# An introduction to the nuCpos package

Hiroaki Kato*, Takeshi Urano
Department of Biochemistry, Shimane University School of Medicine

October 29, 2019

## 1 About nuCpos

*nuCpos*, a derivative of *NuPoP*, is an R package for predicting ***nuc***leosome ***pos***itions. In *nuCpos*, a duration hidden Markov model is trained with a ***C***hemical map of nucleosomes either from budding yeast *Saccharomyces cerevisiae* (Brogaard et al. (2012)), fission yeast *Schizosaccharomyces pombe* (Moyle-Heyrman et al. (2012)), or embryonic stem cells of house mouse *Mus musculus* (Voong et al. (2016)). *nuCpos* outputs the Viterbi (most probable) path of nucleosome-linker states, predicted nucleosome occupancy scores and histone binding affinity (HBA) scores as *NuPoP* does. *nuCpos* can also calculate local and whole nucleosomal HBA scores for a given 147-bp sequence. Furthermore, effect of genetic alterations on nucleosome occupancy can be predicted with this package.

The parental package *NuPoP*, licensed under GPL-2, was developed by Ji-Ping Wang and Liqun Xi. Please refer to Xi et al. (2010) and Wang et al. (2008) for technical details of *NuPoP*. Note that *NuPoP* uses an MNase-seq-based map of budding yeast nucleosomes to train a duration hidden Markov model.

## 2 nuCpos functions

*nuCpos* has four functions: `predNuCpos`, `HBA`, `localHBA` and `mutNuCpos`. The `predNuCpos` function can serve a chemical counterpart of the `predNuPoP` function of *NuPoP*: it predicts the nucleosome positioning and nucleosome occupancy.

The functions `HBA` and `localHBA` receive a sequence of 147-bp DNA and calculate whole nucleosomal and local HBA scores. The `mutNuCpos` function receives a wild-type DNA sequence and information on a genetic alteration to predict the effect of the mutation on nucleosome positioning.

*nuCpos* requires the *Biostrings* package, especially when DNA sequences are given as DNAString objects to the functions `HBA`, `localHBA` and `mutNuCpos`. These functions can also receive DNA sequences as simple character string objects without loading the *Biostrings* package. Note: *nuCpos* requires the *NuPoP* package to perform some example runs.

Load the *nuCpos* package as follows:

```
> library(nuCpos)
```

*hkato@med.shimane-u.ac.jp

# 3 Performing predictions with predNuCpos

The `predNuCpos` function acts like the `predNuPoP` function of *NuPoP*. When the *ActLikePredNuPoP* argument is set as TRUE, `predNuCpos` reads a DNA sequence file in FASTA format and invokes a Fortran subroutine to perform predictions. The prediction results will be saved in the working directory. *TRP1ARS1x1.fasta*, the DNA sequence of *TRP1ARS1* circular minichromosome (1,465 bp) (Fuse et al. (2017)), in `extdata` can be used for an example run. Call the `predNuCpos` function as follows:

```
> predNuCpos(file = system.file("extdata", "TRP1ARS1x1.fasta",
+     package = "nuCpos"), species = "sc", smoothHBA = FALSE,
+     ActLikePredNuPoP = TRUE)
```

The argument *file* is the path to the fasta file. The argument *species* can be specified as follows: mm = *M. musculus*; sc = *S. cerevisiae*; sp = *S. pombe*. Re-scaling of the nucleosome and linker models for the prediction of other species' nucleosomes are not supported. *nuCpos* uses 4th order Markov chain models for the prediction.

The name of the output file will be like *TRP1ARS1x1.fasta_Prediction4.txt*. As in the output file produced by the parental *NuPoP* package, it will contain five columns:

1. `Position`: position in the input DNA sequence.
2. `P-start`: probability that a nucleosome starts at.
3. `Occup`: nucleosome occupancy score.
4. `N/L`: Viterbi path (1 and 0 for the nucleosome and linker states, respectively).
5. `Affinity`: histone binding affinity score.

To import the output into R, the `readNuPoP` function of *NuPoP* can be used:

```
> library(NuPoP)
> results.TRP1ARS1x1 <- readNuPoP("TRP1ARS1x1.fasta_Prediction4.txt",
+     startPos = 1, endPos = 1465)
> results.TRP1ARS1x1[1:5,]
```

```
  Position P.start Occup N/L Affinity
1        1   0.000 0.000   0       NA
2        2   0.000 0.000   0       NA
3        3   0.000 0.000   0       NA
4        4   0.001 0.001   0       NA
5        5   0.005 0.006   0       NA
```

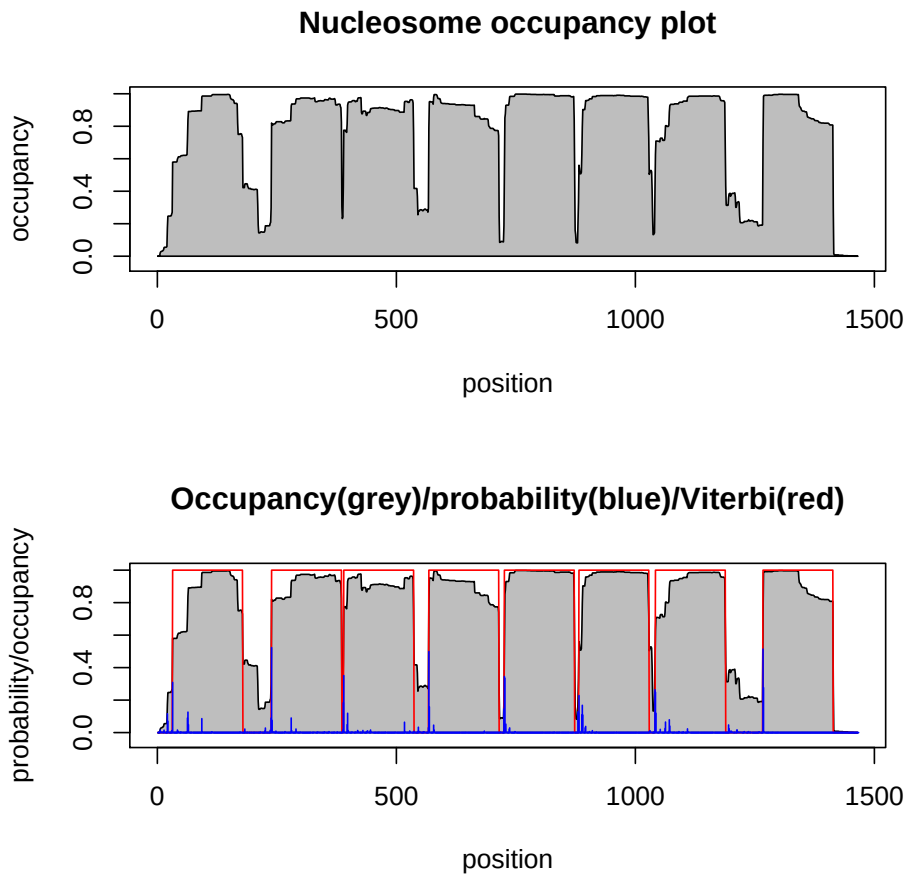
The arguments *startPos* and *endPos* are used to import a part of the prediction results. In this example, the prediction results for the whole tested sequence is imported. First and last 73-bp regions do not have HBA scores (`Affinity`) as they cannot be calculated. The HBA scores start from the 74th position:

```
> results.TRP1ARS1x1[72:76,]
```

```
   Position P.start Occup N/L Affinity
72       72   0.001 0.893   1       NA
73       73   0.000 0.893   1       NA
74       74   0.000 0.893   1    0.346
75       75   0.000 0.893   1   -4.435
76       76   0.000 0.893   1   -3.429
```

For visualization of the prediction results, the `plotNuPoP` function of *NuPoP* can be used. This function draws two plots in the graphical window. The top one shows predicted nucleosome occupancy. In the bottom one, probability of a nucleosome to start at the given position (blue vertical lines) and the Viterbi path (red lines) are shown as well as the nucleosome occupancy (gray).

```
> plotNuPoP(results.TRP1ARS1x1)
```



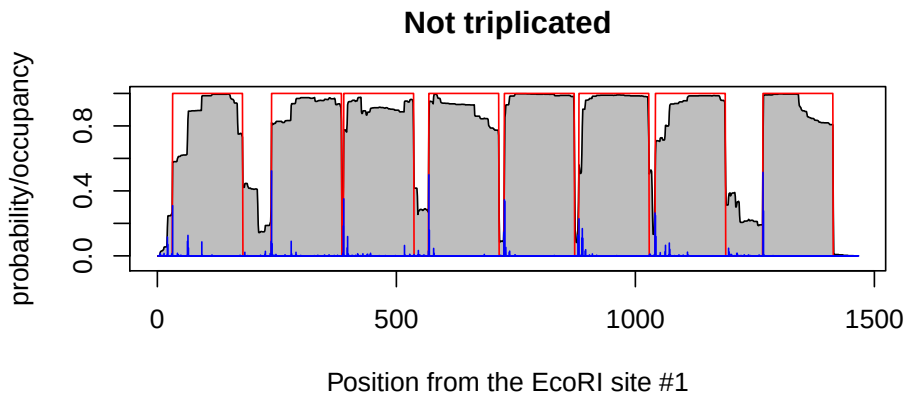

For prediction of nucleosome positioning in short circular DNA, one can use a triplicated sequence for prediction and read only the central copy for the evaluation. By triplicating the DNA, inaccurate prediction near the DNA ends, which are joined to each other in the circular form, can be avoided.

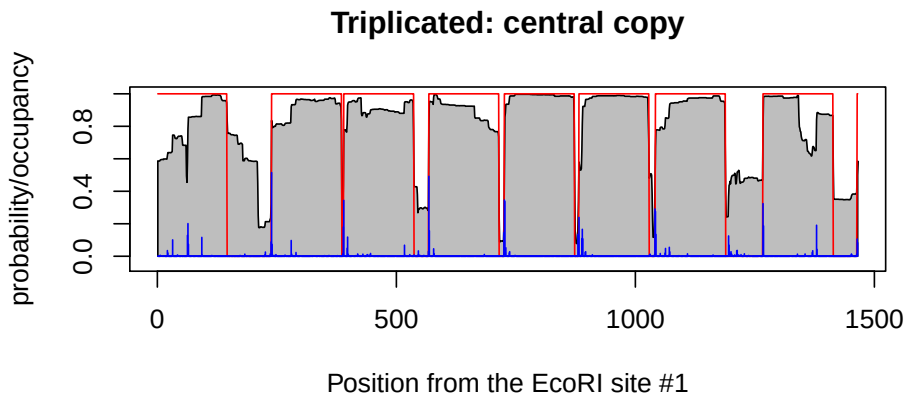
```
> predNuCpos(file = system.file("extdata", "TRP1ARS1x3.fasta",
+     package = "nuCpos"), species = "sc", smoothHBA = FALSE,
```

```
+     ActLikePredNuPoP = TRUE)
> results.TRP1ARS1 <- readNuPoP("TRP1ARS1x3.fasta_Prediction4.txt",
+     startPos = 1466, endPos = 2930)
```

Here, `TRP1ARS1x3.fasta` in `extdata` is a triplicated sequence (4,395 bp) of the *TRP1ARS1* minichromosome (1,465 bp). The central part (from the coordinate 1,466 to 2,930) of the prediction results is read by `readNuPoP`. They are apparently different from the previous results near the terminal regions.

```
> par(mfrow = c(2, 1))
> plot(x = 1:1465, y = results.TRP1ARS1x1[,3], type = "n",
+     ylim = c(-0.05, 1), xlab = "Position from the EcoRI site #1",
+     ylab = "probability/occupancy")
> title("Not triplicated")
> polygon(c(1, 1:1465, 1465), c(0, results.TRP1ARS1x1[,3], 0), col = 8)
> points(x = 1:1465, y = results.TRP1ARS1x1[,4], type = "l", col = 2)
> points(x = 1:1465, y = results.TRP1ARS1x1[, 2], type = "h", col = 4)
> plot(x = 1:1465, y = results.TRP1ARS1[,3], type = "n",
+     ylim = c(-0.05, 1), xlab = "Position from the EcoRI site #1",
+     ylab = "probability/occupancy")
> title("Triplicated: central copy")
> polygon(c(1, 1:1465, 1465), c(0, results.TRP1ARS1[,3], 0), col = 8)
> points(x = 1:1465, y = results.TRP1ARS1[,4], type = "l", col = 2)
> points(x = 1:1465, y = results.TRP1ARS1[, 2], type = "h", col = 4)
```




The main difference between the `predNuCpos` function of *nuCpos* and the `predNuPoP` function of *NuPoP* is the nucleosome maps for constructing statistical models. As *NuPoP* is based on an MNase-based-map, it can be affected by MNase's enzymatic preference to cut AT-rich sequence.

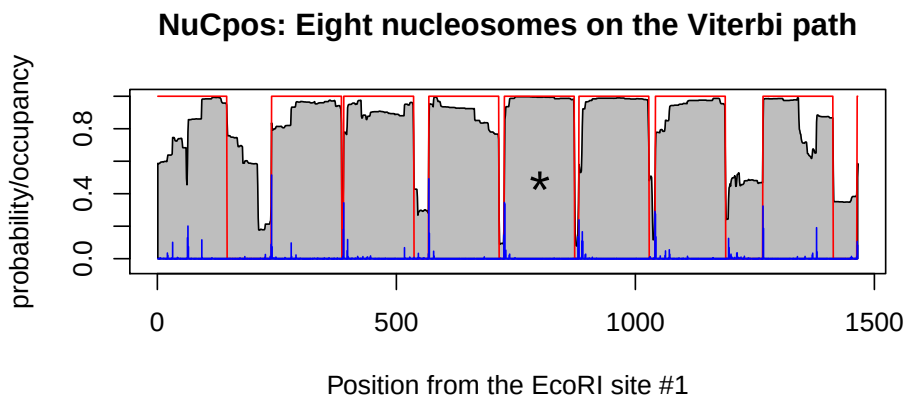
For comparison, use the `predNuPoP` function of *NuPoP* and plot the results alongside those obtained with *nuCpos*. Please ignore the line starting with *Prediction output* and running past the right margin because it cannot be suppressed.

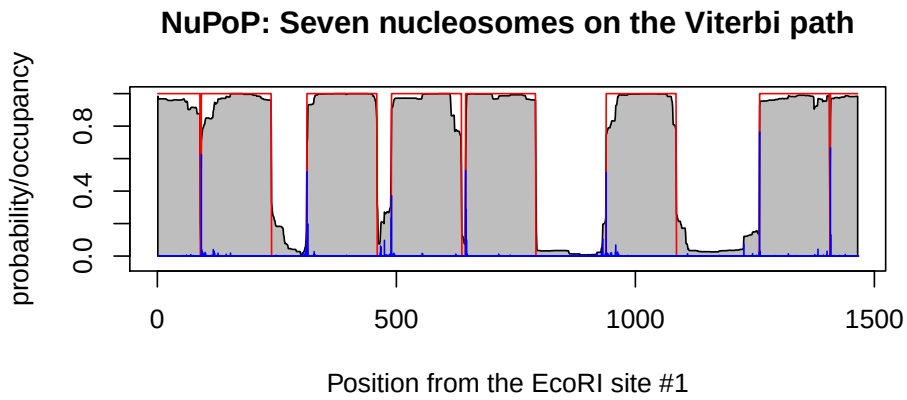
```
> predNuPoP(file = system.file("extdata", "TRP1ARS1x3.fasta",
+     package = "nuCpos"), species = 7, model = 4)
```

```
Prediction output: '/tmp/RtmpKKpfIL/Rbuild2b4d383b366b/nuCpos/vignettes/TRP1ARS1x3.fasta_Prediction4
```

```
> results.NuPoP <- readNuPoP("TRP1ARS1x3.fasta_Prediction4.txt",
+     startPos = 1466, endPos = 2930)
> par(mfrow = c(2, 1))
> plot(x = 1:1465, y = results.TRP1ARS1[,3], type = "n",
+     ylim = c(-0.05, 1), xlab = "Position from the EcoRI site #1",
+     ylab = "probability/occupancy")
> title("NuCpos: Eight nucleosomes on the Viterbi path")
```

```
> polygon(c(1, 1:1465, 1465), c(0, results.TRP1ARS1[,3], 0), col = 8)
> points(x = 1:1465, y = results.TRP1ARS1[,4], type = "l", col = 2)
> points(x = 1:1465, y = results.TRP1ARS1[, 2], type = "h", col = 4)
> text(x = 800, y = 0.5, labels = "*", cex = 2)
> plot(x = 1:1465, y = results.NuPoP[,3], type = "n",
+     ylim = c(-0.05, 1), xlab = "Position from the EcoRI site #1",
+     ylab = "probability/occupancy")
> title("NuPoP: Seven nucleosomes on the Viterbi path")
> polygon(c(1, 1:1465, 1465), c(0, results.NuPoP[,3], 0), col = 8)
> points(x = 1:1465, y = results.NuPoP[,4], type = "l", col = 2)
> points(x = 1:1465, y = results.NuPoP[, 2], type = "h", col = 4)
```

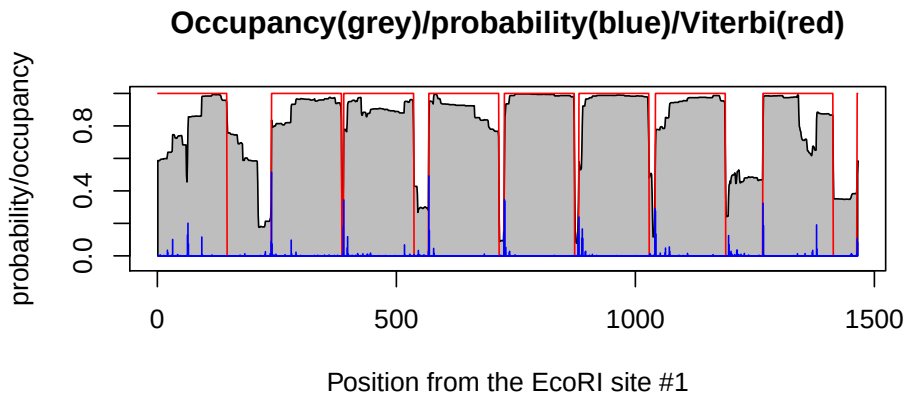


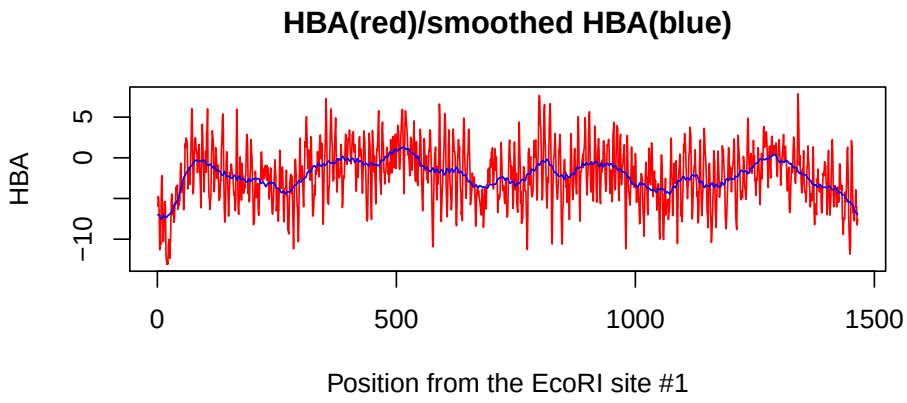




As shown above, eight and seven nucleosomes are predicted along the *TRP1ARS1* minichromosome by *nuCpos* and *NuPoP*, respectively. Most of the nucleosomes predicted by *nuCpos* are in similar locations *in vivo* (Fuse et al. (2017)). The predicted nucleosomes #1-4 appear to correspond to the *in vivo* nucleosomes IV-VII, which locates in the *TRP1* gene. Whereas the predicted nucleosomes #6-8 appear to correspond to the *in vivo* nucleosomes I-III, which locates between the DNase hypersensitive regions (HSR) A and B. The predicted nucleosome #5 (labeled with *) is positioned in the *in vivo* nucleosome free region HSR-A, which contains the DNA replication origin *ARS1*, where origin recognition complex may inhibit nucleosome formation *in vivo*.

By specifying the argument *smoothHBA* as TRUE, HBA scores can be smoothed in a 55-bp window as being done by the `predNuPoP` function of *NuPoP*.

```
> predNuCpos(file = system.file("extdata", "TRP1ARS1x3.fasta",
+     package = "nuCpos"), species = "sc", smoothHBA = TRUE,
+     ActLikePredNuPoP = TRUE)
> results.TRP1ARS1.smooth <- readNuPoP("TRP1ARS1x3.fasta_Prediction4.txt",
+     startPos = 1466, endPos = 2930)
> par(mfrow = c(2, 1))
> plot(x = 1:1465, y = results.TRP1ARS1[,3], type = "n",
+     ylim = c(-0.05, 1), xlab = "Position from the EcoRI site #1",
+     ylab = "probability/occupancy")
> title("Occupancy(grey)/probability(blue)/Viterbi(red)")
> polygon(c(1, 1:1465, 1465), c(0, results.TRP1ARS1[,3], 0), col = 8)
> points(x = 1:1465, y = results.TRP1ARS1[,4], type = "l", col = 2)
> points(x = 1:1465, y = results.TRP1ARS1[, 2], type = "h", col = 4)
> plot(x = 1:1465, y = results.TRP1ARS1[,5], type = "n",
+     xlab = "Position from the EcoRI site #1",
+     ylab = "HBA", main = "HBA(red)/smoothed HBA(blue)")
> points(x = 1:1465, y = results.TRP1ARS1[,5], type = "l", col = 2)
> points(x = 1:1465, y = results.TRP1ARS1.smooth[,5], type = "l", col = 4)
```




As shown as a red line in the bottom one of the above plots, non-smoothed HBA scores in eukaryotic sequences exhibit about 10-bp periodicity. The dyads of predicted nucleosomes usually locate at the coordinates with high HBA scores. HBA scores in the output of `predNuCpos` can be standardized as being done by the `predNuPoP` function of *NuPoP* by specifying the argument *std* of `predNuCpos` as TRUE. The default setting for *std* is FALSE.

When the argument *ActLikePredNuPoP* is set as FALSE, which is the default setting, `predNuCpos` receives a character string or DNAString object as *inseq*. In this case, prediction results will be returned to the R environment, and no file will be generated in the working directory. The input sequence (*inseq*) must not contain characters other than A/C/G/T.

The results will contain five columns:

1. `pos`: position in the input DNA sequence.
2. `pstart`: probability that a nucleosome starts at.
3. `nucoccup`: nucleosome occupancy score.
4. `viterbi`: Viterbi path (1 and 0 for the nucleosome and linker states, respectively).
5. `affinity`: histone binding affinity score.

```
> TRP1ARS1 <- paste(scan(file =
+     system.file("extdata", "TRP1ARS1x1.fasta", package = "nuCpos"),
+     what = character(), skip = 1), sep = "", collapse = "")
> results.TRP1ARS1.internal <-
+     predNuCpos(inseq = TRP1ARS1, species = "sc", smoothHBA = FALSE,
+     ActLikePredNuPoP = FALSE)
> results.TRP1ARS1.internal[72:76,]

   pos       pstart  nucoccup viterbi   affinity
72  72 9.802925e-04 0.8926019       1         NA
73  73 3.182192e-04 0.8929201       1         NA
74  74 5.025299e-05 0.8929704       1  0.3456824
75  75 5.160730e-06 0.8929756       1 -4.4353083
76  76 1.397652e-06 0.8929769       1 -3.4287470
```

# 4 Histone binding affinity score calculation with HBA

HBA score can be calculated for a given 147-bp sequence with the `HBA` function. In the examples bellow, a character string object `inseq` and a DNAString object `INSEQ` with the same 147-bp DNA sequences are given to `HBA`. Note: the *Biostrings* package is required for the latter case.

```
> load(system.file("extdata", "inseq.RData", package = "nuCpos"))
> HBA(inseq = inseq, species = "sc")

      HBA
-2.460025

> for(i in 1:3) cat(substr(inseq, start = (i-1)*60+1,
+     stop = (i-1)*60+60), "\n")

ATCGAGAATCCCGGTGCCGAGGCCGCTCAATTGGTCGTAGACAGCTCTAGCACCGCTTAA
ACGCACGTACGCGCTGTCCCCCGCGTTTTAACCGCCAAGGGGATTACTCCCTAGTCTCCA
GGCACGTGTCAGATATATACATCCGAT

> load(system.file("extdata", "INSEQ_DNAString.RData",
+     package = "nuCpos"))
> INSEQ

  147-letter "DNAString" instance
seq: ATCGAGAATCCCGGTGCCGAGGCCGCTCAATTGGTC...TAGTCTCCAGGCACGTGTCAGATATATACATCCGAT

> HBA(inseq = INSEQ, species = "sc")

      HBA
-2.460025
```

The argument *inseq* is the character string object to be given. Alternatively, a DNAString object can be used here. The length of DNA must be 147 bp. The argument *species* can be specified as follows: mm = *M. musculus*; sc = *S. cerevisiae*; sp = *S. pombe*.

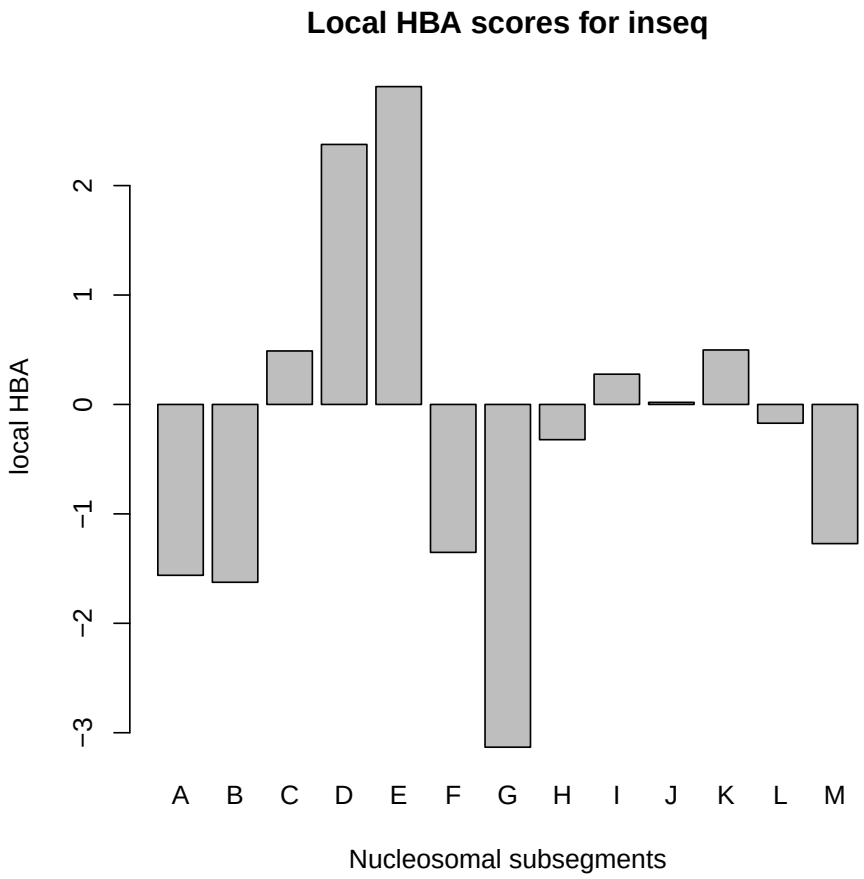
# 5 Local histone binding affinity score calculation with localHBA

Local HBA scores are defined as HBA scores for 13 overlapping subnucleosomal segments named A to M. They can be calculated for a given 147-bp sequence with the `localHBA` function. Like `HBA`, this function can receive either a character string object or a DNAString object. The segment G corresponds to the central 21 bp region, in which the dyad axis passes through the 11th base position. This means that the local HBA score for the G segment implies the relationship between DNA and histone proteins at around superhelical locations -0.5 and +0.5. The neighboring F segment, which is 20 bp in length, is for SHLs -1.5 and -0.5. The result of example run shown below suggests that subsequence of `inseq` around SHL -3.5 and -2.5 is suitable for nucleosome formation.

```
> localHBA(inseq = inseq, species = "sc")

     lHBA_A      lHBA_B      lHBA_C      lHBA_D      lHBA_E      lHBA_F
-1.56140949 -1.62502354  0.48885990  2.37615568  2.90458625 -1.35195919
     lHBA_G      lHBA_H      lHBA_I      lHBA_J      lHBA_K      lHBA_L
-3.13228907 -0.32208031  0.27650871  0.01922002  0.49787625 -0.17151500
     lHBA_M
-1.27186158

> barplot(localHBA(inseq = inseq, species = "sc"),
+     names.arg = LETTERS[1:13], xlab = "Nucleosomal subsegments",
+     ylab = "local HBA", main = "Local HBA scores for inseq")
```




# 6 Prediction of nucleosome positioning in wild-type and mutant sequences with mutNuCpos

The function `mutNuCpos` is designed to examine the effect of genetic alterations on nucleosome positioning. In the example run below, the *TALS* circular minichromosome (1,811 bp) is used as a wild-type sequence. Ichikawa et al. (2014) showed that insertion of telomere repeats into the *TALS* minichromosome inhibits formation of positioned nucleosome *in vivo*. Here, a 178-bp telomere repeat sequence is inserted at the base position 1,464 of *TALS*.

```
> TALS <- paste(scan(file =
+     system.file("extdata", "TALS.fasta", package = "nuCpos"),
+     what = character(), skip = 1), sep = "", collapse = "")
> for(i in 1:23) cat(substr(TALS,
+     start = (i-1)*80+1, stop = (i-1)*80+80), "\n")
```

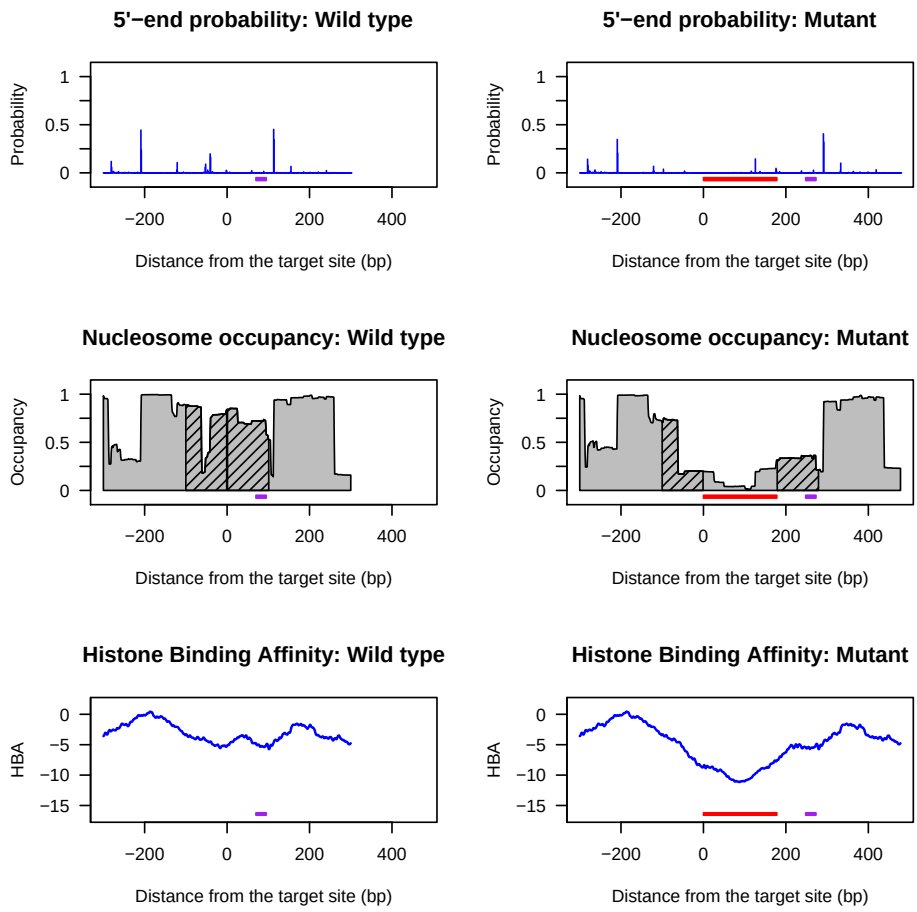
```
AATTCGGTCGAAAAAAGAAAAGGAGAGGGCCAAGAGGGAGGGCATTGGTGACTATTGAGCACGTGAGTATACGTGATTAA
GCACACAAAGGCAGCTTGGAGTATGTCTGTTATTAATTTCACAGGTAGTTCTGGTCCATTGGTGAAAGTTTGCGGCTTGC
```

```
AGAGCACAGAGGCCGCAGAATGTGCTCTAGATTCCGATGCTGACTTGCTGGGTATTATATGTGTGCCCAATAGAAAGAGA
ACAATTGACCCGGTTATTGCAAGGAAAATTTCAAGTCTTGTAAAAGCATATAAAAATAGTTCAGGCACTCCGAAATACTT
GGTTGGCGTGTTTCGTAATCAACCTAAGGAGGATGTTTTGGCTCTGGTCAATGATTACGGCATTGATATCGTCCAACTGC
ATGGAGATGAGTCGTGGCAAGAATACCAAGAGTTCCTCGGTTTGCCAGTTATTAAAAGACTCGTATTTCCAAAAGACTGC
AACATACTACTCAGTGCAGCTTCACAGAAACCTCATTCGTTTATTCCCTTGTTTGATTCAGAAGCAGGTGGGACAGGTGA
ACTTTTGGATTGGAACTCGATTTCTGACTGGGTTGGAAGGCAAGAGAGCCCCGAAAGCTTACATTTTATGTTAGCTGGTG
GACTGACGCCAGAAAATGTTGGTGATGCGCTTAGATTAAATGGCGTTATTGGTGTTGATGTAAGCGGAGGTGTGGAGACA
AATGGTGTAAAAGACTCTAACAAAATAGCAAATTTCGTCAAAAATGCTAAGAAATAGGTTATTACTGAGTAGTATTTATT
TAAGTATTGTTTGTGCACTTGCCTGCAGGCCTTTTGAAAAGCAAGCATAAAAGATCTAAACATAAAATCTGTAAAATAAC
AAGATGTAAAGATAATGCTAAATCATTTGGCTTTTTGATTGATTGTACAGGAAAATATACATCGCAGGGGGTTGACTTTT
ACCATTTCACCGCAATGGAATCAAACTTGTTGAAGAGAATGTTCACAGGCGCATACGCTACAATGACCCGATTCTTGCTA
GCCTTTTCTCGGTCTTGCAAACAACCGCCGGCAGCTTAGTATATAAATACACATGTACATACCTCTCTCCGTATCCTCGT
AATCATTTTCTTGTATTTATCGTCTTTTCGCTGTAAAAACTTTATCACACTTATCTCAAATACACTTATTAACCGCTTTT
ACTATTATCTTCTACGCTGACAGTAATATCAAACAGTGACACATATTAAACACAGTGGTTTCTTTGCATAAACACCATCA
GCCTCAAGTCGTCAAGTAAAGATTTCGTGTTCATGCAGATAGATAACAATCTATATGTTGATAATTAGCGTTGCCTCATC
AATGCGAGATCCGTTTAACCGGACCCTAGTGCACTTACCCCACGTTCGGTCCACTGTGTGCCGAACATGCTCCTTCACTA
TTTTAACATGTGGAATTCGAGCTCGGTCGAAAATGATGAACGGCAATAATGCAACAGTTTTCATATCTGAAACAATTGTA
TATGTGCGAAGCAGCGTGTAAATTTCCCTATTAGGTAATTACATGGCAAAAAAAATAAGAAACTATTTTCTTAAAGTCTA
GAGTCGACCTGCAGGCATGCAAGCTAGCTTGGCGTAATCATGGTCATAGCTGTTTCCTGTGTGAAATTGTTATCCGCTCA
CAATTCCACACAACATACGAGCCGGAAGCATAAAGTGTAAAGCCTGGGGTGCCTAATGAGTGAGCTAACTCACATTAATT
GCGTTGCGCTCACTGCCCGCTTTCCAGTCGGGAAACCTGTCGTGCCAGCGG
```

```
> TTAGGGx29 <- paste(scan(file =
+     system.file("extdata", "TTAGGGx29.fasta", package = "nuCpos"),
+     what = character(), skip = 1), sep = "", collapse = "")
> for(i in 1:3) cat(substr(TTAGGGx29,
+     start = (i-1)*80+1, stop = (i-1)*80+80), "\n")
```

```
TTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTT
AGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAG
GGTTAGGGTTAGGGAGCT
```

```
> mut.results <- mutNuCpos(wtseq = TALS, site = 1464, ins = TTAGGGx29,
+     species = "sc", smoothHBA = TRUE, plot.window = 601,
+     ylim.HBA = c(-15, 0), show.occup.window = TRUE,
+     annotation = data.frame(name = "alpha2", color = "purple",
+     left = 1534, right = 1559), full = TRUE)
```




The function `mutNuCpos` can receive either character string objects or DNAString objects as the arguments *wtseq* and *ins*. The position of insertion is specified by the argument *site*. The argument *species* is set as in other functions such as `predNuCpos`. When *show.occup.window* is TRUE, the regions used for calculation of *Difference in occupancy*, which is stored in the results (or printed in the console when `full = TRUE`), are shaded in the nucleosome occupancy plots.

`mutNuCpos` does not save any data in the working directory. To receive calculation results from the function, use the assignment operator and set the *full* argument as TRUE to do so. One can obtain the prediction results for a wild-type sequence (or a mutant sequence by giving it as *wtseq*) by setting the arguments *ins* and *del* as defaults.

The argument *annotation* is useful for indicating positions of relevant elements. In this example, the alpha-2 operator is shown as purple horizontal lines. The red horizontal lines indicate insertions.

This function receives a wild-type DNA sequence longer than 1,000 bp. The sequence can only contain A, C, G and T. The received sequence is mutagenized and pentaplicated before performing predictions to avoid terminal effects. The coordinates (`pos`) for the results are with respect to the central copy of the pentaplicated sequence.

```
> mut.results[(((1811+76)*2)-3):(((1811+76)*2)+3),]

      pos       pstart  nucoccup viterbi  affinity
3771 -207 8.766279e-06 0.9239269       1 -3.047591
```

```
3772 -206 5.680624e-05 0.9239836       1 -3.118130
3773 -205 6.468577e-05 0.9240480       1 -3.181660
3774 -204 1.446111e-04 0.9241662       1 -3.228653
3775 -203 2.695176e-05 0.9240940       1 -3.318924
3776 -202 3.432725e-04 0.8775106       1 -3.322130
3777 -201 1.876886e-04 0.8438925       1 -3.350687
```

# 7 Acknowledgements

We would like to thank Drs. Shimizu, Fuse and Ichikawa for sharing DNA sequences and *in vivo* data, and giving fruitful comments.

# References

Wang JP, Fondufe-Mittendorf Y, Xi L, Tsai GF, Segal E and Widom J (2008). Preferentially quantized linker DNA lengths in *Saccharomyces cerevisiae*. *PLoS Computational Biology*, 4(9):e1000175.

Xi L, Fondufe-Mittendorf Y, Xia L, Flatow J, Widom J and Wang JP (2010). Predicting nucleosome positioning using a duration hidden markov model. *BMC Bioinformatics*, 11:346.

Brogaard K, Xi L, and Widom J (2012). A map of nucleosome positions in yeast at base-pair resolution. *Nature*, 486(7404):496-501.

Moyle-Heyrman G, Zaichuk T, Xi L, Zhang Q, Uhlenbeck OC, Holmgren R, Widom J and Wang JP (2013). Chemical map of *Schizosaccharomyces pombe* reveals species-specific features in nucleosome positioning. *Proc. Natl. Acad. Sci. U. S. A.*, 110(50):20158-63.

Ichikawa Y, Morohoshi K, Nishimura Y, Kurumizaka H and Shimizu M (2014). Telomeric repeats act as nucleosome-disfavouring sequences in vivo. *Nucleic Acids Res.*, 42(3):1541-1552.

Voong LN, Xi L, Sebeson AC, Xiong B, Wang JP and Wang X (2016). Insights into Nucleosome Organization in Mouse Embryonic Stem Cells through Chemical Mapping. *Cell*, 167(6):1555-1570.

Fuse T, Katsumata K, Morohoshi K, Mukai Y, Ichikawa Y, Kurumizaka H, Yanagida A, Urano T, Kato H, and Shimizu M (2017). Parallel mapping with site-directed hydroxyl radicals and micrococcal nuclease reveals structural features of positioned nucleosomes in vivo. *Plos One*, 12(10):e0186974.