**Clinical Trial Project**

[**What we need to know about the agrep function in R**](https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/agrep)

* The agrep function searches for approximate matches to a “pattern” within another element or string.
* The function uses the generalized Levenshtein edit distance to determine matches.
  + The Levenshtein edit distance is the minimal possibly weighted number to **insertions**, **deletions**, and **substitutions** needed to transform one string into another.
  + For this project, I did not define any “costs”. This means that the maximal number/fraction of all the transformation defaults to 10%. This can be changed if wanted.
    - Transformation = insertions, deletions, and substitutions

**Loading the data into R**

Logo

Description automatically generated

The code shown above loads the data into R. This is the only thing that should be changed in the R script. Convert your data into a csv file if necessary. Also, make sure your script is saved in the same place as the data set or the working directory is set to where the data set is saved. Otherwise, your CSV file will not be found.

*If the file you use has the same format and column names as the one in the template, you can run the script using different datasets to get results.* I pulled the data from the [clinical trials](https://www.clinicaltrials.gov/ct2/home) website. Therefore, if you pull the data of the different companies of interest from the website, it should be in the same format.

**Data cleanup**

For some of the trials, there are multiple conditions, sponsors/collaborators, and locations divided by a | or ;. I separate these out into a new row with all the same data from each of the columns. Also, I omit all the NA’s and empty rows and replace all the hyphens (-) with a space.

**Walking through the for loop using the agrepl function**

A vector is created with all the unique names in the dataset. The for-loop takes each of the unique names and looks through each row of the dataset to see if that unique name “matches” it. The agrepl function, using the 10% detection that I mentioned before, will mark each match as TRUE or FALSE. If the match is marked TRUE, it is a match, and the name in the dataset gets replaced with the name in the vector.

For example:

Table

Description automatically generatedText

Description automatically generated with medium confidence

In doing so, each of the names will be standardized and labeled in the same way. That way, the names can be grouped and counted. This name replacing is done only to the copy of the dataset, and we will pull all the original data based on their NCT numbers.

**Walking through the for loop using the grepl function**

The code runs through the same for-loop above again using the grepl function. The grepl() method takes a pattern and returns TRUE if the string contains that pattern.

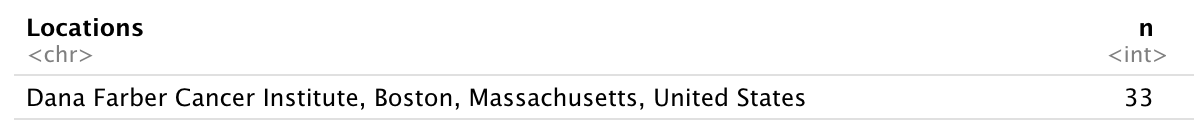
So, in our case, the pattern is each row value, and the string is the vector values. For each vector value, if the string in the row is found in the string in the vector, it will be marked as TRUE. If the match is marked TRUE, the name in the dataset gets replaced with the name in the vector.

For example:

Graphical user interface, application

Description automatically generated with medium confidence

Here we can see that these two sponsor names are the same; the first simply has the location attached to it. The grepl function sees that the words “Dana Farber Cancer Institute” is in “Dana Farber Cancer Institute, Boston, Massachusetts, United States”, and will mark these matches as TRUE. After calling the for-loop, our new results look like this:



Note that the string in the row must be exact to what it matches in the vector. For example, if the row value was “Dana-Farber Cancer Institute”, the match would be marked as FALSE because of the hyphen.

**Finding the top values**

Text

Description automatically generated

To find the top values, group the values in the column you want to count and count each of the values. Then arrange the values by desc(n) to show the top counted values. This can be done using the dyplr package in R.

**Create a new dataset with values that are greater than 5**

Once we group the column we are looking for, we can see which appear 5 or more time. The NCT numbers associated with all the trials are saved into a new vector. Using this vector, the trials from the original dataset are saved into a new dataset. This can then be exported into an excel file.

**Exporting tables to excel**

****

Using the code above, you can export any tables you create in R into an excel file. In this example, top\_conditions\_data is the table that is created and saved in R. “C:\\” is the pathway you want to save the excel file in.

**Exporting R Markdown to HTML**

**Graphical user interface, application

Description automatically generated with medium confidence**

At the top of the R Markdown, there is a knit button. Click the dropdown arrow by the Knit and choose Knit to HTML. This will export the R Markdown and all the outputs to an HTML. It should save in your document folder in finder.