VLOOKUP-style using dplyr

Fiona MacNeill

28/06/2019

Learning Technologies Scenario:

You have been sent a text file, or you have copy and pasted from a Microsoft Word document into a new text file. The text file contains a list of names and you need to match these names up with usernames so that you can actually do something useful with this information. You also have a Excel spreadsheet which you have exported from the VLE with a longer list of names, including the username column. You need to match the names in your text file to the names in the Excel and return only the relevant names and the username column. You could use VLOOKUP in Microsoft Excel, but you decide to live a little and use RStudio and some packages instead. Plus VLOOKUP is ever-so-fussy, can we do better?

About this tutorial

The data used in this tutorial was simulated and fabricated from the imagination of Fiona MacNeill on 28 June 2019.

I made use of the following reference materials when creating this tutorial and I owe the authors my gratitude as always: About dplyr: https://dplyr.tidyverse.org/ The ever-useful dplyr cheatsheet: https://github.com/rstudio/cheatsheets/blob/master/data-transformation.pdf About 'join': https://dplyr.tidyverse.org/reference/join.html

Getting started

If you are new to using RStudio you will need to get setup first by...

1. Installing R from the R Archive:

i. Pick a mirror close-by (geographically speaking), e.g in UK – University of Bristol, Imperial College London
ii. Mac tip: make sure that you check the MD5 hash and SHA hash match. You can do this quickly and easily in terminal. As shown in this video: https://youtu.be/HHdrIlHS2-4
iii. [Mac only] XQuartz Install – information about this is provided at R Archive above.

2. Install RStudio: again do check the MD5. You can get the installer here – https://www.rstudio.com/products/rstudio/download/#download

1. Install the packages that you need

You need two packages for this tutorial: dplyr which is part of Hadley Wickham's Tidyverse suite of packages. This is for data transformation and wrangling. The second package is to read a native Excel document which you have downloaded from the VLE (well not really, everything is fabricated).

2. Load the libraries for the packages that you have installed

Now that you have installed the packages that you need, you need to load the libraries so that they are available in your R environment.

```
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
```

```
library("readxl")
```

3. Import your text-file and get it into a useful format

If you copy the complete directory, *'dplyr_vlookup_tutorial'* to your computer and open the RMarkdown file *'dplyr_vlookup.Rmd'* from within it, you should not need to change the file path information in steps 3 and 4.

We want to import our text file first and then format it as a type of data frame specific to the tidyverse called a tibble. I like to think of a data frame as the fundamental building block of anything you do in R. It essentially a way to ratify the relationship between your variables. So in my text file, I know that I have two variables, a first name and last name. The software doesn't know that yet as I have not described these variables by giving them a name, but if I were to say copy and paste these names into an Excel spreadsheet and then give the columns titles, like 'first' and 'last' then I would be doing the same thing as what I am about to do... What I am about to do is take the data from my text file and format it in a fresh table and tell R what to call it.

For more on this, see data.frame at RDocumentation and also About tibbles as part of the Tidyverse

```
pplnames <- read.delim("dplyr_vlookup_files/names.txt", header=FALSE, sep = "")
# This is similar to when you import unformatted data into Excel and have to specify a
# delimiter (e.g. space, comma etc.). The delimeter or "sep" here is white space, as in
# one or more spaces, tabs, newlines or carriage returns. Much more forgiving than Excel!
# For more information, type 'read.delim' into your 'help' tab; it is part of read.table
# in the core R Utils package.

pplnames
```

```
##          V1         V2   V3
## 1      Andy    Neptune
## 2      Alan        Sun
## 3     Anton      James Moon
## 4      Amal      Hades
## 5      Alex       Hera
## 6     Aidan    Mercury
## 7    Arthur   Ganymede
## 8  Antigone   Callisto
## 9     Barry     Thrace
## 10    Bevin    Proxima
## 11   Barney     Wolfen
## 12   Barron  Tau-Alpha
## 13   Caesar        Sol
## 14    Cairn       Rock
## 15   Bessie     Torino
```

```
## 16   Bethan    Palermo
## 17    Betty   Toutalis
## 18    Alvin      Ceres
## 19     Adam Iris-Pupil
## 20    Anita       Hebe
## 21     Baez       Nabu
```

```
# We call the data to see what it looks like so we know how to name our variables on
# the next line.
```

```
people <- tibble(first = (paste(pplnames$V1)), last = (paste(pplnames$V2, pplnames$V3)))
# We are creating a tibble, a type of frame or table for our date and we are saying what
# the columns are...not that one our names have two surnames, so we are pasting those
# together into a single column.

people
```

```
## # A tibble: 21 x 2
##    first    last
##    <chr>    <chr>
##  1 Andy     "Neptune "
##  2 Alan     "Sun "
##  3 Anton    James Moon
##  4 Amal     "Hades "
##  5 Alex     "Hera "
##  6 Aidan    "Mercury "
##  7 Arthur   "Ganymede "
##  8 Antigone "Callisto "
##  9 Barry    "Thrace "
## 10 Bevin    "Proxima "
## # ... with 11 more rows
```

```
# Lets have another look by calling it again - cool that worked so now we have 21 lines
# of data, same as our original text file, but actually labelled.
```

4. Now we are going to bring in our Excel file

In our imaginary land, this file has been exported from the VLE and includes a much longer list of names and also includes a username column. We want this data so that we can find out usernames for our 21 names.

```
ppl_user <- read_excel("dplyr_vlookup_files/un_list.xlsx")
# Loading in our Excel file data.

names(ppl_user)
```

```
## [1] "First Name" "Last Name"  "Username"
```

```
# This command allows us to find out the column names in our new data.
# Hmmm they are different from what we named the columns from our text file.
# These need to match. We could change the text file, but it is best to avoid
# spaces in column headers, so lets simplify it.
```

5. Lets match those column names and simplify it

So having realised that the column names/headers from our Excel file are not optimal (they have spaces, capital letters and don't match our existing 'people' data) we also need to get our data into a tibble anyway so that it matches our existing data format-wise. So lets create a new tibble for our Excel data.

```
ppl_user <- tibble(first = (paste(ppl_user$'First Name')),
                   last = (paste(ppl_user$'Last Name')),
                   username = (paste(ppl_user$'Username')))
# creating our column names to match and then pasting in the data from the relevant
# columns from the imported Excel data.

ppl_user
```

```
## # A tibble: 50 x 3
##    first       last    username
##    <chr>       <chr>   <chr>
##  1 Alice       Apollo  aap11
##  2 Agatha      Pluto   apl5
##  3 Andy        Neptune ane15
##  4 Alan        Sun     asu24
##  5 Anton James Moon    amo23
##  6 Amal        Hades   aha6
##  7 Alex        Hera    ahe12
##  8 Aidan       Mercury ame20
##  9 Angharad    Jupiter aju1
## 10 Amanda      Mars    ama26
## # ... with 40 more rows
```

```
# Ahh that looks like what we need.
```

6. Now for the magic - lets join together these data sets

Now for the really exciting bit, we are joining the two sets of data and only returning what we ACTUALLY want. We are creating a new name for this data 'combined' as it is good to keep our previous named data sets just to check that everything has worked as we hoped.

```
combined <- semi_join(ppl_user, people,  by = "first")
# My new combined data set. 'Semi_join' is one of joins and it is an incredibly versatile
# command, especially helpful for matching huge datasets. For more info type 'join'
# into your help tab.

combined %>% arrange(first)
```

```
## # A tibble: 21 x 3
##    first    last       username
##    <chr>    <chr>      <chr>
##  1 Adam     Iris-Pupil air16
##  2 Aidan    Mercury    ame20
##  3 Alan     Sun        asu24
##  4 Alex     Hera       ahe12
##  5 Alvin    Eunomia    aeu16
##  6 Alvin    Ceres      ace1
##  7 Amal     Hades      aha6
##  8 Andy     Neptune    ane15
```

```
##  9 Anita    Hebe       ahe16
## 10 Antigone Callisto   aca20
## # ... with 11 more rows
```

```
# In case you are wondering, this '%>%' is a pipe and it is specific to the dplyr package
# it is a way of piping data through commands (or at least that is how I think of it)
# for example as outlined on the dplyr cheatsheet:

# (https://github.com/rstudio/cheatsheets/blob/master/data-transformation.pdf),
# x %>% f(y) becomes f(x, y). In this example I want to order my data by the
# first name in alphabetical order.
```

7. Now you can export your data as a spreadsheet

As you are probably going to need to use this data in .csv format for most VLEs, ePortfolios and other eLearning systems this is the logical conclusion of your data wrangling. Good work! The more that you do this, the quicker it gets and it MUCH more forgiving that VLOOKUP in Excel. Plus you don't get the issue where Excel converts username type data into dates: e.g. dec13 becomes 01 December 2013 OR Dec-13.

```
write.csv(combined, file = "combined.csv")
```