5. Chipseq analysis

We chose NFKB1 as a possible regulator of the upregulated genes in rheumatoid arthritis. Now the next question is: does NFKB1 bind to these upregulated genes? This can be investigated by use of Chipseq. In ChIP-Atlas we searched for a chipseq data set of NFKB1 and found one performed in an Hodgkin Lymphoma Cell Line. Ideally, we would analyze chipseq data of NFKB1 in a rheumatoid arthritis patient, however we did not found one.

The chipseq data can be found on the following website:

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM1556335 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM1556335)

5.1 Convert SRA to fastq

First make a directory to store the files in:

In [6]:

```
mkdir -p /mnt/nfs/r0368280/jupyternotebooks/NFKB1/
cd /mnt/nfs/r0368280/jupyternotebooks/NFKB1/
```

Second, we make a new directory where we simulatenously are going to download the Chipseq data and convert SRA to fastq.

The original files can be found in /mnt/nfs/r0368280/jupyternotebooks/NCBI/

under the account of r0368280

In [3]:

```
mkdir /mnt/nfs/r0368280/jupyternotebooks/NCBI
cd /mnt/nfs/r0368280/jupyternotebooks/NCBI/

vdb-config -s /repository/user/main/public/root=/mnt/nfs/r0368280/jupyternotebook
```

```
refseq  sra  SRR1686077_1.fastq
```

In [ ]:

```
fastq-dump --split-files SRR1686077
```

5.2 Alignment

Now we have the chipseq data in fastq format. Subsequently we are going to align this data to the entire genome by using bowtie2. This can take a long time (several hours).

In [14]:

```
bowtie2-align -x /mnt/nfs/data/resources/bowtie2/hg19 /mnt/storage/r0368280/jupyt
```

```
39932937 reads; of these:
  39932937 (100.00%) were unpaired; of these:
    9435278 (23.63%) aligned 0 times
    21183429 (53.05%) aligned exactly 1 time
    9314230 (23.32%) aligned >1 times
76.37% overall alignment rate
```

We look at the aligment with the head command to see if everything went fine.

In [15]:

```
head -500 ChIP_NFKB1.sam | tail -5
```

```
SRR1686077.410  16      chr1    569908  31      36M     *       0
0       ACTAACCATATACCAATGATGGCGCGATGTAACACG
@DFGEGIGGIGFIIIIIIIIIIIIIIIIIIIII    AS:i:0  XS:i:-6 XN:i:0
XM:i:0  XO:i:0  XG:i:0  NM:i:0  MD:Z:36 YT:Z:UU
SRR1686077.411  16      chr1    569908  31      36M     *       0
0       ACTAACCATATACCAATGATGGCGCGATGTAACACG
###BFFDGGGGG@GGGGEGEEGEEGBHFHFHGGHHH    AS:i:0  XS:i:-5 XN:i:0
XM:i:0  XO:i:0  XG:i:0  NM:i:0  MD:Z:36 YT:Z:UU
SRR1686077.412  16      chr1    569908  31      36M     *       0
0       ACTAACCATATACCAATGATGGCGCGATGTAACACG
:D<FEFBB>@>EGGGGGGGGGDGGD?EBFEF=@==B    AS:i:0  XS:i:-5 XN:i:0
XM:i:0  XO:i:0  XG:i:0  NM:i:0  MD:Z:36 YT:Z:UU
SRR1686077.413  16      chr1    569908  31      36M     *       0
0       ACTAACCATATACCAATGATGGCGCGATGTAACACG    BECDCFDDBDD?EBDDGH?H
HHHHGHHHHHHGGGGF        AS:i:0  XS:i:-5 XN:i:0  XM:i:0  XO:i:0
XG:i:0  NM:i:0  MD:Z:36 YT:Z:UU
SRR1686077.414  16      chr1    569908  31      36M     *       0
0       ACTAACCATATACCAATGATGGCGCGATGTAACACG
CBCFAFGGBGGDGGGDHBHHHHFHFHHHHHHHHHHH    AS:i:0  XS:i:-5 XN:i:0
XM:i:0  XO:i:0  XG:i:0  NM:i:0  MD:Z:36 YT:Z:UU
```

5.3 Convert SAM to BAM file

The SAM file is unsorted and uncompressed. To be able to quickly access the data we are going to convert it to a BAM file, which is a compressed (much smaller) version of a SAM file, and sort the file. The last step is to create an index for quick access. We use samtools to do this.

In [16]:

```
samtools view -S -b ChIP_NFKB1.sam > ChIP_NFKB1.bam
```

In [17]:

```
samtools sort -O bam -o ChIP_NFKB1.sorted.bam ChIP_NFKB1.bam
```

```
[bam_sort_core] merging from 10 files...
```

In [18]:

```
samtools index ChIP_NFKB1.sorted.bam
```

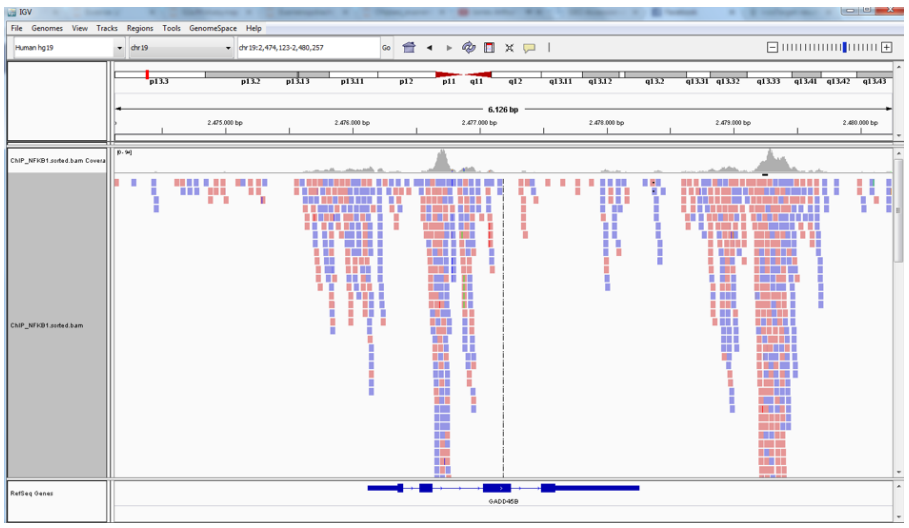
In [8]:

```
ls
```

```
ChIP_NFKB1.bam  ChIP_NFKB1.sorted.bam      refseq  SRR1686077_1.fast
q
ChIP_NFKB1.sam  ChIP_NFKB1.sorted.bam.bai  sra
```

We now have two sorted bam files and copy these two to our computer and load the BAM file into IGV. If we visualize the BAM file in IGV you see the coverage (how many reads map to a certain position).

In the printsceen below you see a lot of coverage for a part of GADD45B. Red is the antisense strand, blue is the sense strand.



5.4 Call peaks

Now we are going to call peaks. With peak calling we try to identify the regions in the genome that are enriched with aligned reads. No control was added because the authors didn't provide one. If you run without control you have less peaks. Why? The program will hesitate more to call something a peak, so you will get less peaks. Also the peaks were called in the entire genome and a q-value of 0.005 was used.

In [1]:

```
mkdir macs2_NFKB1
macs2 callpeak -t  ChIP_NFKB1.sorted.bam --outdir macs2_NFKB1 -n NFKB1 -g hs -q 0
```

```
INFO  @ Fri, 23 Dec 2016 21:08:38:
# Command line: callpeak -t ChIP_NFKB1.sorted.bam --outdir macs2_NFK
B1 -n NFKB1 -g hs -q 0.005
# ARGUMENTS LIST:
# name = NFKB1
# format = AUTO
# ChIP-seq file = ['ChIP_NFKB1.sorted.bam']
# control file = None
# effective genome size = 2.70e+09
# band width = 300
# model fold = [5, 50]
# qvalue cutoff = 5.00e-03
# Larger dataset will be scaled towards smaller dataset.
# Range for calculating regional lambda is: 10000 bps
# Broad region calling is off

INFO  @ Fri, 23 Dec 2016 21:08:38: #1 read tag files...
INFO  @ Fri, 23 Dec 2016 21:08:38: #1 read treatment tags...
INFO  @ Fri, 23 Dec 2016 21:08:38: Detected format is: BAM
INFO  @ Fri, 23 Dec 2016 21:08:38: * Input file is gzipped.
INFO  @ Fri, 23 Dec 2016 21:08:42:  1000000
INFO  @ Fri, 23 Dec 2016 21:08:46:  2000000
INFO  @ Fri, 23 Dec 2016 21:08:49:  3000000
INFO  @ Fri, 23 Dec 2016 21:08:53:  4000000
```

```
Traceback (most recent call last):
  File "/usr/local/bin/macs2", line 614, in <module>
    main()
  File "/usr/local/bin/macs2", line 56, in main
    run( args )
  File "/usr/local/lib/python2.7/dist-packages/MACS2/callpeak_cmd.py
", line 70, in run
    else:       (treat, control) = load_tag_files_options  (options)
  File "/usr/local/lib/python2.7/dist-packages/MACS2/callpeak_cmd.py
", line 396, in load_tag_files_options
    treat = tp.build_fwtrack()
  File "MACS2/IO/Parser.pyx", line 805, in MACS2.IO.Parser.BAMParser
.build_fwtrack (MACS2/IO/Parser.c:11368)
  File "MACS2/IO/Parser.pyx", line 810, in MACS2.IO.Parser.BAMParser
.build_fwtrack (MACS2/IO/Parser.c:11235)
  File "MACS2/IO/Parser.pyx", line 874, in MACS2.IO.Parser.BAMParser
.__build_fwtrack_wo_pysam (MACS2/IO/Parser.c:12256)
ValueError: read of closed file
```

It looks like the above command didn't work but this is due to the fact that we redid the command while it actually worked the first time.

In [20]:

```
ls
#you see what is in the macs2_NFKB1 folder
```

```
ChIP_NFKB1.bam  ChIP_NFKB1.sorted.bam      macs2_NFKB1  sra
ChIP_NFKB1.sam  ChIP_NFKB1.sorted.bam.bai  refseq
SRR1686077_1.fastq
```

We are now going to take a look at the file with the peaks to see if everything went fine.

In [21]:

```
head macs2_NFKB1/NFKB1_peaks.narrowPeak
```

```
chr1    10021   10142   NFKB1_peak_1    95      .       6.65150 12.7
4406    9.52473 50
chr1    14481   14580   NFKB1_peak_2    105     .       6.33487 13.8
4319    10.58679        54
chr1    29296   29384   NFKB1_peak_3    63      .       5.78035 9.37
442     6.30489 35
chr1    96428   96567   NFKB1_peak_4    428     .       18.17081
46.75889        42.80931        76
chr1    96631   96767   NFKB1_peak_5    180     .       10.38332
21.51325        18.03033        62
chr1    150957  151062  NFKB1_peak_6    66      .       6.42055 9.75
961     6.67253 66
chr1    242754  243105  NFKB1_peak_7    492     .       18.76245
53.31419        49.26944        84
chr1    378088  378273  NFKB1_peak_8    1665    .       42.82283
172.38879       166.59174       107
chr1    412155  412449  NFKB1_peak_9    345     .       15.42760
38.34253        34.52510        230
chr1    564664  564800  NFKB1_peak_10   120     .       2.81565 15.3
7391    12.06224        71
```

In the following two commands we sorted on the 5 highest peaks and look at the total number of peaks that were called.

In [23]:

```
cat macs2_NFKB1/NFKB1_peaks.narrowPeak | sort -k9 -g -r | head -5
```

```
chr3    8724705 8724898 NFKB1_peak_19016        2141    .
50.75469        223.63388       214.14240       95
chr20   42033564        42033775        NFKB1_peak_17722        2095
.       44.56903        218.46510       209.57567       111
chr15   81068787        81069028        NFKB1_peak_9642 2095    .
44.56903        218.46510       209.57567       136
chr11   109915401       109915587       NFKB1_peak_5586 2095    .
50.62173        218.42328       209.57567       87
chr4    3895454 3895650 NFKB1_peak_20754        2080    .
48.73769        216.01567       208.04269       100
sort: write failed: standard output: Broken pipe
sort: write error
```

In [22]:

```
wc -l macs2_NFKB1/NFKB1_peaks.narrowPeak
```

```
31282 macs2_NFKB1/NFKB1_peaks.narrowPeak
```

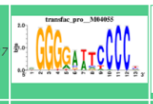
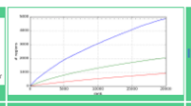
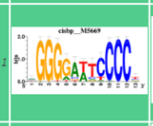
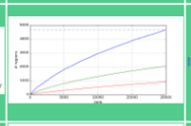
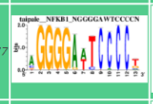
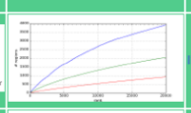
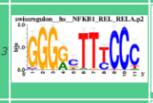
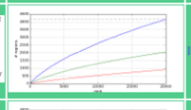
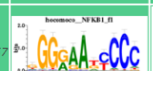
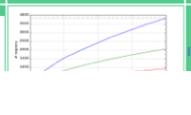
There are 31282 peaks called in the entire genome

5.5 i-cisTarget

We want to check if NFKB1 is really binding to the regions in the genome where the peaks are situated, as a quality control to verify if the chipseq worked. This can be done with i-cisTarget, a tool that analyzes the peaks for enriched motifs. We could also use Rsat (Regulatory Sequence Analysis Tools). Here we prefer i-cisTarget because it tests for known motifs in the database, while rsat does the novo motif discovery via statistical analysis which can generate a lot of false positives.

We see that the first motifs of the list are NFKB2 and the third one is NFKB1. We can indeed conclude that NFKB1 is binding to the regions in the genome where the peaks are located, so that the chipseq did indeed work. However, NFKB2 also binds to the regions of the peaks so the binding is not that specific. In conclusion, the expected motif (that of NFKB1) is enriched in the ChIP-seq peaks.

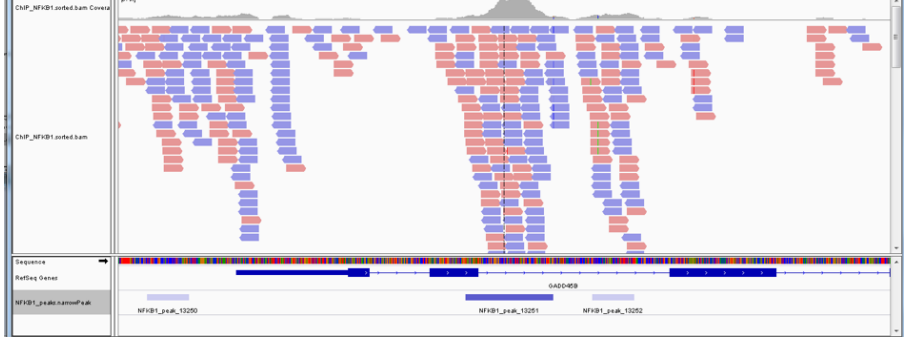
https://gbiomed.kuleuven.be/apps/lcb/i-cisTarget/reports/d43b8fc8a638b7eb1dfe39b2d81bfde13acfbb64/report.html (https://gbiomed.kuleuven.be/apps/lcb/i-cisTarget/reports/d43b8fc8a638b7eb1dfe39b2d81bfde13acfbb64/report.html)

#	Feature	NES	Logo	Recovery Curve	Candidate targets	All regions in top 20000	Database
1	transfac_pro__M04055 Description: V$NFKB2_01: NFKB2 Possible TFs: NFKB2	6.42927			link	link	PWMs
2	cisbp__M5669 Description: NFKB2[gene ID: "ENSG00000077150" species: "Homo sapiens" TF status: "direct" TF family: "Rel" DBDs: "RHD"]; Nfkb2[gene ID: "ENSMUSG00000025225" species: "Mus musculus" TF status: "inferred" TF family: "Rel" DBDs: "RHD"] Possible TFs: NFKB2	6.08241			link	link	PWMs
3	taipale__NFKB1_NGGGGAWTCCCCN Description: NFKB1 dimeric NFAT DBD Possible TFs: NFKB1	5.47777			link	link	PWMs
4	swissregulon__hs__NFKB1_REL_RELA.p2 Description: hs__NFKB1_REL_RELA.p2 Possible TFs: NFKB1, RELA, REL	5.37843			link	link	PWMs
5	hocomoco__NFKB1_f1 Description: NFKB1 Possible TFs: NFKB1	5.01857			link	link	PWMs

5.6 IGV and GREAT tool

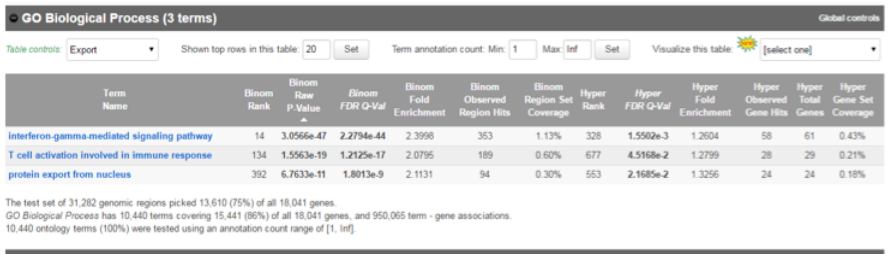
Now we upload the peak file into IGV and look where the peaks are located. Ideally, the genes that were upregulated in RA would now show a peak because our hypothesis was that NKFB1 regulates these genes.

For example, if we look at GADD45B, a gene that was upregulated in RA, we see that there are two peaks located in this gene.



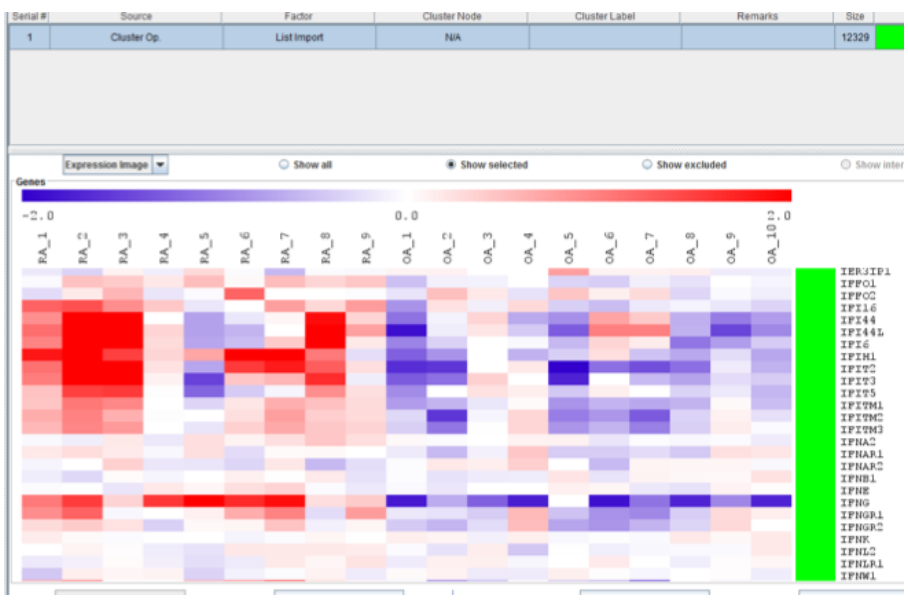
Subsequently we analyze in which genes the 5 highest peaks are located. However these 5 highest peaks aren't located within a gene. With the GREAT Tool you can assign peaks to genes. Association rule: Basal+extension: 5000 bp upstream, 1000 bp downstream, 1000000 bp max extension, curated regulatory domains included.

From GREAT we observe that the five highest peaks are associated with the next genes: CAV3, SSUH2, SRSF6, ZC3H12C. These 4 genes are however not known to be associated with immunological processes. Nevertheless, there are still another 31277 peaks (31282-5) called. GREAT generates a list of genes which are assigned to these 31282 peaks but also the associated biological processes. We see that the genes under the peaks are involved in interfon-gamma-mediated signaling pathway and T cell activation involed in immune respone.



5.7 Integration of expression and Chipseq data

Subsequently we upload the list of genes which GREAT linked to peaks as a cluster in MeV and plot expression values to integrate the expression and chipseq data. Now we see the expression values obtained by microarray of the genes that GREAT linked to peaks.



In the picture above we see that IFNG is upregulated in RA. We also see that IL18R1, IL1R2, HLA-DQA1, GADD45B are upregulated (not shown). Now you can analyze this expression data and investigate how many and which genes that were assigned to a NFKB1 peak are upregulated in rheumatoid arthritis. Or more interesting: how many genes that are upregulated in RA are also bound by NKFB1?

6. Conclusion

Microarray analysis provided a list of genes upregulated in rheumatoid arthritis and Chipseq analysis of the proposed regulator of the upregulated genes in rheumatoid arthrits i.e. NFKB1, provided a list of genes that NFKB1 directly binds. Now to investigate if NFKB1 directly binds the upregulated genes in rheumatoid arthritis, we integrated the expression data and chipseq data.

The next step would be to look at each gene that is upregulated in RA and check if this gene is bound by NKFB1. We did this for a couple of genes (IFNG, IL18R1, IL1R2,...). However, to conclude that NKFB1 is a possible regulator of the upregulated genes in RA, we should investigate all genes, more specially we should investigate how many and which of the upregulated genes are bound by NFKB1.

To conclude, NKFB1 could be a regulator of the upregulated genes in RA but further analysis and research is necessary to conclude this.