NucleoMiner - Tutorial

Jean-Baptiste Veyrieras (jb.veyrieras@gmail.com)
Gael Yvert (gael.yvert@ens-lyon.fr)
CNRS, Ecole Normale Supérieure de Lyon

2010

Contents

1 Introduction

This manual describes the different steps needed to make a full comparison of nucleosome occupancy and modification profile. The steps described in this manual are similar to the steps performed for the results published in the article [5]. We assume that **NucleoMiner** has been previously installed on your machine and that it is correctly running. Otherwise please see the **NucleoMiner** manual. Assume that your:

- `$PATH` is configurated properly
- `$PERLLIB` is configurated properly
- `$LD_LIBRARY_PATH` is configurated properly

2 Mapping tiling array probes on genome

Programs used in this section

- `NM1lq2fsa` (perl) extract probe sequences from .1lq file
- `NMmp2g` (shell) map probes to genome
- `NMmpa` (C) extract probe matches
- `NMepm` (perl) extract probe with unique match

2.1 Extacting probe sequences in fasta format

First, go into the `Affymetrix` directory in `Data` to generate from the .1lq file provided by Affymetrix the corresponding fasta file.

```
$NM1lq2fsa -i S.cerevisiae_tiling.1lq --reverse 1>S.cerevisiae_tiling.fasta
```

This generally takes a few minutes, so be patient. Note that here we have to use the `--reverse` option since the probe sequence in the Affetrix file are stored in reverse order. Then, if you have a look at the fasta file generated by `NM1lq2fsa` you should have something like that

```
$head -10 S.cerevisiae_tiling.fasta
>0_0
TCCTGAACGGTAGCATCTTGACGAC
>1_0
GTCGTCAAGATGCTACCGTTCAGGA
>2_0
TCCTGAACGGTAGCATCTTGACGAC
>3_0
GTCGTCAAGATGCTACCGTTCAGGA
>4_0
NNNNNNNNNNNNNNNNNNNNNNNNN
```

where the unique identifient of each probe is given here by the concatenation of the (x, y) coordinate of the probe on the array (separated by a underscore, i.e x_y). Note here that sequence `4_0` is just a serie of `N`s. This happens when the corresponding sequence in the .1lq file is in fact not defined.

2.2 Creating the probe annotation files

To annotate the probes here simply consists in getting their coordinate on a given genome sequence. To do that, we will use the software `MUMmer` (`http://mummer.sourceforge.net/`) to first rapidly align the probes onto the genome sequence. Second, a short series of scripts will help us to create annotation files and other convenient files.

MUMmer We assume that you have previously installed the last `MUMmer` version ($\geq$ 3.0) on your machine. Otherwise please go to `http://mummer.sourceforge.net/` and follow the install procedures describe in the online manual. **Don't forget to add the `MUMmer` directory to your `$PATH`.** Assuming you are running a shell in bash, this can be simply done as follows:

```
$export PATH=${PATH}:/path/to/MUMmer/directory
```

NMmp2g Once `MUMmer` have been installed, the first step consists in mapping the probe sequences onto the genome sequence. Go into folder `S1/BY_S288` and use the script `NMmp2g` as follows:

```
$NMmp2g ../../Data/BY_S288c/Sequence/Genome.fasta \
       ../../Data/Affymetrix/S.cerevisiae_tiling.fasta \
       NMmp2g.out 2>NMmp2g.log
```

where the first argument (here `../../Data/BY_S288c/Sequence/Genome.fasta`) is the path to the entire genome sequence in fasta format, the second (`../../Data/Affymetrix/S.cerevisiae_tiling.fasta`) is the path to the probe sequences in fasta format - as previously generated - and the last one, `NMmp2g.out` is the output file into which results from `MUMmer` will be stored. The last part `NMmp2g.log` is just a redirection of the standard output (of `MUMmer`) into the file `NMmp2g.log`.

So, if everythink worked fine, you should have a file `NMmp2g.out` that looks like that:

```
$head -10 NMmp2g.out
> 0_0  Len = 25
> 0_0 Reverse  Len = 25
> 1_0  Len = 25
> 1_0 Reverse  Len = 25
> 2_0  Len = 25
> 2_0 Reverse  Len = 25
> 3_0  Len = 25
```

```
> 3_0 Reverse  Len = 25
> 4_0  Len = 25
> 4_0 Reverse  Len = 25
```

As you can see, the output of `MUMmer` needs to be reformated in order to create the probe annotation files. In the above extract of the output, you can note that none of the probes match the genome sequence. When a probe have a match on the genome, we have something like that:

```
> 31_3  Len = 25
  chrVII     707119          1         25
  chrII      643009          1         25
  chrXVI     775770          1         25
  chrXVI     435897          1         25
```

where each line after the header `> 31_3  Len = 25` (that gives you the probe id, as previously defined), gives details on the location and the nature of the match.

NMmpa From the file `NMmp2g.out` we will then create as many files as the number of genome sequences (or chromosomes) per match strand (+ or -). This is done by running the program `NMmpa` which is a C implementation of the R script `makeProbeAnno` included in the R package distributed by David et al. [2].

So, let's do it:

```
$NMmpa -i NMmp2g.out -o NMmpa -a 2560
```

where the option `-a 2560` indicates that the array is a squared matrix of 2560×2560 probes. Once the program ends, you can see that a new folder called `NMmpa` has been created and that its folder contains files with extension `.prb` which names start by the corresponding chromosome name (chrI, chrII, etc...) suffixed by the strand of the match. So if a probe have a match in positive strand (+) on chromosome chrI, the match will be listed inside the file `chrI+.prb`.

Let's now have a look at these files:

```
$head -10 NMmpa/chrI+.prb
86_3 7767 114699 25 0
93_3 7774 27052 25 3
93_3 7774 25702 25 3
208_3 7889 15351 25 3
510_3 8191 32509 25 0
```

```
860_3 8541 192923 25 0
894_3 8575 193187 25 0
1061_3 8742 79283 25 0
1171_3 8852 144051 25 0
1275_3 8956 541 25 3
```

As you can see, all these files are formated as tables of 5 columns sepearated by a whitespace. The meaning of the columns is:

1. the (x, y) probe coordinates on the array,
2. the probe id,
3. the starting position (bp) of the match on the sequence,
4. the length of the match,
5. the uniqueness status of the match, defined as follows:
   - 0 unique perfect match
   - 1 multiple perfect matches
   - 2 unique near match
   - 3 multiple near matches

NMepm Since we are interested here only by probes that have an unique perfect match (PM) on the genome, we will use the `NMepm` perl script to extract these probes from the files previously generated by `NMmpa`. Before running `NMepm` we first need to generate a simple file that lists the name of the chromosomes for which we want to extract the PM probes. Here we will consider all the chromosome sequences. To get this file we can simply extract the chromosome names from the fasta file as follows:

```
$grep '^>' ../../Data/BY_S288c/Sequence/Genome.fasta \
      | grep -vw chrMito \
      | sed 's/^>//' > ../../Data/BY_S288c/Sequence/Chromosomes.txt
```

Note that here we removed the mitochondrial chromosome (chrMito) from the list. We can then run `NMepm` as follows:

```
$NMepm -i NMmpa/ -c ../../Data/BY_S288c/Sequence/Chromosomes.txt \
       -o NMepm -t 4 2>NMepm.log
```

where recall `NMmpa` contains the probe per strand annotation files previously generated with `NMmpa`, and `-t 4` indicates to `NMepm` that we want probe genomic coordinates tiled with unit step of 4bp. As you can see, the program as created a directory called `NMepm` (`-o NMepm`) which contains as many `.prb` files as the number or chromosomes listed in `../../Data/BY_S288c/Sequence/Chromosomes.txt`. Let's have a look at one this file:

```
$head -10 NMepm/chrI.prb
2147 833 2134628 chrI - 41 66 53 55 2
2199 1361 3486360 chrI + 45 70 57 59 2
953 1169 2993594 chrI - 49 74 61 63 2
1530 713 1826811 chrI + 53 78 65 67 2
1417 1001 2563978 chrI - 57 82 69 71 2
771 1239 3172612 chrI + 61 86 73 75 2
1745 1095 2804946 chrI - 65 90 77 79 2
991 1953 5000672 chrI + 69 94 81 83 2
295 1987 5087016 chrI - 73 98 85 87 2
540 1207 3090461 chrI + 77 102 89 91 2
```

All these `.prb` files are 10 columns table separated by a single whitespace. The columns are

1. the x coordinate of the probe onto the array
2. the y coordinate of the probe onto the array
3. the unique probe id
4. the name of the sequence on which the probe is located
5. the strand of the probe match (here remember that this is a unique perfect match)
6. the starting position of the match on the sequence
7. the ending position of the match on the sequence
8. the midpoint of the probe on the sequence
9. the *tiling* point of the probe on the sequence

10. the shift between the midpoint and the *tiling* point ranging from -3 to 3.

Let's explain the meaning of the last two columns. Since we are dealing here with tiling array, it is convenient for further analyses to have a probe coverage which respects everywhere the tiling step (here 4bp). That is to say, the physical distance between any pair of probes located on the same chromosome has to be a multiple of the tiling step. To insure this constrain is fulfill everywhere in the genome, the program `NMepm` implements a simple dynamic algorithm that first identifies chunks of consecutive probes that respect this constrain and then tries to optimize the positionning of these chunks with respect to each other so that it obtains the optimal positionning without shifting the individual probe midpoints by more than 3bp. The final position of the probe is then called the *tiling* point.

Finally, we will extract only 4 columns from these `.prb` files in order to create a unique file which could be used by `NMcl2tab` to extract data from raw Affymetrix `.CEL.` files. To do that, let's use a simple inline command:

```
$awk '{print $4" "$9" "$1" "$2}' NMepm/*.prb > BY_S288c.prb
```

Creating a general script Since we are now familiar with the different stages of the PM probe annotation pipeline, we can now try to create a shell script that will help us to quiclky reproduce these steps, in particular for a new strain (provided that we have the genome sequence of this strain, of course). The script could then look like that:

```
#! /bin/bash

##
# Arguments
##
strain=$1         # The name of the strain
genome_fasta=$2   # The genome sequence in fasta format
probe_fasta=$3    # The probe sequences in fasta format
array_size=$4     # The dimension of the array (yeast tiling array = 2560)
output_dir=$5     # The output directory

output_dir=${output_dir}/${strain}
##
# Create the output directory
##
mkdir -p $output_dir
```

```
##
# Map the probes on the genome using MUMmer
##
echo -n "Map probes on the genome: ..."
NMmp2g $genome_fasta $probe_fasta $output_dir/NMmp2g.out 2>$output_dir/NMmp2g.log
echo " [ OK ]"
# Extract the probes that have matche(s) on the genome and output the result
# by chromosome and by strand (+/-) into folder NMmpa
echo -n "Extract and format probe matches: ..."
NMmpa -i $output_dir/NMmp2g.out -o $output_dir/NMmpa -a $array_size
echo " [ OK ]"
# Now, extract only probes that have a unique perfect match
# on the genome (both strands) and remobed probe which do not match at an
# expected tiling step (4bp).
echo -n "Extract and format PM probes: ..."
grep '^>' $genome_fasta | sed 's/^>//' > $output_dir/seqid.txt
NMepm -i $output_dir/NMmpa -c $output_dir/seqid.txt -o $output_dir/NMepm -t 4 2>$output_dir/N
awk '{print $4" "$9" "$1" "$2}' ${output_dir}/NMepm/*.prb > ${output_dir}/${strain}.prb
echo " [ OK ]"
```

This bash script has been implemented in the program `NMstrain2array` that we can use now to get the list of PM probes on RM_11-1a (RM) genome.

```
$NMstrain2array RM_11-1a ./Data/RM_11-1a/Sequence/Genome.fasta \
                ./Data/Affymetrix/S.cerevisiae_tiling.fasta 2560 S1
Map probes on the genome: ... [ OK ]
Extract and format probe matches: ... [ OK ]
Extract and format PM probes: ... [ OK ]
```

Finally, we have now two files, `BY_S288c.prb` and `RM_11-1a.prb` that give us the probes that have a unique PM on each genome. We can then get an idea of the number of probes that are shared between the two strains as follows:

```
$cat S1/BY_S288c/BY_S288c.prb S1/RM_11-1a/RM_11-1a.prb \
     | cut -d ' ' -f 3,4 | sort | uniq -c \
     | awk '{print $1}' | sort | uniq -c
 368639 1
2501942 2
```

So,BY and RM share 2,501,942 probes and there are 368,639 probes that are specific to either BY and RM. We will now use these two files for the next section.

3 Pre-processing of raw Affymetrix .CEL files

Before running the analyses using the tiling array data sets, we need to extract and to format the raw probe hybridzation signals from the `.CEL` files produced by the Affymetrix platform.

To do so, we need to have the text version of the `.CEL` files and not the binary version. If you don't know which version you have, you can easily check your files by using the command `‘file [my_file]‘`. If the output is something like `data` or `ASCII text` then it's ok. Otherwise if the output contains the term `binary` you have first to convert the `.CEL` files by using the `.CEL` converter tool released by Affymetrix (you can download a Windows version.of this tool from our website `http://www.ens-lyon.fr/LBMC/gisv/ /index.php?option=com_content&task=view&id=37&Itemid=27`).

3.1 Extracting data from .CEL files

So, we assume now that all the `.CEL` files are in text format as they are in each directory `Array` of the strain data folders of this manual (see `Data`).

NMcl2tab The C program `NMcl2tab` will help us to extract all the probe hybridization measurements from the `.CEL` files and to format them into a useful table. Before running the program, we need to write a short configuration file that will indicate to `NMcl2tab` which files he has to consider. The configuration file consists in a table of two columns: the left column is for the id of the experiment when the right column gives the name of the corresponding file.

If you have a look inside directory `Data/BY_S228c/Array` you should see a file called `NucOccupancy.conf`. Let's have a look at this file:

```
$cat Data/BY_S228c/Array/NucOccupancy.conf
CBY01   CBY01-II.CEL
CBY02   CBY02-I.CEL
CBY08   CBY08-II.CEL
```

So, here we will extract the probe hybrization signal from 3 `.CEL` files and each dataset will have its own unique name (CBY01, CBY02 and CBY08). Let's do it:

```
$ NMcl2tab -c Data/BY_S288c/Array/NucOccupancy.conf \
```

```
          -d Data/BY_S288c/Array/ -p S1/BY_S288c/BY_S288c.prb \
          -o Data/BY_S288c/Array/NucOccupancy_raw.txt
--
 extractAffy
--
 CEL description from file [ Data/BY_S288c/Array/NucOccupancy.conf ]
 CEL directory              [ Data/BY_S288c/Array/ ]
 Probe file                 [ S1/BY_S288c/BY_S288c.prb ]
--
 Read probe set
--
Chromosome chrIII : 72257 probes [ OK ]
Chromosome chrII : 192607 probes [ OK ]
Chromosome chrI : 46546 probes [ OK ]
Chromosome chrIV : 355204 probes [ OK ]
Chromosome chrIX : 100626 probes [ OK ]
Chromosome chrVIII : 126301 probes [ OK ]
Chromosome chrVII : 255874 probes [ OK ]
Chromosome chrVI : 61945 probes [ OK ]
Chromosome chrV : 134902 probes [ OK ]
Chromosome chrXIII : 219184 probes [ OK ]
Chromosome chrXII : 242748 probes [ OK ]
Chromosome chrXI : 162453 probes [ OK ]
Chromosome chrXIV : 182756 probes [ OK ]
Chromosome chrX : 170169 probes [ OK ]
Chromosome chrXVI : 220699 probes [ OK ]
Chromosome chrXV : 257613 probes [ OK ]
--
 Import data from .CEL files
--
For experiment CBY01:
  Read data from CEL file CBY01-II.CEL: [ OK ]
For experiment CBY02:
  Read data from CEL file CBY02-I.CEL: [ OK ]
For experiment CBY08:
  Read data from CEL file CBY08-II.CEL: [ OK ]
--
 Export data
--
```

So now, you should have a new file in `Data/BY_S288c/Array` called `NucOccupancy_raw.txt`. Let's have a look at this file:

```
$head -10 Data/BY_S288c/Array/NucOccupancy_raw.txt
chromosome      position        CBY01   CBY02   CBY08
chrIII  38      1.470300e+04    5.044000e+03    1.227300e+04
```

```
chrIII  42      7.693000e+03    1.226100e+04    1.193100e+04
chrIII  226     2.153200e+04    3.195700e+04    3.193200e+04
chrIII  230     7.490000e+03    2.793000e+03    1.007000e+04
chrIII  234     2.126100e+04    3.004300e+04    3.353300e+04
chrIII  238     9.659000e+03    4.564000e+03    8.916000e+03
chrIII  242     1.642300e+04    2.709500e+04    2.464800e+04
chrIII  278     1.581800e+04    6.362000e+03    1.652100e+04
chrIII  358     7.156000e+03    2.731000e+03    7.374000e+03
```

which is in fact a simple table with 5 columns: the two first columns indicate the genomic coordinates of the (probe) data point and the following columns (here 3) report the hybridization intensity as given in each original `.CEL` file. Note that the names of the 3 last columns correspond to the ones specified in the configuration file `NucOccupancy.conf`.

3.2 Normalizing signal

3.2.1 Using NMcl2tab

It is possible to perform the normalization of the tiling arrays at the same time than the extraction from the `.CEL` files. The program `NMcl2tab` proposes 3 types of quantile-based normalization procedure:

- `qq`: quantile normalization,
- `lqq`: log2 transformation followed by quantile normalization,
- `qql`: quantile normalization followed by log2 transformation.

So, suppose we want to quantile normalize the previous arrays using the `qql` procedure. We can then do that by using the `-n` short option as follows:

```
$ NMcl2tab -c Data/BY_S288c/Array/NucOccupancy.conf \
           -d Data/BY_S288c/Array/ -p S1/BY_S288c/BY_S288c.prb \
           -o Data/BY_S288c/Array/NucOccupancy_raw.txt
           -n qql
```

3.2.2 Using another program

Once the data have been extracted using `NMcl2tab`, the outputed file can be directly used by other softwares, in particular by **TileMap** [3] which is available at `http://biogibbs.stanford.edu/~jihk/TileMap/index.htm`.

Normalization can then be done by using the program `tilemap_norm` from the **TileMap** package (note that this program implements only a quantile normalization).

We provide in the directory `Data/BY_S288c/Array/` a template of the configuration file required by `tilemap_norm` to run:

```
$cat Data/BY_S288c/Array/NucOccupancy.norm
##############################
# TileMap Normalization      #
##############################

[Working directory] = .
[Raw Data file]     = NucOccupancy_raw.txt
[Export file]       = NucOccupancy_norm.txt
[Array number]      = 3
[Truncation lower bound before normalization] = -1000000000.0
[Take log2 transformation before normalization] (1:yes; 0:no) = 1
```

We invite you to have a look at the **TileMap** manual to have more details on this file. Finally, the normalization can be done like this (assuming that you have previously added to your `$PATH` variable the location of the **TileMap** binaries):

```
$cd Data/BY_S288c/Array/
$tilemap_norm NucOccupancy.norm
```

You then should have a new file in `Data/BY_S288c/Array/` called `NucOccupancy_norm.txt` that have the same format than `NucOccupancy_raw.txt` but for which the hybridization intensities have been normalized.

3.3 Creating a NucleoMiner .db file

Most of the program implemented in **NucleoMiner** used a binary version of the tiling array datasets. To convert a tiling array dataset, as obtained previously, we will use the program `NMtfb`. This program requires a small configuration file that describes the organization of the dataset and if some simple pre-processing is needed to create the final dataset (that will then be used in subsequent analyses with **NucleoMiner**).

Since the next step of the tutorial deals with the inference of nucleosome occupancy, we will then prepare the corresponding dataset. As we will see in the next section, the inference of nucleosome occupancy is perfomed by

fitting a Hidden Markov Model (HMM) to the average hybrization signal among replicates (and of course per strain).

The program `NMtfb` can be directly used to compute the average hybrization intensities and to store the result in an appropriate binary file that could be used in the next step. To do so, we need first to create the configuration file. For simplicity, we have created this file for you: `Data/BY_S288c/Array/NucOccupancy.dbconf`. Let's have a look at its content

```
cat Data/BY_S288c/Array/NucOccupancy.dbconf
############################
# NMtdb configuration file #
############################
##
# The name of the dataset
##
name = BY_NucOccupancy
##
# Number of arrays to consider
##
narray = 3
##
# Membership of the arrays
##
groupId = 1 1 1
##
# Pre-processing algorithm
#    full   = do nothing
#    mean   = compute the mean across replicates (per group)
#    median = compute the median accross replicates (per group)
##
type = mean
```

As you can, by using the `mean` tag in the pre-processing section of the configuration file, we will ask `NMtdb` to compute for each probe of the input dataset the mean hybridization value and to consider it as the final datapoint. Let's run `NMtdb`:

```
$NMtdb -i Data/BY_S288c/Array/NucOccupancy_norm.txt \
       -c Data/BY_S288c/Array/NucOccupancy.dbconf \
       -o Data/BY_S288c/Array/NucOccupancy_norm.db
--
 tilingdb
```

```
--
 Read configuration file: [ OK ]
--
 Read data file: [ OK ].
--
 Save data: [ OK ]
--
```

Note that the `Read data file:` step can take some times depending on how big is the input dataset. The program has then created a new file `Data/BY_S288c/Array/NucOccupancy_norm.db` which is in fact a binary file,

```
$file Data/BY_S288c/Array/NucOccupancy_norm.db
Data/BY_S288c/Array/NucOccupancy_norm.db: data
```

So, don't try to open this file with a text editor or something else: it is very important to not modify this file by hand, otherwise you will experience some troubles in subsequent analyses that require this file. Besides, since this file is a binary file, it is specific to your machine architecture and operating system, so it is highly recommanded to not send this file to other users, unless they have exactly the same machine architecture and operating system. Unless, send the original input file togeter with the configuration file and the corresponding arguments of `NMtdb`.

Finally, we can have a look to the content of `Data/BY_S288c/Array/NucOccupancy_norm.db` by using the flag option `--print` of `NMtdb` as follows:

```
$NMtdb -i Data/BY_S288c/Array/NucOccupancy_norm.db --print | head -10
chromosome position A1
chrIII 38 1.332212e+01
chrIII 42 1.333482e+01
chrIII 226 1.473298e+01
chrIII 230 1.263931e+01
chrIII 234 1.470365e+01
chrIII 238 1.290735e+01
chrIII 242 1.438298e+01
chrIII 278 1.357645e+01
chrIII 358 1.241799e+01
```

As you can see there is only one column of hybrization intensity values since we ask `NMtdb` to compute the mean from the three replicates.

Practice Do the same data preprocessing for strain RM_11-1a.

4 Inferring nucleosome occupancy

In this section, we will learn how to use **NucleoMiner** in order to infer nucleosome positions from our tiling array dataset. Recall that **NucleoMiner** implements a custom version of the Hidden Markov Model devised by [6], similar also to the one used by [4].

4.1 Signal with unpredictable trends

Here, in order to remove unpredictable trends in the hybridization signal we will use sliding windows to fit localy the HMM (as in [6]). Thus, independent run of the HMM will be successively applied in window of 1kb (i.e. $\sim$250 probes) all along the genome. The model parameters and posteriors of all windows containing a fixed probe will be then averaged and used for a global computation of both state probabilities and most-likely states (among well-positioned nucleosome, fuzzy nucleosome and linker). As we used only probes with unique and perfect matches, we also allowed the HMM to deal with missing data. State probabilities and most-likely states of "missing probes" will be computed in the same way than for observed probes, taking advantage of neighboring observed information.

Inferring nucleosome positionning with **NucleoMiner** requires two steps. The first one consists in fitting the HMM to the data (`NMhmmfit`) and the second one applies the Viterbi algorithm (`NMhmmvit`) to the averaged posterior probabilities as computed and outputed by `NMhmmfit`.

NMhmmfit We assume here that, as described in the previous section, you have created for each strain a data file in **NucleoMiner** format using `NMtdb`, so that binary file `Data/BY_S288c/Array/NucOccupancy_norm.db` contains the average signal among (biological) replicates from the nucleosome mapping chip experiment.

For speed consideration, and since the process is the same for each chromosome, we will fit here the HMM only on chromosome I of BY_S288c (using the `-c` option of `NMhmmfit`).

```
$NMhmmfit -i Data/BY_S288c/Array/NucOccupancy_norm.db \
          -o S2/BY_S288c/NMhmmfit -c chrI
>chrI|nprobe=46546|nwindow=57515|coverage=0.81|
gdl_tiling_hmm_baum_welch stopped: the tiling sequence cannot be built from the HMM!
gdl_tiling_hmm_baum_welch stopped: the tiling sequence cannot be built from the HMM!
```

```
gdl_tiling_hmm_baum_welch stopped: the tiling sequence cannot be built from the HMM!
gdl_tiling_hmm_baum_welch stopped: the tiling sequence cannot be built from the HMM!
gdl_tiling_hmm_baum_welch stopped: the tiling sequence cannot be built from the HMM!
0%|=          | 10%
```

As you can see, sometimes the message

```
gdl_tiling_hmm_baum_welch stopped: the tiling sequence cannot be built from the HMM!
```

appears. Don't worry too much about this message: it is just a warning. It means that for a given sliding window, the program was unable to fit the HMM to the data within that window. For example, this can happen when the window contain a large amount of missing data, or the data quality, or more precisely the signal to noise ratio is low.

Now, as you can see, the program `NMhmmfit` has created a new directory `S2/BY_S288c/NMhmmfit/` which contains a new file:

```
$ls S2/BY_S288c/NMhmmfit/
chrI.nhw
```

Don't try to open this file with a text editor. Again this is a binary file,

```
$file S2/BY_S288c/NMhmmfit/chrI.nhw
S2/BY_S288c/NMhmmfit/chrI.nhw: data
```

and you must not modify this file, otherwise you will experiment troubles when using `NMhmmvit`.

NMhmmvit Once the `.nhw` files have been generated by `NMhmmfit`, you can use first `NMhmmvit` to have a look at the content of these binary files. To do so, just type the following line:

```
$NMhmmvit -i Data/BY_S288c/Array/NucOccupancy_norm.db \
         -r S2/BY_S288c/NMhmmfit/ -c chrI \
         --print 1>S2/BY_S288c/NMhmmfit/chrI.txt
```

where the output is redirected to the file `S2/BY_S288c/NMhmmfit/chrI.txt`. Let's look at this file, at least at its header:

```
$head -1 S2/BY_S288c/NMhmmfit/chrI.txt
chromosome probe position virtual ignore chunk mu(0) var(0) mu(1) var(1) pr(in,0) pr(in,0)
pr(in,0) pr(in,0) pr(in,0) pr(in,0) pr(in,0) pr(in,0)
pr(in,0) pr(out,0) pr(out,0) pr(out,0) pr(out,31) pr(out,32)
pr(out,33) pr(out,34) pr(out,35) pr(out,36) pr(out,37) pr(out,77)
pr(0,0) pr(0,1) pr(0,2)
```

As you can see, the file is organized as a table which columns are seperated by single whitespace. The three columns (*chromosome, probe and position*) give the location of the current probe, the fourth column, *virtual*, is either '0' (for no) or '1' (for yes) depending on the status of the current probe. A probe is said to be "virtual" if no probe has been reported at that position so that results for that position have been imputed by the HMM. The fith column, *ignore* is either 'y' (for yes) or 'n' (for no). If a probe is flagged as to be ignored, it generally means that `NMhmmfit` was unable to fit the HMM around this probe. The sixth column, *chunk*, gives the index of the chromosome chunk to which the probe belongs. In fact, it is common that the HMM cannot be really run on the entire chromosome (e.g. due to the presence of long repeat regions that introduce a long break within the tiling coverage) but rather on contiguous pieces of the chromosome, here denominated as *chunks*. Then, the following columns depends on the HMM parametrization. The four first columns, (*mu(0), var(0), mu(1), var(1)*, give the mean and variance estimates for the two gaussian distributions used in the HMM, one for the linker state (*mu(0), var(0)*) and one for the nucleosomal state, either well-positioned or fuzzy (*mu(1), var(1)*). The following 23 columns correspond to the estimates - at the current probe - of the transition probabilities of the HMM. Since we used here the default parametrization of the HMM (i.e, at least 31 probes and at most 38 probes for a well-positionned nucleosome), these 23 columns are ordered as described in Figure 1.

So, as you can see, we have all the information in hands to determine the most likely positions of the nucleosomes along the chromosome. This is done by applying the Viterbi algorithm as implemented in `NMhmmvit`. Let's do it:

```
$NMhmmvit -i Data/BY_S288c/Array/NucOccupancy_norm.db \
          -r S2/BY_S288c/NMhmmfit -c chrI \
          -o S2/BY_S288c/NMhmmvit 2>/dev/null
$head -10 S2/BY_S288c/NMhmmvit/chrI.nuc
chromosome probe position virtual ignore chunk calling s1 p1 p2 p3
chrI 1 55 0 n 1 0 10.0845 0.000 0.585 0.414
chrI 2 59 0 n 1 0 10.0594 0.000 0.553 0.447
chrI 3 63 0 n 1 0 10.0315 0.333 0.369 0.298
chrI 4 67 0 n 1 0 9.88305 0.500 0.277 0.223
chrI 5 71 0 n 1 0 9.72784 0.600 0.221 0.179
chrI 6 75 0 n 1 0 9.59281 0.667 0.184 0.149
chrI 7 79 0 n 1 0 9.47126 0.714 0.158 0.128
chrI 8 83 0 n 1 0 9.36506 0.750 0.138 0.112
chrI 9 87 0 n 1 0 9.27393 0.778 0.123 0.099
```

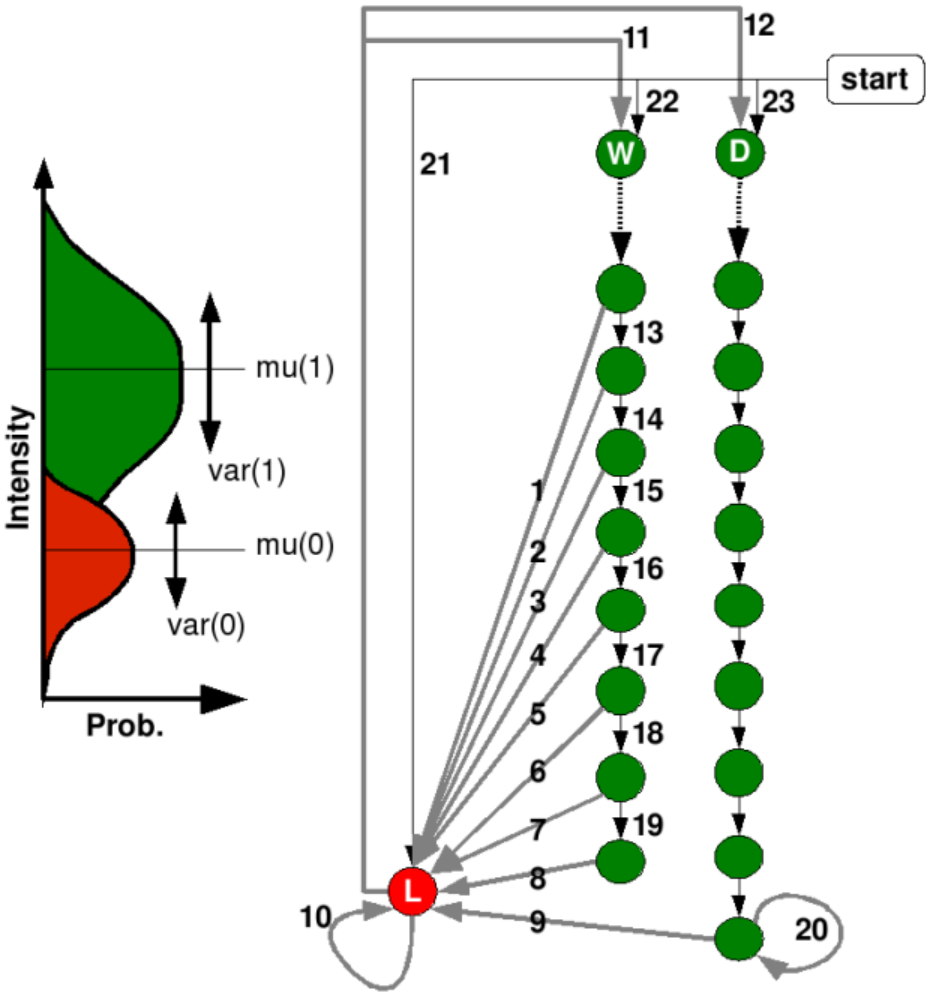

Figure 1:

Now, you should have a new directory `S2/BY_S288c/NMhmmvit/` which contains a new file `chrI.nuc`. As you can see, this file is again a simple table which columns are seperated by a single whitespace. As for the `--print` option, the first six columns give details about the current probe location and its status. Follow four columns that summarize the result of `NMhmmvit`:

- *calling* is 0,1,2 depending on the most likely state at the current probe,
  - 0 linker,

  - 1 fuzzy nucleosome,
  - 2 well-localized nucleosome,

- *s1* is the smoothed hybridization signal using the output of the HMM,
- *p1* is the posterior probability that the current probe belongs to a linker,
- *p2* is the posterior probability that the current probe belongs to a fuzzy nucleosome,
- *p3* is the posterior probability that the current probe belongs to a well-localized nucleosome.

```
$NMhmm2plot -i S2/BY_S288c/NMhmm2plot.config \
            -c chrI -s 100000 -e 105000 -o S2/BY_S288c/
```

This creates two text file (`.txt`) in the directory `S2/BY_S288c/` with the same prefix obtained by concatenating the name of the chromosome and the starting and the ending positions, namely `chrI_100000_105000`. The file suffixed by `_d` contains a table of 5 columns separated by a single whitespace,

```
$ head -10 S2/BY_S288c/chrI_100000_105000_d.txt
100003 6.61 8.25 0.79 1.00
100007 NA 8.27 0.79 1.00
100011 NA 8.28 0.80 1.00
100015 8.73 8.27 0.79 1.00
100019 6.52 8.15 0.73 1.00
100023 7.39 8.11 0.72 1.00
100027 6.85 8.09 0.71 1.00
100031 7.00 8.08 0.70 1.00
100035 6.84 8.08 0.70 1.00
100039 6.42 8.08 0.71 1.00
```

where the first column is the position of the interrogated point, the second one the average hybrization signal for the corresponding probe (`NA` when missing), the third one the smoothed signal (by the HMM), the fourth one gives the probability to be within a nucleosome (either fuzzy or delocalized) and the fifth column the most likely state at this point (as computed by `NMhmmvit`).

As depicted in Figure 2, we can then use the NucleoMiner R package to plot these data by using the R function `NMhmm2plot` (see the documentation for more details).

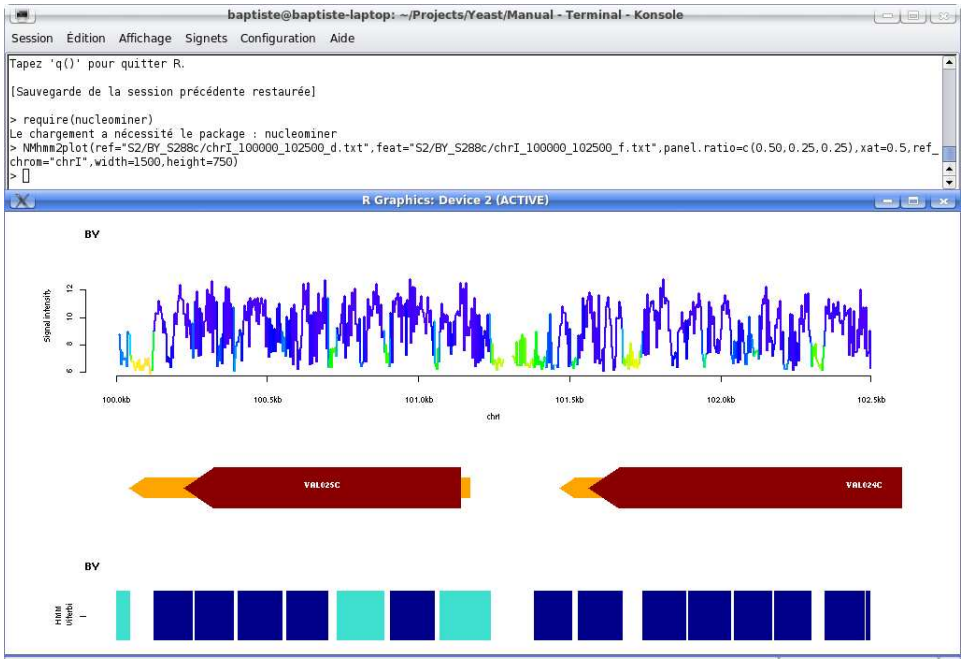

Figure 2:

Practice Fit the HMM for the remaining chromosomes of BY_S288c and the entire genome of RM_11-1a.

4.2 Detrended signal

Sometimes it is possible to use additional data to detrend the signal so that the sliding window strategy as used previously is no longer required to fit the HMM. For example, [4] used data from tiling array on which they hybridized the full DNA (including DNA from nucleosomes and linkers) to remove the probe effect and so to normalize their signal. In that case, one can be interested to either fit the HMM on the entire chromosome without using sliding window, or when nucleosome positions are already known on some regions, to train the HMM on these region and then to use the trained version of the HMM to predict nucleosome positions elsewhere on the genome.

TODO

4.3 Threading the inference process

Using sliding windows to fit the HMM can lead to very long (in time) and heavy (in cpu/memory) processes. One simple way to speed up t process, is to split the inference by chromosome.

Linux/Unix clusters with qsub utility If you have a cluster with qsub utility to submit jobs, you have to write two short scripts. The first one will just loop on chromosome and submit the job to the cluster for the current chromosome,

```
$cat NMQsubHmm.sh
#!/bin/bash

QUE = $1
# man qsub to obtain an explanation of the options
for i in "I" "II" "III" "IV" "V" "VI" "VII" "VIII" \
         "IX" "X" "XI" "XII" "XIII" "XIV" "XV" "XVI"
do
   qsub -m b -m e -e ./chr${i}.err -q $QUE -o ./chr${Roman}.out ./NMQsubHmmRun.sh chr${i}
done
```

where the program takes as single argument the name of the queue where you want to submit all your jobs (note that `chr${i}` is the name of the chromosome, so modify the elements of the loop or the stem name (chr) if the sequence names are different). The second program is a very short piece of code that just call `NMhmmfit` (to modify if you want to use different arguments that the one specified by default).

```
$cat NMQsubHmmRun.sh
#!/bin/bash
#$ -cwd
#$ -V

chr=$1
#
NMhmmfit -i ./data.db -o NMhmmfit -c ${chr}
```

4.4 Averaging nucleosome occupancy profiles around genomic features

An interesting issue with nucleosome occupancy profile is to look at the pattern of nucleosome density around specific genomic features like for example

the transcriptions start sites (TSS) or end sites (TES). We can also be interested in computing average nucleosome occupancy profiles over cis-regions like the entire cis-region of genes. To do so we need first to reformat the output of `NMhmmvit` into a tiling array dataset like format. This is done by the perl script `NMnuc2toc` as follows:

```
$NMnuc2toc -i S2/BY_S288c/NMhmmvit 1>S2/BY_S288c/BY_S288c_nuc.txt
```

Then, as for the original dataset, we have now to convert this file into a `.db` file by using `NMtdb`:

```
$NMtdb -i S2/BY_S288c/BY_S288c_nuc.txt \
       -c S2/BY_S288c/BY_S288c_nuc.dbconf \
       -o S2/BY_S288c/BY_S288c_nuc.db
```

Now, we will create a new folder `S2/BY_S288c/Features` to store the results for any feature we want to investigate. Let's look first at the gene transcript boundaries (TSS and TES):

```
$mkdir -p S2/BY_S288c/Features/Transcript
$echo "transcrpit ." >S2/BY_S288c/Features/Transcript/feature_id.txt
```

The file `S2/BY_S288c/Features/Transcript/feature_id.txt` is required by `NMt2feat`. It tells to the program that we are interested only in feature of type *transcript*, whatever the ID of the feature (that's why we used the wildcard '.' after the transcript tag).

NMt2feat We can then run `NMt2feat` as follows:

- TSS (`--start`)

  ```
  $NMt2feat -i S2/BY_S288c/BY_S288c_nuc.db \
            -o S2/BY_S288c/Features/Transcript/TSS \
            -f Data/BY_S288c/Features/features.gff \
            -l S2/BY_S288c/Features/Transcript/feature_id.txt \
            -w 500 -n 50 --start
  ```

- TES (`--end`)

  ```
  $NMt2feat -i S2/BY_S288c/BY_S288c_nuc.db \
            -o S2/BY_S288c/Features/Transcript/TES \
            -f Data/BY_S288c/Features/features.gff \
            -l S2/BY_S288c/Features/Transcript/feature_id.txt \
            -w 500 -n 50 --end
  ```

where `-w 500 -n 50` indicates to `NMt2feat` that we want to consider 500bp from either side of the feature boundary and that we want to split these 500bp into 50 bins of equal sizes (i.e here 10bp). As you can see, the program `NMt2feat` creates a file suffixed by `_avg.txt` which is organized as a table with 6 columns separated by a single whitespace. Let's have a look at this table for the TSS:

```
$head -10 S2/BY_S288c/Features/Transcript/TSS_avg.txt
start 0 -500 -490 9330.19 11026
start 1 -490 -480 9296.27 11079
start 2 -480 -470 10313.1 12316
start 3 -470 -460 9256.08 11084
start 4 -460 -450 10371.6 12320
start 5 -450 -440 9372.42 11092
start 6 -440 -430 10484.1 12331
start 7 -430 -420 9491.17 11098
start 8 -420 -410 10593.3 12333
start 9 -410 -400 9552.06 11102
```

The columns indicate:

1. the interrogated feature boundary (here start=TSS),
2. the bin index (here from 0 to 99),
3. the starting position of the bin,
4. the ending position of the bin,
5. the total amount of signal into that bin,
6. the total number of probes falling into that bin.

Using the function `NMt2featplot` of the R pcackage, we can easily plot the content of the `_avg.txt` file. An exemple for the TSS is depicted in Figure 3.

NMt2featcis This program is an extension of `NMt2feat` which allows us to look at the entire *cis*-profile around a given genomic feature.

```
$NMt2featcis -i S2/BY_S288c/BY_S288c_nuc.db \
             -o S2/BY_S288c/Features/Transcript/txnuc \
             -f Data/BY_S288c/Features/features.gff \
             -l S2/BY_S288c/Features/Transcript/feature_id.txt \
             -u 500 -d 500 -p 50 -w 50 -s 10
```

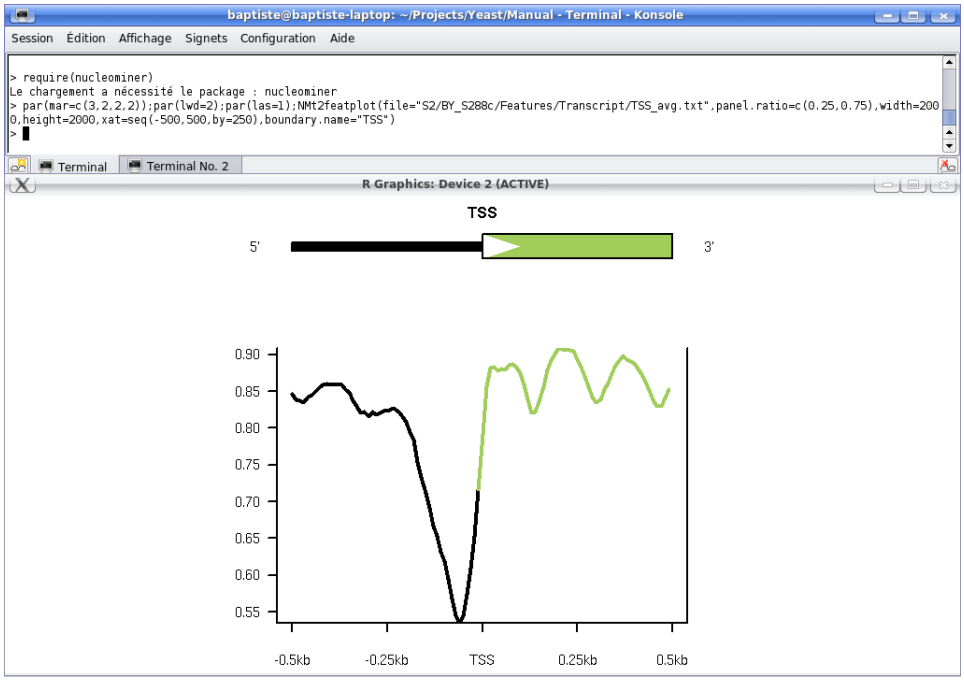

Figure 3:

Here we asked `NMt2featcis` to compute the *cis*-profile around all the transcripts annotated in BY genome by using a upstream and downstream region of 500bp (`-u 500 -d 500`) each divided into 50 bins of equal size (`-p 50 -w 50`) and by splitting the transcript region in 10 equal parts, whatever the size of the transcript region (`-s 10`). The program has created a file `S2/BY_S288c/Features/Transcript/txnuc_cis.txt` which is organized as a table with 9 columns separated by a single whitespace.

```
$head -10 S2/BY_S288c/Features/Transcript/txnuc_cis.txt
u 0 0 4710 0 10 10458.5 12350 4692
u 1 1 4710 10 20 9320.86 11108 4693
u 2 2 4710 20 30 10341.2 12352 4694
u 3 3 4710 30 40 9277.35 11113 4694
u 4 4 4710 40 50 10398.5 12356 4696
u 5 5 4710 50 60 9395.63 11121 4698
u 6 6 4710 60 70 10510.1 12364 4699
u 7 7 4710 70 80 9513.81 11125 4699
u 8 8 4710 80 90 10622.1 12366 4700
u 9 9 4710 90 100 9574.55 11129 4702
```

where the columns indicate:

1. the region:
    - `u` upstream the feature,
    - `i` inside the feature (here the transcribed region),
    - `d` downstream the feature,
2. the absolute bin index,
3. the relative bin index (w.r.t the region it belongs to),
4. the total number of interrogated features,
5. the starting position of the bin,
6. the ending position of the bin,
7. the total amount of signal into that bin,
8. the total number of probes falling into that bin,
9. the total number of features that contribute to that bin.

Finally, using the function `NMt2featcisplot` of the R pcackage, we can easily plot the content of the `_cis.txt` file. An exemple is depicted in Figure 4.

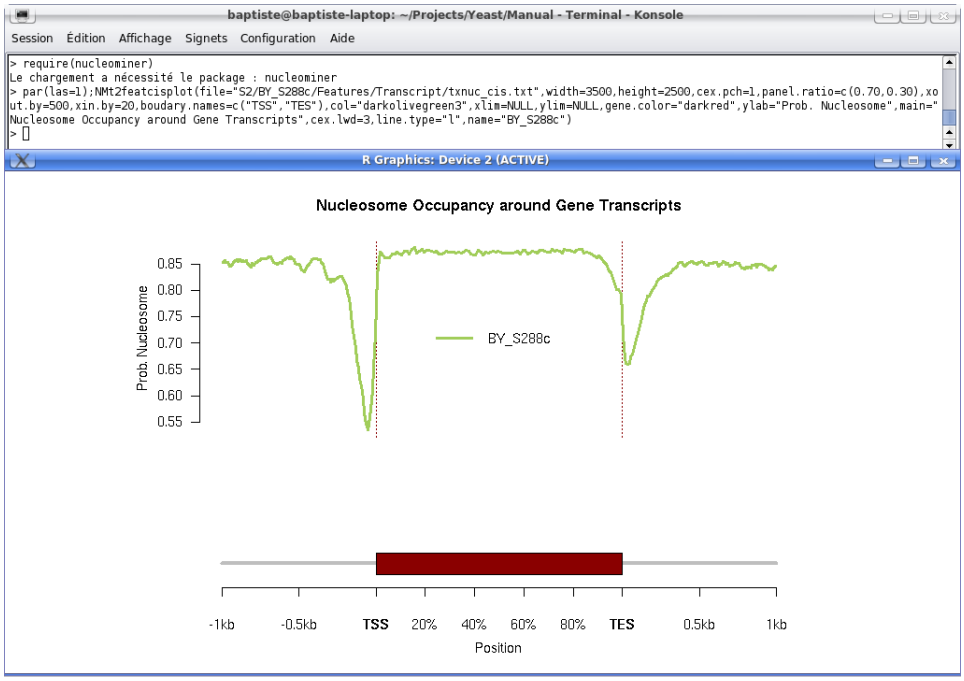
baptiste@baptiste-laptop: ~/Projects/Yeast/Manual - Terminal - Konsole
Session Édition Affichage Signets Configuration Aide
> require(nucleominer)
Le chargement a nécessité le package : nucleominer
> par(las=1);NMt2featcisplot(file="S2/BY_S288c/Features/Transcript/txnuc_cis.txt",width=3500,height=2500,cex.pch=1,panel.ratio=c(0.70,0.30),xout.by=500,xin.by=20,boudary.names=c("TSS","TES"),col="darkolivegreen3",xlim=NULL,ylim=NULL,gene.color="darkred",ylab="Prob. Nucleosome",main="Nucleosome Occupancy around Gene Transcripts",cex.lwd=3,line.type="l",name="BY_S288c")
>
R Graphics: Device 2 (ACTIVE)
Nucleosome Occupancy around Gene Transcripts
Prob. Nucleosome
0.85
0.80
0.75
0.70
0.65
0.60
0.55
BY_S288c
-1kb
-0.5kb
TSS
20%
40%
60%
80%
TES
0.5kb
1kb
Position

Figure 4:

5 Aligning genome sequences

Before aligning our nucleosome occupancy profiles (as computed in the previous section), we need first to align the DNA sequences of the two strains. We will then use the sequence alignments to initiate the nucleosome alignment. Note that it is difficult to say something about nucleosomes in regions where sequences have poor matches.

In this section, we will assume that the sequences of our strains are of relatively good quality and that the divergence between the two strains is modest, with eventually just a few and small genomic rearrangements (translocation, inversion, etc...). We will then use **MUMmer** to perform the sequence alignments, using one strain as the reference and the other as the query. However, the output of **MUMmer** needs to be reprocessed and reformated in order to get a clean picture of the whole alignment. This entire process can be realized by one **NucleoMiner** program, namely `NMgxcomp`, as follows:

```
$NMgxcomp Data/BY_S288c/Sequence/Genome.fasta \
          Data/RM_11-1a/Sequence/Genome.fasta \
          S3/BY_RM 2>S3/BY_RM.log
```

where the firt argument `Data/BY_S288c/Sequence/Genome.fasta` of `NMgxcomp` gives the genome sequence in fasta format of the strain used as reference (here BY), the second argument `Data/RM_11-1a/Sequence/Genome.fasta` gives the genome sequence in fasta format of the strain used as the query (here RM) and the last argument `S3/BY_RM` indicates the stem name/path of the output files.

As you can see, the folder `S3` contains now a bunch of files prefixed by `BY_RM`. Some are direct outputs from **MUMmer** programs, while those containing the suffix `_gxcomp` have been created by the script `NMmum2gxc` of **NucleoMiner**. Here, we will describe only these files.

The .out file The file `S3/BY_RM_gxcomp.out` is just an output of the progession of the script `NMmum2gxc`. Nevertheless, the outputed table give you a first overview of the nature of the raw alignment between the sequences of the two strains. Here, and in the subsequent files the prefix *cis* is used to describe a chromosome-to-chromosome relationship while the prefix *trans* is used to flag putative translocation events.

```
$ cat S3/BY_RM_gxcomp.out
```

```
##
# Table of relationships to explore
##
type ref qry T(match|ref) %(match|qry) %(match|ref) %(+|ref) %(-|ref)
cis     chrVIII supercontig_1.12        506622  0.938   0.900   1.000   0.000
trans   chrVIII supercontig_1.17        15760   0.029   0.028   0.000   1.000
cis     chrX    supercontig_1.8 684930  0.976   0.918   1.000   0.000
cis     chrVII  supercontig_1.2 1032008 0.993   0.946   0.000   1.000
cis     chrIX   supercontig_1.14        400804  1.000   0.911   0.000   1.000
cis     chrII   supercontig_1.6 785026  1.000   0.965   0.000   1.000
cis     chrXIV  supercontig_1.7 763497  1.000   0.973   0.043   0.957
cis     chrIII  supercontig_1.15        300505  0.965   0.949   0.971   0.029
cis     chrXV   supercontig_1.3 1034494 0.975   0.948   1.000   0.000
cis     chrVI   supercontig_1.16        231275  0.938   0.856   1.000   0.000
trans   chrVI   supercontig_1.8 13642   0.055   0.050   1.000   0.000
cis     chrXII  supercontig_1.10        566426  0.554   0.525   1.000   0.000
cis     chrXII  supercontig_1.13        441487  0.432   0.409   1.000   0.000
cis     chrXVI  supercontig_1.4 882793  0.964   0.931   1.000   0.000
cis     chrI    supercontig_1.17        165518  0.992   0.719   0.000   1.000
cis     chrXI   supercontig_1.9 641056  1.000   0.962   1.000   0.000
cis     chrV    supercontig_1.11        545509  0.998   0.946   1.000   0.000
cis     chrIV   supercontig_1.1 1471886 0.995   0.961   0.000   1.000
cis     chrXIII supercontig_1.5 884250  0.987   0.957   1.000   0.000
##
# Exploring cis relationships: wait...
##
##
# Exploring trans relationships:wait...
##
##
# Merge+Output results:wait...
##
```

The .map file The file `S3/BY_RM_gxcomp.map` gives an overview of the organization of the final sequence alignments. For each reference sequence, you have something like that:

```
>chrX [24352 - 729989]
chrX 24352 197492 cis + supercontig_1.8 1 29024 202068 99.14 0.23 0.24
chrX 204115 354637 cis + supercontig_1.8 1 202521 353040 99.59 0.20 0.21
chrX 354638 354832 cis + supercontig_1.8 1 353377 353570 97.07 0.00 0.00
chrX 355170 400123 cis + supercontig_1.8 1 353571 398451 99.17 0.06 0.06
chrX 400124 472455 cis + supercontig_1.8 1 398731 471082 99.31 0.10 0.10
chrX 483971 538095 cis + supercontig_1.8 1 471083 525249 99.36 0.07 0.07
chrX 538433 541087 cis + supercontig_1.8 1 525250 527879 97.71 0.00 0.00
```

```
chrX 541426 543597 cis + supercontig_1.8 1 527880 530052 99.08 0.00 0.00
chrX 543695 620997 cis + supercontig_1.8 1 530053 607313 99.42 0.10 0.11
chrX 621007 628798 cis + supercontig_1.8 1 607448 615241 99.12 0.01 0.01
chrX 628799 713797 cis + supercontig_1.8 1 615525 700524 99.68 0.11 0.12
chrX 715484 729989 cis + supercontig_1.8 1 702196 716707 99.47 0.02 0.02
*chrX - - - - supercontig_1.8 - - - - 0.92 0.95
```

where `>chrX [24352 - 729989]` indicates that the following section (until reaching a new `>` sign, like for a fasta file) deals with the alignments starting from position 24,352 (bp) to position 729,989 (bp) on reference sequence called chrX. Then comes a table into which each row stands for a full-length alignment (including moderate gaps). The column meaning is:

1. the name of the reference sequence,
2. the starting position of the alignment of the reference sequence,
3. the ending position of the alignment of the reference sequence,
4. the nature of the alignment,
5. the name of the query sequence involved in the alignment,
6. the strand of the alignment (1 is normal and -1 is reverse),
7. the starting position of the alignment of the query sequence,
8. the ending position of the alignment of the query sequence,
9. the percentage of identity,
10. the percentage of insertion (in the reference, or deletion in query),
11. the percentage of deletion (in the reference, or insertion in query).

The .poly file The file `S3/BY_RM_gxcomp.poly` lists the sequence polymorphisms as detected by the program from the alignments. There are three kinds of sequence polymorphisms: SNP, insertion (w.r.t the reference sequence) and deletion (w.r.t the reference sequence). We can then easily know how many SNPs (snp), insertions (ins) and deletions (del) are present in our alignment as follows:

```
$cut -d ' ' -f 3 S3/BY_RM_gxcomp.poly | sort | uniq -c
   3252 del
   2973 ins
  47817 snp
```

So here we have 47,817 SNPs, 2,973 insertions (of at least 1bp) and 3,252 deletions (of at least 1bp). If we now look at the way each polymorphism is described in the file `S3/BY_RM_gxcomp.poly` we see something like that:

```
$grep -w snp S3/BY_RM_gxcomp.poly | head -1
cis chrVIII snp 12719 T supercontig_1.12 2874 A 1
```

where the columns indicate

1. the nature of the alignment (see above),
2. the name of the reference sequence,
3. the kind of polymoprhism (here a SNP),
4. the position of the SNP in the reference sequence,
5. the allele of the SNP in the reference sequence,
6. the name of the query sequence,
7. the position of the SNP in the query sequence,
8. the allele of the SNP in the query sequence,
9. the length in bp of the polymorphism (always 1 for a SNP).

For an insertion we have

```
$grep -w ins S3/BY_RM_gxcomp.poly | head -1
cis chrVIII ins 14780 14780 supercontig_1.12 4934 - 1
```

where the fifth column corresponding previously to the allele of the SNP in the reference sequence indicates now the ending position of the insertion in the reference sequence, the eighth column corresponding to the allele of the SNP in the query sequence indicates is now empty (-) and the last column indicates the length of the insertion (here is a single base insertion). Finally, for a deletion we have:

```
$grep -w del S3/BY_RM_gxcomp.poly | head -1
cis chrVIII del 17132 - supercontig_1.12 7286 7286 1
```

where the fifth column is now empty (-), the eighth column indicates the ending position of the deletion in the querty sequence and the last column indicates the length of the deletion (here is a single base deletion).

The .gff file The file `S3/BY_RM_gxcomp.gff` contains the same information than the .poly file but formating in gff version 3, so that you can simply plug this file into your favorite genome browser to see where the polymoprhisms are all along the query sequence.

The .c2c file The file `S3/BY_RM_gxcomp.c2c` is a simple table that describes how the genome sequence can be aligned. This file will be very usefull in the next step of this tutorial. Let's have a look to its content:

```
$head -10 S3/BY_RM_gxcomp.c2c
chrVIII 12680 14779 c + - supercontig_1.12 1 2835 4934
chrVIII 14780 14780 c + i supercontig_1.12 1 4934 4934
chrVIII 14781 15406 c + - supercontig_1.12 1 4935 5560
chrVIII 15407 15407 c + i supercontig_1.12 1 5560 5560
chrVIII 15408 17132 c + - supercontig_1.12 1 5561 7285
chrVIII 17132 17132 c + d supercontig_1.12 1 7286 7286
chrVIII 17133 18415 c + - supercontig_1.12 1 7287 8569
chrVIII 18416 18416 c + i supercontig_1.12 1 8569 8569
chrVIII 18417 18566 c + - supercontig_1.12 1 8570 8719
chrVIII 18566 18566 c + d supercontig_1.12 1 8720 8720
```

which is a plain text file organized as a table of 10 columns (separated by a whitespace):

1. the name of the reference sequence,
2. when column 6 is:
    - (-) the starting position of the unagapped local alignment on the reference,
    - (i) the starting position of the insertion in the reference,
    - (d) the position of the deletion in the reference,
3. when column 6 is:
    - (-) the ending position of the unagapped local alignment on the reference,
    - (i) the ending position of the insertion in the reference,
    - (d) the position of the deletion in the reference,

4. the nature of the alignment (c for cis and t for trans, see above),
5. the strand of the alignment (+ for normal, - for reverse),
6. the nature of the event:
   - (-) ungapped alignment,
   - (i) insertion (in the reference),
   - (d) deletion (in the reference),
7. the name of the query sequence,
8. the strand of the alignment (1 for normal, -1 for reverse),
9. when column 6 is:
   - (-) the starting position of the unagapped local alignment on the query,
   - (i) the position of the deletion in the query,
   - (d) the starting position of the insertion in the query,
10. when column 6 is:
    - (-) the ending position of the unagapped local alignment on the reference,
    - (i) the position of the deletion in the query,
    - (d) the ending position of the insertion in the query,

Finally, you can use the R function `NMgcxplot` from the nucleominer R package to visualy inspect the alignements (e.g. see Figure 5).

You can also plot the SNP density along the entire genome by using another R function from the nucleominer R package, namely `NMsnpxplot`. This function needs a file that gives the size (in bp) of the chromosomes we want to plot. This file can be simply created by using an single line perl function as follows:

```
$cat Data/BY_S288c/Sequence/Genome.fasta | perl -ane \

          'if (/^>(.+)/){print $seqid," ",$size,"\n" if ($size); \
           $seqid=$1;$size=0;}else{$size+=scalar(split(//));} \
           END{print $seqid," ",$size,"\n";}' \
           > Data/BY_S288c/Sequence/Genome.size
```

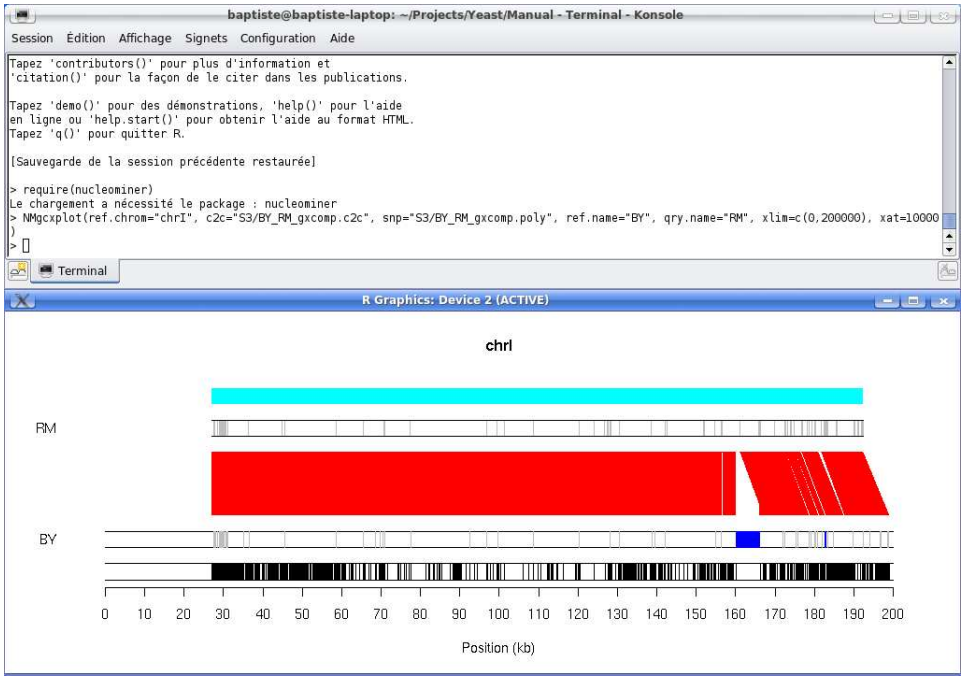

Figure 5:

We need also to slightly reformat the `.map` file as follows:

```
$grep -v '^>' S3/BY_RM_gxcomp.map | grep -v '\*' > S3/BY_RM_gxcomp.txt
```

Now, we have all the required files to run `NMsnpxplot`. Result is depicted in Figure 6.

6 Aligning nucleosome occupancy profiles

At this point we have:

- the nucleosome occupancy for each strain,
- the sequence alignment of the strain genomes,

so we can try now to look at the differences in nucleosome occupancy between the two strains. To do that, we will simply align the two nucleosome occupancy profiles based on what we already know about the sequence alignment.

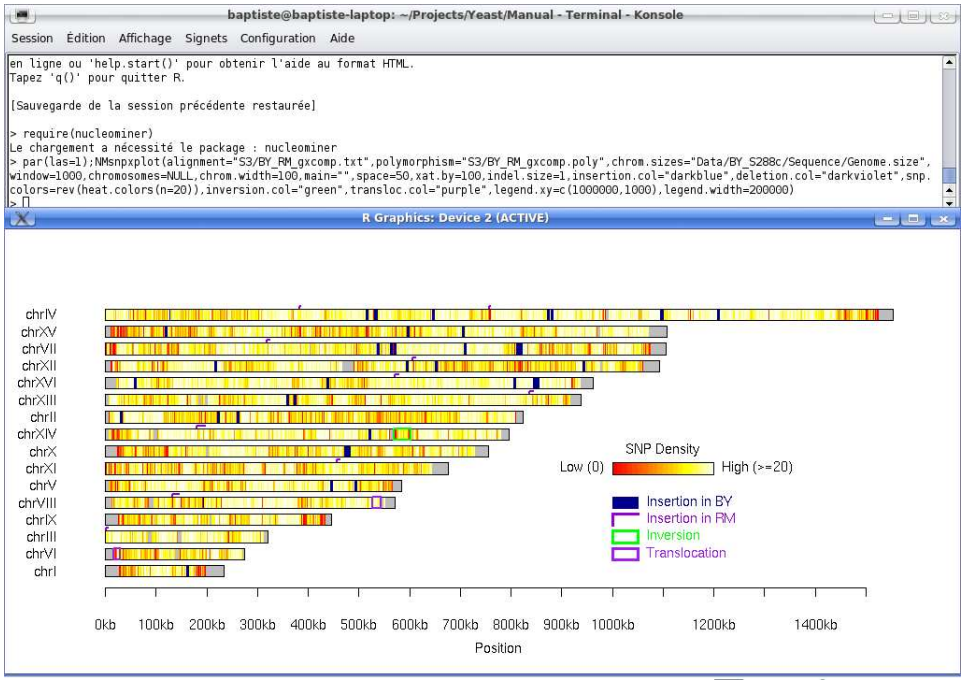

Figure 6:

By nucleosome occupancy profiles we mean here the most-likely nucleosome occupancy profiles of the two strains (as obtained from the Viterbi algorithm (`NMhmmvit`) on each chromosome).

NMalign2 The program `NMalign2` implements a simple dynamic algorithm to align the nucleosome profile of a query on the nucleosome profile of the reference according to the genome alignment of the query on the reference. The algorithm works chromosome-by-chromosome as follows:

- Assumption: two nucleosomes are said to be unambiguously aligned if the distance between their midpoints is lower than half of their average size.
- Initiation: find all unambiguously aligned nucleosomes along the chromosome.
- Recursion: between two consecutive unambiguously aligned nucleosomes

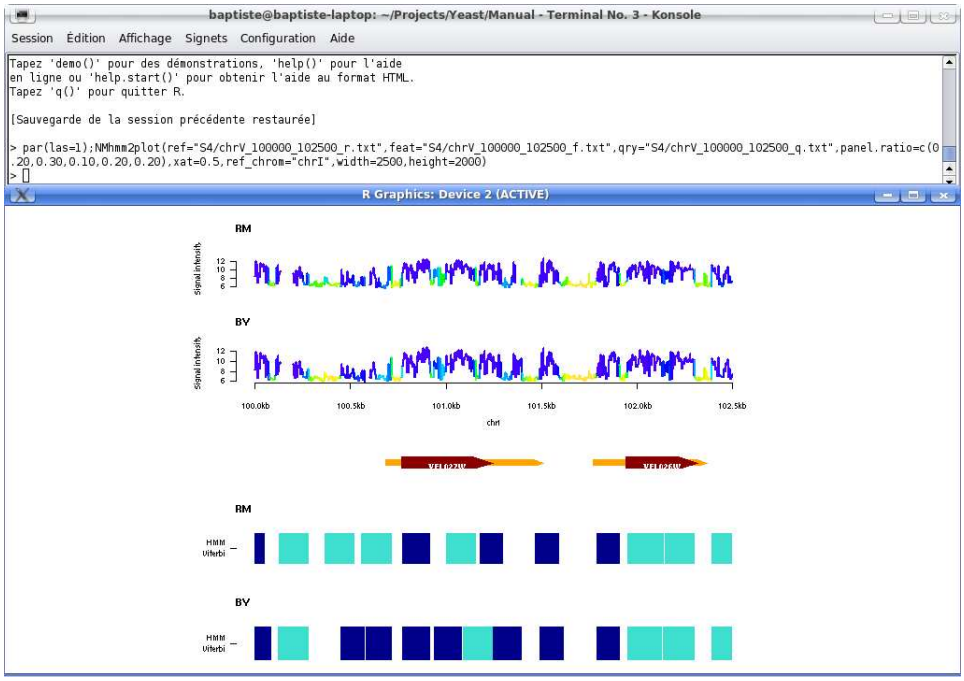

Figure 7:

- if there is no unaligned nucleosome in the interval, then go to the next unambiguously aligned nucleosome.
- else:
  - Align nucleosomes by minimizing physical distance between aligned pairs.
  - The remaining nucleosome(s) are considered as insertion(s) (if they are reference nucleosomes) or deletion(s) (if they are query nucleosomes).
  - Go to the next unambiguously aligned nucleosome.

To run `NMalign2`, we need first to write a short configuration file that will tell the program where he can find the difference sources of information require to process the nucleosome alignment. We already put such a file into the folder `S4`:

```
$NMalign2 -c S4/NMalign.conf -o S4/NMalign
```

This can take some times, so be patient. In fact, `NMalign` is doing a bit more jobs than aligning the nucleosome profiles, increasing the computation time. This should be improved in the next release of **NucleoMiner**. As you can see, the program has created a new directory `S4/NMalign` into which one file per chromosome has been created. This `.nal` file is a binary file storing the result of the alignment and some other useful information that will help to speed up subsequent analyses.

In order to generate human readable version of these `.nal` files, we have to run again `NMalign` by using now the `--print=` flag as follows:

```
$NMalign2 -i S4/NMalign/ -o S4/NMalign/ -c S4/NMalign.conf --print
```

The .nuc file Then, as you can see, we have now as many `nuc` files in directory `S4/NMalign/` than the number of studied chromosomes. These `.nuc` files are text files organized as a table which columns are seperated by a single whitespace. From left to right, the meaning of each column is:

1. the reference sequence,
2. the reference chunk index,
3. the ignore status of the reference nucleosome (y=yes/n=no),
4. the outlier status of the reference nucleosome (0=ok,1=outlier),
5. the starting position of the nucleosome on the reference,
6. the ending position of the nucleosome on the reference,
7. the status of the nucleosome on the reference (1=fuzzy, 2=well-localized),
8. the shift in bp between the midpoints of the reference and the corresponding query nucleosomes,
9. the fraction of the insertion/deletion spanning the nucleosome w.r.t the total length of the reference nucleosome,
10. the average probability that the probes covering the reference nucleosome fall into a nucleosome (either fuzzy or well-localized),
11. the query chunk index,

12. the ignore status of the query nucleosome (y=yes/n=no),
13. the outlier status of the query nucleosome (0=ok,1=outlier),
14. the starting position of the query nucleosome on the reference,
15. the ending position of the query nucleosome on the reference,
16. the status of the nucleosome on the query (1=fuzzy, 2=well-localized),
17. the shift in bp between the midpoints of the query and the corresponding reference nucleosomes,
18. the fraction of the insertion/deletion spanning the nucleosome w.r.t the total length of the query nucleosome,
19. the average probability that the probes covering the query nucleosome fall into a nucleosome (either fuzzy or well-localized).
20. if aligned, the p-value of the nucleosome (see next section),
21. if aligned, the effect size corresponding to the p-value (see next section),
22. if aligned, the status of the aligned nucleosome:
    - -1 = insertion/deletion,
    - 0 = fuzzy/fuzzy,
    - 1 = fuzzy/well-localized or well-localized/fuzzy,
    - 2 = well-localized/well-localized,
23. the probability that the region corresponds in fact to a linker / linker configuration,
24. the probability that the region corresponds in fact to a nucleosome (reference) / linker (query) configuration (insertion),
25. the probability that the region corresponds in fact to a linker (query) / nucleosome (reference) configuration (deletion),
26. the probability that the region corresponds in fact to a nucleosome / nucleosome configuration.

Now, we can use the function `NMnxcplot` of the nucleominer R package to visualize the content of these `.nuc` files. An exemple is depicted in Figure 8.

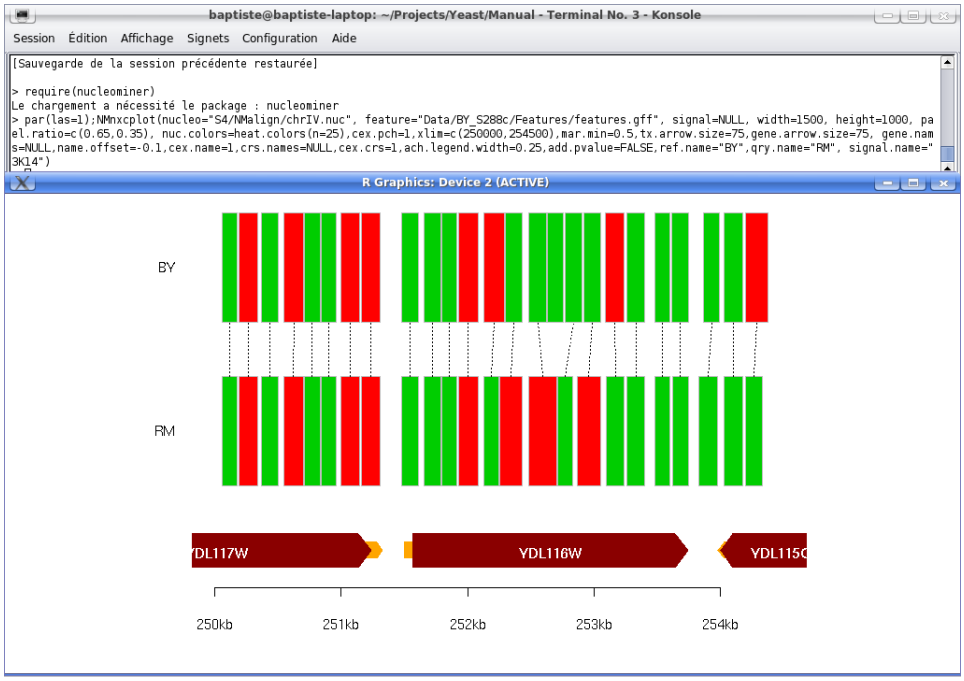

Figure 8:

7 Looking for Single Nucleosome EpiPolymorphism

Once we have aligned the nucleosomes between the two strains, we can now try to investigate the putative strain specific properties of the nucleosomes. Here, we are interested in describing the divergence of the two strains in histone H3 acetylation at Lysine 14 (H3K14).

7.1 Creating the dataset

Original dataset We need now to create a new dataset which will include for each strain the two experiments, i.e the 3 replicates used to establish the nucleosome occupancy profile (NUC) and the 6 replicates from the H3K14 ChIP-CHIP experiment (ChIP). As you can see, the file `Data/BY_S288c/Array/NucH3K14.conf` contains the 3 `.CEL` previously used to infer the nucleosome occupancy and then come the 6 replicates for the ChIP-CHIP experiment.

```
$cat Data/BY_S288c/Array/NucH3K14.conf
```

```
CBY01   CBY01-II.CEL
CBY02   CBY02-I.CEL
CBY08   CBY08-II.CEL
BY02    BY02-I.CEL
BY07    BY07-I.CEL
BY10    BY10-I.CEL
BY14    BY14.CEL
BY16    BY16.CEL
BY17    BY17.CEL
```

So, let's extract and normalize the data from these nine `.CEL` files:

```
$NMcl2tab -c Data/BY_S288c/Array/NucH3K14.conf \
          -d Data/BY_S288c/Array/ -p S1/BY_S288c/BY_S288c.prb \
          -o Data/BY_S288c/Array/NucH3K14_norm.txt
          -n qql
```

And then use `NMtdb` to export the dataset into the binary format required by **NucleoMiner** (note that this time, the `type` attribute of the `NMtdb` configuration file, namely `Data/BY_S288c/Array/NucH3K14.dbconf`, is set to `full`, i.e that no pre-processing action is asked to `NMtdb`).

```
$cd Data/BY_S288c/Array/
$NMtdb -i NucH3K14_norm.txt -o NucH3K14_norm.db -c NucH3K14.dbconf
```

Log-ratio dataset Another useful view of our data is to compute for each strain the average log ratio between the ChIP signal and the NUC signal, so that any acetylated nucleosome should exhibit positive values while non acetylated nucleosomes should exhibit negative values. To do so, we will use the option `--substract` of `NMtdb` as the intensities are already log transformed during the normalization procedure: rm

```
$NMtdb -i Data/BY_S288c/Array/NucH3K14_norm.txt \
       -o Data/BY_S288c/Array/NucH3K14_log.db \
       -c Data/BY_S288c/Array/NucH3K14_log.dbconf \
       --substract
```

7.2 Running the ANOVA

We are now ready to run `NManova2`. This program will loop on aligned nucleosomes (only) and for each pair of aligned nucleosomes it will perform an ANOVA (ANalysis Of VAriance) in order to test if there is any interaction

between the two stains and the two experiments (NUC and ChIP). A significant interaction will then mean that for this nucleosome, one strain exhibits hybridization signal in the ChIP experiment while the other one seems to have no signal for that nucleosome. By default, `NManova2` will also includes a probe effect into the model.

We will use as input the results of the nucleosome alignment as previously computed by `NMalign2`.

```
$NManova2 -a S4/NMalign -o S5/NManova -i NucH3K14_norm.db
```

By default, the ANOVA will be performed on the whole genome. To parallelize the computation or to just focus on a given chromosome you can ask `NManova2` to run on a single chromosome For example, suppose we want to run the analysis only on chromosome `chrI`:

```
$NManova2 -a S4/NMalign -o S5/NManova -i NucH3K14_norm.db -s chrI
```

Now, assuming you ran the ANOVA on the whole genome, you should see on the folder `S5/MNabova` as many files as the number of chromosomes, each file being prefixed by the corresponding chromosome name.

7.3 Computing the FDR

The program `NManofdr2` implements the procedure of Benjamini and Hochberg [1] to determine the p-value cutoffs corresponding to low, moderate and large FDR values. Let's run the program

```
$NManofdr2 -i S5/NManova -d HucH3K14_norm.db -o S5/FDR.txt
```

The output of the program is stored in the file `FDR.txt` which is a simple table like this:

```
$cat S5/FDR.txt
###
# Output of NManofdr2
###
FDR Pval #SNEP #NUC %SNEP
1.00e-04 2.89e-06 1822 62824 0.02900
1.00e-03 4.74e-05 2984 62824 0.04750
1.00e-02 8.44e-04 5303 62824 0.08441
5.00e-02 6.84e-03 8602 62824 0.13692
1.00e-01 1.75e-02 10997 62824 0.17504
```

where the first column provides the FDR value, the second one the corresponding p-value cut-off, the third one indicates the corresponding number of SNEPs (i.e. nucleosomes for which the p-value from the ANOVA is lower than the p-value cut-off), the fourth has the same value for all rows and corresponds to the total number of nucleosome having a valid p-value and the last column is just the fraction of SNEPS, i.e the ratio between the third and the fourth column.

7.4 Outputting the results

The program `NManout2` will allow us to output the results in various formats. First, let's use the flag option `--summary` to get a single file summarizing the ANOVA results as a table with the following columns:

1. the chromosome name,
2. the chromosome chunk index,
3. the starting position of the reference nucleosome (here BY),
4. the ending position of the reference nucleosome (here BY),
5. the starting position of the query nucleosome (here RM),
6. the ending position of the query nucleosome (here RM),
7. the p-value from the ANOVA for the interaction term (-1 for unscanned nucleosome),
8. the sign of the interaction (0 = reference, 1 = query),
9. the value of the interaction term,
10. the status of the reference nucleosome (1 = fuzzy, 2 = well-localized),
11. the status of the query nucleosome (1 = fuzzy, 2 = well-localized).

So let's generate this file:

```
$NManout2 -i S5/NManova -d HucH3K14_norm.db --summary 1>S5/NManova_summary.txt
```

Once it is done, you should end up with the following file:

```
$head S5/NManova_summary.txt
chrIII 1 2722 2850 2682 2818 2.78859e-14 0 1.26806 2 2
chrIII 1 2858 2978 2826 2986 9.98774e-18 0 1.4422 2 1
chrIII 1 3006 3138 3014 3146 2.53145e-26 0 1.14352 2 2
chrIII 1 3154 3274 3170 3326 5.93676e-07 0 0.879466 2 1
chrIII 1 3294 3430 3334 3454 8.5769e-18 0 1.42377 2 2
chrIII 1 3438 3558 3462 3626 1.4782e-24 0 1.57474 2 1
chrIII 1 3582 3738 3634 3798 6.06824e-25 0 1.48753 1 1
chrIII 1 3770 3922 3834 3962 2.89049e-07 0 0.840172 1 2
chrIII 1 4002 4138 3982 4134 4.31898e-06 0 1.00544 2 1
chrIII 1 4314 4466 4374 4498 0.032396 0 0.517827 1 2
```

where the meaning of the columns has been discussed above.

To get a more detailed picture of the SNEPs, we can then ask to `NManout2` to output only the nucleosomes which p-values are lower a pre-defined cut-off together with additional information such that neighboring nucleosomes and known functional annotations. So, here we will look at the nucleosomes which p-values are smaller than the cut-off corresponding to a FDR of 1%. Then, we will ask `NManout2` to output for each SNEP the functional annotations (as provided in the file `Data/BY_S288c/Features/features.gff`) falling symmetrically within a 5kb window around the SNEP. Finally, we will ask also to get for each SNEP the 5 nucleosomes before and the 5 nucleosomes after, in order to investigate if the SNEP is isolated or not. This multiple query can be done as follows:

```
$NManout2 -i S5/NManova -d HucH3K14_norm.db -o S5/NManout \
         -f Data/BY_S288c/Features/features.gff \
         -w 2500 -p 8.44e-4 -b 5
```

The command generates 3 files whith the stem name `NManout` in the folder `S5`:

- `NManout_snep_table.txt` is the file providing the result of the multiple query,
- `NManout_snep.gff` is a file in GFF format (version 3) listing all the SNEPs called at the chosen p-value cut-off. This file can then be used as input to your favorite genome browser.
- `NManout_snep_tiling.txt` is a file that will be useful to do some extra analysis on the SNEP distribution w.r.t to functional annotation using either `NMt2feat` or `NMt2featcis`.

References

[1] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57:289–300, 1995.

[2] L. David, W. Huber, M. Granovskaia, J. Toedling, C.J. Palm, L. Bofkin, T. Jones, R.W. Davis, and L.M. Steinmetz. A high-resolution map of transcription in the yeast genome. *Proc. Natl. Acad. Sci. U.S.A.*, 103:5320–5325, Apr 2006.

[3] H. Ji and W.H. Wong. TileMap: create chromosomal map of tiling array hybridizations. *Bioinformatics*, 21:3629–3636, Sep 2005.

[4] W. Lee, D. Tillo, N. Bray, R.H. Morse, R.W. Davis, T.R. Hughes, and C. Nislow. A high-resolution atlas of nucleosome occupancy in yeast. *Nat. Genet.*, 39:1235–1244, Oct 2007.

[5] Muniyandi Nagarajan, Jean-Baptiste Veyrieras, Maud de Dieuleveult, Hlne Bottin, Steffen Fehrmann, Anne-Laure Abraham, Sverine Croze, Lars M. Steinmetz, Xavier Gidrol, and Gal Yvert. Natural single-nucleosome epi-polymorphisms in yeast. *PLoS Genet*, 6(4):e1000913, 04 2010.

[6] G.C. Yuan, Y.J. Liu, M.F. Dion, M.D. Slack, L.F. Wu, S.J. Altschuler, and O.J. Rando. Genome-scale identification of nucleosome positions in S. cerevisiae. *Science*, 309:626–630, Jul 2005.