The code

First we grab a list of English words. There is a handy text file available on GitHub for this purpose. It has almost half a million words. We also make a vector of all twenty amino acids. I checked that the spike protein actually has at least one of every amino acid, and it does. We then make a 20 x 20 matrix to hold the count of words in the dictionary that match all 380 possible mutations. Why 380? Well, there are 400 combinations but 20 of those are not mutations, A to A, C to C and so on.

I use grep to find matches, iterate using two for-loops and fill out the matrix with the matches. Feel free to tell me a more efficient way to do this. I used ggplot to show the results because it is complicated to show the colourscale in base R.

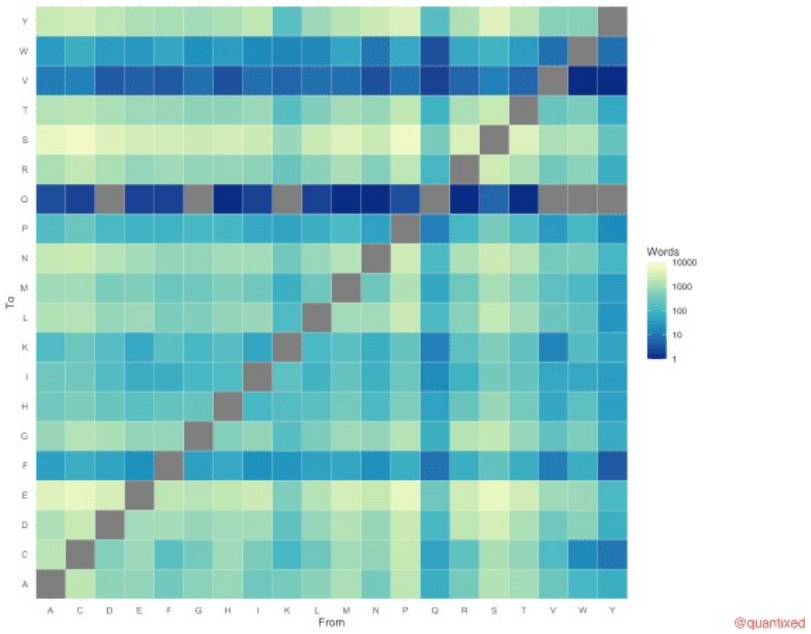
```
library(ggplot2)
library(reshape2)
# character vector of 20 amino acids
aaList <- c("A","C","D","E","F","G","H","I","K","L","M","N","P","Q","
R","S","T","V","W","Y")
# character vector of English words
allWords <- readLines("https://github.com/dwyl/english-words/raw/master/
words_alpha.txt", warn = FALSE)
# empty matrix to hold results
mutMatrix <- matrix(nrow = length(aaList), ncol = length(aaList))

for(i in 1:length(aaList)) {
  for(j in 1:length(aaList)) {
    if(i == j) {
      # no mutation is set as 0
      mutMatrix[i,j] = 0
    } else {
      # make string for grepping
      grepStr <- paste0("^",aaList[i],".*",aaList[j],"$")
      # list of words that match the grep expression
      subList <- grep(grepStr, allWords, value = TRUE, ignore.case =
TRUE)
      # store number of words in matrix
      mutMatrix[i,j] = length(subList)
    }
  }
}
# log transform for easier visualisation
mutMatrixLog <- log10(mutMatrix)

# ggplot version
mutDF <- melt(mutMatrixLog)
mutDF$from <- aaList[mutDF$Var1]
mutDF$to <- aaList[mutDF$Var2]
ggplot(data = mutDF, aes(x = from, y = to, fill = value)) +
  geom_tile(colour = "white") +
  scale_fill_distiller(palette = "YlGnBu", limits = c(0,4),
```

```
                      breaks=c(0,1,2,3,4),
                      labels=c("1","10","100","1000", "10000")) +
  labs(x = "From", y = "To", fill = "Words") +
  theme_minimal(base_size = 10) +
  coord_equal()
```

The result



Heatmap of word frequency

Almost all the mutations have some words in the dictionary to describe them. For example the N501Y “Nelly” mutation could easily be **Nabobery** or **Nuttily**. There are 2407 matches for the “from N to Y” combination.

The grey tiles show where there are no matches at all. So D to Q, G to Q etc. have no matches and I guess that not surprising. There are only 29 words in the dictionary that end in Q. Many of those are abbreviations that would receive a stern frown in a Scrabble match.

```
subList <- grep("Q$", allWords, value = TRUE, ignore.case = TRUE)
length(subList)
[1] 30
subList
 [1] "aeq"      "antiq"    "aq"       "colloq"   "cq"       "eq"
"esq"      "faq"      "freq"     "hq"       "iq"
[12] "iraq"     "liq"      "loq"      "meq"      "nastaliq" "pdq"
"pontacq"  "pq"       "q"        "req"      "seq"
[23] "seqq"     "shoq"     "sq"       "sqq"      "suq"      "tareq"
"umiaq"    "zindiq"
```

The heat map shows that mutations that lead to valine (V), tryptophan (W) or phenylalanine (F) are, like Glutamine (Q), lower frequency. The resulting residue is a tighter constraint than the starting letter/amino acid. Here, the lowest frequencies were Q and Y.

The top three most bountiful mutations are: C to S, P to S and S to E with 7149, 6576 and 5422 words respectively.

Conclusion

Although this naming strategy is fun and memorable, the answer is No: it cannot be used as a general naming system for mutations.