Introduction to dplyr

Paul M. Magwene

What is dplyr?

dplyr is a package that provides a “grammar for data manipulation”

Key “verbs” in the dplyr package:

- select()
- filter()
- mutate()
- arrange()
- summarize()
- group_by()

All these functions return new data frames instead of modifying existing data frames

`select()` subsets columns

```
names(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length"
[4] "Petal.Width"  "Species"
```

```
# select two columns
df <- select(iris, Sepal.Length, Petal.Length)
head(df, 3)
  Sepal.Length Petal.Length
1          5.1          1.4
2          4.9          1.4
3          4.7          1.3
```

```
# select everything BUT the species column
df <- select(iris, -Species)
head(df, 3)
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1          5.1         3.5          1.4         0.2
2          4.9         3.0          1.4         0.2
3          4.7         3.2          1.3         0.2
```

select() has some specialized functions for powerful filtering

```
df <- select(iris, starts_with("Petal"))
head(df, 3)
  Petal.Length Petal.Width
1          1.4         0.2
2          1.4         0.2
3          1.3         0.2
```

```
df <- select(iris, ends_with("Length"))
head(df, 3)
  Sepal.Length Petal.Length
1          5.1          1.4
2          4.9          1.4
3          4.7          1.3
```

filter() selects rows that match criteria

```
# get only the I. setosa specimens
df <- filter(iris, Species == "setosa")
head(df, 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
```

```
# filter on mulitple criteria
df <- filter(iris, Species == "setosa", Sepal.Length < 5)
head(df, 3)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          4.9         3.0          1.4         0.2  setosa
2          4.7         3.2          1.3         0.2  setosa
3          4.6         3.1          1.5         0.2  setosa
```

mutate() adds or transforms columns

```
df <- mutate(iris, Species = str_to_upper(Species))
head(df, 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
```

arrange() sorts rows according to values of one or more columns

```
# sort by Sepal.Length
df <- arrange(iris, Sepal.Length)
head(df, 3)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          4.3         3.0          1.1         0.1  setosa
2          4.4         2.9          1.4         0.2  setosa
3          4.4         3.0          1.3         0.2  setosa
```

```
# sort by Sepal.Length then by Petal.Length
df <- arrange(iris, Sepal.Length, Petal.Length)
head(df, 3)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          4.3         3.0          1.1         0.1  setosa
2          4.4         3.0          1.3         0.2  setosa
3          4.4         3.2          1.3         0.2  setosa
```

summarize() transforms and collapses

summarize() applies functions to one or more variables (columns) in the data frame, reducing a vector of values to a single value and returning the results in a data frame

```
mean.Lengths <-
  summarize(iris,
            avg.Sepal.Length = mean(Sepal.Length),
            avg.Petal.Length = mean(Petal.Length))

mean.Lengths
  avg.Sepal.Length avg.Petal.Length
1         5.843333            3.758
```

group_by() is used for conditioning (faceting) and transforming

```
# apply grouping
grouped.df <- group_by(iris, Species)

# summarize grouped data frame
mean.by.group <-
  summarize(grouped.df,
            avg.Sepal.Length = mean(Sepal.Length),
            avg.Petal.Length = mean(Petal.Length))

mean.by.group
# A tibble: 3 x 3
  Species    avg.Sepal.Length avg.Petal.Length
  <fct>                 <dbl>            <dbl>
1 setosa                 5.01             1.46
2 versicolor             5.94             4.26
3 virginica              6.59             5.55
```

The pipe operator, %>%

`dplyr` also provides a new operator called a pipe

- The pipe operator is %>%

Using pipes:

- `x %>% f()` is equivalent to `f(x)`
- `x %>% f(y)` is equivalent to `f(x,y)`.

Examples:

- Single argument function:

```
pi %>% cos()
[1] -1
```

- For single argument functions, after the pipe you can drop the parentheses:

```
pi %>% cos  # same as above
[1] -1
```

- Multi-argument functions

```
100 %>% log(base=10) # 100 is treated as the first argument
[1] 2
```

Building pipelines with the pipe operator

The pipe operator allows us to build analysis “pipelines”.

A pipeline series of function calls that filter and/or transform our data

```
letters %>%          # start with letters vector
  str_to_upper %>%  # convert to upper case
  tail(10) %>%      # get last 10 elements
  str_flatten("-")  # join into single string, separated by '-'
[1] "Q-R-S-T-U-V-W-X-Y-Z"
```

The pipe operator helps to make our intent clearer, as compared to nested function calls:

```
str_flatten(tail(str_to_upper(letters), 10), "-")
[1] "Q-R-S-T-U-V-W-X-Y-Z"
```

The dplyr verbs functions are designed to work well with piping!

```
filtered.Sepal.means <-
  iris %>%
  filter(Species != "virginica") %>%
  group_by(Species) %>%
  summarize(avg.Sepal.Length = mean(Sepal.Length),
            avg.Sepal.Width = mean(Sepal.Width))


filtered.Sepal.means
# A tibble: 2 x 3
  Species    avg.Sepal.Length avg.Sepal.Width
  <fct>                 <dbl>           <dbl>
1 setosa                 5.01            3.43
2 versicolor             5.94            2.77
```