METHOD ARTICLE

Using regulatory genomics data to interpret the function of disease variants and prioritise genes from expression studies [version 1; referees: 2 approved with reservations]

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.

Corresponding author: Enrico Ferrero (enrico.x.ferrero@gsk.com)

Author roles: Ferrero E: Conceptualization, Formal Analysis, Investigation, Methodology, Project Administration, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: EF is a full time employee of GSK.

How to cite this article: Ferrero E. Using regulatory genomics data to interpret the function of disease variants and prioritise genes from expression studies [version 1; referees: 2 approved with reservations] F1000Research 2018, 7:121 (doi: 10.12688/f1000research.13577.1)

Copyright: © 2018 Ferrero E. This is an open access article distributed under the terms of the Creative Commons Attribution Licence, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

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

Abbreviations
CAGE: cap analysis of gene expression; DHS: DNase I hypersensitive site; eQTL: expression quantitative trait locus; GWAS: genome-wide association study; PheWAS: phenome-wide association study; SLE: systemic lupus erythematosus; SNP: single nucleotide polymorphism; TSS: transcription start site

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 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\textsuperscript{14}, FANTOM5 correlations between promoters and enhancers\textsuperscript{21} and promoter capture Hi-C data\textsuperscript{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:

\begin{verbatim}
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"))
\end{verbatim}

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\textsuperscript{28}.

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

\begin{verbatim}
library(recount)
# uncomment the following line to download dataset
download_study("SRP062966")
load(file.path("SRP062966", "rse_gene.RData"))
rse <- scale_counts(rse_gene)
\end{verbatim}

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

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

\begin{verbatim}
assay(rse)[1:10, 1:10]
\end{verbatim}
We note that genes are annotated using the GENCODE\textsuperscript{32} 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:

```r
colData(rse)
## DataFrame with 117 rows and 21 columns
##           project  sample  experiment           run
##              <character> <character> <character> <character>
## SRR2443263   SRP062966  SRS1048033  SRX1168388  SRR2443263
## SRR2443262   SRP062966  SRS1048034  SRX1168387  SRR2443262
## SRR2443261   SRP062966  SRS1048035  SRX1168386  SRR2443261
## SRR2443260   SRP062966  SRS1048036  SRX1168385  SRR2443260
## SRR2443259   SRP062966  SRS1048037  SRX1168384  SRR2443259
## ...                                     ...         ...
## SRR2443150   SRP062966  SRS1048146  SRX1168275  SRR2443150
## SRR2443149   SRP062966  SRS1048147  SRX1168274  SRR2443149
## read_count_as_reported_by_sra  reads_downloaded
##                                              <integer>        <integer>
## SRR2443263                                103977424        103977424
## SRR2443262                                125900891        125900891
## SRR2443261                                129803063        129803063
## SRR2443260                                105335395        105335395
## SRR2443259                                101692332        101692332
## ...                                     ...              ...
## SRR2443151                                87315854         87315854
## SRR2443150                                96825506         96825506
## SRR2443148                               121365435        121365435
## SRR2443147                               104038425        104038425
## 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
```
<table>
<thead>
<tr>
<th>sra_misreported_paired_end</th>
<th>mapped_read_count</th>
<th>auc</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE</td>
<td>103499268</td>
<td>5149333280</td>
</tr>
<tr>
<td>FALSE</td>
<td>125499809</td>
<td>6244059473</td>
</tr>
<tr>
<td>FALSE</td>
<td>125043355</td>
<td>6201504759</td>
</tr>
<tr>
<td>FALSE</td>
<td>104872856</td>
<td>5211910530</td>
</tr>
<tr>
<td>FALSE</td>
<td>101258496</td>
<td>5033612693</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>FALSE</td>
<td>86874384</td>
<td>4319264868</td>
</tr>
<tr>
<td>FALSE</td>
<td>96316303</td>
<td>4787601223</td>
</tr>
<tr>
<td>FALSE</td>
<td>120819733</td>
<td>6009515064</td>
</tr>
<tr>
<td>FALSE</td>
<td>103588909</td>
<td>5153702232</td>
</tr>
<tr>
<td>FALSE</td>
<td>112640054</td>
<td>5598306153</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sharq_beta_tissue</th>
<th>sharq_beta_cell_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>biosample_submission_date</th>
<th>biosample_publication_date</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-08-28T16:41:29.000</td>
<td>2015-09-16T01:24:17.350</td>
</tr>
<tr>
<td>2015-08-28T16:41:35.000</td>
<td>2015-09-16T01:24:13.450</td>
</tr>
<tr>
<td>2015-08-28T16:41:33.000</td>
<td>2015-09-16T01:24:12.433</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2015-08-28T16:42:24.000</td>
<td>2015-09-16T01:19:06.787</td>
</tr>
<tr>
<td>2015-08-28T16:42:23.000</td>
<td>2015-09-16T01:19:05.557</td>
</tr>
<tr>
<td>2015-08-28T16:42:22.000</td>
<td>2015-09-16T01:19:04.583</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bioasig_beach</th>
<th>avg_read_length</th>
<th>geo_accession</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRR2443263.bw</td>
<td>50</td>
<td>GSM1863749</td>
</tr>
<tr>
<td>SRR2443262.bw</td>
<td>50</td>
<td>GSM1863748</td>
</tr>
<tr>
<td>SRR2443261.bw</td>
<td>50</td>
<td>GSM1863747</td>
</tr>
<tr>
<td>SRR2443260.bw</td>
<td>50</td>
<td>GSM1863746</td>
</tr>
<tr>
<td>SRR2443259.bw</td>
<td>50</td>
<td>GSM1863745</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>title</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SRR2443263.bw</td>
<td>control18</td>
</tr>
<tr>
<td>SRR2443262.bw</td>
<td>control17</td>
</tr>
<tr>
<td>SRR2443261.bw</td>
<td>control16</td>
</tr>
<tr>
<td>SRR2443260.bw</td>
<td>control15</td>
</tr>
<tr>
<td>SRR2443259.bw</td>
<td>control14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bigwig_file</th>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRR2443263</td>
<td>SRR2443263.bw</td>
</tr>
<tr>
<td>SRR2443262</td>
<td>SRR2443262.bw</td>
</tr>
<tr>
<td>SRR2443261</td>
<td>SRR2443261.bw</td>
</tr>
<tr>
<td>SRR2443260</td>
<td>SRR2443260.bw</td>
</tr>
<tr>
<td>SRR2443259</td>
<td>SRR2443259.bw</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
The most interesting part of the metadata is contained in the `characteristics` column, which is a CharacterList object:

```r
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)$disease_status <- factor(colData(rse)$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")]
```

```
# DataFrame with 117 rows and 4 columns
#    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**:

```r
library(DESeq2)
dds <- DESeqDataSet(rse, ~ disease_status)
dds <- DESeq(dds)
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** and **limma** 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) for visualisation purposes:

```r
vsd <- 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:

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

```r
table
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 0.0000 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).

```r
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).

```r
plotPCA(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:

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

```
## 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     0.1736365 0.08062862   2.15015047
## ENSG000000000460.16 234.6479796     0.20589404 0.07445624   2.76530274
## ...                        ...            ...        ...          ...
## ENSG00000283695.1    0.0000000             NA         NA           NA
## ENSG00000283696.1  19.1311904    0.252144173  0.1545613  1.631353425
## ENSG00000283697.1  14.9180870    0.179070242  0.1522931  1.175826692
## ENSG00000283698.1   0.2289885   -0.003056215  0.7578201 -0.004032903
## ENSG00000283699.1    0.2289885    0.021962044  1.1315739  0.019408404
##                          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 5.398951 -0.003056215 0.7578201 -0.004032903
## ...                         ...          ...
## ENSG00000283695.1            NA           NA
## ENSG00000283696.1     0.1028158    0.3075119
## ENSG00000283697.1     0.2396641    0.4987872
## ENSG00000283698.1     0.9845153           NA
## ENSG00000283699.1     0.9967822           NA

We can look at a summary of the results:

```r
summary(res)
```

```
## 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%
## (mean count < 1)
## [1] see ‘cooksCutoff’ argument of ?results
## [2] see ‘independentFiltering’ argument of ?results
```

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

```r
plotMA(res, ylim = c(-5,5))
```
For convenience, we will save our differentially expressed genes (DEGs) in another object:}

defs <- subset(res, padj < 0.05)
defs <- as.data.frame(defs)
head(defs)

##                      baseMean log2FoldChange      lfcSE      stat
## ENSG00000000419.12  389.90251      0.6628823 0.11427371  5.800829
## ENSG00000000460.16  234.64798      0.2058940 0.07445624  2.765303
## ENSG00000002549.12 1970.95648      0.8657769 0.25181202  3.438187
## ENSG00000003096.13   11.18475     -0.7894018 0.25613621 -3.081961
## ENSG00000003147.17   71.79432      0.6113739 0.15162606  4.032116
## ENSG00000003249.13  119.18587     -0.8520562 0.27061961 -3.148538

##                          pvalue         padj
## ENSG00000000419.12 6.598777e-09 3.058479e-06
## ENSG00000000460.16 5.686999e-03 4.643041e-02
## ENSG00000002549.12 5.856225e-04 9.776328e-03
## ENSG00000003096.13 2.056419e-03 2.291728e-02
## ENSG00000003147.17 5.527679e-05 1.927054e-03
## ENSG00000003249.13 1.640893e-03 1.955034e-02

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:

rowData(rse)

## DataFrame with 58037 rows and 3 columns
##                  gene_id bp_length          symbol
##              <character> <integer> <CharacterList>
## 1     ENSG00000000003.14      4535          TSPAN6
## 2      ENSG00000000005.5      1610            TNMD
## 3     ENSG000000000419.12      1207            DPM1
## 4     ENSG000000000457.13      6883           SCYL3
## 5     ENSG000000000460.16      5967           Clorf112

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:
## DataFrames with 6 rows and 9 columns

<table>
<thead>
<tr>
<th>gene_id</th>
<th>bp_length</th>
<th>symbol</th>
<th>baseMean</th>
<th>log2FoldChange</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG00000283444.1</td>
<td>831</td>
<td>NA</td>
<td>2.756993</td>
<td>1.3404014</td>
</tr>
<tr>
<td>ENSG00000283479.1</td>
<td>420</td>
<td>NA</td>
<td>1.928773</td>
<td>1.9512651</td>
</tr>
<tr>
<td>ENSG00000283485.1</td>
<td>2190</td>
<td>ASPH</td>
<td>277.956104</td>
<td>1.3415229</td>
</tr>
<tr>
<td>ENSG00000283571.1</td>
<td>306</td>
<td>NA</td>
<td>1.791920</td>
<td>1.8502738</td>
</tr>
<tr>
<td>ENSG00000283602.1</td>
<td>2089</td>
<td>NA</td>
<td>130.233552</td>
<td>0.5752086</td>
</tr>
<tr>
<td>ENSG00000283623.1</td>
<td>594</td>
<td>ATG5</td>
<td>107.731105</td>
<td>0.4144398</td>
</tr>
</tbody>
</table>

### 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.

```r
library(gwascat)
# uncomment the following line to download file and build the gwasloc object all in one step
snps <- makeCurrentGwascat()
# uncomment the following line to download file
download.file("http://www.ebi.ac.uk/gwas/api/search/downloads/alternative",
destfile = "gwas_catalog_v1.0.1-associations_e90_r2017-12-04.tsv")
snps <- read.delim("gwas_catalog_v1.0.1-associations_e90_r2017-12-04.tsv",
check.names = FALSE, stringsAsFactors = FALSE)
gnome(snps) <- "GRCh38"
```

snps

## gwasloc instance with 61107 records and 37 attributes per record.
## Extracted: 2017-12-04
## Genotype: GRCh38
## Excerpt:
## GRanges object with 5 ranges and 3 metadata columns:
## seqnames ranges strand | DISEASE/TRAIT SNPS
## <Rle> <IRanges> <Rle> | <character> <character>
## [1] chr1 [203186754, 203186754] * | YKL-40 levels rs4950928
## [2] chr13 [39776775, 39776775] * | Psoriasis rs7993214

---

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 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 <- keepStandardChromosomes(txdb)
txdb
```

```r
## 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
```

Figure 4. Manhattan plot showing variants significantly associated with SLE.
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"

```r
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" "chrX"  "chrY"  "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)
```

```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>
## [1]    chr16 [ 31301932, 31301932]      + |     intron     40161         1      143788               ENSG00000169896.16
## [5]    chr8 [11491677, 11491677]      * |     intergenic <NA>          <NA>          <NA>          <NA>
## ...     ...    ...    ...    ...    ...    ...    ...    ...    ...
## [295]  chr6  [137874014, 137874014]    + |     intron     6162          <NA>          <NA>          <NA>
## [296]  chr6  [32619077, 32619077]      * |     intergenic <NA>          <NA>          <NA>          <NA>
## [297]  chr6  [137685367, 137685367]    + |     intron     11552         2       99581               ENSG00000070047.11
## [298]  chrX [153924366, 153924366]     - |     intergenic <NA>          <NA>          <NA>          <NA>
## [299]  chr5  [160459613, 160459613]    * |     intergenic <NA>          <NA>          <NA>          <NA>
##         LOCEND   TXID   CDSID             GENEID
##               <integer> <character> <integerList>        <character>
## [1]      40161          143788               ENSG00000169896.16
## [2]      12531            2       99581               ENSG00000070047.11
## [3]      51074            3       34101               ENSG000000168297.15
```
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))
```

```r
snps_anno

# GRanges object with 299 ranges and 11 metadata columns:
#>          seqnames ranges strand |           SNPS    P.VALUE
#>          <Rle>  <IRanges>  <Rle> |           <character>      <numeric>
#>       [1] chr16 [ 31301932,  31301932]      + |  rs9888739     2e-23
#>       [2] chr11 [ 589564,  589564]         + |  rs4963128     3e-10
#>       [3] chr3  [ 58384450,  58384450]      + |  rs6445975     7e-09
#>       [4] chr1  [173340574, 173340574]      * |  rs10798269    1e-07
#>       [5] chr8  [ 11491677,  11491677]      * |  rs13277113    1e-10
#>       ...     ...                     ... |           ...      ...
#>   [295] chr6 [137874014, 137874014]      + |  rs5029937     5e-13
#>   [296] chr6  [32619077, 32619077]        | |  rs9271366     1e-07
#>   [297] chr6  [137685367, 137685367]      + |  rs6920220     4e-07
#>   [298] chrX [153924366, 153924366]       | |  rs2269368     8e-07
#>   [299] chr5  [160459613, 160459613]      * |  rs2431099     2e-06
```
We can visualise where these SNPs are located with ggplot2 (Figure 5).
loc <- data.frame(table(snps_anno$LOCATION))

ggplot(data = loc, aes(x = reorder(Var1, -Freq), y = Freq)) +
  geom_bar(stat="identity")

As expected\(^1\), 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:

```r
snps_easy <- subset(snps_anno, LOCATION == "coding" | LOCATION == "promoter" |
LOCATION == "threeUTR" | LOCATION == "fiveUTR")
```

```r
snps_easy <- as.data.frame(snps_easy)
```

```r
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</td>
<td>4e-10</td>
<td>coding</td>
</tr>
<tr>
<td>chr7</td>
<td>128954129</td>
<td>128954129</td>
<td>1</td>
<td>-</td>
<td>rs10488631</td>
<td>2e-11</td>
<td>promoter</td>
</tr>
<tr>
<td>chr11</td>
<td>55368743</td>
<td>55368743</td>
<td>1</td>
<td>+</td>
<td>rs7927370</td>
<td>7e-06</td>
<td>coding</td>
</tr>
<tr>
<td>chr6</td>
<td>137874929</td>
<td>137874929</td>
<td>1</td>
<td>+</td>
<td>rs2230926</td>
<td>1e-17</td>
<td>coding</td>
</tr>
<tr>
<td>chr11</td>
<td>118702810</td>
<td>118702810</td>
<td>1</td>
<td>+</td>
<td>rs4639966</td>
<td>1e-16</td>
<td>promoter</td>
</tr>
<tr>
<td>chr16</td>
<td>30624338</td>
<td>30624338</td>
<td>1</td>
<td>-</td>
<td>rs7186852</td>
<td>3e-07</td>
<td>promoter</td>
</tr>
</tbody>
</table>

---

**Figure 5.** Barplot showing genomic locations associated with SLE variants.
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
snps_easy_in_degs
```

```
# DataFrame with 7 rows and 24 columns
#
# gene_id bp_length symbol baseMean
# <character> <integer> <list> <numeric>
# ENSG00000096968 ENSG00000096968.13 6170 JAK2 1279.47795
# ENSG00000099834 ENSG00000099834.18 3873 CDHR5 10.20177
# ENSG00000115267 ENSG00000115267.5 4528 IFIH1 1415.91330
# ENSG00000120280 ENSG00000120280.5 1855 CXorf21 637.78094
# ENSG00000185507 ENSG00000185507.19 2628 IRF7 4883.20891
# ENSG00000204366 ENSG00000204366.3 1875 ZBTB12 22.99200
# ENSG00000275106 ENSG00000275106.1 790 NA 10.32171
#
# log2FoldChange lfcSE stat pvalue
# <numeric> <numeric> <numeric> <numeric>
# ENSG00000096968 0.4854343 0.1553513 3.124753 1.779545e-03
# ENSG00000099834 0.8539586 0.2666557 3.202476 1.362516e-03
# ENSG00000115267 1.1494945 0.2729847 4.210838 2.544247e-05
# ENSG00000120280 0.7819504 0.1541707 5.071977 3.937038e-07
# ENSG00000185507 1.4062704 0.2992536 4.699260 2.611057e-06
# ENSG00000204366 -0.3892298 0.1348705 -2.885952 3.902318e-03
# ENSG00000275106 0.7344844 0.2305300 3.186068 1.442206e-03
#
# padj seqnames start end width
# <numeric> <factor> <integer> <integer> <integer>
# ENSG00000096968 2.068794e-02 chr9 4984530 4984530 1
# ENSG00000099834 1.732902e-02 chr11 625085 625085 1
# ENSG00000115267 1.120363e-03 chr2 162267541 162267541 1
# ENSG00000120280 6.047898e-05 chrX 30559729 30559729 1
# ENSG00000185507 2.298336e-04 chr11 614318 614318 1
# ENSG00000204366 3.584479e-02 chr6 31902549 31902549 1
# ENSG00000275106 1.797861e-03 chr7 128954129 128954129 1
#
# strand SNPS P.VALUE LOCATION LOCSTART
# <factor> <character> <numeric> <factor> <integer>
# ENSG00000096968 + rs1887428 1e-06 fiveUTR 141
# ENSG00000099834 - rs58688157 5e-13 promoter NA
# ENSG00000115267 - rs1990760 4e-08 coding 2836
# ENSG00000120280 - rs887369 5e-10 coding 627
# ENSG00000185507 - rs1061502 9e-11 coding 217
# ENSG00000204366 - rs558702 8e-21 promoter NA
# ENSG00000275106 - rs10488631 2e-11 promoter NA
#
# LOCEND QUERYID TXID
# <integer> <integer> <character>
# ENSG00000096968 141 329 86536
# ENSG00000099834 NA 208 105793
# ENSG00000115267 2836 233 29219
# ENSG0000000156853.12
```
## ENSG00000120280       627       192      194672                   692823
## ENSG00000185507       217       317      105777 385431,385427,385428,...
## ENSG00000204366        NA       116       65993
## ENSG00000275106        NA        23       77786

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"))
```

---

Page 20 of 42
### 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\(^1\). We will use blood eQTL data from the GTEx consortium\(^2\). To get the data, you will have to register and download the file GTEx_Analysis_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_Analysis_v7_eQTL.tar.gz")

gtex_blood <- read.delim(gzfile("GTEx_Analysis_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>
</tr>
</thead>
<tbody>
<tr>
<td>1_231153_CTT_C_b37</td>
<td>ENSG00000223972.4</td>
<td>219284</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>1_61920_G_A_b37</td>
<td>ENSG000000238009.2</td>
<td>-67303</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>1_64649_A_C_b37</td>
<td>ENSG000000238009.2</td>
<td>-64574</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>1_115746_C_T_b37</td>
<td>ENSG000000238009.2</td>
<td>-13477</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>1_135203_G_A_b37</td>
<td>ENSG000000238009.2</td>
<td>5980</td>
<td>51</td>
<td>51</td>
</tr>
<tr>
<td>1_980816_T_C_b37</td>
<td>ENSG0000000268903.1</td>
<td>852121</td>
<td>21</td>
<td>23</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>maf  pval_nominal</th>
<th>slope</th>
<th>slope_se</th>
<th>pval_nominal_threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0.0191740 3.69025e-08 1.319720 0.233538</td>
<td>1.35366e-04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 0.0281690 7.00836e-07 0.903786 0.178322</td>
<td>8.26088e-05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 0.0220386 5.72066e-07 1.110040 0.217225</td>
<td>8.26088e-05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 0.0628492 6.50297e-10 0.858203 0.134436</td>
<td>8.26088e-05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 0.0698630 6.67194e-10 0.811790 0.127255</td>
<td>8.26088e-05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 0.0318560 6.35694e-05 0.501916 0.123743</td>
<td>8.52870e-05</td>
<td></td>
<td></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)
```

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
# ... ... ... ... ... ... ...
# [105238] X [154999204, 154999204] * | X_154999204_TA_T_b37
# [105239] X [155004280, 155004280] * | X_155004280_A_G_b37
# [105240] X [155011926, 155011926] * | X_155011926_T_C_b37
# [105241] X [155014420, 155014420] * | X_155014420_A_G_b37
# [105242] X [155186978, 155186978] * | X_155186978_G_C_b37
```
## Dimensions

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>ENSG00000223972.4</td>
<td>219284</td>
<td>13</td>
<td>13</td>
<td>0.0191740</td>
</tr>
<tr>
<td>[2]</td>
<td>ENSG00000238009.2</td>
<td>-67303</td>
<td>18</td>
<td>20</td>
<td>0.0281690</td>
</tr>
<tr>
<td>[3]</td>
<td>ENSG00000238009.2</td>
<td>-64574</td>
<td>16</td>
<td>16</td>
<td>0.0220386</td>
</tr>
<tr>
<td>[4]</td>
<td>ENSG00000238009.2</td>
<td>-13477</td>
<td>45</td>
<td>45</td>
<td>0.0628492</td>
</tr>
<tr>
<td>[5]</td>
<td>ENSG00000238009.2</td>
<td>5980</td>
<td>51</td>
<td>51</td>
<td>0.0698630</td>
</tr>
</tbody>
</table>

## Dimensions (continued)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1052538]</td>
<td>ENSG00000168939.6</td>
<td>1730</td>
<td>219</td>
<td>274</td>
<td>0.390313</td>
</tr>
<tr>
<td>[1052539]</td>
<td>ENSG00000168939.6</td>
<td>6806</td>
<td>186</td>
<td>224</td>
<td>0.303523</td>
</tr>
<tr>
<td>[1052540]</td>
<td>ENSG00000168939.6</td>
<td>14452</td>
<td>222</td>
<td>279</td>
<td>0.379076</td>
</tr>
<tr>
<td>[1052541]</td>
<td>ENSG00000168939.6</td>
<td>16946</td>
<td>215</td>
<td>265</td>
<td>0.360054</td>
</tr>
<tr>
<td>[1052542]</td>
<td>ENSG00000168939.6</td>
<td>189504</td>
<td>250</td>
<td>321</td>
<td>0.436141</td>
</tr>
</tbody>
</table>

## pval_nominal slope slope_se pval_nominal_threshold

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>3.69025e-08</td>
<td>1.319720</td>
<td>0.233538</td>
<td>1.35366e-04</td>
</tr>
<tr>
<td>[2]</td>
<td>7.00836e-07</td>
<td>0.903786</td>
<td>0.178322</td>
<td>8.26088e-05</td>
</tr>
<tr>
<td>[3]</td>
<td>5.72066e-07</td>
<td>1.110040</td>
<td>0.217225</td>
<td>8.26088e-05</td>
</tr>
<tr>
<td>[4]</td>
<td>6.50297e-10</td>
<td>0.858203</td>
<td>0.134436</td>
<td>8.26088e-05</td>
</tr>
<tr>
<td>[5]</td>
<td>6.67194e-10</td>
<td>0.811790</td>
<td>0.127255</td>
<td>8.26088e-05</td>
</tr>
</tbody>
</table>

## Dimensions (continued)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1052538]</td>
<td>6.72752e-05</td>
<td>-0.157810</td>
<td>0.0390413</td>
<td>0.000130368</td>
<td></td>
</tr>
<tr>
<td>[1052539]</td>
<td>1.91420e-08</td>
<td>0.230301</td>
<td>0.0398809</td>
<td>0.000130368</td>
<td></td>
</tr>
<tr>
<td>[1052540]</td>
<td>3.88977e-05</td>
<td>0.157608</td>
<td>0.0377434</td>
<td>0.000130368</td>
<td></td>
</tr>
<tr>
<td>[1052541]</td>
<td>4.17781e-05</td>
<td>0.159699</td>
<td>0.0384025</td>
<td>0.000130368</td>
<td></td>
</tr>
<tr>
<td>[1052542]</td>
<td>1.24355e-04</td>
<td>0.145560</td>
<td>0.0374390</td>
<td>0.000130368</td>
<td></td>
</tr>
</tbody>
</table>

## min_pval_nominal pval_beta

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>3.69025e-08</td>
<td>4.67848e-05</td>
</tr>
<tr>
<td>[2]</td>
<td>6.50297e-10</td>
<td>1.11312e-06</td>
</tr>
<tr>
<td>[3]</td>
<td>6.50297e-10</td>
<td>1.11312e-06</td>
</tr>
<tr>
<td>[4]</td>
<td>6.50297e-10</td>
<td>1.11312e-06</td>
</tr>
<tr>
<td>[5]</td>
<td>6.50297e-10</td>
<td>1.11312e-06</td>
</tr>
</tbody>
</table>

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

```
seqlevelsStyle(gtex_blood)
```

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

```
seqlevels(gtx_blood)
```

```
# [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"
```

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

```
seqlevels(gtx_blood)
```

```
# [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"
```
library(ttracklayer)
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 to compute the overlap between GWAS SNPs and blood eQTLs:

library(GenomicRanges)

hits <- findOverlaps(snps_hard, gtex_blood)

snps_hard_in_gtex_blood <- snps_hard[queryHits(hits)]
gtex_blood_with_snps_hard <- gtex_blood[subjectHits(hits)]

mcols(snps_hard_in_gtex_blood) <- cbind(mcols(snps_hard_in_gtex_blood), mcols(gtex_blood_with_snps_hard))

snps_hard_in_gtex_blood <- as.data.frame(snps_hard_in_gtex_blood)

head(snps_hard_in_gtex_blood)

##   seqnames    start      end width strand       SNPS P.VALUE   LOCATION
## 1    chr11   589564   589564     1      +  rs4963128   3e-10     intron
## 2     chr3 58384450 58384450     1      +  rs6445975   7e-09     intron
## 3     chr8 11491677 11491677     1      * rs13277113 1e-10     intergenic
## 4     chr8 11491677 11491677     1      * rs13277113 1e-10     intergenic
## 5     chr8 11491677 11491677     1      * rs13277113 1e-10     intergenic
## 6     chr8 11491677 11491677     1      * rs13277113 1e-10     intergenic

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 eGene, a gene whose expression is influenced by an eQTL. We note that gene IDs in GTEx are mapped to GENCODE v19, 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)
```

<table>
<thead>
<tr>
<th>ensembl_id</th>
<th>seqnames</th>
<th>start</th>
<th>end</th>
<th>width</th>
<th>strand</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG00000130513</td>
<td>chr19</td>
<td>18370523</td>
<td>18370523</td>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>ENSG00000140497</td>
<td>chr15</td>
<td>75018695</td>
<td>75018695</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>ENSG00000172890</td>
<td>chr11</td>
<td>71476633</td>
<td>71476633</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>ENSG00000214894</td>
<td>chr6</td>
<td>31668965</td>
<td>31668965</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>ENSG00000214894</td>
<td>chr6</td>
<td>30973212</td>
<td>30973212</td>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>ENSG00000214894</td>
<td>chr6</td>
<td>31753256</td>
<td>31753256</td>
<td>1</td>
<td>+</td>
</tr>
</tbody>
</table>

```r
## DataFrame with 6 rows and 30 columns
##   ensembl_id seqnames start end width strand
## 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  ENSG00000214894     chr6  31753256  31753256         1        +
### SNPS
##   P.VALUE LOCATION variant_id gene_id.x
## 1 5e-06 intergenic 19_18481333_A_G_b37 ENSG00000130513.6
## 2 6e-15     intron 15_75311036_C_A_b37 ENSG00000140497.12
## 3 1e-20     intron 11_71187679_C_T_b37  ENSG00000172890.7
## 4 8e-08     intron  6_31636742_A_G_b37  ENSG00000214894.2
## 5 6e-92 intergenic  6_30940989_T_C_b37  ENSG00000214894.2
## 6 2e-52     intron  6_31721033_G_A_b37  ENSG00000214894.2
### tss_distance ma_samples ma_count maf pval_nominal slope
## 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
### slope_se pval_nominal_threshold min_pval_nominal pval_beta
## 1 0.0520458            2.52102e-05      1.76820e-11 1.23175e-07
## 2 0.0235858            6.38531e-05      2.44784e-27 1.10743e-22
## 3 0.0310305            4.46719e-05      1.05596e-33 7.87659e-28
## 4 0.0846154            6.02220e-05      3.17673e-13 1.77790e-08
## 5 0.0670210            6.02220e-05      3.17673e-13 1.77790e-08
## 6 0.0840440            6.02220e-05      3.17673e-13 1.77790e-08
### gene_id.y bp_length symbol baseMean log2FoldChange
## 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  ENSG000000214894.6     2171 LINC00243 74.95034      1.2684089
## 6  ENSG000000214894.6     2171 LINC00243 74.95034      1.2684089
### lfcSE stat pvalue padj
## 1 0.28347645  2.781079 5.417861e-03 0.0448154406
## 2 0.08814542 -3.385012 7.850510e-04 0.0119267855
## 3 0.08976429  2.918499 3.517209e-03 0.0333810138
## 4 0.27106143  4.679415 2.876950e-06 0.0002442643
## 5 0.27106143  4.679415 2.876950e-06 0.0002442643
## 6 0.27106143  4.679415 2.876950e-06 0.0002442643
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,
  snp_location = snps_hard_in_gtex_blood_in_degs$LOCATION,
  gene_id = snps_hard_in_gtex_blood_in_degs$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)))
```

<table>
<thead>
<tr>
<th>snp_id</th>
<th>snp_pvalue</th>
<th>snp_location</th>
<th>gene_id</th>
<th>gene_symbol</th>
<th>gene_pvalue</th>
<th>gene_log2foldchange</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG00000096968</td>
<td>1e-06</td>
<td>fiveUTR</td>
<td>ENSG00000096968.13</td>
<td>JAK2</td>
<td>0.4854343</td>
<td>0.4854343</td>
</tr>
<tr>
<td>ENSG00000099834</td>
<td>5e-13</td>
<td>promoter</td>
<td>ENSG00000099834.18</td>
<td>CDHR5</td>
<td>0.8539586</td>
<td>0.8539586</td>
</tr>
<tr>
<td>ENSG00000115267</td>
<td>4e-08</td>
<td>coding</td>
<td>ENSG00000115267.5</td>
<td>IFIH1</td>
<td>1.1494945</td>
<td>1.1494945</td>
</tr>
<tr>
<td>ENSG00000120280</td>
<td>5e-10</td>
<td>coding</td>
<td>ENSG00000120280.5</td>
<td>CXorf21</td>
<td>0.7819504</td>
<td>0.7819504</td>
</tr>
<tr>
<td>ENSG00000185507</td>
<td>9e-11</td>
<td>coding</td>
<td>ENSG00000185507.19</td>
<td>ZBTB12</td>
<td>-0.3892298</td>
<td>-0.3892298</td>
</tr>
<tr>
<td>ENSG00000204366</td>
<td>8e-21</td>
<td>promoter</td>
<td>ENSG00000204366.3</td>
<td>NADSYN1</td>
<td>0.7344844</td>
<td>0.7344844</td>
</tr>
<tr>
<td>ENSG00000275106</td>
<td>2e-11</td>
<td>promoter</td>
<td>ENSG00000275106.1</td>
<td>LINC00243</td>
<td>1.2684089</td>
<td>1.2684089</td>
</tr>
<tr>
<td>ENSG00000096968</td>
<td>1e-06</td>
<td>fiveUTR</td>
<td>ENSG00000096968.13</td>
<td>JAK2</td>
<td>0.4854343</td>
<td>0.4854343</td>
</tr>
<tr>
<td>ENSG00000099834</td>
<td>5e-13</td>
<td>promoter</td>
<td>ENSG00000099834.18</td>
<td>CDHR5</td>
<td>0.8539586</td>
<td>0.8539586</td>
</tr>
<tr>
<td>ENSG00000115267</td>
<td>4e-08</td>
<td>coding</td>
<td>ENSG00000115267.5</td>
<td>IFIH1</td>
<td>1.1494945</td>
<td>1.1494945</td>
</tr>
<tr>
<td>ENSG00000120280</td>
<td>5e-10</td>
<td>coding</td>
<td>ENSG00000120280.5</td>
<td>CXorf21</td>
<td>0.7819504</td>
<td>0.7819504</td>
</tr>
<tr>
<td>ENSG00000185507</td>
<td>9e-11</td>
<td>coding</td>
<td>ENSG00000185507.19</td>
<td>ZBTB12</td>
<td>-0.3892298</td>
<td>-0.3892298</td>
</tr>
<tr>
<td>ENSG00000204366</td>
<td>8e-21</td>
<td>promoter</td>
<td>ENSG00000204366.3</td>
<td>NADSYN1</td>
<td>0.7344844</td>
<td>0.7344844</td>
</tr>
<tr>
<td>ENSG00000275106</td>
<td>2e-11</td>
<td>promoter</td>
<td>ENSG00000275106.1</td>
<td>LINC00243</td>
<td>1.2684089</td>
<td>1.2684089</td>
</tr>
</tbody>
</table>

**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_tssAssociations.bed", destfile = "enhancer_tss_associations.bed")
fantom <- read.delim("enhancer_tssAssociations.bed", skip = 1, stringsAsFactors = FALSE)
head(fantom)
```

<table>
<thead>
<tr>
<th>X.chrom</th>
<th>chromStart</th>
<th>chromEnd</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr1</td>
<td>858252</td>
<td>861621</td>
</tr>
<tr>
<td>chr1</td>
<td>894178</td>
<td>956888</td>
</tr>
<tr>
<td>chr1</td>
<td>901376</td>
<td>956888</td>
</tr>
</tbody>
</table>
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
## 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
##            blockCount blockSizes chromStarts          name_1
## 1             2  401,1001     0,2368   chr1:858256-858648
## 2            2  1001,401     0,62309 chr1:856563-956812
## 3            2  1001,401     0,55111 chr1:956563-956812
## 4            2  1001,401     0,271985 chr1:1173386-1173736
## 5            2  1001,401     0,6712  chr1:941791-942135
## 6            2  1001,401     0,70169  chr1:1005293-1005547
##          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 NM_001142467,NM_021170    HES4 R:0.187  FDR:6.32949888009368e-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)
```

```
##     chr   start     end  symbol
## 1  chr1  858256  858648  SAMD11
## 2  chr1  894178  956888    NOC2L
## 3  chr1  901376  956888   PLEKHN1
## 4  chr1  901376 1173762   PLEKHN1
## 5  chr1  935051  942164    AGRN
## 6  chr1  935051 1005621   RNF223
```

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
## 2  chr1  894178  956888    NOC2L
## 3  chr1  901376  956888   PLEKHN1
## 4  chr1  941963  941964    AGRN
## 5  chr1 1005293 1005547   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)
```

```r
tansom
## 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))
```

```r
tansom
## 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))
```

```r
snps_hard_in_fantom <- as.data.frame(snps_hard_in_fantom)
```

```r
tansom
##    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
```
## 3  chr6  32082981  32082981     1      -  rs1150754   6e-29     intron
## 4  chr6  32082981  32082981     1      -  rs1150754   6e-29     intron
## 5  chr6  32082981  32082981     1      -  rs1150754   6e-29     intron
## 6  chr6  32689659  32689659     1      *  rs3129716   4e-09 intergenic
## 7  chr6  32689659  32689659     1      *  rs3129716   4e-09 intergenic
## 8  chr6  32689659  32689659     1      *  rs3129716   4e-09 intergenic
## 9  chr6  32689659  32689659     1      *  rs3129716   4e-09 intergenic
##10 chr6  32689659  32689659     1      *  rs3129716   4e-09 intergenic
##11 chr6  32689659  32689659     1      *  rs3129716   4e-09 intergenic
##12 chr1  235876577 235876577     1      -  rs9782955   1e-09     intron
##13 chr7  50267214  50267214     1      *  rs11185603  4e-07 intergenic
##14 chr11 73152652  73152652     1      *   rs1235667  7e-11 intergenic

# symbol
# 1  NAB1
# 2  STAT4
# 3  LY6G6C
# 4  TNXB
# 5  PPT2
# 6  HLA-DQB1
# 7  HLA-DQB
# 8  HLA-DMA
# 9  HLA-DQA
#10 HLA-DPA1
#11 HLA-DPB1
#12 LYST
#13 IKZF1
#14 FCHSD2

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)

snps_hard_in_fantom_in_degs
```

```r
## DataFrame with 2 rows and 18 columns
## symbol seqnames     start       end     width   strand        SNPS
##   <factor> <factor> <integer> <integer> <integer> <factor> <character>
## 1  HLA-DOA     chr6  32689659  32689659         1        *   rs3129716
## 2    IKZF1     chr7  50267214  50267214         1        *  rs11185603
##     P.VALUE   LOCATION            gene_id bp_length  baseMean
##   <numeric>   <factor>        <character> <integer> <numeric>
## 1     4e-09 intergenic ENSG00000204252.13      4012  962.7578
## 2     4e-07 intergenic ENSG00000185811.16      9784 7183.7639
## log2FoldChange      lfcSE      stat       pvalue       padj
##        <numeric>  <numeric> <numeric>    <numeric>  <numeric>
## 1  -0.4424595 0.15882236 -2.785877 0.0053383163 0.04431304
## 2  -0.2575717 0.07647486 -3.368057 0.0007569983 0.01162554
## ensembl_id
##       <character>
## 1 ENSG000000204252
## 2 ENSG00000185811
```
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)))

prioritised_hits
```

<table>
<thead>
<tr>
<th>snp_id</th>
<th>snp_pvalue</th>
<th>snp_location</th>
<th>gene_id</th>
<th>gene_symbol</th>
<th>gene_pvalue</th>
<th>gene_log2foldchange</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG00000096968</td>
<td>rs1887428</td>
<td>1e-06</td>
<td>fiveUTR</td>
<td>ENSG00000096968.13</td>
<td>2.068794e-02</td>
<td>0.4854343</td>
</tr>
<tr>
<td>ENSG00000099834</td>
<td>rs58688157</td>
<td>5e-13</td>
<td>promoter</td>
<td>ENSG00000099834.18</td>
<td>1.732902e-02</td>
<td>0.8539586</td>
</tr>
<tr>
<td>ENSG00000115267</td>
<td>rs1990760</td>
<td>4e-08</td>
<td>coding</td>
<td>ENSG00000115267.5</td>
<td>1.120363e-03</td>
<td>1.1494945</td>
</tr>
<tr>
<td>ENSG00000120280</td>
<td>rs887369</td>
<td>5e-10</td>
<td>coding</td>
<td>ENSG00000120280.5</td>
<td>2.298336e-04</td>
<td>1.4062704</td>
</tr>
<tr>
<td>ENSG00000185507</td>
<td>rs1061502</td>
<td>9e-11</td>
<td>coding</td>
<td>ENSG00000185507.19</td>
<td>3.584479e-02</td>
<td>0.3892298</td>
</tr>
<tr>
<td>ENSG00000204366</td>
<td>rs558702</td>
<td>8e-21</td>
<td>promoter</td>
<td>ENSG00000204366.3</td>
<td>1.797861e-02</td>
<td>0.7344844</td>
</tr>
<tr>
<td>ENSG00000275106</td>
<td>rs1048631</td>
<td>2e-11</td>
<td>promoter</td>
<td>ENSG00000275106.1</td>
<td>4.481544e-02</td>
<td>0.7883703</td>
</tr>
<tr>
<td>1</td>
<td>rs8105429</td>
<td>5e-06</td>
<td>intergenic</td>
<td>ENSG00000130513.6</td>
<td>1.192679e-02</td>
<td>-0.2959934</td>
</tr>
<tr>
<td>2</td>
<td>rs2289583</td>
<td>6e-15</td>
<td>intron</td>
<td>ENSG00000140497.16</td>
<td>1.120363e-03</td>
<td>1.1494945</td>
</tr>
<tr>
<td>3</td>
<td>rs3794060</td>
<td>1e-20</td>
<td>intron</td>
<td>ENSG00000172890.11</td>
<td>2.298336e-04</td>
<td>1.4062704</td>
</tr>
<tr>
<td>4</td>
<td>rs9267531</td>
<td>8e-08</td>
<td>intron</td>
<td>ENSG00000214894.6</td>
<td>3.584479e-02</td>
<td>0.3892298</td>
</tr>
<tr>
<td>5</td>
<td>rs114090659</td>
<td>6e-92</td>
<td>intergenic</td>
<td>ENSG00000214894.6</td>
<td>1.120363e-03</td>
<td>1.1494945</td>
</tr>
<tr>
<td>6</td>
<td>rs3131379</td>
<td>2e-52</td>
<td>intron</td>
<td>ENSG00000214894.6</td>
<td>2.298336e-04</td>
<td>1.4062704</td>
</tr>
<tr>
<td>11</td>
<td>rs3129716</td>
<td>4e-09</td>
<td>intergenic</td>
<td>ENSG00000204252.13</td>
<td>1.192679e-02</td>
<td>-0.2959934</td>
</tr>
<tr>
<td>21</td>
<td>rs11185603</td>
<td>4e-07</td>
<td>intergenic</td>
<td>ENSG00000185811.16</td>
<td>1.192679e-02</td>
<td>-0.2959934</td>
</tr>
</tbody>
</table>

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_S1.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
baits <- GRanges(seqnames = pchic$baitChr, ranges = IRanges(start = pchic$baitSt, end = pchic$baitEnd))
tsss <- promoters(txdb, upstream = 0, downstream = 1, columns = "gene_id")
hits <- nearest(baits, tsss)
baits$gene_id <- unlist(tsss[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] * | ENSG00000230727.10
## [51139] chrY [22732049, 22743996] * | ENSG00000230727.10
## [51140] chrY [22732049, 22743996] * | ENSG00000230727.10
## [51141] chrY [22732049, 22743996] * | ENSG00000230727.10
## [51142] chrY [22732049, 22743996] * | ENSG00000230727.10
## -----
## 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)
```

```
## 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  
```

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)
```

```
## Accessing overlapping SNPs
snps_hard_in_pchic = snps_hard[queryHits(hits)]
```

```r
pchic_with_snps_hard = pchic[subjectHits(hits)]
```

```r
mcols(snps_hard_in_pchic) <- cbind(mcols(snps_hard_in_pchic),
                                  mcols(pchic_with_snps_hard))
```

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

```
## 1    chr6  31753256  31753256     1      +   rs3131379   2e-52
## 2    chr6  32696681  32696681     1      *   rs2647012   8e-06
## 3  chr16  30631546  30631546     1      *   rs7197475   3e-08
## 4  chr20   4762059   4762059     1      *   rs6084875   2e-06
## 5    chr6  32689659  32689659     1      *   rs3129716   4e-09
## 6    chr6  31668965  31668965     1      +   rs9267531   8e-08
## 7    chr6  31951083  31951083     1      +   rs1270942  2e-165
## 8    chr6 106140931 106140931     1      -    rs6568431  5e-14
## 9    chr7  28146272  28146272     1      -    rs849142   9e-11
## 10   chr2  65381229  65381229     1      -    rs268134  1e-10
## 11   chr7  39850937  39850937     1      -    rs14312327 6e-09
## 12   chr9  86916761  86916761     1      *    rs19002901 3e-06
## 13  chr11  65637829  65637829     1      *    rs931127  7e-06
## 14  chr19  18370523  18370523     1      *    rs8105429  5e-06
## 15  chr6  85977731  85977731     1      *    rs1052138  4e-06
## 16   chr5  39406395  39406395     1      -    rs391467  8e-06
## 17  chr16  31315385  31315385     1      +    rs11860650 2e-20
```

## seqinfo: 24 sequences from an unspecified genome; no seqlengths
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
## DataFrame with 4 rows and 18 columns
##              gene_id seqnames     start       end     width   strand
##          <character> <factor> <integer> <integer> <integer> <factor>
## 1 ENSG00000106052.13     chr7  28146272  28146272         1        -
## 2  ENSG00000219797.2     chr6  31753256  31753256         1        +
## 3  ENSG00000219797.2     chr6  31668965  31668965         1        +
## 4  ENSG00000245532.6    chr11  65637829  65637829         1        *
##          SNPS   P.VALUE   LOCATION bp_length       symbol    baseMean
##   <character> <numeric>   <factor> <integer>       <list>   <numeric>
## 1    rs849142     9e-11     intron      9165      TAX1BP1  2406.26093
## 2   rs3131379     2e-52     intron       498           NA    74.58175
## 3   rs9267531     8e-08     intron       498           NA    74.58175
## 4    rs931127     7e-06 intergenic     22767 NEAT1,MIR612 17580.27601
##     log2FoldChange     lfcSE      stat       pvalue         padj
##            <numeric> <numeric> <numeric>    <numeric>    <numeric>
## 1      0.3438396 0.1205716  2.851746 4.347982e-03 0.0386695506
## 2      0.5586633 0.1116884  5.001982 5.674388e-07 0.0000798169
## 3      0.5586633 0.1116884  5.001982 5.674388e-07 0.0000798169
## 4      0.5259525 0.1366133  3.849935 1.181492e-04 0.0032213554
##        ensembl_id
##       <character>
## 1 ENSG00000106052
## 2 ENSG00000219797
## 3 ENSG00000219797
## 4 ENSG00000245532
```

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

```r
prioritised_hits <- unique(rbind(prioritised_hits, data.frame(
    snp_id = snps_hard_in_pchic_in_degs$SNPS,
    snp_pvalue = snps_hard_in_pchic_in_degs$P.VALUE,
    snp_location = snps_hard_in_pchic_in_degs$LOCATION,
    gene_id = snps_hard_in_pchic_in_degs$gene_id,
    gene_symbol = snps_hard_in_pchic_in_degs$symbol,
    gene_pvalue = snps_hard_in_pchic_in_degs$padj,
    gene_log2foldchange = snps_hard_in_pchic_in_degs$log2FoldChange)))
prioritised_hits
```
## 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\(^4\) and enhancer - promoter relationships from CAGE\(^19\) and promoter capture Hi-C experiments\(^25\).

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.

### References


### Bioconductor Workflow

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\(^4\) and enhancer - promoter relationships from CAGE\(^19\) and promoter capture Hi-C experiments\(^25\).

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.

### 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\(^4\) and enhancer - promoter relationships from CAGE\(^19\) and promoter capture Hi-C experiments\(^25\).

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.

### Bioconductor Workflow

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\(^4\) and enhancer - promoter relationships from CAGE\(^19\) and promoter capture Hi-C experiments\(^25\).

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.

### 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\(^4\) and enhancer - promoter relationships from CAGE\(^19\) and promoter capture Hi-C experiments\(^25\).

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.

### Bioconductor Workflow

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\(^4\) and enhancer - promoter relationships from CAGE\(^19\) and promoter capture Hi-C experiments\(^25\).

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.

### 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\(^4\) and enhancer - promoter relationships from CAGE\(^19\) and promoter capture Hi-C experiments\(^25\).

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.

### 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\(^4\) and enhancer - promoter relationships from CAGE\(^19\) and promoter capture Hi-C experiments\(^25\).

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


Publisher Full Text


PubMed Abstract | Publisher Full Text | Free Full Text


Published Abstract | Publisher Full Text


Published Abstract | Publisher Full Text


Published Abstract | Publisher Full Text | Free Full Text


PubMed Abstract | Publisher Full Text | Free Full Text


PubMed Abstract | Publisher Full Text | Free Full Text


Published Abstract | Publisher Full Text | Free Full Text


PubMed Abstract | Publisher Full Text | Free Full Text


Published 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


Publisher Full Text


Publisher Full Text


PubMed Abstract | Publisher Full Text | Free Full Text


PubMed Abstract | Publisher Full Text | Free Full Text


Publisher 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


Data Source
I want to preface this long review with some very broad comments. I think this undertaking is very worthwhile from several perspectives. Bioconductor is used along various avenues to create a unifiable analytic process from very diverse data resources: state-of-the-art transcriptomics from recount, current GWAS catalog from EMBL/EBI, variant annotation for SLE GWAS hits from the eponymous package using GENCODE for gene models, eQTL data from GTEx, enhancer annotation from FANTOM5, and promoter capture data whose origins could be better described. This is a tour de force but I feel it should be communicated more clearly and executed more cleanly. The paper is full of "dumps" of show events for R objects that impede the narrative flow drastically. A diagram that shows how the various resources combine in a scientifically coherent way would be a huge step forward for the paper and for practitioners. More reckoning of limitations that arise from complexity is also in order. eQTLs are far from simple, and should not be used as 'lists'. Enhancer and promoter 'lists' also need to be used with care.

What then about this paper? It shows the resources and it shows a path. Isn't that enough? I don't think so. If Bioconductor and online publication make it easier to do and to publish complex analyses, then the presentation should be of at least as high a quality as we find in articles that are behind paywalls. In this case I feel the quality would be improved through condensation. The object dumps should be removed and replaced by meaningful tabulations and diagrams. The big picture should be stated more clearly and concisely. The limitations should also be discussed clearly. I would love to see a small set of functions that carry out the salient operations chained together to produce the solution. Then, given the programmatic compactness, we can discuss how to evaluate the robustness of the results of the analysis by carrying out sensitivity analysis. In particular, it would be great to see how the different elements of the system contribute to the ultimate enumeration of targets.

---

The premise of this article is that "therapeutic targets with a genetic link to the disease under investigation are more likely to progress through the drug discovery pipeline". GWAS, PheWAS, eQTLs, epigenomic roadmap projects, and other general studies of gene regulation should be harvested to improve capacity to define genetic and genomic origins of disease, with an aim to fostering design of treatments that are focused on the molecular events underlying the disease process. The introduction concludes with mention of STOPGAP, and POSTGAP, and INFERNO, but it is not clear whether the paper is intended to describe how content of STOPGAP is developed from basic data resources like those readily available to
Bioconductor users. I feel that the introduction, though well-referenced, is too long and does not clearly state the paper's main goal.

There is no discussion of the experimental design underlying the RNA-seq study. Presumably the data were generated from this component of ref 28:

"Finally, we tested the levels of Alu transcripts in blood cells of SLE patients and controls using RNAseq (99 active SLE, 18 healthy controls; Fig. S12). RNA-seq reads mapping to Alu elements were found at significantly higher levels in SLE subjects than controls (p=6.5E-6), Fig. 4E). Hierarchical clustering of the most highly expressed Alu RNAs (Fig. S13) segregated Interferon Signature Metric (ISM)-high SLE subjects from control and ISM-low patients"....

There is no discussion of heterogeneity of SLE or the difficulty of learning from a collection of 18 cases. A reference to https://www.ncbi.nlm.nih.gov/pubmed/25102991 may be in order.

Even though online publications are often free from page count limitations, entirely too much space is consumed by long row-broken R print events. On the one hand the recoding of SRA annotation on phenotype is important and should be exposed, on the other hand, the author could carry out the recoding programmatically in a well-parameterized function and simply update the key object by applying this function. The function can go in a package related to the paper/workflow. Instead of printing out a dataframe on p.7, it would be much better to have a contingency table showing the final layout of case and control characteristics.

p.7 "For simplicity, we select the first 18 (healthy) and the last 18 (SLE) samples from the original RangedSummarizedExperiment object". Is this essential to the performance of the workflow? Would a more systematic matching be possible? What kind of "simplicity" does this arbitrary selection create? I understand that the main purpose of the paper is to illustrate a process, but if this thinning of the data is not essential to the illustration, why do it?

p. 8: "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 and limma are good alternatives to DESeq2 for performing differential expression analyses." This suggests that you can't adjust for confounders in DESeq2, is this so? Did you not have access to any relevant cofactors in the SLE data?

p. 9: You are really using 59000 genes after vst to do exploratory visualization of SLE vs control expression patterns? Would gene filtering be helpful? Is there any chance of batch effect or other surrogate variable effect that should be assessed prior to such presentations?

By page 12 we have completed a relatively elementary differential expression analysis. It seems to me that the length of this part of the process is excessive, because the real interest is in learning about regulatory elements from other resources.

At this point I hope I have made clear how I think the rest of the paper should be revised to make its points more effectively.

Is the rationale for developing the new method (or application) clearly explained? Yes
Is the description of the method technically sound?
Partly

Are sufficient details provided to allow replication of the method development and its use by others?
Yes

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?
Partly

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.
obtain the gene-level read counts.

- `rse$FIELD` can be used instead rather than `colData(rse)$FIELD`, which may simplify the code.

- Some explanation of the other factors (anti-rho, ISM) would be helpful, given that the effort has already been taken to define them.

- The simplicity of the model used in the DE analysis is probably unhelpful in the context described in the workflow. I would like to see more elaboration on how to handle batch effects and other confounding factors that are almost definitely present in large-scale studies. For example, what happens to the DE genes when additional explanatory factors are added to the model, e.g., anti-rho or ism status? Presumably age and sex are also relevant factors, if that information is available in the data set.

- Generally, some of the plots could be accompanied by more commentary in text, explaining how to interpret the plot. For example, the MA plot in Figure 3 shows that DE genes are detected in both directions, at a range of abundances. It would be similarly useful to have text for the heatmap in Figure 1 and the Manhattan plot in Figure 4, among others.

- LD expansion seems like quite an important step, especially when SNPs are being linked to genes based on overlaps to promoters/UTRs. If the LD blocks are large, expansion would result in many more potential causal SNPs and a greater number of overlaps (and thus candidate genes). While I appreciate the attempt to simplify the workflow, skipping this step seems like it would unnecessarily reduce the number of candidate genes.

- `snps` seems to have GRCh38 coordinates. Is this also the case for GENCODE 25? It would be helpful to have a cautionary note regarding the need to make sure the same version of the genome is used throughout a workflow. I recognise that this is mentioned later when `liftOver()` is used, but it is better to be explicit about this where possible.

- Oscillating between `head()` and `tail()` to preview the dataset is unhelpful and confusing.

- While I don't expect a thorough examination of the set of (7 easy, 4 hard, 3 via Hi-C) candidate genes for SLE, some discussion of the biological significance of the detected genes would be appreciated. It would provide a high-level validation of the workflow and link it back to the drug discovery context.

- For the promoter Hi-C section, you could consider using the `linkOverlaps()` method in the `InteractionSet` package, to link SNPs to gene promoters via the identified Hi-C interactions. This might be simpler than the current code, and possibly faster; the `nearest()` step in particular takes quite a long time.

**Is the rationale for developing the new method (or application) clearly explained?**
Yes

**Is the description of the method technically sound?**
Yes

**Are sufficient details provided to allow replication of the method development and its use by others?**
Yes
If any results are presented, are all the source data underlying the results available to ensure full reproducibility?
Partly

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

Referee Expertise: Computational biology, bioinformatics

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

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