Published in final edited form as:
Nat Protoc. 2017 December ; 12(12): 2478–2492. doi:10.1038/nprot.2017.124.

Chromatin state discovery and genome annotation with ChromHMM

Jason Ernst[1,2,3,4,5] and **Manolis Kellis**[6,7]

[1]Department of Biological Chemistry, University of California, Los Angeles, Los Angeles, California, USA

[2]Computer Science Department, University of California, Los Angeles, California, USA

[3]Eli and Edythe Broad Center of Regenerative Medicine and Stem Cell Research at University of California, Los Angeles, Los Angeles, California, USA

[4]Jonsson Comprehensive Cancer Center, University of California, Los Angeles, Los Angeles, California, USA

[5]Molecular Biology Institute, University of California, Los Angeles, California, USA

[6]Broad Institute of MIT and Harvard, Cambridge, Massachusetts, USA

[7]MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, Massachusetts, USA

SUMMARY

Non-coding DNA regions play central roles in human biology, evolution, and disease. ChromHMM helps annotate the non-coding genome using epigenomic information across one or multiple cell types. It combines multiple genome-wide epigenomic maps, and uses combinatorial and spatial mark patterns to infer a complete annotation for each cell type. ChromHMM learns chromatin state signatures using a multivariate hidden Markov model that explicitly models the combinatorial presence or absence of each mark. ChromHMM uses these signatures to generate a genome-wide annotation for each cell type, by calculating the most probable state in each genomic segment. ChromHMM provides automated enrichment analysis of the resulting annotations to facilitate the functional interpretations of each chromatin state. ChromHMM is distinguished by its modeling emphasis on combinations of marks, its tight integration with downstream functional enrichment analyses, its speed, and its ease of use. Chromatin states are learned, annotations produced, and enrichments computed within one day.

Keywords

chromatin states; ChromHMM; genome annotation; epigenomics; segmentation; regulatory regions; enhancers; promoters; regulatory genomics

Correspondences should be addressed to J.E. (jason.ernst@ucla.edu) or M.K. (manoli@mit.edu).

Author Contributions

J.E. and M.K. wrote this protocol and previously developed ChromHMM.

Competing Financial Interests

The authors declare that they have no competing financial interests.

INTRODUCTION

Mapping of epigenomic marks, such as histone modifications, histone variants, regions of open chromatin, and related marks has emerged as a powerful means to annotate genomes, to identify putative regulatory elements, and to study their changing activity across different cell types and in human disease[1–4]. Individual marks can be studied in isolation, either through aggregation of their genome-wide signal tracks relative to a set of predetermined annotations[5], such as transcription start sites or exon boundaries, or by discovery of narrow peaks or broader domains where that mark is present in greater frequency than for surrounding regions[6]. However, additional information can be gained by studying combination of multiple marks in their spatial context. Such patterns, termed ‘chromatin states’, often capture known classes of genomic elements such as promoters, enhancers, transcribed, repressed, and repetitive regions[7], and can also capture novel classes or subclasses of elements. Recognizing chromatin states, and identifying their genomic occurrences in each cell type, provides a systematic annotation of DNA elements and regulatory control regions across cell types, which can then be used to interpret GWAS results, study gene regulation, and analyze cellular differentiation, among many other applications (Fig. 1).

Development of the protocol

To address the challenge of interpreting many genome-wide maps of diverse epigenomic marks, we described the concept of chromatin states, whereby a generative machine learning model can be used to infer the hidden chromatin state in a given cell type based on the observed epigenomic marks at each genomic position[7]. We termed these hidden states of the genome ‘chromatin states,’ a term that captures both the probabilistic nature of a multi-state model, and the biological nature of the state of chromatin at that location. We used a multivariate hidden Markov model (HMM), which allowed probabilistic modeling of both the combinatorial presence and absence of multiple marks (in the emission parameters of the multivariate HMM) and the spatial constraints of how these mark combinations occur relative to each other across the genome (in the transition matrix of the HMM).

The mark emission probability vector of each state represents the probability with which each mark is found in that state. Dependencies between marks are captured by the different chromatin states, but within each state, mark emission probabilities are assumed to be independent. The transition probabilities from each state to each other state enable the model to capture the positional biases of chromatin states relative to each other, such as broad transcribed or repressed domains. The model parameters are learned from the data *de novo* based on an unsupervised machine learning procedure that iteratively attempts to maximize the model fit to the data.

We implemented our method in a robust open source software package, ChromHMM[8], which enables learning of chromatin states, annotates their occurrences across the genome, and facilitates biological interpretation of these states by automatically computing state enrichments for external annotations. Given that histone modification marks occur at the resolution of individual nucleosomes, ChromHMM partitions the genome in 200-nucleotide intervals by default, which roughly corresponds to the resolution of a nucleosome and spacer region, although the interval size can be altered by a user-specified parameter. For each genomic interval, ChromHMM then determines the presence or absence of each mark based on the significance of observed count of sequencing reads relative to a Poisson background distribution, or alternatively accepts user-specified binarization such as those of peak callers. ChromHMM uses the resulting presence-absence calls to learn a chromatin state model, and an annotation of state occurrences across the genome. The ChromHMM software includes parallelization support for multi-processor machines which can substantially reduce run-time.

We originally applied the chromatin state modeling framework to genome-wide ChIP-seq maps of 41 epigenomic modifications and related factors from a single cell type, CD4T cells[5,7,9], showing that their combinations and relative genomic positions were highly informative of distinct biological functions. We later demonstrated the applicability of the method to simultaneous annotation of multiple cell types, by virtual concatenation of nine genome-wide epigenomic maps each from nine cell types, allowing us to learn a common set of chromatin states and their locations across the nine cell types[1] (Fig. 2). We subsequently scaled this approach to more than 100 cell and tissue types (Figs. 1b,c)[3], and also showed open chromatin assays can also be used as features[10,11]. When not all marks are available in all cell types, ChromHMM can make the best assignments with the available data. Alternatively, ChromHMM can also be applied to imputed data produced by a program like ChromImpute[12], which enables chromatin state annotations to be produced based on a consistent set of marks across a concatenated set of cell types even if each mark is not mapped in every cell type considered. Another approach for applying ChromHMM to multiple cell type data is to stack the tracks from the multiple cell types over the same genomic location[8,13] (Fig. 2).

Applications of the method

ChromHMM has been widely used in diverse applications, including studies of the non-coding genome, gene regulation, and human disease. The ChromHMM software has been downloaded thousands of times and cited in the literature more than 500 times as of April 2017. Large consortium projects including ENCODE[10], Roadmap Epigenomics[3], Blueprint[14], CEEHRC[15], Mouse ENCODE[16], and Drosophila modENCODE[17], have used ChromHMM as a basis for their analyses. ChromHMM reference annotations for human are incorporated as reference tracks in popular genome browsers including UCSC[18] and Ensembl[19]. In the context of gene regulation, ChromHMM has been used for understanding long-range chromatin interactions[20], nascent transcripts[21], cellular reprogramming[22,23], topologically-associated domains[24], transcription factor binding preferences[1,25], and regulatory motif disruptions in massively parallel reporter assays[26]. In the context of human disease, ChromHMM annotations have been analyzed in conjunction with genome-wide association studies (GWAS) for predicting the cell types relevant to human traits or predicting mechanism of individual GWAS identified loci[1,4,27,28]. ChromHMM has had applications to studying diseases such as Alzheimer's disease[29,30], neurodegeneration[31], type 2 diabetes[32] and cancer[33–36], among many other diseases. ChromHMM has also been used for studying chromatin state variation across individuals[37], chromatin state changes across different *Drosophila* species[38], and aging[39,40], among many other applications.

In this protocol, we will focus on running ChromHMM to obtain a new chromatin state model and constructing genome-wide annotations, rather than the diverse applications the resulting annotations enable. We also note that for many of these applications, a researcher can simply use existing chromatin state annotations available for more than 100 cell or tissue types based on data generated by the Roadmap Epigenomics[3], ENCODE[10], and Blueprint[14] projects (Table 1). These chromatin state annotations can be accessed through multiple different browsers and web portals (Table 1). Additionally chromatin state annotations for SNPs can be queried directly through databases, including HaploReg[41] or RegulomeDB[42].

Comparison with Other Methods

ChromHMM is among the first, most widely used, and most widely cited tools for chromatin state discovery and annotation, but a number of other software packages are now available that address this problem. Before ChromHMM, HMMSeg[43] used a hidden Markov model to partition the genome into two states[44]. Segway was later developed by the same group using Dynamic Bayesian Networks and used to generate annotations based on more states[45]. A number of other tools have appeared since for learning chromatin states and annotating the genome including TreeHMM[46], GATE[47], diHMM[48], CMINT[49], hiHMM[50], IDEAS[51], Segway-GBR[52], STAN[53], GenoSTAN[54], EpiCSeg[55], and Spectacle[56], some of which are tailored to specific use cases, such as strand-specific annotations[53], hierarchical modeling[48], or reduced running time but with decreased ability to distinguish different types of broader domains[56]. Additional chromatin state methods and models have been proposed, but without new publicly available software applications designed for the task[57–64]. Additional tools exist for related problems in unsupervised epigenome annotation: jMOSAiCS[65] and scHMM[66] call peaks or domains for epigenomic marks in the context of other marks, but only annotate peaks or explicitly observed combinations of them which can grow exponentially with the number of marks; ChromaSig[67] discovers multi-mark epigenomic patterns in fixed size windows, but leaves large portions of the genome unannotated to any pattern.

ChromHMM has several characteristics distinguishing it from most methods in this vibrant space. (1) While most other methods seek to model the signal levels of each mark, which can lead to overfitting small signal variation due to noise, ChromHMM focuses its modeling power on combinations of epigenomic marks, by using binary presence/absence input features. This has enabled ChromHMM to discover chromatin states such as a state associated with Zinc finger genes (despite the diffuse signal of its associated marks and small genome coverage)[3,7,12] and a putative bivalent promoter state[68] that are often missed by other methods that directly model signal levels, even when applied on the same datasets[54]. (2) ChromHMM's robust and efficient implementation including multi-core parallelization enables it to be used for large scale applications as demonstrated by learning models based on a dozen marks across more than 100 cell and tissue types while using the entire genome for training[12]. (3) ChromHMM can work directly from aligned reads so separate signal generation software is not needed. (4) ChromHMM also has practical advantages including its ease of use, easy installation (it only requires unzipping a zip file), and broad portability (it only requires Java, which is commonly installed on many systems, and has no external dependencies or additional system requirements).

ChromHMM also provides a tight integration with downstream chromatin state enrichment analyses, including sets of external annotation files for commonly studied species, which facilitates biological interpretation of the discovered chromatin states. After producing a chromatin state annotation, ChromHMM also automatically provides a report that includes the model parameters, state enrichments for external annotations, and chromatin state annotations in formats for both downstream computational analysis and browser visualization.

Experimental Design

ChromHMM has a number of different high level commands (Box 1), which provide support for binarizing aligned reads, learning models based on the binarized data, and conducting downstream analyses. ChromHMM is flexible in the types of data it can model. Any type of data that can be binarized in a meaningful way can in principle be applied to ChromHMM. ChromHMM's provided binarization procedure uses the given set of aligned reads. If desired, manipulations to reads such as collapsing duplicate reads or downsampling an imbalanced and deeply sequenced dataset should be conducted prior to providing the reads to ChromHMM. For all text files ChromHMM supports both unzipped files and gzipped files with a .gz extension.

Box 1

Summary of Main ChromHMM Commands

This box summarizes some of the main commands of ChromHMM. Additional details about these commands and other commands can be found in the ChromHMM manual. Additional details about some of the output formats are presented in Box 2. For the file naming 'N' corresponds to the number of states and 'celltype' corresponds to one of the cell types.

BinarizeBam

This command converts a set of bam files of aligned reads into binarized data files in a specified output directory, which can then be used as input to LearnModel.

BinarizeBed

Similar to BinarizeBam, but takes aligned reads in bed format instead of bam format.

BinarizeSignal

This command takes a set of signal files, by default assumed to represent counts, and converts them into binarized data in a specified output directory, which can then be used as input to LearnModel.

LearnModel

This command takes as input a set of binarized data files and then has ChromHMM learn a chromatin state model. After learning the model, ChromHMM produces a genome annotation in both a simple four column bed format and bed formats designed to view in the browser. After producing the genome annotation, ChromHMM also computes overlap and neighborhood enrichments for a set of external annotations. In addition to learning the model, this command is effectively also executing the MakeSegmentation, MakeBrowserFiles, OverlapEnrichment, and NeighborhoodEnrichments commands. A webpage is a created with links to all the files and images created (see Box 2, and Fig. 3).

MakeSegmentation

This command takes a previously learned model and binarized data and outputs for each cell type a segmentation file containing a genome annotation in a four column bed format in files named celltype_N_segments.bed.

MakeBrowserFiles

This command takes as input a segmentation file containing a genome annotation in four column bed format and converts it into two types of bed formats for viewing in the browser, one in which a genome annotation can be displayed on a single line and the other in which there is one line per state, named celltype_N_dense.bed and celltype_N_expanded.bed respectively.

OverlapEnrichment

This command computes the fold enrichment of each state of a ChromHMM generated annotation for a set of external annotations generating the output in text and image formats, producing files named celltype_N_overlap.txt, celltype_N_overlap.png, and celltype_N_overlap.svg.

NeighborhoodEnrichment

This command computes the fold enrichment of each state relative to a set of anchor positions generating the output in text and image formats, producing files named celltype_N_anchorfeature_neighborhood.txt, celltype_N_anchorfeature_neighborhood.png, and celltype_N_anchorfeature_neighborhood.svg.

CompareModels

This command compares emission parameters of models with different numbers of states to a reference model in terms of parameter correlations. The reference model is usually selected to be the model with the largest number of states being considered. The output contains for each state of the reference model and for each other model being compared the maximum correlation of the state of the reference model with any state of the model being compared. The output is generated in .txt, .png, and .svg image formats.

Reorder

This command reorders states of the model or the columns of the emission matrix, and outputs updated model parameter files.

Box 2

Default Output Files of ChromHMM LearnModel Command

For the following files ‘N’ indicates the number of states and ‘celltype’ indicates the cell type. If ChromHMM was applied in concatenated mode to multiple cell types there would be corresponding files for each cell type.

webpage_N.html

This is a webpage which contains the commands used to learn the model and links to all the files generated as part of the output.

emissions_N.txt

This a tab delimited text file containing the emission parameters of the hidden Markov model. The first row is a header row indicating each mark used to define the model and the first column indicates the state. The values corresponding to probability of observing the mark of the column conditioned on being in the state of the row.

emissions_N.png, emissions_N.svg

These files display the contents of emissions_N.txt as a heatmap in .png and .svg image formats respectively.

transitions_N.txt

This is a tab delimited text file containing the transition parameters of the model. The first row and first column are header rows and columns respectively. The values correspond to the probability under the model conditioned on being in the state of the row of transitioning to the state of the column.

transitions_N.png, transitions_N.svg

These files display the contents of transitions_N.txt as a heatmap in .png and .svg image formats respectively.

model_N.txt

This file contains the emission, transition, and initialization parameters of the hidden Markov model in a format for which ChromHMM can parse all the parameters.

celltype_N_segments.bed

This file is a segmentation of the genome containing ChromHMM’s genome annotation for a cell type in an easy to parse format. This is a four column file in .bed format. The first column gives the chromosome, the second column gives the beginning coordinate of the segment, the third column gives the end coordinate of the segment, and the fourth column gives the state. The fourth column includes both the state prefix, which indicates how the states were ordered and the state number. The state prefixes are ‘E’ for emission based ordering, ‘T’ for transition based ordering, and ‘U’ for user specified ordering.

celltype_N_dense.bed

This file gives ChromHMM’s genome annotation in a format that can be loaded into a browser and displayed on a single line with different states being differentiated by different colors.

celltype_N_expanded.bed

This file gives ChromHMM’s genome annotation in a format that can be loaded into a browser and displayed such that each state appears on a different line.

celltype_N_overlap.txt

This file gives the enrichment of the states for various external annotations. The first row and column are header rows and columns respectively. The second column contains the % of the genome each state covers. The last row indicates the % of the genome each external annotation covers. The remaining rows correspond to different states and the remaining columns different external annotations. The values in these rows and columns correspond to the fold enrichments for the presence of the external annotation in the state. The fold enrichments are computed as the ratio of the fraction of bases assigned to the state that are in the external category to the fraction of bases in the genome that are in the external category.

celltype_N_overlap.png, celltype_N_overlap.svg

These files display the contents of celltype_N_overlap.txt in .png and .svg image formats respectively except without the bottom row containing the base percentages.

celltype_N_anchorfeature_neighborhood.txt

This file contains the fold enrichment of each chromatin state at fixed positions relative to a set of ‘anchorfeature’ positions where each ‘anchorfeature’ corresponds to one coordinate file provided to the neighborhood enrichment analysis.

celltype_N_anchorfeature_neighborhood.png, celltype_N_anchorfeature_neighborhood.svg

These files display the contents of celltype_N_anchorfeature_neighborhood.txt in .png and .svg image formats respectively.

Level of expertise needed to implement the protocol

ChromHMM uses a command-line interface, which facilitates scripting, but requires a minimum level of familiarity with command-line execution of programs. It also generates reports of model parameters and enrichments for external genomic annotations, which requires some familiarity with genome biology to help in the biological interpretation of the results. Beyond basic scripting and some understanding of genomics, no other specialized skills are needed.

Limitations of ChromHMM

ChromHMM can automatically infer models and compute enrichments for the states of the models based on external annotations without prior biological knowledge, however the state interpretation still involves a knowledgeable human. The interpretations one gives to states are only candidate state annotations, and one should not assume each instance assigned to a state is a true instance of the annotation.

While ChromHMM has a `CompareModels` command that facilitates comparing emission parameters of models with different number of states it does not automatically decide on the number of states one should use. While sometimes there can be a clear lower bound on the number of states since key biological state distinctions are not resolved with fewer states, the upper bound can be less clear. In general, the number of biologically-meaningful chromatin states that can be discovered will increase with the number of input tracks, though this will depend on the extent of redundant information in the tracks. Lastly, even for a fixed set of input tracks, the desired resolution of biological interpretation can dictate the number of states.

ChromHMM works on fixed bin sizes (default 200bp), but for some applications one can obtain more relevant higher-resolution boundaries by directly considering nucleosome depletion from histone modification signal[1], DNase I peaks[69] or footprints within them[70,71]. These boundaries could then still be used in conjunction with chromatin state annotations[72].

MATERIALS

EQUIPMENT

Hardware: ChromHMM (mirrored at http://www.biolchem.ucla.edu/labs/ernst/ChromHMM and http://compbio.mit.edu/ChromHMM) can be run on any system supporting Java 1.5 or later (https://java.com/en/download/). A user manual with more detailed documentation is also available from the ChromHMM website. The software including source code and documentation is also available through a version control repository at https://github.com/jernst98/ChromHMM. The recommended input data to ChromHMM is coordinates of aligned reads in either bam, bed, or tagAlign formats. This protocol is written assuming aligned read data will be used. If aligned reads are unavailable, such as is the case with imputed data, ChromHMM provides support for such data through the BinarizeSignal command described in the user manual.

PROCEDURE

Installation TIMING 5 min

1 If not already installed, install Java from https://java.com/en/download/ and follow its corresponding instructions.

2 Download ChromHMM from http://www.biolchem.ucla.edu/labs/ernst/ChromHMM or http://compbio.mit.edu/ChromHMM/. This will save a file ChromHMM.zip to a local computer. After the file has saved, then unzip the file. Note this protocol is based on ChromHMM v1.12.

3 From a command line change into the directory with the unzipped ChromHMM files.

Binarization TIMING 1 hr

4 Prepare a tab delimited text specifying the design file for the cell-mark-aligned read files according to whether there is: a single cell type (option A) or if there are multiple cell types whether they will be treated independently (option B), concatenated (option C), or the features stacked to provide a single genome annotation (option D) (Fig. 2).

A. Single cell type

i. Format a single file in the form:

```
cell1    mark1        cell1_mark1_file
cell1    mark2        cell1_mark2_file
```

B. Multiple cell types treating independently

i. Format a file for each cell type in the form:

File 1:

```
cell1    mark1        cell1_mark1_file
cell1    mark2        cell1_mark2_file
```

File 2:

```
cell2    mark1        cell2_mark1_file
cell2    mark2        cell2_mark2_file
```

C. Multiple cell types concatenating

i. Format a single file in the form:

```
cell1    mark1        cell1_mark1_file
cell1    mark2        cell1_mark2_file
cell2    mark1        cell2_mark1_file
cell2    mark2        cell2_mark2_file
```

Note if some mark is only specified in a subset of the cell types then it will be treated as missing in the remaining cell types (encoded with a ‘2’) in the binarized files.

D. Multiple cell types stacking features

i. Format a single file in the form:

```
genome   cell1_mark1
```

```
cell1_mark1_file
         genome   cell1_mark2
cell1_mark2_file
         genome   cell2_mark1
cell2_mark1_file
         genome   cell2_mark2
cell2_mark2_file
```

If there are multiple files for the same cell and mark combination, then each can be specified and their reads will be pooled or alternatively the files can be combined outside of ChromHMM.

CAUTION: The .bed or .bam files supplied to `BinarizeBed` or `BinarizeBam` should in general contain the locations of aligned reads. ChromHMM does also allow binarizing data based on already called peaks opposed to aligned reads in which case the '-peaks' flag would need to be specified when executing the `BinarizeBam` or `BinarizeBed` commands described below otherwise the output would not be meaningful. If supplying already called peaks to ChromHMM it is not recommended to use broad peak calls. ChromHMM already considers spatial information, and if supplying broad peaks then ChromHMM could have states corresponding to combination of marks that do not actually co-occur at the resolution at which it operates.

5 If control data such as whole cell extract or IgG ChIP data[73] is available, consider using it either as an additional feature (option A), or by adjusting the binarization threshold locally based on the relative level of the control reads (option B)[8]. Option A is recommended if the control data is relatively flat except in specific positions often associated with artifacts or if the control data was not sequenced deeply. Option B is recommended if the control data shows peaks in other than artifact associated regions and is sequenced sufficiently deep.

A. Treat control data as an additional feature.

i. Format the cell-mark-aligned read file(s) to have the form:

```
cell1     mark1          cell1_mark1_file
cell1     mark2          cell1_mark2_file
cell1     control        cell1_control_file
```

B. Use control data to adjust the binarization threshold locally.

i. Format the cell-mark-aligned read file(s) to have the form:

```
   cell1    mark1    cell1_mark1_file
cell1_control_file
   cell1    mark2    cell1_mark2_file
cell1_control_file
```

where the control file is the fourth column. Mark specific control files can also be specified.

6 Determine if the genome assembly corresponding to the data has a chromosome length file in the CHROMSIZES directory. Chromosome length files for the assemblies: hg18, hg19, hg38, mm9, mm10, rn5, rn6, danRer7, danRer10, dm3, dm6, ce6, and ce10 are currently provided in the CHROMSIZES directory. If the desired assembly is already included nothing needs to be done for this step.

If the desired assembly is not found in the directory, then it must be prepared and placed in the CHROMSIZES directory. The format is a two column format where the first column is the chromosome name and the second is the chromosome length. Such information for a number of additional assemblies is available from the UCSC genome browser[74]. Linux and Mac users can obtain such files directly by first downloading the fetchChromSizes script from the corresponding directory for their system here: http://hgdownload.cse.ucsc.edu/admin/exe/. At the command prompt type

```
./fetchChromSizes assembly > assembly.txt
```

7 Execute the command to binarize the data. If the reads are in BED or tagAlign format execute the BinarizeBed command (option A). If the reads are stored in BAM files enter the command BinarizeBam (option B).

A. Execute BinarizeBed

i. Enter the command

```
java -mx4000M -jar ChromHMM.jar BinarizeBed
chrlengthfile
inputdir cellmarkfiletable outputdir
```

`chrlengthfile` contains the chromosome length file described in step 6, `inputdir` is the directory with the aligned reads, `cellmarkfiletable` is the design file described in step 5, and `outputdir` is the directory where the binarized data should be written. For each cell type and chromosome, ChromHMM will generate one binarized data file. The first line will contain the cell type and chromosome. The second line will specify the marks, and the remaining lines for each consecutive bin the binary call for each mark. ‘- `mx4000M`‘ specifies the amount of memory given to Java and should be adjusted as needed for this and other commands.

When executing the commands also add any appropriate optional parameters. Optional parameters are specified immediately after `BinarizeBam` or `BinarizeBed.` The full set of options are discussed in the manual. Some options to note:

- `-b binsize` – This specifies the bin size that ChromHMM will use. The default is 200 base pair bins. If setting it to a different value then the same option and value needs to be included with other commands. If using the default ChromHMM binarization procedure this should not be made too small otherwise there will be too few reads to reliably make presence calls.
- `-p poissonthreshold` – This specifies the tail probability of the Poisson distribution that the binarization threshold should correspond to. The default value is 0.0001.
- `-f foldthresh` – This parameter specifies a threshold on the minimum fold enrichment for the signal to be called present even if the Poisson significance threshold is met. By default this value is 0, but setting it to larger values may be useful when working with deeply and imbalanced sequenced datasets.
- `-n shift` – This parameter specifies the amount reads are shifted to determine their bin assignment. Reads are shifted in a strand aware way from the 5′ end in the direction of the 3′ end. The default read shift is 100 base pairs. ChromHMM only directly supports a single read shift value for all marks. If different shift amounts are needed for different marks they should be binarized separately and then merged together outside of ChromHMM.
- `-center` – Specify this flag if reads should be placed in the bin based on the center coordinate position.
- `-peaks` – As discussed in step 4, include this flag if the .bed or .bam files correspond to peak calls instead of aligned reads. It is not recommended to use peaks calls based on broad domains as input.

B. **Execute BinarizeBam**

i. Enter the command

```
java -mx4000M -jar ChromHMM.jar BinarizeBam
chrlengthfile
inputdir cellmarkfiletable outputdir
```

The options for BinarizeBam are the same as BinarizeBed above.

->TROUBLESHOOTING

File Preparation for Enrichment Analyses TIMING 30 min

8 If interested in including external coordinate annotations not already included with ChromHMM in the overlap enrichment analysis of the ChromHMM genome annotation, then prepare additional external coordinate files for this analysis. ChromHMM includes some files for overlap enrichment analysis for the assemblies: hg18, hg19, hg38, mm9, mm10, rn5, rn6, danRer7, danRer10, dm3, dm6, ce6, and ce10. If using one of these assemblies add additional files into the corresponding assembly subdirectory within the `COORDS` directory. If using a different assembly then first create a subdirectory for that assembly within the `COORDS` directory. Coordinate files should be in .bed format. This step can also be done after executing the `LearnModel` command with enrichments then computed by running `OverlapEnrichment`, though they would not be automatically included in the report generated by the `LearnModel` command.

9 If interested in including external position annotations not already included with ChromHMM in the neighborhood enrichment analysis of the ChromHMM genome annotation, then prepare additional external position files for this analysis. ChromHMM includes some files for the same assemblies listed for overlap enrichments above. If using one of these assemblies add additional files into the corresponding assembly subdirectory within the `ANCHORFILES` directory. If using a different assembly then first create a subdirectory for that assembly within the `ANCHORFILES` directory. The format of the file should be a tab delimited text file where the first column is the chromosome, the second the coordinate, and optionally the third is the strand (‘+’ or ‘−’). This step can also be done after executing `LearnModel` with enrichments then computed by running `NeighborhoodEnrichment`, though they would not be automatically included in the report generated by the `LearnModel` command.

Model Learning TIMING 4hr (highly variable)

10 To learn a model and have automatic enrichments computed execute the `LearnModel` command:

```
java -mx4000M -jar ChromHMM.jar LearnModel inputdir
outputdir numstates assembly
```

In this command `inputdir` specifies the directory containing the binarized data which will often be the output directory from step 7. `outputdir` is the directory where the output files from the `LearnModel` command should go. `numstates` specifies the number states. `assembly` specifies the genome assembly for the input.

When executing the commands also add any appropriate optional parameters. Optional parameters are specified immediately after `LearnModel`. The full set of options and additional details are discussed in the manual. Some options to note:

- `-b binsize` – This specifies the bin size and should match what is specified for `BinarizeBam` or `BinarizeBed`.
- `-p maxprocessors` – This parameter specifies the maximum number of processors ChromHMM should use. To have ChromHMM try to use as many processors to which it has access add the '-p 0' flag. If the '-p' option is specified ChromHMM uses a parallel model learning procedure, which can significantly reduce runtime when multiple processors are available. If the '-p' option is not specified, then ChromHMM uses a single processor and an alternative model learning procedure designed for a single processor.
- `-init information|random|load` – This parameter specifies the approach to initializing the parameters of the model. `information` is the default approach[8]. `random` initializes the parameter randomly based on a random seed. `load` initializes the parameters based on the contents of a model file with smoothing away from 0 by default. Note for the `information` initialization the number of states cannot be greater than the number of observed mark combinations.
- `-r maxiterations` – This parameter specifies the maximum number of iterations over all the input in the model learning. The default is 200.
- `-printstatebyline` – This flag specifies to also output the state assignment of each bin printed one line per bin in consecutive order. The first two lines are header lines. These files end with '_maxstate.txt'
- `-printposterior` – This flag specifies to also output the posterior probabilities over state assignments for each bin. The first two lines are header lines, then each consecutive line has the posterior probability for each state in order in tab delimited format. These files end with `'_posterior.txt'`.

->TROUBLESHOOTING

11 Repeat step 10 for models with different numbers of states. If a computer cluster is available, then these models should be learned in parallel.

Post-Model Learning Analysis TIMING 2 hr

12 If interested in comparing models with different numbers of states at the emission parameter level, which can complement comparisons at the enrichment level, use the `CompareModels` command to generate a heatmap comparison by entering the command:

```
java –mx4000M –jar ChromHMM.jar CompareModels
referencemodel comparedir outputprefix
```

The `referencemodel` parameter specifies a file which contains the emission parameters of a reference model to which other models should be compared. Generally this will be a model with the largest number of states being considered. The file should start with ‘`emissions_`’ and end with ‘`.txt`’ or ‘`.txt.gz`’. The `comparedir` parameter specifies the directory that has the model emission parameters that should be compared to the `referencemodel`. The files should start with ‘`emissions_`’ and end with ‘`.txt`’ or ‘`.txt.gz`’. `outputprefix` contains the prefix of the output files including possibly a directory. ‘`.txt`’, ‘`.svg`’, and ‘`.png`’ are appended to this prefix to produce files containing the output (Box 1).

13 If interested in computing additional overlap enrichments other than those that were already computed after applying the `LearnModel` command, then use the `OverlapEnrichment` command by entering:

```
java -mx4000M -jar ChromHMM.jar OverlapEnrichment
inputsegment inputcoorddir outfileprefix
```

In this command `inputsegment` specifies the segmentation file for which overlap enrichments should be computed. `inputcoorddir` specifies the directory containing the external coordinates for overlap enrichment analysis. These files should be prepared according to Step 8. `outfileprefix` contains the prefix, including possibly the directory, of the output files which will have the extensions ‘`.txt`’, ‘`.png`’, and ‘`.svg`’ added to them (Box 1).

When executing the commands also add any appropriate optional parameters. Optional parameters are specified immediately after `OverlapEnrichment.` The full set of options are discussed in the manual. Some options to note:

- `-b binsize` – This specifies the bin size and should match what was used in `LearnModel`.
- `-center` – This specifies to compute the enrichment based on the center position only of each interval in the external coordinate file. With this option each entry has effectively equal weight, while by default entries that are wider will be given greater weight.
- `-multicount` – This specifies that bases overlapped by multiple intervals should be counted multiple times. The default is to count a base only once.
- `-m labelmappingfile` – This specifies a tab delimited column mapping state IDs to descriptive names to display on the heatmap. The

first column contains the state ID including the prefix and the second column contains the descriptive name.

- `-uniformscale` – This species the color scale should be uniform for the entire heatmap. The default is to use a column specific coloring scale.

14 If interested in computing additional neighborhood enrichments other than those that were already computed after applying the `LearnModel` command, then use the `NeighborhoodEnrichment` command by entering:

```
java -mx4000M -jar ChromHMM.jar NeighborhoodEnrichment
inputsegment anchorpositions outfileprefix
```

In this command `inputsegment` specifies the segmentation file for which neighborhood enrichments should be computed. `anchorpositions` specifies the directory containing the files for neighborhood enrichment analysis. These files should be prepared according to Step 9. `outfileprefix` contains the prefix including possibly directory, of the output files which will have the extensions `'.txt'`, `'.png'`, and `'.svg'` added to them (Box 1).

When executing the commands also add any appropriate optional parameters. Optional parameters are specified immediately after `NeighborhoodEnrichment.` The full set of options are discussed in the manual. Some options to note:

- `-b binsize` – This specifies the bin size and should match what was used in `LearnModel`.
- `-l numleftintervals` – This specifies the number of columns to the left of the anchor positions for which enrichments should be shown. The default is 10.
- `-r numrightintervals` – This specifies the number of columns to the right of the anchor positions for which enrichments should be shown. The default is 10.
- `-s spacing` – This specifies the spacing in base pairs at which enrichments should be displayed. The default is 200.

15 If desiring the states of the model to be in a different order, then reorder the states of the model by executing the `Reorder` command by entering:

```
java -mx4000M -jar ChromHMM.jar Reorder –o
stateorderingfile inputmodel outputdir
```

`inputmodel` is the model file produced by `LearnModel` that should be reordered. `outputdir` is the directory where the reordered model and emission and transition parameter files should be written. The option ‘ `-o stateorderingfile’` specifies a file which gives the state reordering. The format of the file is a two column tab delimited with the first column the old state number and the second column the new state number without the letter prefix. Additional options can be found in the user manual.

16 If the states of the model were reordered in Step 15 and to generate a segmentation corresponding to the reordered states, or desiring a segmentation for data not used in learning an existing model, but in the same format, then apply the `MakeSegmentation` command by entering:

```
java -mx4000M -jar ChromHMM.jar MakeSegmentation modelfile
inputdir outputdir
```

In the above `modelfile` is the file containing the model for which segmentation files should be produced. `inputdir` specifies the directory containing the binarized input data. `outputdir` specifies the directory where the segmentation files should go.

17 If desiring a change of state colors in the browser files use the `MakeBrowserFiles` command by entering:

```
java -mx4000M -jar ChromHMM.jar MakeBrowserFiles –c
colormappingfile segmentfile segmentationname
outputfileprefix
```

In the above `segmentfile` is the segmentation file in four column bed format. `segmentationname` is a name for the segmentation. `outputfileprefix` is the prefix including directory for the regenerated browser files. The extension `‘_browserexpanded.bed’` and `‘_browserdense.bed’` are appended to `outputfileprefix`.

When executing the commands also add any appropriate optional parameters. Optional parameters are specified immediately after `MakeBrowserFiles`. The full set of options are discussed in the manual. The option noted above: ‘ `-c colormappingfile’` specifies a file containing the state color mapping. The file is a two column tab delimited format. The first column gives the state number without a prefix and the second column the color in R,G,B format with values between 0 and 255.

TIMING

Steps 1–3, installation: 5 min

Steps 4–7, binarization: 1hr

Steps 8–9, file preparation for enrichment analysis: 30min

Steps 10–11, model learning: 4hr (highly variable)

Steps 12–17, post model learning analysis: 2hr

TROUBLESHOOTING

Step	Problem	Possible Reason	Solution
7	There are no present calls (1 values) in the binarized data	Peak calls were provided opposed to aligned reads and the '-peaks' flag was not specified	Use aligned reads or add the '-peaks' option
10	Error message that the Information initialization strategy can only support a certain number of states	The binarization might not have been done correctly and there are no present calls. Alternatively more states are being asked for than there are observed combination of marks	If there are no present calls see troubleshooting for step 7. If there are more states than combination of marks either reduce the number of states or instead use the Random initialization strategy
10	Error message indicating insufficient memory	Java does not have access to enough memory or there is insufficient memory available on the system	Increase the memory Java has access to by increasing the value associated with the '-mx' flag. Alternatively if using the '-p' flag set it to a small non-zero integer value to reduce the number of processors and thus memory used simultaneously at a cost of additional runtime.

ANTICIPATED RESULTS

The expected output of ChromHMM through the main 'LearnModel' command consists of a number of different files summarized in Box 2. These files are all linked to by an automatically generated webpage report file (Fig. 3). The output of other specific commands of ChromHMM are included in their description in Box 1.

Acknowledgments

We acknowledge the ENCODE and Roadmap Epigenomics Consortium for generating and processing data to which we have previously applied ChromHMM. We acknowledge users of ChromHMM that have provided useful feedback on the software. We acknowledge funding from the US National Institutes of Health grants U54HG004570, RC1HG005334 (M.K.), R01ES024995, U01HG007912 and U01MH105578 (J.E.), the US National Science Foundation Postdoctoral Fellowship #0905968 and CAREER Award #1254200 (J.E.), and an Alfred P. Sloan Fellowship (J.E.).

References

1. Ernst J, et al. Mapping and analysis of chromatin state dynamics in nine human cell types. Nature. 2011; 473:43–49. [PubMed: 21441907]
2. Maurano MT, et al. Systematic Localization of Common Disease-Associated Variation in Regulatory DNA. Science. 2012; 337:1190–1195. [PubMed: 22955828]
3. Roadmap Epigenomics Consortium et al. Integrative analysis of 111 reference human epigenomes. Nature. 2015; 518:317–330. [PubMed: 25693563]
4. Claussnitzer M, et al. FTO Obesity Variant Circuitry and Adipocyte Browning in Humans. N Engl J Med. 2015; 373:895–907. [PubMed: 26287746]
5. Barski A, et al. High-Resolution Profiling of Histone Methylations in the Human Genome. Cell. 2007; 129:823–837. [PubMed: 17512414]
6. Zhang Y, et al. Model-based Analysis of ChIP-Seq (MACS). Genome Biol. 2008; 9:R137. [PubMed: 18798982]

7. Ernst J, Kellis M. Discovery and characterization of chromatin states for systematic annotation of the human genome. Nat Biotechnol. 2010; 28:817–825. [PubMed: 20657582]

8. Ernst J, Kellis M. ChromHMM: automating chromatin-state discovery and characterization. Nat Methods. 2012; 9:215–216. [PubMed: 22373907]

9. Wang Z, et al. Combinatorial patterns of histone acetylations and methylations in the human genome. Nat Genet. 2008; 40:897–903. [PubMed: 18552846]

10. ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome. Nature. 2012; 489:57–74. [PubMed: 22955616]

11. Hoffman MM, et al. Integrative annotation of chromatin elements from ENCODE data. Nucleic Acids Res. 2013; 41:827–841. [PubMed: 23221638]

12. Ernst J, Kellis M. Large-scale imputation of epigenomic datasets for systematic annotation of diverse human tissues. Nat Biotechnol. 2015; 33:364–376. [PubMed: 25690853]

13. Mortazavi A, et al. Integrating and mining the chromatin landscape of cell-type specificity using self-organizing maps. Genome Res. 2013; 23:2136–2148. [PubMed: 24170599]

14. Javierre BM, et al. Lineage-Specific Genome Architecture Links Enhancers and Non-coding Disease Variants to Target Gene Promoters. Cell. 2016; 167:1369–1384.e19. [PubMed: 27863249]

15. Lorzadeh A, et al. Nucleosome Density ChIP-Seq Identifies Distinct Chromatin Modification Signatures Associated with MNase Accessibility. Cell Rep. 2016; 17:2112–2124. [PubMed: 27851972]

16. Yue F, et al. A comparative encyclopedia of DNA elements in the mouse genome. Nature. 2014; 515:355–364. [PubMed: 25409824]

17. Roy S, et al. Identification of Functional Elements and Regulatory Circuits by Drosophila modENCODE. Science. 2010; 330:1787–1797. [PubMed: 21177974]

18. Rosenbloom KR, et al. ENCODE Data in the UCSC Genome Browser: year 5 update. Nucleic Acids Res. 2013; 41:D56–D63. [PubMed: 23193274]

19. Cunningham F, et al. Ensembl 2015. Nucleic Acids Res. 2015; 43:D662–D669. [PubMed: 25352552]

20. Denholtz M, et al. Long-Range Chromatin Contacts in Embryonic Stem Cells Reveal a Role for Pluripotency Factors and Polycomb Proteins in Genome Organization. Cell Stem Cell. 2013; 13:602–616. [PubMed: 24035354]

21. Core LJ, et al. Analysis of nascent RNA identifies a unified architecture of initiation regions at mammalian promoters and enhancers. Nat Genet. 2014; 46:1311–1320. [PubMed: 25383968]

22. Wapinski OL, et al. Hierarchical Mechanisms for Direct Reprogramming of Fibroblasts to Neurons. Cell. 2013; 155:621–635. [PubMed: 24243019]

23. Chronis C, et al. Cooperative Binding of Transcription Factors Orchestrates Reprogramming. Cell. 2017; 168:442–459.e20. [PubMed: 28111071]

24. Pope BD, et al. Topologically associating domains are stable units of replication-timing regulation. Nature. 2014; 515:402–405. [PubMed: 25409831]

25. Ernst J, Kellis M. Interplay between chromatin state, regulator binding, and regulatory motifs in six human cell types. Genome Res. 2013; 23:1142–1154. [PubMed: 23595227]

26. Kheradpour P, et al. Systematic dissection of regulatory motifs in 2000 predicted human enhancers using a massively parallel reporter assay. Genome Res. 2013; 23:800–811. [PubMed: 23512712]

27. ENCODE Project Consortium et al. An integrated encyclopedia of DNA elements in the human genome. Nature. 2012; 489:57–74. [PubMed: 22955616]

28. Hibar DP, et al. Common genetic variants influence human subcortical brain structures. Nature. 2015; 520:224–229. [PubMed: 25607358]

29. Gjoneska E, et al. Conserved epigenomic signals in mice and humans reveal immune basis of Alzheimer/'s disease. Nature. 2015; 518:365–369. [PubMed: 25693568]

30. De Jager PL, et al. Alzheimer's disease: early alterations in brain DNA methylation at ANK1, BIN1, RHBDF2 and other loci. Nat Neurosci. 2014; 17:1156–1163. [PubMed: 25129075]

31. Frost B, Hemberg M, Lewis J, Feany MB. Tau promotes neurodegeneration through global chromatin relaxation. Nat Neurosci. 2014; 17:357–366. [PubMed: 24464041]

32. Parker SCJ, et al. Chromatin stretch enhancer states drive cell-specific gene regulation and harbor human disease risk variants. Proc Natl Acad Sci. 2013; 110:17921–17926. [PubMed: 24127591]

33. Taberlay PC, Statham AL, Kelly TK, Clark SJ, Jones PA. Reconfiguration of nucleosome-depleted regions at distal regulatory elements accompanies DNA methylation of enhancers and insulators in cancer. Genome Res. 2014; 24:1421–1432. [PubMed: 24916973]

34. Al-Tassan NA, et al. A new GWAS and meta-analysis with 1000Genomes imputation identifies novel risk variants for colorectal cancer. Sci Rep. 2015; 5:10442. [PubMed: 25990418]

35. Lay FD, et al. Reprogramming of the human intestinal epigenome by surgical tissue transposition. Genome Res. 2014; 24:545–553. [PubMed: 24515120]

36. Fiziev P, et al. Systematic Epigenomic Analysis Reveals Chromatin States Associated with Melanoma Progression. Cell Rep. 2017; 19:875–889. [PubMed: 28445736]

37. Kasowski M, et al. Extensive variation in chromatin states across humans. Science. 2013; 342:750–752. [PubMed: 24136358]

38. Brown EJ, Bachtrog D. The chromatin landscape of Drosophila: comparisons between species, sexes, and chromosomes. Genome Res. 2014; 24:1125–1137. [PubMed: 24840603]

39. Day K, et al. Differential DNA methylation with age displays both common and dynamic features across human tissues that are influenced by CpG landscape. Genome Biol. 2013; 14:R102. [PubMed: 24034465]

40. Horvath S. DNA methylation age of human tissues and cell types. Genome Biol. 2013; 14:3156.

41. Ward LD, Kellis M. HaploReg: a resource for exploring chromatin states, conservation, and regulatory motif alterations within sets of genetically linked variants. Nucleic Acids Res. 2012; 40:D930–934. [PubMed: 22064851]

42. Boyle AP, et al. Annotation of functional variation in personal genomes using RegulomeDB. Genome Res. 2012; 22:1790–1797. [PubMed: 22955989]

43. Day N, Hemmaplardh A, Thurman RE, Stamatoyannopoulos JA, Noble WS. Unsupervised segmentation of continuous genomic data. Bioinforma Oxf Engl. 2007; 23:1424–1426.

44. Thurman RE, Day N, Noble WS, Stamatoyannopoulos JA. Identification of higher-order functional domains in the human ENCODE regions. Genome Res. 2007; 17:917. [PubMed: 17568007]

45. Hoffman MM, et al. Unsupervised pattern discovery in human chromatin structure through genomic segmentation. Nat Methods. 2012; 9:473–476. [PubMed: 22426492]

46. Biesinger J, Wang Y, Xie X. Discovering and mapping chromatin states using a tree hidden Markov model. BMC Bioinformatics. 2013; 14:S4.

47. Yu P, et al. Spatiotemporal clustering of the epigenome reveals rules of dynamic gene regulation. Genome Res. 2013; 23:352–364. [PubMed: 23033340]

48. Marco E, et al. Multi-scale chromatin state annotation using a hierarchical hidden Markov model. Nat Commun. 2017; 8:15011. [PubMed: 28387224]

49. Roy S, Sridharan R. Chromatin module inference on cellular trajectories identifies key transition points and poised epigenetic states in diverse developmental processes. Genome Res. 2017; doi: 10.1101/gr.215004.116

50. Sohn KA, et al. hiHMM: Bayesian non-parametric joint inference of chromatin state maps. Bioinformatics. 2015; 31:2066–2074. [PubMed: 25725496]

51. Zhang Y, An L, Yue F, Hardison RC. Jointly characterizing epigenetic dynamics across multiple human cell types. Nucleic Acids Res. 2016; :gkw278.doi: 10.1093/nar/gkw278

52. Libbrecht MW, et al. Joint annotation of chromatin state and chromatin conformation reveals relationships among domain types and identifies domains of cell-type-specific expression. Genome Res. 2015; doi: 10.1101/gr.184341.114

53. Zacher B, Lidschreiber M, Cramer P, Gagneur J, Tresch A. Annotation of genomics data using bidirectional hidden Markov models unveils variations in Pol II transcription cycle. Mol Syst Biol. 2014; 10:768–768. [PubMed: 25527639]

54. Zacher B, et al. Accurate Promoter and Enhancer Identification in 127 ENCODE and Roadmap Epigenomics Cell Types and Tissues by GenoSTAN. PLOS ONE. 2017; 12:e0169249. [PubMed: 28056037]

55. Mammana A, Chung HR. Chromatin segmentation based on a probabilistic model for read counts explains a large portion of the epigenome. Genome Biol. 2015; 16:151. [PubMed: 26206277]

56. Song J, Chen KC. Spectacle: fast chromatin state annotation using spectral learning. Genome Biol. 2015; 16:33. [PubMed: 25786205]

57. Duttke SHC, et al. Human Promoters Are Intrinsically Directional. Mol Cell. 2015; 57:674–684. [PubMed: 25639469]

58. Filion GJ, et al. Systematic protein location mapping reveals five principal chromatin types in Drosophila cells. Cell. 2010; 143:212–224. [PubMed: 20888037]

59. Hamada M, Ono Y, Fujimaki R, Asai K. Learning chromatin states with factorized information criteria. Bioinformatics. 2015; 31:2426–2433. [PubMed: 25810430]

60. Jaschek, R., Tanay, A. Spatial clustering of multivariate genomic and epigenomic information. Springer; 2009. p. 183

61. Kharchenko PV, et al. Comprehensive analysis of the chromatin landscape in Drosophila melanogaster. Nature. 2011; 471:480–485. [PubMed: 21179089]

62. Larson JL, Huttenhower C, Quackenbush J, Yuan GC. A tiered hidden Markov model characterizes multi-scale chromatin states. Genomics. 2013; 102:1–7. [PubMed: 23570996]

63. Roudier F, et al. Integrative epigenomic mapping defines four main chromatin states in Arabidopsis: Organization of the Arabidopsis epigenome. EMBO J. 2011; 30:1928–1938. [PubMed: 21487388]

64. Won KJ, et al. Comparative annotation of functional regions in the human genome using epigenomic data. Nucleic Acids Res. 2013; 41:4423–4432. [PubMed: 23482391]

65. Zeng X, et al. jMOSAiCS: joint analysis of multiple ChIP-seq datasets. Genome Biol. 2013; 14:R38. [PubMed: 23844871]

66. Choi H, Fermin D, Nesvizhskii AI, Ghosh D, Qin ZS. Sparsely correlated hidden Markov models with application to genome-wide location studies. Bioinformatics. 2013; 29:533–541. [PubMed: 23325620]

67. Hon G, Ren B, Wang W. ChromaSig: A Probabilistic Approach to Finding Common Chromatin Signatures in the Human Genome. PLoS Comput Biol. 2008; 4:e1000201. [PubMed: 18927605]

68. Bernstein BE, et al. A Bivalent Chromatin Structure Marks Key Developmental Genes in Embryonic Stem Cells. Cell. 2006; 125:315–326. [PubMed: 16630819]

69. Thurman RE, et al. The accessible chromatin landscape of the human genome. Nature. 2012; 489:75–82. [PubMed: 22955617]

70. Boyle AP, et al. High-resolution genome-wide in vivo footprinting of diverse transcription factors in human cells. Genome Res. 2011; 21:456–464. [PubMed: 21106903]

71. Neph S, et al. An expansive human regulatory lexicon encoded in transcription factor footprints. Nature. 2012; 489:83–90. [PubMed: 22955618]

72. Ernst J, et al. Genome-scale high-resolution mapping of activating and repressive nucleotides in regulatory regions. Nat Biotechnol. 2016; 34:1180–1190. [PubMed: 27701403]

73. Landt SG, et al. ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. Genome Res. 2012; 22:1813–1831. [PubMed: 22955991]

74. Kent WJ, et al. The human genome browser at UCSC. Genome Res. 2002; 12:996–1006. [PubMed: 12045153]

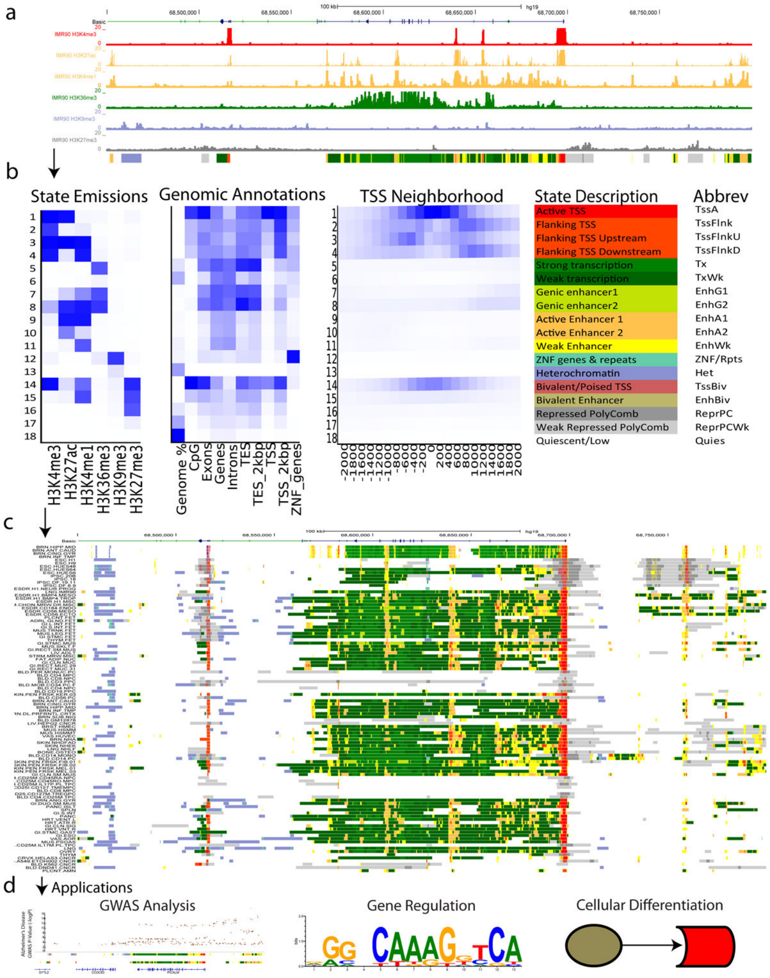
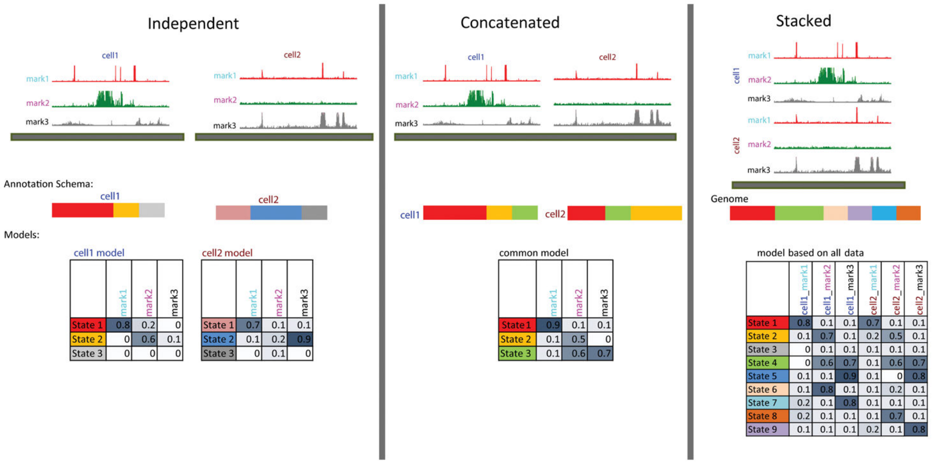


Figure 1. Overview of ChromHMM

(a) Tracks of multiple histone modifications are shown from one cell type, IMR90. From such types of tracks, ChromHMM learns a set of chromatin state definitions de novo, and then assigns each location in the genome to an instance of each state. The chromatin states assignments for IMR90 based on the model in (b) are shown below the histone modifications. **(b)** The panel displays on left a heatmap of emission parameters where each row corresponds to a different state and each column a different mark for the Roadmap Epigenomics 18-state expanded model defined based on the observed data for six histone modifications (H3K4me1, H3K4me3, H3K9ac, H3K27ac, H3K36me3, and H3K9me3) from Ref. [3]. The darker the blue color corresponds to a greater probability of observing the mark in the state. The heatmap to the right of the emission parameters displays the overlap enrichment for various external genomic annotations in IMR90 cells (epigenome E017) similar to what was previously shown for H1-hESC cells in Ref. [3]. A darker blue color corresponds to a greater fold enrichment for a column specific coloring scale. The heatmap to the right of that shows fold enrichment for each state for each 200bp bin position within 2kb around a set of transcription start sites (TSS). A darker blue color corresponds to a greater fold enrichment and there is one color scale for the entire heatmap. Shown to the right of that are candidate state descriptions for each state followed by a state mnemonic. **(c)**

The panel displays the browser view of ChromHMM genome annotation based on the model in (b), which was defined across 98 cell and tissue types[3]. Each row below the genes corresponds to one of the cell or tissue type. **(d)** The panel highlights application areas of ChromHMM, which include GWAS analysis, gene regulation, and cellular differentiation among others. The GWAS example shows overlap between chromatin state annotations and single-nucleotide polymorphisms (SNPs) associated with Alzheimer's Disease (AD) similar to Ref. [29]. Roadmap epigenomics chromatin state annotations based on the model in (b) for primary monocytes cells (E029) and below it brain hippocampus (E071).



cell1 model

	mark1	mark2	mark3
State 1	0.8	0.2	0
State 2	0	0.6	0.1
State 3	0	0	0

cell2 model

	mark1	mark2	mark3
State 1	0.7	0.1	0.1
State 2	0.1	0.2	0.9
State 3	0	0.1	0

common model

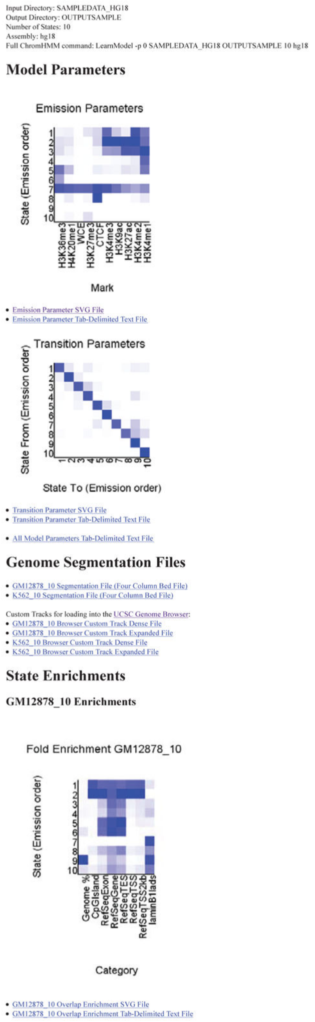
	mark1	mark2	mark3
State 1	0.9	0.1	0.1
State 2	0.1	0.5	0
State 3	0.1	0.6	0.7

model based on all data

	cell1_mark1	cell1_mark2	cell1_mark3	cell2_mark1	cell2_mark2	cell2_mark3
State 1	0.8	0.1	0.1	0.7	0.1	0.1
State 2	0.1	0.7	0.1	0.2	0.5	0.1
State 3	0	0.1	0.1	0.1	0.1	0.1
State 4	0	0.6	0.7	0.1	0.6	0.7
State 5	0.1	0.1	0.9	0.1	0	0.8
State 6	0.1	0.8	0.1	0.1	0.2	0.1
State 7	0.2	0.1	0.8	0.1	0.1	0.1
State 8	0.2	0.1	0.1	0.1	0.7	0.1
State 9	0.1	0.1	0.1	0.2	0.1	0.8

Figure 2. Overview of Different Options for Handling Multiple Cell Types in ChromHMM

(Left) Multiple cell types are treated independently leading to a different model and annotation for each cell type. **(Center)** Multiple cell types are effectively concatenated leading to one shared model for all cell types but cell type specific annotations. **(Right)** Data from multiple cell types are stacked leading to one model based on an expanded set of features and one annotation of the genome.



Input Directory: SAMPLEDATA_HG18
Output Directory: OUTPUTSAMPLE
Number of States: 10
Assembly: hg18
Full ChromHMM command: LearnModel -p 0 SAMPLEDATA_HG18 OUTPUTSAMPLE 10 hg18

Model Parameters

- Emission Parameter SVG File
- Emission Parameter Tab-Delimited Text File

- Transition Parameter SVG File
- Transition Parameter Tab-Delimited Text File

- All Model Parameters Tab-Delimited Text File

Genome Segmentation Files

- GM12878_10 Segmentation File (Four Column Bed File)
- K562_10 Segmentation File (Four Column Bed File)

Custom Tracks for loading into the UCSC Genome Browser:
- GM12878_10 Browser Custom Track Dense File
- GM12878_10 Browser Custom Track Expanded File
- K562_10 Browser Custom Track Dense File
- K562_10 Browser Custom Track Expanded File

State Enrichments

GM12878_10 Enrichments

- GM12878_10 Overlap Enrichment SVG File
- GM12878_10 Overlap Enrichment Tab-Delimited Text File

Figure 3. Example webpage screenshot

The figure displays a screenshot of a portion of the webpage automatically generated by the ChromHMM `LearnModel` command on the sample data. The webpage contains images and links to the files generated by ChromHMM.

Table 1

Summary of Some Major Available ChromHMM Reference Models and Annotations Defined Across Multiple Cell and Tissue Types.

Model	Input Marks	Cell/Tissue Types	Assembly	# of States	Ref.	URLs
Roadmap Epigenomics Core Model	5-marks observed human data (H3K4me1, H3K4me3, H3K27me3, H3K9me3, H3K36me3)	127 diverse ones (111 Roadmap, 16 ENCODE)	hg19	15	3	http://compbio.mit.edu/roadmap https://www.encodeproject.org http://epigenomegateway.wustl.edu/ https://genome.ucsc.edu/ (Roadmap Epigenomics Integrative Analysis Hub)
Roadmap Epigenomics Expanded Model	6-marks observed human data (H3K4me1, H3K4me3, H3K27me3, H3K9me3, H3K36me3, H3K27ac)	98 diverse ones (82 Roadmap, 16 ENCODE)	hg19	18	3	http://compbio.mit.edu/roadmap https://www.encodeproject.org http://epigenomegateway.wustl.edu/ https://genome.ucsc.edu/ (Roadmap Epigenomics Integrative Analysis Hub)
Roadmap Epigenomics Imputation Based Model	12-marks imputed human data (H3K4me1, H3K4me3, H3K27me3, H3K9me3, H3K36me3, H3K27ac, H3K9ac, H4K20me1, H3K79me2, H3K4me2, H2A.Z, UW DNaseI)	127 diverse ones (111 Roadmap,16 ENCODE)	hg19	25	12	http://compbio.mit.edu/roadmap http://epigenomegateway.wustl.edu/ https://genome.ucsc.edu/ (Roadmap Epigenomics Integrative Analysis Hub)
ENCODE Integrative Analysis ChromHMM Model	14 marks observed human data (H3K4me1, H3K4me3, H3K27me3, H3K36me3, H3K27ac, H3K9ac, H4K20me1, H3K4me2, Pol2, CTCF, FAIRE-seq, UW DNaseI, Duke	6 ENCODE Cell Types	hg19	25	11, 25	https://genome.ucsc.edu/cgi-bin/hgTrackUi?g=wgEncodeAwgSegmentation&db=hg19

Model	Input Marks	Cell/Tissue Types	Assembly	# of States	Ref.	URLs
	DNaseI, Input)					
Ernst et al, 2011 *Nature* Model	10 marks observed human data (H3K4me1, H3K4me3, H3K27me3, H3K36me3, H3K27ac, H3K9ac, H4K20me1, H3K4me2, CTCF, Input)	9 ENCODE Cell Types	hg18 and hg19 liftover	15	1	https://genome.ucsc.edu/cgi-bin/hgTrackUi?g=wgEncodeBroadHmm&db=hg19 https://www.encodeproject.org
Ensembl Human Models	Minimum of 6-marks observed human data (H3K4me1, H3K4me3, H3K27me3, H3K9me3, H3K36me3, and H3K27ac)	74 Cell Types from Blueprint, Roadmap Epigenomics, and ENCODE	hg38	25	14, 19	http://ensembl.org/info/genome/funcgen/regulatory_segmentation.html ftp://ftp.ensembl.org/pub/release-87/data_files/homo_sapiens/GRCh38/segmentation_file/085/
Mouse ENCODE	4-marks observed mouse and human data (H3K4me1, H3K4me3, H3K36me3, H3K27me3)	15 mouse ENCODE cell types and 6 ENCODE cell types	mm9	7	16	https://www.encodeproject.org http://main.genomebrowser.bx.psu.edu/cgibin/hgTrackUi?g=meryChromHmm7s&db=mm9