Tests for local alignments and DNA shape

Uploadings

Load libraries

```
#library(Biostrings)
library(ggplot2)
library(DNAshapeR)
projpath <- "/home/davidfm/Projects/UBMI-IFC/EnhaProm/"
```

Load functions and libraries

```
setwd(projpath)
source("scripts/genome-functions.R")
```

DNA shape metrics

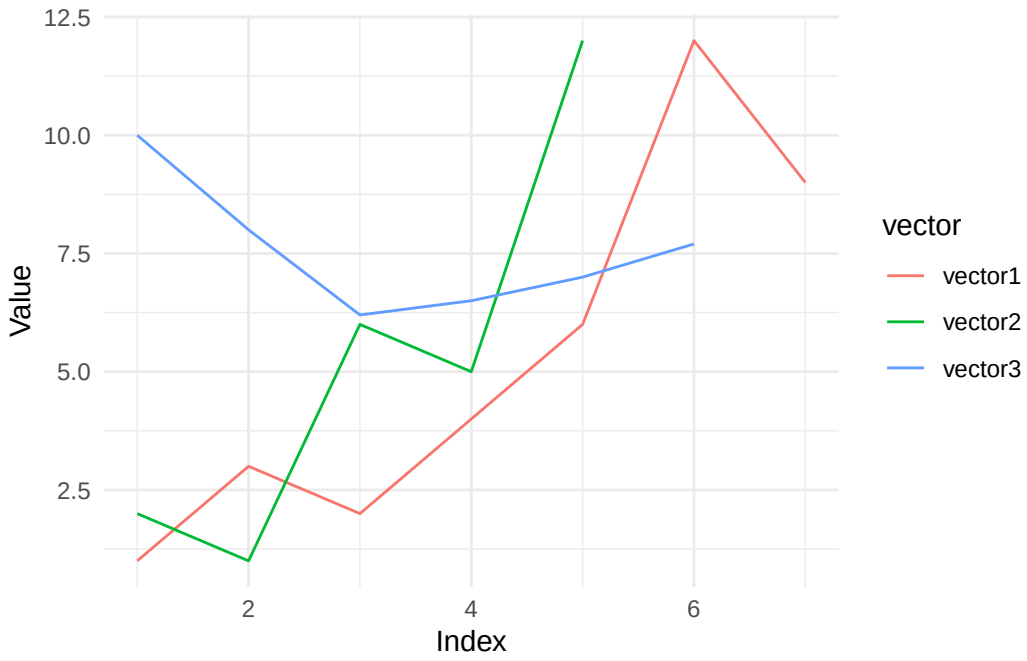
```
#shape_promoters <- getShape("datasets/testing/promoters_train_positive.fasta", shapeType= c
#getShape("datasets/testing/promoters_train_positive.fasta",
#         shapeType= c("Stretch", "Tilt", "Buckle", "Shear"))
#getShape("datasets/testing/promoters_train_positive.fasta")
#getShape("datasets/testing/promoters_train_positive.fasta", parse=FALSE)
# setwd(paste0(projpath, "liltests/"))
setwd(projpath)
shape_seqs <- getShape("datasets/GB-Testing/10seqs.fasta")
```

```
Reading the input sequence......
Reading the input sequence......
Reading the input sequence......
Reading the input sequence......
Reading the input sequence......

Parsing files......

Record length: 505

Warning in rbind(`1` = c(NA, NA, NA, NA, NA, NA, NA, 4.67, 4.92, 4.63, 4.92, :
number of columns of result is not a multiple of vector length (arg 1)

Record length: 504

Warning in rbind(`1` = c(NA, NA, NA, NA, NA, NA, 36.61, 31.5, 36.01, 34.17, :
number of columns of result is not a multiple of vector length (arg 1)

Record length: 505

Warning in rbind(`1` = c(NA, NA, NA, NA, NA, NA, NA, -2.4, -6.1, -4.81, :
number of columns of result is not a multiple of vector length (arg 1)

Record length: 504

Warning in rbind(`1` = c(NA, NA, NA, NA, NA, NA, -1.63, -2.51, -2.33, -1.19, :
number of columns of result is not a multiple of vector length (arg 1)

Record length: 505

Warning in rbind(`1` = c(NA, NA, NA, NA, NA, NA, NA, -6.16, -7.26, -5.59, :
number of columns of result is not a multiple of vector length (arg 1)

Done
```

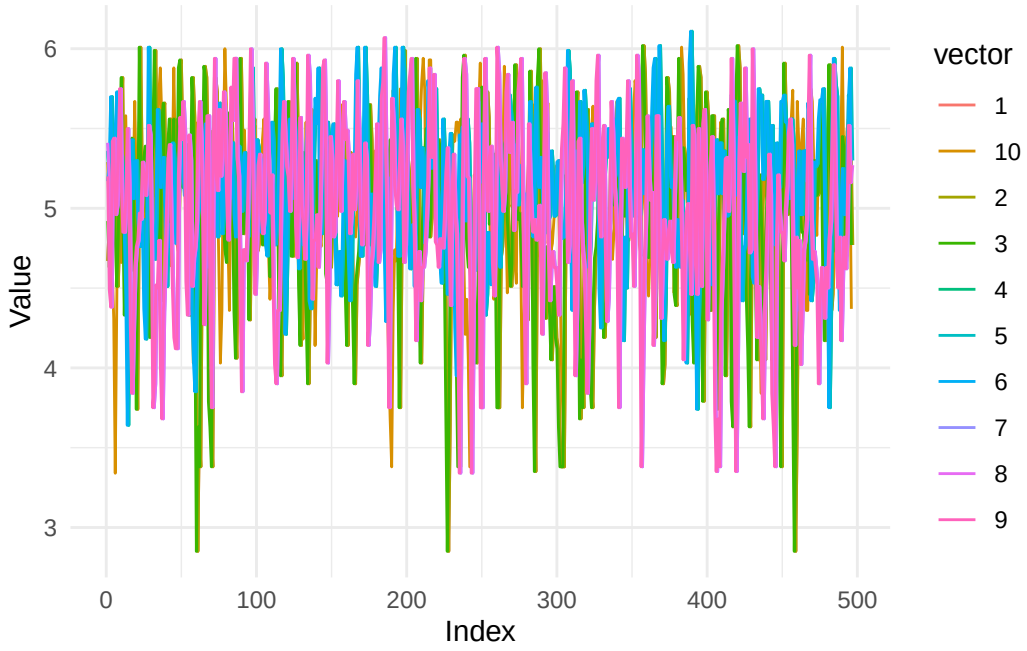
```r
# Change matrix rows into list of vectors
ss_mgw <- t(shape_seqs$MGW)
small_ <- t(shape_seqs$MGW[c(3,5,8),200:300])
#         ^If not transposed, this becomes a mess
# ss_mgw[c(1:10,(nrow(ss_mgw)-10):nrow(ss_mgw)),1:3]
split_factor1 <- rep(1:ncol(ss_mgw), each = nrow(ss_mgw))
split_factor2 <- rep(1:ncol(small_), each = nrow(small_))

ss_mgw_list <- split(ss_mgw, split_factor1)
ss_mgw_list <- lapply(ss_mgw_list, na.omit)
small__list <- split(small_, split_factor2)
# ^There are no NAs in this list since no 'border' columns are considerated
any(is.na(small_))
```

```
[1] FALSE
```

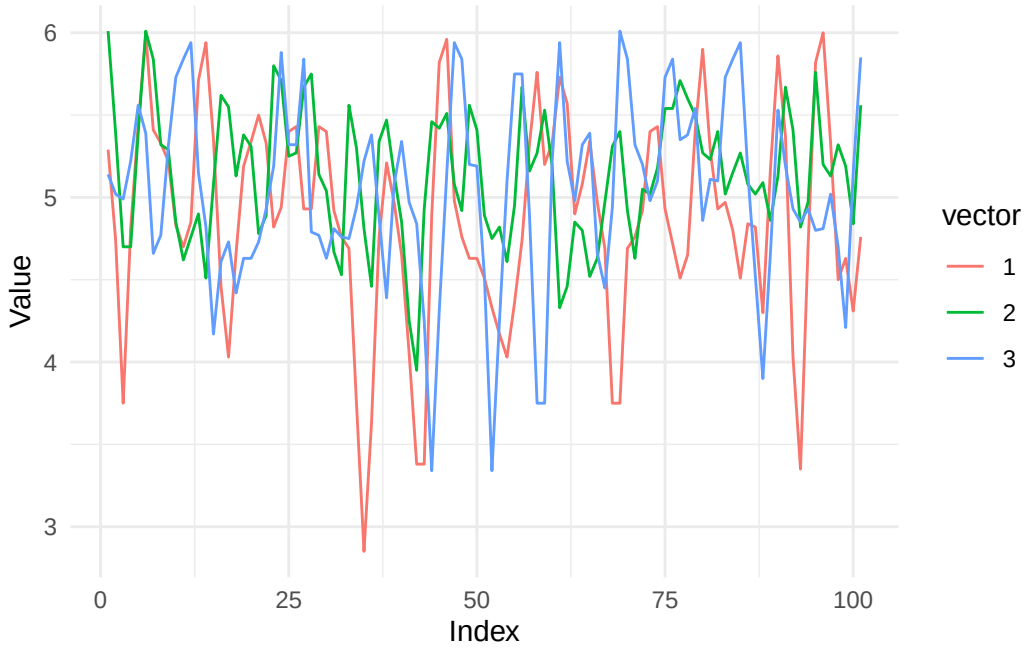
```r
# Plot list of vectors
linesplot <- function(vector_list) {
# Create a data frame to hold the data
  df <- do.call(rbind, lapply(vector_list,
                              function(x) data.frame(x = seq_along(x), y = x)))
  df$vector <- rep(names(vector_list), sapply(vector_list, length))

  # Create the plot
  ggplot(df, aes(x = x, y = y, color = vector)) +
    geom_line() +
    labs(x = "Index", y = "Value") +
    theme_minimal()
}

# Sample data: a list of vectors
vector_list <- list(
  vector1 = c(1, 3, 2, 4, 6, 12, 9),
  vector2 = c(2, 1, 6, 5, 12),
  vector3 = c(10, 8, 6.2, 6.5, 7, 7.7)
)

linesplot(vector_list)
```




```
linesplot(ss_mgw_list)
```



```
linesplot(small__list)
```

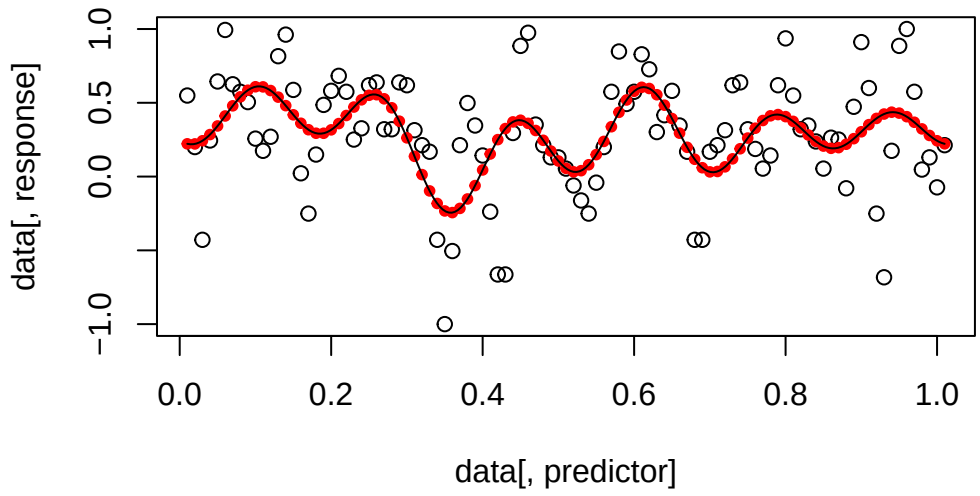


Regressions

```
## Shape data as dataframe per sequence
seqshape_df <- function(seqshape) {
  # Normalization
  #: norm_val <- max(c(max(seqshape), abs(min(seqshape))))
  #: norm_seqshape <- seqshape / norm_val
  norm_seqshape <- 2 * (seqshape - min(seqshape)) / (max(seqshape) - min(seqshape)) - 1

  # Positions in 'relative' values
  rel_positions <- seq_along(seqshape) / 100

  return(data.frame(mgw_values = norm_seqshape,
                    positions = rel_positions))
}
```

```r
# Example shape data
sshape3 <- shape_seqs$MGW[3,200:300]
sshape5 <- shape_seqs$MGW[5,200:300]
sshape8 <- shape_seqs$MGW[8,200:300]

trunc_fourier(seqshape_df(sshape3), "mgw_values", "positions",
              n_peaks=7, aem_option=1, bottomtop=TRUE,
              silent=TRUE, check_again=TRUE, only_coefficients=FALSE)
```

```
         1         2 3         4 5           6 7 8 9         10 11         12 13
l_coefs 0 0.1007206 0 0.1217824 0 -0.08472715 0 0 0 0.08941141  0 -0.197114  0
s_coefs 0 0.0000000 0 0.5000000 0  0.25000000 0 0 0 0.40000000  0  0.300000  0
         14
l_coefs   0
s_coefs   0
```

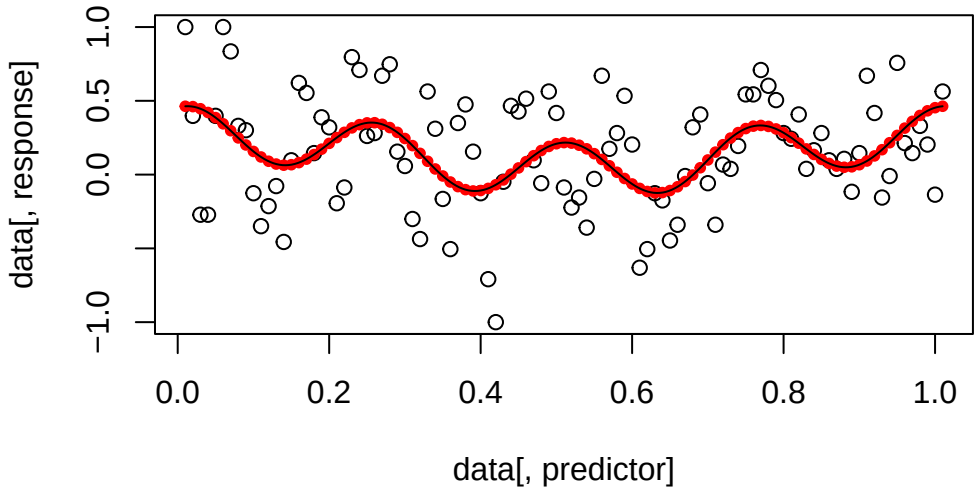



```
Call:
lm(formula = as.formula(f_i), data = data)

Coefficients:
```

```
                        (Intercept)         cos(pi * ((2 * positions)))
                            0.29007                             0.10072
 cos(pi * ((4 * positions) - 0.5))  cos(pi * ((6 * positions) - 0.25))
                            0.12178                            -0.08473
cos(pi * ((10 * positions) - 0.4))  cos(pi * ((12 * positions) - 0.3))
                            0.08941                            -0.19711
```

```
trunc_fourier(seqshape_df(sshape5), "mgw_values", "positions",
              n_peaks=7, aem_option=1, bottomtop=TRUE,
              silent=TRUE, check_again=TRUE, only_coefficients=FALSE)
```

```
         1         2 3 4 5 6 7         8 9 10 11 12 13 14
l_coefs 0 0.1237253 0 0 0 0 0 0.1848824 0  0  0  0  0  0
s_coefs 0 0.0500000 0 0 0 0 0 0.1000000 0  0  0  0  0  0
```

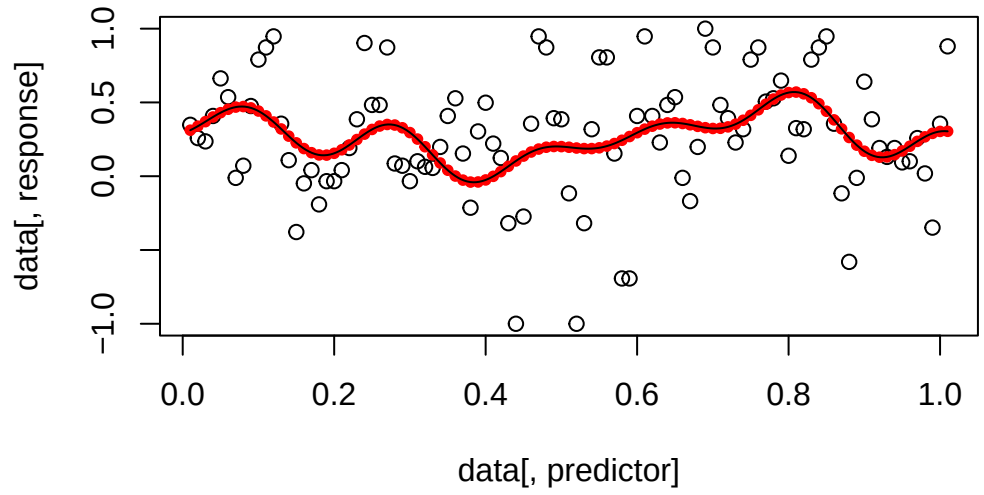



```
Call:
lm(formula = as.formula(f_i), data = data)

Coefficients:
                        (Intercept)  cos(pi * ((2 * positions) - 0.05))
```

```
                            0.1555                                  0.1237
 cos(pi * ((8 * positions) - 0.1))
                            0.1849
```

```
trunc_fourier(seqshape_df(sshape8), "mgw_values", "positions",
              n_peaks=7, aem_option=1, bottomtop=TRUE,
              silent=TRUE, check_again=TRUE, only_coefficients=FALSE)
```

```
         1           2         3 4 5 6 7          8 9 10          11 12 13 14
l_coefs 0 -0.05255758 0.1358541 0 0 0 0 0.07942362 0  0 -0.08769285  0  0  0
s_coefs 0  0.50000000 0.3000000 0 0 0 0 0.30000000 0  0  0.05000000  0  0  0
```




```
Call:
lm(formula = as.formula(f_i), data = data)

Coefficients:
                        (Intercept)    cos(pi * ((2 * positions) - 0.5))
                            0.25014                              -0.05256
  cos(pi * ((3 * positions) - 0.3))    cos(pi * ((8 * positions) - 0.3))
                            0.13585                               0.07942
```

```
cos(pi * ((11 * positions) - 0.05))
                           -0.08769
```

```
testseq1 <- paste0("TTCTCCTGCCTTAGCCTCCCAAGTCACTGGGATTACAGGTGCCCACCACCATACCAGGCTAA",
                   "TTTTTGTATTTTTAGTGGAGATGCGGTTTCACCATGTTGGCCGGGCCAGTCTCGAACTCCTGACGTC",
                   "AAGTGATCTTCCCGCCTCGACTCCTGATATCAAGTGATCTTCCCGCCTCGGCCTCCCAGAGTGCTGA",
                   "GATTACAGACGTGAACCCATGCCTGGCCAGGAATTTTGTTTTTTAGGAAGGCTTTCTACTAATGGAA",
                   "TTCCTGGCCTTGAGAGGATGTTACTTTAGAAGGAAAGGATTTTTTTGTTATTAAAAGGTAAGATTCC",
                   "TGGATTCTTATTGGACTGTTGTCTCTGTTATGAGTAATCCATCTTTAGTCATTCACCACTAGGGTTG",
                   "TATTTAATTAAGTCTGAGTTATTTTATGGTGATTTTGTTTTGTTTTGTTTTGTTTTTACCGAATTTT",
                   "GTTCTCATTGCCGTGGCTTGAGGGCAATGACGTGAT") # Pasted so the output don't overfl
testseq2 <- "CAAGCTTGTGCAGTGTTGCTGTTCTATCTCGGCTCACTGCAAGCTGTTCACGCCATGTTCTGTT"
#sort(sapply(ls()[grep("seq", ls())], function(x) object.size(get(x))))
```

Get-Interspaces

(This was practically already done) This functions measures all the spaces in between a given 'kmer', the produced data can either be represented 'as-is' or in terms of position relative to sequence length.

```
setwd(projpath)
source("scripts/trash-code.R")
source("scripts/interpolation-functions.R")
get_interspaces(testseq1, k = 3, kmer = "CTC", print = TRUE)
```

```
 [1]   0   5  18 115 121 147 152 178 184 355 470
 [1]   2  15 112 118 144 149 175 181 352 467 500
InterSpaces:
lengths: 2 10 94 3 23 2 23 3 168 112 30
positions: 1 10 65 116.5 132.5 148 163.5 179.5 268 411 485
```

```
   lengths positions
1        2       1.0
2       10      10.0
3       94      65.0
4        3     116.5
5       23     132.5
6        2     148.0
7       23     163.5
8        3     179.5
```

```
9      168     268.0
10     112     411.0
11      30     485.0
```

```
get_interspaces(testseq1, k = 3, kmer = "CTC", relative = TRUE, print = TRUE)
```

```
 [1]   0   5  18 115 121 147 152 178 184 355 470
 [1]   2  15 112 118 144 149 175 181 352 467 500
InterSpaces:
lengths: 0.004 0.02 0.188 0.006 0.046 0.004 0.046 0.006 0.336 0.224 0.06
positions: 0.002 0.02 0.13 0.233 0.265 0.296 0.327 0.359 0.536 0.822 0.97
```

```
   lengths positions
1    0.004     0.002
2    0.020     0.020
3    0.188     0.130
4    0.006     0.233
5    0.046     0.265
6    0.004     0.296
7    0.046     0.327
8    0.006     0.359
9    0.336     0.536
10   0.224     0.822
11   0.060     0.970
```

Kmer-Interspaces

(This function was already done and moved to 'trash-code' since it wasn't of use at the moment. Its' only modification was moving interspaces' computation to an individual function an adding the 'acc_peaks' ("accentuate peaks") and 'rel' ("relative") paramters). This function used the previous one and adjusts the data to many trigonometric linear models from which it chooses the 'best' one.

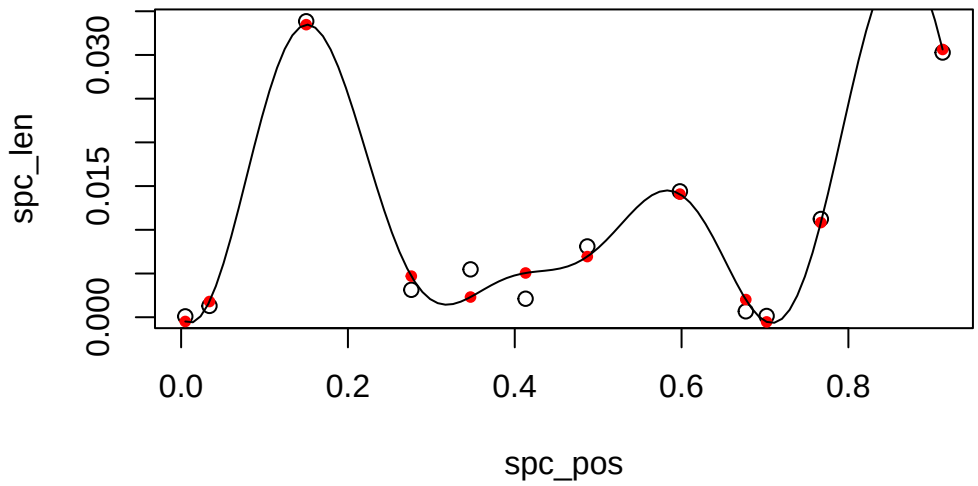
```
kmer_interspace_polynome(testseq1, kmer="CTG", aem_option=5, acc_peaks=TRUE)
```

```
 [1]   0   8  29 124 155 195 221 269 332 348 357 413
 [1]   5  26 121 152 192 218 266 329 345 354 410 500
InterSpaces:
lengths: 0.01 0.036 0.184 0.056 0.074 0.046 0.09 0.12 0.026 0.012 0.106 0.174
positions: 0.005 0.034 0.15 0.276 0.347 0.413 0.487 0.598 0.677 0.702 0.767 0.913
```

```
 Best Case chosen:  7 - 1


Call:
lm(formula = spc_len ~ cos(2 * pi * spc_pos) + sin(2 * pi * spc_pos) +
    cos(4 * pi * spc_pos) + sin(4 * pi * spc_pos) + cos(6 * pi *
    spc_pos) + sin(6 * pi * spc_pos) + cos(8 * pi * spc_pos))

Coefficients:
          (Intercept)  cos(2 * pi * spc_pos)  sin(2 * pi * spc_pos)
            0.0137238              0.0079268             -0.0017201
cos(4 * pi * spc_pos)  sin(4 * pi * spc_pos)  cos(6 * pi * spc_pos)
           -0.0019572             -0.0005418             -0.0119401
sin(6 * pi * spc_pos)  cos(8 * pi * spc_pos)
           -0.0043036             -0.0078595
```

```
kmer_interspace_polynome(testseq1, kmer="CTG", aem_option=7, acc_peaks=TRUE)
```

```
 [1]   0   8  29 124 155 195 221 269 332 348 357 413
 [1]   5  26 121 152 192 218 266 329 345 354 410 500
InterSpaces:
lengths: 0.01 0.036 0.184 0.056 0.074 0.046 0.09 0.12 0.026 0.012 0.106 0.174
positions: 0.005 0.034 0.15 0.276 0.347 0.413 0.487 0.598 0.677 0.702 0.767 0.913

 Best Case chosen:  7 - 1
```




```
Call:
lm(formula = spc_len ~ cos(2 * pi * spc_pos) + sin(2 * pi * spc_pos) +
    cos(4 * pi * spc_pos) + sin(4 * pi * spc_pos) + cos(6 * pi *
    spc_pos) + sin(6 * pi * spc_pos) + cos(8 * pi * spc_pos))

Coefficients:
          (Intercept)  cos(2 * pi * spc_pos)  sin(2 * pi * spc_pos)
            0.0137238              0.0079268             -0.0017201
cos(4 * pi * spc_pos)  sin(4 * pi * spc_pos)  cos(6 * pi * spc_pos)
           -0.0019572             -0.0005418             -0.0119401
sin(6 * pi * spc_pos)  cos(8 * pi * spc_pos)
           -0.0043036             -0.0078595
```

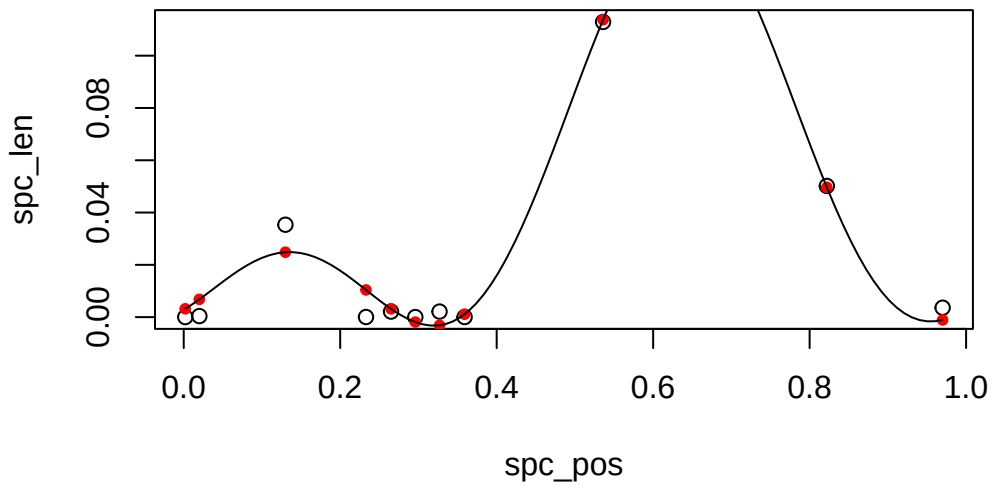
```
kmer_interspace_polynome(testseq1, kmer="CTC", aem_option=7, acc_peaks=TRUE)
```

```
 [1]   0   5  18 115 121 147 152 178 184 355 470
 [1]   2  15 112 118 144 149 175 181 352 467 500
InterSpaces:
lengths: 0.004 0.02 0.188 0.006 0.046 0.004 0.046 0.006 0.336 0.224 0.06
positions: 0.002 0.02 0.13 0.233 0.265 0.296 0.327 0.359 0.536 0.822 0.97
```

```
 Best Case chosen:  4 - 1


Call:
lm(formula = spc_len ~ cos(2 * pi * spc_pos) + sin(2 * pi * spc_pos) +
    cos(4 * pi * spc_pos) + sin(4 * pi * spc_pos))

Coefficients:
          (Intercept)  cos(2 * pi * spc_pos)  sin(2 * pi * spc_pos)
              0.05025               -0.04161               -0.04964
cos(4 * pi * spc_pos)  sin(4 * pi * spc_pos)
             -0.00581                0.03897
```

```r
kmer_interspace_polynome(testseq1, kmer="CTC", aem_option=8, acc_peaks=TRUE)
```

```
 [1]   0   5  18 115 121 147 152 178 184 355 470
 [1]   2  15 112 118 144 149 175 181 352 467 500
InterSpaces:
lengths: 0.004 0.02 0.188 0.006 0.046 0.004 0.046 0.006 0.336 0.224 0.06
positions: 0.002 0.02 0.13 0.233 0.265 0.296 0.327 0.359 0.536 0.822 0.97

 Best Case chosen:  4 - 1
```



```
Call:
lm(formula = spc_len ~ cos(2 * pi * spc_pos) + sin(2 * pi * spc_pos) +
    cos(4 * pi * spc_pos) + sin(4 * pi * spc_pos))

Coefficients:
          (Intercept)  cos(2 * pi * spc_pos)  sin(2 * pi * spc_pos)
              0.05025               -0.04161               -0.04964
cos(4 * pi * spc_pos)  sin(4 * pi * spc_pos)
             -0.00581                0.03897
```

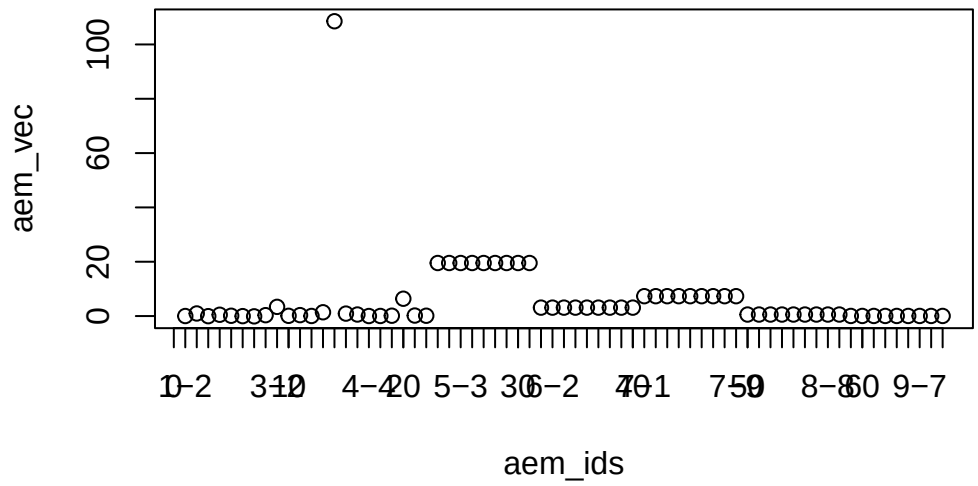
At a certain point I wanted to visualize the process of model selection so I added the parameter 'silent' (Since the output is really long, we won't evaluate the following cell)

```
kmer_interspace_polynome(testseq1, kmer="CTC", aem_option=7, acc_peaks=TRUE, silent=FALSE)
kmer_interspace_polynome(testseq1, kmer="CTC", aem_option=8, acc_peaks=TRUE, silent=FALSE)
```

To visualize this in a plot I added the paramter 'aem_graph' ('AEM' standing for 'Adjust-Error Metric')

```
kmer_interspace_polynome(testseq1, kmer="CTC", silent=TRUE, aem_option=5, acc_peaks=TRUE, aem
```

```
 [1]   0   5  18 115 121 147 152 178 184 355 470
 [1]   2  15 112 118 144 149 175 181 352 467 500
InterSpaces:
lengths: 0.004 0.02 0.188 0.006 0.046 0.004 0.046 0.006 0.336 0.224 0.06
positions: 0.002 0.02 0.13 0.233 0.265 0.296 0.327 0.359 0.536 0.822 0.97
```

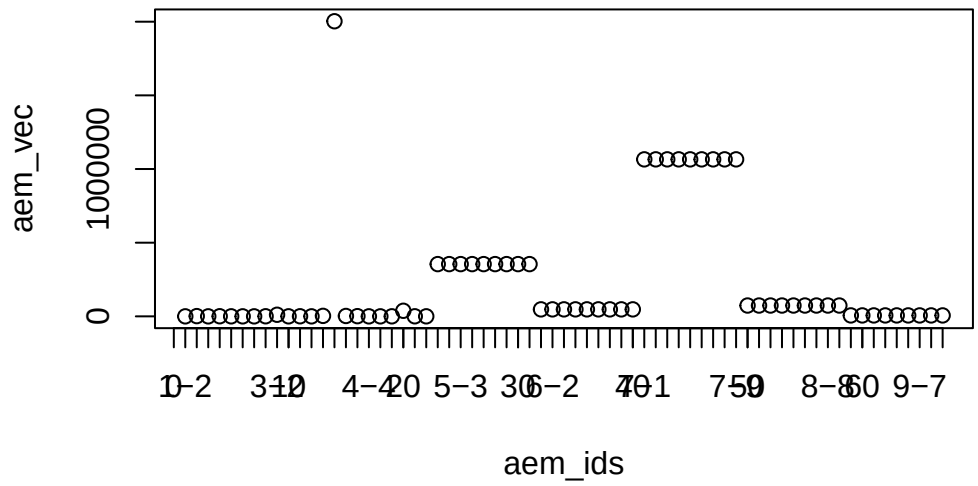



```
 Best Case chosen:  4 - 1
```

```
Call:
lm(formula = spc_len ~ cos(2 * pi * spc_pos) + sin(2 * pi * spc_pos) +
    cos(4 * pi * spc_pos) + sin(4 * pi * spc_pos))

Coefficients:
          (Intercept)  cos(2 * pi * spc_pos)  sin(2 * pi * spc_pos)
              0.05025               -0.04161               -0.04964
cos(4 * pi * spc_pos)  sin(4 * pi * spc_pos)
             -0.00581                0.03897
```

```
kmer_interspace_polynome(testseq1, kmer="CTC", silent=TRUE, aem_option=7, acc_peaks=TRUE, aem
```

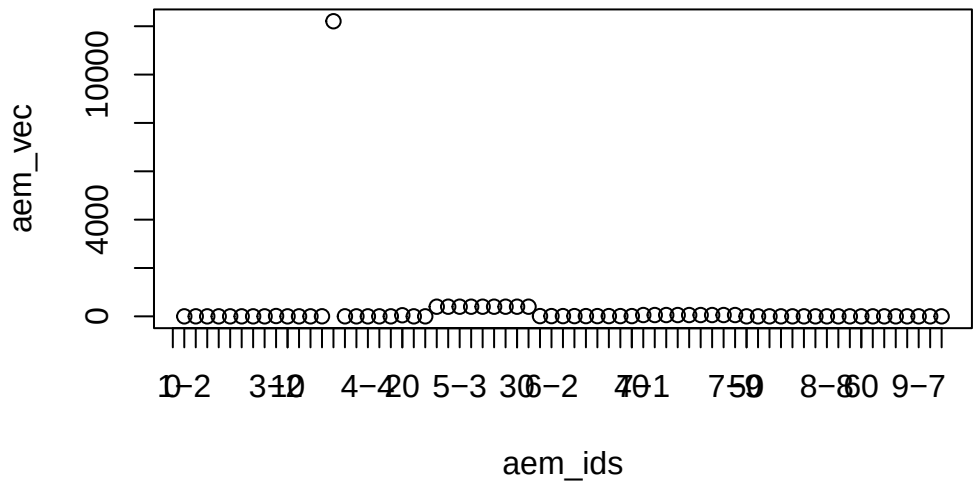
```
 [1]   0   5  18 115 121 147 152 178 184 355 470
 [1]   2  15 112 118 144 149 175 181 352 467 500
InterSpaces:
lengths: 0.004 0.02 0.188 0.006 0.046 0.004 0.046 0.006 0.336 0.224 0.06
positions: 0.002 0.02 0.13 0.233 0.265 0.296 0.327 0.359 0.536 0.822 0.97
```




```
 Best Case chosen:  4 - 1


Call:
lm(formula = spc_len ~ cos(2 * pi * spc_pos) + sin(2 * pi * spc_pos) +
    cos(4 * pi * spc_pos) + sin(4 * pi * spc_pos))

Coefficients:
          (Intercept)  cos(2 * pi * spc_pos)  sin(2 * pi * spc_pos)
              0.05025               -0.04161               -0.04964
cos(4 * pi * spc_pos)  sin(4 * pi * spc_pos)
             -0.00581                0.03897
```

```
kmer_interspace_polynome(testseq1, kmer="CTC", silent=TRUE, aem_option=8, acc_peaks=TRUE, aem
```

```
 [1]   0   5  18 115 121 147 152 178 184 355 470
 [1]   2  15 112 118 144 149 175 181 352 467 500
InterSpaces:
lengths: 0.004 0.02 0.188 0.006 0.046 0.004 0.046 0.006 0.336 0.224 0.06
positions: 0.002 0.02 0.13 0.233 0.265 0.296 0.327 0.359 0.536 0.822 0.97
```




```
 Best Case chosen:  4 - 1
```

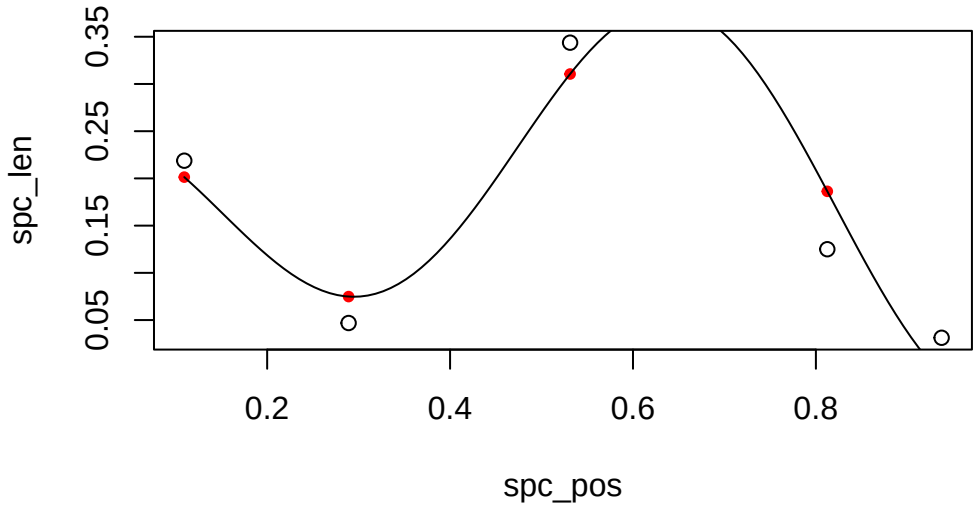
```
Call:
lm(formula = spc_len ~ cos(2 * pi * spc_pos) + sin(2 * pi * spc_pos) +
    cos(4 * pi * spc_pos) + sin(4 * pi * spc_pos))

Coefficients:
          (Intercept)  cos(2 * pi * spc_pos)  sin(2 * pi * spc_pos)
              0.05025               -0.04161               -0.04964
cos(4 * pi * spc_pos)  sin(4 * pi * spc_pos)
             -0.00581                0.03897
```

In case I wanted to watch a spacific trigonometric case, I added the 'fixed' parameter:

```
kmer_interspace_polynome(testseq2, kmer="GTT", silent=TRUE, fixed=TRUE, case=2, subcase=10)
```

```
[1]  0 17 23 48 59 64
[1] 14 20 45 56 61 64
InterSpaces:
```

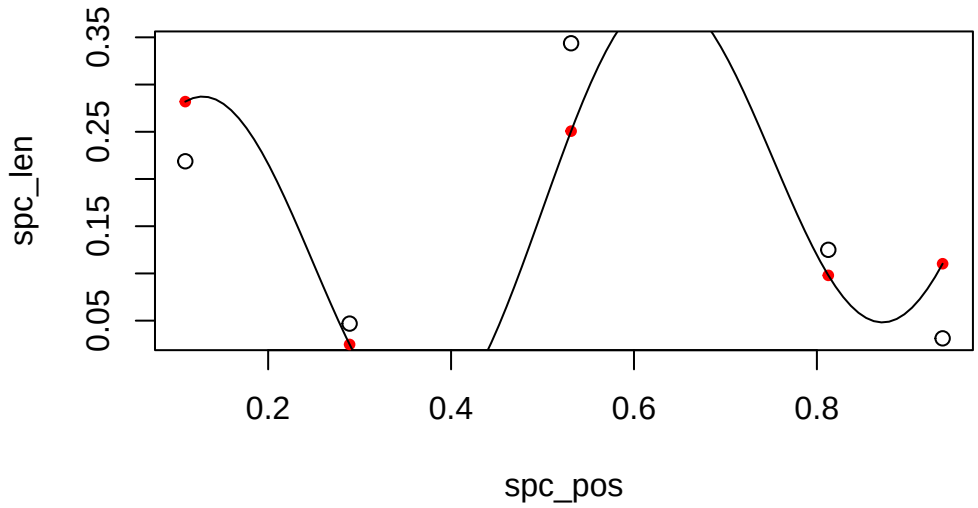
```
lengths: 0.21875 0.046875 0.34375 0.125 0.03125
positions: 0.109375 0.2890625 0.53125 0.8125 0.9375
```




```
Call:
lm(formula = spc_len ~ cos(2 * pi * spc_pos) + cos(3 * pi * spc_pos))

Coefficients:
          (Intercept)  cos(2 * pi * spc_pos)  cos(3 * pi * spc_pos)
              0.18878               -0.08092                0.14612
```

```
kmer_interspace_polynome(testseq2, kmer="GTT", silent=TRUE, fixed=TRUE, case=2, subcase=8)
```

```
[1]  0 17 23 48 59 64
[1] 14 20 45 56 61 64
InterSpaces:
lengths: 0.21875 0.046875 0.34375 0.125 0.03125
positions: 0.109375 0.2890625 0.53125 0.8125 0.9375
```



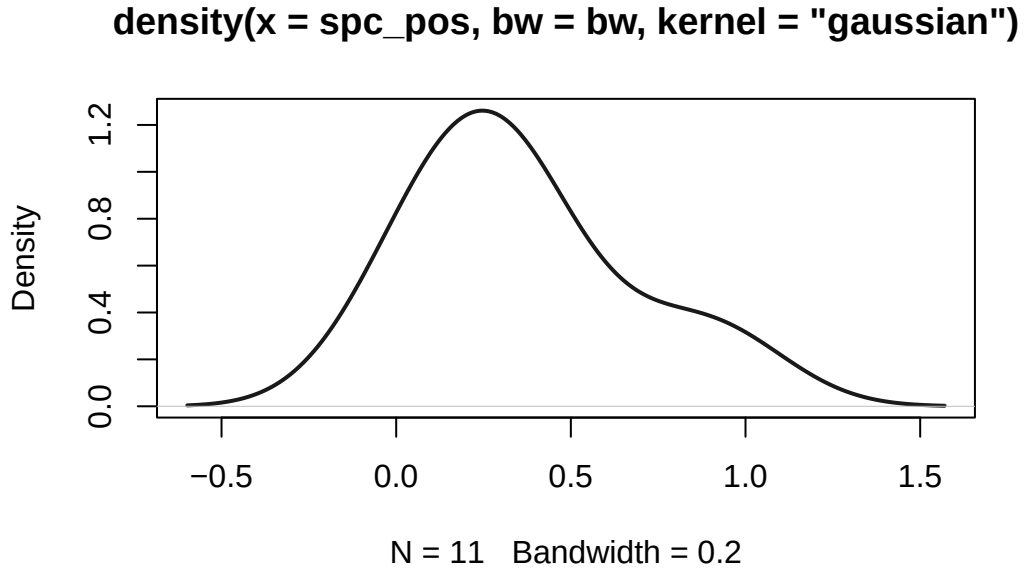

```
Call:
lm(formula = spc_len ~ cos(1 * pi * spc_pos) + sin(4 * pi * spc_pos))

Coefficients:
          (Intercept)  cos(1 * pi * spc_pos)  sin(4 * pi * spc_pos)
              0.16772               -0.08245                0.19561
```

I also got the curiosity to apply Kernel Density Estimation to the data, so I added the parameters 'kdesmooth' and 'bw' (for "bandwidth"):

```
kmer_interspace_polynome(testseq1, kmer="CTC", silent=TRUE, kdesmooth=TRUE, bw = 0.2)
```

```
 [1]   0   5  18 115 121 147 152 178 184 355 470
 [1]   2  15 112 118 144 149 175 181 352 467 500
InterSpaces:
lengths: 0.004 0.02 0.188 0.006 0.046 0.004 0.046 0.006 0.336 0.224 0.06
positions: 0.002 0.02 0.13 0.233 0.265 0.296 0.327 0.359 0.536 0.822 0.97

Loading required package: scales
```

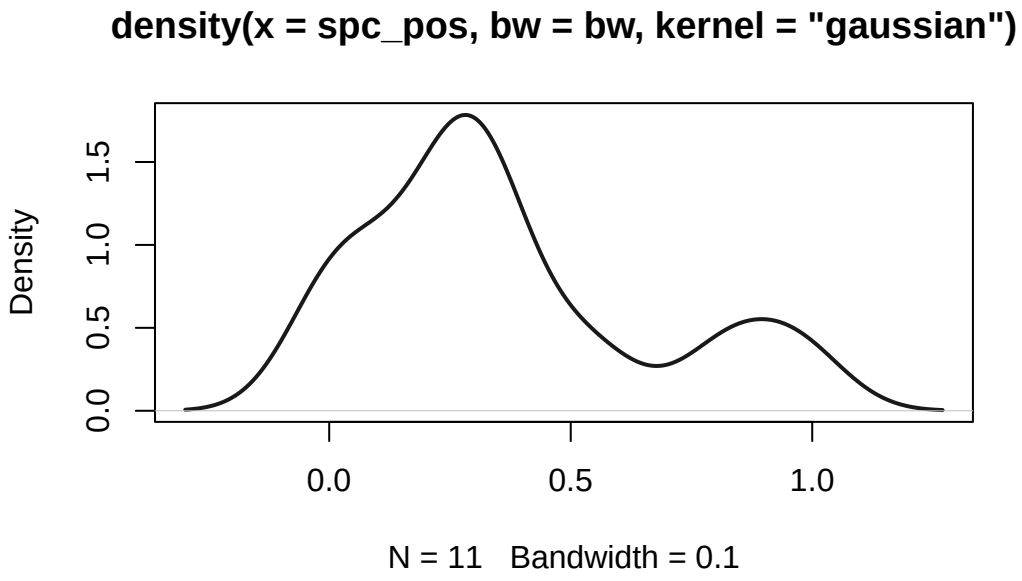
```
Attaching package: 'scales'

The following object is masked from 'package:datawizard':

    rescale

[1] 0.2

Call:
    density.default(x = spc_pos, bw = bw, kernel = "gaussian")

Data: spc_pos (11 obs.);    Bandwidth 'bw' = 0.2

       x                y
 Min.   :-0.598   Min.   :0.002206
 1st Qu.:-0.056   1st Qu.:0.077116
 Median : 0.486   Median :0.374828
 Mean   : 0.486   Mean   :0.460642
 3rd Qu.: 1.028   3rd Qu.:0.770248
 Max.   : 1.570   Max.   :1.261022
```



```
[1] "finished"
```

```
kmer_interspace_polynome(testseq1, kmer="CTC", silent=TRUE, kdesmooth=TRUE, bw = 0.1)
```

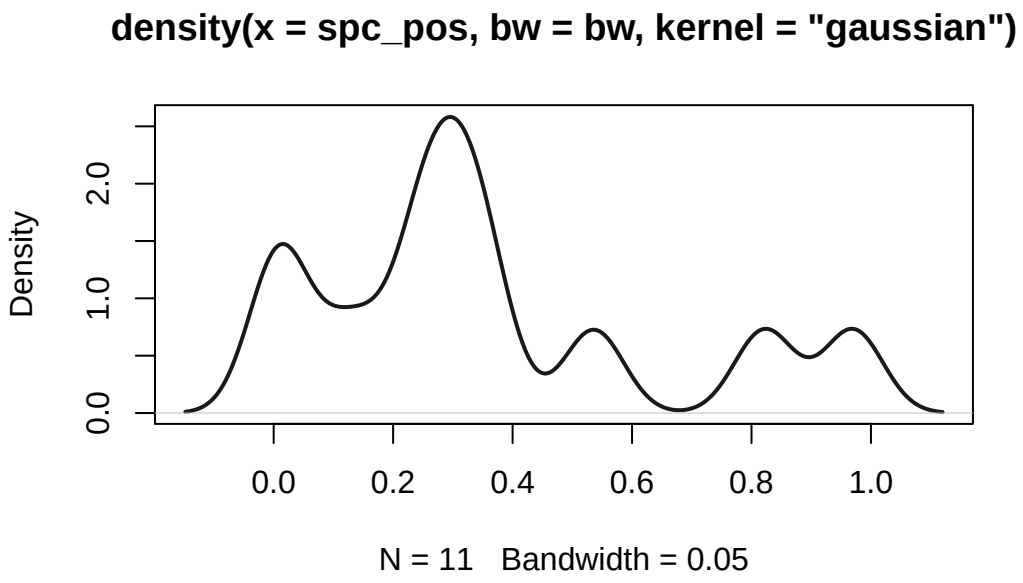
```
 [1]   0   5  18 115 121 147 152 178 184 355 470
 [1]   2  15 112 118 144 149 175 181 352 467 500
InterSpaces:
lengths: 0.004 0.02 0.188 0.006 0.046 0.004 0.046 0.006 0.336 0.224 0.06
positions: 0.002 0.02 0.13 0.233 0.265 0.296 0.327 0.359 0.536 0.822 0.97
[1] 0.1

Call:
    density.default(x = spc_pos, bw = bw, kernel = "gaussian")

Data: spc_pos (11 obs.);    Bandwidth 'bw' = 0.1

       x                y          
 Min.   :-0.298   Min.   :0.00409  
 1st Qu.: 0.094   1st Qu.:0.27074  
 Median : 0.486   Median :0.48539  
 Mean   : 0.486   Mean   :0.63694  
 3rd Qu.: 0.878   3rd Qu.:1.02560  
 Max.   : 1.270   Max.   :1.78358  
```



```
[1] "finished"
```

```
kmer_interspace_polynome(testseq1, kmer="CTC", silent=TRUE, kdesmooth=TRUE, bw = 0.05)
```

```
 [1]   0   5  18 115 121 147 152 178 184 355 470
 [1]   2  15 112 118 144 149 175 181 352 467 500
InterSpaces:
lengths: 0.004 0.02 0.188 0.006 0.046 0.004 0.046 0.006 0.336 0.224 0.06
positions: 0.002 0.02 0.13 0.233 0.265 0.296 0.327 0.359 0.536 0.822 0.97
[1] 0.05

Call:
    density.default(x = spc_pos, bw = bw, kernel = "gaussian")

Data: spc_pos (11 obs.);    Bandwidth 'bw' = 0.05

       x                y          
 Min.   :-0.148   Min.   :0.00818  
 1st Qu.: 0.169   1st Qu.:0.32513  
 Median : 0.486   Median :0.63738  
 Mean   : 0.486   Mean   :0.78767  
 3rd Qu.: 0.803   3rd Qu.:1.05675  
 Max.   : 1.120   Max.   :2.58171  
```



```
[1] "finished"
```

```
#kmer_interspace_polynome(windows=testseq1, kmer="CTC", silent=TRUE, aem_option=7, rel = FALS
```