Awakening Data Scientist

Removing the Final Veils of Ignorance

R of the Day: grep() and grepl()

July 20, 2015January 31, 2017 gut3socipsych Computer Programming, Doing Things with R, R of the Day data science, R programming

Anyone who interacts with data sets will inevitably need to filter or select data points, columns, or rows based on a value; for instance, you may need to filter a data set based on an income variable being more than $50,000. Base R provides users with the basic comparison operators (i.e., >, <, ==) for such data manipulations; however, oftentimes you may need to filter a data set based on a partial character string that is beyond the scope of comparison operators.

Base R provides such functions (*grep* and *grepl*) that match character patterns in specified vector. While both of these functions find patterns, they return different output types based on those patterns. Specifically, *grep* returns numeric values that correspond to the indexed locations of the patterns and *grepl* returns a logical vector in which "TRUE" represents a pattern match.

Let's take a look at the basic outputs for both functions using the CO2 data set included in R's data library.

```
> head(CO2)
 Plant Type   Treatment  conc uptake
1 Qn1  Quebec nonchilled 95   16.0
2 Qn1  Quebec nonchilled 175  30.4
3 Qn1  Quebec nonchilled 250  34.8
4 Qn1  Quebec nonchilled 350  37.2
5 Qn1  Quebec nonchilled 500  35.3
6 Qn1  Quebec nonchilled 675  39.2

#grep return the index value of each matched pattern
> grep("non", CO2$Treatment)
 [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63

#grepl returns a logical output for each indices in the original vector
#with "TRUE" representing matched patterns
> grepl("non", CO2$Treatment)
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[11] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[21] TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[31] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[41] FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[51] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[61] TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[71] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[81] FALSE FALSE FALSE FALSE
```

Now, let's apply these functions to practical needs.

Filtering with *grep:*

```
> #position indexing application
> #filter data set based on values in a column
> filter_for_value<-CO2[grep("non", CO2$Treatment), ]
> head(filter_for_value)
  Plant Type   Treatment  conc uptake
1 Qn1   Quebec nonchilled 95   16.0
2 Qn1   Quebec nonchilled 175  30.4
3 Qn1   Quebec nonchilled 250  34.8
4 Qn1   Quebec nonchilled 350  37.2
5 Qn1   Quebec nonchilled 500  35.3
6 Qn1   Quebec nonchilled 675  39.2

> #filter data set based on values that do not match the specified pattern
> filter_for_not_a_value<-CO2[-(grep("non", CO2$Treatment)),]
> head(filter_for_not_a_value)
   Plant Type   Treatment conc uptake
22 Qc1   Quebec chilled   95   14.2
23 Qc1   Quebec chilled   175  24.1
24 Qc1   Quebec chilled   250  30.3
25 Qc1   Quebec chilled   350  34.6
26 Qc1   Quebec chilled   500  32.5
27 Qc1   Quebec chilled   675  35.4
```

Selecting columns with *grep*:

```
> select_columns<-CO2[, grep("T", colnames(CO2))]
> head(select_columns)
  Type   Treatment
1 Quebec nonchilled
2 Quebec nonchilled
3 Quebec nonchilled
4 Quebec nonchilled
5 Quebec nonchilled
6 Quebec nonchilled

> dont_select_columns<-CO2[, -(grep("T", colnames(CO2)))]
> head(dont_select_columns)
  Plant conc uptake
1 Qn1   95   16.0
2 Qn1   175  30.4
3 Qn1   250  34.8
4 Qn1   350  37.2
5 Qn1   500  35.3
6 Qn1   675  39.2
```

The other great feature about *grep* and *grepl* is their adaptation by other packages in R. I am a huge fan and user of the *dplyr* package by Hadley Wickham because it offer a powerful set of easy-to-use "verbs" and syntax to manipulate data sets. However, strong and effective packages such as *dplyr* incorporate base R functions to increase their practicality. This is how I typically use *grep* and *grepl* with *dplyr:*

```
> library(dplyr)
> CO2_dplyr<-tbl_df(CO2) #converting CO2 into a local data frame
>
> #dplyr filtering with grepl
> filter_dplyr_for_value_non<-CO2_dplyr %>% filter(grepl("non", Treatment))
> filter_dplyr_for_value_non
Source: local data frame [42 x 5]
  Plant Type   Treatment  conc uptake
1 Qn1   Quebec nonchilled 95   16.0
2 Qn1   Quebec nonchilled 175  30.4
3 Qn1   Quebec nonchilled 250  34.8
4 Qn1   Quebec nonchilled 350  37.2
5 Qn1   Quebec nonchilled 500  35.3
6 Qn1   Quebec nonchilled 675  39.2
7 Qn1   Quebec nonchilled 1000 39.7
8 Qn2   Quebec nonchilled 95   13.6
9 Qn2   Quebec nonchilled 175  27.3

> filter_dplyr_for_not_a_value<-CO2_dplyr %>% filter(!(grepl("non", Treatment)))
> filter_dplyr_for_not_a_value
Source: local data frame [42 x 5]
  Plant Type   Treatment conc uptake
1 Qc1   Quebec chilled   95   14.2
2 Qc1  Quebec  chilled   175  24.1
3 Qc1  Quebec  chilled   250  30.3
4 Qc1  Quebec  chilled   350  34.6
5 Qc1  Quebec  chilled   500  32.5
6 Qc1  Quebec  chilled   675  35.4
7 Qc1  Quebec  chilled   1000 38.7
8 Qc2  Quebec  chilled   95   9.3
9 Qc2  Quebec  chilled   175  27.3

> #dplyr selecting with grep
> select_dplyr_columns<-CO2_dplyr %>% select(grep("T", colnames(CO2_dplyr)))
> select_dplyr_columns
Source: local data frame [84 x 2]
  Type   Treatment
1 Quebec nonchilled
2 Quebec nonchilled
3 Quebec nonchilled
4 Quebec nonchilled
5 Quebec nonchilled
6 Quebec nonchilled
7 Quebec nonchilled
8 Quebec nonchilled
9 Quebec nonchilled

> dont_select_dplyr_column<-CO2_dplyr %>% select(-grep("T", colnames(CO2_dplyr)))
> dont_select_dplyr_column
Source: local data frame [84 x 3]
  Plant conc uptake
1 Qn1   95   16.0
```

```
2 Qn1   175  30.4
3 Qn1   250  34.8
4 Qn1   350  37.2
5 Qn1   500  35.3
6 Qn1   675  39.2
7 Qn1   1000 39.7
8 Qn2   95   13.6
9 Qn2   175  27.3
```

Some things to know about the *grep* and *grepl* functions:

- Using regular expressions (programming symbol pattern) will increase their functionality
- Specified patterns are case sensitive (“t” does not equal “T”)
- Any matching pattern will be returned despite the context in which that pattern is located (i.e., grep(“the”, data) with return matches for “the“, “theme”, “heather”, “breathe“, and so on–this is where regular expressions are useful for specifying where in a string the pattern should appear.