

# KmeRtone User Manual



## Contents

<b>Installation</b>	<b>2</b>
<b>Overview of kmeRtone operations</b>	<b>2</b>
kmeRtone Input Flags - Overview . . . . .	2
kmeRtone Input Flags - Additional Description . . . . .	3
kmeRtone Objects . . . . .	4
Code Convention . . . . .	5
<b>Quick examples</b>	<b>6</b>
Single-nucleotide resolution case coordinates: non-specific patterns . . . . .	6
Single-nucleotide resolution case coordinates: specific patterns . . . . .	9
ChIP-sequencing peaks and related profiling assays spanning large genomic regions . . . . .	14
<b>References</b>	<b>17</b>

# Installation

```
# Install from CRAN
install.packages('kmeRtone')

# Install directly from GitHub
devtools::install_github('SahakyanLab/kmeRtone', ref = 'master')
```

Alternatively, download and install using the [latest release files from here](#).

## Overview of kmeRtone operations

**kmeRtone** contains many modules. The core module (SCORE) calculates the z-score of k-meric enrichment and depletion. Briefly, the input source are case coordinates for the DNA-related phenomenon under study (e.g. DNA damage, DNA binding, DNA breakage, etc.) and a reference to the chromosome-separated FASTA files. **kmeRtone** calculates the k-mer z-score for every k-mer sequence and generates a table of all k-mer sequences and their associated z-scores. Here, the resulting z-scores indicate how enriched ( $z \gg 1$ ) or depleted ( $z \ll 1$ ) a given k-mer sequence is under the studied phenomenon.

### kmeRtone Input Flags - Overview

Here, we highlight some of the key arguments as input to the **kmeRtone** function. Please refer to the documentation of the function for further details on the required and optional arguments.

#### 1. Case coordinate

Flag	Class	Description
case.coor.path	<character>	A path to a <b>folder</b> containing chromosome-separated genomic coordinates or chromosome-combined BED files. This flag is ignored when case.coor is not NULL.
case	<genomic.coordinate>	A pre-loaded <genomic.coordinate> class object..

#### 2. Genome

Flag	Class	Description
genome.name	<character>	Available: “hg19” or “hg38”. User’s own genome name.
genome.path	<character>	A path to a user’s <b>folder</b> containing chromosome-separated fasta files. Default is NULL. The file name must be the name of chromosome.
genome	<genome>	Pre-loaded <genome> class object. Default is NULL. The two flags above are ignored when this is used.

#### 3. Case characteristics

Flag	Class	Description
strand.sensitive	<bool>	Does strand polarity matter?
single.case.length	<int>	Default is NULL for unspecified/varied length.

Flag	Class	Description
case.pattern	<character>	Default is NULL for no pattern.

#### 4. Case coordinate operation

Flag	Class	Description
rm.case.kmer.overlaps	<bool>	Default is TRUE. This is important to remove neighbouring effect.
merge.replicates	<bool>	Default is TRUE. When merging replicates, duplicated coordinates coming from different replicates are removed.
k	<int>	Length of k-mer
ctrl.rel.pos	<character>	Position of control regions relative to the case positions. Input is a vector of length two: <code>c(from, to)</code>

#### 5. Other module flags

Flag	Class	Description
kmer.table	<data.table>	Pre-loaded k-mer table with calculated score. Default is NULL.

#### 6. kmeRtone module

Flag	Class	Description
module	<character>	Available module: “score”, “tune”, “explore”, “evolution”, “genic element”, “cancer”, etc.

#### 7. Other

Flag	Class	Description
ncpu	<int>	Number of CPU cores. Default is 1.
output.path	<character>	A path to an output <b>folder</b> . Default is “data/”

### kmeRtone Input Flags - Additional Description

Flag	Description
single.case.length	The case length unit is number of nucleotide. In an event where case happens in between two nucleotide e.g. DNA breakage, the case.length is 2 nt.
case.coor.path	Three situations can happen. (1) A folder containing a BED file. A second or more BED files indicates a presence of replicates. (2) A folder containing chromosome-separated files. The file name must be the name of chromosome. (3) A folder containing sub-folders of chromosome-separated files, indicating a presence of replicates. In situation (2) and (3), the coordinates must be a 1-based index due to R language conventions. Alternatively, user can specify this with the <code>case.coor.1st.idx</code> argument

## kmeRtone Objects

kmeRtone introduce two class objects: `<genome>` and `<genomic.coordinate>`

### 1. `<genome>`

kmeRtone comes with two pre-built `<genome>`: hg19 and hg38. The `<genome>`s are saved as uncompressed RDS binary object for fast loading. `print.genome` function is built to print the `<genome>` object. It will show the genome name (e.g. hg19) and genome length by chromosome. The default `base::print` showing the very long sequence will crash the R console.

`<genome>` is an S3-class object with the following contents:

```
$seq
  named <character> vector
    chr1          chr2          ...
  c(AACTCGTACC....., ACGTTGGTTC....) ...

$chr.names
<character> vector
c(chr1, chr2, ...)

$length
<character> vector
c(2947924, 2093123, ...)

$name
<character>
hg19
```

### 2. `<genomic.coordinate>`

`<genomic.coordinate>` is an S3-class object. The reason for building this class is to reduce data redundancy in genomic coordinate table (e.g. repeated number of chromosome name and unnecessary column end when case length is fixed). It also helps with organisation of kmeRtone configuration (e.g. k-mer size, case length, etc.) as the `<genomic.coordinate>` object will carry and contain those information. It utilises `<data.table>` to use its inherent feature to update by reference (instead of memory copy) for genomic coordinate table and coordinate status (case vs. k-mer coordinate). This will help to reduce memory (RAM) consumption and keep track what the coordinates refer to (whether the case itself or k-mer). The contents of the `<genomic.coordinate>` object are as follow:

```
$chr1
<data.table>
  start strand ...
1:   12      +
2:   16      +
3:  499      -
...

$chr2 __C__.externalptr

$chr3 ...
```

```

$chr... ...

$chr.names
  <character> vector
  c(chr1, chr2, ...)

$status
  <data.table> single row
    is.kmer
1:    TRUE

$case.length
  <character>
2

$case.pattern
  <character> vector
  c(CT, TT, ...)

```

## Code Convention

- Table column name is written in lowercase and snake\_case.
- Function name is written in camelCase. The function filename if it is saved will be the same like the function name except for workflow functions which begin with capital case corresponds to their module letter.
- Module workflow code begins with a function calling (left-aligned) and ends with variable assignment (right-aligned).
- Workflow boolean is designed to make it natural to read in English e.g. `if(coor$status$is.kmer)` or `if(coor$is.strand.sensitive)`.
- Looping uses singular and plural as variable name i.e. `for (chr.name in chr.names)`.
- The code finish at a standard column number 80 for better viewing.
- This symbol `<>` refers to R class object e.g. `<character>`

## Quick examples

Below we will demonstrate various capabilities of the default `kmeRtone SCORE` function to quantify the k-mer enrichment and depletion.

### Single-nucleotide resolution case coordinates: non-specific patterns

Suppose we expose a DNA sequence to ionisation radiation and record the positions that are broken. For example, the A-T bond is broken between positions 10,000-10,001 on chromosome 1. Here, we are working with single-nucleotide resolution data, where specific bonds are broken between two adjacent nucleotides. Below we show an example simulation of generating files that can be used by `kmeRtone`.

The chromosomes and start positions are randomly sampled with a `width` of 2 as in our simulated example, we are interested in the breakage phenomenon between adjacent two nucleotides. The `width` depends on your DNA phenomenon under study. For instance, if you are working with DNA damages, where, for instance, the thymine base at position 50,000 on chromosome 1 is damaged, then the `width` variable is 1 and the `single.case.len` is also 1. For the DNA breakage example, the `width` variable is 2, hence the `single.case.len` is also 2. Finally, save the results as chromosome-separated files in your desired folder.

```
library(data.table)
library(kmeRtone)

#' 1. Randomly generate genomic positions and save results
dir.create("./data", showWarnings = FALSE)

set.seed(1234)
for(chr in 1){
  genomic_coor <- data.table::data.table(
    seqnames = paste0("chr", chr),
    start = sample(
      x = 10000:10000000,
      size = 100000,
      replace = FALSE
    ),
    width = 2 # 2 for bonds between bases, 1 for the base
  )
  data.table::setorder(genomic_coor, -start)

  data.table::fwrite(
    genomic_coor,
    paste0("./data/chr", chr, ".csv")
  )
}
```

To run a k-mer enrichment and depletion analysis with `kmeRtone` on this simulated dataset, run the below function. Here, we specified `k` as 4, thus, the z-scores are calculated for each 4-mer sequence.

You will need to be conscious on the specified control range from which you sample the negative control k-mer population. If it's too close to the case regions, you may have too much of a sequence-context bias from the case region that may influence your negative control population, even though the coordinates are not overlapping. Similarly, if it's too far from the case regions, you may not capture enough of the local sequence variation and only capture the broad sequence influence. One of the advantages of `KmeRtone` is the inherent flexibility that comes with the choice of the control regions, giving the user full control of which genomic regions to start and stop the sampling. This gives you more accurate statistical testing than simply

comparing the case regions to the overall genome-wide average. With that said, unless you know what range makes sense, we otherwise recommend that you experiment with different ranges and make an informed decision based on this. For now, we will stick to the default values.

Please refer to the documentation of the function for further details on the required and optional arguments. Please note, that your results will differ depending on which `ctrl.rel.pos` range you use, whether you have a specific `case.pattern` of interest, whether to remove overlapping case k-mers in case regions with `rm.case.kmer.overlaps`, whether to merge replicates or treat them separately with `merge.replicate`, and more.

```
#' 2. Run kmeRtone `score` function
```

```
kmeRtone::kmeRtone(  
  case.coor.path="./data",  
  genome.name="hg19",  
  strand.sensitive=FALSE,  
  k=4,  
  ctrl.rel.pos=c(80, 500),  
  case.pattern=NULL,  
  single.case.len=2,  
  output.dir="output",  
  module="score",  
  rm.case.kmer.overlaps=FALSE,  
  merge.replicate=TRUE,  
  verbose=TRUE  
)
```

The above should generate the below output. The results are saved in the path you specified in the `output.dir` argument.

```
-----  
                        Extraction of Case K-mers  
-----  
Extracting 2-mers from chr1.....DONE! -- 3.23 secs  
Extracting 2-mers from chr2.....DONE! -- 3.28 secs  
Extracting 2-mers from chr3.....DONE! -- 2.64 secs  
Extracting 2-mers from chr4.....DONE! -- 2.56 secs  
Extracting 2-mers from chr5.....DONE! -- 2.31 secs  
Extracting 2-mers from chr6.....DONE! -- 2.33 secs  
Extracting 2-mers from chr7.....DONE! -- 2.04 secs  
Extracting 2-mers from chr8.....DONE! -- 1.97 secs  
Extracting 2-mers from chr9.....DONE! -- 1.75 secs  
Extracting 2-mers from chr10.....DONE! -- 1.82 secs  
Extracting 2-mers from chr11.....DONE! -- 1.75 secs  
Extracting 2-mers from chr12.....DONE! -- 1.8 secs  
Extracting 2-mers from chr13.....DONE! -- 1.35 secs  
Extracting 2-mers from chr14.....DONE! -- 1.25 secs  
Extracting 2-mers from chr15.....DONE! -- 1.22 secs  
Extracting 2-mers from chr16.....DONE! -- 1.04 secs  
Extracting 2-mers from chr17.....DONE! -- 0.97 secs  
Extracting 2-mers from chr18.....DONE! -- 0.97 secs  
Extracting 2-mers from chr19.....DONE! -- 0.74 secs  
Extracting 2-mers from chr20.....DONE! -- 0.71 secs  
Extracting 2-mers from chr21.....DONE! -- 0.53 secs  
Extracting 2-mers from chr22.....DONE! -- 0.55 secs
```

Total time taken: 37.2 secs

-----  
Extraction of Control K-mers  
-----

Building control regions of chr1.....DONE! -- 3.14 secs  
Building control regions of chr2.....DONE! -- 3.14 secs  
Building control regions of chr3.....DONE! -- 2.57 secs  
Building control regions of chr4.....DONE! -- 2.56 secs  
Building control regions of chr5.....DONE! -- 2.31 secs  
Building control regions of chr6.....DONE! -- 2.28 secs  
Building control regions of chr7.....DONE! -- 2.06 secs  
Building control regions of chr8.....DONE! -- 1.98 secs  
Building control regions of chr9.....DONE! -- 1.77 secs  
Building control regions of chr10.....DONE! -- 1.78 secs  
Building control regions of chr11.....DONE! -- 1.9 secs  
Building control regions of chr12.....DONE! -- 1.79 secs  
Building control regions of chr13.....DONE! -- 1.36 secs  
Building control regions of chr14.....DONE! -- 1.32 secs  
Building control regions of chr15.....DONE! -- 1.16 secs  
Building control regions of chr16.....DONE! -- 1.06 secs  
Building control regions of chr17.....DONE! -- 1.06 secs  
Building control regions of chr18.....DONE! -- 0.95 secs  
Building control regions of chr19.....DONE! -- 0.71 secs  
Building control regions of chr20.....DONE! -- 0.76 secs  
Building control regions of chr21.....DONE! -- 0.53 secs  
Building control regions of chr22.....DONE! -- 0.63 secs

Total time taken: 36.97 secs

Extracting 2-mers from chr1.....DONE! -- 3.11 secs  
Extracting 2-mers from chr2.....DONE! -- 3.13 secs  
Extracting 2-mers from chr3.....DONE! -- 2.62 secs  
Extracting 2-mers from chr4.....DONE! -- 2.43 secs  
Extracting 2-mers from chr5.....DONE! -- 2.41 secs  
Extracting 2-mers from chr6.....DONE! -- 2.22 secs  
Extracting 2-mers from chr7.....DONE! -- 2.17 secs  
Extracting 2-mers from chr8.....DONE! -- 1.93 secs  
Extracting 2-mers from chr9.....DONE! -- 1.86 secs  
Extracting 2-mers from chr10.....DONE! -- 1.78 secs  
Extracting 2-mers from chr11.....DONE! -- 1.85 secs  
Extracting 2-mers from chr12.....DONE! -- 1.76 secs  
Extracting 2-mers from chr13.....DONE! -- 1.31 secs  
Extracting 2-mers from chr14.....DONE! -- 1.32 secs  
Extracting 2-mers from chr15.....DONE! -- 1.18 secs  
Extracting 2-mers from chr16.....DONE! -- 1.09 secs  
Extracting 2-mers from chr17.....DONE! -- 1.02 secs  
Extracting 2-mers from chr18.....DONE! -- 1.07 secs  
Extracting 2-mers from chr19.....DONE! -- 0.68 secs  
Extracting 2-mers from chr20.....DONE! -- 0.72 secs  
Extracting 2-mers from chr21.....DONE! -- 0.53 secs  
Extracting 2-mers from chr22.....DONE! -- 0.55 secs

Total time taken: 36.97 secs  
-----



### Calculation of K-mer Susceptibility

-----  
The 2-mer scores are saved at output/score\_2-mer.csv

FINISH! Total time taken: 1.85 mins

## Single-nucleotide resolution case coordinates: specific patterns

Suppose we expose a DNA sequence to UV-light and record the pyrimidine-pyrimidone (6-4) photoproduct and cyclobutane pyrimidine dimers as done in this paper (Hu et al. 2017). We can retrieve the deposited data through the GEO repository (Edgar, Domrachev, and Lash 2002) under the accession identifier [GSE98025](#). For the purposes of this demonstration, we shall download one of the samples: NHF1\_CPD\_10J\_48h\_A from [GSM2585697](#).

```
#!/bin/bash
wget -O UV_data.bed.gz https://ftp.ncbi.nlm.nih.gov/geo/samples/GSM2585nnn/GSM2585697/
suppl/GSM2585697%5FNCPPD2DA5N.1.cu.bo.hg19.coToBa.coToBe.unSo.coBeToSiFr.slBeb6.
coToFiRa10.soBe.coBeToFa.gePyDi.soBe.bed.gz

gunzip UV_data.bed.gz

head UV_data.bed
```

which should show the below as output.

chr1	10108	10118	+
chr1	10163	10173	+
chr1	10194	10204	-
chr1	10275	10285	-
chr1	10299	10309	-
chr1	10342	10352	-
chr1	10346	10356	+
chr1	10357	10367	+
chr1	10377	10387	-
chr1	10397	10407	-

As per the GEO repository, “the Bed files contain genomic locations of damages of the most common two dinucleotides at the damage sites for (6-4)PP and CPD. Each interval length is 10 nt, and the pyrimidine dimer is located at the 4-5th positions”.

KmeRtone requires you to have column headers for it to work. For the purposes of this demonstration, we will import this data into R using the `fread` function from the `data.table` library, and label the four columns as the following: `seqnames`, `start`, `end`, and `strand`.

```
library(data.table)
library(kmeRtone)

df <- data.table::fread("UV_data.bed")
data.table::setnames(df, c("seqnames", "start", "end", "strand"))

print(head(df))
```

which should show the below as output.

	seqnames	start	end	strand
	<char>	<int>	<int>	<char>
1:	chr1	10108	10118	+
2:	chr1	10163	10173	+
3:	chr1	10194	10204	-
4:	chr1	10275	10285	-
5:	chr1	10299	10309	-
6:	chr1	10342	10352	-

The GEO repository outlined that the dimer is located at the 4-5th position. Before proceeding with the `kmeRtone` analysis, we shall briefly verify this with the below commands.

```
library(BSgenome.Hsapiens.UCSC.hg19)
hg19 <- BSgenome.Hsapiens.UCSC.hg19

library(plyranges)
df_granges <- plyranges::as_granges(df)

print(head(df_granges))
```

which should show the below as output.

```
GRanges object with 6 ranges and 0 metadata columns:
      seqnames      ranges strand
      <Rle>      <IRanges> <Rle>
[1]      chr1 10108-10118      +
[2]      chr1 10163-10173      +
[3]      chr1 10194-10204      -
[4]      chr1 10275-10285      -
[5]      chr1 10299-10309      -
[6]      chr1 10342-10352      -
-----
seqinfo: 25 sequences from an unspecified genome; no seqlengths
```

Next, we visualise the top 10 genomic sequences.

```
sequences <- getSeq(hg19, df_granges[1:10])

print(sequences)
```

which should show the below as output.

```
DNAStringSet object of length 10:
      width seq
[1]    11 CAACCCTAACC
[2]    11 TAACCCTAACC
[3]    11 AGGGTTAGGGT
[4]    11 AGGGTTAGGGT
[5]    11 GGGGTTGGGGT
[6]    11 AGGGTTAGGGT
[7]    11 TAACCCTACCC
[8]    11 TAACCCTAACC
[9]    11 AGGGTTAGGGT
[10]   11 AGGGTTAGGGG
```

We can see from the above print statement, that, indeed, the middle two nucleotides are of the *CT* or *TT* motifs.

Again, we will split this `data.table` into chromosome-separated files. We will only focus on the 22 autosomes, but you can use the X and Y chromosomes too, if available. Here, we save it as `csv` files since we are already operating on it with `data.table`, however, if you are using `plyranges` or similar libraries, you may continue using the `bed` file format.

```
dir.create(path = "./data", showWarnings = FALSE)
for(chr in 1:22){
  data.table::fwrite(
    df[seqnames == paste0("chr", chr)],
    paste0("./data/chr", chr, ".csv")
  )
}
```

To run a k-meric enrichment and depletion analysis with `kmeRtone` on this publicly-available dataset, run the below function. Here, we specified `k` as 4, thus, the z-scores are calculated for each 4-mer sequence. The same negative control range as described previously applies here too. For now, we will stick to the default values.

We set the `single.case.len` argument to 2 as this retrieved dataset specifically records the pyrimidine dimer at the 4-5th positions. We also set the `case.pattern` argument to *CT* as we know these UV-induced damaged sites are of the *CT* motif, and we are specifically interested in the k-meric enrichment and depletion in the context of this *CT* motif. We have data recording the damaged site on both the plus and minus strand, hence, we set the `strand.sensitive` argument to *TRUE*.

Please refer to the documentation of the function for further details on the required and optional arguments. Please note, that your results will differ depending on which `ctrl.rel.pos` range you use, whether to remove overlapping case k-mers in case regions with `rm.case.kmer.overlaps`, whether to merge replicates or treat them separately with `merge.replicate`, and more.

```
#' 2. Run kmeRtone `score` function
kmeRtone::kmeRtone(
  case.coor.path="./data",
  genome.name="hg19",
  strand.sensitive=TRUE,
  k=4,
  ctrl.rel.pos=c(80, 500),
  case.pattern=c("CT","TT"),
  output.dir="output",
  module="score",
  rm.case.kmer.overlaps=FALSE,
  merge.replicate=TRUE,
  verbose=TRUE
)
```

The above should generate the below output. The results are saved in the path you specified in the `output.dir` argument.

```
-----
                        Extraction of Case K-mers
-----
Extracting 4-mers from chr1.....DONE! -- 8.54 secs
Extracting 4-mers from chr2.....DONE! -- 10.65 secs
```

```

Extracting 4-mers from chr3.....DONE! -- 8.45 secs
Extracting 4-mers from chr4.....DONE! -- 8.82 secs
Extracting 4-mers from chr5.....DONE! -- 9.62 secs
Extracting 4-mers from chr6.....DONE! -- 6.77 secs
Extracting 4-mers from chr7.....DONE! -- 6.77 secs
Extracting 4-mers from chr8.....DONE! -- 6.73 secs
Extracting 4-mers from chr9.....DONE! -- 5.24 secs
Extracting 4-mers from chr10.....DONE! -- 6.38 secs
Extracting 4-mers from chr11.....DONE! -- 5.94 secs
Extracting 4-mers from chr12.....DONE! -- 6.11 secs
Extracting 4-mers from chr13.....DONE! -- 4.82 secs
Extracting 4-mers from chr14.....DONE! -- 5.32 secs
Extracting 4-mers from chr15.....DONE! -- 3.04 secs
Extracting 4-mers from chr16.....DONE! -- 2.86 secs
Extracting 4-mers from chr17.....DONE! -- 2.38 secs
Extracting 4-mers from chr18.....DONE! -- 3.04 secs
Extracting 4-mers from chr19.....DONE! -- 1.83 secs
Extracting 4-mers from chr20.....DONE! -- 2.1 secs
Extracting 4-mers from chr21.....DONE! -- 1.64 secs
Extracting 4-mers from chr22.....DONE! -- 1.15 secs

```

Total time taken: 1.97 mins

---

Extraction of Control K-mers

---

```

Building control regions of chr1.....DONE! -- 8.69 secs
Building control regions of chr2.....DONE! -- 9.24 secs
Building control regions of chr3.....DONE! -- 7.22 secs
Building control regions of chr4.....DONE! -- 6.85 secs
Building control regions of chr5.....DONE! -- 6.59 secs
Building control regions of chr6.....DONE! -- 6.18 secs
Building control regions of chr7.....DONE! -- 5.93 secs
Building control regions of chr8.....DONE! -- 6.24 secs
Building control regions of chr9.....DONE! -- 5.1 secs
Building control regions of chr10.....DONE! -- 5.02 secs
Building control regions of chr11.....DONE! -- 4.99 secs
Building control regions of chr12.....DONE! -- 6.32 secs
Building control regions of chr13.....DONE! -- 3.53 secs
Building control regions of chr14.....DONE! -- 3.46 secs
Building control regions of chr15.....DONE! -- 3.15 secs
Building control regions of chr16.....DONE! -- 3.09 secs
Building control regions of chr17.....DONE! -- 3.02 secs
Building control regions of chr18.....DONE! -- 2.7 secs
Building control regions of chr19.....DONE! -- 2.16 secs
Building control regions of chr20.....DONE! -- 2.24 secs
Building control regions of chr21.....DONE! -- 1.41 secs
Building control regions of chr22.....DONE! -- 1.47 secs

```

Total time taken: 1.75 mins

```

Extracting 4-mers from chr1.....DONE! -- 29.23 secs
Extracting 4-mers from chr2.....DONE! -- 34.54 secs
Extracting 4-mers from chr3.....DONE! -- 24.51 secs
Extracting 4-mers from chr4.....DONE! -- 19.64 secs

```

```

Extracting 4-mers from chr5.....DONE! -- 18.15 secs
Extracting 4-mers from chr6.....DONE! -- 17.05 secs
Extracting 4-mers from chr7.....DONE! -- 15.26 secs
Extracting 4-mers from chr8.....DONE! -- 13.53 secs
Extracting 4-mers from chr9.....DONE! -- 12.37 secs
Extracting 4-mers from chr10.....DONE! -- 13.15 secs
Extracting 4-mers from chr11.....DONE! -- 13.85 secs
Extracting 4-mers from chr12.....DONE! -- 15.26 secs
Extracting 4-mers from chr13.....DONE! -- 10.26 secs
Extracting 4-mers from chr14.....DONE! -- 8.96 secs
Extracting 4-mers from chr15.....DONE! -- 8.36 secs
Extracting 4-mers from chr16.....DONE! -- 7.45 secs
Extracting 4-mers from chr17.....DONE! -- 8.99 secs
Extracting 4-mers from chr18.....DONE! -- 7.3 secs
Extracting 4-mers from chr19.....DONE! -- 5.66 secs
Extracting 4-mers from chr20.....DONE! -- 5.73 secs
Extracting 4-mers from chr21.....DONE! -- 2.79 secs
Extracting 4-mers from chr22.....DONE! -- 4.39 secs

```

Total time taken: 4.94 mins

-----  
 Calculation of K-mer Susceptibility  
 -----

The 4-mer scores are saved at output/score\_4-mer.csv

FINISH! Total time taken: 8.66 mins

To explore the results, we run the below command with the resulting k-mer score table displayed after.

```

scores <- data.table::fread('./output/score_4-mers.csv')
print(scores)

```

This highlights one of the key advantages of the *kmerTone* software. It is highly flexible, where the user can specify the case k-mer patterns (*CT* and *TT* patterns in this example) and quantify the corresponding k-mer enrichment and depletion z-scores in the context of these patterns. As a result, the k-mer table only includes k-mer sequences with a *CT* and *TT* motif central to the k-mer. This functionality allows for a more comprehensive approach to understanding the functional implications of specific DNA sequences on a genomic scale.

	kmer <char>	case <int>	control <int>	case_skew <num>	control_skew <num>	z <num>
1:	ACTA	172188	7164905	0.093850907	0.0013140439	-734.98933
2:	ACTC	199682	8271891	0.020853157	0.0003088774	-790.02857
3:	ACTG	183576	10687722	0.019664880	-0.0003821207	-968.57235
4:	ACTT	941818	10809720	0.067322986	-0.0005173122	-291.95047
5:	ATTA	1542837	12150674	0.072150849	-0.0007008665	102.75058
6:	ATTC	752161	10080861	0.051631233	-0.0019745337	-399.40541
7:	ATTG	798686	9024810	0.022557050	0.0010810200	-253.89860
8:	ATTT	3936871	19988383	0.068213310	-0.0034174350	1073.66234
9:	CCTA	178612	6457019	0.060130338	-0.0005428201	-669.85683
10:	CCTC	276033	11817164	0.011255901	-0.0016657127	-956.94049
11:	CCTG	233643	14169493	-0.005311522	0.0001087548	-1128.42319
12:	CCTT	1097911	10403912	0.043268535	-0.0027333949	-110.26574

13:	CTTA	885911	7917213	0.068468503	-0.0005101543	-42.91182
14:	CTTC	762270	10692993	0.050546394	-0.0028651473	-443.96560
15:	CTTG	716068	10115452	0.027653798	-0.0015493129	-436.14909
16:	CTTT	2981906	15085303	0.058601445	-0.0034241937	934.47375
17:	GCTA	150920	6088128	0.061197986	0.0002365259	-670.99816
18:	GCTC	160017	7727359	0.016404507	-0.0028991794	-791.18668
19:	GCTG	217984	12044290	-0.049434821	0.0006109119	-1020.65321
20:	GCTT	654241	8471154	0.045541322	-0.0010154461	-342.77052
21:	GTTA	359542	6274672	0.083912311	-0.0006272838	-441.29095
22:	G TTC	277604	6793487	0.056656244	-0.0020132518	-585.89553
23:	G TTG	360434	7178590	-0.004616657	0.0009609129	-528.67826
24:	G TTT	1583482	11779426	0.057679216	-0.0028104935	177.10006
25:	TCTA	682672	8806301	0.063099116	-0.0017163847	-347.10919
26:	TCTC	910395	13001127	0.027013549	-0.0023806398	-504.67948
27:	TCTG	777519	13250152	0.026007081	-0.0018381676	-632.59802
28:	TCTT	2413768	14131267	0.058897955	-0.0043706626	602.27398
29:	TTTA	3904809	16155176	0.061480344	-0.0025543516	1497.59726
30:	TTTC	2550044	15498635	0.036584467	-0.0042352762	558.16449
31:	TTTG	2891034	15209799	0.023224908	-0.0030150957	850.09419
32:	TTTT	9497245	30534758	0.063873576	-0.0065422493	3271.57838
	kmer	case	control	case_skew	control_skew	z

## ChIP-sequencing peaks and related profiling assays spanning large genomic regions

Suppose we have a bed file that represents chromatin profiling assays with ChIP-seq. Here, the data spans larger genomic regions than the aforementioned single-nucleotide examples. We will use one of the many publicly-available ChIP-seq dataset from the ENCODE project (The ENCODE Project Consortium 2012). While the below is just an example, you can replace it with your dataset of interest.

```
#!/bin/bash
wget -O chipseq_example.bed.gz https://www.encodeproject.org/files/ENCFF579UXQ/@download/ENCFF579UXQ.b
gunzip chipseq_example.bed.gz

head chipseq_example.bed
```

which should show the below as output.

chr1	10008	10200	.	0	.	0.719818	-1	-1	75
chr1	10360	10520	.	0	.	0.331552	-1	-1	75
chr1	16160	16320	.	0	.	0.597667	-1	-1	75
chr1	17430	17531	.	0	.	0.148326	-1	-1	75
chr1	29320	29417	.	0	.	0.209401	-1	-1	75
chr1	104940	105073	.	0	.	0.431891	-1	-1	75
chr1	180920	181200	.	0	.	0.462428	-1	-1	75
chr1	181380	181580	.	0	.	0.745993	-1	-1	75
chr1	183220	183380	.	0	.	0.287927	-1	-1	75
chr1	183700	183900	.	0	.	0.335915	-1	-1	75

KmeRtone requires you to have column headers for it to work. Therefore, we follow the same process as in the previous section.

```
library(data.table)
library(kmeRtone)

df <- data.table::fread("chipseq_example.bed")
df <- df[, .(V1, V2, V3)]
data.table::setnames(df, c("seqnames", "start", "end"))

print(head(df))
```

which should show the below as output.

	seqnames	start	end
	<char>	<int>	<int>
1:	chr1	10008	10200
2:	chr1	10360	10520
3:	chr1	16160	16320
4:	chr1	17430	17531
5:	chr1	29320	29417
6:	chr1	104940	105073

Again, we will split this data.table into chromosome-separated files. We will only focus on the 22 autosomes, but you can use the X and Y chromosomes too, if available.

```
dir.create(path = "./data", showWarnings = FALSE)
for(chr in 1:22){
  data.table::fwrite(
    df[seqnames == paste0("chr", chr)],
    paste0("./data/chr", chr, ".csv")
  )
}
```

To run a k-meric enrichment and depletion analysis with **kmeRtone** on this publicly-available dataset, run the below function. Here, we specified **k** as 4, thus, the z-scores are calculated for each 4-mer sequence.

The same negative control range as described previously applies here too. For now, we will stick to the default values. Please note, we set the **single.case.len** argument to *NULL* as these chromatin profiling assays tend to have variable widths.

Please refer to the documentation of the function for further details on the required and optional arguments. Please note, that your results will differ depending on which **ctrl.rel.pos** range you use, whether you have a specific **case.pattern** of interest, whether to remove overlapping case k-mers in case regions with **rm.case.kmer.overlaps**, whether to merge replicates or treat them separately with **merge.replicate**, and more.

```
#' 2. Run kmeRtone `score` function
kmeRtone::kmeRtone(
  case.coor.path="./data",
  genome.name="hg19",
  strand.sensitive=FALSE,
  k=4,
  ctrl.rel.pos=c(80, 500),
  case.pattern=NULL,
  single.case.len=NULL,
```

```

output.dir="output",
module="score",
rm.case.kmer.overlaps=FALSE,
merge.replicate=TRUE,
verbose=TRUE
)

```

The above should generate the below output. The results are saved in the path you specified in the `output.dir` argument.

```

-----
                        Extraction of Case K-mers
-----
Extracting 4-mers from chr1.....DONE! -- 3.62 secs
Extracting 4-mers from chr2.....DONE! -- 3.81 secs
Extracting 4-mers from chr3.....DONE! -- 2.99 secs
Extracting 4-mers from chr4.....DONE! -- 2.87 secs
Extracting 4-mers from chr5.....DONE! -- 2.67 secs
Extracting 4-mers from chr6.....DONE! -- 2.59 secs
Extracting 4-mers from chr7.....DONE! -- 2.37 secs
Extracting 4-mers from chr8.....DONE! -- 2.13 secs
Extracting 4-mers from chr9.....DONE! -- 2.03 secs
Extracting 4-mers from chr10.....DONE! -- 2.06 secs
Extracting 4-mers from chr11.....DONE! -- 2.09 secs
Extracting 4-mers from chr12.....DONE! -- 2.04 secs
Extracting 4-mers from chr13.....DONE! -- 1.6 secs
Extracting 4-mers from chr14.....DONE! -- 1.37 secs
Extracting 4-mers from chr15.....DONE! -- 1.37 secs
Extracting 4-mers from chr16.....DONE! -- 1.38 secs
Extracting 4-mers from chr17.....DONE! -- 1.2 secs
Extracting 4-mers from chr18.....DONE! -- 1.12 secs
Extracting 4-mers from chr19.....DONE! -- 0.91 secs
Extracting 4-mers from chr20.....DONE! -- 1.01 secs
Extracting 4-mers from chr21.....DONE! -- 0.57 secs
Extracting 4-mers from chr22.....DONE! -- 0.64 secs

Total time taken: 42.93 secs
-----
                        Extraction of Control K-mers
-----
Building control regions of chr1.....DONE! -- 3.78 secs
Building control regions of chr2.....DONE! -- 3.86 secs
Building control regions of chr3.....DONE! -- 3.17 secs
Building control regions of chr4.....DONE! -- 2.91 secs
Building control regions of chr5.....DONE! -- 2.8 secs
Building control regions of chr6.....DONE! -- 2.7 secs
Building control regions of chr7.....DONE! -- 2.54 secs
Building control regions of chr8.....DONE! -- 2.28 secs
Building control regions of chr9.....DONE! -- 2.2 secs
Building control regions of chr10.....DONE! -- 2.21 secs
Building control regions of chr11.....DONE! -- 2.28 secs
Building control regions of chr12.....DONE! -- 2.18 secs
Building control regions of chr13.....DONE! -- 1.61 secs
Building control regions of chr14.....DONE! -- 1.63 secs

```



```

Building control regions of chr15.....DONE! -- 1.47 secs
Building control regions of chr16.....DONE! -- 1.46 secs
Building control regions of chr17.....DONE! -- 1.37 secs
Building control regions of chr18.....DONE! -- 1.18 secs
Building control regions of chr19.....DONE! -- 1.08 secs
Building control regions of chr20.....DONE! -- 0.98 secs
Building control regions of chr21.....DONE! -- 0.67 secs
Building control regions of chr22.....DONE! -- 0.74 secs

```

```
Total time taken: 45.26 secs
```

```

Extracting 4-mers from chr1.....DONE! -- 4.76 secs
Extracting 4-mers from chr2.....DONE! -- 4.32 secs
Extracting 4-mers from chr3.....DONE! -- 3.63 secs
Extracting 4-mers from chr4.....DONE! -- 3.22 secs
Extracting 4-mers from chr5.....DONE! -- 3.28 secs
Extracting 4-mers from chr6.....DONE! -- 3.22 secs
Extracting 4-mers from chr7.....DONE! -- 2.94 secs
Extracting 4-mers from chr8.....DONE! -- 2.66 secs
Extracting 4-mers from chr9.....DONE! -- 2.45 secs
Extracting 4-mers from chr10.....DONE! -- 2.68 secs
Extracting 4-mers from chr11.....DONE! -- 2.7 secs
Extracting 4-mers from chr12.....DONE! -- 2.69 secs
Extracting 4-mers from chr13.....DONE! -- 1.74 secs
Extracting 4-mers from chr14.....DONE! -- 1.84 secs
Extracting 4-mers from chr15.....DONE! -- 1.65 secs
Extracting 4-mers from chr16.....DONE! -- 1.65 secs
Extracting 4-mers from chr17.....DONE! -- 1.83 secs
Extracting 4-mers from chr18.....DONE! -- 1.23 secs
Extracting 4-mers from chr19.....DONE! -- 1.27 secs
Extracting 4-mers from chr20.....DONE! -- 1.28 secs
Extracting 4-mers from chr21.....DONE! -- 0.79 secs
Extracting 4-mers from chr22.....DONE! -- 0.9 secs

```

```
Total time taken: 53.02 secs
```

#### ----- Calculation of K-mer Susceptibility -----

```
The 4-mer scores are saved at output/score_4-mer.csv
```

```
FINISH! Total time taken: 2.35 mins
```

## References

- Edgar, Ron, Michael Domrachev, and Alex E Lash. 2002. "Gene Expression Omnibus: NCBI Gene Expression and Hybridization Array Data Repository." *Nucleic Acids Research*, January. <https://doi.org/10.1093/nar/30.1.207>.
- Hu, Jinchuan, Ogun Adebali, Sheera Adar, and Aziz Sancar. 2017. "Dynamic Maps of UV Damage Formation and Repair for the Human Genome." *Proceedings of the National Academy of Sciences of the United States of America*, June. <https://doi.org/10.1073/pnas.1706522114>.
- The ENCODE Project Consortium. 2012. "An Integrated Encyclopedia of DNA Elements in the Human Genome." *Nature*, September. <https://doi.org/10.1038/nature11247>.