METHOD ARTICLE

Using regulatory genomics data to interpret the function of disease variants and prioritise genes from expression studies [version 1; referees: awaiting peer review]

Enrico Ferrero
Computational Biology, GSK, Medicines Research Centre, Stevenage, SG1 2NY, UK

Abstract
The identification of therapeutic targets is a critical step in the research and development of new drugs, with several drug discovery programmes failing because of a weak linkage between target and disease. Genome-wide association studies and large-scale gene expression experiments are providing insights into the biology of several common and complex diseases, but the complexity of transcriptional regulation mechanisms often limit our understanding of how genetic variation can influence changes in gene expression. Several initiatives in the field of regulatory genomics are aiming to close this gap by systematically identifying and cataloguing regulatory elements such as promoters and enhancers across different tissues and cell types. In this Bioconductor workflow, we will explore how different types of regulatory genomic data can be used for the functional interpretation of disease-associated variants and for the prioritisation of gene lists from gene expression experiments.

This article is included in the Bioconductor gateway.
Introduction

Discovering and bringing new drugs to the market is a long, expensive and inefficient process. Increasing the success rates of drug discovery programmes would be transformative to the pharmaceutical industry and significantly improve patients’ access to medicines. Of note, the majority of drug discovery programmes fail for efficacy reasons, with up to 40% of these failures due to a lack of a clear link between the target and the disease under investigation.

Target selection, the first step in drug discovery programmes, is thus a critical decision point. It has previously been shown that therapeutic targets with a genetic link to the disease under investigation are more likely to progress through the drug discovery pipeline, suggesting that genetics can be used as a tool to prioritise and validate drug targets in early discovery.

Over the last decade, genome-wide association studies (GWASs) have revolutionised the field of human genetics, allowing to survey DNA mutations associated with disease and other complex traits on an unprecedented scale. Similarly, phenome-wide association studies (PheWAS) are emerging as a complementary methodology to decipher the genetic bases of the human phenome. While many of these associations might not actually be relevant for the disease aetiology, these methods hold much promise to guide pharmaceutical scientists towards the next generation of drug targets.

Arguably, one of the biggest challenges in translating findings from GWASs to therapies is that the great majority of single nucleotide polymorphisms (SNPs) associated with disease are found in non-coding regions of the genome, and therefore cannot be easily linked to a target gene. Many of these SNPs could be regulatory variants, affecting the expression of nearby or distal genes by interfering with the process of transcription (e.g.: binding of transcription factors at promoters or enhancers).

The most established way to map disease-associated regulatory variants to target genes is probably to use expression quantitative trait loci (eQTLs), variants that affect the expression of specific genes. Over the last few years, the GTEx consortium assembled a valuable resource by performing large-scale mapping of genome-wide correlations between genetic variants and gene expression across 44 human tissues.

However, depending on the power of the study, it might not be possible to detect all existing regulatory variants as eQTLs. An alternative is to use information on the location of promoters and distal enhancers across the genome and link these regulatory elements to their target genes. Large, multi-centre initiatives such as ENCODE, Roadmap Epigenomics and BLUEPRINT mapped regulatory elements in the genome by profiling a number of chromatin features, including DNase hypersensitive sites (DHSs), several types of histone marks and binding of chromatin-associated proteins in a large number of cell lines, primary cell types and tissues. Similarly, the FANTOM consortium used cap analysis of gene expression (CAGE) to identify promoters and enhancers across hundreds of cells and tissues.

Knowing that a certain stretch of DNA is an enhancer is however not informative of the target gene(s). One way to infer links between enhancers and promoters in silico is to identify significant correlations across a large panel of cell types, an approach that was used for distal and promoter DHSs as well as for CAGE-defined promoters and enhancers. Experimental methods to assay interactions between regulatory elements also exist. Chromatin interaction analysis by paired-end tag sequencing (ChIA-PET) couples chromatin immunoprecipitation with DNA ligation and sequencing to identify regions of DNA that are interacting thanks to the binding of a specific protein. Promoter capture Hi-C extends chromatin conformation capture by using “baits” to enrich for promoter interactions and increase resolution.

Overall, linking genetic variants to their candidate target genes is not straightforward, not only because of the complexity of the human genome and transcriptional regulation, but also because of the variety of data types and approaches that can be used. To address this, we developed STOPGAP (systematic target opportunity assessment by genetic association predictions), a database of disease variants mapped to their most likely target gene(s) using different types of regulatory genomic data. The database is currently undergoing a major overhaul and will eventually be superseded by POSTGAP. A similar resource and valid alternative is INFERNO (inferring the molecular mechanisms of noncoding variants).
Workflow Overview
In this workflow, we will explore how regulatory genomic data can be used to connect the genetic and transcriptional layers by providing a framework for the functional annotation of SNPs from GWASs. We will use eQTL data from GTEx\cite{14}, FANTOM5 correlations between promoters and enhancers\cite{21} and promoter capture Hi-C data\cite{25}.

We start with a common scenario: we ran a RNA-seq experiment comparing patients with a disease and healthy individuals, and would like to discover key disease genes and potential therapeutic targets by integrating genetic information in our analysis.

Install required packages
R version 3.4.2 and Bioconductor version 3.6 were used for the analysis. The code below will install all required packages and dependencies from Bioconductor and CRAN:

```r
source("https://bioconductor.org/biocLite.R")
# uncomment the following line to install packages
#biocLite(c("DESeq2", "GenomicFeatures", "GenomicRanges", "ggplot2", "gwascat", "recount", "pheatmap", "RColorBrewer", "rtracklayer", "R.utils", "splitstackshape", "VariantAnnotation"))
```

Gene expression data and differential gene expression analysis
The RNA-seq data we will be using comes from blood of patients with systemic lupus erythematosus (SLE) and healthy controls\cite{28}.

We are going to use recount\cite{29} to obtain gene-level counts:

```r
library(recount)
# uncomment the following line to download dataset
download_study("SRP062966")
load(file.path("SRP062966", "rse_gene.RData"))
rse <- scale_counts(rse_gene)
rse
```

Other Bioconductor packages that can be used to access data from gene expression experiments directly in R are GEOquery\cite{30} and ArrayExpress\cite{31}.

So, we have 117 samples. This is what the data looks like:

```r
assay(rse)[1:10, 1:10]
```

```
#  SRR2443263 SRR2443262 SRR2443261 SRR2443260 SRR2443259
#  SRR2443263 19 6 10 10 8
#  SRR2443262 0 0 0 0 0
#  SRR2443261 489 238 224 323 281
#  SRR2443260 594 503 530 670 775
#  SRR2443259 232 173 166 252 268
#  SRR2443258 21554 18918 14260 19869 26586
#  SRR2443257 94 57 45 59 35
#  SRR2443256 500 397 358 407 500
```
We note that genes are annotated using the GENCODE v25 annotation, which will be useful later on. Let’s look at the metadata to check how we can split samples between cases and controls:

colData(rse)

## DataFrame with 117 rows and 21 columns
##
##                project      sample  experiment         run          read_count_as_reported_by_sra reads_downloaded
##            <character> <character> <character> <character> <numeric>                  <integer>
## SRR2443263   SRP062966  SRS1048033  SRX1168388  SRR2443263          103977424        103977424
## SRR2443262   SRP062966  SRS1048034  SRX1168387  SRR2443262          125900891        125900891
## SRR2443261   SRP062966  SRS1048035  SRX1168386  SRR2443261          129803063        129803063
## SRR2443260   SRP062966  SRS1048036  SRX1168385  SRR2443260          105335395        105335395
## SRR2443259   SRP062966  SRS1048037  SRX1168384  SRR2443259          101692332        101692332
## ...                                  ...              ...
## SRR2443151   SRP062966  SRS1048145  SRX1168276  SRR2443151          87315854         87315854
## SRR2443150   SRP062966  SRS1048146  SRX1168275  SRR2443150          96825506         96825506
## SRR2443148   SRP062966  SRS1048147  SRX1168273  SRR2443148         121365435        121365435
## SRR2443147   SRP062966  SRS1048148  SRX1168272  SRR2443147         104038425        104038425
## SRR2443149   SRP062966  SRS1048149  SRX1168274  SRR2443149         113083096        113083096
##
## proportion_of_reads_reported_by_sra_downloaded paired_end
##                                                 <numeric>  <logical>
## SRR2443263                                              1      FALSE
## SRR2443262                                              1      FALSE
## SRR2443261                                              1      FALSE
## SRR2443260                                              1      FALSE
## SRR2443259                                              1      FALSE
## ...                                                   ...        ...
## SRR2443151                                              1      FALSE
## SRR2443150                                              1      FALSE
## SRR2443148                                              1      FALSE
## SRR2443147                                              1      FALSE
## SRR2443149                                              1      FALSE
## sra_misreported_paired_end mapped_read_count auc
## <logical> <integer> <numeric>
## SRR2443263 FALSE 103499268 514933280
## SRR2443262 FALSE 125499809 6244059473
## SRR2443261 FALSE 125043355 6201504759
## SRR2443260 FALSE 104872856 5211910530
## SRR2443259 FALSE 101258496 5033612693
## ... ... ... ... ...
## SRR2443151 FALSE 86874384 4319264868
## SRR2443150 FALSE 96316303 4787601223
## SRR2443148 FALSE 120819733 6009515064
## SRR2443147 FALSE 103588909 5153702232
## SRR2443149 FALSE 112640054 5598306153
#### sharq_beta_tissue sharq_beta_cell_type
## <character> <character>
## SRR2443263 NA NA
## SRR2443262 NA NA
## SRR2443261 NA NA
## SRR2443260 NA NA
## SRR2443259 NA NA
## ... ... ... ...
## SRR2443151 NA NA
## SRR2443150 NA NA
## SRR2443148 NA NA
## SRR2443147 NA NA
## SRR2443149 NA NA
#### biosample_submission_date biosample_publication_date
## <character> <character>
## SRR2443263 2015-08-28T16:41:29.000 2015-09-16T01:24:17.350
## SRR2443261 2015-08-28T16:41:27.000 2015-09-16T01:24:14.823
## SRR2443260 2015-08-28T16:41:35.000 2015-09-16T01:24:13.450
## SRR2443259 2015-08-28T16:41:33.000 2015-09-16T01:24:12.433
## ... ... ... ...
## SRR2443151 2015-08-28T16:42:24.000 2015-09-16T01:19:06.787
## SRR2443150 2015-08-28T16:42:23.000 2015-09-16T01:19:05.557
## SRR2443149 2015-08-28T16:42:22.000 2015-09-16T01:19:04.583
#### biosample_update_date avg_read_length geo_accession
## <character> <integer> <character>
## SRR2443263 2015-09-16T01:28:05.297 50 GSM1863749
## SRR2443262 2015-09-16T01:28:05.027 50 GSM1863748
## SRR2443261 2015-09-16T01:28:04.803 50 GSM1863747
## SRR2443260 2015-09-16T01:28:04.587 50 GSM1863746
## SRR2443259 2015-09-16T01:28:04.347 50 GSM1863745
## ... ... ...
## SRR2443151 2015-09-16T01:23:41.897 50 GSM1863637
## SRR2443150 2015-09-16T01:23:41.453 50 GSM1863636
## SRR2443148 2015-09-16T01:23:41.093 50 GSM1863634
## SRR2443147 2015-09-16T01:23:40.840 50 GSM1863633
## SRR2443149 2015-09-16T01:23:40.597 50 GSM1863635
#### bigwig_file title
## <character> <character>
## SRR2443263 SRR2443263.bw control18
## SRR2443262 SRR2443262.bw control17
## SRR2443261 SRR2443261.bw control16
## SRR2443260 SRR2443260.bw control15
## SRR2443259 SRR2443259.bw control14
## ... ... ... ...

Page 5 of 38
The most interesting part of the metadata is contained in the characteristics column, which is a CharacterList object:

colData(rse)$characteristics

## CharacterList of length 117
## [[1]] disease status: healthy tissue: whole blood anti-ro: control ism: control
## [[2]] disease status: healthy tissue: whole blood anti-ro: control ism: control
## [[3]] disease status: healthy tissue: whole blood anti-ro: control ism: control
## [[4]] disease status: healthy tissue: whole blood anti-ro: control ism: control
## [[5]] disease status: healthy tissue: whole blood anti-ro: control ism: control
## [[6]] disease status: healthy tissue: whole blood anti-ro: control ism: control
## [[7]] disease status: healthy tissue: whole blood anti-ro: control ism: control
## [[8]] disease status: healthy tissue: whole blood anti-ro: control ism: control
## [[9]] disease status: healthy tissue: whole blood anti-ro: control ism: control
## [[10]] disease status: healthy tissue: whole blood anti-ro: control ism: control
## ... 
## <107 more elements>
Let’s create some new columns with this information that can be used for the differential expression analysis. We will also make sure that they are encoded as factors and that the correct reference layer is used:

```r
# disease status
colData(rse)$disease_status <- sapply(colData(rse)$characteristics, "[", 1)
colData(rse)$disease_status <- sub("disease status: ", ",", colData(rse)$disease_status)
colData(rse)$disease_status <- sub("systemic lupus erythematosus \(SLE\)"", "SLE", colData(rse)$disease_status)
colData(rse)$disease_status <- factor(colData(rse)$disease_status, levels = c("healthy", "SLE"))

# tissue
colData(rse)$tissue <- sapply(colData(rse)$characteristics, "[", 2)
colData(rse)$tissue <- sub("tissue: ", ",", colData(rse)$tissue)
colData(rse)$tissue <- factor(colData(rse)$tissue)

# anti-ro
colData(rse)$anti_ro <- sapply(colData(rse)$characteristics, "[", 3)
colData(rse)$anti_ro <- sub("anti-ro: ", ",", colData(rse)$anti_ro)
colData(rse)$anti_ro <- factor(colData(rse)$anti_ro)

# ism
colData(rse)$ism <- sapply(colData(rse)$characteristics, "[", 4)
colData(rse)$ism <- sub("ism: ", ",", colData(rse)$ism)
colData(rse)$ism <- factor(colData(rse)$ism)
```

We can have a look at the new format:

```r
colData(rse)[c("disease_status", "tissue", "anti_ro", "ism")]
```

```
            disease_status      tissue  anti_ro      ism
                  <factor>    <factor> <factor> <factor>
SRR2443263        healthy whole blood  control  control
SRR2443262        healthy whole blood  control  control
SRR2443261        healthy whole blood  control  control
SRR2443260        healthy whole blood  control  control
SRR2443259        healthy whole blood  control  control
...                   ...         ...      ...      ...
SRR2443151            SLE whole blood      med  ISM_low
SRR2443150            SLE whole blood     high  ISM_low
SRR2443148            SLE whole blood     high  ISM_high
SRR2443147            SLE whole blood     high  ISM_high
SRR2443149            SLE whole blood     high  ISM_high
```

It looks more readable. Let’s now check how many samples we have in each group:

```r
table(colData(rse)$disease_status)
```

```
# healthy     SLE
# 18      99
```

To speed up code execution we will limit the number of SLE samples. For simplicity, we select the first 18 (healthy) and the last 18 (SLE) samples from the original `RangedSummarizedExperiment` object:

```r
rse <- rse[, c(1:18, 82:99)]
```
Now we are ready to perform a simple differential gene expression analysis with DESeq2:\[33\]:

```
library(DESeq2)
dds <- DESeqDataSet(rse, ~ disease_status)
dds <- DESeq(dds)
```

```
## class: DESeqDataSet
## dim: 58037 36
## metadata(1): version
## assays(5): counts mu cooks replaceCounts replaceCooks
## rownames(58037): ENSG00000000003.14 ENSG00000000005.5 ...
##   ENSG00000283698.1 ENSG00000283699.1
## rowData names(25): gene_id bp_length ... maxCooks replace
## colnames(36): SRR2443263 SRR2443262 ... SRR2443166 SRR2443165
## colData names(27): project sample ... sizeFactor replaceable
```

Note that we used an extremely simple model; in the real world you will probably need to account for co-variables, potential confounders and interactions between them. edgeR\[34\] and limma\[35\] are good alternatives to DESeq2 for performing differential expression analyses.

We can now look at the data in more detail. We use the variance stabilising transformation (VST)\[36\] for visualisation purposes:

```
vst <- vst(dds, blind = FALSE)
```

First, let’s look at distances between samples to see if we can recover a separation between SLE and healthy samples:

```
sampleDists <- as.matrix(dist(t(assay(vsd))))
rownames(sampleDists) <- vsd$disease_status
sampleDists[c(1, 18, 19, 36), c(1, 18, 19, 36)]
```

```
##         SRR2443263 SRR2443248 SRR2443182 SRR2443165
## healthy    0.00000   106.6933   93.30292   99.84061
## healthy  106.69330     0.0000  115.87958  127.27997
## SLE       93.30292   115.8796    0.00000  115.06568
## SLE       99.84061   127.2800  115.06568    0.00000
```

We will use the pheatmap and RColorBrewer packages for drawing the heatmap (Figure 1).

```
library(pheatmap)
library(RColorBrewer)
```

```
colors <- colorRampPalette(rev(brewer.pal(9, "Blues")))(255)
pheatmap(sampleDists, col = colors)
```

Similarly, we can perform a principal component analysis (PCA) on the most variable 500 genes (Figure 2).

```
pplotPCA(vsd, intgroup = "disease_status")
```
Figure 1. Clustered heatmap showing distances between samples.

Figure 2. Principal component analysis with samples coloured according to their disease status.
This looks better, we can see some separation of healthy and SLE samples along both PC1 and PC2, though some SLE samples appear very similar to the healthy ones. Next, we select genes that are differentially expressed below a 0.05 adjusted p-value threshold:

```
res <- results(dds, alpha = 0.05)
res
```

```results
## log2 fold change (MLE): disease status SLE vs healthy
## Wald test p-value: disease status SLE vs healthy
## DataFrame with 58037 rows and 6 columns
## baseMean log2FoldChange lfcSE stat
## <numeric> <numeric> <numeric> <numeric>
## ENSG00000000003.14 10.4189981 -0.20051804 0.24868451 -0.80631496
## ENSG00000000005.5 0.0317823 0.03330732 2.96442394 0.01123568
## ENSG000000000419.12 389.9025130 0.66288230 0.11427371 5.80082925
## ENSG000000000457.13 636.6928414 0.17336365 0.08062862 2.15015047
## ENSG000000000460.16 234.6479796 0.20589404 0.07445624 2.76530274
## ...                        ...            ...        ...          ...
## ENSG000000000283699.1 0.5398951 -0.003056215 0.7578201 -0.004032903
##                          pvalue         padj
## <numeric>    <numeric>
## ENSG00000000003.14 4.200613e-01 6.706002e-01
## ENSG00000000005.5 9.910354e-01           NA
## ENSG000000000419.12 6.598777e-09 3.058479e-06
## ENSG000000000457.13 3.154331e-02 1.463634e-01
## ENSG000000000460.16 5.686999e-03 4.643041e-02
## ...                        ...          ...
## ENSG000000000283699.1 0.9967822 NA
```

We can look at a summary of the results:

```
summary(res)
```

```summary
## out of 43005 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up) : 2526, 5.9%
## LFC < 0 (down) : 1069, 2.5%
## outliers [1] : 0, 0%
## low counts [2] : 14735, 34%
```

We can also visualise the log fold changes using an MA plot (Figure 3).

```
plotMA(res, ylim = c(-5,5))
```
Figure 3. MA plot showing genes differentially expressed in SLE patients compared to healthy patients.

For convenience, we will save our differentially expressed genes (DEGs) in another object:

```r
degs <- subset(res, padj < 0.05)
degs <- as.data.frame(degs)
head(degs)
```

We also map the GENCODE gene IDs to gene symbols using the annotation in the original RangedSummarize-dExperiment object, which is going to be convenient later on:

```r
rowData(rse)
```
```r
# Accessing GWAS data

We have more than 3500 genes of interest at this stage. Since we know that therapeutic targets with genetic evidence are more likely to progress through the drug discovery pipeline, one way to prioritise them could be to check which of these can be genetically linked to SLE. To get hold of relevant GWAS data, we will be using the gwascat Bioconductor package, which provides an interface to the GWAS catalog. An alternative is to use the GRASP database with the grasp2db package.

```
## [2] chr13 [ 39776775,  39776775]   * | Psoriasis rs7993214

# P-VALUE
# <numeric>
## [1] 1e-13
## [2] 2e-06
## [3] 3e-18
## [4] 7e-06
## [5] 8e-06

seqinfo: 23 sequences from GRCh38 genome; no seqlengths

snps is a gwasloc object which is simply a wrapper around a GRanges object, the standard way to express genomic ranges in Bioconductor. We are interested in SNPs associated with SLE:

```r
snps <- subsetByTraits(snps, tr = "Systemic lupus erythematosus")
```

## gwasloc instance with 402 records and 37 attributes per record.
## Extracted: 2017-12-04
## Genome:  GRCh38
## Excerpt:
## GRanges object with 5 ranges and 3 metadata columns:
##       seqnames                 ranges strand |
##          <Rle>              <IRanges>  <Rle> |
##   [1]    chr16 [ 31301932,  31301932]      * |
##   [2]    chr11 [   589564,    589564]      * |
##   [3]     chr3 [ 58384450,  58384450]      * |
##   [4]     chr1 [173340574, 173340574]      * |
##   [5]     chr8 [ 11491677,  11491677]      * |
##                      DISEASE/TRAIT        SNPS   P-VALUE
##                        <character> <character> <numeric>
##   [1] Systemic lupus erythematosus   rs9888739     2e-23
##   [2] Systemic lupus erythematosus   rs4963128     3e-10
##   [3] Systemic lupus erythematosus   rs6445975     7e-09
##   [4] Systemic lupus erythematosus  rs10798269     1e-07
##   [5] Systemic lupus erythematosus  rs13277113     1e-10

seqinfo: 23 sequences from GRCh38 genome; no seqlengths

We can visualise these as a Manhattan plot to look at the distribution of GWAS p-values over chromosomes on a negative log scale (Figure 4). Note that p-values lower than 1e-25 are truncated in the figure and that we have to load ggplot2\(^4\) to modify the look of the plot:

```r
library(ggplot2)
traitsManh(gwr = snps, sel = snps, traits = "Systemic lupus erythematosus")  
  + theme(legend.position = "none",  
          axis.title.x = element_blank(),  
          axis.text.x = element_blank())
```

We note here that genotyping arrays typically include a very small fraction of all possible SNPs in the human genome, and there is no guarantee that the tag SNPs on the array are the true causal SNPs\(^4\). The alleles of other SNPs can be imputed from tag SNPs thanks to the structure of linkage disequilibrium (LD) blocks present in chromosomes. Thus, when linking variants to target genes in a real-world setting, it is important to take into consideration neighbouring SNPs that are in high LD and inherited with the tag SNPs. For simplicity, we will skip this LD expansion.
step and refer the reader to the Ensembl REST API\textsuperscript{43}, the Ensembl Linkage Disequilibrium Calculator and the Bioconductor packages trio\textsuperscript{44} and ldblock\textsuperscript{45} to perform this task.

Annotation of coding and proximal SNPs to target genes
In order to annotate these variants, we need a a TxDb object, a reference of where transcripts are located on the genome. We can build this using the GenomicFeatures\textsuperscript{46} package and the GENCODE v25 gene annotation:

```r
library(GenomicFeatures)
# uncomment the following line to download file
gencode.v25.annotation.gff3.gz", destfile = "gencode.v25.annotation.gff3.gz")

txdb <- makeTxDbFromGFF("gencode.v25.annotation.gff3.gz")

## TxDb object:
## # Db type: TxDb
## # Supporting package: GenomicFeatures
## # Data source: gencode.v25.annotation.gff3.gz
## # Organism: NA
## # Taxonomy ID: NA
## # miRBase build ID: NA
## # Genome: NA
## # transcript_nrow: 198093
## # exon_nrow: 1182765
## # cds_nrow: 704859
```
We also have to convert the `gwasloc` object into a standard `GRanges` object:

```r
snps <- GRanges(snps)
```

Let's check if the `gwasloc` and `TxDb` object use the same notation for chromosomes:

```r
seqlevelsStyle(snps)
## [1] "UCSC"

seqlevels(snps)
## [1] "chr1" "chr13" "chr15" "chr3" "chr8" "chr11" "chr18" "chr10"
## [9] "chr7" "chr12" "chr2" "chr6" "chr4" "chr19" "chrX" "chr16"
## [17] "chr20" "chr5" "chr14" "chr17" "chr21" "chr9" "chr22"

seqlevelsStyle(txdb)
## [1] "UCSC"

seqlevels(txdb)
## [1] "chr1" "chr2" "chr3" "chr4" "chr5" "chr6" "chr7" "chr8"
## [9] "chr9" "chr10" "chr11" "chr12" "chr13" "chr14" "chr15" "chr16"
## [17] "chr17" "chr18" "chr19" "chr20" "chr21" "chr22" "chrX" "chrY"
## [25] "chrM"
```

OK, they do. Now we can annotate our SNPs to genes using the `VariantAnnotation` package:

```r
library(VariantAnnotation)
snps_anno <- locateVariants(snps, txdb, AllVariants())
snps_anno <- unique(snps_anno)

snps_anno
```

```r
# GRanges object with 299 ranges and 9 metadata columns:
## # seqnames ranges strand | LOCATION LOCSTART LOCEND QUERYID TXID CDSID GENEID
## # <Rle> <IRanges> <Rle> | <factor> <integer> <integer> <character> <IntegerList> <character> <character>
## # [1] chr16 [3 31301932, 31301932] + | intron 40161 1 143788 ENSG00000169896.16
## # [3] chr3 58384450, 58384450] + | intron 51074 3 34101 ENSG00000168297.15
## # [4] chr1 [173340574, 173340574] * | intergenic <NA> 4 34101 ENSG00000168297.15
## # [5] chr8 [11491677, 11491677] * | intergenic <NA> 5 17707 5 ENSG00000168297.15
## ... ... ... ... ... ... ... ... ...
## [295] chr6 [137874014, 137874014] + | intron 6162 6 143788 ENSG00000169896.16
## [296] chr6 [32619077, 32619077] * | intergenic <NA> 7 17707 6 ENSG00000169896.16
## [297] chr6 [137685367, 137685367] + | intron 11552 8 34101 ENSG00000169896.16
## [298] chrX [153924366, 153924366] - | intergenic <NA> 9 17707 5 ENSG00000169896.16
## [299] chr5 [160459613, 160459613] * | intergenic <NA> 10 17707 5 ENSG00000169896.16
## # LOCEND QUERYID TXID CDSID GENEID
## # <integer> <integer> <character> <IntegerList> <character> <character>
## # [1] 40161 1 143788 ENSG00000169896.16
## # [2] 12531 2 99581 ENSG00000070047.11
## # [3] 51074 3 34101 ENSG00000168297.15
```
We lost all the metadata from the original `snps` object, but we can recover it using the `QUERYID` column in `snps_anno`. We will only keep the SNP IDs and GWAS p-values:

```r
snps_metadata <- snps[snps_anno$QUERYID]
mcols(snps_anno) <- cbind(mcols(snps_metadata)[c("SNPS", "P-VALUE")],
mcols(snps_anno))
```

---

We see that the QQ-plot indicates significant associations between the phenotypes and the genotypes:
## LOCATION LOCSTART LOCEND QUERYID TXID CDSID
## <factor> <integer> <integer> <integer> <character> <IntegerList>
## [1] intron 40161 40161 1 143788
## [2] intron 12531 12531 2 99581
## [3] intron 51074 51074 3 34101
## [4] intergenic <NA> <NA> 4 <NA>
## [5] intergenic <NA> <NA> 5 <NA>
## ... ... ... ... ... ... ... ...
## [295] intron 6162 6162 393 64150
## [296] intergenic <NA> <NA> 397 <NA>
## [297] intron 11552 11552 398 64145
## [298] intron 1770 1770 399 196900
## [299] intergenic <NA> <NA> 400 <NA>

## GENEID
## <character>
## [1] ENSG00000169896.16
## [2] ENSG00000070047.11
## [3] ENSG00000168297.15
## [4] <NA>
## [5] <NA>
## ... ... ...
## [295] ENSG000000118503.14
## [296] <NA>
## [297] ENSG00000230533.2
## [298] ENSG00000089820.15
## [299] <NA>

## PRECEDEID
## <CharacterList>
## [1]
## [2]
## [3]
## [4] ENSG00000076321.10,ENSG00000117592.8,ENSG00000117593.9,...
## [5] ENSG00000079459.12,ENSG00000136573.12,ENSG00000136574.17,...
## ... ...
## [295] ENSG00000030110.12,ENSG00000112473.17,ENSG00000112511.17,...
## [296] <NA>
## [297]
## [298]
## [299] ENSG000000118322.12,ENSG000000145864.12,ENSG00000253417.5,...
## FOLLOWID
## <CharacterList>
## [1]
## [2]
## [3]
## [4] ENSG000000094975.13,ENSG00000117560.7,ENSG00000117586.10,...
## [5] ENSG00000104643.9,ENSG00000154316.15,ENSG00000154319.14,...
## ... ...
## [295] ENSG000000166278.14,ENSG00000168477.17,ENSG00000196126.10,...
## [296] <NA>
## [297]
## [298]
## [299] ENSG00000113312.10,ENSG00000135083.14,ENSG00000145861.7,...

We can visualise where these SNPs are located with ggplot2 (Figure 5).
Figure 5. Barplot showing genomic locations associated with SLE variants.

loc <- data.frame(table(snps_anno$LOCATION))
ggplot(data = loc, aes(x = reorder(Var1, -Freq), y = Freq)) +
  geom_bar(stat="identity")

As expected, the great majority of SNPs are located within introns and in intergenic regions. For the moment, we will focus on SNPs that are either coding or in promoter and UTR regions, as these can be assigned to target genes rather unambiguously:

snps_easy <- subset(snps_anno, LOCATION == "coding" | LOCATION == "promoter" | LOCATION == "threeUTR" | LOCATION == "fiveUTR")
snps_easy <- as.data.frame(snps_easy)
head(snps_easy)

<table>
<thead>
<tr>
<th>seqnames</th>
<th>start</th>
<th>end</th>
<th>width</th>
<th>strand</th>
<th>SNPS</th>
<th>P.VALUE</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr4</td>
<td>101829919</td>
<td>101829919</td>
<td>1</td>
<td>+</td>
<td>rs10516487 4e-10</td>
<td>coding</td>
<td></td>
</tr>
<tr>
<td>chr7</td>
<td>128954129</td>
<td>128954129</td>
<td>1</td>
<td>-</td>
<td>rs10488631 2e-11</td>
<td>promoter</td>
<td></td>
</tr>
<tr>
<td>chr11</td>
<td>55368743</td>
<td>55368743</td>
<td>1</td>
<td>+</td>
<td>rs7927370 7e-06</td>
<td>coding</td>
<td></td>
</tr>
<tr>
<td>chr6</td>
<td>137874929</td>
<td>137874929</td>
<td>1</td>
<td>+</td>
<td>rs2230926 1e-17</td>
<td>coding</td>
<td></td>
</tr>
<tr>
<td>chr11</td>
<td>118702810</td>
<td>118702810</td>
<td>1</td>
<td>+</td>
<td>rs4639966 1e-16</td>
<td>promoter</td>
<td></td>
</tr>
<tr>
<td>chr16</td>
<td>30624338</td>
<td>30624338</td>
<td>1</td>
<td>-</td>
<td>rs7186852 3e-07</td>
<td>promoter</td>
<td></td>
</tr>
<tr>
<td>LOCSTART</td>
<td>LOCEND</td>
<td>QUERYID</td>
<td>TXID</td>
<td>CDSID</td>
<td>GENEID</td>
<td>PRECEDEID</td>
<td></td>
</tr>
<tr>
<td>137</td>
<td>137</td>
<td>46105</td>
<td>170258</td>
<td></td>
<td>ENSG00000153064.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NA</td>
<td>NA</td>
<td>23</td>
<td>77786</td>
<td></td>
<td>ENSG00000275106.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>860</td>
<td>860</td>
<td>45</td>
<td>101610</td>
<td>370677</td>
<td>ENSG00000181958.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>380</td>
<td>380</td>
<td>57</td>
<td>64150</td>
<td>232398</td>
<td>ENSG00000118503.14</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Now we can check if any of the genes we found to be differentially expressed in SLE is also genetically associated with the disease:

```r
snps_easy_in_degs <- merge(degs, snps_easy, by.x = "gene_id", by.y = "GENEID", all = FALSE)
```

```r
tsnp_easy_in_degs
```

---

### DataFrame with 7 rows and 24 columns

<table>
<thead>
<tr>
<th>gene_id</th>
<th>bp_length</th>
<th>symbol</th>
<th>baseMean</th>
<th>log2FoldChange</th>
<th>lfcSE</th>
<th>stat</th>
<th>pvalue</th>
<th>padj</th>
<th>seqnames</th>
<th>start</th>
<th>end</th>
<th>width</th>
<th>strand</th>
<th>SNPS</th>
<th>P.VALUE</th>
<th>LOCATION</th>
<th>LOCSTART</th>
<th>LOCEND</th>
<th>QUERYID</th>
<th>TXID</th>
<th>CDSID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG00000096968</td>
<td>6170</td>
<td>JAK2</td>
<td>1279.478</td>
<td>0.4854343</td>
<td>0.1553513</td>
<td>3.124753</td>
<td>1.779545e-03</td>
<td>2.068794e-02</td>
<td>chr9</td>
<td>4984530</td>
<td>4984530</td>
<td>1</td>
<td>+</td>
<td>rs1887428</td>
<td>fiveUTR</td>
<td>141</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENSG00000099834</td>
<td>3873</td>
<td>CDHR5</td>
<td>10.02177</td>
<td>0.8539586</td>
<td>0.2666557</td>
<td>3.202476</td>
<td>1.362316e-03</td>
<td>1.732902e-02</td>
<td>chr11</td>
<td>625085</td>
<td>625085</td>
<td>1</td>
<td>+</td>
<td>rs58688157</td>
<td>162267541</td>
<td>126267541</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENSG00000115267</td>
<td>4528</td>
<td>IFIH1</td>
<td>1415.9133</td>
<td>1.1494945</td>
<td>0.2729847</td>
<td>4.210838</td>
<td>2.544247e-05</td>
<td>1.120363e-03</td>
<td>chr2</td>
<td>60559729</td>
<td>60559729</td>
<td>1</td>
<td>+</td>
<td>rs1990760</td>
<td>162267541</td>
<td>126267541</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENSG00000120280</td>
<td>1855</td>
<td>CXorf21</td>
<td>637.78094</td>
<td>0.7819504</td>
<td>0.1541707</td>
<td>5.071977</td>
<td>3.937038e-07</td>
<td>5.047898e-05</td>
<td>chrX</td>
<td>30559729</td>
<td>30559729</td>
<td>1</td>
<td>+</td>
<td>rs887369</td>
<td>coding</td>
<td>627</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENSG00000185507</td>
<td>2628</td>
<td>IRF7</td>
<td>4883.209</td>
<td>1.4062704</td>
<td>0.2992536</td>
<td>4.699260</td>
<td>2.611057e-06</td>
<td>2.298336e-04</td>
<td>chr12</td>
<td>614318</td>
<td>614318</td>
<td>1</td>
<td>+</td>
<td>rs1061502</td>
<td>coding</td>
<td>217</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENSG00000204366</td>
<td>1875</td>
<td>ZBTB12</td>
<td>22.99200</td>
<td>-0.3892298</td>
<td>0.1348705</td>
<td>-2.885952</td>
<td>3.902318e-03</td>
<td>3.584479e-02</td>
<td>chr6</td>
<td>31902549</td>
<td>31902549</td>
<td>1</td>
<td>-</td>
<td>rs10488631</td>
<td>promoter</td>
<td>NA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---
So, we have 7 genes showing differential expression in SLE that are also genetically associated with the disease. While this is an interesting result, these hits are likely to be already well-known as potential SLE targets given their clear genetic association.

We will store essential information about these hits in a results `data.frame`:

```r
prioritised_hits <- unique(data.frame(
  snp_id = snps_easy_in_degs$SNPS,
  snp_pvalue = snps_easy_in_degs$P.VALUE,
  snp_location = snps_easy_in_degs$LOCATION,
  gene_id = snps_easy_in_degs$gene_id,
  gene_symbol = snps_easy_in_degs$symbol,
  gene_pvalue = snps_easy_in_degs$padj,
  gene_log2foldchange = snps_easy_in_degs$log2FoldChange))
```

Use of regulatory genomic data to map intronic and intergenic SNPs to target genes

But what about all the SNPs in introns and intergenic regions? Some of those might be regulatory SNPs affecting the expression level of their target gene(s) through a distal enhancer. Let’s create a dataset of candidate regulatory SNPs that are either intronic or intergenic and remove the annotation obtained with `VariantAnnotation`:

```r
snps_hard <- subset(snps_anno, LOCATION == "intron" | LOCATION == "intergenic", select = c("SNPS", "P.VALUE", "LOCATION"))
```

---

## ENSG00000120280       627       192      194672                   692823
## ENSG00000185507       217       317      105777 385431,385427,385428,...
## ENSG00000204366        NA       116       65993
## ENSG00000275106        NA        23       77786
## PRECEDEID FOLLOWID
## <list>   <list>
## ENSG00000096968
## ENSG00000099834
## ENSG00000115267
## ENSG00000120280
## ENSG00000185507
## ENSG00000204366
## ENSG00000275106

Use of regulatory genomic data to map intronic and intergenic SNPs to target genes

But what about all the SNPs in introns and intergenic regions? Some of those might be regulatory SNPs affecting the expression level of their target gene(s) through a distal enhancer. Let’s create a dataset of candidate regulatory SNPs that are either intronic or intergenic and remove the annotation obtained with `VariantAnnotation`:

```r
snps_hard <- subset(snps_anno, LOCATION == "intron" | LOCATION == "intergenic", select = c("SNPS", "P.VALUE", "LOCATION"))
```

---

## ENSG00000120280       627       192      194672                   692823
## ENSG00000185507       217       317      105777 385431,385427,385428,...
## ENSG00000204366        NA       116       65993
## ENSG00000275106        NA        23       77786
## PRECEDEID FOLLOWID
## <list>   <list>
## ENSG00000096968
## ENSG00000099834
## ENSG00000115267
## ENSG00000120280
## ENSG00000185507
## ENSG00000204366
## ENSG00000275106

Use of regulatory genomic data to map intronic and intergenic SNPs to target genes

But what about all the SNPs in introns and intergenic regions? Some of those might be regulatory SNPs affecting the expression level of their target gene(s) through a distal enhancer. Let’s create a dataset of candidate regulatory SNPs that are either intronic or intergenic and remove the annotation obtained with `VariantAnnotation`:

```r
snps_hard <- subset(snps_anno, LOCATION == "intron" | LOCATION == "intergenic", select = c("SNPS", "P.VALUE", "LOCATION"))
```
eQTL data. A well-established way to gain insights into target genes of regulatory SNPs is to use eQTL data, where correlations between genetic variants and expression of genes are computed across different tissues or cell types\textsuperscript{13}. We will use blood eQTL data from the GTEx consortium\textsuperscript{14}. To get the data, you will have to register and download the file GTEx\textsubscript{Analysis}\textsubscript{v7}\_eQTL.tar.gz from the GTEx portal to the current working directory:

```r
# uncomment the following line to extract the gzipped archive file
#untar("GTEx\textsubscript{Analysis}\textsubscript{v7}\_eQTL.tar.gz")
gtex\_blood &lt;
read\_delim(gzfile("GTEx\textsubscript{Analysis}\textsubscript{v7}\_eQTL/Whole\_Blood.v7.signif\_variant\_gene\_pairs.txt.gz"), stringsAsFactors = FALSE)
head(gtex\_blood)
```

<table>
<thead>
<tr>
<th>variant_id</th>
<th>gene_id</th>
<th>tss_distance</th>
<th>ma_samples</th>
<th>ma_count</th>
<th>maf</th>
<th>pval_nominal</th>
<th>slope</th>
<th>slope_se</th>
<th>pval_nominal_threshold</th>
<th>min_pval_nominal</th>
<th>pval_beta</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 _231153_CTT_C_b37</td>
<td>ENSG00000223972.4</td>
<td>219284</td>
<td>13</td>
<td>13</td>
<td>0.0191740</td>
<td>3.69025e-08</td>
<td>1.319720</td>
<td>0.233538</td>
<td>1.35366e-04</td>
<td>3.69025e-08</td>
<td>4.67784e-05</td>
</tr>
<tr>
<td>2 _61920_G_A_b37</td>
<td>ENSG00000238009.2</td>
<td>-67303</td>
<td>18</td>
<td>20</td>
<td>0.0281690</td>
<td>7.00836e-07</td>
<td>0.903786</td>
<td>0.178322</td>
<td>8.26088e-05</td>
<td>6.50297e-10</td>
<td>8.52870e-05</td>
</tr>
<tr>
<td>3 _64649_A_C_b37</td>
<td>ENSG00000238009.2</td>
<td>-64574</td>
<td>16</td>
<td>16</td>
<td>0.0220386</td>
<td>5.72066e-07</td>
<td>1.110040</td>
<td>0.217225</td>
<td>8.26088e-05</td>
<td>6.50297e-10</td>
<td>8.52870e-05</td>
</tr>
<tr>
<td>4 _115746_C_T_b37</td>
<td>ENSG00000238009.2</td>
<td>-13477</td>
<td>45</td>
<td>45</td>
<td>0.0628492</td>
<td>6.50297e-10</td>
<td>0.858203</td>
<td>0.134436</td>
<td>8.26088e-05</td>
<td>6.50297e-10</td>
<td>8.52870e-05</td>
</tr>
<tr>
<td>5 _135203_G_A_b37</td>
<td>ENSG00000238009.2</td>
<td>5980</td>
<td>51</td>
<td>51</td>
<td>0.0698630</td>
<td>6.67194e-10</td>
<td>0.811790</td>
<td>0.127255</td>
<td>8.26088e-05</td>
<td>6.50297e-10</td>
<td>8.52870e-05</td>
</tr>
<tr>
<td>6 _988016_T_C_b37</td>
<td>ENSG00000268903.1</td>
<td>852121</td>
<td>21</td>
<td>23</td>
<td>0.0318560</td>
<td>6.35694e-05</td>
<td>0.501916</td>
<td>0.123743</td>
<td>8.52870e-05</td>
<td>6.50297e-10</td>
<td>8.52870e-05</td>
</tr>
</tbody>
</table>

seqinfo: 23 sequences from GRCh38 genome; no seqlengths
We have to extract the genomic locations of the SNPs from the IDs used by GTEx:

```r
locs <- strsplit(gtex_blood$variant_id, "_")
gtex_blood$chr <- sapply(locs, "[", 1)
gtex_blood$start <- sapply(locs, "[", 2)
gtex_blood$end <- sapply(locs, "[", 2)
tail(gtex_blood)
##                   variant_id           gene_id tss_distance ma_samples
## 1052537  X_154999134_G_A_b37 ENSG00000168939.6         1660        207
## 1052538 X_154999204_TA_T_b37 ENSG00000168939.6         1730        186
## 1052539  X_155004280_A_G_b37 ENSG00000168939.6         6806        186
## 1052540  X_155011926_T_C_b37 ENSG00000168939.6        14452        222
## 1052541  X_155014420_A_G_b37 ENSG00000168939.6        16946        215
## 1052542  X_155186978_G_C_b37 ENSG00000168939.6       189504        250
##         ma_count      maf pval_nominal     slope  slope_se
## 1052537      259 0.351902  3.19266e-05 -0.162062 0.0383749
## 1052538      274 0.390313  6.72752e-05 -0.157810 0.0390413
## 1052539      224 0.303523  1.91420e-08  0.230301 0.0398809
## 1052540      240 0.397096  3.89776e-05  0.157608 0.0377434
## 1052541      265 0.360054  4.17881e-05  0.159699 0.0380225
## 1052542      321 0.436141  1.24355e-04  0.145560 0.0374390
##         pval_nominal_threshold min_pval_nominal   pval_beta chr     start
## 1052537            0.000130368       1.9142e-08 2.75084e-05   X 154999134
## 1052538            0.000130368       1.9142e-08 2.75084e-05   X 154999204
## 1052539            0.000130368       1.9142e-08 2.75084e-05   X 155004280
## 1052540            0.000130368       1.9142e-08 2.75084e-05   X 155011926
## 1052541            0.000130368       1.9142e-08 2.75084e-05   X 155014420
## 1052542            0.000130368       1.9142e-08 2.75084e-05   X 155186978
##               end
## 1052537 154999134
## 1052538 154999204
## 1052539 155004280
## 1052540 155011926
## 1052541 155014420
## 1052542 155186978
```

We can then convert the data.frame into a GRanges object:

```r
gtex_blood <- makeGRangesFromDataFrame(gtex_blood, keep.extra.columns = TRUE)
gtex_blood
```

```r
# GRanges object with 1052542 ranges and 12 metadata columns:
# seqnames ranges strand | variant_id
# <Rle> <IRanges> <Rle> | <character>
# [1] 1 [231153, 231153] * | 1_231153_CTT_C_b37
# [3] 1 [ 64649,  64649] * | 1_64649_A_C_b37
# [4] 1 [115746, 115746] * | 1_115746_C_T_b37
# [1052538] X [154999204, 154999204] * | X_154999204_TA_T_b37
# [1052539] X [155004280, 155004280] * | X_155004280_A_G_b37
# [1052540] X [155011926, 155011926] * | X_155011926_T_C_b37
# [1052541] X [155014420, 155014420] * | X_155014420_A_G_b37
# [1052542] X [155186978, 155186978] * | X_155186978_G_C_b37
```

```
## gene_id tss_distance ma_samples ma_count maf
## <character> <integer> <integer> <integer> <numeric>
```
## ENSG00000223972.4  219284    13        13 0.0191740
## ENSG00000238009.2  -67303    18        20 0.0281690
## ENSG00000238009.2  -64574    16        16 0.0220386
## ENSG00000238009.2  -13477    45        45 0.0628492
## ENSG00000238009.2    5980    51        51 0.0698630
## ...                ...          ...        ...       ...
## ENSG00000168939.6    1730    219       274  0.390313
## ENSG00000168939.6    6806    186       224  0.303523
## ENSG00000168939.6   14452    222       279  0.379076
## ENSG00000168939.6   16946    215       265  0.360054
## ENSG00000168939.6  189504    250       321  0.436141

## pval_nominal  slope  slope_se  pval_nominal_threshold
##                <numeric> <numeric> <numeric>              <numeric>
## ...          ...        ...       ...                    ...
## ENSG00000168939.6  6.72752e-05 -0.157810 0.0390413            0.000130368
## ENSG00000168939.6  1.91420e-08  0.230301 0.0398809            0.000130368
## ENSG00000168939.6  3.88977e-05  0.157608 0.0377434            0.000130368
## ENSG00000168939.6  4.17781e-05  0.159699 0.0384025            0.000130368
## ENSG00000168939.6  1.24355e-04  0.145560 0.0374390            0.000130368

## min_pval_nominal  pval_beta
##                    <numeric>   <numeric>
## ...              ...         ...
## ENSG00000168939.6  1.9142e-08 2.75084e-05
## ENSG00000168939.6  1.9142e-08 2.75084e-05
## ENSG00000168939.6  1.9142e-08 2.75084e-05
## ENSG00000168939.6  1.9142e-08 2.75084e-05
## ...        

seqinfo: 23 sequences from an unspecified genome; no seqlengths

We also need to ensure that the chromosome notation is consistent with the previous objects:

```r
seqlevelsStyle(gtex_blood)
```

```r
## [1] "NCBI"    "Ensembl"
```

```r
seqlevels(gtx_blood)
```

```r
## [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14"
## [15] "15" "16" "17" "18" "19" "20" "21" "22" "X"
```

```r
seqlevelsStyle(gtx_blood) <- "UCSC"
```

```r
seqlevels(gtx_blood)
```

```r
## [1] "chr1" "chr2" "chr3" "chr4" "chr5" "chr6" "chr7" "chr8"
## [9] "chr9" "chr10" "chr11" "chr12" "chr13" "chr14" "chr15" "chr16"
## [17] "chr17" "chr18" "chr19" "chr20" "chr21" "chr22" "chrX"
```
From the publication\(^6\), we know the genomic coordinates are mapped to genome reference GRCh37, so we will have to uplift them to GRCh38 using \texttt{rtracklayer}\(^4\) and a mapping ("chain") file. The \texttt{R.utils} package is required to extract the gzipped file:

```r
library(rtracklayer)
library(R.utils)
# uncomment the following line to download file
download.file("http://hgdownload.cse.ucsc.edu/goldenPath/hg19/liftOver/hg19ToHg38.over.chain.gz", destfile = "hg19ToHg38.over.chain.gz")
# uncomment the following line to extract gzipped file
gunzip("hg19ToHg38.over.chain.gz")
ch <- import.chain("hg19ToHg38.over.chain")
gtex_blood <- unlist(liftOver(gtex_blood, ch))
```

We will use the GenomicRanges package\(^4\) to compute the overlap between GWAS SNPs and blood eQTLs:

```r
library(GenomicRanges)
hits <- findOverlaps(snps_hard, gtex_blood)
gtex_blood_with_snps_hard = gtex_blood[subjectHits(hits)]
mcols(snps_hard_in_gtex_blood) <- cbind(mcols(snps_hard_in_gtex_blood),
mcols(gtxe_blood_with_snps_hard))
snps_hard_in_gtex_blood <- as.data.frame(snps_hard_in_gtex_blood)
head(snps_hard_in_gtex_blood)
```

<table>
<thead>
<tr>
<th>seqnames</th>
<th>start</th>
<th>end</th>
<th>width</th>
<th>strand</th>
<th>SNPS</th>
<th>P.VALUE</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr11</td>
<td>589564</td>
<td>589564</td>
<td>1</td>
<td>+</td>
<td>rs4963128</td>
<td>3e-10</td>
<td>intron</td>
</tr>
<tr>
<td>chr3</td>
<td>58384450</td>
<td>58384450</td>
<td>1</td>
<td>+</td>
<td>rs6445975</td>
<td>7e-09</td>
<td>intron</td>
</tr>
<tr>
<td>chr8</td>
<td>11491677</td>
<td>11491677</td>
<td>1</td>
<td>*</td>
<td>rs13277113</td>
<td>1e-10</td>
<td>intergenic</td>
</tr>
<tr>
<td>chr8</td>
<td>11491677</td>
<td>11491677</td>
<td>1</td>
<td>*</td>
<td>rs13277113</td>
<td>1e-10</td>
<td>intergenic</td>
</tr>
<tr>
<td>chr8</td>
<td>11491677</td>
<td>11491677</td>
<td>1</td>
<td>*</td>
<td>rs13277113</td>
<td>1e-10</td>
<td>intergenic</td>
</tr>
<tr>
<td>chr8</td>
<td>11491677</td>
<td>11491677</td>
<td>1</td>
<td>*</td>
<td>rs13277113</td>
<td>1e-10</td>
<td>intergenic</td>
</tr>
</tbody>
</table>

So, we have 59 blood eQTL variants that are associated with SLE. We can now check whether any of the genes differentially expressed in SLE is an \textit{eGene}, a gene whose expression is influenced by an eQTL. We note that gene IDs in GTEx are mapped to GENCODE v19\(^4\), while we are using the newer v25 for the DEGs. To match the gene IDs in
the two objects, we will simply strip the last bit containing the GENCODE gene version, which effectively gives us Ensembl gene IDs:

```r
```

```r
snps_hard_in_gtex_blood_in_degs <-. merge(snps_hard_in_gtex_blood, degs, by = "ensembl_id", all = FALSE)
```

```
## DataFrame with 6 rows and 30 columns
##            ensembl_id seqnames  start       end       width   strand
##       <character> <factor> <integer> <integer> <integer> <factor>
## 1 ENSG00000130513    chr19  18370523  18370523         1        *
## 2 ENSG00000140497    chr15  75018695  75018695         1        +
## 3 ENSG00000172890    chr11  71476633  71476633         1        +
## 4 ENSG00000214894     chr6  31668965  31668965         1        +
## 5 ENSG00000214894     chr6  30973212  30973212         1        *
## 6 ENSG00000140497     chr6  31753256  31753256         1        +
##          SNPS   P.VALUE   LOCATION          variant_id          gene_id.x
##   <character> <numeric>   <factor>         <character>        <character>
## 1   rs8105429     5e-06 intergenic 19_18481333_A_G_b37  ENSG00000130513.6
## 2   rs2289583     6e-15     intron 15_75311036_C_A_b37 ENSG00000140497.12
## 3   rs3794060     1e-20     intron 11_71187679_C_T_b37  ENSG00000172890.7
## 4   rs9267531     8e-08     intron  6_31636742_A_G_b37  ENSG00000214894.2
## 5 rs114090659     6e-92 intergenic  6_30940989_T_C_b37  ENSG00000214894.2
## 6   rs3131379     2e-52     intron  6_31721033_G_A_b37  ENSG00000214894.2
##    tss_distance ma_samples  ma_count       maf pval_nominal     slope
##      <integer>  <integer> <integer> <numeric>    <numeric> <numeric>
## 1        -4208        166       189 0.2560980  7.87256e-11  0.350964
## 2       145330        170       191 0.2588080  7.57250e-06 -0.107460
## 3        23524        183       231 0.3130080  1.91380e-31  0.407266
## 4       838306         49        54 0.0731707  3.36144e-08  0.479659
## 5       142553         83        91 0.1233060  7.00411e-11  0.453255
## 6       922597         50        55 0.0745257  2.69451e-08  0.479935
##            gene_id.y bp_length    symbol   baseMean log2FoldChange
##          <character> <integer>    <list>  <numeric>      <numeric>
## 1  ENSG00000130513.6      2087     GDF15    6.75448      0.7883703
## 2 ENSG00000140497.16      5000    SCAMP2 3483.03109     -0.2959934
## 3 ENSG00000172890.11     16263   NADSYN1 4020.56224      0.2619770
## 4  ENSG000000214894.6      2171 LINC00243   74.95034      1.2684089
## 5  ENSG00000214894.6      2171 LINC00243   74.95034      1.2684089
## 6  ENSG00000214894.6      2171 LINC00243   74.95034      1.2684089
##        lfcSE      stat       pvalue         padj
##    <numeric> <numeric>    <numeric>    <numeric>
## 1 0.28347645  2.781079 5.417861e-03 0.0448154406
## 2 0.08814542 -3.358012 7.850510e-04 0.0119267855
## 3 0.08976429  2.918499 3.517209e-03 0.333810138
## 4 0.27106143  4.679415 2.876950e-06 0.0002424263
## 5 0.27106143  4.679415 2.876950e-06 0.0002424263
## 6 0.27106143  4.679415 2.876950e-06 0.0002424263
We can add these 4 genes to our list:

```r
prioritised_hits <- unique(rbind(prioritised_hits, data.frame(
snp_id = snps_hard_in_gtex_blood_in_degs$SNPS,
snp_pvalue = snps_hard_in_gtex_blood_in_degs$P.VALUE,
gene_id = snps_hard_in_gtex_blood_in_degs$gene_id.y,
gene_symbol = snps_hard_in_gtex_blood_in_degs$symbol,
gene_pvalue = snps_hard_in_gtex_blood_in_degs$padj,
gene_log2foldchange = snps_hard_in_gtex_blood_in_degs$log2FoldChange)))
```

```
##                      snp_id snp_pvalue snp_location            gene_id
## ENSG00000096968   rs1887428      1e-06      fiveUTR ENSG00000096968.13
## ENSG00000099834  rs58688157      5e-13     promoter ENSG00000099834.18
## ENSG00000115267   rs1990760      4e-08       coding ENSG00000115267.5
## ENSG00000120280   rs887369       5e-10       coding ENSG00000120280.5
## ENSG00000185507   rs1061502      9e-11       coding ENSG00000185507.19
```

**FANTOM5 data.** The FANTOM consortium profiled gene expression across a large panel of tissues and cell types using CAGE[19,21]. This technology allows mapping of transcription start sites (TSSs) and enhancer RNAs (eRNAs) genome-wide. Correlations between these promoter and enhancer elements across a large panel of tissues and cell types can then be calculated to identify significant promoter - enhancer pairs. In turn, we will use these correlations to map distal regulatory SNPs to target genes.

We can read in and have a look at the enhancer - promoter correlation data in this way:

```r
# uncomment the following line to download the file
download.file("http://enhancer.binf.ku.dk/presets/enhancer_tss_associations.bed", destfile = "enhancer_tss_associations.bed")
fantom <- read.delim("enhancer_tss_associations.bed", skip = 1,
stringsAsFactors = FALSE)
head(fantom)
```

```
##   X.chrom chromStart chromEnd
## 1    chr1     858252   861621
## 2    chr1     894178   956888
## 3    chr1     901376   956888
## 4    chr1     926753   989456
## 5    chr2     91409065 91409065
## 6    chr2     91409065 91409065
```
## 4    chr1     901376  1173762
## 5    chr1     935051   942164
## 6    chr1     935051  1005621
##                                                                        name
## 1                                 chr1:858256-858648;NM_152486;SAMD11;R:0.404;FDR:0
## 2               chr1:956563-956812;NM_015658;NOC2L;R:0.202;FDR:8.01154668254404e-08
## 3                   chr1:956563-956812;NM_001160184,NM_032129;PLEKHN1;R:0.422;FDR:0
## 4                 chr1:1173386-1173736;NM_001160184,NM_032129;PLEKHN1;R:0.311;FDR:0
## 5   chr1:941791-942135;NM_001142467,NM_021170;HES4;R:0.187;FDR:6.3294988809368e-07
## 6 chr1:1005293-1005547;NM_001142467,NM_021170;HES4;R:0.236;FDR:6.28221217150423e-11
##   score strand thickStart thickEnd itemRgb blockCount blockSizes
## 1   404      .     858452   858453   0,0,0          2   401,1001
## 2   202      .     956687   956688   0,0,0          2   1001,401
## 3   422      .     956687   956688   0,0,0          2   1001,401
## 4   311      .    1173561  1173562   0,0,0          2   1001,401
## 5   187      .     941963   941964   0,0,0          2   1001,401
## 6   236      .    1005420  1005421   0,0,0          2   1001,401
##   chromStarts
## 1      0,2368
## 2     0,62309
## 3     0,55111
## 4    0,271985
## 5      0,6712
## 6     0,70169

Everything we need is in the fourth column, **name**: genomic location of the enhancer, gene identifiers, Pearson correlation coefficient and significance. We will use the `splitstackshape` package to parse it:

```r
library(splitstackshape)
fantom <- as.data.frame(cSplit(fantom, splitCols = "name", sep = ";", direction = "wide"))
head(fantom)
```

```r
##   X.chrom chromStart chromEnd score strand thickStart thickEnd itemRgb blockCount blockSizes chromStarts               name_1
## 1    chr1     858252   861621   404      .     858452   858453   0,0,0          2   401,1001 chr1:858256-858648
## 2    chr1     894178   956888   202      .     956687   956688   0,0,0          2   1001,401 chr1:956563-956812
## 3    chr1     901376   956888   422      .     956687   956688   0,0,0          2   1001,401 chr1:956563-956812
## 4    chr1     901376  1173762   311      .    1173561  1173562   0,0,0          2   1001,401 chr1:956563-956812
## 5    chr1     935051   942164   187      .     941963   941964   0,0,0          2   1001,401 chr1:956563-956812
## 6    chr1     935051  1005621   236      .    1005420  1005421   0,0,0          2   1001,401 chr1:956563-956812
##                   name_2  name_3  name_4                   name_5
## 1              NM_152486  SAMD11 R:0.404                    FDR:0
## 2              NM_015658   NOC2L R:0.202 FDR:8.01154668254404e-08
## 3 NM_001160184,NM_032129 PLEKHN1 R:0.422                    FDR:0
## 4 NM_001160184,NM_032129 PLEKHN1 R:0.311;FDR:0
## 5   chr1:941791-942135;NM_001142467,NM_021170;HES4;R:0.187;FDR:6.3294988809368e-07
## 6 NM_001142467,NM_021170;HES4;R:0.236;FDR:6.28221217150423e-11
```
Now we can extract the genomic locations of the enhancers and the correlation values:

```r
locs <- strsplit(as.character(fantom$name_1), "[-]")
fantom$chr <- sapply(locs, "[", 1)
fantom$start <- as.numeric(sapply(locs, "[", 2))
fantom$end <- as.numeric(sapply(locs, "[", 3))
fantom$symbol <- fantom$name_3
fantom$corr <- sub("R:", "", fantom$name_4)
fantom$fdr <- sub("FDR:", "", fantom$name_5)
head(fantom)
```

```
##   X.chrom chromStart chromEnd score strand thickStart thickEnd itemRgb
## 1    chr1     858252   861621   404      .     858452   858453   0,0,0
## 2    chr1     894178   956888   202      .     956687   956688   0,0,0
## 3    chr1     901376   956888   422      .     956687   956688   0,0,0
## 4    chr1     901376  1173762   311      .    1173561  1173562   0,0,0
## 5    chr1     935051   942164   187      .     941963   941964   0,0,0
## 6    chr1     935051  1005621   236      .    1005420  1005421   0,0,0
##                   name_2  name_3  name_4                   name_5  chr
## 1              NM_152486  SAMD11 R:0.404                    FDR:0 chr1
## 2              NM_015658   NOC2L R:0.202 FDR:8.011546668254404e-08 chr1
## 3 NM_001160184,NM_032129 PLEKHN1 R:0.422                    FDR:0 chr1
## 4 NM_001160184,NM_032129 PLEKHN1 R:0.311                    FDR:0 chr1
## 5 NM_001142467,NM_021170    HES4 R:0.187 FDR:6.32949888009368e-07 chr1
## 6 NM_001142467,NM_021170    HES4 R:0.236 FDR:6.28221217150423e-11 chr1
##     start     end  symbol  corr                  fdr
## 1  858256  858648  SAMD11 0.404                    0
## 2  956563  956812   NOC2L 0.202 8.011546668254404e-08
## 3  956563  956812 PLEKHN1 0.422                    0
## 4 1173386 1173736 PLEKHN1 0.311                    0
## 5  941963  942164    HES4 0.187 6.32949888009368e-07
## 6 1005420 1005542    HES4 0.236 6.28221217150423e-11
```

We can select only the enhancer - promoter pairs with a decent level of correlation and significance and tidy the data at the same time:

```r
fantom <- unique(subset(fantom, subset = corr >= 0.25 & fdr < 1e-5, select = c("chr", "start", "end", "symbol")))
head(fantom)
```

```
##     chr   start     end  symbol
## 1  chr1  858256  858648  SAMD11
## 3  chr1  956563  956812 PLEKHN1
## 4  chr1 1173386 1173736 PLEKHN1
## 13 chr1 1136075 1136463   ISG15
## 14 chr1  956563  956812    AGRN
## 27 chr1 1060905 1061095  RNF223
```
Now we would like to check whether any of our candidate regulatory SNPs are falling in any of these enhancers. To do this, we have to convert the `data.frame` into a `GRanges` object:

```r
fantom <- makeGRangesFromDataFrame(fantom, keep.extra.columns = TRUE)
```

```
## GRanges object with 33957 ranges and 1 metadata column:
##         seqnames                 ranges strand |   symbol
##            <Rle>              <IRanges>  <Rle> | <factor>
##       1     chr1     [ 858256,  858648]      * |   SAMD11
##       3     chr1     [ 956563,  956812]      * |  PLEKHN1
##       4     chr1    [1173386, 1173736]      * |  PLEKHN1
##      13     chr1    [1136075, 1136463]      * |    ISG15
##      14     chr1    [ 956563,  956812]      * |     AGRN
## ... ...                    ...    ... .      ...
##   66929     chrX [154256125, 154256514]      * |     F8A2
##   66932     chrY [ 2871660,  2871926]      * |     ZFY
##   66933     chrY [ 2872046,  2872325]      * |     ZFY
##   66940     chrY [ 21664138, 21664302]      * |    KDM5D
##   66941     chrY [ 22735456, 22735677]      * |   EIF1AY
## -------
## seqinfo: 24 sequences from an unspecified genome; no seqlengths
```

Similar to the GTEx data, the FANTOM5 data is also mapped to GRCh37\(^1\), so we will have to uplift the GRCh37 coordinates to GRCh38:

```r
fantom <- unlist(liftOver(fantom, ch))
```

```
## GRanges object with 34160 ranges and 1 metadata column:
##         seqnames                 ranges strand |   symbol
##            <Rle>              <IRanges>  <Rle> | <factor>
##       1     chr1     [ 922876,  923268]      * |   SAMD11
##       3     chr1     [1021183, 1021432]      * |  PLEKHN1
##       4     chr1    [1238006, 1238356]      * |  PLEKHN1
##      13     chr1    [1200695, 1201083]      * |    ISG15
##      14     chr1    [1021183, 1021432]      * |     AGRN
## ... ...                    ...    ... .      ...
##   66929     chrX [155027850, 155028239]      * |     F8A2
##   66932     chrY [ 3003619,  3003885]      * |     ZFY
##   66933     chrY [ 3004005,  3004284]      * |     ZFY
##   66940     chrY [ 19502252, 19502416]      * |    KDM5D
##   66941     chrY [ 20573570, 20573791]      * |   EIF1AY
## -------
## seqinfo: 24 sequences from an unspecified genome; no seqlengths
```

We can now compute the overlap between SNPs and enhancers:

```r
hits <- findOverlaps(snps_hard, fantom)
snps_hard_in_fantom = snps_hard[queryHits(hits)]
fantom_with_snps_hard = fantom[subjectHits(hits)]
mcols(snps_hard_in_fantom) <- cbind(mcols(snps_hard_in_fantom),
mcols(fantom_with_snps_hard))
snps_hard_in_fantom <- as.data.frame(snps_hard_in_fantom)
```

```
##    seqnames     start       end width strand       SNPS P.VALUE   LOCATION
## 1      chr2 191099907 191099907     1      -  rs7574865   9e-14     intron
## 2      chr2 191099907 191099907     1      -  rs7574865   9e-14     intron
```
We note that some of the SNPs are assigned to more than one gene. This is because enhancers are promiscuous and can regulate multiple genes.

We can now check if any of these genes is differentially expressed in our RNA-seq data:

```r
snps_hard_in_fantom_in_degs <- merge(snps_hard_in_fantom, degs, by = "symbol", all = FALSE)
```

```r
snps_hard_in_fantom_in_degs
```

We can now check if any of these genes is differentially expressed in our RNA-seq data:
We have identified 2 genes whose putative enhancers contain SLE GWAS SNPs. Let’s add these to our list:

```r
prioritised_hits <- unique(rbind(prioritised_hits, data.frame(
  snp_id = snps_hard_in_fantom_in_degs$SNPS,
  snp_pvalue = snps_hard_in_fantom_in_degs$P.VALUE,
  snp_location = snps_hard_in_fantom_in_degs$LOCATION,
  gene_id = snps_hard_in_fantom_in_degs$gene_id,
  gene_symbol = snps_hard_in_fantom_in_degs$symbol,
  gene_pvalue = snps_hard_in_fantom_in_degs$padj,
  gene_log2foldchange = snps_hard_in_fantom_in_degs$log2FoldChange)))
```

```
##                      snp_id snp_pvalue snp_location            gene_id
## ENSG00000096968   rs1887428      1e-06      fiveUTR ENSG00000096968.13
## ENSG00000099834  rs58688157      5e-13     promoter ENSG00000099834.18
## ENSG00000115267   rs1990760      4e-08       coding ENSG00000115267.5
## ENSG00000120280    rs887369      5e-10       coding ENSG00000120280.5
## ENSG00000185507   rs1061502      9e-11       coding ENSG00000185507.19
## ENSG00000204366    rs558702      8e-21     promoter  ENSG00000204366.3
## ENSG00000275106  rs10488631      2e-11     promoter  ENSG00000275106.1
## 1                 rs8105429      5e-06   intergenic  ENSG00000130513.6
## 2                 rs2289583      6e-15       intron ENSG00000140497.16
## 3                 rs3794060      1e-20       intron ENSG00000172890.11
## 4                 rs9267531      8e-08       intron ENSG00000214894.6
## 5               rs114090659      6e-92   intergenic  ENSG00000214894.6
## 6                 rs3131794     2e-52       intron  ENSG00000214894.6
## 11                rs3129716     4e-09   intergenic ENSG00000204252.13
## 21               rs11185603     4e-07   intergenic ENSG00000185811.16
## gene_symbol  gene_pvalue gene_log2foldchange
## ENSG00000096968        JAK2 2.068794e-02           0.4854343
## ENSG00000099834       CDHR5 1.732902e-02           0.8539586
## ENSG00000115267       IFIH1 1.120363e-03           1.1494945
## ENSG00000120280      CXorf21 6.047898e-05           0.7819504
## ENSG00000185507        IRF7 2.298336e-02          -0.3892298
## ENSG00000204366         NA 1.797861e-02           0.7344844
## 1                     GDF15 4.481444e-02          0.7883703
## 2                    SCAMP2 1.192679e-02          -0.2959934
## 3                   NADSYN1 3.338101e-02          0.2619770
## 4                LINC00243 2.442643e-04          1.2684089
## 5                LINC00243 2.442643e-04          1.2684089
## 6                LINC00243 2.442643e-04          1.2684089
## 11               HLA-DOA 4.431304e-02          -0.4424595
## 21                IKZF1 1.162554e-02          -0.2575717
```

**Promoter Capture Hi-C data.** More recently, chromatin interaction data was generated across 17 human primary blood cell types. More than 30,000 promoter baits were used to capture promoter-interacting regions genome-wide. These regions were then mapped to enhancers based on the Ensembl Regulatory Build and can be accessed in the supplementary data of the paper:

```
# uncomment the following line to download file
# uncomment the following lines to extract zipped files
#unzip("mmc4.zip")
#unzip("DATA_SI.zip")
pchic <- read.delim("ActivePromoterEnhancerLinks.tsv", stringsAsFactors = FALSE)
head(pchic)
```
In this case, we will have to map the promoter baits to genes first. We can do this by converting the baits to a `GRanges` object and then using the `TxDb` object we previously built to extract positions of transcription start sites (TSSs):

```r
# Convert baits to GRanges
baits <- GRanges(seqnames = pchic$baitChr, ranges = IRanges(start = pchic$baitSt, end = pchic$baitEnd))

# Extract positions of TSSs
promoters <- promoters(txdb, upstream = 0, downstream = 1, columns = "gene_id")

# Find nearest TSSs to baits
hits <- nearest(baits, tss)

# Map baits to genes
baits$gene_id <- unlist(tss[hits]$gene_id)
```

```
## GRanges object with 51142 ranges and 1 metadata column:
## seqnames ranges strand | gene_id
## <Rle> <IRanges> <Rle> | <character>
## [1] chr1 [1206873, 1212438] * | ENSG00000186827.10
## [2] chr1 [1206873, 1212438] * | ENSG00000186827.10
## [3] chr1 [1206873, 1212438] * | ENSG00000186827.10
## [4] chr1 [1206873, 1212438] * | ENSG00000186827.10
## [5] chr1 [1206873, 1212438] * | ENSG00000186827.10
## ... ... ... ... ... ... ...
## [51138] chrY [22732049, 22743996] * | ENSG000000230727.1
## [51139] chrY [22732049, 22743996] * | ENSG000000230727.1
## [51140] chrY [22732049, 22743996] * | ENSG000000230727.1
## [51141] chrY [22732049, 22743996] * | ENSG000000230727.1
## [51142] chrY [22732049, 22743996] * | ENSG000000230727.1
## -------
## seqinfo: 24 sequences from an unspecified genome; no seqlengths
```
Now we can create a `GRanges` object of the enhancers in the promoter capture Hi-C data with the bait annotation attached:

```r
pchic <- GRanges(seqnames = pchic$oeChr, ranges = IRanges(start = pchic$oeSt, end = pchic$oeEnd), gene_id = baits$gene_id)
pchic <- unique(pchic)
```

## GRanges object with 25232 ranges and 1 metadata column:
```
#  seqnames ranges  strand gene_id
# <Rle>  <IRanges> <Rle> |        <character>
# [1] chr1 [ 943676,  957199] * | ENSG00000186827.10
# [2] chr1 [1034268, 1040208] * | ENSG00000186827.10
# [3] chr1 [1040208, 1043143] * | ENSG00000186827.10
# [4] chr1 [1069045, 1083958] * | ENSG00000186827.10
# [5] chr1 [1083958, 1091234] * | ENSG00000186827.10
# ...                                              ...
# [25228] chrY [23401616, 23404873] * | ENSG00000230727.1
# [25229] chrY [23404938, 23407193] * | ENSG00000230727.1
# [25230] chrY [23409014, 23410287] * | ENSG00000230727.1
# [25231] chrY [23410287, 23411837] * | ENSG00000230727.1
# [25232] chrY [23411837, 23412539] * | ENSG00000230727.1
```
```
## seqinfo: 24 sequences from an unspecified genome; no seqlengths
```

Next, we basically repeat the steps we have taken when working with the FANTOM5 data to find SLE GWAS SNPs overlapping with these enhancers:

```r
hits <- findOverlaps(snps_hard, pchic)

snps_hard_in_pchic = snps_hard[queryHits(hits)]
pchic_with_snps_hard = pchic[subjectHits(hits)]
mcols(snps_hard_in_pchic) <- cbind(mcols(snps_hard_in_pchic), mcols(pchic_with_snps_hard))

snps_hard_in_pchic <- as.data.frame(snps_hard_in_pchic)
```

```r
#  seqnames start  end  width strand gene_id
# 1  chr6   31753256  31753256   1      +   rs3131379   ENSG00000219797.2
# 2  chr6   32696681  32696681   1      *  rs2647012   ENSG00000204290.10
# 3  chr16  30631546  30631546   1      *  rs7197475   ENSG00000212536.1
# 4  chr20  4762059   4762059   1      *  rs6084875   ENSG00000219797.2
# 5  chr6   32689659  32689659   1      *  rs3129716   ENSG00000204290.10
# 6  chr6   31668965  31668965   1      +  rs9267531   ENSG00000219797.2
# 7  chr6   31951083  31951083   1      +  rs1270942   ENSG00000204290.10
# 8  chr6  106140931 106140931   1      -  rs6568431   ENSG00000212536.1
# 9  chr7   28146272  28146272   1      -  rs849142   ENSG00000219797.2
#10  chr2   65381229  65381229   1      -  rs268134   ENSG00000204290.10
#11  chr17  39850937  39850937   1      -  rs14312327   ENSG00000204290.10
#12  chr9   86916761  86916761   1      *  rs190029011 ENSG00000204290.10
#13  chr11  65637829  65637829   1      *  rs931127   ENSG00000204290.10
#14  chr19  18370523  18370523   1      *  rs8105429  ENSG00000204290.10
#15  chr16  85977731  85977731   1      *  rs10521318 ENSG00000204290.10
#16  chr5   39406395  39406395   1      -  rs3914167   ENSG00000204290.10
#17  chr16  31315385  31315385   1      +  rs11860650 ENSG00000204290.10
```
We check if any of these enhancers containing SLE variants are known to putatively regulate genes differentially expressed in SLE:

```r
snps_hard_in_pchic_in_degs <- merge(snps_hard_in_pchic, degs, by = "gene_id", all = FALSE)
```

```r

snps_hard_in_pchic_in_degs[, ecols] <- lapply(snps_hard_in_pchic_in_degs[, ecols], as.numeric)
```

```r
```

And finally we add these 3 genes to our list. These are our final results:
## SNP Annotation

<table>
<thead>
<tr>
<th>snp_id</th>
<th>snp_pvalue</th>
<th>snp_location</th>
<th>gene_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG00000096968</td>
<td>rs1887428</td>
<td>1e-06</td>
<td>fiveUTR ENSG00000096968.13</td>
</tr>
<tr>
<td>ENSG00000099834</td>
<td>rs58688157</td>
<td>5e-13</td>
<td>promoter ENSG00000099834.18</td>
</tr>
<tr>
<td>ENSG00000115267</td>
<td>rs1990760</td>
<td>4e-08</td>
<td>coding ENSG00000115267.5</td>
</tr>
<tr>
<td>ENSG00000120280</td>
<td>rs887369</td>
<td>5e-10</td>
<td>coding ENSG00000120280.5</td>
</tr>
<tr>
<td>ENSG00000185507</td>
<td>rs1061502</td>
<td>9e-11</td>
<td>coding ENSG00000185507.19</td>
</tr>
<tr>
<td>ENSG00000204366</td>
<td>rs558702</td>
<td>8e-21</td>
<td>promoter ENSG00000204366.3</td>
</tr>
<tr>
<td>ENSG00000275106</td>
<td>ra10488631</td>
<td>2e-11</td>
<td>promoter ENSG00000275106.1</td>
</tr>
<tr>
<td>1</td>
<td>rs8105429</td>
<td>5e-06</td>
<td>intergenic ENSG000000130513.6</td>
</tr>
<tr>
<td>2</td>
<td>rs2289583</td>
<td>6e-15</td>
<td>intron ENSG000000140497.16</td>
</tr>
<tr>
<td>3</td>
<td>rs3794060</td>
<td>1e-20</td>
<td>intron ENSG000000172890.11</td>
</tr>
<tr>
<td>4</td>
<td>rs9267531</td>
<td>8e-08</td>
<td>intron ENSG000000214894.6</td>
</tr>
<tr>
<td>5</td>
<td>rs114090659</td>
<td>6e-92</td>
<td>intergenic ENSG000000214894.6</td>
</tr>
<tr>
<td>6</td>
<td>rs3131379</td>
<td>2e-52</td>
<td>interon ENSG000000214894.6</td>
</tr>
<tr>
<td>11</td>
<td>rs3129716</td>
<td>4e-09</td>
<td>intergenic ENSG000000204252.13</td>
</tr>
<tr>
<td>21</td>
<td>rs1185603</td>
<td>4e-07</td>
<td>intergenic ENSG000000185811.16</td>
</tr>
<tr>
<td>12</td>
<td>ra849142</td>
<td>9e-11</td>
<td>intron ENSG000000106052.13</td>
</tr>
<tr>
<td>22</td>
<td>rs3131379</td>
<td>2e-52</td>
<td>intron ENSG000000219797.2</td>
</tr>
<tr>
<td>31</td>
<td>rs9267531</td>
<td>8e-08</td>
<td>intron ENSG000000219797.2</td>
</tr>
<tr>
<td>41</td>
<td>ra931127</td>
<td>7e-06</td>
<td>intergenic ENSG000000245532.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>gene_symbol</th>
<th>gene_pvalue</th>
<th>gene_log2foldchange</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAK2</td>
<td>2.068794e-02</td>
<td>0.4854343</td>
</tr>
<tr>
<td>CDHR5</td>
<td>1.732902e-02</td>
<td>0.8539586</td>
</tr>
<tr>
<td>IFIHI</td>
<td>1.120363e-03</td>
<td>1.1494945</td>
</tr>
<tr>
<td>CXorf21</td>
<td>6.047898e-05</td>
<td>0.7819504</td>
</tr>
<tr>
<td>IRF7</td>
<td>2.298336e-04</td>
<td>1.4062704</td>
</tr>
<tr>
<td>ZBTB12</td>
<td>3.584479e-02</td>
<td>-0.3892298</td>
</tr>
<tr>
<td>GDF15</td>
<td>4.481544e-02</td>
<td>0.7883703</td>
</tr>
<tr>
<td>SCAMP2</td>
<td>1.192679e-02</td>
<td>-0.2959934</td>
</tr>
<tr>
<td>NADSYN1</td>
<td>3.338101e-02</td>
<td>0.2619770</td>
</tr>
<tr>
<td>LINC00243</td>
<td>2.442643e-04</td>
<td>1.2684089</td>
</tr>
<tr>
<td>LINC00243</td>
<td>2.442643e-04</td>
<td>1.2684089</td>
</tr>
<tr>
<td>LINC00243</td>
<td>2.442643e-04</td>
<td>1.2684089</td>
</tr>
<tr>
<td>HLA-DOA</td>
<td>4.431304e-02</td>
<td>-0.4424595</td>
</tr>
<tr>
<td>IKZF1</td>
<td>1.162554e-02</td>
<td>-0.2575717</td>
</tr>
<tr>
<td>TAX1BP1</td>
<td>3.866955e-02</td>
<td>0.3438396</td>
</tr>
<tr>
<td>NA</td>
<td>7.981690e-05</td>
<td>0.5586633</td>
</tr>
<tr>
<td>NA</td>
<td>7.981690e-05</td>
<td>0.5586633</td>
</tr>
<tr>
<td>NEAT1, M....</td>
<td>3.221355e-03</td>
<td>0.5259525</td>
</tr>
</tbody>
</table>

### Conclusions

In this Bioconductor workflow we have used several packages and datasets to demonstrate how regulatory genomic data can be used to annotate significant hits from GWASs and provide an intermediate layer connecting genetics and transcriptomics. Overall, we identified 17 SLE-associated SNPs that we mapped to 16 genes differentially expressed in SLE, using eQTL data and enhancer-promoter relationships from CAGE and promoter capture Hi-C experiments.

While simplified, the workflow also demonstrates some real-world challenges encountered when working with genomic data from different sources, such as the use of different genome references and gene annotation conventions, the parsing of files with custom formats into Bioconductor-compatible objects and the mapping of genomic locations to genes.

As the sample size and power of GWASs and gene expression studies continue to increase, it will become more and more challenging to identify truly significant hits and interpret them. The use of regulatory genomics data as presented here can be an important skill and tool to gain insights into large biomedical datasets and help in the identification of biomarkers and therapeutic targets.
Data and software availability
Download links for all datasets are part of the workflow. Software packages required to reproduce the analysis can be installed as part of the workflow. Code is available at https://github.com/enricoferrero/bioconductor-regulatory-genomics-workflow.

Archived code as at time of publication: http://doi.org/10.5281/zenodo.1154235

License: CC-BY 4.0

Competing interests
EF is a full time employee of GSK.

Grant information
The author(s) declared that no grants were involved in supporting this work.

References

PubMed Abstract | Publisher Full Text


PubMed Abstract | Publisher Full Text | Free Full Text

PubMed Abstract | Publisher Full Text

PubMed Abstract | Publisher Full Text

PubMed Abstract | Publisher Full Text | Free Full Text

PubMed Abstract | Publisher Full Text | Free Full Text

PubMed Abstract | Publisher Full Text | Free Full Text

PubMed Abstract | Publisher Full Text | Free Full Text

PubMed Abstract | Publisher Full Text | Free Full Text

PubMed Abstract | Publisher Full Text | Free Full Text


PubMed Abstract | Publisher Full Text | Free Full Text

PubMed Abstract | Publisher Full Text | Free Full Text


Publisher Full Text

PubMed Abstract | Publisher Full Text | Free Full Text

PubMed Abstract | Publisher Full Text | Free Full Text


PubMed Abstract | Publisher Full Text | Free Full Text

PubMed Abstract | Publisher Full Text | Free Full Text

PubMed Abstract | Publisher Full Text | Free Full Text

PubMed Abstract | Publisher Full Text | Free Full Text

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com