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 RPackage gateway.

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\(^1\). 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\(^2\), with up to 40% of these failures due to lack of a clear link between the target and the disease under investigation\(^3\).

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\(^4,5\).

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\(^6\). Similarly, phenome-wide association studies (PheWAS) are emerging as a complementary methodology to decipher the genetic bases of the human phenotype\(^7\). While many of these associations might not actually be relevant for the disease aetiology\(^8\), these methods hold much promise to guide pharmaceutical scientists towards the next generation of drug targets\(^9\).

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\(^10\). 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)\(^11\).

The most established way to map disease-associated regulatory variants to target genes is probably to use expression quantitative trait loci (eQTLs)\(^12\), 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\(^13\).

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\(^14\), Roadmap Epigenomics\(^15\) and BLUEPRINT\(^16,17\) 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\(^18\).

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\(^19\) as well as for CAGE-defined promoters and enhancers\(^16\). Experimental methods to assay interactions between regulatory elements also exist. Chromatin interaction analysis by paired-end tag sequencing (ChIA-PET)\(^20,21\) 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\(^22,23\) 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\(^24\). 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)\(^25\).
Workflow
Overview
In this workflow, we will explore how regulatory genomic data can be used to connect the genetic and transcrip-
tional layers by providing a framework for the functional annotation of SNPs from GWASs. We will use eQTL data from GTEx\(^{14}\), FANTOM5 correlations between promoters and enhancers\(^{21}\) and promoter capture Hi-C data\(^{25}\).

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:

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

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

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

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

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

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

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

```
#       SRR2443263 SRR2443262 SRR2443261 SRR2443260 SRR2443259
# ENSG0000000003.14  19      6     10       10        8
# ENSG00000000005.5  0       0      0        0        0
# ENSG00000000419.12 489     238    224      323       281
# ENSG00000000457.13 594     503    530      670       775
# ENSG00000000460.16 232     173    166      252       268
# ENSG00000000938.12 21554   18918  14260    19869     26586
# ENSG00000000971.15  94      57     45       59        35
# ENSG00000001036.13 500     397    358      407       500
```
We note that genes are annotated using the GENCODE v25 annotation, which will be useful later on. Let's look at the metadata to check how we can split samples between cases and controls:

colData(rse)

# DataFrame with 117 rows and 21 columns
# project sample experiment run
# <character> <character> <character> <character>
# SRR2443263 SRF062966 SRS1048033 SRX1168388 SRR2443263
# SRR2443262 SRF062966 SRS1048034 SRX1168387 SRR2443262
# SRR2443261 SRF062966 SRS1048035 SRX1168386 SRR2443261
# SRR2443260 SRF062966 SRS1048036 SRX1168385 SRR2443260
# SRR2443259 SRF062966 SRS1048037 SRX1168384 SRR2443259
# ... ... ... ...
# SRR2443151 SRF062966 SRS1048145 SRX1168276 SRR2443151
# SRR2443150 SRF062966 SRS1048146 SRX1168275 SRR2443150
# SRR2443148 SRF062966 SRS1048147 SRX1168273 SRR2443148
# SRR2443147 SRF062966 SRS1048148 SRX1168272 SRR2443147
# SRR2443149 SRF062966 SRS1048149 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
| SRR2443263 | FALSE | 103499268 | 5149333280 |
| SRR2443262 | FALSE | 125499809 | 6244059473 |
| SRR2443261 | FALSE | 125043355 | 6201504759 |
| SRR2443260 | FALSE | 104872856 | 5211910530 |
| SRR2443259 | FALSE | 101258496 | 5033612693 |
| ... | ... | ... | ... |
| SRR2443151 | FALSE | 86874384 | 4319264868 |
| SRR2443150 | FALSE | 96316303 | 4787601223 |
| SRR2443148 | FALSE | 120819733 | 6009515064 |
| SRR2443147 | FALSE | 103588909 | 5153702232 |
| SRR2443149 | FALSE | 112640054 | 5598306153 |

<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>biosample_update_date</th>
<th>avg_read_length</th>
<th>geo_accession</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-09-16T01:28:05.297</td>
<td>50</td>
<td>GSM1863749</td>
</tr>
<tr>
<td>2015-09-16T01:28:05.027</td>
<td>50</td>
<td>GSM1863748</td>
</tr>
<tr>
<td>2015-09-16T01:28:04.803</td>
<td>50</td>
<td>GSM1863747</td>
</tr>
<tr>
<td>2015-09-16T01:28:04.587</td>
<td>50</td>
<td>GSM1863746</td>
</tr>
<tr>
<td>2015-09-16T01:28:04.347</td>
<td>50</td>
<td>GSM1863745</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2015-09-16T01:23:41.897</td>
<td>50</td>
<td>GSM1863637</td>
</tr>
<tr>
<td>2015-09-16T01:23:41.453</td>
<td>50</td>
<td>GSM1863636</td>
</tr>
<tr>
<td>2015-09-16T01:23:41.093</td>
<td>50</td>
<td>GSM1863634</td>
</tr>
<tr>
<td>2015-09-16T01:23:40.840</td>
<td>50</td>
<td>GSM1863633</td>
</tr>
<tr>
<td>2015-09-16T01:23:40.597</td>
<td>50</td>
<td>GSM1863635</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bigwig_file</th>
<th>title</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>
The most interesting part of the metadata is contained in the characteristics column, which is a CharacterList object:

```r
colData(rse)$characteristics
```

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

```r
# disease status
colData(rse)$disease_status <- sapply(colData(rse)$characteristics, ",1")
colData(rse)$disease_status <- sub("disease status: ", ",", colData(rse)$disease_status)
colData(rse)$disease_status <- sub("systemic lupus erythematosus (SLE)\n", "SLE", colData(rse)$disease_status)
colData(rse)$disease_status <- factor(colData(rse)$disease_status, levels = c("healthy", "SLE"))
# tissue
colData(rse)$tissue <- sapply(colData(rse)$characteristics, ",2")
colData(rse)$tissue <- sub("tissue: ", ",", colData(rse)$tissue)
colData(rse)$tissue <- factor(colData(rse)$tissue)
# anti-ro
colData(rse)$anti_ro <- sapply(colData(rse)$characteristics, ",3")
colData(rse)$anti_ro <- sub("anti-ro: ", ",", colData(rse)$anti_ro)
colData(rse)$anti_ro <- factor(colData(rse)$anti_ro)
# ism
colData(rse)$ism <- sapply(colData(rse)$characteristics, ",4")
colData(rse)$ism <- sub("ism: ", ",", colData(rse)$ism)
colData(rse)$ism <- factor(colData(rse)$ism)
```

We can have a look at the new format:

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

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

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

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

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

```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 ...
## colnames(36): SRR2443263 SRR2443262 ... SRR2443166 SRR2443165
## colData names(27): project sample ... sizeFactor replaceable
```

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

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

```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)]
```

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

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

```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:

\[
\text{res} < - \text{results}(\text{dds, alpha = 0.05})
\]

```
res
```

## 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
## ENSG00000000419.12 389.9025130     0.66288230 0.11427371   5.80082925
## ENSG00000000457.13  636.6928414     0.1736365  0.08062862  2.15015047
## ENSG00000000460.16 234.6479796     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  1.17583692
## ENSG00000283698.1   0.2289885    0.021962044  1.1315739  0.019408404
## ENSG00000283699.1   0.5398951   -0.003056215  0.7578201 -0.004032903
##                          pvalue         padj
##                       <numeric>    <numeric>
## ENSG00000000003.14  4.200613e-01  6.70302e-01
## ENSG00000000005.5  9.910354e-01           NA
## ENSG00000000419.12  6.598777e-09           NA
## ENSG00000000457.13  3.154331e-02  1.463634e-01
## ENSG00000000460.16 5.398951e-06  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:

```
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).

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

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

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

```
#                      baseMean log2FoldChange      lfcSE      stat
# ENSG00000000419.12  389.90251      0.6628823 0.11427371  5.800829
# ENSG00000000460.16  234.64798      0.2058940 0.07445624  2.765303
# ENSG000000002549.12 1970.95648      0.8657769 0.25181202  3.438187
# ENSG000000003096.13  11.18475     -0.7894018 0.25613621 -3.081961
# ENSG000000003147.17   71.79432      0.6113739 0.15162606  4.032116
# ENSG000000003249.13  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
# ENSG000000002549.12 5.856225e-04 9.776328e-03
# ENSG000000003096.13 1.184758e+00 2.291728e-02
# ENSG000000003147.17 5.527679e-05 1.927054e-03
# ENSG000000003249.13 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:

```r
rowData(rse)
```

```
# DataFrame with 58037 rows and 3 columns
# gene_id bp_length symbol
# <character> <integer> <CharacterList>
# 1 ENSG0000000003.14  4535     TSPAN6
# 2 ENSG0000000005.5   1610      TNMD
# 3 ENSG00000000419.12 1207      DPM1
# 4 ENSG00000000457.13  6883     SCYL3
# 5 ENSG00000000460.16  5967     Clorf112
```
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

```r
#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")
```

```r
snps <- read.delim("gwas_catalog_v1.0.1-associations_e90_r2017-12-04.tsv",
check.names = FALSE, stringsAsFactors = FALSE)
```

```r
genome(snps) <- "GRCh38"
```

```r
snps
```

# gwasloc instance with 61107 records and 37 attributes per record.
# Extracted: 2017-12-04
# Genome: GRCh38
# Excerpt:
# GRanges object with 5 ranges and 3 metadata columns:
# seqnames ranges strand | DISEASE/TRAIT SNPS
# <Rle> <IRanges> <Rle> | <character> <character>
# [1] chr1 [203186754, 203186754] * | YKL-40 levels rs4950928
<table>
<thead>
<tr>
<th></th>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
<th>DISEASE/TRAIT</th>
<th>SNPS</th>
<th>P-VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>chr1</td>
<td>[159711078, 159711078]</td>
<td>*</td>
<td>Psoriasis</td>
<td>rs7993214</td>
<td>1e-13</td>
</tr>
</tbody>
</table>

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

step and refer the reader to the Ensembl REST API\(^\text{43}\), the Ensembl Linkage Disequilibrium Calculator and the Bioconductor packages trio\(^\text{44}\) and ldblock\(^\text{45}\) 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\(^\text{46}\) package and the GENCODE v25 gene annotation:

```r
library(GenomicFeatures)
# uncomment the following line to download file
taxdb <- makeTxDbFromGFF("gencode.v25.annotation.gff3.gz")
taxdb <- keepStandardChromosomes(txdb)
taxdb
```

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

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

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

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

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 QUERIYID TXID CDSID GENEID
## <Rle> <IRanges> <Rle> | <factor> <integer> <integer> <character> <character> <CharacterList> <character>
## [1] chr16 [31301932, 31301932] + | intron 40161
## [5] chr8 [11491677, 11491677] * | intergenic <NA>
## ... ... ... ... ... ... ...
## [295] chr6 [137874014, 137874014] + | intron 6162
## [296] chr6 [32619077, 32619077] * | intergenic <NA>
## [297] chr6 [137685367, 137685367] + | intron 11552
## [298] chrX [153924366, 153924366] - | intron 1770
## [299] chr5 [160459613, 160459613] * | intergenic <NA>
## ... ...
## [477] chr16 [31301932, 31301932] + | intron 40161
## [478] chr11 [589564, 589564] + | intron 12531
## [479] chr3 [58384450, 58384450] + | intron 51074
## [480] chr1 [173340574, 173340574] * | intergenic <NA>
## [481] chr8 [11491677, 11491677] * | intergenic <NA>
```
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(sns_anno) <- cbind(mcols(snps_metadata)[c("SNPS", "P-VALUE")],
mcols(snps_anno))
```

## GRanges object with 299 ranges and 11 metadata columns:

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges strand</th>
<th>SNPS</th>
<th>P.VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr16</td>
<td>[31301932, 31301932]</td>
<td>+</td>
<td>rs9888739</td>
</tr>
<tr>
<td>chr11</td>
<td>[589564, 589564]</td>
<td>+</td>
<td>rs4963128</td>
</tr>
<tr>
<td>chr3</td>
<td>[58384450, 58384450]</td>
<td>+</td>
<td>rs6445975</td>
</tr>
<tr>
<td>chr1</td>
<td>[173340574, 173340574]</td>
<td>*</td>
<td>rs10798269</td>
</tr>
<tr>
<td>chr8</td>
<td>[11491677, 11491677]</td>
<td>*</td>
<td>rs13277113</td>
</tr>
<tr>
<td>chr6</td>
<td>[137874014, 137874014]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>chr6</td>
<td>[32619077, 32619077]</td>
<td>*</td>
<td>rs9271366</td>
</tr>
<tr>
<td>chr6</td>
<td>[137685367, 137685367]</td>
<td>+</td>
<td>rs6920220</td>
</tr>
<tr>
<td>chrX</td>
<td>[153924366, 153924366]</td>
<td>-</td>
<td>rs2269368</td>
</tr>
<tr>
<td>chr5</td>
<td>[160459613, 160459613]</td>
<td>*</td>
<td>rs2431099</td>
</tr>
<tr>
<td>LOCATION</td>
<td>LOCSTART</td>
<td>LOCEND</td>
<td>QUERYID</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>intron</td>
<td>40161</td>
<td>40161</td>
<td>1</td>
</tr>
<tr>
<td>intron</td>
<td>12531</td>
<td>12531</td>
<td>2</td>
</tr>
<tr>
<td>intron</td>
<td>51074</td>
<td>51074</td>
<td>3</td>
</tr>
<tr>
<td>intergenic</td>
<td>&lt;NA&gt;</td>
<td>&lt;NA&gt;</td>
<td>4</td>
</tr>
<tr>
<td>intergenic</td>
<td>&lt;NA&gt;</td>
<td>&lt;NA&gt;</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>intron</td>
<td>6162</td>
<td>6162</td>
<td>393</td>
</tr>
<tr>
<td>intergenic</td>
<td>&lt;NA&gt;</td>
<td>&lt;NA&gt;</td>
<td>397</td>
</tr>
<tr>
<td>intron</td>
<td>11552</td>
<td>11552</td>
<td>398</td>
</tr>
<tr>
<td>intron</td>
<td>1770</td>
<td>1770</td>
<td>399</td>
</tr>
<tr>
<td>intergenic</td>
<td>&lt;NA&gt;</td>
<td>&lt;NA&gt;</td>
<td>400</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GENEID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG00000169896.16</td>
</tr>
<tr>
<td>ENSG00000070047.11</td>
</tr>
<tr>
<td>ENSG00000168297.15</td>
</tr>
<tr>
<td>&lt;NA&gt;</td>
</tr>
<tr>
<td>&lt;NA&gt;</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>ENSG000001118503.14</td>
</tr>
<tr>
<td>&lt;NA&gt;</td>
</tr>
<tr>
<td>ENSG00000230533.2</td>
</tr>
<tr>
<td>ENSG00000089820.15</td>
</tr>
<tr>
<td>&lt;NA&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PRECEDEID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG00000076321.10,ENSG000000117592.8,ENSG000000117593.9,...</td>
</tr>
<tr>
<td>ENSG00000079459.12,ENSG000000136573.12,ENSG000000136574.17,...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>ENSG000000118322.12,ENSG000000145864.12,ENSG000000253417.5,...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FOLLOWID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG00000076321.10,ENSG000000117592.8,ENSG000000117593.9,...</td>
</tr>
<tr>
<td>ENSG00000079459.12,ENSG000000136573.12,ENSG000000136574.17,...</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

seqinfo: 23 sequences from GRCh38 genome; no seqlengths

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

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

As expected\(^1\), 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")
snps_easy <- as.data.frame(snps_easy)
head(snps_easy)
```

<table>
<thead>
<tr>
<th>seqnames</th>
<th>start</th>
<th>end</th>
<th>width</th>
<th>strand</th>
<th>SNPS</th>
<th>P.VALUE</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr4</td>
<td>101829919</td>
<td>101829919</td>
<td>1</td>
<td>+</td>
<td>rs10516487</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>

\(^1\) As expected, the great majority of SNPs are located within introns and in intergenic regions. For the moment, we will focus on SNPs that are either coding or in promoter and UTR regions, as these can be assigned to target genes rather unambiguously.
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)
snps_easy_in_degs
```

```r
## DataFrame with 7 rows and 24 columns
##                            gene_id bp_length symbol   baseMean  log2FoldChange  lfcSE  stat   pvalue  padj  seqnames        start       end      width  strand        SNPS   P.VALUE LOCATION LOCSTART LOCEND  QUERYID  TXID  CDSID
##                        <character> <integer> <list>  <numeric>          <numeric> <numeric> <numeric> <numeric> <numeric> <factor> <integer> <integer> <factor> <character> <numeric> <factor> <integer> <integer> <integer> <integer> <character> <character> <numeric> <list>
## ENSG00000096968 ENSG00000096968.13      6170    JAK2 1279.47795      0.4854343 0.1553513  3.124753 1.779545e-03 2.068794e-02  chr9  4984530  4984530         1        +   rs1887428     1e-06 fiveUTR 141
## ENSG00000099834 ENSG00000099834.18      3873   CDHR5  10.20177      0.8539586 0.2666557  3.202476 1.362516e-03 1.732902e-02  chr11    625085  625085         1        -  rs58688157     5e-13 promoter NA
## ENSG00000115267 ENSG00000115267.5      4528 IFIH1 1415.91330      0.7819504 0.1541707  5.071977 2.544247e-05 1.120363e-08  chr2 162267541 162267541         1        -   rs1990760     4e-08 coding 162267541
## ENSG00000120280 ENSG00000120280.5      1855 Cxorf21  637.78094      0.7106270 0.2992536  4.699260 2.611057e-06 6.047898e-09  chrX  30559729 30559729         1        -    rs887369     5e-10 coding 30559729
## ENSG00000185507 ENSG00000185507.19      2628   IRF7 4883.20891      1.4062704 0.2992536  4.699260 2.611057e-06 2.298336e-07  chr11  614318  614318         1        -   rs1061502     9e-11 coding 614318
## ENSG00000204366 ENSG00000204366.3      1875  ZBTB12  22.99200      -0.3892298 0.1348705 -2.885952 3.902318e-03 3.584479e-09  chr6  31902549 31902549         1        -    rs1887428     1e-06 coding 31902549
## ENSG00000275106 ENSG00000275106.1       790   NA    10.32171      0.7344844 0.2305300  3.186068 1.442206e-03 3.179861e-09  chr7 128954129 128954129         1        -   rs10488631     2e-11 promoter NA
```
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"))
```

Use of regulatory genomic data to map intronic and intergenic SNPs to target genes
##     [3]     chr3 [ 58384450,  58384450]      + |   rs6445975     7e-09
##     [4]     chr1 [173340574, 173340574]      * |  rs10798269     1e-07
##     [5]     chr8 [ 11491677, 11491677]      * |  rs13277113     1e-10
##     ...                    ...                    ...    ...
##   [246]     chr6 [137874014, 137874014]      + |   rs5029937     5e-13
##   [247]     chr6 [ 32619077,  32619077]      * |   rs9271366     1e-07
##   [248]     chr6 [137685367, 137685367]      + |   rs6920220     4e-07
##   [249]     chrX [153924366, 153924366]      - |   rs2269368     8e-07
##   [250]     chr5 [160459613, 160459613]      * |   rs2431099     2e-06
## LOCATION
##     <factor>
##     [1]     intron
##     [2]     intron
##     [3]     intron
##     [4]     intergenic
##     [5]     intergenic
##     ...                    ...
##   [246]     intron
##   [247]     intergenic
##   [248]     intron
##   [249]     intron
##   [250]     intergenic
## seqinfo: 23 sequences from GRCh38 genome; no seqlengths

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\(^6\). We will use blood eQTL data from the GTEx consortium\(^7\). 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:

```
# 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)
```

```
# variant_id  gene_id       tss_distance  ma_samples  ma_count
# 1 1_231153_CTT_C_b37 ENSG00000223972.4       219284         13       13
# 2 1_61920_G_A_b37 ENSG00000238009.2        -67303          18       20
# 3 1_64649_A_C_b37 ENSG00000238009.2        -64574          16       16
# 4 1_115746_C_T_b37 ENSG00000238009.2        -13477          45       45
# 5 1_135203_G_A_b37 ENSG00000238009.2         5980           51       51
# 6 1_988016_T_C_b37 ENSG00000268903.1       852121          21       23

# maf pval_nominal         slope   slope_se pval_nominal_threshold
# 1 0.0191740  3.69025e-08 1.319720 0.233538            1.35366e-04
# 2 0.0281690  7.00836e-07 0.903786 0.178322            8.26088e-05
# 3 0.0220386  5.72066e-07 1.110040 0.217225            8.26088e-05
# 4 0.0628492  6.50297e-10 0.858203 0.134436            8.26088e-05
# 5 0.0698630  6.67194e-10 0.811790 0.127255            8.26088e-05
# 6 0.0318560  6.35694e-05 0.501916 0.123743            8.52870e-05

# min_pval_nominal     pval_beta
# 1 3.69025e-08 4.67848e-05
# 2 6.50297e-10 1.11312e-06
# 3 6.50297e-10 1.11312e-06
# 4 6.50297e-10 1.11312e-06
# 5 6.50297e-10 1.11312e-06
# 6 6.35694e-05 5.44487e-02
```
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)
```

.userAssistant:

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

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

We can then convert the data.frame into a GRanges object:
We also need to ensure that the chromosome notation is consistent with the previous objects:

```r
seqlevelsStyle(gtex_blood)
```

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

```r
glevels(gtex_blood)
```

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

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

```r
seqlevels(gtex_blood)
```

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

```r
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:

```r
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
##          variant_id            gene_id tss_distance ma_samples ma_count
## 1  _11_589564_T_C_b37 ENSG00000177042.10      -105969        212      250
## 2  _3_58370177_G_T_b37  ENSG00000168291.8       -49407        205      250
## 3   _3_8_11349186_G_A_b37 ENSG00000154319.10       16962        157      180
## 4   _4_8_11349186_G_A_b37 ENSG00000136573.8       -2324        157      180
## 5   _5_8_11349186_G_A_b37 ENSG00000255518.1       -66284        157      180
## 6   _6_8_11349186_G_A_b37 ENSG00000255354.1       -68343        157      180
##          maf pval_nominal     slope  slope_se pval_nominal_threshold
## 1 0.339674  4.51059e-10 -0.194589 0.0301828            3.35947e-05
## 2 0.338753  2.05231e-12  0.179408 0.0244587            6.23219e-05
## 3 0.243902  6.46308e-27  0.778785 0.0656311            3.79430e-05
## 4 0.243902  7.37464e-07 -0.281643 0.0305280            3.75653e-05
## 5 0.243902  8.41301e-08 -0.243121 0.0442629            3.66297e-05
## 6 0.243902  8.41301e-08 -0.243121 0.0442629            3.66297e-05
##          min_pval_nominal   pval_beta
## 1 5.23982e-30 1.63019e-24
## 2 3.39499e-13 3.97374e-09
## 3 8.46904e-29 2.22416e-23
## 4 2.97871e-19 2.22082e-14
## 5 8.28459e-08 4.81268e-04
## 6 2.67616e-08 1.37119e-04
```

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:

```
degs$ensembl_id <- sub("(ENSG\[0-9\]+)\.[0-9]+", "\1", degs$gene_id)
```

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

## DataFrame with 6 rows and 30 columns

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

<table>
<thead>
<tr>
<th>SNPS</th>
<th>P.VALUE</th>
<th>LOCATION</th>
<th>variant_id</th>
<th>gene_id.x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5e-06</td>
<td>intergenic</td>
<td>19_18481333_A_G_b37</td>
<td>ENSG00000130513.6</td>
</tr>
<tr>
<td>2</td>
<td>6e-15</td>
<td>intron</td>
<td>15_75311036_C_A_b37</td>
<td>ENSG00000140497.12</td>
</tr>
<tr>
<td>3</td>
<td>1e-20</td>
<td>intron</td>
<td>11_71187679_C_T_b37</td>
<td>ENSG00000172890.7</td>
</tr>
<tr>
<td>4</td>
<td>8e-08</td>
<td>intron</td>
<td>6_31636742_A_G_b37</td>
<td>ENSG00000214894.2</td>
</tr>
<tr>
<td>5</td>
<td>6e-92</td>
<td>intergenic</td>
<td>6_30940989_T_C_b37</td>
<td>ENSG00000214894.2</td>
</tr>
<tr>
<td>6</td>
<td>2e-52</td>
<td>intron</td>
<td>6_31721033_G_A_b37</td>
<td>ENSG00000214894.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>tss_distance</th>
<th>ma_samples</th>
<th>ma_count</th>
<th>maf</th>
<th>pval_nominal</th>
<th>slope</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-4208</td>
<td>166</td>
<td>189</td>
<td>0.2560980</td>
<td>0.350964</td>
</tr>
<tr>
<td>2</td>
<td>145330</td>
<td>170</td>
<td>191</td>
<td>0.2588080</td>
<td>-0.107460</td>
</tr>
<tr>
<td>3</td>
<td>23524</td>
<td>183</td>
<td>231</td>
<td>0.3130080</td>
<td>0.407266</td>
</tr>
<tr>
<td>4</td>
<td>83806</td>
<td>49</td>
<td>54</td>
<td>0.0731707</td>
<td>0.479659</td>
</tr>
<tr>
<td>5</td>
<td>142553</td>
<td>83</td>
<td>91</td>
<td>0.1230300</td>
<td>0.453255</td>
</tr>
<tr>
<td>6</td>
<td>922597</td>
<td>50</td>
<td>55</td>
<td>0.0745257</td>
<td>0.479935</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>lfcSE</th>
<th>stat</th>
<th>pvalue</th>
<th>padj</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.28347645</td>
<td>2.781079</td>
<td>5.417861e-03</td>
<td>0.0448154406</td>
</tr>
<tr>
<td>2</td>
<td>0.08814542</td>
<td>3.385012</td>
<td>2.44784e-27</td>
<td>1.10743e-22</td>
</tr>
<tr>
<td>3</td>
<td>0.0310305</td>
<td>4.467199</td>
<td>1.05596e-33</td>
<td>7.87659e-28</td>
</tr>
<tr>
<td>4</td>
<td>0.0846154</td>
<td>6.02220e-05</td>
<td>3.17673e-13</td>
<td>1.77790e-08</td>
</tr>
<tr>
<td>5</td>
<td>0.0670210</td>
<td>6.02220e-05</td>
<td>3.17673e-13</td>
<td>1.77790e-08</td>
</tr>
<tr>
<td>6</td>
<td>0.0840440</td>
<td>6.02220e-05</td>
<td>3.17673e-13</td>
<td>1.77790e-08</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>gene_id.y</th>
<th>bp_length</th>
<th>symbol</th>
<th>baseMean</th>
<th>log2FoldChange</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ENSG00000130513.6</td>
<td>2087</td>
<td>GDF15</td>
<td>6.75448</td>
<td>0.7883703</td>
</tr>
<tr>
<td>2</td>
<td>EN3G00000140497.16</td>
<td>5000</td>
<td>SCAMP2</td>
<td>4383.03109</td>
<td>-0.2959934</td>
</tr>
<tr>
<td>3</td>
<td>ENSG00000172890.11</td>
<td>16263</td>
<td>NADSYN1</td>
<td>4020.56224</td>
<td>0.2619770</td>
</tr>
<tr>
<td>4</td>
<td>ENSG00000214894.6</td>
<td>2171</td>
<td>LINCO0243</td>
<td>74.95034</td>
<td>1.2684089</td>
</tr>
<tr>
<td>5</td>
<td>ENSG00000214894.6</td>
<td>2171</td>
<td>LINCO0243</td>
<td>74.95034</td>
<td>1.2684089</td>
</tr>
<tr>
<td>6</td>
<td>ENSG00000214894.6</td>
<td>2171</td>
<td>LINCO0243</td>
<td>74.95034</td>
<td>1.2684089</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>lfcSE</th>
<th>stat</th>
<th>pvalue</th>
<th>padj</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.28347645</td>
<td>2.781079</td>
<td>5.417861e-03</td>
<td>0.0448154406</td>
</tr>
<tr>
<td>2</td>
<td>0.08814542</td>
<td>3.385012</td>
<td>2.44784e-27</td>
<td>1.10743e-22</td>
</tr>
<tr>
<td>3</td>
<td>0.0310305</td>
<td>4.467199</td>
<td>1.05596e-33</td>
<td>7.87659e-28</td>
</tr>
<tr>
<td>4</td>
<td>0.0846154</td>
<td>6.02220e-05</td>
<td>3.17673e-13</td>
<td>1.77790e-08</td>
</tr>
<tr>
<td>5</td>
<td>0.0670210</td>
<td>6.02220e-05</td>
<td>3.17673e-13</td>
<td>1.77790e-08</td>
</tr>
<tr>
<td>6</td>
<td>0.0840440</td>
<td>6.02220e-05</td>
<td>3.17673e-13</td>
<td>1.77790e-08</td>
</tr>
</tbody>
</table>
We can add these 4 genes to our list:

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

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

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

### FANTOM5 data. The FANTOM consortium profiled gene expression across a large panel of tissues and cell types using CAGE[6,21]. 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:
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"))
```

```r
head(fantom)
```

<table>
<thead>
<tr>
<th>chrom</th>
<th>chromStart</th>
<th>chromEnd</th>
<th>score</th>
<th>strand</th>
<th>thickStart</th>
<th>thickEnd</th>
<th>itemRgb</th>
<th>blockCount</th>
<th>blockSizes</th>
<th>chromStarts</th>
<th>name_1</th>
<th>name_2</th>
<th>score</th>
<th>strand</th>
<th>thickStart</th>
<th>thickEnd</th>
<th>itemRgb</th>
<th>blockCount</th>
<th>blockSizes</th>
<th>chromStarts</th>
<th>name_1</th>
<th>name_2</th>
<th>score</th>
<th>strand</th>
<th>thickStart</th>
<th>thickEnd</th>
<th>itemRgb</th>
<th>blockCount</th>
<th>blockSizes</th>
<th>chromStarts</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr1</td>
<td>858252</td>
<td>861621</td>
<td>404</td>
<td>.</td>
<td>858452</td>
<td>858453</td>
<td>0,0,0</td>
<td>2</td>
<td>401,1001</td>
<td>chr1:858256-858648;NM_152486;SAMD11;R:0.404;FDR:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chr1</td>
<td>894178</td>
<td>956688</td>
<td>202</td>
<td>.</td>
<td>956687</td>
<td>956688</td>
<td>0,0,0</td>
<td>2</td>
<td>1001,401</td>
<td>chr1:956563-956812;NM_015658;NOC2L;R:0.202;FDR:8.01154668254404e-08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chr1</td>
<td>901376</td>
<td>917362</td>
<td>422</td>
<td>.</td>
<td>901376</td>
<td>956688</td>
<td>0,0,0</td>
<td>2</td>
<td>1001,401</td>
<td>chr1:1173386-1173736;NM_001160184,NM_032129;PLEKHN1;R:0.422;FDR:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chr1</td>
<td>935051</td>
<td>942164</td>
<td>187</td>
<td>.</td>
<td>935051</td>
<td>942164</td>
<td>0,0,0</td>
<td>2</td>
<td>1001,401</td>
<td>chr1:941791-942135;NM_001142467,NM_021170;HES4;R:0.187;FDR:6.32949888009368e-07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chr1</td>
<td>935051</td>
<td>1005621</td>
<td>236</td>
<td>.</td>
<td>935051</td>
<td>1005621</td>
<td>0,0,0</td>
<td>2</td>
<td>1001,401</td>
<td>chr1:1005293-1005547,NM_001160184,NM_021170;HES4;R:0.236;FDR:6.28221217150423e-11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Now we can extract the genomic locations of the enhancers and the correlation values:

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

```
##   X.chrom chromStart chromEnd score strand thickStart thickEnd itemRgb
## 1    chr1     858252   861621   404      .     858452   858453   0,0,0
## 2    chr1     894178   956888   202      .     956687   956688   0,0,0
## 3    chr1     901376   956888   422      .     956687   956688   0,0,0
## 4    chr1     901376  1173762   311      .    1173561  1173562   0,0,0
## 5    chr1     935051   942164   187      .     941963   941964   0,0,0
## 6    chr1     935051  1005621   236      .    1005420  1005421   0,0,0
##   blockCount blockSizes chromStarts               name_1
## 1          2   401,1001      0,2368   chr1:858256-858648
## 2          2   1001,401     0,62309   chr1:956563-956812
## 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  chr
## 1              NM_152486  SAMD11 R:0.404                    FDR:0 chr1
## 2              NM_015658   NOC2L R:0.202 FDR:8.01154668254404e-08 chr1
## 3 NM_001160184,NM_032129 PLEKHN1 R:0.422                    FDR:0 chr1
## 4 NM_001160184,NM_032129 PLEKHN1 R:0.311                    FDR:0 chr1
## 5 NM_001142467,NM_021170    HES4 R:0.187 6.3294988809368e-07 chr1
## 6 NM_001142467,NM_021170    HES4 R:0.236 FDR:6.28221217150423e-11 chr1
##     start     end  symbol  corr                  fdr
## 1  858256  858648  SAMD11 0.404                    0
## 2  956563  956812   NOC2L 0.202 8.01154668254404e-08
## 3  956563  956812 PLEKHN1 0.422                    0
## 4 1173386 1173736 PLEKHN1 0.311                    0
## 5  941971  942135    HES4 0.187 6.3294988809368e-07
## 6 1005293 1005547    HES4 0.236 6.28221217150423e-11
```

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

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

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

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

```r
fantom
## 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\(^\text{19}\), so we will have to uplift the GRCh37 coordinates to GRCh38:

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

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

We can now compute the overlap between SNPs and enhancers:

```r
hits <- findOverlaps(snps_hard, fantom)
snps_hard_in_fantom = snps_hard[queryHits(hits)]
fantom_with_snps_hard = fantom[subjectHits(hits)]

mcols(snps_hard_in_fantom) <- cbind(mcols(snps_hard_in_fantom), mcols(fantom_with_snps_hard))
snps_hard_in_fantom <- as.data.frame(snps_hard_in_fantom)
```

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

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

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

This will combine the SNPs with the gene expression data, allowing us to identify differentially expressed genes.
We have identified 2 genes whose putative enhancers contain SLE GWAS SNPs. Let’s add these to our list:

```r
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>1e-06</td>
<td>fiveUTR</td>
<td>ENSG00000096968.13</td>
<td>JAK2</td>
<td>2.068794e-02</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>1.732902e-02</td>
<td>0.8539586</td>
</tr>
<tr>
<td>ENSG00000115267</td>
<td>4e-08</td>
<td>coding</td>
<td>ENSG00000115267.75</td>
<td>IFIH1</td>
<td>1.120363e-03</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>6.047898e-05</td>
<td>0.7819504</td>
</tr>
<tr>
<td>ENSG00000185507</td>
<td>9e-11</td>
<td>coding</td>
<td>ENSG00000185507.19</td>
<td>IRF7</td>
<td>2.298336e-04</td>
<td>1.4062704</td>
</tr>
<tr>
<td>ENSG00000204366</td>
<td>8e-21</td>
<td>promoter</td>
<td>ENSG00000204366.3</td>
<td>GDF15</td>
<td>4.481544e-02</td>
<td>0.7883703</td>
</tr>
</tbody>
</table>

Promoter Capture Hi-C data. More recently, chromatin interaction data was generated across 17 human primary blood cell types\(^2\). 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\(^4\) and can be accessed in the supplementary data of the paper:

```r
# 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)
```
## baitChr  baitSt  baitEnd  baitID  oeChr  oeSt  oeEnd  oeID  
## 1    chr1 1206873 1212438    254  chr1  943676  957199  228  
## 2    chr1 1206873 1212438    254  chr1 1034268 1040208  235  
## 3    chr1 1206873 1212438    254  chr1 1040208 1043143  236  
## 4    chr1 1206873 1212438    254  chr1 1069045 1083958  242  
## 5    chr1 1206873 1212438    254  chr1 1083958 1091234  243  
## 6    chr1 1206873 1212438    254  chr1 1585571 1619752  304  

## cellType.s.  
## 1          nCD8  
## 2 nCD4,nCD8,Mac0,Mac1,Mac2,Mon  
## 3     nCD4,nCD8,Mac0,Mac1,Mac2,MK  
## 4                            nCD8  
## 5                            nCD8  
## 6                             Neu  

## sample.s.  
## 1        C0066PH1  
## 2 S007DDH2,S007G7H4,C0066PH1,S00390H1,S001MJH1,S001S7H2,S0022IH2,S00622H1,S00B54H1,S004BTH2,C000S5H2  
## 3 S007DDH2,S007G7H4,C0066PH1,S00390H1,S001MJH1,S001S7H2,S0022IH2,S00622H1,S00B54H1,S004BTH2  
## 4        C0066PH1,S00C2FH1  
## 5        C0066PH1,S00C2FH1  
## 6        C000S5H1  

In this case, we will have to map the promoter baits to genes first. We can do this by converting the baits to a GRanges object and then using the TxDb object we previously built to extract positions of transcription start sites (TSSs):

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

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

# Find nearest TSS for each bait
hits <- nearest(baits, tss)

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

## GRanges object with 51142 ranges and 1 metadata column:  
## seqnames ranges strand | gene_id
## <Rle> | <IRanges> | <Rle> | <character>
## [1] chr1 [1206873, 1212438]  * | ENSG00000186827.10  
## [2] chr1 [1206873, 1212438]  * | ENSG00000186827.10  
## [3] chr1 [1206873, 1212438]  * | ENSG00000186827.10  
## [4] chr1 [1206873, 1212438]  * | ENSG00000186827.10  
## [5] chr1 [1206873, 1212438]  * | ENSG00000186827.10  
## ... ... ... ... ...  
## [51138] chrY [22732049, 22743996]  * | ENSG00000230727.1  
## [51139] chrY [22732049, 22743996]  * | ENSG00000230727.1  
## [51140] chrY [22732049, 22743996]  * | ENSG00000230727.1  
## [51141] chrY [22732049, 22743996]  * | ENSG00000230727.1  
## [51142] chrY [22732049, 22743996]  * | ENSG00000230727.1  
```
Now we can create a GRanges object of the enhancers in the promoter capture Hi-C data with the bait annotation attached:

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

## GRanges object with 25232 ranges and 1 metadata column:

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
<th>gene_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr1</td>
<td>[943676, 957199]</td>
<td>*</td>
<td>ENSG00000186827.10</td>
</tr>
<tr>
<td>chr1</td>
<td>[1034268, 1040208]</td>
<td>*</td>
<td>ENSG00000186827.10</td>
</tr>
<tr>
<td>chr1</td>
<td>[1040208, 1043143]</td>
<td>*</td>
<td>ENSG00000186827.10</td>
</tr>
<tr>
<td>chr1</td>
<td>[1069045, 1083958]</td>
<td>*</td>
<td>ENSG00000186827.10</td>
</tr>
<tr>
<td>chr1</td>
<td>[1083958, 1091234]</td>
<td>*</td>
<td>ENSG00000186827.10</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>chrY</td>
<td>[23401616, 23404873]</td>
<td>*</td>
<td>ENSG00000230727.1</td>
</tr>
<tr>
<td>chrY</td>
<td>[23404938, 23407193]</td>
<td>*</td>
<td>ENSG00000230727.1</td>
</tr>
<tr>
<td>chrY</td>
<td>[23409014, 23410287]</td>
<td>*</td>
<td>ENSG00000230727.1</td>
</tr>
<tr>
<td>chrY</td>
<td>[23410287, 23411837]</td>
<td>*</td>
<td>ENSG00000230727.1</td>
</tr>
<tr>
<td>chrY</td>
<td>[23411837, 23412539]</td>
<td>*</td>
<td>ENSG00000230727.1</td>
</tr>
</tbody>
</table>

## seqinfo: 24 sequences from an unspecified genome; no seqlengths

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

```r
hits <- findOverlaps(snps_hard, pchic)
snps_hard_in_pchic = snps_hard[queryHits(hits)]
pchic_with_snps_hard = pchic[subjectHits(hits)]
mcols(snps_hard_in_pchic) <- cbind(mcols(snps_hard_in_pchic), mcols(pchic_with_snps_hard))
snps_hard_in_pchic <- as.data.frame(snps_hard_in_pchic)
```

## seqnames  start  width  strand  SNPS  P.VALUE
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>chr6</td>
<td>31753256</td>
<td>1</td>
<td>+</td>
<td>rs3131379</td>
<td>2e-52</td>
</tr>
<tr>
<td>chr6</td>
<td>32696681</td>
<td>1</td>
<td>*</td>
<td>rs2647012</td>
<td>8e-06</td>
</tr>
<tr>
<td>chr16</td>
<td>30631546</td>
<td>1</td>
<td>*</td>
<td>rs7197475</td>
<td>3e-08</td>
</tr>
<tr>
<td>chr20</td>
<td>4762059</td>
<td>1</td>
<td>*</td>
<td>rs6084875</td>
<td>2e-06</td>
</tr>
<tr>
<td>chr6</td>
<td>32689659</td>
<td>1</td>
<td>*</td>
<td>rs3129716</td>
<td>4e-09</td>
</tr>
<tr>
<td>chr6</td>
<td>31668965</td>
<td>1</td>
<td>+</td>
<td>rs9267531</td>
<td>8e-08</td>
</tr>
<tr>
<td>chr6</td>
<td>31951083</td>
<td>1</td>
<td>+</td>
<td>rs1270942</td>
<td>2e-165</td>
</tr>
<tr>
<td>chr6</td>
<td>106140931</td>
<td>1</td>
<td>-</td>
<td>rs656843</td>
<td>5e-14</td>
</tr>
<tr>
<td>chr7</td>
<td>28146272</td>
<td>1</td>
<td>-</td>
<td>rs849142</td>
<td>9e-11</td>
</tr>
<tr>
<td>chr2</td>
<td>65381229</td>
<td>1</td>
<td>-</td>
<td>rs268134</td>
<td>1e-10</td>
</tr>
<tr>
<td>chr17</td>
<td>39850937</td>
<td>1</td>
<td>-</td>
<td>rs14312327</td>
<td>6e-09</td>
</tr>
<tr>
<td>chr9</td>
<td>86916761</td>
<td>1</td>
<td>*</td>
<td>rs19002901</td>
<td>3e-06</td>
</tr>
<tr>
<td>chr11</td>
<td>65637829</td>
<td>1</td>
<td>*</td>
<td>rs931127</td>
<td>7e-06</td>
</tr>
<tr>
<td>chr19</td>
<td>18370523</td>
<td>1</td>
<td>*</td>
<td>rs8105429</td>
<td>5e-06</td>
</tr>
<tr>
<td>chr16</td>
<td>85977731</td>
<td>1</td>
<td>*</td>
<td>rs10521318</td>
<td>4e-06</td>
</tr>
<tr>
<td>chr5</td>
<td>39406395</td>
<td>1</td>
<td>-</td>
<td>rs3914167</td>
<td>8e-06</td>
</tr>
<tr>
<td>chr16</td>
<td>31315385</td>
<td>1</td>
<td>+</td>
<td>rs11860650</td>
<td>2e-20</td>
</tr>
</tbody>
</table>

## LOCATION  gene_id
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>intron</td>
<td>ENSG00000219797.2</td>
</tr>
<tr>
<td>intergenic</td>
<td>ENSG00000204290.10</td>
</tr>
<tr>
<td>intergenic</td>
<td>ENSG00000180096.11</td>
</tr>
<tr>
<td>intergenic</td>
<td>ENSG00000212536.1</td>
</tr>
</tbody>
</table>
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)

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 and enhancer - promoter relationships from CAGE and promoter capture Hi-C experiments.

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

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

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

License: CC-BY 4.0

Competing interests
EF is a full time employee of GSK.

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

References


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(22) 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.

---

**Author Response 14 Feb 2018**

**Enrico Ferrero**, GlaxoSmithKline, UK

Vincent, many thanks for reviewing my paper in depth. In response to your comments (please note that I took the liberty to format some of your points and omit some parts for readability):

- [...] I think this undertaking is very worthwhile from several perspectives [...] and promoter capture data whose origins could be better described.
  --> Promoter capture Hi-C is indeed briefly introduced as a technique in the introduction. I have now added some more context on the Javierre et al., 2016 dataset and emphasised its relevance for this workflow at the beginning of the "Promoter capture Hi-C data" subsection.

- [...] The paper is full of "dumps" of show events for R objects that impede the narrative flow drastically.
  --> I didn't realise how annoying this was until you mentioned it. I removed the great majority of dumps, leaving only a few to document the structure of datasets just imported or very final objects. For all dumps, I also ensured that a minimal amount of rows were printed.

- 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.
  --> I included a diagram providing a schematic overview of the workflow as figure 1 and referenced it in the last paragraph of the introduction. Please note that the diagram is created in R with the DiagrammeR package but the code is hidden as it is not strictly relevant for the purposes of the workflow.

- 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.

--> I added a few sentences at the beginning of the "eQTL data" subsection cautioning on the complexity of GWAS/eQTL integration and provided a short overview of available alternatives which are more methodologically robust.

- [...] In particular, it would be great to see how the different elements of the system contribute to the ultimate enumeration of targets.
--> I added figure 7 to show the relative contributions of the different approaches to the final results.

- [...] 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 expanded that paragraph to provide more context on STOPGAP, POSTGAP and INFERNO and to clarify the intent of mentioning those resources in the introduction.

- I feel that the introduction, though well-referenced, is too long and does not clearly state the paper's main goal.
--> I shortened the introduction by removing the paragraph about GWAS and PheWAS and by removing or shortening several other sentences. I added a short, final paragraph stating more clearly the main goals of the workflow.

- There is no discussion of the experimental design underlying the RNA-seq study. Presumably the data were generated from this component of ref 28: [...] 
--> That's correct. I added more context on the original study, including an overview of the experimental design, in the third paragraph of the "Gene expression data and differential gene expression analysis" section.

- 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.
--> I addressed this point with a better introduction to SLE and its heterogeneity in the second paragraph of the "Gene expression data and differential gene expression analysis" section.

- [...] 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.
--> I agree this is a more effective way to summarise the data. I removed the dataframe printing and included a contingency table showing features of case and control samples.

- 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?
--> Indeed, this was mostly done to speed up execution while compiling the document. I removed that chunk and all 117 samples are now used in the analysis.

- 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?

--> I reworded that sentence to clarify that DESeq2 is equivalent to edger and limma when it comes to multiple cofactors in the model. I also included a better description of the metadata available for this dataset and explained why it is not possible to include demographic statistics (unavailable) or other experimental factors (collinear with disease status) in the model.

- 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?

--> I have now applied a simple filter to remove genes with extremely low counts directly on the dds object and ahead of VST, as documented in the DESeq2 vignette [1] and the Bioconductor RNA-seq workflow [2]. This reduces the number of genes considerably, helping to speed up code execution too. I also clarified in the "Gene expression data and differential gene expression analysis" section that one of the aims of the hierarchical clustering and PCA in figure 2 and 3 is indeed to assess presence of batch effects or surrogate variables. Note that all available experimental variables are now included as annotation in the heatmap in figure 1.

- 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.

--> The "Gene expression data and differential gene expression analysis" has now been considerably condensed by removing superfluous object dumps, merging code chunks and reducing the text to a minimum. One could go as far as removing the exploratory data analysis and figures, but I'd rather keep them to provide some context and a minimal differential expression analysis to be used as the starting point for the integration of the GWAS data.

- 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.

--> Indeed. The workflow was largely redacted, condensed and improved by limiting R object dumps, providing more context on the features of the datasets used and more insights into the methodology and results of the analysis through the use of visualisations and data summaries.


**Competing Interests:** No competing interests were disclosed.
gene discovery in disease contexts. However, it would benefit from some more elaboration in certain sections. I have listed my comments below in more detail, ordered by the location in the workflow they refer to. For most part, I believe they are easily addressed.

- The final paragraph of the introduction seems out of place; I do not see any reference to POSTGAP, STOPGAP or INFERNO anywhere else in the article. Was the workflow presented here used to identify the candidate genes in these resources?

- A more comprehensive description of the SLE data set, and the motivation behind using it, would be helpful.

- There seems to be a typo when loading the SRP062966 dataset; it should be `load(file.path("SRP062966", "rse_gene.Rdata"))`, at least on my machine.

- I don’t see why it’s desirable to call `scale_counts()`. Major DE analysis frameworks are easily capable of handling differences in library sizes. Direct scaling would actually be detrimental to NB models like `edgeR` and `DESeq2`, as it distorts the mean-variance relationship. In particular, scaled counts can have sub-Poisson variation, which cannot be handled by NB models. It seems better to call `read_counts()` to 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.

---

Author Response 14 Feb 2018

**Enrico Ferrero, GlaxoSmithKline, UK**

Aaron, many thanks for reviewing my paper in depth.

In response to your comments:

- The final paragraph of the introduction seems out of place; I do not see any reference to POSTGAP, STOPGAP or INFERNO anywhere else in the article. Was the workflow presented here used to identify the candidate genes in these resources?
  --> I expanded the paragraph to provide more context on STOPGAP, POSTGAP and INFERNO and to clarify why they are mentioned in the introduction but not used in the actual workflow.

- A more comprehensive description of the SLE data set, and the motivation behind using it, would be helpful.
More background and details on the dataset have been added in the second and third paragraph of the section “Gene expression data and differential gene expression analysis”.

- There seems to be a typo when loading the SRP062966 dataset; it should be load(file.path("SRP062966", "rse_gene.Rdata")), at least on my machine.
  --> Fixed, thanks.

- I don't see why it's desirable to call scale_counts(). Major DE analysis frameworks are easily capable of handling differences in library sizes. Direct scaling would actually be detrimental to NB models like edgeR and DESeq2, as it distorts the mean-variance relationship. In particular, scaled counts can have sub-Poisson variation, which cannot be handled by NB models. It seems better to call read_counts() to obtain the gene-level read counts.
  --> For this section, I followed the recount quick start guide [1] and workflow [2]. Both show scaling of the counts with scale_counts() before feeding these to DESeq2. I tried switching to read_counts() but, somewhat counter-intuitively, the function returns values with decimal numbers, which in turn causes an error (“some values in assay are not integers”) when calling the DESeqDataSet() function. As both scale_counts() and read_counts() seem to be acceptable, but the first one is the preferred approach by the recount developers, I switched back to scale_counts() after encountering the DESeq2 error above. The other option would have been to manually round the numbers returned by read_counts() but that seemed more questionable to me than scaling them.

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

- Some explanation of the other factors (anti-rho, ISM) would be helpful, given that the effort has already been taken to define them.
  --> I added context for these experimental factors in the third paragraph of the “Gene expression data and differential gene expression analysis” section and after printing the rse$characteristics object is printed.

- 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.
  --> I agree this is not ideal, but there are good reasons why other factors are not included. First, age, gender or other demographics are not available for this dataset. Second, the ISM and anti-Ro factors are disease characteristics and are obviously only measured on the SLE patients (and not on the healthy ones). If either or both of those factors are included in the model, you get the classic “model matrix is not full rank" error [3] because they are both collinear with the disease status (all healthy samples are “control” for both anti-Ro and ISM). I’ve been more explicit about these shortcomings in the paragraph following the code chunk where the model is built.

- 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.
I expanded the main text and legends for figures 2, 4 and 5 (previously 1, 3 and 4) to include a better description and explanation of the plots. I believe figures 3 and 6 (previously 2 and 5) were already adequately described. I also added 3 new figures (1, 7 and 8) to clarify the steps involved in the workflow and to provide a more in-depth understanding of the final results.

- 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.

--> Unfortunately I can't come up with a good way to perform this step in R as part of the workflow at present. The ldblock package hasn't been updated in a while and its functions rely on downloading the HapMap data from the NCBI website, which was retired in 2016 and is no longer available for download [4]. Even if it was still available, it would require downloading several GBs of data, one chromosome at a time. The previously referenced trio package uses data structures specific to case - parent trio studies which are not compatible with the use case presented in the workflow and are not designed for hundreds of SNPs, and was thus removed. The Ensembl LD Calculator is a web UI with a limit of 20 SNPs per query that can't be integrated in a programmatic workflow, so it was removed too. I guess the Ensembl REST API could be an option, but it would require introducing a few new libraries and a considerable amount of code to interact with the API and parse its output into R/Bioconductor objects, with the risk of distracting the reader from the main purpose of this (Bioconductor) workflow. It would also require performing several hundreds queries in a for loop making compilation of the document extremely long and following the workflow impractical. I modified the text in the manuscript to communicate more clearly the reasons for skipping this step. If you have other suggestions on how to do this, I would be happy to consider them.

- 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.

--> I added a clarification and a warning about this in the third paragraph of the "Accessing GWAS data" section, after importing the GWAS data.

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

--> I removed all instances of tail() and replaced them with head().

- 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.

--> I added a new section "Functional analysis of prioritised hits" (and a new figure, 8) where I describe the biological significance and functional relevance of the results, while also discussing some of the hits in more detail from a drug discovery perspective.

- 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.
Thanks, I had heard of the InteractionSet package but hadn't used it before. I agree it's better to represent the promoter capture Hi-C data in this native structure. I still had to use the nearest() function (which executes almost instantaneously on my laptop) to map promoters to gene IDs though. Also, note that I didn't need the linkOverlaps() function in the end and simply used findOverlaps(..., use.region = "second") instead.

[2] https://f1000research.com/articles/6-1558/v1

**Competing Interests:** No competing interests were disclosed.