Eighth R Practice exercise composing a function and constructing a data frame; files not containing search strings

Alan E. Berger Feb 2, 2020

available at https://github.com/AlanBerger/Practice-programming-exercises-for-R

Introduction

This is the eighth in a sequence of programming exercises in "composing" an R function to carry out a particular task. Several of these "exercise files" likely
will take several sessions to master the content. The material below practices composing a logical sequences of steps to program a function that will accomplish a specified task, and preparing a corresponding data frame.

The idea of this set of exercises is to practice correct use of R constructs and built in functions (functions that "come with" the basic R installation), while learning how to "put together" a correct sequence of blocks of commands that will obtain the desired result.
Note these exercises are quite cumulative - one should do them in order.

In these exercises, there will be a statement of what your function should do (what are the input variables and what the function should return) and a sequence of "hints". To get the most out of these exercises, try to write your function using as few hints as possible.
Note there are often several ways to write a function that will obtain the correct result. For these exercises the directions and hints may point toward a particular approach intended to practice particular constructs in R and a particular line of reasoning, even if there is a more efficent way to obtain the same result.
There may also be an existing R function or package that will do what is stated for a given practice exercise, but here the point is to practice formulating a logical sequence of steps, with each step a section of code, to obtain a working function, not to find an existing solution or a quick solution using a more powerful R construct that is better addressed later on.

Motivation for this exercise

We will compose a function that returns a vector of the names of files in a folder that do NOT contain any of the entries of search.strings in their name. This will be an opportunity to practice use of the **%in%** function; and then use of the **setdiff** function applied to two vectors: `setdiff(V1, V2)` for vectors V1 and V2 of the same type (e.g., numeric or character) will give a vector consisting of the entries of V1 that are **not** equal to any entry of V2.

In the previous (seventh) exercise file we prepared the function

search_for_filenames_containing_any_of_the_patterns_and_output_file_info(directory, search.strings)

that returns a data frame containing information on the file names in directory that match **ANY entry** of search.strings (i.e., that have 1 or more of the entries of search.strings in their file name). The first column of the returned data frame has the names of the files. The final version is copied here:

```
search_for_filenames_matching_any_of_the_patterns_and_output_file_info <-
                                        function(directory, search.strings){

# directory is an absolute path (full path) or a path relative to the R working directory to the
# folder to be searched. If want to search the R working directory itself,
# can set directory = "." with the line of code:   directory <- "."
# or could set directory to be the full path to the R working directory.
```

```r
# search.strings is a character string vector

# Return a data frame of the file names (not including folders) in directory that contain
# ANY of the entries of the search.strings vector somewhere in their file name,
# (or at the beginning of the filename, or at the end of the filename, if so specified).
# The search will be case insensitive (treats lower case and upper case letters as the same).

# The first step is to initialize the filenames vector: filenames <- character(0)

# In a for loop, use R's list.files function to list all the files (file names) matching the
# entries of search.strings, one by one,
# eliminating names of folders, and appending them to filenames

# Use the unique function to eliminate duplicates (keep only 1 copy of each file name)

# For the files whose names contain any of the character strings in search.strings, use
# R's file.info function to get the file size and last modification time as in the previous function.

# The output data frame will contain the file size and the last modification time
# for each file that has any member of search.strings somewhere in its file name
# (or at the beginning or at the end of the file name, if so specified)
#
# We will get the file names without the folder path leading to the files included in the name.

# check that search.strings is a non-empty character vector

ns <- length(search.strings)
if(ns < 1) stop("no entries in search.strings")
if(!is.character(search.strings)) stop("search.strings is not a character vector")

# We will return a data frame whose first column contains the files in directory
# that contain any character string in the search.strings vector somewhere in their name.
# The second column will contain the last time (and date) the file was modified,
# and the third column will be the file size (in bytes).

# The first step is to initialize the filenames vector
filenames <- character(0)

# Then in a for loop, for each entry S of search strings,
# use list.files to get the vector V the file names in directory that
# contain S in their file name and append V to filenames (after eliminating any folder names).
# do not include the path to the file in the file name.

# To eliminate names of any folders (directories) that are in V:
# do paste(directory, "/", V, sep = "") to get the filenames including either
# the relative path from the R working directory or the absolute path (depending on what
# directory is); use these names in the R dir.exists function to check for folder
# (directory) names in filenames

   for (k in 1:ns) {
      V <- list.files(directory, pattern = search.strings[k],
                      full.names = FALSE, ignore.case = TRUE)
#     exclude directory names from V (we need to do this since we are "adding"
```

```
#     the file names in V to filenames and want only file names, not folder names
#     if V is empty (character(0)) then skip this
      if(length(V) > 0) V <- V[!dir.exists(paste(directory, "/", V, sep = ""))]
      filenames <- c(filenames, V)
   }

### It is important to note we could have also "run the for loop" in this fashion:
### for (S in search.strings) {
###    V <- list.files(directory, pattern = S,
##############   rest of the for loop
###

nf <- length(filenames)
if(nf == 0) {
   print("no files contain any of the search strings")
   return("no files contain any of the search strings")
}

filenames <- unique(filenames)
nf <- length(filenames)  # need to do this since may have eliminated some duplicate name(s)

# If got to here, at least 1 file has a character string in search.strings
# in its name, so get the information on these file(s) into a data frame.

################################### get the data frame to be output

# Get the desired output data frame using vectors

dfcolnames <- c("file.name", "modif.date", "size.in.bytes")
# initialize the 3 vectors that will hold this information on the files
# whose names matched any of the members of search.strings

fname <- character(0)
fdate <- character(0)
fsize <- numeric(0)

for(k in 1:nf) {
    finfo <- file.info(paste(directory, "/", filenames[k], sep = ""))
# needed to include the path to the file so file.info can locate it
    fname <- c(fname, filenames[k])
    fdate <- c(fdate, as.character(finfo$mtime))
    fsize <- c(fsize, finfo$size)
}

df <- data.frame(fname, fdate, fsize, stringsAsFactors = FALSE)
colnames(df) <- dfcolnames

################################### finished getting the data frame to be output

# Write the data frame out to a tab delimited text file called scrlisting.txt in directory
# (i.e., in the folder specified by the argument directory this function was called with).

outpfilename <- paste(directory, "/", "scrlisting.txt", sep = "")
```

```
# One can rename this "scratch file" as desired after viewing it (best viewed in Excel or equivalent).
write.table(df, file = outpfilename,
            append = FALSE, quote = FALSE, sep = "\t",
            row.names = FALSE, col.names = TRUE)
# This call to write.table will write out a data frame
# as one would usually want; it specifies the column separator to be a tab

return(df)
}
```

Exercises

The function for this exercise will be to construct a modified version of the search function called

```
search_for_filenames_matching_NONE_of_the_patterns_and_output_file_names <-
                                          function(directory, search.strings)
```

that will return the vector of file names in directory that do **NOT** match **ANY entry** of search.strings

One way to view this, is that the first function in this sequence obtained the **intersection** of the sets of file names matching each entry of search strings, while the previous function (the function above) obtained the **union** U of the sets of file names matching each entry of search.strings This programming exercise will be to construct a version that will return the vector of file names in directory that **DO NOT match ANY** entry of search.strings. This can be viewed as obtaining, relative to the set of all the files in directory, the **complement of U**.

One could as in the previous functions return information on these files, but this is more a practice exercise in using the %in% function (and in the next exercise, using the setdiff function) so we are just going to concentrate on getting the vector of the file names.

Hints: Use the

search_for_filenames_matching_any_of_the_patterns_and_output_file_info

function to find the file names in directory that match one or more of the file names in search.strings, call this vector **Vany**. Then remove these file names from the vector of all the file names in directory. One approach for this is to use **list.files** to get the vector of all the file and folder names in directory, then remove the folder names from this vector, resulting in the vector **Vall**. Then construct a logical vector Vlogical the same length as Vall such that: Vlogical[k] is TRUE if Vall[k] is **not** an entry of Vany (and Vlogical[k] is FALSE if Vall[k] is an entry of Vany). The **%in%** function is very convenient for this (and this is good practice using it).

Recall from the fifth article in this series: The **%in%** function addresses the question of whether or not each entry of some vector v occurs in another vector w: `z <- v %in% w` obtains a logical vector z with z[k] being TRUE if v[k] is equal to some entry in w, and z[k] being FALSE if v[k] is not equal some entry in w. Note then the logical vector `nz <- !z` has the property that nz[k] will be TRUE if v[k] is NOT an entry of w, and nz[k] will be FALSE if v[k] is an entry of w; hence v[nz] gives the entries of v that are NOT in w.

An example of how %in% behaves:

```
?"%in%"  # look at the help on %in% Note because of the special
#            character % one needs to "protect" %in% by enclosing it in
#            either quotes or apostrophes when "asking for help" on it

v <- c(1, 2, 3, 4, 5, NA)
```

```
w <- c(12, 3, 8, 22, 4)
v %in% w
[1] FALSE FALSE  TRUE  TRUE FALSE FALSE

v <- c(1, 2, 3, 4, 5, NA)
w <- c(12, 3, 8, 22, 4, NA)
v %in% w
[1] FALSE FALSE  TRUE  TRUE FALSE  TRUE

# note %in% will declare a match for an NA in v if there is an NA in w
```

From the discussion above, Vall[Vlogical] is the desired vector of file names, and we can obtain Vlogical using the `%in%` function and logical negation (!). Try doing this - a working version is given below.

```
search_for_filenames_matching_NONE_of_the_patterns_and_output_file_names <-
                                        function(directory, search.strings){

# directory is an absolute path (full path) or a path relative to the R working directory to the
# folder to be searched. If want to search the R working directory itself,
# can set directory = "."
# or could set directory to be the full path to the R working directory.

# search.strings is a text string vector

# Return a vector of the file names (not including folders) in directory that contain
# NONE of the entries of the search.strings vector in their file name,
# the search will be case insensitive (treats lower case and upper case letters as the same).
# Since this is mainly an exercise in using the %in% function, just return the vector of file
# names (and don't bother with getting the last modification data or file size).

# Use R's list.files function to list all the files (file names) in directory
# then eliminate names of folders; put the result in Vall

# Then use the previous function
# search_for_filenames_matching_any_of_the_patterns_and_output_file_info
# to find all the files in directory that match some entry in search.strings
# and put the vector of these file names in Vany

# Then eliminate from Vall any entries that are in Vany using the %in% function
# and logical negation (!)

# We will get the file names without the folder path leading to the files included in the name.

# check that search.strings is a non-empty character vector

ns <- length(search.strings)
if(ns < 1) stop("no entries in search.strings")
if(!is.character(search.strings)) stop("search.strings is not a character vector")

Vall <- list.files(directory, full.names = FALSE, ignore.case = TRUE)
nf <- length(Vall)
if(nf == 0) {
   print("no files in directory")
   return("no files in directory")
```

```
}
#  exclude directory names from Vall
Vall <- Vall[!dir.exists(paste(directory, "/", Vall, sep = ""))]

nf <- length(Vall)
if(nf == 0) {
   print("no files in directory")
   return("no files in directory")
}

# now get Vany
df.Vany <-
search_for_filenames_matching_any_of_the_patterns_and_output_file_info(directory, search.strings)

# need to handle the case that NO file names were a match for any of the search strings
# in which case df.Vany is the character string: "no files contain any of the search strings"
# rather than a data frame
if(class(df.Vany) == "character") {
   Vany <- character(0)
   } else {
   Vany <- df.Vany$file.name  # the column of file names
}
# When programming, one should always be sure the function handles "extreme cases",
# here that would be when NONE the file names in directory match some entry in search.strings,
# and when ALL of the file names match search.strings

# eliminate names in Vany from Vall

Vlogical <- !(Vall %in% Vany)  # we want entries in Vall that are NOT in Vany so need to use !
filenames <- Vall[Vlogical]  # the desired file names (names NOT in Vany)

nf <- length(filenames)
if(nf == 0) {
   print("all files contain contain an entry of search strings")
   return("all files contain contain an entry of search strings")
}

# If got to here, at least 1 file has no entry of search.strings in its name
return(filenames)  # here we are just returning the file names without other information on them
}
```

Test runs on my computer

Recalling from the previous two exercise files, I have constructed in my R working directory a folder called test_dir containing a small number of files with names I picked to conveniently test the functions that search for file names satisfying various conditions. Here are all the file names (and the one folder name) in the test_dir folder:

```
directory <- "test_dir"
list.files(directory)  # 9 files and 1 folder
 [1] "001.csv"         "002.csv"         "003.txt"
 [4] "004csvfile.txt"  "005txt2csv"      "1.csv"
 [7] "308.csv"         "folder001txtcsv" "scrlisting.txt"
```

```
[10] "txt.csv"
```

We will use this folder and files (9 filenames and 1 folder name) to test the search function written immediately above (you can construct a similar test_dir folder and files with these filenames and also a folder in it called "folder001txtcsv" to run tests, the only difference will be the dates and sizes of the files). Or you could test the function using a suitable folder in your computer for which you can pick search.strings so that you will get a known reasonable length vector for the test run.

Some test runs

```
directory <- "test_dir"
list.files(directory)  # 9 files and 1 folder
 [1] "001.csv"         "002.csv"         "003.txt"
 [4] "004csvfile.txt"  "005txt2csv"      "1.csv"
 [7] "308.csv"         "folder001txtcsv" "scrlisting.txt"
[10] "txt.csv"

search.strings <- c("2", "3", "4", "5")
search_for_filenames_matching_NONE_of_the_patterns_and_output_file_names(directory, search.strings)
[1] "001.csv"        "1.csv"          "scrlisting.txt"    "txt.csv"

search.strings <- c("csv")
search_for_filenames_matching_NONE_of_the_patterns_and_output_file_names(directory, search.strings)
[1] "003.txt"        "scrlisting.txt"

# test case when all the file names match some member of search.strings
search.strings <- c("csv", "txt")
search_for_filenames_matching_NONE_of_the_patterns_and_output_file_names(directory, search.strings)
[1] "all files contain contain an entry of search strings"
[1] "all files contain contain an entry of search strings"
# when there are no such files (no files in directory that don't have an entry of search.strings
# in their file name), the search function both prints this message and returns it, so here it is
# output twice

search.strings <- c("00")
search_for_filenames_matching_NONE_of_the_patterns_and_output_file_names(directory, search.strings)
[1] "1.csv"          "308.csv"        "scrlisting.txt"      "txt.csv"

# test case when none of the file names match some member of search.strings,
# one should get all the file names (but not the folder name)
search.strings <- c("00987")
search_for_filenames_matching_NONE_of_the_patterns_and_output_file_names(directory, search.strings)
[1] "no files contain any of the search strings"
[1] "001.csv"        "002.csv"        "003.txt"        "004csvfile.txt"
[5] "005txt2csv"     "1.csv"          "308.csv"        "scrlisting.txt"
[9] "txt.csv"

# note  [1] "no files contain any of the search strings" came from
# "looking for" the files that matched ANY entry of search.strings -
# in this case there were none, and so one obtained all the file names
# in test_dir (and, correctly, not the folder name)
```

The next exercise is to use the **setdiff** function in place of using the %in% function to get the file names that are in Vall but not in Vany.

Hint: In the function above you simply need to replace the following 2 lines with 1 line using setdiff

```
Vlogical <- !(Vall %in% Vany)  # we want entries in Vall that are NOT in Vany so need to use !
filenames <- Vall[Vlogical]  # the desired file names (names NOT in Vany)
```

A correct line using setdiff is: `filenames <- setdiff(Vall, Vany)`

One should do the test runs again to check this works correctly.

This exercise was intended to practice the very useful **%in%** function, and also illustrate that finding an existing R function (here setdiff) that can do exactly what you want can result in very concise easy to understand code, which leaves less room for bugs to occur and to hide.

Hope this was informative and good practice. The next set of exercises will address dealing with using subsets of an individual row of a data frame. =