SNVsMutations package to determine the mutation type and counts for a set of single nucleotide variants in a vcf file

Federica Boselli[*]

[*]federica.boselli@mail.polimi.it

1 June 2023

Abstract

This package takes:a set of mutations (single nucleotide variants, SNVs) in VCF format, the corresponding reference genome (e.g., human genome hg38) and a parameter “context_length” which is a positive, odd integer and determines for each mutation (only SNVs; other mutations like indels can be ignored) the corresponding mutation type. Lastly, the package can provide a graphical output which plots the summarized mutation type counts.

Package

SNVsMutations 0.1.0

Contents

1 Using the SNVsMUtations package to determine the mutation type for a set of single nucleotide variants in a genome

1.1 Background

The **`SNVsMutations`** package is designed to analyze a set of single nucleotide variants (SNVs) in VCF format, along with the corresponding reference genome (e.g., human genome hg38), and the parameter “context_length.” The goal of this package, developed for the Scientific Programming course at Politecnico of Milan as part of the Bioinformatics for Computational Genomics MSc program, is to determine the mutation types for each SNV and provide a graphical summary of the mutation type counts.

To get started with the SNVsMutations package in R, follow this vignette for a guide on how to use it effectively.

1.1.1 SNVs (Single Nucleotide Variants)

Single-nucleotide variant (SNV), also known as single-nucleotide polymorphism (SNP), is the variant of a single nucleotide that occurs at a specific genomic position.

1.1.2 VCF file

VCF is a text file format (most likely stored in a compressed manner). It contains meta-information lines, a header line, and then data lines each containing information about a position in the genome. The format also has the ability to contain genotype information on samples for each position. From: https://samtools.github.io/hts-specs/VCFv4.2.pdf

1.2 Installation and dependencies

This package depends from: VariantAnnotation, Biostrings, BSgenome.Hsapiens.UCSC.hg38, GenomicRanges. So to be able to install this package, these 4 packages must be installed.

For example, to manually install a package from BiocManager: `BiocManager::install("Biostrings")`

After that, `SNVsMutations` can be installed from the command line as follows: `R CMD install SNVsMutations.tar.gz`.

To load the packages if they have been already installed:

```r
library(BiocStyle)
library(knitr)
library(BSgenome.Hsapiens.UCSC.hg38)
library(VariantAnnotation)
library(GenomicRanges)
library(ggplot2)
library(SNVsMutations)
```

1.3 Loading the VCF file

To load the VCF file it is possible to load the vcf example file from the VariabtAnnotation package documentation. It can be found at: https://bioconductor.org/packages/release/bioc/vignettes/VariantAnnotation/inst/doc/VariantAnnotation.pdf For example:

```r
fl <- system.file("extdata", "chr22.vcf.gz", package="VariantAnnotation")
vcf <- readVcf(fl, "hg38")[1:200]
```

In alternative, it is possible to load another vcf file by specifying the path

```r
file <- ('/Users/federicaboselli/Downloads/HG02024_VN049_KHVTrio.chr1.vcf')
vcf2 <- readVcf(file,"hg38")
```

In both cases the vcf file can be read by using the `readVcf`()` function. In this case we selected only the first 200 rows to make analysis more easy, smooth and less computationally expensive.

In this example vignette we will use chr22.vcf.gz

1.4 Getting the mutation types from the vcf file through the `SNV_Types()` function

The `SNV_Types()` function takes a VCF file, a context length, and a reference genome as input. It processes only the single nucleotide variants (SNVs) present in the VCF file and returns a vector of characters representing the mutation types.

Regarding the function inputs: - the vcf file is the one previously loaded by the user (see ‘loading the vcf file’) - the context_length parameter should be a positive odd integer. It specifies the number of bases to include both upstream and downstream of each single nucleotide variant (SNV) position in the mutation type string. - the reference genome should be in concordance with the used vcf file and can be loaded by using the BSgenome library. In particular, the BSgenome library should correspond to the reference genome.

For example, to load the human reference genome hg38

```r
ref_genome <- Hsapiens
```

Overall, a possible example on how to use the function can be this one:

```r
mut_type <- SNV_Types(vcf, context_length = 3, ref_genome)
mut_type
```

```
##   [1] "C[T>C]C" "C[C>T]G" "C[C>T]C" "T[C>T]G" "C[C>T]G" "G[C>T]G" "G[T>C]G"
##   [8] "G[C>T]G" "A[C>T]C" "A[T>C]G" "A[C>T]A" "C[C>T]A" "G[T>C]C" "A[C>T]C"
##  [15] "T[C>T]C" "C[C>A]C" "T[C>T]T" "T[C>T]T" "G[C>T]T" "T[C>T]C" "C[C>T]A"
##  [22] "C[C>T]G" "G[T>G]A" "G[C>T]T" "C[T>C]G" "G[C>T]G" "C[C>T]C" "T[C>T]C"
##  [29] "T[C>T]A" "T[T>C]G" "G[C>T]T" "C[C>T]T" "C[C>T]C" "A[C>A]G" "G[C>T]G"
##  [36] "G[C>T]C" "T[C>G]A" "G[C>T]A" "A[C>G]A" "A[T>C]A" "C[T>C]C" "A[C>T]A"
##  [43] "G[C>T]G" "C[C>T]G" "A[C>T]C" "G[T>C]C" "G[C>T]T" "G[C>T]C" "C[T>A]A"
##  [50] "A[C>T]C" "A[C>T]C" "G[T>C]A" "T[C>T]A" "C[C>T]C" "C[C>T]C" "T[C>G]G"
##  [57] "C[T>C]G" "C[C>T]T" "A[C>T]A" "C[C>T]C" "C[T>C]C" "T[C>G]T" "A[C>T]G"
##  [64] "A[T>G]G" "T[C>T]C" "T[C>T]A" "A[C>T]C" "G[T>G]G" "C[T>C]G" "C[C>G]A"
##  [71] "C[T>C]T" "C[C>T]T" "G[C>T]G" "G[C>T]C" "G[C>T]T" "G[C>T]A" "C[C>T]A"
##  [78] "T[T>A]C" "T[C>T]A" "C[C>T]A" "G[C>T]T" "G[C>T]G" "G[T>C]T" "A[C>T]A"
##  [85] "C[T>C]A" "C[C>T]G" "T[C>T]T" "C[C>T]A" "G[C>T]T" "T[C>T]A" "A[C>T]G"
##  [92] "A[T>G]C" "A[C>T]C" "C[C>T]G" "C[C>T]C" "G[C>A]A" "G[C>G]T" "A[T>C]G"
##  [99] "G[C>T]C" "C[C>T]A" "G[C>T]T" "G[C>T]A" "T[C>G]C" "T[C>T]T" "C[C>T]A"
## [106] "G[C>T]T" "C[T>A]T" "G[C>A]G" "C[T>G]G" "C[C>G]C" "C[C>T]C" "T[T>C]G"
## [113] "G[C>T]C" "A[C>T]C" "C[C>T]C" "C[C>G]G" "G[C>T]G" "C[C>G]T" "T[T>A]C"
## [120] "T[C>T]G" "C[C>T]A" "C[C>T]G" "A[C>T]A" "C[C>T]C" "C[C>T]G" "T[C>T]A"
## [127] "C[T>C]T" "C[C>T]T" "G[C>G]A" "G[C>T]G" "G[C>A]T" "A[C>T]C" "C[C>T]G"
## [134] "G[C>T]G" "G[C>T]A" "G[C>T]G" "C[C>T]A" "T[C>A]A" "C[T>G]C" "G[C>T]A"
## [141] "C[C>T]C" "C[C>G]G" "G[C>T]T" "C[C>T]G" "G[C>T]C" "C[T>C]G" "T[C>A]C"
## [148] "A[C>G]G" "G[T>A]G" "G[C>T]G" "G[C>T]G" "G[C>T]G" "A[T>C]A" "G[T>C]A"
## [155] "C[C>T]G" "G[C>T]G" "T[C>T]T" "A[C>G]C" "C[C>T]C" "C[C>T]C" "T[C>G]G"
## [162] "C[C>T]T" "G[T>G]C" "C[C>G]C" "C[C>T]T" "G[T>G]C" "G[C>T]C" "C[C>T]T"
## [169] "A[C>A]C" "G[C>T]A" "C[C>T]A" "A[C>T]A" "C[T>G]T" "G[C>T]C" "T[C>T]A"
## [176] "G[C>G]C" "G[C>T]A" "G[C>T]T" "T[T>A]C" "A[C>T]A" "C[C>T]C" "A[T>C]C"
## [183] "T[C>T]A" "C[C>T]G" "G[T>A]T" "C[T>A]C" "C[T>A]C" "T[C>T]T" "G[C>G]C"
## [190] "C[T>C]G" "T[C>T]C" "G[C>T]T" "T[C>T]C" "T[C>T]C" "A[C>T]A"
```

NB: The mutation “C[G>A]A” corresponds to “T[C>T]G” when considering the reverse strand, indicating a redundancy that can be resolved by converting the mutation types identified for REF bases A and G to their corresponding reverse complements. Therefore, in this function all mutation types are reported with either C or T as the mutated REF base.

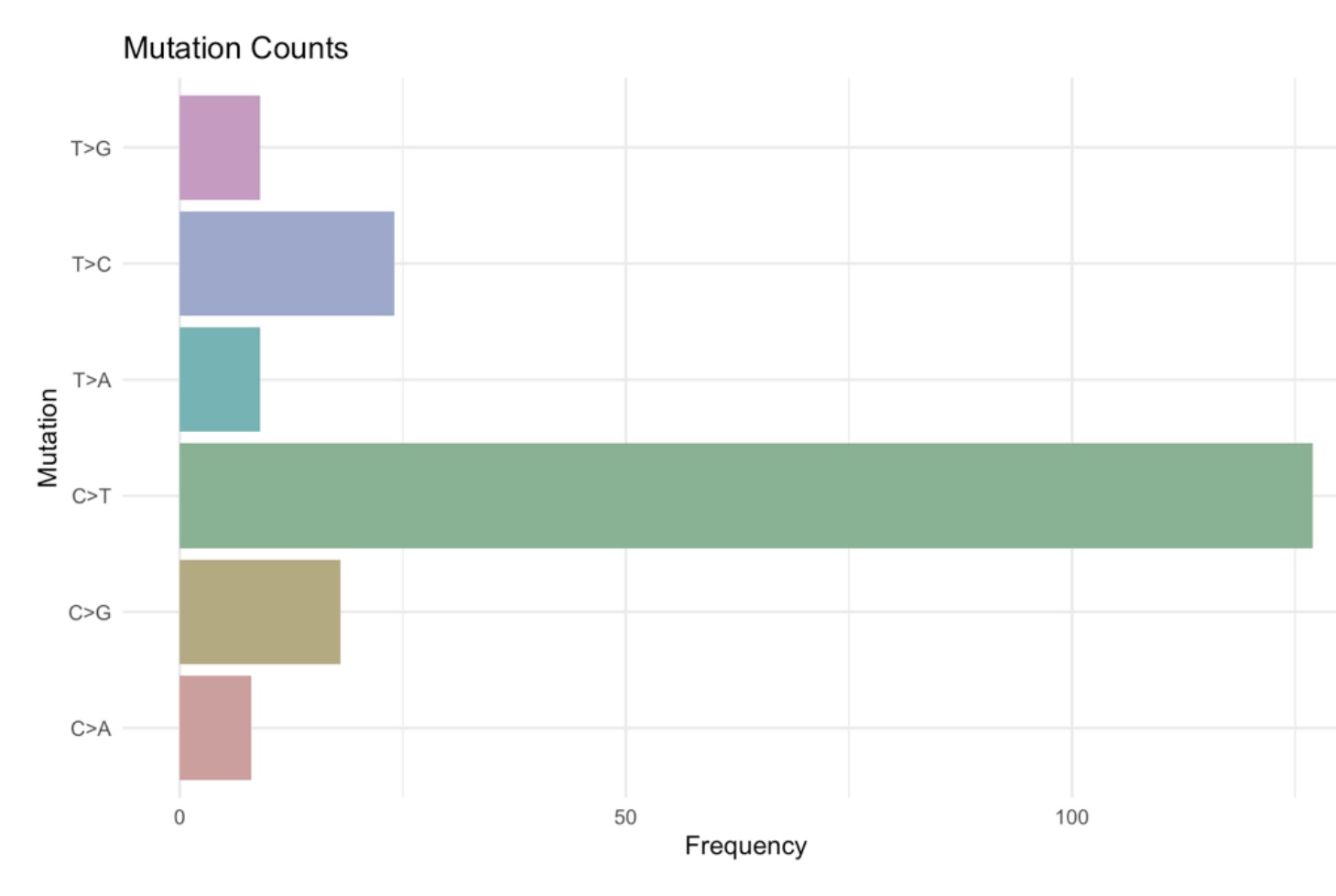
1.5 Generating the mutation counts and a final graphical output with the `SNV_Counts()` function

The function `SNV_Counts()`t takes either a vector of mutation types (mutationstype_vector) or a VCF file, reference genome, and context length as inputs. It counts the occurrences of different mutation types and generates a plot to visualize the counts. If the onlySNV parameter is set to TRUE, it specifically counts single-nucleotide variant (SNV) mutation types.

If a mutationstype_vector is provided, it is used directly to count the mutation types. Otherwise, the mutation types are computed using the MutationsType function based on the vcf_file, reference_genome, and context_length parameters. The function also creates a bar plot showing the mutation counts with different colors for each mutation type. The plot is displayed if plot is set to TRUE. In particular: If onlySNV is TRUE, the plot shows the mutation types by considering only reference and alternative allele (SNV only). If onlySNV is FALSE, the plot shows the mutation types by also considering the bases upstream and downstream (taking into account the ‘context_length’ variable) of each SNV.

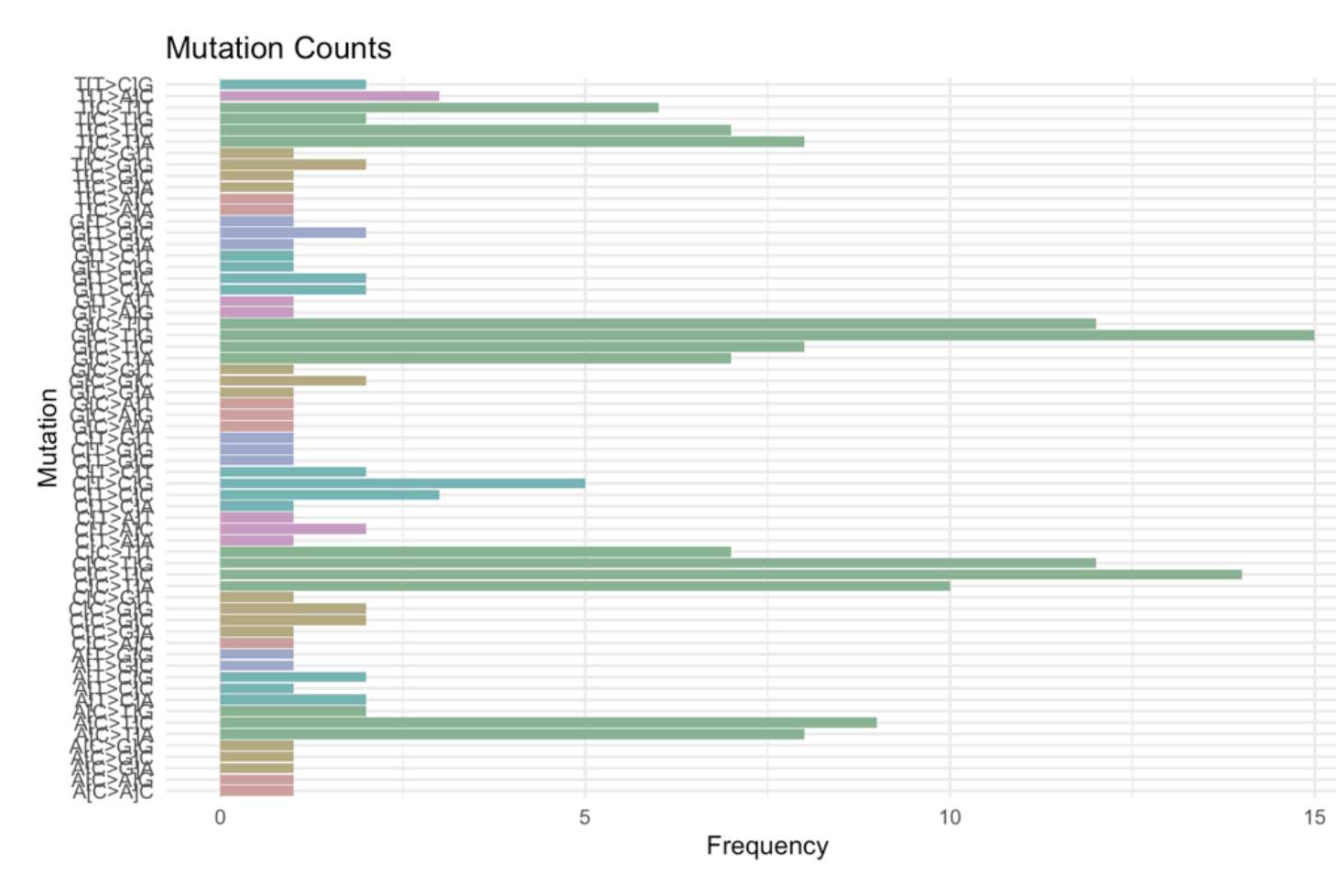
Example usage of the `SNV_Counts()` function using the mut_type object previously generated through the `SNV_Types()` function:

```r
# plotting only the Single Nucleotide Variants
SNV_Counts(mut_type, plot=TRUE, onlySNV=TRUE)
```



```
##     Mutation Freq   Color
## C>A      C>A    8 #CEA09E
## C>G      C>G   18 #B5AB85
## C>T      C>T  127 #8BB594
## T>A      T>A    9 #79B5B7
## T>C      T>C   24 #9FABCC
## T>G      T>G    9 #C79FC1
```

```r
# plotting the Single Nucleotide Variants with the context bases
SNV_Counts(mut_type, plot=TRUE, onlySNV=FALSE)
```



```
##    Mutation Freq   Color
## 1   A[C>A]C    1 #CEA09E
## 2   A[C>A]G    1 #CEA09E
## 3   A[C>G]A    1 #B5AB85
## 4   A[C>G]C    1 #B5AB85
## 5   A[C>G]G    1 #B5AB85
## 6   A[C>T]A    8 #8BB594
## 7   A[C>T]C    9 #8BB594
## 8   A[C>T]G    2 #8BB594
## 9   A[T>C]A    2 #79B5B7
## 10  A[T>C]C    1 #79B5B7
## 11  A[T>C]G    2 #79B5B7
## 12  A[T>G]C    1 #9FABCC
## 13  A[T>G]G    1 #9FABCC
## 14  C[C>A]C    1 #CEA09E
## 15  C[C>G]A    1 #B5AB85
## 16  C[C>G]C    2 #B5AB85
## 17  C[C>G]G    2 #B5AB85
## 18  C[C>G]T    1 #B5AB85
## 19  C[C>T]A   10 #8BB594
## 20  C[C>T]C   14 #8BB594
## 21  C[C>T]G   12 #8BB594
## 22  C[C>T]T    7 #8BB594
## 23  C[T>A]A    1 #C79FC1
## 24  C[T>A]C    2 #C79FC1
## 25  C[T>A]T    1 #C79FC1
## 26  C[T>C]A    1 #79B5B7
## 27  C[T>C]C    3 #79B5B7
## 28  C[T>C]G    5 #79B5B7
## 29  C[T>C]T    2 #79B5B7
## 30  C[T>G]C    1 #9FABCC
## 31  C[T>G]G    1 #9FABCC
## 32  C[T>G]T    1 #9FABCC
## 33  G[C>A]A    1 #CEA09E
## 34  G[C>A]G    1 #CEA09E
## 35  G[C>A]T    1 #CEA09E
## 36  G[C>G]A    1 #B5AB85
## 37  G[C>G]C    2 #B5AB85
## 38  G[C>G]T    1 #B5AB85
## 39  G[C>T]A    7 #8BB594
## 40  G[C>T]C    8 #8BB594
## 41  G[C>T]G   15 #8BB594
## 42  G[C>T]T   12 #8BB594
## 43  G[T>A]G    1 #C79FC1
## 44  G[T>A]T    1 #C79FC1
## 45  G[T>C]A    2 #79B5B7
## 46  G[T>C]C    2 #79B5B7
## 47  G[T>C]G    1 #79B5B7
## 48  G[T>C]T    1 #79B5B7
## 49  G[T>G]A    1 #9FABCC
## 50  G[T>G]C    2 #9FABCC
## 51  G[T>G]G    1 #9FABCC
## 52  T[C>A]A    1 #CEA09E
## 53  T[C>A]C    1 #CEA09E
## 54  T[C>G]A    1 #B5AB85
## 55  T[C>G]C    1 #B5AB85
## 56  T[C>G]G    2 #B5AB85
## 57  T[C>G]T    1 #B5AB85
## 58  T[C>T]A    8 #8BB594
## 59  T[C>T]C    7 #8BB594
## 60  T[C>T]G    2 #8BB594
## 61  T[C>T]T    6 #8BB594
## 62  T[T>A]C    3 #C79FC1
## 63  T[T>C]G    2 #79B5B7
```

Session info

```
## R version 4.2.3 (2023-03-15)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur ... 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0
.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack
.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] SNVsMutations_0.1.0               ggplot2_3.4.2
##  [3] VariantAnnotation_1.44.1          Rsamtools_2.14.0
##  [5] SummarizedExperiment_1.28.0       Biobase_2.58.0
##  [7] MatrixGenerics_1.10.0             matrixStats_0.63.0
##  [9] BSgenome.Hsapiens.UCSC.hg38_1.4.5 BSgenome_1.66.3
## [11] rtracklayer_1.58.0                Biostrings_2.66.0
## [13] XVector_0.38.0                    GenomicRanges_1.50.2
## [15] GenomeInfoDb_1.34.9               IRanges_2.32.0
## [17] S4Vectors_0.36.2                  BiocGenerics_0.44.0
## [19] knitr_1.43                        BiocStyle_2.26.0
##
## loaded via a namespace (and not attached):
##  [1] bitops_1.0-7             bit64_4.0.5              filelock_1.0.2

##  [4] progress_1.2.2           httr_1.4.6               tools_4.2.3

##  [7] bslib_0.4.2              utf8_1.2.3               R6_2.5.1

## [10] DBI_1.1.3                colorspace_2.1-0         withr_2.5.0

## [13] tidyselect_1.2.0         prettyunits_1.1.1        bit_4.0.5

## [16] curl_5.0.0               compiler_4.2.3           cli_3.6.1

## [19] xml2_1.3.4               DelayedArray_0.24.0      labeling_0.4.2

## [22] bookdown_0.34            sass_0.4.6               scales_1.2.1

## [25] rappdirs_0.3.3           stringr_1.5.0            digest_0.6.31

## [28] rmarkdown_2.21           pkgconfig_2.0.3          htmltools_0.5.5

## [31] highr_0.10               dbplyr_2.3.2             fastmap_1.1.1

## [34] rlang_1.1.1              rstudioapi_0.14          RSQLite_2.3.1

## [37] jquerylib_0.1.4          BiocIO_1.8.0             generics_0.1.3

## [40] farver_2.1.1             jsonlite_1.8.4           BiocParallel_1.32.6

## [43] dplyr_1.1.2              RCurl_1.98-1.12          magrittr_2.0.3

## [46] GenomeInfoDbData_1.2.9   Matrix_1.5-4.1           Rcpp_1.0.10

## [49] munsell_0.5.0            fansi_1.0.4              lifecycle_1.0.3

## [52] stringi_1.7.12           yaml_2.3.7               zlibbioc_1.44.0

## [55] BiocFileCache_2.6.1      grid_4.2.3               blob_1.2.4

## [58] parallel_4.2.3           crayon_1.5.2             lattice_0.21-8

## [61] GenomicFeatures_1.50.4   hms_1.1.3                KEGGREST_1.38.0

## [64] pillar_1.9.0             rjson_0.2.21             codetools_0.2-19

## [67] biomaRt_2.54.1           XML_3.99-0.14            glue_1.6.2

## [70] evaluate_0.21            BiocManager_1.30.20      png_0.1-8

## [73] vctrs_0.6.2              gtable_0.3.3             cachem_1.0.8

## [76] xfun_0.39                restfulr_0.0.15          tibble_3.2.1

## [79] GenomicAlignments_1.34.1 AnnotationDbi_1.60.2     memoise_2.0.1
```