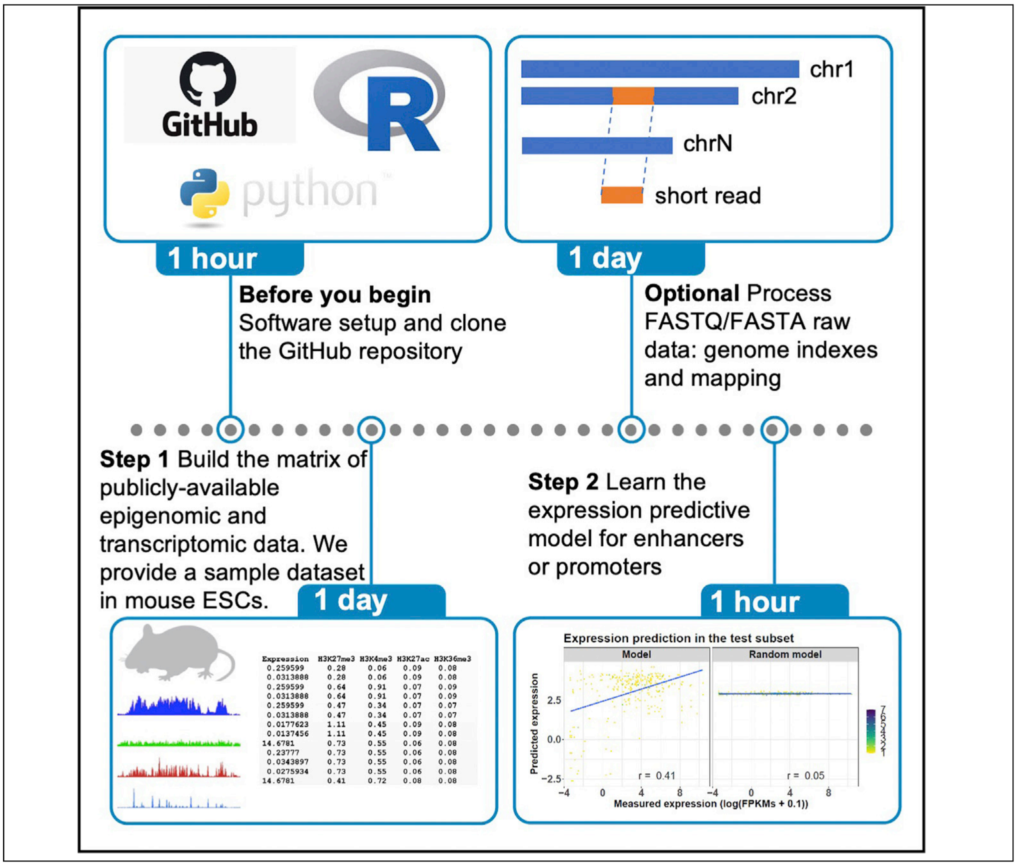
STAR Protocols

CellPress
OPEN ACCESS

Protocol

A computational pipeline to learn gene expression predictive models from epigenetic information at enhancers or promoters



Before you begin Software setup and clone the GitHub repository

Optional Process FASTQ/FASTA raw data: genome indexes and mapping

Step 1 Build the matrix of publicly-available epigenomic and transcriptomic data. We provide a sample dataset in mouse ESCs.

Step 2 Learn the expression predictive model for enhancers or promoters

```
Expression  H3K27me3 H3K4me3 H3K27ac H3K36me3
 0.259599     0.28     0.06    0.09    0.08
 0.0313888    0.28     0.06    0.09    0.08
 0.259599     0.64     0.91    0.07    0.09
 0.0313888    0.64     0.91    0.07    0.09
 0.259599     0.47     0.34    0.07    0.07
 0.0313888    0.47     0.34    0.07    0.07
 0.0177623    1.11     0.45    0.09    0.08
 0.0137456    1.11     0.45    0.09    0.08
14.6781       0.73     0.55    0.06    0.08
 0.23777      0.73     0.55    0.06    0.08
 0.0343897    0.73     0.55    0.06    0.08
 0.0275934    0.73     0.55    0.06    0.08
14.6781       0.41     0.72    0.08    0.08
```

Here, we present a computational pipeline to obtain quantitative models that characterize the relationship of gene expression with the epigenetic marking at enhancers or promoters in mouse embryonic stem cells. Our protocol consists of (i) generating predictive models of gene expression from epigenetic information (such as histone modification ChIP-seq) at enhancers or promoters and (ii) assessing the performance of these predictive models. This protocol could be applied to other biological scenarios or other types of epigenetic data.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Mar González-Ramírez, Enrique Blanco, Luciano Di Croce

gmgz@novonordisk.com (M.G.-R.)
luciano.dicroce@crg.eu (L.D.C.)

Highlights

Protocol to study the relationship between epigenetic marking and gene expression

Steps to calculate ChIP-seq signal strength from mapped reads over regulatory regions

Pipeline written in R to learn predictive models of gene expression from epigenetic data

Steps for ChIP-seq and RNA-seq raw data analysis in case-mapped reads are not available

González-Ramírez et al., STAR Protocols 4, 101948
March 17, 2023 © 2022 The Author(s).
https://doi.org/10.1016/j.xpro.2022.101948

Protocol

A computational pipeline to learn gene expression predictive models from epigenetic information at enhancers or promoters

Mar González-Ramírez,[1,4,5,*] Enrique Blanco,[1] and Luciano Di Croce[1,2,3,6,*]

[1]Centre for Genomic Regulation (CRG), Barcelona Institute of Science and Technology (BIST), Dr. Aiguader 88, 08003 Barcelona, Spain

[2]Universitat Pompeu Fabra (UPF), Barcelona, Spain

[3]Institució Catalana de Recerca i Estudis Avançats (ICREA), Pg. Lluis Companys 23, 08010 Barcelona, Spain

[4]Present address: Genetics Centre of Excellence, Novo Nordisk Research Centre Oxford, Innovation Building Roosevelt Drive, Old Road Campus, Oxford, OX3 7FZ, UK

[5]Technical contact

[6]Lead contact

*Correspondence: gmgz@novonordisk.com (M.G.-R.), luciano.dicroce@crg.eu (L.D.C.)
https://doi.org/10.1016/j.xpro.2022.101948

SUMMARY

Here, we present a computational pipeline to obtain quantitative models that characterize the relationship of gene expression with the epigenetic marking at enhancers or promoters in mouse embryonic stem cells. Our protocol consists of (i) generating predictive models of gene expression from epigenetic information (such as histone modification ChIP-seq) at enhancers or promoters and (ii) assessing the performance of these predictive models. This protocol could be applied to other biological scenarios or other types of epigenetic data.
For complete details on the use and execution of this protocol, please refer to Gonzalez-Ramirez et al. (2021).[1]

BEFORE YOU BEGIN

Timing: 60 min

In brief, our protocol consists of the use of available epigenetic information (such as ChIP-seq samples of histone modifications) at enhancers and promoters, and RNA-seq data, to construct gene expression predictive models in mouse Embryonic Stem Cells (mESCs). As an example, we employ a combination of post-translational histone modifications (i.e., H3K27me3, H3K27ac, H3K4me3 and H3K4me1). For simplicity, we will focus on chromosome 19 (mouse assembly: mm10), rather than the full genome, which would require higher computation time.

This section includes the minimal software and hardware requirements, the installation procedures, as well as the format of the files to be processed throughout this protocol.

Hardware

This bioinformatic pipeline is designed to be run in command line under UNIX operating systems (e.g., Linux or macOS). However, users from Windows platforms can reproduce likewise this protocol by running virtual machines (e.g., Oracle VM VirtualBox, https://www.virtualbox.org), containers (e.g., Docker, https://www.docker.com), or emulation software (e.g., Cygwin, https://www.cygwin.com). Recommended computational requirements: 8 GB of RAM and 50 GB of storage.

Install R, RStudio, python, GCC, and make

Timing: 30 min

1. R is a freely available language and environment for statistical computing and graphics.[2] We recommend to download and install the latest version of R, which can be found in https://cran.r-project.org/.
2. RStudio is an integrated user-friendly development environment for R.[3] We recommend to download and install the latest version of RStudio, which can be found in https://www.rstudio.com/.
3. Python is a scripting, general-purpose, high-level, and interpreted programming language.[4] We recommend to download and install the latest version of Python, which can be found in https://www.python.org/.
4. GNU Compiler Collection (GCC) is an optimizer compiler produced by the GNU project initially developed to handle C and C++ programming language compilation in Linux platforms, which can be found in https://gcc.gnu.org/.
5. GNU Make is a tool to control the generation of executables and other non-source files of a program from source code files, which can be found in https://www.gnu.org/software/make/.

Download this protocol and our dataset from GitHub

Timing: 10 min

6. Clone our GitHub repository (~400 MB), by running the following command on your terminal.

```
% git clone https://github.com/margonram/gene_expression_prediction
```

Note: This repository includes the following input data and scripts needed to learn and evaluate two predictive models of gene expression from epigenetic information at enhancers or promoters in mESCs. Output folders for the storage of future results and graphics are also provided inside the repository.

a. Examine the `data/` folder contents.
   i. It contains two BED files (`Enhancer_gene_chr19.bed` and `Promoter_gene_chr19.bed`) of the active and repressed (poised) regulatory elements (enhancers and promoters, respectively) found in mESCs. On each line of both files, the first three columns represent the coordinates (in mm10) of the regulatory element (chromosome, start position and end position), while the fourth position contains the name of the target gene linked to the current region.

Note: We provide as well the full lists of active and repressed (poised) regulatory elements for all chromosomes in mm10 (`Enhancer_gene.bed` and `Promoter_gene.bed`), in case the reader would like to run this protocol genome-wide. Chromatin state and Hi-C interaction data were combined in our previous publication to generate these lists.[1]

```
% head -5 data/Enhancer_gene_chr19.bed

chr19    10012600    10014600    Fen1

chr19    10012600    10014600    Fth1

chr19    10012600    10014600    Incenp
```

```
chr19    10012600    10014600    Tmem258

chr19    10129000    10130600    Fen1

% head -5 data/Promoter_gene_chr19.bed

chr19    10202800    10205000    Fen1

chr19    10202800    10205000    Tmem258

chr19    10239600    10241200    Myrf

chr19    10304200    10305600    Dagla

chr19    10456200    10458400    Lrrc10b
```

ii. Moreover, it contains the file `gene-FPKMs.txt` with the expression of all 25,457 genes measured by FPKMs (Fragments Per Kilobase of transcript per Million mapped reads) in mESCs from a previously published paired-end RNA-seq experiment of our lab.[5]

Note: It is worth mentioning that files encoding other units of expression, such as RNA-seq tag normalized counts or microarray normalized signal intensities, would be likewise useful for this purpose.[6]

```
% head -5 gene-FPKMs.txt

0610005C13Rik    0.0740133

0610009B22Rik    31.9622

0610009E02Rik    0.137977

0610009L18Rik    0.274511

0610010B08Rik    18.0307
```

iii. Finally, BAM files of mapped reads for chromosome 19 (mm10) of the ChIP-seq experiments used in here are provided so the reader will be able to exactly reproduce this protocol. Total number of reads of each BAM file is shown below:

Histone mark	Mapped reads	Histone mark	Mapped reads
H3K27ac	652,175	H3K4me3	1,108,480
H3K27me3	704,903	H3K36me3	762,963

b. Examine the `scripts/` folder.
   i. It contains a Python script named `prepare_matrix.py` to generate the matrices that combine observed transcriptomic and epigenetic values necessary to learn the predictive models afterwards. This folder also includes two R scripts named `LM_enhancer.R` and `LM_promoter.R` to generate and evaluate the gene expression predictive models from enhancers and promoters, respectively, from previously generated matrices.

c. The `results`/ folder will store new data generated by the scripts of this repository in form of plain text files.

d. The `plots/` folder will store new graphics produced during the learning and evaluation steps of the protocol in PDF format.

Install SeqCode

Timing: 10 min

7. Clone the SeqCode GitHub repository (~20 MB) in your computer, by typing in your terminal the following command.

```
% git clone https://github.com/eblancoga/seqcode
```

Note: SeqCode full distribution contains the source code of the applications, the Makefile to generate the binaries, and a set of automatical tests to check all programs work well in small real datasets.

```
% cd seqcode/

% ls

bin include LICENSE Makefile objects README.md src tests
```

8. Enter into the `seqcode/` folder and use Make to generate SeqCode binaries in the bin/ folder.

```
% make clean

% make all

% ls bin/

buildChIPprofile     matchpeaks          produceTESmaps

combineChIPprofiles  matchpeaksgenes     produceTESplots

combineTSSmaps       processmacs         produceTSSmaps

combineTSSplots      produceGENEmaps     produceTSSplots

computemaxsignal     produceGENEplots    recoverChIPlevels

findPeaks            producePEAKmaps     scorePhastCons

genomeDistribution   producePEAKplots
```

9. Users can run a set of Perl scripts in the `tests/` folder to check SeqCode installation was correct.
10. Copy the binary files on a place that is accessible from your current working path.

⚠ CRITICAL: For further information on SeqCode functions, please access the user manual and the examples of usage for graphical annotation of high-throughput data at http://ldicrocelab.crg.es.

Download and process the `chromInfo.txt` file from UCSC genome browser

Timing: 5 min

11. Download the `chromInfo.txt` file (mouse, mm10) from UCSC Genome Browser[7]: http://hgdownload.soe.ucsc.edu/goldenPath/mm10/database/chromInfo.txt.gz.

Note: Information on the catalogue of chromosomes and their corresponding size is necessary for running SeqCode on BAM files. Alternatively, we can use the BioMart tool from Ensembl,[8] or the NCBI datasets at https://www.ncbi.nlm.nih.gov/data-hub/genome to download the same information.

12. Clean the `chromInfo.txt` file by removing alternative fragments of chromosomes and sequences of unknown location by running the following filter. The resulting `ChromInfo.txt` file will be used throughout this protocol:

```
% zcat chromInfo.txt.gz | grep -v "_" | awk 'BEGIN{OFS="\t"}{print $1,$2}' > \
data/ChromInfo.txt

% cat data/ChromInfo.txt

chr1  195471971

chr2  182113224

chrX 171031299

chr3  160039680

chr4  156508116

chr5  151834684

chr6  149736546

chr7  145441459

chr10 130694993

chr8  129401213

chr14 124902244

chr9  124595110

chr11 122082543

chr13 120421639

chr12 120129022

chr15 104043685

chr16 98207768

chr17 94987271

chrY  91744698

chr18 90702639

chr19 61431566

chrM  16299
```

Install R packages needed for this protocol

Timing: 5 min

13. To run our protocol, it is required to previously install the R packages `caret`[9] and `viridis`.[10]

Note: caret is an R package that implements the learning step of predictive modelling. In addition to this, caret setup will initiate the installation of ggplot2,[11] which is employed to visualize the results of the evaluation, generating publication-ready plots. viridis is a well-known package of graphical palettes that we will use to colour our plots. Open RStudio and run the following commands.

```
> install.packages(‘‘caret’’)

> install.packages(‘‘viridis’’)
```

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Enhancer_gene_chr19.bed	This protocol	https://github.com/margonram/gene_expression_prediction/blob/main/data/Enhancer_gene_chr19.bed
Promoter_gene_chr19.bed	This protocol	https://github.com/margonram/gene_expression_prediction/blob/main/data/Promoter_gene_chr19.bed
gene-FPKMs.txt	This protocol	https://github.com/margonram/gene_expression_prediction/blob/main/data/gene-FPKMs.txt
H3K27ac_mESC_chr19.bam	This protocol	https://github.com/margonram/gene_expression_prediction/blob/main/data/H3K27ac_mESC_chr19.bam
H3K27me3_mESC_chr19.bam	This protocol	https://github.com/margonram/gene_expression_prediction/blob/main/data/H3K27me3_mESC_chr19.bam
H3K36me3_mESC_chr19.bam	This protocol	https://github.com/margonram/gene_expression_prediction/blob/main/data/H3K36me3_mESC_chr19.bam
H3K4me3_mESC_chr19.bam	This protocol	https://github.com/margonram/gene_expression_prediction/blob/main/data/H3K4me3_mESC_chr19.bam
chromInfo.txt	UCSC genome browser	http://hgdownload.soe.ucsc.edu/goldenPath/mm10/database/chromInfo.txt.gz
Software and algorithms		
R	R core Team[2]	https://www.R-project.org/
RStudio	RStudio Team[3]	http://www.rstudio.com/
Caret R package	Kuhn et al.[9]	https://CRAN.R-project.org/package=caret
Viridis R package	Garnier et al.[11]	https://CRAN.R-project.org/package=viridis.
Python	Van Rossum and Drake[4]	https://www.python.org/
GCC	The GNU Project	https://gcc.gnu.org
Make	The GNU Project	https://www.gnu.org/software/make
prepare_matrix.py	This protocol	https://github.com/margonram/gene_expression_prediction/blob/main/scripts/prepare_matrix.py
LM_enhancer.R	This protocol	https://github.com/margonram/gene_expression_prediction/blob/main/scripts/LM_enhancer.R
LM_promoter.R	This protocol	https://github.com/margonram/gene_expression_prediction/blob/main/scripts/LM_promoter.R
SeqCode	Blanco et al.[12]	https://github.com/eblancoga/seqcode

(*Continued on next page*)

Continued

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Other		
Apple iMac (R) Intel (R) Core i7 @ 3,5 GHz, 8 GB memory, macOS Mojave	N/A	N/A

STEP-BY-STEP METHOD DETAILS

Construct the matrix of epigenomic and transcriptomic signal on the sets of enhancers and promoters, and their associated target genes, respectively

⏲ Timing: 1 day

Prior to the generation of gene expression predictive models with this protocol, it is necessary to build a table or matrix linking the epigenomic and transcriptomic information quantified on the regulatory elements and their associated target genes, respectively. The columns in this matrix will represent the variables that we will use to learn and evaluate the predictive models. Each row contains one target gene and the corresponding matching enhancer or promoter region (see an example below). We have decided to place the expression of the target genes in the first column (in FPKMs) and the count of reads of each type of epigenetic data (i.e., ChIP-seq of histone modifications) normalized by the sequencing depth at the regulatory regions of interest (i.e., enhancers or promoters) in the following columns.

Optional: If you would like to apply our protocol on your own set of transcriptomic and epigenomic experiments, please follow steps 1–5 to build your matrix file linking expression and epigenetic information. Alternatively, if you prefer just to see how the predictive model generation and training steps work on the sample data on chromosome 19 of mESCs from our GitHub repository, you can go directly to point 6 of this protocol.

1. Select of a collection of epigenetic experiments.

   Note: To run this protocol, users need to select which types of epigenetic information (histone modifications, chromatin accessibility, etc.) are more interesting for them to use as predictive variables to train computational gene expression predictive models. This will depend on the question we are addressing. In our case, we aimed to study (i) whether enhancer epigenetic landscape is capable of predicting gene expression as previously shown for promoters, and (ii) compare differences in variable contribution between enhancer and promoter predictive models. As we are interested in ranking each histone modification for their variable importance in the predictive models obtained in mESCs, we employ a diverse set of ChIP-seq experiments from a heterogeneous panel of histone modifications.

   ⚠ CRITICAL: In our previous publication we used up to ten different histone modifications.[1] Here, for the sake of clarity, we selected the following histone modifications: H3K27me3, H3K27ac, H3K4me3 and H3K36me3. Remarkably, other types of high-throughput information such as ChIP-seq experiments of transcription factors and chromatin accessibility data[13–16] proved to be effective in the modelling of expression in other biological contexts. Therefore, our readers are encouraged to integrate such data into predictive models alternative to the one with histone marks described along this article.

2. Collect the files of mapped reads.

Note: From the collection of high-throughput experiments selected before, we need to obtain files of reads in BAM/SAM format mapped on the appropriate genome assembly. In our example, we mapped raw data files of each sample on the mm10 version of the mouse genome. Precomputed files of mapped reads can be retrieved from databases such as ENCODE,[17] where readers will find a vast catalogue of epigenetic data on mESC. Troubleshooting 1.

3. Calculate normalized count of reads at enhancers.

Note: We will use the `recoverChIPlevels` function from SeqCode[12] to assign one value of signal strength (i.e., average count of reads) for every ChIP-seq experiment on each element of our collections of genomic regions.

a. Execute the following commands on the terminal for each file of mapped reads of the histone modifications of our example (step 2).

Note: Readers can apply the same commands on other BAM files of reads that could be included in the analysis to determine the abovementioned values on our set of enhancers:

```
% bin/recoverChIPlevels -vnd data/ChromInfo.txt data/H3K27me3_mESC_chr19.bam \
data/Enhancer_gene_chr19.bed Enhancer_H3K27me3_chr19
% bin/recoverChIPlevels -vnd data/ChromInfo.txt data/H3K27ac_mESC_chr19.bam \
data/Enhancer_gene_chr19.bed Enhancer_H3K27ac_chr19
% bin/recoverChIPlevels -vnd data/ChromInfo.txt data/H3K4me3_mESC_chr19.bam \
data/Enhancer_gene_chr19.bed Enhancer_H3K4me3_chr19
% bin/recoverChIPlevels -vnd data/ChromInfo.txt data/H3K36me3_mESC_chr19.bam \
data/Enhancer_gene_chr19.bed Enhancer_H3K36me3_chr19
```

⚠ CRITICAL: Option `-d` in the `recoverChIPlevels` command is only necessary when working with our BAM/SAM files containing mapped reads in only one chromosome. When dealing with the full set of chromosomes users should not include this option.

b. Examine the output of `recoverChIPlevels` SeqCode command.

Note: The output consists on a new BED file in which three new columns (average value, maximum value and total value) have been added at each line after the information regarding the identification of every enhancer and target gene pair in the original input BED data. All values are normalized by SeqCode for the total number of reads inside each BAM file. We will use column number five (average value) for this protocol.

```
% head -5 \
Enhancer_H3K27ac_chr19_recoverChIPlevels/PEAKsignal_Enhancer_H3K27ac_chr19.bed
chr19  10012600  10014600  Fen1     8.64  18.40  1728.06
chr19  10012600  10014600  Fth1     8.64  18.40  1728.06
chr19  10012600  10014600  Incenp   8.64  18.40  1728.06
chr19  10012600  10014600  Tmem258  8.64  18.40  1728.06
chr19  10129000  10130600  Fen1     7.12  16.87  1139.26
```

Note: In the example provided through our GitHub repository, each line corresponds to the Hi-C-top enhancer-gene associations identified in our previous publication.[1] Troubleshooting 2.

4. Elaborate the matrix linking target gene expression and epigenomic information for enhancers.

Note: Once we have calculated the normalized number of reads for each epigenetic item, we need to generate a matrix linking gene expression data and epigenetic signal strength on each pair of enhancer-gene elements to learn the predictive models later. For this purpose, we provide in our repository a python script (`prepare_matrix.py`) that constructs the observations matrix by combining the actual expression data (RNA-seq FPKM values) and the output of each `recoverChIPlevels` function call (ChIP-seq average signal values).

a. Create the parameter file named `ChIPlevels_enhancers.txt` (see Box 1) in the `data/` folder.

Note: The parameter file named `ChIPlevels_enhancers.txt` (see Box 1) contains the location in the user file system of the BED files generated by the `recoverChIPlevels` commands in step 3, and a unique identifier (i.e., name of the corresponding histone modification). This file will have as many rows as epigenetic samples we would like to utilize as variables to learn the predictive models in the next steps.

b. Run our Python script on the terminal to finally link expression data and epigenomic ChIP-seq levels for each pair of gene-enhancer record:

```
% ./scripts/prepare_matrix.py data/ChIPlevels_enhancers.txt \
data/gene-FPKMs.txt > results/matrix_observations_enhancers.txt
% head -5 results/matrix_observations_enhancers.txt
expression  H3K27me3  H3K4me3  H3K27ac  H3K36me3
47.2144  1.70  4.28  8.64  2.99
563.153  1.70  4.28  8.64  2.99
46.8749  1.70  4.28  8.64  2.99
307.058  1.70  4.28  8.64  2.99
```

⚠ CRITICAL: prepare_matrix.py is intended to be executed by the python2 interpreter. However, it can be run under python3 by changing "python" in the first line (shebang) of our script for "python3".

Note: Both input and output files on this command for chromosome 19 of mouse (mm10) are provided in the `data/` and `results/` folders of our GitHub repository, respectively. This resulting matrix_observations_enhancers.txt archive will be used as input data to learn predictive models in the next steps of the protocol. Troubleshooting 3.

5. Repeat steps 3 and 4 for promoters.

Note: In previous steps, we have quantified the signal strength of each ChIP-seq experiment on the enhancers previously linked to genes in mESCs. Now, to calculate the accuracy of a similar predictive model for gene expression in promoters, we will repeat this calculation over the promoters linked to the same set of genes using the `Promoter_gene_chr19.bed` (provided in our GitHub repository). This file contains the coordinates of the promoters of

Box 1. Contents of the file `ChIPlevels_enhancers.txt`

```
Enhancer_H3K27ac_chr19_recoverChIPlevels/PEAKsignal_Enhancer_H3K27ac_chr19.bed. H3K27ac

Enhancer_H3K27me3_chr19_recoverChIPlevels/PEAKsignal_Enhancer_H3K27me3_chr19.bed H3K27me3

Enhancer_H3K4me3_chr19_recoverChIPlevels/PEAKsignal_Enhancer_H3K4me3_chr19.bed. H3K4me3

Enhancer_H3K36me3_chr19_recoverChIPlevels/PEAKsignal_Enhancer_H3K36me3_chr19.bed H3K36me3
```

chromosome 19 involved in the Hi-C-top interactions as identified in our previous publication.[1] After running the SeqCode commands, we will generate now a parameter file called `ChIPlevels_promoters.txt` (Box 2) to specify the location of the normalized counts of each ChIP-seq experiment processed by SeqCode.

```
% bin/recoverChIPLevels -vnd data/ChromInfo.txt \

data/H3K27me3_mESC_chr19.bam data/Promoter_gene_chr19.bed \

Promoter_H3K27me3_chr19

% bin/recoverChIPlevels -vnd data/ChromInfo.txt \

data/H3K27ac_mESC_chr19.bam data/Promoter_gene_chr19.bed \

Promoter_H3K27ac_chr19

% bin/recoverChIPLevels -vnd data/ChromInfo.txt \

data/H3K4me3_mESC_chr19.bam data/Promoter_gene_chr19.bed \

Promoter_H3K4me3_chr19

% bin/recoverChIPLevels -vnd data/ChromInfo.txt \

data/H3K36me3_mESC_chr19.bam data/Promoter_gene_chr19.bed \

Promoter_H3K36me3_chr19
```

All output files follow the same format and layout as in the case of enhancers.

```
% head -5 \

Promoter_H3K27ac_chr19_recoverChIPlevels/PEAKsignal_Promoter_H3K27ac_chr19.bed

chr19  10202800  10205000  Fen1     13.42  39.87  2953.20

chr19  10202800  10205000  Tmem258    13.42  39.87  2953.20

chr19  10239600  10241200  Myrf      7.67  15.33  1226.66

chr19  10304200  10305600  Dagla    3.68 7.67 515.20

chr19  10456200  10458400  Lrrc10b    2.93 7.67 644.00

% ./scripts/prepare_matrix.py data/ChIPlevels_promoters.txt \

data/gene-FPKMs.txt > results/matrix_observations_promoters.txt

% head -5 results/matrix_observations_promoters.txt

expression  H3K27me3  H3K4me3  H3K27ac  H3K36me3

47.2144  1.26   153.31   13.42  2.20

307.058  1.26   153.31   13.42  2.20

12.3486  2.28    57.73    7.67  2.46

0.259599 15.54   75.08    3.68  2.96
```

Box 2. Contents of the file `ChIPlevels_promoters.txt`

```
Promoter_H3K27ac_chr19_recoverChIPlevels/PEAKsignal_Promoter_H3K27ac_chr19.bed H3K27ac

Promoter_H3K27me3_chr19_recoverChIPlevels/PEAKsignal_Promoter_H3K27me3_chr19.bed H3K27me3

Promoter_H3K4me3_chr19_recoverChIPlevels/PEAKsignal_Promoter_H3K4me3_chr19.bed H3K4me3

Promoter_H3K36me3_chr19_recoverChIPlevels/PEAKsignal_Promoter_H3K36me3_chr19.bed H3K36me3
```

Learning the predictive models for enhancers or promoters

Timing: 1 h

The next part of this protocol consists in learning the predictive models of gene expression by finding the optimal weight for the signal strength of each epigenetic variable (i.e., H3K27ac) on each class of genomic regions (enhancers and promoters, separately), that predicts a value of expression that fits better with the observed FPKM values in mESCs. We will show step by step how to (i) learn an enhancer predictive model from a part of the input matrix of observations (training set) and a random model as a control (generated by shuffling gene expression values), and (ii) evaluate the accuracy of both models on the rest of input data (test set).

Note: We will focus first on the R script `LM_enhancer.R` which contains the steps to generate the gene expression predictive model based on enhancer histone decoration. Similar steps will be indicated later to perform the same operation on promoter histone marking.

6. Prepare the workspace directory. Open RStudio and set the path of the main folder of the clone of our GitHub repository inside your computer (`yourpath/`) as workspace directory.

```
> setwd("yourpath")
```

7. Load the R packages in the current RStudio session.

Note: We load `caret`[9] and `viridis`.[10] Moreover, `caret` will load the `ggplot2` plotting package[11] as well.

```
> library(caret)

> library(viridis)
```

8. Load the transcriptomic and epigenomic matrix of observed counts.

Note: We load the resulting file from point 5 (`matrix_observations_enhancers.txt`) into a variable named `data`, and perform a log2 transformation, adding a pseudocount of 0.1. We need to convert our data into a data frame.

```
> data <- read.table("results/matrix_observations_enhancers.txt",

                    header = T) # substitute by your file
```

```
> head(data,n=5)

expression H3K27me3 H3K4me3 H3K27ac H3K36me3

1    47.2144    1.70    4.28    8.64    2.99

2    563.1530 1.70    4.28    8.64    2.99

3    46.8749    1.70    4.28    8.64    2.99

4    307.0580 1.70    4.28    8.64    2.99

5    47.2144    1.01    4.35    7.12    2.39

> data <- log2(data+0.1)

> head(data,n=5)

  expression H3K27me3 H3K4me3 H3K27ac H3K36me3

1    5.564207 0.8479969 2.130931 3.127633 1.627607

2    9.137639 0.8479969 2.130931 3.127633 1.627607

3    5.553818 0.8479969 2.130931 3.127633 1.627607

4    8.262837 0.8479969 2.130931 3.127633 1.627607

5    5.564207 0.1505597 2.153805 2.851999 1.316146

> data <- data.frame(data)

> nrow(data)

[1] 1426

> ncol(data)

[1] 5
```

⚠ CRITICAL: We provide examples of expression and epigenomic observations for enhancers and promoters in mESCs in our GitHub repository. Therefore, unless you are processing your own high-throughput experiments, steps 1–5 can be skipped when working with intermediate results from our GitHub examples.

9. Create the training and test sets.

   Note: We will randomly partition the input matrix of enhancer-gene observations stored in the `data` variable into two sets: 80% of the BED entries for the training set and the remaining 20% for the test set.

   a. Choose the randomization seed. We generated our current partitioning using 47.

```
> set.seed(47)
```

   b. Create both partitions by permutating indexes of the elements in the data structures and extracting the data associated to such indexes into the `dataTrain` and `dataTest` variables.

```
> trainIndex <-createDataPartition(data$expression,

                                    p = .8,list = FALSE, times = 1)

> head(trainIndex,n=2)
```

```
Resample1
[1,]    1
[2,]    2
> tail(trainIndex,n=2)
Resample1
[1141,]    1424
[1142,]    1425
> dataTrain <- data[ trainIndex,]
> dataTest <- data[-trainIndex,]
> nrow(dataTrain)
[1] 1142
> nrow(dataTest)
[1] 284
> head(dataTrain,n=2)
expression H3K27me3 H3K4me3 H3K27ac H3K36me3
1    5.564207 0.8479969 2.130931 3.127633 1.627607
2    9.137639 0.8479969 2.130931 3.127633 1.627607
> head(dataTest,n=2)
expression H3K27me3 H3K4me3 H3K27ac H3K36me3
5    5.564207 0.1505597 2.153805 2.851999 1.316146
8    5.564207 1.2509616 1.815575 3.412782 1.731183
```

Optional: Under certain circumstances, instead of partitioning the data in two sets (training and test), we might be interested in learning the predictive model in a particular biological context and evaluate its performance on a second one (e.g., mESCs vs. a posterior cell differentiated state, see our previous publication[1]). In this case, readers will need to adapt this R script to load two files of matrices of observations: one will be stored inside the `dataTrain` variable, and the other one into the `dataTest` variable. Both files will need to be previously processed with the command in the step 8. Troubleshooting 4.

10. Randomize gene expression.

```
> expression_rd <- sample(dataTrain$expression)
> dataTrain_rd <- dataTrain
> dataTrain_rd$expression <- expression_rd
> head(dataTrain_rd,n=2)
expression H3K27me3 H3K4me3 H3K27ac H3K36me3
1    9.137639 0.8479969 2.130931 3.127633 1.627607
2    5.466689 0.8479969 2.130931 3.127633 1.627607
```

Note: As a control, we will generate a second predictive model in whose training set the expression data of target genes coupled to enhancers from mESCs is shuffled. Performance on this random model, when comparing predicted vs. observed gene expression, will be useful as a reference to evaluate the results of the model trained on the correct matching between expression of target genes and enhancers.

11. Define the training conditions.

```
> train_control <- trainControl(method="repeatedcv",
                              number=10, repeats = 3)
```

Note: Cross validation (CV) is a statistical method to obtain better estimations when fitting a model to a limited dataset. By iteratively splitting data into k parts, CV method is based on using k-1 fractions of the original input for fitting k-1 models to the observations leaving one part for evaluating the sample error at each iteration.[18] Prior to executing the learning step, we will define the set of technical conditions that must be followed. Thus, for both predictive models (real and random), we will perform a 10× cross-validation (`method=''repeatedcv''` and `number=10`), repeated three times (`repeats=3`). We can modify these parameters by those that best suit our needs.

12. Learn the predictive models.

Note: To learn the parameters of our models (real and random), we will fit gene expression values predicted using the observed epigenomic read counts with the observed gene expression (original or shuffled) at each enhancer-gene pair. Gene expression must be always the first term of the formula. Terms after the symbol "~" correspond to the counts of each piece of epigenetic information we selected before. We can run several fitting methods, in our case we opted for a linear model (`lm`).

```
> model <- train(expression~H3K36me3+H3K4me3+H3K27me3+H3K27ac,
              data=dataTrain, trControl=train_control,
              method="lm")
> model_rd <-train(expression~H3K36me3+H3K4me3+H3K27me3+H3K27ac,
                 data=dataTrain_rd, trControl=train_control,
                 method="lm")
```

Note: Indeed, we compared linear regression (lm) to other existing methods such as neural networks, support vector machines, or random forests in our previous publication.[1] While no other approach yielded better performance, a higher amount of running time was required in the other methods. Please, refer to `caret`[9] for further information on all the available methodologies.

13. Save the resulting model.

Note: We visualize the content of the model and the model summary, and save their respective outputs. Among other things, we can see the actual weight assigned to each epigenetic feature and their significance, as well as the error statistics of the CV method.

```
> model
Linear Regression
1142 samples
  4 predictor
No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 1027, 1029, 1028, 1028, 1026, 1028, ...
Resampling results:
RMSE    Rsquared    MAE
2.739144 0.1969188  2.092115
Tuning parameter 'intercept' was held constant at a value of TRUE
> model_print <- capture.output(print(model))
> out_print <- paste("results/Enhancer_predictive_model.txt",
                      sep = "")
> writeLines(model_print,out_print)
```

```
> summary(model)
Call:
lm(formula = .outcome ~ ., data = dat)
Residuals:
    Min      1Q   Median     3Q    Max
-8.0254 -1.5536   0.0824 1.6723 8.5237
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.39910  0.58615  5.799  8.64e-09 ***
H3K36me3     0.08289  0.09154  0.906  0.365
H3K4me3      0.05827  0.06853  0.850  0.395
H3K27me3    -0.91799  0.07708  -11.909 < 2e-16 ***
H3K27ac      0.13557  0.14893  0.910  0.363
–
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 2.741 on 1137 degrees of freedom
Multiple R-squared: 0.1978, Adjusted R-squared: 0.195
F-statistic: 70.09 on 4 and 1137 DF, p-value: < 2.2e-16
```

```
> model_sum <- capture.output(summary(model))

> out_sum <-paste("results/Enhancer_predictive_model_summary.txt",

                  sep = "")

> writeLines(model_sum,out_sum)
```

14. Calculate the importance of each variable.

```
> varImp(model, scale = FALSE)

lm variable importance

          Overall

H3K27me3  11.9091

H3K27ac    0.9103

H3K36me3   0.9056

H3K4me3    0.8503

> model_imp <- capture.output(varImp(model))

> out_imp <- paste("results/Enhancer_varImp.txt", sep = "")

> writeLines(model_imp,out_imp)
```

Note: To define the impact of each histone modification over gene expression prediction, we calculate their importance into the predictive model. Importance is defined as the contribution of each variable in the linear regression predictive model and corresponds to the absolute value of the t-statistic for each model parameter. We save the output of calculating variable importance into a file.

15. Calculate the correlation between predicted and measured expression.

```
> exp_pred <- predict(model,dataTest)

> r <- round(cor(exp_pred,dataTest$expression),2)

> r

[1] 0.42

> exp_pred_rd <- predict(model_rd,dataTest)

> r_rd <- round(cor(exp_pred_rd,dataTest$expression),2)

> r_rd

[1] -0.01
```

Note: Finally, once we have generated the model learned on the training set, we can evaluate its performance on the test set (step 9). For this purpose, we predict the expression of genes in the test set using the real and the random models, and then we calculate Pearson's correlation (r) between predicted and measured expression in both cases to assess predictive model's performance.

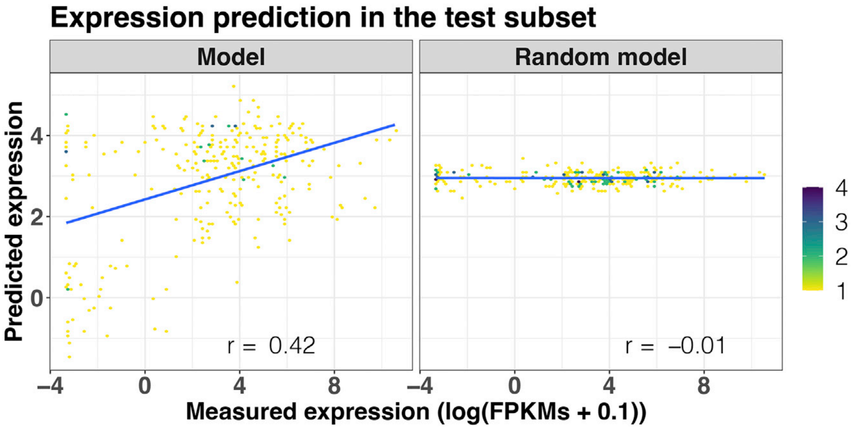


Figure 1. Performance of the enhancer model and its corresponding random model

Predicted expression of the test subset of genes calculated by the models versus their measured expression by RNA-seq. Model performances are represented by the Pearson's correlation (*r*) between predicted and measured expression values. Color scale on the right represents the density of dots in the scatter plots.

16. Plot predicted vs. measured expression.

```
> predicted <-c (exp_pred, exp_pred_rd)
> measured <- c(dataTest$expression, dataTest$expression)
> models <- c(rep("Model", nrow(dataTest)),
            rep("Random model", nrow(dataTest)))
> text <- data.frame(label=c(paste("r = ", r),
                  paste("r = ", r_rd)),
                  models=c("Model", "Random model"))
> c <- data.frame(predicted, measured, models)
> pdf("plots/Enhancer_prediction.pdf", width = 10, height = 5)
> ggplot(c) +
  geom_hex(aes(measured, predicted), bins = 100) +
  scale_fill_gradientn("", colours = rev(viridis(300)))+
  geom_smooth(aes(measured, predicted), method = "lm", level=0)+
  labs(title="Expression prediction in the test subset",
      x="Measured expression (log(FPKMs + 0.1))",
      y = "Predicted expression") +
  geom_text(data = text,
           mapping = aes(x = -Inf, y = -Inf, label = label), hjust = -2,
           vjust = -1, size=7) +
  theme_bw() +
  theme(legend.position="right",
       axis.text=element_text(size=20, face="bold"),
       axis.title=element_text(size=20, face="bold"),
       legend.text=element_text(size=20),
       legend.title=element_text(size=20),
       plot.title = element_text(size=24,face="bold"),
       strip.text = element_text(size=20,face="bold")) +
  facet_wrap(~ models)
> dev.off()
```

Note: We use the `ggplots2` R package to generate publication-ready plots to visualize how well our predictive model perform compared to their corresponding random model (Figure 1).

17. Plot the variable importance.

Note: Finally, we can show in a barplot the different contribution of each histone modification into the gene expression prediction based in enhancer epigenetic marking (Figure 2A).

```
> out_varImp <- paste("plots/Enhancer_varImp.pdf",sep="")
> p <- ggplot(varImp(model,scale = FALSE))+
  geom_bar(stat="identity", fill="darkviolet")+
  labs(title="Enhancer model variable importance",
       x="", y = "importance") +
  theme_bw() +
  theme(legend.position="bottom",
        axis.text=element_text(size=12,face="bold"),
        axis.title=element_text(size=12,face="bold"),
        plot.title = element_text(size=15,face="bold"),
        strip.text = element_text(size=12,face="bold"))
> ggsave(out_varImp,plot=p,device = "pdf",width = 5, height = 4)
```

18. Repeat steps 6–17 for promoter analysis.

Note: To generate a predictive model of gene expression from epigenetic information at promoter regions, we will resume from step 6 with the file of data observations in promoter regions by running the R script `LM_promoter.R`. It follows the same steps shown in the R script `LM_enhancer.R`, being adapted to read and output promoter files. We provide in our GitHub repository an example of the file `matrix_observations_promoter.txt` for promoters. Graphical results are shown in Figures 2B and 3.

19. Comparing the variable importance of histone marks between enhancer and promoter models.

Note: Finally, we will compare the ranking of variable importance between enhancer and promoter models to highlight potential differences. Figure 2.

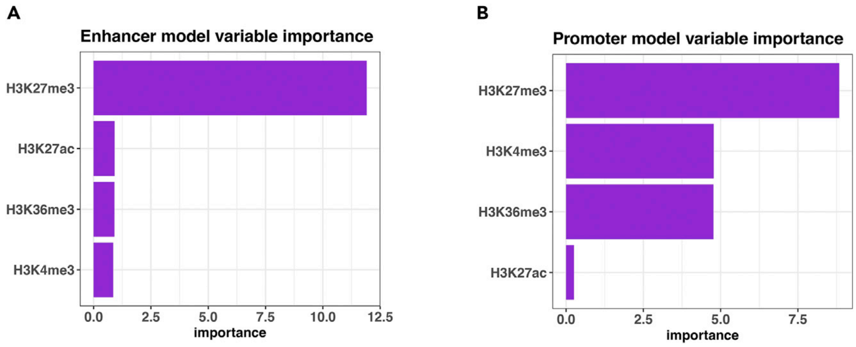


Figure 2. Variable importance

(A) Importance of each histone modification used to train the enhancer predictive model. Importance is defined as the contribution of each variable in the linear regression predictive model and corresponds to the absolute value of the t-statistic for each model parameter.

(B) As for A, but for promoters.

EXPECTED OUTCOMES

Plots in Figures 1 and 3 show the expression predicted by the computational models (real and random) based in epigenetic decoration of enhancers and promoters vs. measured expression from published RNA-seq experiments in mESCs.[5] Their color scale represents the density of dots, and the blue line is the regression line. We plot *r-value* which represents the Pearson's correlation between measured and predicted gene expression. We expect the random model from both, enhancers and promoters, to have an almost horizontal regression line, with a *r-value* lower than the one from the real model and usually close to 0 (Figures 1 and 3). However, the *r-value* of the random model could virtually take any value depending on the result of the randomization (i.e., it could take the same value as the real model, but the probability of this happening is very low). The higher the *r-value* of the real predictive model, the higher the correlation between predicted and measured expression, and therefore, the better the performance of the predictive model.

Plots in Figure 2 show the ranking of variable importance. The value of importance cannot be fairly compared between different models (i.e., the same variable in the enhancer or the promoter model). However, we can compare the changes in the ranking in which each variable appears on each model.

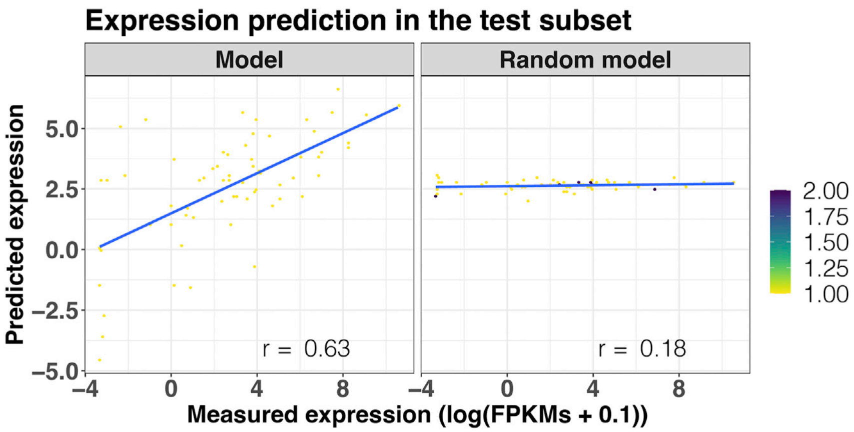


Figure 3. Performance of the promoter model and its corresponding random model

Predicted expression of the test subset of genes calculated by the models versus their measured expression by RNA-seq. Model performances are represented by the Pearson's correlation (*r*) between predicted and measured expression values. Color scale on the right represents the density of dots in the scatter plots.

H3K4me3, a histone mark mostly associated to active promoters, is the variable showing a more dramatic change as expected (null contribution in enhancers, substantial contribution in promoters). H3K27me3, mostly present in transcriptionally inactive regions is the most important contributor in both cases as it is the best feature to distinguish between expressed and silent genes.[1] The fact that H3K27ac has little importance for both, enhancer and promoter predictive models, may be surprising for the reader especially in the case of enhancers, as H3K27ac is considered the canonical marker of active enhancers.[19] However, as we demonstrated in our previous publication,[1] this is due to H3K27me3 and H3K27ac being mutually exclusive,[20] and therefore not independent variables.

LIMITATIONS

To obtain proper models of gene expression prediction from enhancer epigenetic information, we need repressed (poised) and active enhancers. However, poised enhancers have only been identified in embryonic stem cells.[21,22] They might exist in other cell types, but further research is needed in order to confirm it. On the other hand, the type of epigenetic data and gene expression data in which one could be potentially interested might not be publicly available for the cellular model in which the reader is planning to produce gene expression predictive models.

TROUBLESHOOTING

Problem 1

At step 2, BAM or SAM files are not available for our epigenetic information of interest.

Potential solution

If processed files of mapped reads (BAM/SAM) are not available for our epigenetic data of interest, we should download their raw data files in FASTQ format and perform the mapping over our current genome assembly. We propose using popular mapping tools such as BOWTIE[23] or BWA.[24,25] Prior to mapping, indexes from the appropriate genome assembly must be generated from the FASTA sequences of the chromosomes. Both genome indexing (which is generated only once at each computer) and ChIP-seq read mapping are time-consuming operations that will take between 1–2 h running in ordinary workstations.

In brief, we will execute the following steps to generate the index:

- Get a copy of the sequence of each chromosome of the genome of interest (mouse, mm10 assembly in our example) in FASTA format from the UCSC genome browser. We suggest to download the same chromosomes described in our previous chromInfo.txt file: https://hgdownload.soe.ucsc.edu/goldenPath/mm10/chromosomes/.
- FASTA sequences of the chromosomes must be concatenated together in a single multi-FASTA file with the UNIX zcat command:

```
% zcat chr*.fa.gz > genome.fa
```

- Let genome.fa be a multi-FASTA file combining the chromosomes of a genome, the following BOWTIE command generates and stores the genome indexing files into the output folder:

```
% bowtie-build genome.fa output/genome
```

Next, for the mapping of the FASTQ file of one ChIP-seq experiment, we will perform the following steps:

- Alignment of the ChIP-seq reads from a FASTQ raw data file (sample.fastq) to the appropriate genome index (output/genome), discarding multi-locus reads, and saving the output in SAM format (sample.sam) with BOWTIE:

```
% bowtie -p 4 -t -m 1 -S -q output/genome sample.fastq sample.sam
```

SAM files contain the location of each read in the genome.[26] To remove unaligned reads and to convert the resulting SAM file into BAM format to save storage space, users will run the following SAMTools command:

```
% samtools view -b -F 0×4 -o sample.bam sample.sam
```

Problem 2

At step 3, we would like to identify our own set of enhancers and their target genes.

Potential solution

For this protocol we provide the coordinates of enhancers and their target genes identified in mESCs in our previous publication.[1] We used the combination of several histone modifications (H3K27me3, H3K27ac, H3K4me3 and H3K4me1) to identify active and poised enhancers, as well as promoters. Moreover, we used Hi-C interactions to associate enhancers to promoters and target genes, as long as promoters and enhancers share the same chromatin state (i.e., both active or both poised). The same approach can be utilized for other cell types as well. If Hi-C data is not available, we can predetermine a 1 Mb maximum distance between enhancer and promoter of same chromatin state to assign potential enhancer-gene associations. This arbitrary distance approach indeed showed an acceptable performance of gene expression prediction in our previous publication.[1]

Problem 3

At step 4, we would like to use expression data from another cell-type different from mESCs.

Potential solution

For this particular protocol we provide FPKMs for mESCs in our GitHub repository. However, readers might be interested in learning predictive models for other scenarios. In that case, we need to calculate FPKMs in the RNA-seq samples performed over the cell-type of our interest. We propose software such as Cufflinks[27] and DESeq2[28] for this purpose. If previously we need to map reads of RNA-seq raw data files to generate the corresponding BAM files, we suggest using TopHat.[29] Similar to ChIP-seq analysis, genome indexing is necessary and mapping of RNA-seq plus read counting are time-consuming operations that will take between 4–5 h running in ordinary workstations.

For the mapping of the FASTQ files of an RNA-seq experiment, we will perform the following steps on the same indexes generated for ChIP-seq analysis.

- Get a copy of the annotated exons of all genes in the genome of interest (mouse, mm10 assembly in this example) according to the RefSeq consortium as provided by the UCSC genome browser. The UCSC tool genePredToGtf can be used to convert annotations in GTF format for mapping and quantification: https://hgdownload.soe.ucsc.edu/goldenPath/mm10/database/refGene.txt.gz.
- We suggest two alternative TopHat commands, to process single-end (`sample.fastq`) and paired-end samples (`sample1.fastq` and `sample2.fastq`), respectively:

```
% tophat -no-coverage-search -p 4 -g 1 -G refGene.gtf -o project_name output/genome \
sample.fastq

% tophat -no-coverage-search -mate-inner-dist 100 -p 4 -g 1 -G refGene.gtf -o project_name \
-library-type=fr-firststrand output/genome sample1.fastq sample2.fastq
```

- To quantify the expression in terms of FPKMs/RPKMs for all mouse transcripts as annotated by RefSeq, users can execute the cufflinks program over the previously mapped reads (sample.bam):

```
% cufflinks -max-bundle-frags 5000000 -p 1 -G refGene.gtf -o project_name sample.bam
```

Problem 4

At step 9, we would like to train and evaluate our models in two different cellular scenarios, but the high-throughput data is distributed from two different sources.

Potential solution

In case our training and test set come from two different sources, we need to normalize both, epigenetic data and expression data, to get rid of any possible bias. We proposed a normalization based on a local regression (LOESS) whose good performance we demonstrate in our previous publication.[1] Without this normalization, correct interpretation of the results can be misleading in many scenarios.

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Luciano Di Croce luciano.dicroce@crg.eu.

Materials availability

This study did not generate new unique reagents.

Data and code availability

The datasets and code generated during this study are available at GitHub: https://github.com/margonram/gene_expression_prediction.

(DOI at Zenodo: https://doi.org/10.5281/zenodo.7341909).

ACKNOWLEDGMENTS

The work in the Di Croce laboratory is supported by grants from ''la Caixa'' Foundation (HR20-00411), the Spanish Ministry of Science and Innovation (PID2019-108322GB-100), ''Fundación Vencer El Cancer'' (VEC), and the European Regional Development Fund (FEDER) and from AGAUR. We acknowledge support from the Spanish Ministry of Science and Innovation to the EMBL partnership, the Centro de Excelencia Severo Ochoa, and the CERCA Programme/Generalitat de Catalunya.

AUTHOR CONTRIBUTIONS

M.G.R. has designed and implemented the current bioinformatics protocol and written this manuscript. E.B. has participated in testing and revising the source code and in the editing of this manuscript. L.D.C. has contributed into the revision of this manuscript and in the supervision of the full protocol.

DECLARATION OF INTERESTS

M.G.R. is currently a full-time employee of Novo Nordisk Research Centre Oxford Ltd.

REFERENCES

1. González-Ramírez, M., Ballaré, C., Mugianesi, F., Beringer, M., Santanach, A., Blanco, E., and Di Croce, L. (2021). Differential contribution to gene expression prediction of histone modifications at enhancers or promoters. PLoS Comput. Biol. *17*, e1009368. https://doi.org/10.1371/journal.pcbi.1009368.
2. R Core Team (2020). R: A Language and Environment for Statistical Computing (R Core Team). https://www.R-project.org/.
3. RStudio Team (2020). RStudio: Integrated Development Environment for R (RStudio Team). http://www.rstudio.com/.
4. Van Rossum, G., and Drake, F.L. (2009). Python 3 Reference Manual (CreateSpace).
5. Beringer, M., Pisano, P., Di Carlo, V., Blanco, E., Chammas, P., Vizán, P., Gutiérrez, A., Aranda, S., Payer, B., Wierer, M., and Di Croce, L. (2016). EPOP functionally links elongin and polycomb in pluripotent stem cells. Mol. Cell *64*, 645–658. https://doi.org/10.1016/j.molcel.2016.10.018.
6. Karlić, R., Chung, H.R., Lasserre, J., Vlahovicek, K., and Vingron, M. (2010). Histone modification levels are predictive for gene expression. Proc. Natl. Acad. Sci. USA *107*, 2926–2931. https://doi.org/10.1073/pnas.0909344107.
7. Karolchik, D., Hinrichs, A.S., and Kent, W.J. (2007). The UCSC genome browser. Curr. Protoc. Bioinformatics *Chapter 1*, Unit 1.4. https://doi.org/10.1002/0471250953.bi0104s17.
8. Cunningham, F., Allen, J.E., Allen, J., Alvarez-Jarreta, J., Amode, M.R., Armean, I.M., Austine-Orimoloye, O., Azov, A.G., Barnes, I., Bennett, R., et al. (2022). Ensembl 2022. Nucleic Acids Res. *50*, D988–D995. https://doi.org/10.1093/nar/gkab1049.
9. Kuhn, M., Wing, J., Weston, S., Williams, A., Keefer, C., Engelhardt, A., Cooper, T., Mayer, Z., Kenkel, B., Team, R.C., et al. (2022). Classification and regression training. https://CRAN.R-project.org/package=caret.
10. Hadley, W. (2016). ggplot2: Elegant Graphics for Data Analysis (Springer-Verlag New York). https://ggplot2.tidyverse.org.
11. Garnier, S., Ross, N., Rudis, B., Sciani, B., Camargo, A.P., and Scherer, C. (2021). Viridis - Colorblind-Friendly Color Maps for R (R Package). https://CRAN.R-project.org/package=viridis.
12. Blanco, E., González-Ramírez, M., and Di Croce, L. (2021). Productive visualization of high-throughput sequencing data using the SeqCode open portable platform. Sci. Rep. *11*, 19545. https://doi.org/10.1038/s41598-021-98889-7.
13. Ouyang, Z., Zhou, Q., and Wong, W.H. (2009). ChIP-Seq of transcription factors predicts absolute and differential gene expression in embryonic stem cells. Proc. Natl. Acad. Sci. USA. *106*, 21521–21526. https://doi.org/10.1073/pnas.0904863106.
14. Cheng, C., and Gerstein, M. (2012). Modeling the relative relationship of transcription factor binding and histone modifications to gene expression levels in mouse embryonic stem cells. Nucleic Acids Res. *40*, 553–568. https://doi.org/10.1093/nar/gkr752.
15. Duren, Z., Chen, X., Jiang, R., Wang, Y., and Wong, W.H. (2017). Modeling gene regulation from paired expression and chromatin accessibility data. Proc. Natl. Acad. Sci. USA *114*, E4914–E4923. https://doi.org/10.1073/pnas.1704553114.
16. Schmidt, F., Kern, F., and Schulz, M.H. (2020). Integrative prediction of gene expression with chromatin accessibility and conformation data. Epigenet. Chromatin *13*, 4. https://doi.org/10.1186/s13072-020-0327-0.
17. ENCODE Project Consortium, Moore, J.E., Purcaro, M.J., Pratt, H.E., Epstein, C.B., Shoresh, N., Adrian, J., Kawli, T., Davis, C.A., Dobin, A., et al. (2020). Expanded encyclopaedias of DNA elements in the human and mouse genomes. Nature *583*, 699–710. https://doi.org/10.1038/s41586-020-2493-4.
18. Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proceedings of the 14th International Joint Conference on Artificial Intelligence. (IJCAI'95), *2* (Morgan Kaufmann Publishers Inc.). https://dl.acm.org/doi/proceedings/10.5555/1643031.
19. Creyghton, M.P., Cheng, A.W., Welstead, G.G., Kooistra, T., Carey, B.W., Steine, E.J., Hanna, J., Lodato, M.A., Frampton, G.M., Sharp, P.A., et al. (2010). Histone H3K27ac separates active from poised enhancers and predicts developmental state. Proc. Natl. Acad. Sci. USA *107*, 21931–21936. https://doi.org/10.1073/pnas.1016071107.
20. Shema, E., Jones, D., Shoresh, N., Donohue, L., Ram, O., and Bernstein, B.E. (2016). Single-molecule decoding of combinatorially modified nucleosomes. Science *352*, 717–721. https://doi.org/10.1126/science.aad7701.
21. Rada-Iglesias, A., Bajpai, R., Swigut, T., Brugmann, S.A., Flynn, R.A., and Wysocka, J. (2011). A unique chromatin signature uncovers early developmental enhancers in humans. Nature *470*, 279–283. https://doi.org/10.1038/nature09692.
22. Zentner, G.E., Tesar, P.J., and Scacheri, P.C. (2011). Epigenetic signatures distinguish multiple classes of enhancers with distinct cellular functions. Genome Res. *21*, 1273–1283. https://doi.org/10.1101/gr.122382.111.
23. Langmead, B., Trapnell, C., Pop, M., and Salzberg, S.L. (2009). Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol. *10*, R25. https://doi.org/10.1186/gb-2009-10-3-r25.
24. Li, H., and Durbin, R. (2009). Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics *25*, 1754–1760. https://doi.org/10.1093/bioinformatics/btp324.
25. Li, H., and Durbin, R. (2010). Fast and accurate long-read alignment with Burrows-Wheeler transform. Bioinformatics *26*, 589–595. https://doi.org/10.1093/bioinformatics/btp698.
26. Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., and Durbin, R.; 1000 Genome Project Data Processing Subgroup (2009). The sequence alignment/map format and SAMtools. Bioinformatics *25*, 2078–2079. https://doi.org/10.1093/bioinformatics/btp352.
27. Trapnell, C., Williams, B.A., Pertea, G., Mortazavi, A., Kwan, G., van Baren, M.J., Salzberg, S.L., Wold, B.J., and Pachter, L. (2010). Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. Nat. Biotechnol. *28*, 511–515. https://doi.org/10.1038/nbt.1621.
28. Love, M.I., Huber, W., and Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol. *15*, 550. https://doi.org/10.1186/s13059-014-0550-8.
29. Trapnell, C., Pachter, L., and Salzberg, S.L. (2009). TopHat: discovering splice junctions with RNA-Seq. Bioinformatics *25*, 1105–1111. https://doi.org/10.1093/bioinformatics/btp120.