Vignette: Sasquatch in R

Ron Schweßinger, Maria Suciu, Simon McGowan, Jelena Telenius, Doug Higgs and Jim Hughes

31 Dezember 2015

Contents

1. Introduction

Insert tiny Introduction

2. Initialize

First of all, we *source* the Sasquatch functions distributed in *functions_sasq_r_utility.R*

```
source("/home/ron/fusessh/Sasquatch_offline/Sasquatch/R_utility/functions_sasq_r_utility.R")
```

We set some parameters like the fragmentation type to select the desired data. Here, we analyse DNase-seq data.

```
frag.type <- "DNase"
```

We also set the path to the database directory from where to read the preprocessed data, which depends on the fragmentation type and should contain subdirectories for every preprocessed or downloaded tissue.

```
data.dir <- file.path("/home/ron/fusessh/database_assembly/idx_correct_assembly/human",
    frag.type)

print(data.dir)
```

```
## [1] "/home/ron/fusessh/database_assembly/idx_correct_assembly/human/DNase"
```

3. Wrapper Functions like Web-Utility

The wrapper functions assemble the basic Sasquatch functions to mirror the web-tool functionality.

3.1. Workflow 1: Analyse Single *k*-mers

We first select the tissue which is expected to be a subdirectory in your data repository. Here we are interest in the whole lood, primary erythroid data.

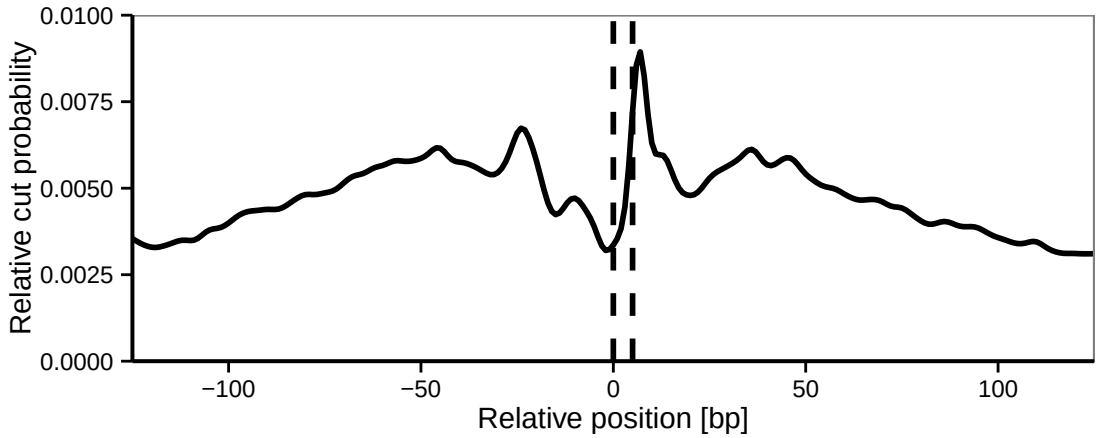
```
tissue <- "human_erythroid_hg18"
```

Plot profile of a single k-mer GATA-factors are prominent transcription factors in erythroid cells. They bind to the onsensus sequence of *WGATAA*. First, we plot the average profile of relative DNase I frequencies. To plot the average profile a selected *k*-mer, we use the *PlotSingleKmer* function. Sasquatch supports *k*-mers of length 5-7.

```
single.plot <- PlotSingleKmer(
  kmer="WGATAA",  #selected k-mer
  tissue=tissue,  #selecet tissue
  plot.shoulders=FALSE,  #decide if to estimate and plot the footprint shoulder regions
  data.dir=data.dir,  #path to data repository
  frag.type=frag.type,  #fragmentation type ["DNase" or "ATAC"]
  smooth=TRUE  #flag if to smooth the cut profile
  )
```

Note that we can query an ambivalent FASTA character that is decoded into *AGATAA* and *TGATAA* and Sasquatch retrieves the average of the corresponding profiles. Sasquatch can handle all FASTA characters (A, C, G, T, U, R, Y, K, M, S, W, B, D, H, V, N).

```
single.plot
```



To calculate the Shoulder-to-Footprint Ratio of a single k-mer To estimate the strength of a footprint we utilize the shoulder-to-footprint ratio (SFR). To calculate the SFR for a single *k*-mer we use the *GetSFR* function. Not

```
sfr <- GetSFR(
  kmer = "WGATAA",
  tissue = tissue,
  data.dir = data.dir,
  vocab.file = TRUE,  #flag if to use a vocabulary file
  frag.type = frag.type
  )

print(sfr)
```

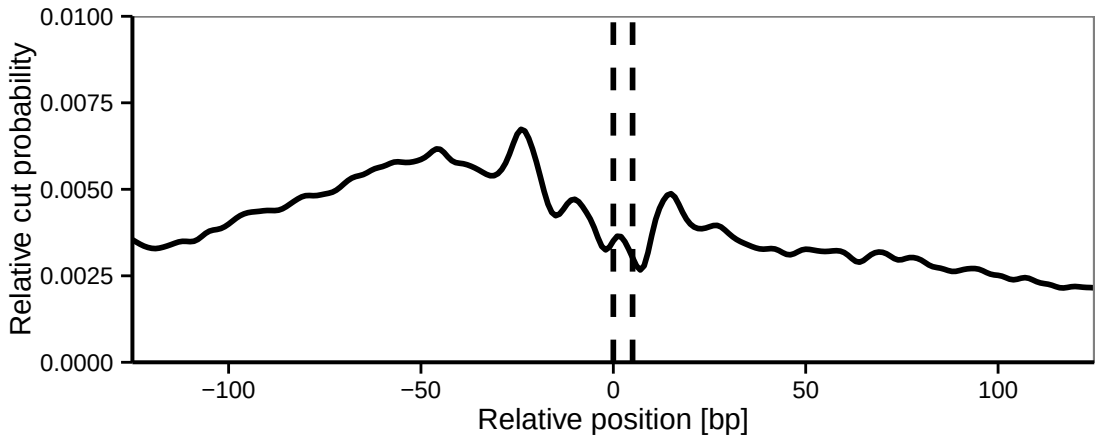
```
## [1] 1.638966
```

Note that we set the *vocab.flag = TRUE* to read the precalculated SFRs from the precalulated vocabulary file and speed the calculation time up dramatically. The default path to the vocabulary file is set to *vocab.file=paste0(data.dir,"/",tissue,"/vocabulary_",tissue,".txt")*. If no vocabulary file has been precalculated yet, set the flag to *FALSE* to run the calculation from the raw cut profile.

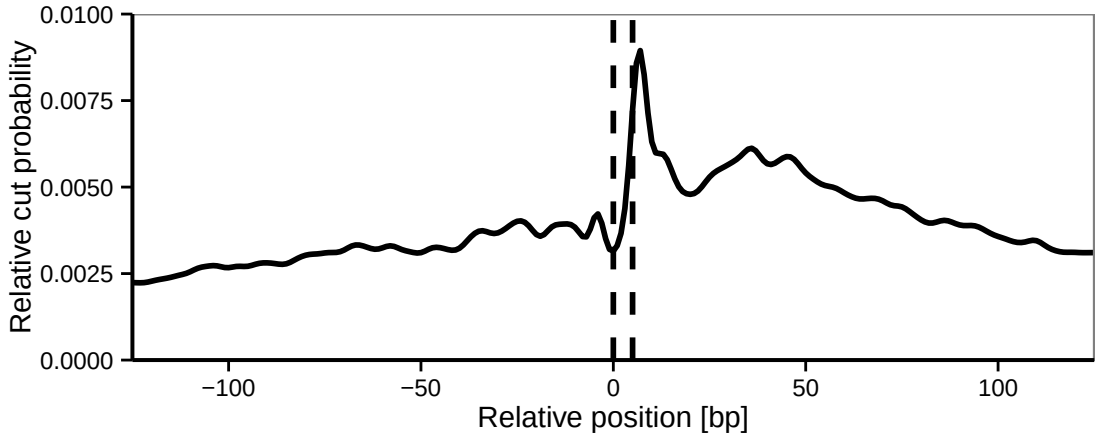
Plot strand-sprecific and background profiles of a single k-mer DNase I exhibits a strinking strand bias (**cite pyDNase paper**). To investigate that, we can also plot the average cut profiles separately for both DNA strands.

```
single.strands.plot <- PlotSingleStrands(
  kmer = "WGATAA",
  tissue = tissue,
  data.dir = data.dir,
  frag.type = frag.type,
  smooth=TRUE
  )
```

```
single.strands.plot$plot.plus
```




```
single.strands.plot$plot.minus
```



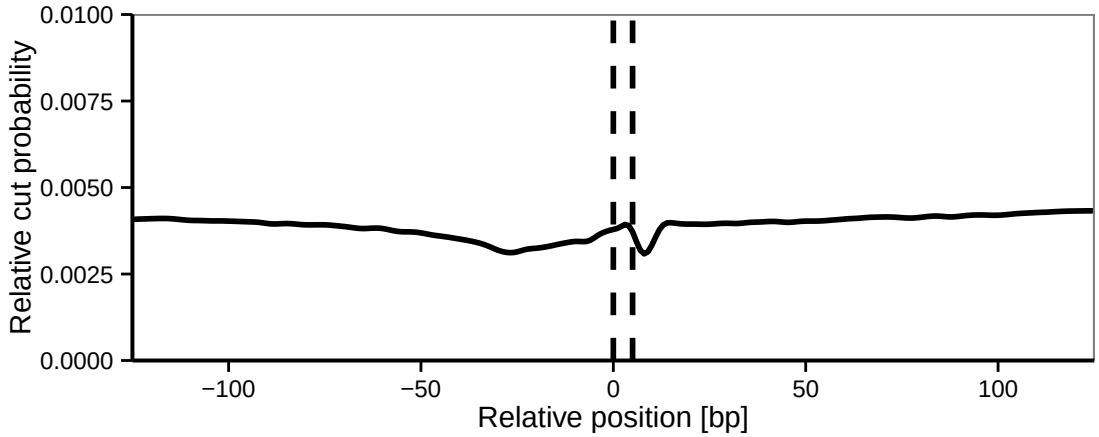
Note that the single average profiles are merged from both strands. The merging procedure differs for DNase-seq and ATAC-seq derived data. ATAC-seq profiles are merged by averaging both strand profiles. In contrast, DNase-seq profiles are merged with respect to the assay's strand imbalance, see manual/publication.

To rule out thath sequence bias or other biases affect the estimated footprint, we can plot the relative cut probabilities around the k-mer from the genome-wide background digestion assays. We can use the same function but set the *background.flag* = *TRUE* and define the path to the background repository and select the name of the genome-wide background experiment (*default* = *"hg18_human_JH60"*)

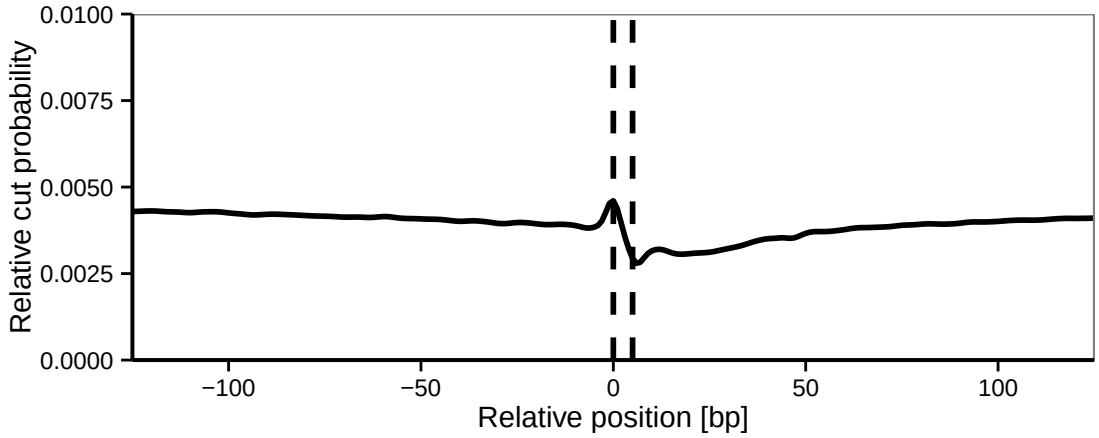
```
background.tissue <- "hg18_human_JH60"
background.dir <- "/home/ron/fusessh/database_assembly/idx_correct_assembly/background/"

background.plot <- PlotSingleStrands(
  kmer = "WGATAA",
  tissue = background.tissue,
  data.dir = background.dir,
  frag.type = frag.type,
  smooth = TRUE,
  background.flag = TRUE
  )
```

```
background.plot$plot.plus
```




```
background.plot$plot.minus
```



3.2. Workflow 2: Split and Analyse a Longer Sequence

Sasquatch works on a k-mer basis. Currently, it can handle k-mers of length 5 to 7. To analyse longer sequences we can scan the sequence with a sliding window. We use the *QueryLongSequence* function. The input sequence is split into k-mers of length kl and the SFR ratio of the corresponding average profile is calculated. The results are reported in a dataframe. If the plot flag is set *plot=TRUE* the function also retrieves a list of average profile plots, one for each splitted k-mer. Options for the plots can be set in the function call as well.

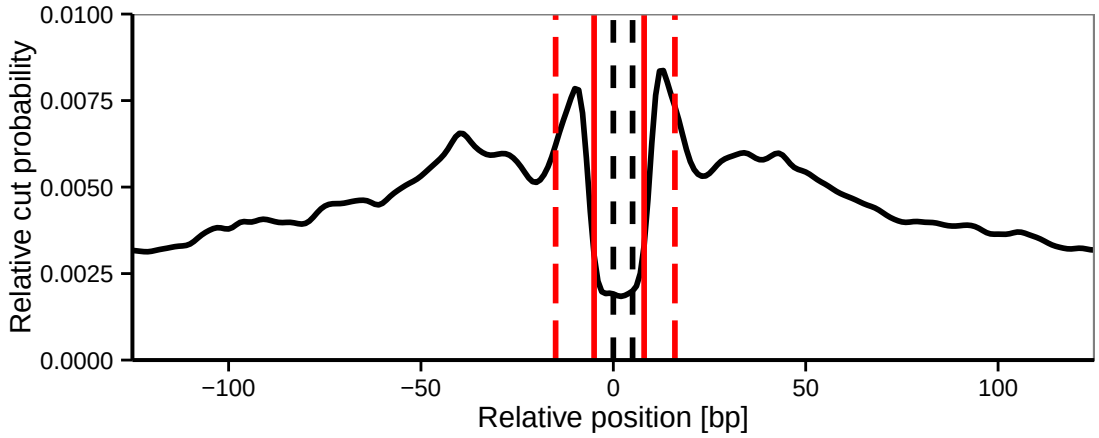
```
dissect.list <- QueryLongSequence(
  sequence="AGCACGTGTT",
  kl=6,
  tissue=tissue,
  data.dir=data.dir,
  vocab.flag=TRUE,
  frag.type=frag.type,
  plots=TRUE,
  smooth=TRUE,
  plot.shoulders=TRUE
  )

dissect.list$df
```

```
##     kmer      sfr
## 1 AGCACG 1.173508
## 2 GCACGT 1.369013
## 3 CACGTG 3.032849
## 4 ACGTGT 1.381040
```

We see that position 3 scores highest with a common E-box motif consensus sequence (*CACGTG*). We access the corresponding plot from the list.

```
dissect.list$plots[[3]]
```



3.3. Workflow 3: Compare Reference against Variance Sequences

We can use Sasquatch to estimate the impact of sequence variation e.g. from SNPs. The key idea of using Sasquatch for predicting the impact of sequence variation is to use the (altered) potential to produce strong footprints as proxy. To compare two sequences we use the *CompareSequences* function. Both input sequences are split up into *k*-mers of length *kl*. The *k*-mers are then compared pairwise by means of their SFRs. The damage is calculated as the difference between the reference SFR and the variant SFR (*reference - variant = damage*). A positive damage is thus associated with reducing the footprint characteristic, while a negative damage is associated with introducing or strengthening a footprint characteristic. The total damage is thatn calculates as the sum of all *k*-mer comparisons along the sequence pair (*damage.mode = "exhaustive"*) or as the highest damage from a single *k*-mer pair (*damage.mode = "local"*). The function retrieves a summary, a detailed data frame listing the pairwise comaprison and profile overlay plots. The default *plots="highest"* only reports the plot of the highest scoring *k*-mer pair.

For example, we estimate the impact of a SNP in an artificial GATA-site in a tissue specific manner.

```
compare.list <- CompareSequences(
  sequence1="ATAGATAATCGCT",  #reference sequence
  sequence2="ATAGATCATCGCT",  #variant sequence
  kl=6,
  damage.mode="exhaustive",  #mode to calcualte the overall damage
  tissue=tissue,
  data.dir=data.dir,
  vocab.flag=TRUE,
  frag.type=frag.type,
  plots="highest"  #which plots to report
  )

compare.list$summary
```

```
##    sequence.ref  sequence.var kmer.ref kmer.var  SFR.ref  SFR.var
## 1 ATAGATAATCGCT ATAGATCATCGCT   AGATAA   AGATCA 1.661522 1.062608
##   total.damage
## 1    0.8217635
```

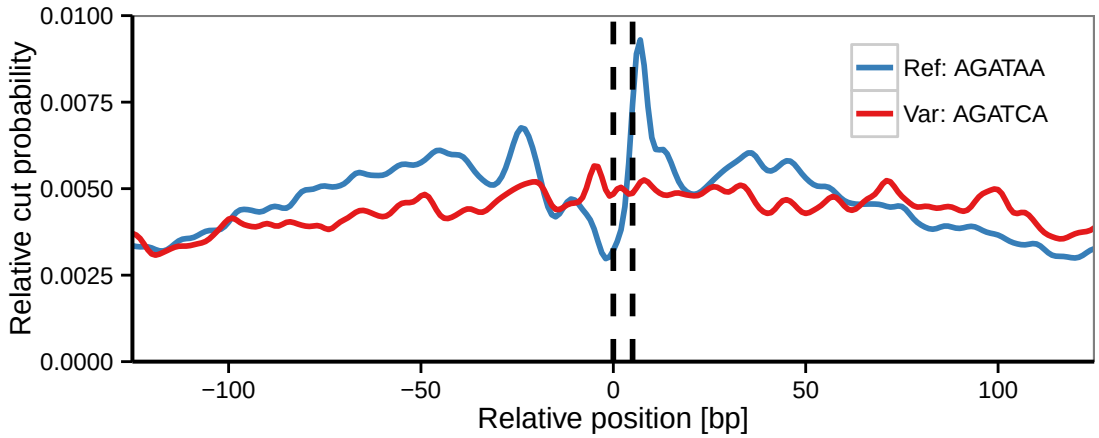
```
compare.list$df
```

```
##   kmer.ref kmer.var  sfr.ref  sfr.var      damage
```

```
## 1   ATAGAT   ATAGAT 1.203296 1.203296  0.00000000
## 2   TAGATA   TAGATC 1.424045 1.226321  0.19772430
## 3   AGATAA   AGATCA 1.661522 1.062608  0.59891405
## 4   GATAAT   GATCAT 1.279457 1.117357  0.16210047
## 5   ATAATC   ATCATC 1.189754 1.327197 -0.13744242
## 6   TAATCG   TCATCG 1.275166 1.342269 -0.06710328
## 7   AATCGC   CATCGC 1.251912 1.184341  0.06757042
```

To visualize the difference in the average footprints, we access the overlay plot of the highest k-mer pair.

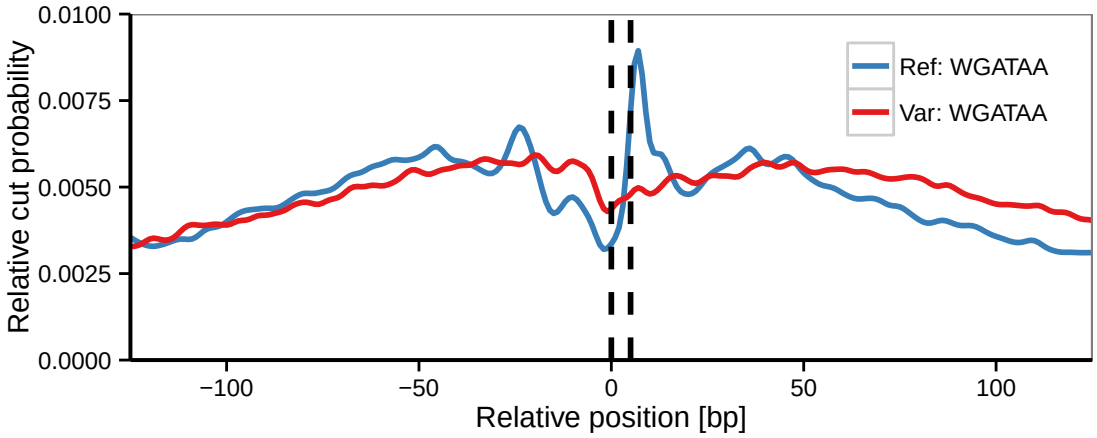
```
compare.list$plots
```



We can produce the same plot using the *PlotOverlapKmers* wrapper function. This function can also be used to quickly compare the footprinting potential of a single k-mer across different tissues. For example, we can compare the GATA-core motif in primary erythroid against ENCODE MCF7 data.

```
compare.plot <- PlotOverlapKmers(
  kmer1="WGATAA",
  kmer2="WGATAA",
  tissue1="human_erythroid_hg18",
  tissue2="ENCODE_MCF_7_UW_merged",
  data.dir=data.dir,
  frag.type="DNase",
  smooth=TRUE
  )
```

```
compare.plot
```



3.4. Workflow 4: Query Batches of Reference and Variant Sequence Pairs

To scan the damaging potential of multiple SNPs, we can can query pairs of reference and variant sequences as batch using the *RefVarBatch* function.

The input is a three column data frame *id ref var*. First we produce an examplary data frame.

```
batch.in <- data.frame(
        id=c("1", "2", "3"),
        ref=c("ATAGATAATCGCT", "ATAGATAATCGCT", "ATAGATAATCGCT"),
        var=c("ATAGATCATCGCT", "ATAGATTATCGCT", "ATAGATGATCGCT")
        )
```

Now we run the function. The result is an eight column data frame listing the id and sequences, the highest scoring k-mers and the reference and variant SFR with the tesulting damage. Note that this function specifically profits from a precalulated vocabuly file.

```
batch.results <- RefVarBatch(
  ref.var.df=batch.in,
  kl=6,
  damage.mode="exhaustive",
  tissue=tissue,
  data.dir=data.dir,
  vocab.flag=TRUE,
  frag.type=frag.type
  )

batch.results
```

```
##   id  sequence.ref  sequence.var kmer.ref kmer.var  SFR.ref  SFR.var
## 1  1 ATAGATAATCGCT ATAGATCATCGCT   AGATAA   AGATCA 1.661522 1.062608
## 2  2 ATAGATAATCGCT ATAGATTATCGCT   ATAATC   ATTATC 1.189754 1.475918
## 3  3 ATAGATAATCGCT ATAGATGATCGCT   AGATAA   AGATGA 1.661522 1.218602
##   total.damage
## 1    0.8217635
## 2   -0.1773051
## 3    0.2610159
```

To get a quick impression of potentially relevant transcription factors we use the JASPAR data base to query the short sequenes against transcription factor binding motifs. We first require the relevant packages.

```
library(Biostrings)
library(TFBSTools)
```

The relevant JASPAR motifs as PWMs of human and mouse were stored and distributed in Robjects. We load the relevant human object.

```
load("/home/ron/fusessh/database_assembly/jaspar/jaspar2014.human.9606.all.versions")
```

The *QueryJaspar* function scans an input sequences with all loaded PWMs and report the highest scoring factors with their relative matching score. For querying a whole Workflow 4 resulting dataframe we utilize the wrapper for the batch query *QueryJasparBatch*. Per default the sequence (reference or variant) with the highest SFR is queried.

```
batch.results.jaspar <- QueryJasparBatch(
  df=batch.results,  #data frame as result from Workflow 4
  damage.threshold=0.3,  #absolute damage threshold to query a SNP
  match.threshold=0.8,  #relative matching score to report a match
  pwm.data=human.pwm  #presaved JASPAR PWM object
  )

batch.results.jaspar
```

```
##   id  sequence.ref  sequence.var kmer.ref kmer.var  SFR.ref  SFR.var
## 1  1 ATAGATAATCGCT ATAGATCATCGCT   AGATAA   AGATCA 1.661522 1.062608
## 2  2 ATAGATAATCGCT ATAGATTATCGCT   ATAATC   ATTATC 1.189754 1.475918
## 3  3 ATAGATAATCGCT ATAGATGATCGCT   AGATAA   AGATGA 1.661522 1.218602
##   total.damage                          jaspar
## 1    0.8217635 GATA3=0.86;SRY=0.85;FOXL1=0.81;
## 2   -0.1773051                                .
## 3    0.2610159                                .
```

3.5. Workflow 5: Exhaustive *in silico* Mutation

Another popular task is to perform an exhaustive *in silico* mutation for a genomic sequence of interest by querying each possible base substitution at every base position. We can use that to find for example potential motifs that can be disrupted by various mutations or mutations that would introduce a k-mer with a strong footprint potential.

First, we load genome data and get the genomic sequence as a character string. Note that, for mutating on a sliding k-mer window basis we have to extract *kl-1* bases surrounding the sequence of interest.

```
library(BSgenome)
library(BSgenome.Hsapiens.UCSC.hg18)
genome <- BSgenome.Hsapiens.UCSC.hg18

seq <- as.character(getSeq(genome, "chr16", start=103489, end=103539))
```

Now we can query the *in silico* mutation function *InSilicoMutation*. The functions splits the sequence into windows of size *klx2-1*. For each window, the center base is mutated into all three possible bases.

Then the reference as well as the variant windows are analysed on a sliding *k*-mer bases. Per default (*damage.mode=exhaustive*), like in Workflow 3, the single SFRs ar summed up. The damage is calculated for each variant and reported according to the selected *report* mode. With the default *report="all"*, all three possible mutations are reported for each position. The resulting dataframe is the input for ploting the Rainbowplots. Alternatively, we can select to only report one possible mutation per position to retrieve a data frame that is straightforward to convert to bedGraph oder BigWig format for visualization in a genome browser. We can set *report=max* to only report the highes possible damage per posiion or *report=maxabs* to only report the highest absolute damage per position. Note that, like for the batch analysis, using the vocabluary file dramatiaćally speeds up the analysis here.

```
insilico.df <- InSilicoMutation(sequence=seq,
                          kl=7,  #select k-mer size
                          chr="chr16",  #select chromosome
                          position=103489,  #start coordinate of sequence
                          report="all",  #report mode
                          damage.mode="exhaustive",  #damage calculatin mode
                          tissue=tissue,
                          data.dir=data.dir,
                          vocab.flag=TRUE,
                          frag.type=frag.type
)
```

```
## [1] "Processing 117 sequence windows:"
```

```
head(insilico.df)
```

```
##     chr    pos ref.base var.base       ref.seq       var.seq      damage
## 1 chr16 103489        G        A GCCCTCGACCCTC GCCCTCAACCCTC -0.09473268
## 2 chr16 103489        G        C GCCCTCGACCCTC GCCCTCCACCCTC -0.43370636
## 3 chr16 103489        G        T GCCCTCGACCCTC GCCCTCTACCCTC -2.55384750
## 4 chr16 103490        A        C CCCTCGACCCTCT CCCTCGCCCCTCT -2.41872565
## 5 chr16 103490        A        G CCCTCGACCCTCT CCCTCGGCCCTCT  0.01863947
## 6 chr16 103490        A        T CCCTCGACCCTCT CCCTCGTCCCTCT -0.11988915
```

Note that the mutation is implemented as apply function and can easily be parallelized. To monitor progress we recommend the *pbapply* package.

To visualize the results in a plot we parse the dataframe into the *RainbowPlot* function.

```
rainbow.plot <- RainbowPlot(insilico.df)
```

```
rainbow.plot
```