Imagine you have a function that only takes one argument, but you would really like to work on a vector of values. A short example on how function Vectorize() can accomplish this. Let’s say we have a data.frame

xy <- data.frame(sample = c("C\_pre\_sample1", "C\_post\_sample1", "T\_pre\_sample2",

"T\_post\_sample2", "NA\_pre\_sample1"),

value = runif(5))

# sample value

# 1 C\_pre\_sample1 0.3048032

# 2 C\_post\_sample1 0.3487163

# 3 T\_pre\_sample2 0.3359707

# 4 T\_post\_sample2 0.6698358

# 5 NA\_pre\_sample1 0.9490707

and you want to subset only samples that start with C\_pre or T\_pre. Of course you can construct a nice regular expression, implement an anonymouse function using lapply/sapply or use one of those fancy tidyverse functions.

A long winded way would be to find matches using regular expression for each level, combine them and subset. This is for pedagogical reasons, so please bare with me.

i.ind <- do.call(cbind, list(

grepl(pattern = "^C\_pre", x = xy$sample),

grepl(pattern = "^T\_pre", x = xy$sample)

))

i.ind

# [,1] [,2]

# [1,] TRUE FALSE

# [2,] FALSE FALSE

# [3,] FALSE TRUE

# [4,] FALSE FALSE

# [5,] FALSE FALSE

# Find those rows in `xy` that have at least one TRUE and use that to subset the

# data.frame.

xy[rowSums(i.ind) > 0, ]

# sample value

# 1 C\_pre\_sample1 0.3048032

# 3 T\_pre\_sample2 0.3359707

The same can be achieved using a *vectorized* version of the grepl function. We designate which argument exactly is being vectorized, in our case pattern because that’s the argument that is varying.

vgrepl <- Vectorize(grepl, vectorize.args = "pattern")

Here we use function Vectorize and we tell it to vectorize argument pattern. What this will do is run the grepl function for any element of the vector we pass in, just like we did in the i.ind objects a few lines above.

This would be an equivalent of doing it using an anonymouse function

tmp <- sapply(c("^C\_pre", "^T\_pre"), FUN = function(pt, input) {

grepl(pt, x = input)

}, input = xy$sample)

tmp

# ^C\_pre ^T\_pre

# [1,] TRUE FALSE

# [2,] FALSE FALSE

# [3,] FALSE TRUE

# [4,] FALSE FALSE

# [5,] FALSE FALSE

While this can be somewhat verbose, you can use vgrepl as you would use grepl, with the minor detail that you pass a whole vector to pattern instead of a single regular expression.

i.vec <- vgrepl(pattern = c("^C\_pre", "^T\_pre"), x = xy$sample)

# ^C\_pre ^T\_pre

# [1,] TRUE FALSE

# [2,] FALSE FALSE

# [3,] FALSE TRUE

# [4,] FALSE FALSE

# [5,] FALSE FALSE

xy[rowSums(i.vec) > 0, ]

# sample value

# 1 C\_pre\_sample1 0.3048032

# 3 T\_pre\_sample2 0.3359707