 11:10:00         3    IL
## 4               2  1950     1     3  1/3/1950 11:55:00         3    IL
## 5               3  1950     1     3  1/3/1950 16:00:00         3    OH
## 6               4  1950     1    13 1/13/1950  5:25:00         3    AR
## 7               5  1950     1    25 1/25/1950 19:30:00         3    MO
## 8               6  1950     1    25 1/25/1950 21:00:00         3    IL
## 9               7  1950     1    26 1/26/1950 18:00:00         3    TX
## 10              8  1950     2    11 2/11/1950 13:10:00         3    TX
## ..            ...   ...   ...   ...       ...      ...       ...   ...
## Variables not shown: FIPS_number (int), State_number (int), F_scale (int),
##   Injuries (int), Fatalities (int), Property_loss (int), Crop_loss (int),
##   S_lat (dbl), S_lon (dbl), E_lat (dbl), E_lon (dbl), Length (dbl), Width
##   (int), ns (int), sn (int), sg (int), f1 (int), f2 (int), f3 (int), f4
##   (int), prop (dbl), Severity (dbl)
```

Fifth and final verb : summarise

- reduce variables to values
- calculates summary statistics and hence collapses a data frame to a single row
- performs operations on data that has been grouped by one or more variables
- group_by is used to summarise chunks of data

Example 11

```
with(tornado, tapply(F_scale, Year, max))
```

base R approach to get the maximum `F-scale` (tornado intensity) for each year

```
## 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964
##    4    4    4    5    4    5    5    5    5    4    5    4    4    4    5
## 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979
##    5    5    4    5    4    5    5    4    5    5    4    5    5    4    4
```

```
## 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994
##    4    4    5    4    5    5    4    4    4    4    5    5    5    4    4
## 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009
##    4    5    5    5    5    4    4    4    4    4    4    4    5    5    4
## 2010 2011 2012 2013 2014
##    4    5    4    5    4
```

```
by_year <- group_by(tornado, Year)              # group the tornado data by Year
summarise(by_year, max.F_scale =  max(F_scale)) # use summarise to calculate the maximum F_scale value
```

dplyr approach

```
## Source: local data frame [65 x 2]
##
##     Year max.F_scale
##    (int)       (int)
## 1   1950           4
## 2   1951           4
## 3   1952           4
## 4   1953           5
## 5   1954           4
## 6   1955           5
## 7   1956           5
## 8   1957           5
## 9   1958           5
## 10  1959           4
## ..   ...         ...
```

Example 12

```
by_state <- group_by(tornado, State)            # group by state
summarise(by_state, max.F_scale = max(F_scale)) # use summarise to calculate the maximum F_scale value
```

get maximum F-scale by state

```
## Source: local data frame [49 x 2]
##
##    State max.F_scale
##    (chr)       (int)
## 1     AL           5
## 2     AR           4
## 3     AZ           3
## 4     CA           3
## 5     CO           4
## 6     CT           4
## 7     DC           0
## 8     DE           3
## 9     FL           4
## 10    GA           4
## ..   ...         ...
```

More on summarise

- n() counts the number of individuals

Example 13

```
summarise(by_state,                                          # group by state
          num = n(),                                         # total number of tornadoes
          num.strong = sum(F_scale %in% c(4,5)),             # number of strong tornadoes calculated by a
          prop.strong = num.strong/num,                       # proportion of strong tornadoes
          avg_severity = mean(100*Fatalities + 2*Injuries) # average severity costs
          )
```

```
## Source: local data frame [49 x 5]
##
##    State   num num.strong  prop.strong avg_severity
##    (chr) (int)      (int)        (dbl)        (dbl)
## 1     AL  2011         50 0.0248632521   48.3958230
## 2     AR  1765         29 0.0164305949   29.5240793
## 3     AZ   237          0 0.0000000000    2.5063291
## 4     CA   414          0 0.0000000000    0.4251208
## 5     CO  2023          1 0.0004943154    0.5289174
## 6     CT    99          2 0.0202020202   18.2626263
## 7     DC     2          0 0.0000000000    0.0000000
## 8     DE    61          0 0.0000000000    5.7377049
## 9     FL  3211          3 0.0009342884    7.1535347
## 10    GA  1507         11 0.0072992701   20.3769078
## ..   ...   ...        ...          ...          ...
```

Some more examples on chaining

Example 14

```
tornado %>%
  group_by(Year) %>%           # group by year
  summarize(count = n()) %>%   # count the number of tornadoes by year
  arrange(desc(count))         # arrange the counts in descending order
```

get the tornadoes by year arranged by highest counts

```
## Source: local data frame [65 x 2]
##
##     Year count
##    (int) (int)
## 1   2004  1836
## 2   2011  1775
## 3   2008  1737
## 4   1998  1440
## 5   2003  1415
```

```
## 6   1999  1361
## 7   2010  1318
## 8   1992  1312
## 9   2005  1263
## 10  1995  1251
## ..   ...   ...
```

Example 15

```
tornado %>%
  filter(F_scale>=3) %>%            # subset rows where intensity is equal to or higher than 3
  group_by(State) %>%               # group by state
  summarize(avg.loss=mean(Property_loss),n=n()) %>% # calculate average property loss by state and coun
  arrange(desc(avg.loss))           # arrange the average property loss by state
```

calculate the mean property loss for each state with tornado intensity greater or equal to 3 and get the highest loss

```
## Source: local data frame [44 x 3]
##
##    State avg.loss     n
##    (chr)    (dbl) (int)
## 1     AL 38.67708   192
## 2     MA 30.44444     9
## 3     MD 28.22222     9
## 4     MO 25.25926   162
## 5     OK 17.94118   272
## 6     NC 13.15686    51
## 7     FL 12.77500    40
## 8     IL 12.00000   145
## 9     NY 10.36000    25
## 10    TN 10.14286   126
## ..   ...      ...   ...
```

References & Resources

- Hadley Wickham's dplyr tutorial at UseR! 2014, Part 1
- Hadley Wickham's dplyr tutorial at UseR! 2014, Part 2
- Data Wrangling Cheatsheet
- Walk Through Vignette
- Another tutorial by Kevin Markham