Tidyr And Functions Exercises

Kim Roth & Johnathan Clementi

3/27/2020

These are exercises based on the exercises in R for Data Science by Wickham and Grolemund.

The website where the book appears is (and will always be) **free to use**, and is licensed under the [Creative Commons Attribution-NonCommercial-NoDerivs 3.0](http://creativecommons.org/licenses/by-nc-nd/3.0/us/) License.

We will use the data set fueleconomy:vehicles

library(tidyverse)

## -- Attaching packages ------------------------------------------------------------------------------------------------------------------------------ tidyverse 1.3.0 --

## v ggplot2 3.2.1 v purrr 0.3.3  
## v tibble 2.1.3 v dplyr 0.8.3  
## v tidyr 1.0.0 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.4.0

## -- Conflicts --------------------------------------------------------------------------------------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(fueleconomy)  
vehicles<-fueleconomy::vehicles

Data for this file.

vehicles

## # A tibble: 33,442 x 12  
## id make model year class trans drive cyl displ fuel hwy cty  
## <int> <chr> <chr> <int> <chr> <chr> <chr> <int> <dbl> <chr> <int> <int>  
## 1 27550 AM Ge~ DJ Po ~ 1984 Specia~ Auto~ 2-Whe~ 4 2.5 Regu~ 17 18  
## 2 28426 AM Ge~ DJ Po ~ 1984 Specia~ Auto~ 2-Whe~ 4 2.5 Regu~ 17 18  
## 3 27549 AM Ge~ FJ8c P~ 1984 Specia~ Auto~ 2-Whe~ 6 4.2 Regu~ 13 13  
## 4 28425 AM Ge~ FJ8c P~ 1984 Specia~ Auto~ 2-Whe~ 6 4.2 Regu~ 13 13  
## 5 1032 AM Ge~ Post O~ 1985 Specia~ Auto~ Rear-~ 4 2.5 Regu~ 17 16  
## 6 1033 AM Ge~ Post O~ 1985 Specia~ Auto~ Rear-~ 6 4.2 Regu~ 13 13  
## 7 3347 ASC I~ GNX 1987 Midsiz~ Auto~ Rear-~ 6 3.8 Prem~ 21 14  
## 8 13309 Acura 2.2CL/~ 1997 Subcom~ Auto~ Front~ 4 2.2 Regu~ 26 20  
## 9 13310 Acura 2.2CL/~ 1997 Subcom~ Manu~ Front~ 4 2.2 Regu~ 28 22  
## 10 13311 Acura 2.2CL/~ 1997 Subcom~ Auto~ Front~ 6 3 Regu~ 26 18  
## # ... with 33,432 more rows

A subset for parts of the homework.

hybridSubset=filter(vehicles, fuel=="Regular Gas and Electricity")  
hybridSubset=select(hybridSubset, make, model, year, hwy)  
hybridSubset

## # A tibble: 8 x 4  
## make model year hwy  
## <chr> <chr> <int> <int>  
## 1 Ford C-MAX Energi Plug-in Hybrid 2014 41  
## 2 Ford C-Max Energi Plug-in Hybrid 2013 41  
## 3 Ford Fusion Energi Plug-in Hybrid 2013 41  
## 4 Ford Fusion Energi Plug-in Hybrid 2014 41  
## 5 Honda Accord Plug-in Hybrid 2014 46  
## 6 Toyota Prius Plug-in Hybrid 2012 49  
## 7 Toyota Prius Plug-in Hybrid 2013 49  
## 8 Toyota Prius Plug-in Hybrid 2014 49

1. Why are gather() and spread() not perfectly symmetrical?  
   Carefully consider the following example:

hybridSubset  
   
hybridSubset %>%   
 spread(year, hwy) %>%   
 gather("year", "hwy", `2012`:`2014`)

1A. Gather and Spread are not totally symmetric if there are instances of non-existant observations for all categories within all variables in the starting data, known as implicit NA. For instance, the original data here does not have an observation for the Ford C-MAX Energi Plug-in in 2012. However, after spreading and then gathering, the observation is created and only has NA for the hwy reading.

1. Make the command from 1 give back the original data. You can use spread and gather or pivot\_wider and pivot\_longer.  
   2A.

hybridSubset %>%   
 spread(year, hwy) %>%   
 gather("year", "hwy", `2012`:`2014`) %>%  
 filter(!is.na(hwy))

## # A tibble: 8 x 4  
## make model year hwy  
## <chr> <chr> <chr> <int>  
## 1 Toyota Prius Plug-in Hybrid 2012 49  
## 2 Ford C-Max Energi Plug-in Hybrid 2013 41  
## 3 Ford Fusion Energi Plug-in Hybrid 2013 41  
## 4 Toyota Prius Plug-in Hybrid 2013 49  
## 5 Ford C-MAX Energi Plug-in Hybrid 2014 41  
## 6 Ford Fusion Energi Plug-in Hybrid 2014 41  
## 7 Honda Accord Plug-in Hybrid 2014 46  
## 8 Toyota Prius Plug-in Hybrid 2014 49

1. Why does spread or pivot\_wider on this tibble fail? Why doesn’t it fail on the full hybridSubset?

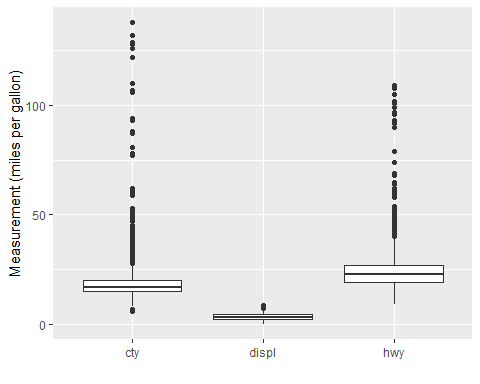
hybridSubset2 = select(hybridSubset, year, make, hwy)  
  
hybridSubset2 %>%  
 spread(year, hwy)

3A. Spread or pivot\_wider fails to spread this tibble because it cannot create a unique key for each new observation as it spreads

1. Make a graph of the average hwy, cty, and displ with the original data set vehicles. Use the example in the videos to help you. 4A.

mpgDispl = select(vehicles, make, model, year, hwy, cty, displ)  
   
mpgDispl = mpgDispl %>%   
 pivot\_longer(c("hwy", "cty", "displ"), names\_to = "mpg", values\_to = "Average")  
  
ggplot(data = mpgDispl) +  
 geom\_boxplot(aes(x = mpg, y = Average)) +  
 labs(x = "", y = "Measurement (miles per gallon)")

## Warning: Removed 57 rows containing non-finite values (stat\_boxplot).



1. Use separate to split the displ variable into parts named integer and decimal using the data set vehicles (Yes this is a silly exercise, but the data set did not have a better option.) 5A.

vehicles = separate(vehicles, displ, into = c("displInt", "displDec"))

## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 8176 rows [10,  
## 13, 16, 42, 44, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145,  
## 147, 149, ...].

1. Use unite to combine the variables make and model into one column called car in the original data set, vehicles. 6A.

vehicles = unite(vehicles, car, make, model)

1. Give an example of an explict NA from vehicles. Give an example of a possible implicit NA from vehicles. 7A. Explicit NA: hwy, cty, and displ measurements for electric cars Implicit NA: observations for cars that were not made during certain years in the study (i.e. car wasn’t created until certian year, or car was discontinued after a certain year)
2. Give an example of a data set (other than the one in the book) where using fill in data set is a good idea. Give an example of a data set where using fill is a bad idea. 8A. A good candidate data set for using fill on is one that has many redundant observations. Using fill on a data set with many unique observations in succession could introduce a lot of bad data.
3. Why is TRUE not a parameter to rescale01() from the reading? What would happen if x contained a single missing value, and na.rm was FALSE? 9A. The function rescales a series of numbers to a scale from 0 to 1. The parameters being passed to the function are numeric, not boolean (which TRUE is). The missing value would be ignored by the function.

rescale01 <- function(x) {  
 rng <- range(x, na.rm = TRUE)  
 (x - rng[1]) / (rng[2] - rng[1])  
}  
rescale01(c(0, NA, 10))

## [1] 0 NA 1

1. Write a function that calculates the product of two numbers. Test your function on some values. 10A.

prdct = function(x, y){  
 x \* y  
}  
  
prdct(4,4)

## [1] 16

prdct(5,5)

## [1] 25

prdct(6,6)

## [1] 36

1. Practice turning the following code snippets into functions. Think about what each function does. What would you call it? How many arguments does it need?

`sum(x>10,na.rm=TRUE)/sum(!is.na(x))`

median(x, na.rm = TRUE)

mean(x)/ sd(x) 11A.

#This function takes a vector and calculates the sum of all numbers greater than 10 divided by the sum of all of the numbers in the vector  
greaterThanTen = function(x){  
 sum(x>10, na.rm = TRUE)/sum(!is.na(x))  
}  
  
# This function calculates the median of a set of numbers (provided in the form of a vector)  
calculateMedian = function(x){  
 median(x, na.rm = TRUE)  
}  
  
# If the coefficient of variation is the standard deviation over the mean, then this function is the inverse coefficient of variation  
inverseCV = function(x){  
 mean(x) / sd(x)  
}

Here is some data to test it on.

test1=c(1,2,3,4)  
test2=c(0,5,10,15,20)  
test3=c(-3,0,3,6,9,12,15,18,21)  
test4=c(1,0,100, NA, -1000)  
  
greaterThanTen(test1)

## [1] 0

greaterThanTen(test2)

## [1] 0.4

calculateMedian(test1)

## [1] 2.5

calculateMedian(test2)

## [1] 10

inverseCV(test1)

## [1] 1.936492

inverseCV(test2)

## [1] 1.264911

1. Write a function that takes in a vector and outputs the percentage of the vector that is NA. 12A.

pctNA = function(x, count = 0){  
 for(y in x) {  
 if(is.na(y)) {  
 count = count + 1  
 } else {}  
 }  
 count/(length(x))  
}

pctNA(test1)

## [1] 0

1. Write same\_mean(), a function that takes two vectors and returns if they have the same mean or not. 13A.

same\_mean = function(x, y){  
 if(mean(x) == mean(y)) {  
 return(TRUE)  
 } else {  
 return(FALSE)  
 }  
}

same\_mean(test1, test2)

## [1] FALSE

same\_mean(test1, test1)

## [1] TRUE

1. What do the following functions do? 14A. path = the working directory of the current R script combine = concatenate x and y strings without space between them

```r  
path <- function() getwd()  
combine <- function(x,y) paste0(x, y)  
```

1. Read the source code for each of the following three functions, puzzle out what they do, and then brainstorm better names.

* f1 <- function(string, prefix) {  
   substr(string, 1, nchar(prefix)) == prefix  
  }  
  f2 <- function(x) {  
   if (length(x) <= 1) return(NULL)  
   x[-length(x)]  
  }  
  f3 <- function(x, y) {  
   rep(y, length.out = length(x))  
  }
* 15A. f1 determines if string has a prefix at the start of the character vector. It could be renamed testForPrefix

f2 removes the last element of a vector. It could be renamed rmvLstElement

f3 replicates or truncates the second vector (y) so that it is the same length as the first vector (x). It could be renamed lengthAdjustor