

Chromatinsight

This tutorial shows the steps to obtain the *degree of discernibility* of epigenomic features by genomic regions from ChIP-seq data in two sets of samples. The degree of discernibility is the *degree of dimorphism* if the datasets are separated in approximately two halves according to the sex of the individual, and it shows how easy or difficult is for a machine learning algorithm to predict the dataset from which the data from a genomic region is coming from. We will calculate also the FDR of discernibility.

Part 1: Chromatinsight (Python)

Before we start, we need to have:

Python 2.7

pandas installed, see https://pandas.pydata.org/pandas-docs/stable/getting_started/install.html

sklearn installed, see <https://scikit-learn.org/stable/install.html>






















A quick way to install both libraries is by using these pip commands:

```
pip install pandas
```

```
pip install sklearn
```

In this tutorial, we are going to use the output of ChromHMM, using only chrX for simplicity, but you can use the whole genome (you will have a separated file for each chromosome and sample). Using ChromHMM, we have generated binary files (ten for each dataset) using windows of 200 bases. If you need help using ChromHMM to preprocess the ChIP-seq data, you can consult the ChromHMM manual: http://compbio.mit.edu/ChromHMM/ChromHMM_manual.pdf

The files we are going to use here have been already preprocessed, so they can be used directly in Chromatinsight:

Name	Date modified	Type	Size
 monocyte_S00_mal_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S01_mal_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S02_mal_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S03_mal_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S04_mal_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S05_mal_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S06_mal_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S07_mal_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S08_mal_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S09_mal_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S10_fem_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S11_fem_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S12_fem_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S13_fem_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S14_fem_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S15_fem_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S16_fem_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S17_fem_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S18_fem_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 monocyte_S19_fem_chrX_binary.txt	22/01/2021 12:56	TXT File	3,033 KB
 regionFile_chrX_selection.bed	22/01/2021 13:01	BED File	1 KB

You can find these files in the folder `/utest/input`, or here:
<https://github.com/ricolab/Chromatinsight/tree/main/utest/input>

The files are text files having a row for each 200-base window in the chromosome, and one column for each histone modification. A '0' means there is no histone modification, and a '1' means there are enough reads aligned in that window to be considered a peak.

[illegible]

The input of Chromatinsight also needs a region file, which is a bed file, ie, a text file where each row is a region, specifying the chromosome, start and end coordinates for each region. In this unit test we have included few regions to make it faster, but you can include as many as you need. The regions can be TADs (topologically associating domains), genes, Hi-C fragments, or anything that fulfils your needs.

chr	start	end
X	71527500	71551875
X	71551875	71562500
X	71562500	72022500
X	72022500	76640000
X	76640000	76675625
X	76675625	76692500
X	130135625	130148750
X	130148750	130173125
X	130173125	130890000
X	130890000	130917500
X	130917500	131357500
X	131357500	131365000
X	131365000	131367500

The code you need to run this unit test is in

https://github.com/ricolab/Chromatinsight/blob/main/chromatinsight_utest.py

You need to run Python 2.7 on the folder where you have downloaded Chromatinsight. Once there, you need to import the libraries, including the one that contains Chromatinsight:

```
1 import chromatinsight as ci
2 import os
```

The next lines need to be modified to match the folders you are actually using:

```
4 myRegionFile = "./utest/input/regionFile_chrX_selection.bed"
5 myInputFolder = "./utest/input"
6 myOutputFolder = "./utest/output"
7 if not os.path.isdir(myOutputFolder): os.mkdir(myOutputFolder)
```

Now we are ready to run the algorithm. We will run it twice, first for the observed data:

```
9 # RF_seed = 0 and label_seed = 0 are set to generate a deterministic random behaviour
10 resultObserved = ci.testPrediction(prefix = "mono",
11                                   histmod = "ac",
12                                   regionFile = myRegionFile,
13                                   chrom = "chrX",
14                                   inputFolder = myInputFolder,
15                                   outputFolder = myOutputFolder,
16                                   output = "mono_ac_chrX_real.txt",
17                                   totRandomStates = 11,
18                                   randomize = False,
19                                   verbose = True,
20                                   RF_seed = 0)
```

and then another time where the labels of the datasets will be scrambled, to display the random behaviour:

```

21 resultRnd = ci.testPrediction(prefix = "mono",
22                               histmod = "ac",
23                               regionFile = myRegionFile,
24                               chrom = "chrX",
25                               inputFolder = myInputFolder,
26                               outputFolder = myOutputFolder,
27                               output = "mono_ac_chrX_rnd.txt",
28                               totRandomStates = 11,
29                               randomize = True,
30                               verbose = True,
31                               label_seed = 0,
32                               RF_seed = 0)

```

The output should be two files, one for the observed data, and another one for the random data. You can see how your output matches the expected output by checking the data on https://github.com/ricolab/Chromatinsight/tree/main/utest/output_expected

Each output file is a text file including a column with a region identifier, and then the ratio of times the algorithm has managed to predict correctly the dataset of each sample. To improve the result, we have made this several times (in this case 11 times, as the totRandomStates variable shows), but you can use more.

chrom_init-end_region	1.0	1.0	1.0	1.0	1.0	1.0
chrX_0-71527500_Starting	1.0	1.0	1.0	1.0	1.0	1.0
chrX_71527500-71551875_TAD	0.3333333333333333	0.5	0.5	0.3333333333333333	0.5	0.3333333333333333
chrX_71551875-71562500_TAD	0.5	0.6666666666666666	0.5	0.5	0.5	0.5
chrX_71562500-72022500_TAD	0.5	0.5	0.5	0.5	0.5	0.5
chrX_72022500-76640000_TAD	0.5	0.6666666666666666	0.5	0.5	0.5	0.8333333333333333
chrX_76640000-76675625_TAD	0.3333333333333333	0.5	0.5	0.5	0.5	0.5
chrX_76675625-76692500_TAD	0.5	0.5	0.5	0.5	0.5	0.5
chrX_76692500-130135625_interTAD	0.6666666666666666	0.6666666666666666	0.6666666666666666	0.6666666666666666	0.6666666666666666	0.6666666666666666
chrX_130135625-130148750_TAD	0.5	0.5	0.3333333333333333	0.5	0.5	0.5
chrX_130148750-130173125_TAD	0.5	0.5	0.5	0.5	0.5	0.5
chrX_130173125-130890000_TAD	1.0	1.0	0.8333333333333333	1.0	1.0	1.0
chrX_130890000-130917500_TAD	0.5	0.5	0.8333333333333333	1.0	1.0	1.0
chrX_130917500-131357500_TAD	0.5	0.5	0.6666666666666666	0.6666666666666666	0.6666666666666666	0.6666666666666666

Part 2: chromatinsight.tools (analysis in R)

Now, we are going to use the *chromatinsight.tools* library in R to analyse these data. You can download the library from the GitHub project page, at <https://github.com/ricolab/chromatinsight.tools>, but perhaps it's faster to install it using devtools:

```
devtools::install_github("ricolab/chromatinsight.tools")
```

If you don't have devtools installed, then you can do so using these command:

```
install.packages("devtools")
```

If you are using Linux, you might need first:

```
sudo apt-get install libgit2-dev
```

Once the library is installed and loaded, you can check the help by using directly the R command for it. For example, `?getRegionData` shows:

A screenshot of an R Help window titled 'R Help on 'getRegionData''. The window has a menu bar with 'Archivo' and 'Editar'. Below the menu bar, it shows 'getRegionData' from 'package:chromatinsight.tools' with a link to 'R Documentation'. The main text describes the function: 'Loads the output text files of Chromatinsight (currently Python only), Using as output the median of the number of trials performed (in the totRandomStates variable)'. It includes a 'Description:' section with the same text and a 'Usage:' section with the function signature:

```
getRegionData(  
  myPath = "",  
  histmod = "ac",  
  prefix = "prefix",  
  suffix = "",  
  totRandomStates = 11  
)
```

We recommend installing also the `biomaRt` library in R to download the genes and gene types that are in each region, but it is not necessary if you do not need to have this information.

Now we are ready to start doing something in R with the output of Chromatinsight. First, we need to load the libraries.

```
library(chromatinsight.tools)  
library(biomaRt)
```

Then, we set the input and output folders according to where we have them in our machine:

```
myInputFolder = "./utest/input"  
myOutputFolder = "./utest/output"
```

To load the different genes and gene types in our region file, we can run the following code:

```
gene_types = as.data.frame(read.table(header = TRUE, text = "  
gene_type  
3prime_overlapping_ncrna  
antisense  
lincRNA
```

```

miRNA
misc_RNA
processed_transcript
protein_coding
pseudogene
rRNA
sense_intronic
sense_overlapping
snoRNA
snRNA
") )

ensembl75 = useMart(host='feb2014.archive.ensembl.org',
  biomart='ENSEMBL_MART_ENSEMBL',
  dataset='hsapiens_gene_ensembl')

allBioMartChr = getBM(mart = ensembl75, attributes =
  c("ensembl_gene_id", "wikigene_name", "chromosome_name",
  "start_position", "end_position", "gene_biotype"))

```

In our case, we have selected the host to be `feb2014.archive.ensembl.org`, because this is the version of the database that has been used to align the ChIP-seq fastq files originally, but if you are using different data you might want to change this.

Using the command `View(allBioMartChr)` you can see all the genes that have been downloaded:

	ensembl_gene_id	wikigene_name	chromosome_name	start_position	end_position	gene_biotype
1	ENSG00000261657	SLC25A26	HG991_PATCH	66119285	66465398	protein_coding
2	ENSG00000223116		13	23551994	23552136	miRNA
3	ENSG00000233440		13	23708313	23708703	pseudogene
4	ENSG00000207157		13	23726725	23726825	misc_RNA
5	ENSG00000229483		13	23743974	23744736	lincRNA
6	ENSG00000252952	RNU6-58P	13	23791571	23791673	snRNA
7	ENSG00000235205		13	23817659	23821323	pseudogene
8	ENSG00000232849		13	93708910	93710179	lincRNA
9	ENSG00000236803		13	23841645	23843289	pseudogene
10	ENSG00000232163		13	23947562	23948200	pseudogene
11	ENSG00000252365		13	94021100	94021213	snoRNA
12	ENSG00000229558	SACS-AS1	13	23993110	24002818	lincRNA
13	ENSG00000235784		13	94047230	94048108	pseudogene
14	ENSG00000224394		13	94470677	94488397	antisense
15	ENSG00000212559		13	94677724	94677852	rRNA

We load now both files that Chromatinsight has generated in the previous step, and store them in a table using the function `getRegionData`:

```

dimorphismMono = getRegionData(myInputFolder, prefix = "mono", suffix
= "real", totRandomStates = 11)
dimorphismMonoRnd = getRegionData(myInputFolder, prefix = "mono",
suffix = "rnd", totRandomStates = 11)

```

Using the command `View(dimorphismMono)` you can see the data:

	chrom	init	end	region	comments	tot_genes	disparity
1	chrX	0	71527500	Starting	-	0	1.0000000
2	chrX	71527500	71551875	TAD	-	0	0.5000000
3	chrX	71551875	71562500	TAD	-	0	0.6666667
4	chrX	71562500	72022500	TAD	-	0	0.5000000
5	chrX	72022500	76640000	TAD	-	0	0.6666667
6	chrX	76640000	76675625	TAD	-	0	0.5000000
7	chrX	76675625	76692500	TAD	-	0	0.5000000
8	chrX	76692500	130135625	interTAD	-	0	0.5000000
9	chrX	130135625	130148750	TAD	-	0	0.5000000
10	chrX	130148750	130173125	TAD	-	0	0.5000000
11	chrX	130173125	130890000	TAD	-	0	1.0000000
12	chrX	130890000	130917500	TAD	-	0	0.8333333
13	chrX	130917500	131357500	TAD	-	0	0.6666667
14	chrX	131357500	131365000	TAD	-	0	0.5000000
15	chrX	131365000	131367500	TAD	-	0	0.5000000
16	chrX	131367500	155270400	Ending	-	0	0.6666667

The disparity shows the degree of discernibility (or dimorphism, in this case). When it is 0.5 or lower, it is certain that the machine wasn't guessing better than randomly the origin of the genomic region by looking at the epigenomics data. When it is 1.0, then we know the data could be distinguished every time, so there is some clear, unmissable pattern. But when this parameter lies between 0.5 and 0.1, we need to set an objective threshold to define when the difference is significant. The library can calculate an FDR to fulfil this need, observing the data in the random distribution:

	chrom	init	end	region	comments	tot_genes	disparity
1	chrX	0	71527500	Starting	-	0	0.5833333
2	chrX	71527500	71551875	TAD	-	0	0.3333333
3	chrX	71551875	71562500	TAD	-	0	0.3333333
4	chrX	71562500	72022500	TAD	-	0	0.5000000
5	chrX	72022500	76640000	TAD	-	0	0.5000000
6	chrX	76640000	76675625	TAD	-	0	0.3333333
7	chrX	76675625	76692500	TAD	-	0	0.5000000
8	chrX	76692500	130135625	interTAD	-	0	0.5833333
9	chrX	130135625	130148750	TAD	-	0	0.5000000
10	chrX	130148750	130173125	TAD	-	0	0.5000000
11	chrX	130173125	130890000	TAD	-	0	0.5000000
12	chrX	130890000	130917500	TAD	-	0	0.4166667
13	chrX	130917500	131357500	TAD	-	0	0.5000000
14	chrX	131357500	131365000	TAD	-	0	0.5000000
15	chrX	131365000	131367500	TAD	-	0	0.5000000
16	chrX	131367500	155270400	Ending	-	0	0.5000000

As you can observe, when randomising the dataset labels, the algorithm was not guessing very well the origin of the genomic regions. To calculate the FDR, simply use the function `addFDR` with both tables:


```
dimorphismMonoFdr = addFDR(dimorphismMono, dimorphismMonoRnd)
```

Using the View command we can check how the table looks like:

	chrom	init	end	region	comments	tot_genes	disparity	FDR
1	chrX	0	71527500	Starting	-	0	1.0000000	0.00
2	chrX	71527500	71551875	TAD	-	0	0.5000000	0.75
3	chrX	71551875	71562500	TAD	-	0	0.6666667	0.00
4	chrX	71562500	72022500	TAD	-	0	0.5000000	0.75
5	chrX	72022500	76640000	TAD	-	0	0.6666667	0.00
6	chrX	76640000	76675625	TAD	-	0	0.5000000	0.75
7	chrX	76675625	76692500	TAD	-	0	0.5000000	0.75
8	chrX	76692500	130135625	interTAD	-	0	0.5000000	0.75
9	chrX	130135625	130148750	TAD	-	0	0.5000000	0.75
10	chrX	130148750	130173125	TAD	-	0	0.5000000	0.75
11	chrX	130173125	130890000	TAD	-	0	1.0000000	0.00
12	chrX	130890000	130917500	TAD	-	0	0.8333333	0.00
13	chrX	130917500	131357500	TAD	-	0	0.6666667	0.00
14	chrX	131357500	131365000	TAD	-	0	0.5000000	0.75
15	chrX	131365000	131367500	TAD	-	0	0.5000000	0.75
16	chrX	131367500	155270400	Ending	-	0	0.6666667	0.00

A new column has been added. Since we were not using many genomic regions, we only have two possible FDR values (either 0% or 75%), but in a full experiment we would have a range between 0 and 100%. Anyway, a common threshold to consider significant a value is using the FDR at 5% (which is fulfilled here by the seven regions where the FDR is 0%).

Now, we have a table with all the genomic regions, the degree of discernibility and the FDR. We can add some columns to the table including the Ensembl identifiers of the genes in each region, and the number of each gene type:

```
dimorphismMonoMore = fillTableWithGeneTypes(dimorphismMonoFdr,
allBioMartChr, gene_types)
```

This larger table should make easier the interpretation of the data:

	chrom	init	end	region	comments	tot_genes	disparity	FDR	3prime_overlapping_ncrna	antisense	lincRNA	m
1	chrX	0	71527500	Starting	ENSG00000102309, ENSG00000186871, ENSG00000198034>	1089	1.0000000	0.00	0	46	72	7
2	chrX	71527500	71551875	TAD	ENSG00000147099	1	0.5000000	0.75	0	0	0	
3	chrX	71551875	71562500	TAD	ENSG00000147099	1	0.6666667	0.00	0	0	0	
4	chrX	71562500	72022500	TAD	ENSG00000147099, ENSG00000067177, ENSG00000184911>	11	0.5000000	0.75	0	1	1	
5	chrX	72022500	76640000	TAD	ENSG00000184911, ENSG00000159123, ENSG00000184388>	85	0.6666667	0.00	0	2	8	
6	chrX	76640000	76675625	TAD		0	0.5000000	0.75	0	0	0	
7	chrX	76675625	76692500	TAD		0	0.5000000	0.75	0	0	0	
8	chrX	76692500	130135625	interTAD	ENSG00000186416, ENSG00000221494, ENSG00000233484>	677	0.5000000	0.75	1	24	20	4
9	chrX	130135625	130148750	TAD	ENSG00000228659	1	0.5000000	0.75	0	1	0	
10	chrX	130148750	130173125	TAD	ENSG00000228659	1	0.5000000	0.75	0	1	0	
11	chrX	130173125	130890000	TAD	ENSG00000228659, ENSG00000241198, ENSG00000237650>	18	1.0000000	0.00	0	1	1	
12	chrX	130890000	130917500	TAD	ENSG00000213468	1	0.8333333	0.00	0	0	1	
13	chrX	130917500	131357500	TAD	ENSG00000213468, ENSG00000241418, ENSG00000228089>	11	0.6666667	0.00	0	1	1	
14	chrX	131357500	131365000	TAD	ENSG00000232160	1	0.5000000	0.75	0	1	0	
15	chrX	131365000	131367500	TAD	ENSG00000232160	1	0.5000000	0.75	0	1	0	
16	chrX	131367500	155270400	Ending	ENSG00000226388, ENSG00000231686, ENSG00000235699>	505	0.6666667	0.00	0	31	30	7

To save the output you simply write the table using R:

```
setwd(myOutputFolder)
```



```
write.table(dimorphismMonoMore, "dimorphismMono_output.txt", sep =
"\t", row.names = FALSE, quote = FALSE)
```

The library `chromatinsight.tools` comes with some more built-in functions that allow the visualisation of the epigenomic data (requires to install the library `ggplot2`):

pileup

You give it the filename prefix to find a set of files of ChromHMM binaries and it adds up the "ones" and "zeroes" that are at the same position for the set of files, dividing by the number of files. So it gives the ratio of samples having a "one" at a specific position, ie, the ratio of samples being H3K27ac or H3K4me1 at a ChromHMM bin.

Example:

```
pileup(prefix = "mono*S*mal", direc = "/data/", chrom = "chr10")
```

Will merge data from files (within /data folder)

```
monocytes_S2901_mal_DATA_binary.txt
monocytes_S3454_mal_DATA_binary.txt
monocytes_S5715_mal_DATA_binary.txt
etc
```

drawpile

You give it the frequency of a histone mark at every ChromHMM bin, ie, the output of two `pileup` functions (see `pileup`), and draws the graph comparing them.

For H3K27ac the first group is red, the second is blue.

For H3K4me1 the first group is pink, the second is forest green.

To use UCSC Genome Browser coordinate format (such as `chrX:10,000-25,000`) see `drawpilegb`.

drawpilegb Calls `drawpile` with UCSC Genome Browser coordinate format, such as `chrX:10,000-25,000` (See `drawpile` and `densitpile` for more details).

densitpile

You give it the output of two `pileup` function and it shows the graph comparing them.

For H3K27ac the first group is red, the second is blue, when they overlap it's purple.

For H3K4me1 the first group is pink, the second is forest green, when they overlap it's grey.

To use UCSC Genome Browser coordinate format (such as `chrX:10,000-25,000`) see `drawpilegb`.

compareDistributions

Displays the violin plots of the distributions in two analyses retrieved using `getRegionData`. Especially useful to compare random vs observed data.