Comparison of nucleosome dynamics using the NucDyn package

Diana Buitrago, Ricard Illa

2019-04-09

Contents

1 Introduction 2

2 Package content 2

2.1 Sample Data 2

2.2 Method 2

2.3 Usage 3

References 7

1 Introduction

NucDyn is a package designed to compare nucleosome positioning between two different cell conditions using the mapped reads from MNase-seq experiments. It identifies two different types of nucleosome rearrengements, working at the fragment level:

- Occupancy differences: nucleosome inclusions and evictions
- Nucleosome displacements: upstream or downstream shifts

The package requires preliminary detection of nucleosome positions from the MNase-seq data sets, which can be obtained using *nucleR*.

The main functions of *NucDyn* are:

- `nucleosomeDynamics` finds pairs of fragments between the two experiments, corresponding to shifts, and detects fragments removed and inserted.
- `findHotspots` groups all the fragments corresponding to a nucleosome change and scores the detected hotspots.

For more details about the functions arguments and options refer to the *NucDyn* manual.

2 Package content

2.1 Sample Data

The examples analysed in this vignette were obtained from a study where MNaseSeq experiments for *S. cerevisiae* in G2 and M cell cycle phases were performed (Deniz et al. 2016). Sequencing reads retrieved from ENA repository under accession PRJEB6970 were mapped to the budding yeast genome (sacCer3 version) using Bowtie and imported into *R* using *ShortRead*. We provide, within *NucDyn* package, reads mapped to chromosome II from these experiments.

2.2 Method

NucDyn allows comparison of two diferent MNase-seq experiments (Condition 1 and Condition 2) and detects local changes such as:

- **Shift -**, were the occupancy is shifted upstream on Condition 2 relative to Condition 1.
- **Shift +**, were the occupancy is shifted downstream on Condition 2 relative to Condition 1.
- **Inclusion**, where the nucleosome occupancy is significantly increased.
- **Eviction**, where the nucleosome occupancy is significantly decreased.

NucDyn identifies those changes working at read-level. Its pipeline sequentially pairs fragments of one experiment with fragments of the second experiment to remove those that are not informative, i.e. that did not change between the two conditions but only represent differences in MNase efficiency or PCR duplicates. First, all reads that are identical on both experiments are discarded. Then, the reads that either start or end at the same position are discarded as well. Next, the reads from one experiment whose range is completely contained by the range of a read in the other experiment, are paired and removed too.

With the informative fragments, *NucDyn* then pairs reads from one experiment to reads in the other experiment whose center is shifted either upstream or downstream relative to each other. In an attempt to find the best possible combination of pairs, a dynamic programming algorithm is used. It is defined with the following conditions:

1. The largest possible number of pairs is found.
2. The centers of the paired reads are as close as possible.
3. Reads whose center is futher apart than a given distance (by default 74 bp, i.e. half of a nucleosome length) are never paired.

To achieve that, the dynamic programming algorithm works in the following way:

1. Gaps are highly penalized (to achieve maximum number of pairs).
2. Pairs are given a score that is inversely proportional to the centers distance (to prioratize close pairs).
3. Pairs whose centers are at a distance larger than 74 bp. are given a score of -Infinity so that it can never happen.

Once the pairs of fragments corresponding to shifts have been found, accumulations of them are considered shift hotspots.

In order to identify significant changes in occupancy (insertions and deletions), the coverage of the reads in both experiments is used. First a normalized z score accross the genome is calculated, assuming a hypergeometric distrubition. This score is normalized on a given window (by default 10000 bp.). In this way, coverage fluctuations accross big segments of the genome are taken into account and locally significant differences are detected. Positive peaks of that z score mean that the coverage of the Experiment 1 is significantly higher than the coverage of the Experiment 2, and therefore it is considered a deletion. Similarly, negative z score peaks represent regions where the coverage of the Experiment 2 is significantly higher and therefore classified as insertions.

Finally, the *NucDyn* pipeline scores the hotspots found. With this intention, a p-value is calculated for each point on the sequence, that accounts for statistically significant differences of coverage in a given window (10000 bp. by default). Fisher's test comparing reads from the two experiments at the given position is calculated. The score given to each hotspot corresponds to the p-value at its peak position. Then, a threshold is applied to report only the more relevant hotspots.

2.3 Usage

Using the MNase-seq data described above, we will explain how to detect changes in nucleosomes between G2 and M cell cycle phases. First we need to load the library and the sample mapped reads in the two phases:

```
library(NucDyn)
data(readsG2_chrII)
data(readsM_chrII)
```

Data are in GRanges format from the *GenomicRanges* package. BAM files containing aligened reads can by imported in R using `scanBam` function from *Rsamtools* package and then transformed to GRanges format.

```
head(readsG2_chrII)
## RangedData with 6 rows and 0 value columns across 1 space
##      space    ranges |
```

```
##   <factor> <IRanges> |
## 1    chrII    10-168 |
## 2    chrII    43-167 |
## 3    chrII    44-162 |
## 4    chrII    56-185 |
## 5    chrII    61-342 |
## 6    chrII    62-172 |
```

Now we will use *NucDyn* to obtain fragment shifts (+ or -), inclusions and evictions between G2 and M:

```
dyn <- nucleosomeDynamics(setA=readsG2_chrII, setB=readsM_chrII)
## Starting chrII
## chrII done
## Combining the calculations from different chromosomes...

print(dyn)
## Set a:
## originals: 420776 reads | coinciding: 26495 reads | same.start: 105504 reads | same.end: 44738 reads | co
## Set b:
## originals: 219964 reads | coinciding: 26495 reads | same.start: 105504 reads | same.end: 44738 reads | co
```

The output contains all original reads as well as the different pairings identified between the two experiments: identical, with same start, same end, contained, shifts, indels or unpaired reads. Each group is a GRanges object, for instance to obtain all reads identified as left.shifts from G2 (setA) to M (setB):

```
head(dyn@set.a[["left.shifts"]])
## GRanges object with 6 ranges and 1 metadata column:
##       seqnames      ranges strand |        type
##          <Rle>   <IRanges>  <Rle> |       <Rle>
##   [1]    chrII   1695-1827      * | left.shifts
##   [2]    chrII   1933-2076      * | left.shifts
##   [3]    chrII   1935-2058      * | left.shifts
##   [4]    chrII   6109-6252      * | left.shifts
##   [5]    chrII 12533-12688      * | left.shifts
##   [6]    chrII 12540-12683      * | left.shifts
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqlengths
head(dyn@set.b[["left.shifts"]])
## GRanges object with 6 ranges and 1 metadata column:
##       seqnames      ranges strand |        type
##          <Rle>   <IRanges>  <Rle> |       <Rle>
##   [1]    chrII   1625-1752      * | left.shifts
##   [2]    chrII   1911-2050      * | left.shifts
##   [3]    chrII   1917-2050      * | left.shifts
##   [4]    chrII   6069-6199      * | left.shifts
##   [5]    chrII 12513-12630      * | left.shifts
##   [6]    chrII 12513-12637      * | left.shifts
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

In order to group the paired reads and detect significant changes, we need previously identified nucleosome positions. We will load the precomputed nuclesome calls obtained with *nucleR*:

```
data(nuc_chrII)
head(nuc_chrII)
## $nucleoG2
## GRanges object with 4907 ranges and 0 metadata columns:
##          seqnames        ranges strand
##             <Rle>     <IRanges>  <Rle>
##      [1]    chrII       159-305      *
##      [2]    chrII     1521-1667      *
##      [3]    chrII     1624-1770      *
##      [4]    chrII     1812-1958      *
##      [5]    chrII     1937-2083      *
##      ...      ...           ...    ...
##   [4903]    chrII 808720-808866      *
##   [4904]    chrII 809288-809434      *
##   [4905]    chrII 809607-809753      *
##   [4906]    chrII 809910-810056      *
##   [4907]    chrII 810071-810217      *
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqlengths
##
## $nucleoM
## GRanges object with 4862 ranges and 0 metadata columns:
##          seqnames        ranges strand
##             <Rle>     <IRanges>  <Rle>
##      [1]    chrII     1604-1750      *
##      [2]    chrII     1700-1846      *
##      [3]    chrII     1770-1916      *
##      [4]    chrII     1868-2080      *
##      [5]    chrII     2059-2205      *
##      ...      ...           ...    ...
##   [4858]    chrII 808722-808868      *
##   [4859]    chrII 809289-809435      *
##   [4860]    chrII 809607-809753      *
##   [4861]    chrII 809911-810057      *
##   [4862]    chrII 810075-810221      *
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

Now we can use the nucleosome positions in the two experiments to obtain the hotspots:

```
hs <- findHotspots(dyn, nuc_chrII)
## Starting chrII
## chrII done
head(hs)
##    start  end peak    nreads     score      type   chr
## 3    483  509  497 0.4798962 0.4767569 INCLUSION chrII
## 8   1515 1720 1611 1.6028045 0.1964027 INCLUSION chrII
## 14  1778 1813 1814 0.0000000 0.5680894 INCLUSION chrII
## 18  1835 1847 1848 0.0000000 0.8222719 INCLUSION chrII
```

```
## 19  1864 1915 1888 0.4775369 0.5083698 INCLUSION chrII
## 22  1917 2035 1988 1.0403746 0.4123206 INCLUSION chrII
```

```
sessionInfo()
## R version 3.5.1 (2018-07-02)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17134)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=Spanish_Spain.1252  LC_CTYPE=Spanish_Spain.1252
## [3] LC_MONETARY=Spanish_Spain.1252 LC_NUMERIC=C
## [5] LC_TIME=Spanish_Spain.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] NucDyn_0.99.0    BiocStyle_2.10.0
##
## loaded via a namespace (and not attached):
##  [1] SummarizedExperiment_1.10.1 tidyselect_0.2.4
##  [3] xfun_0.5                    purrr_0.2.5
##  [5] lattice_0.20-35             colorspace_1.3-2
##  [7] htmltools_0.3.6             stats4_3.5.1
##  [9] yaml_2.2.0                  rlang_0.2.2
## [11] pillar_1.3.0                glue_1.3.0
## [13] BiocParallel_1.14.2         RColorBrewer_1.1-2
## [15] BiocGenerics_0.26.0         bindrcpp_0.2.2
## [17] matrixStats_0.54.0          GenomeInfoDbData_1.1.0
## [19] foreach_1.4.4               plyr_1.8.4
## [21] bindr_0.1.1                 stringr_1.3.1
## [23] zlibbioc_1.26.0             Biostrings_2.48.0
## [25] munsell_0.5.0               gtable_0.2.0
## [27] hwriter_1.3.2               codetools_0.2-15
## [29] evaluate_0.12               latticeExtra_0.6-28
## [31] Biobase_2.40.0              knitr_1.20
## [33] IRanges_2.14.11             doParallel_1.0.11
## [35] GenomeInfoDb_1.16.0         parallel_3.5.1
## [37] Rcpp_0.12.18                scales_1.0.0
## [39] BiocManager_1.30.4          DelayedArray_0.6.5
## [41] S4Vectors_0.18.3            XVector_0.20.0
## [43] ShortRead_1.38.0            Rsamtools_1.32.3
## [45] ggplot2_3.0.0               digest_0.6.16
## [47] stringi_1.1.7               bookdown_0.9
## [49] dplyr_0.7.6                 GenomicRanges_1.32.6
## [51] grid_3.5.1                  tools_3.5.1
## [53] bitops_1.0-6                magrittr_1.5
## [55] RCurl_1.95-4.11             lazyeval_0.2.1
## [57] tibble_1.4.2                crayon_1.3.4
```

```
## [59] pkgconfig_2.0.2            Matrix_1.2-14
## [61] nucleR_2.12.1              assertthat_0.2.0
## [63] rmarkdown_1.11             rstudioapi_0.7
## [65] iterators_1.0.10           R6_2.2.2
## [67] GenomicAlignments_1.16.0    compiler_3.5.1
```

References

Deniz, Özgen, Oscar Flores, Martí Aldea, Montserrat Soler-López, and Modesto Orozco. 2016. "Nucleosome Architecture Throughout the Cell Cycle." *Scientific Reports* 6 (January): 19729. doi:10.1038/srep19729.