Semi-automated genome annotation using epigenomic data and Segway

Eric G. Roberts[1], Mickaël Mendez[1,2], Coby Viner[1,2], Mehran Karimzadeh[1,3], Rachel C.W. Chan[1,2], Rachel Ancar[4], Davide Chicco[1], Jay R. Hesselberth[4], Anshul Kundaje[5,6], Michael M. Hoffman[1,2,3,7]

[1] Princess Margaret Cancer Centre, University Health Network, Toronto, Ontario, Canada; [2] Department of Computer Science, University of Toronto, Toronto, Ontario, Canada; [3] Department of Medical Biophysics, University of Toronto, Toronto, Ontario, Canada; [4] Department of Biochemistry and Molecular Genetics, University of Colorado Denver, Aurora, Colorado, USA. [5] Department of Computer Science, Stanford University, Stanford, California, USA; [6] Department of Genetics, Stanford University, Stanford, California, USA. [7] Vector Institute for Artificial Intelligence; Correspondence should be addressed to M.M.H. (michael.hoffman@utoronto.ca).

Keywords

Annotation, Chromatin, Chromatin analysis, Epigenomics, Software, Gene regulation, Dynamic Bayesian network

Authors

Hoffman Lab

Website: hoffmanlab.org

Address:

Toronto Medical Discovery Tower 11-401

101 College Street

Toronto, ON M5G 1L7

Canada

Telephone: +1 416 634 8736

Eric G. Roberts, eroberts@uhnresearch.ca

Mickaël Mendez, mendez.mickael@gmail.com

Coby Viner, cviner@cs.toronto.edu

Mehran Karimzadeh, mehran.karimzade@gmail.com

Rachel Chan, rachelchan@cs.toronto.edu

Davide Chicco, davide.chicco@gmail.com

Michael M. Hoffman

Address:

Toronto Medical Discovery Tower 11-311

101 College Street

Toronto, ON M5G 1L7

Canada

michael.hoffman@utoronto.ca

Telephone: +1 416 581 8789

Hesselberth Lab

Website: hesselberthlab.github.io

Address:

University of Colorado School of Medicine

12801 E 17TH Ave

RC1 South 10104

Aurora CO 80045

USA

Telephone: +1 303 724 5486

Rachel Ancar, rachel.ancar@ucdenver.edu

Jay R. Hesselberth

jay.hesselberth@gmail.com

Telephone: +1 303 724 5384

Kundaje Lab

Website: sites.google.com/site/anshulkundaje

Anshul Kundaje

Address:

Department of Genetics

300 Pasteur Dr, Lane Building, L301

Stanford, CA 94305-5120

USA

akundaje@stanford.edu

Telephone: +1 650 723 2353

Biochemical techniques measure many individual properties of chromatin along the genome. These properties include DNA accessibility (measured by DNase-seq) and the presence of individual transcription factors and histone modifications (measured by ChIP-seq). Segway is software that transforms multiple datasets on chromatin properties into a single annotation of the genome that a biologist can more easily interpret. This protocol describes how to use Segway to annotate the genome, starting with reads from

a ChIP-seq experiment. It includes pre-processing of data, training the Segway model, annotating the genome, assigning biological meanings to labels, and visualizing the annotation in a genome browser. Segway is unique in its use of a modifiable Dynamic Bayesian Network which allows it to handle missing data, model length constraints for annotated regions, supervise regions, and uses custom graphical models for specific project needs. Additionally, Segway defaults to a single base-pair analysis on the genome to precisely predict the underlying data distribution and the resulting annotation. This protocol takes less than 8 hours including data preparation, computation, analysis and visualization.

INTRODUCTION

Segway[1,2] is software that discovers patterns in genomic signal datasets, and then transforms multiple datasets into a simple annotation, labeling the best pattern at every position in the genome. Each input data set comes from a biochemical technique that measures some property along the genome. Often the property relates to local chromatin biology, such as DNA accessibility (measured by DNase-seq[3,4] or ATAC-seq[5]) and the presence of individual transcription factors and histone modifications (measured by ChIP-seq[6]). The input data could, however, include any property quantified along the genome in a locus-specific manner. Given some genome-aligned datasets, Segway constructs a statistical model of recurring patterns across these datasets. In the model, every base has a hidden *label* that determines which pattern is generated at that position. Then, Segway uses that model to annotate the whole genome automatically with the best label for every position. Finally, supporting tools visualize and summarize the model and annotation to reveal how these patterns associate with known and novel biological phenomena.

Motivation

Researchers often have multiple functional genomic datasets that they wish to understand. While analysts have a rich choice of peak-calling methods[7–9], post hoc comparisons of peak calls are unwieldy, at best. At worst, they have decreased power to detect phenomena associated with low signal in a single dataset that are revealed as significant when we jointly consider multiple datasets. To discover more potentially significant regions, one needs a method of integrative analysis across multiple datasets.

To perform integrative analysis on unprecedented quantities of genomic signal data, researchers in the ENCODE Pilot Project[10] developed the first semi-automated genome annotation method, HMMSeg[11,12]. Semi-automated genome annotation jointly analyzes multiple datasets in an unsupervised fashion, allowing the discovery of both known and novel patterns. It usually works by creating a *segmentation*, which is an annotation that has one label at every position. Since the ENCODE Pilot, multiple semi-automated genome annotation methods have been developed[13–18], including HMMSeg's successor, Segway. Segway is one of the most powerful methods for semi-automated genome annotation methods, capable of analyzing multiple datasets at 1–base-pair resolution, handling heterogeneous patterns of missing data, and modeling signal level directly rather than binarizing.

Signal data usually comes from sequencing assays with a technical resolution of 1 bp, but positional stochasticity introduced in the standard ChIP-seq process means a slightly lower effective resolution. Thus, we recommend using 10 bp in this protocol. Techniques like ChIP-exo[19] and ChIP-nexus[20], however, can deliver effective resolutions of up to 1 bp. Additionally, open chromatin assays like DNase (cite both Crawford and Stam method papers) or ATAC-seq

(cite Greenleaf paper) can reflect footprints at 1 bp resolution, and other work shows Segway's use to identify these footprints[21].

Various combinations of multiple functional genomic datasets have been used to model specific genomic features. The Segway method was used to build a model for predicting transcript start sites using a mixture of gaussians and virtual evidence. For transcription start sites, input datasets of FANTOM5 CAGE, various histone marks, and DNase were selected (cite https://www.biorxiv.org/content/10.1101/2020.01.30.926923v1.full.pdf+html). SegRNA (cite https://www.biorxiv.org/content/10.1101/2020.07.28.225193v1.full.pdf+html), was developed using Segway to characterizing patterns of a cell type's transcriptome using stranded aware model using the available concatenation option. For this model PRO-seq, RNA-seq, and CAGE datasets were integrated. ~~When attempting to model patterns of genomic datasets, the interpretation and quality of the resulting model depends on the data selection, it's quality, and your model parameters.~~

Comparison of Segway to similar methods

Segway in unique in its use of a Dynamic Bayesian Network. This model allows for more nuanced tuning of genomic state parameters. The Segway model can transparently predict over missing data, have minimum and maximum length state lengths, specify a fixed length on which state transitions must occur, have states supervised for a given label, and amongst other various features have any other graphical model feature a sufficiently advanced user may supply themselves. Segway was found to have similar performance for equivalent data resolutions with other genome annotation software[22] such as ChromHMM[13]. Additionally, there

have been various comparisons between the Segway model[22] and other genome annotation software[23] that have certain computational[24] or domain specialties (cite hierarchical annotation software?).

Limitations of Segway

To use experimental data with Segway efficiently it must be in a Genomedata file format. Segway cannot work directly with aligned reads. It is highly recommended to process experimental data into a signal enrichment format. For a given genome annotation project, the exact parameters needed require basic experience with machine learning techniques and the type of analysis. For example, Segway cannot guess the best number of genomic labels for a given set of data. In the resulting annotation, Segway does not assign biological meaning to the resulting labels. Segway instead relies on other tools, such as Segtools, to help analyze the resulting learned parameters on labels and enrichment analysis for each label on existing reference datasets. These tools allow an experienced biologist to assign biological meaning to each label. For annotations of gene regulation this process has been fully automated[25].

Experimental design

In this protocol, we focus on how to use Segway to create a simple regulatory annotation for a given cell type by selecting datasets from histone marks and transcription factors. This protocol also provides simple data interpretation for the resulting annotation resulting in visualizations, gene enrichment, and a list of gene IDs given a gene annotation source. For a regulatory annotation, these steps are considered suggestions and the specific analysis will depend on the outcome needed for the experimenter and for modeling different sets of genomic features. The process has the following major steps (Figure 1):

1. Create bedGraph[26] signal from aligned reads from ChIP-seq data or download signal data from an existing project.
2. Create Genomedata[27] archives containing the signal data.
3. Train the Segway model.
4. Use Segway to produce an annotation in browser-extensible data (BED) format[28].
5. View and analyze the resulting annotation.

In the PROCEDURE section we illustrate how to perform these steps both for any set of data and specifically on five ChIP-seq experiments for a human B-cell lymphoma cell line, DOHH2[29]. One can apply Segway to any number of signal datasets but we recommend using at least two. Obviously, Segway cannot find combinatorial patterns in single datasets, and purpose-made tools such as peak callers perform better at this task. Data from open chromatin assays such as DNase-seq or ATAC-seq proves particularly informative in Segway analyses, and one can often generate in the laboratory more easily. For simplicity we focus on ChIP-seq datasets here. It is important to ensure that any ChIP-seq data used is of sufficient quality. Suggestions on assessing the quality of these datasets is provided in **Box 4** | Quality control for ChIP-seq data.

The set of ChIP-seq targets we use here—H3K4me1, H3K4me3, H3K27ac, H3K27me3, and CTCF—prove sufficient to identify the most interesting recurring patterns and provide a useful baseline for epigenomic regulatory characterization. If we only had the resources for two ChIP-seq targets, H3K27ac (an “activating” mark) and H3K27me3 (a “repressive” mark) seem most useful. The other three datasets provide the means for identifying particular kinds of epigenomic patterns such as insulators (CTCF) or distinguishing between promoters (H3K4me3) and enhancers (H3K4me1). With more resources, one might discover a greater diversity of patterns adding in H3K36me3 (often found at transcribed gene bodies), or H3K9me3 (constitutive heterochromatin). With this protocol, adding ChIP-seq data for other transcription factors is

unlikely to contribute much to results, although an annotation trained only with transcription factor data might prove interesting.

The ChIP-seq targets for this protocol are filtered by a multi-read mappability score. The Umap[30] project provides lists of uniquely mappable regions for various assemblies and different genomic dataset read lengths. The files from the datasets contain a mappability score which is the probability that a randomly selected read of a fixed length in a given region is uniquely mappable. Smaller scores indicate less confidence about the observed ChIP-seq signal and are removed from the analysis.

In order to efficiently store and improve speed of analysis, the datasets for these targets are stored in a Genomedata[31] archive. These archives are designed for dense genomic signal data and allow for efficient random access. Genomedata archives also provide information on where data is effectively present using a reference sequence or as used in this protocol, contig (cite?) locations using AGP (cite) files. In order to map chromosome identifiers used by NCBI for use in visualization in the UCSC Genome Brower, an assembly report (cite?) provided by NCBI is used for translational purposes.

To further reduce noise and unnecessary computation we can remove problematic regions from the hg38 reference genome. A blacklist by ENCODE and a list of hard-masked cognates and pseudoautosomal regions provided by the Genome Reference Consortium (cite) are excluded from model training and from the resulting annotation. Functional genomic experiments often produce artifact signal in certain regions of the genome. If there is a curated list of these blacklisted regions for the genome and assembly you are using in your annotation, we

recommend excluding these regions from your analysis. Additionally, the masked regions have unmasked exact sequence duplicates elsewhere in the genome. Data will not align to the masked regions and may be removed from the analysis. For more details regarding the masked regions, a document is available online (http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.28_GRCh38.p13/GRCh38_major_release_seqs_for_alignment_pipelines/README_analysis_sets.txt).

The training process automatically discovers recurring patterns in the signal data you supplied. Training relies on an expectation-maximization[38] (EM) process that seeks a local maximum likelihood. The likelihood is the probability of generating the given data from the model and its learned parameters. Segway can optimize from multiple sets of initial values simultaneously. Each simultaneous training *instance* results in locally optimized parameters and Segway picks the winner with the best likelihood.

For increased computational speed, we can train on only a fraction of the genome. For this experiment, the *minibatch* feature is employed specifying which fraction of your data you wish to use. The minibatch feature uses a different randomly selected part of the genome in each training round. In Segway we set the number of training rounds with the `--max-train-rounds`. In most cases, the patterns found after five rounds are quite similar to those after 100. To increase the speed of this experiment, we will set the maximum number of rounds to 10. Here, we can also speed up training by reducing the resolution of the signal data with the `--resolution` option and it is increased from a default resolution of 1 base pair to 10.

To aid the experimenter in understanding their segmentation results, Segtools[39] (segtools.hoffmanlab.org) is a collection of command-line tools that enables exploratory data analysis of genome segmentations, such as the output of Segway. Each tool provides distinct information such as the distribution of segment lengths. Existing standard annotations of gene elements such as those provided by GENCODE can be used to calculate enrichment for a given labels and give us a list of gene IDs for labels of interest. Combining Segtools plots and analysis enables you to assign a biological meaning to each annotation label.

To visualize genomic regions of interest, The UCSC Genome Browser can load signal tracks and segmentations. To visualize all used and created datasets, a track hub[41] is created —a collection of genome annotation files on a web server. This allows for example to visualize, compare and functionally annotate Segway annotations (**Box 6**).

Existing Segway annotations

Segway has already produced a number of useful segmentations that are freely available for downloading or viewing in a genome browser. Several of these are available from the Segway website (segway.hoffmanlab.org). The Ensembl Regulatory Build[32] has Segway annotations of chromatin state across 74 cell types. You can view the Regulatory build segmentations both in Ensembl[33] and in the UCSC Genome Browser[34].

Adapting Segway to other tasks

While most published examples of Segway's use involve semi-automated genome annotation of chromatin state, it is highly adaptable. One can perform semi-automated genome annotation on any kind of genomic signal data. Since you can also supply an arbitrary Graphical Model Toolkit[35] (GMTK) dynamic Bayesian network (DBN) model, you can also use Segway as a framework for various different inferences on genomic signal data[36].

Primary audience

This protocol was designed for bioinformaticians and other biologists who wish to produce genomic annotations automatically. The signal data can come from public resources, such as ENCODE, or from your own experiments. You should have Linux experience.

MATERIALS

EQUIPMENT AND SOFTWARE

- Linux server or workstation
- At least 15 GB of free disk space
- At least 6 GB of memory
- Internet connection
- A Bash shell
- (Recommended) Debian 8, Red Hat Enterprise Linux 7, or CentOS 7

- (Recommended) Cluster system running Grid Engine, SLURM, IBM Platform Load Sharing Facility (LSF), Portable Batch System (PBS), or Torque

Required data

- ChIP-seq read alignments in Binary Alignment/Map (BAM) format (**Box 2**) or ChIP-seq sequence data tracks from a public source such as ENCODE (**Box 3**).

PROCEDURE

Load signal into Genomedata archives • Timing < 2.5 h

1| Install the prerequisite software in **Box 1**.

2| Download *assembly golden path* files for your genome. To download the human genome assembly version GRCh38/hg38, execute:

```
wget --recursive --no-directories --no-parent --accept '*chr*.agp.gz' --reject
'*comp.agp.gz'
ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.28_GRCh38.p13/
GCA_000001405.28_GRCh38.p13_assembly_structure/Primary_Assembly/assembled_chromosom
es/AGP/
```

⚠ Critical Step

The genome assembly version used to align all your signal data must match each other and the assembly chosen in this step. Always check that assembly versions match and never assume. Failing to ensure consistency will yield nonsensical results from Segway, or any

other genome analysis software. There is no guarantee that Genomedata will warn of data in unexpected positions.

3| Download DOHH2 cell line signal data tracks CTCF, H3K4me1, H3K4me3, H3K27ac, H3K27me3 as outlined in **Box 3** or generate the signal files from raw reads as outlined in **Box 2.**

4| Download the list of uniquely mappable regions from the Umap project for your genome and corresponding to the closest read length of your data, (use the smaller read length in case of ties). For the DOHH2 cell line signal data from ENCODE with read smallest read length of 36 base pairs, execute:

```
wget https://bismap.hoffmanlab.org/raw/hg38/k36.umap.bedgraph.gz
```

5|

Filter the Umap file for uniquely mappable regions with a multi-read mappability score greater or equal to 0.75. Execute:

```
zcat k36.umap.bedgraph.gz | awk 'BEGIN {FS=OFS="\t"} {if ($4 >= .75) print $1, $2, $3}' | bedtools merge > k36_umap_multiread_filtered.bed
```

⚠ **Critical Step**

Signal files downloaded from ENCODE (from **Box 1**), or generated from **Box 2** do not distinguish true zero-valued regions from unmappable regions. MACS2[8] does not distinguish true zero-values from missing data in its output. Segway attempts to accurately and separately model missing data versus zero-valued data. Ideally, your signal files should contain missing data when data is actually missing (by omitting the data) and zero-valued data when there is a known zero value result. If it is impossible to distinguish between zero-valued data and missing data, we recommend setting all missing data to zero. Avoid removing zero-valued data, if possible.

6| Download the GRCh38 assembly report. Execute:

```
wget
ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.28_GRCh38.p13/GCA_000001405.28_GRCh38.p13_assembly_report.txt
```

7| Create Genomedata archives containing your reference assembly and signal tracks. Specifically, create an individual archive for each individual signal track. To create a Genomedata archive with the GRCh38 human assembly from NCBI the signal files from **Step 3**, execute:

```
genomedata-load --assembly --sequence 'chr*.agp.gz' –track
H3K4me1=ENCFF509XSM_DOHH2_H3K4me1.bigWig --maskfile k36_umap_multiread_filtered.bed
ENCFF509XSM_DOHH2_H3K4me1.genomedata
```

This will result in a file called `ENCFF509XSM_DOHH2_H3K4me1.genomedata` containing signal from `ENCFF509XSM.bigWig` masked by regions contained in

`k36.umap_multiread_filtered.bed` and assembly data from the given AGP files. The `--track` option from the previous command allows one to assign a name to a track for a given signal file. In this case, it assigns the track name of `H3K4me1` to the signal data found in `ENCFF509XSM.bigWig`. If possible, it is recommended that you create your archives simultaneously.

8| Create unique named Genomedata archives for the remaining signal files by repeating **Step 7** for each remaining signal file gathered in **Step 3**. If you do this with the suggested datasets, you will have five new Genomedata directories containing the signal data (`ENCFF509XSM_DOHH2_H3K4me1.genomedata`, `ENCFF745GML_DOHH2_H3K4me3.genomedata`, `ENCFF592CSV_DOHH2_H3K27me3.genomedata`, `ENCFF890NAY_DOHH2_H3K27ac.genomedata`, and `ENCFF884IIL_DOHH2_CTCF.genomedata`).

Train Segway model from data • Timing < 30 min

9| (Optional) Set a random seed. Segway optimizes model parameters from randomly selected initial values. Usually it is better to let this random selection work unconstrained, but to reproduce the **ANTICIPATED RESULTS** here exactly, you must ensure the same sequence of random numbers. Do this by setting the `SEGWAY_RAND_SEED` to a positive (32 bit) integer. For example, to replicate **ANTICIPATED RESULTS**, execute:

```
export SEGWAY_RAND_SEED=22426492
```

? TROUBLESHOOTING

10| (Optional) Store the number of effective cores on your machine. This number is the product of the number of processors and each processor's number of Central Processing Unit (CPU) cores (Figure 2).

Export an environment variable containing the maximum available number of cores to use for other aspects of the protocol, execute:

```
export NUM_THREADS=$(getconf _NPROCESSORS_ONLN)
```

11| (Optional) Limit Segway's processor usage. To do this, set the `SEGWAY_NUM_LOCAL_JOBS` environment variable to the maximum number of processes you wish Segway to use. Smaller values for `SEGWAY_NUM_LOCAL_JOBS` will result in slower running times and therefore the protocol will take longer to perform. This step only applies to users who run Segway without a cluster environment such as Grid Engine[37], and we recommend using such an environment if possible (**Box 5**).

On a cluster, use instead the number of slots allocated to your job. Set `SEGWAY_NUM_LOCAL_JOBS` using the value from **Step 10** by executing:

```
export SEGWAY_NUM_LOCAL_JOBS="$NUM_THREADS"
```

12| (Recommended, Optional) Download a list of masked regions from the Genome Reference Consortium (https://www.ncbi.nlm.nih.gov/grc) to exclude from the analysis. To download the masked regions into a 0-based BED file format, execute:

```
wget –q –O -
ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.28_GRCh38.p13/GRCh38_major_release_seqs_for_alignment_pipelines/unmasked_cognates_of_masked_CEN_PAR.txt | tail -n +2 | awk –v OFS='\t' '{ print $2,$3-1,$4 }' > GRCh38_masked_cognates.bed
```

13| (Recommended, Optional) Merge a blacklist for your genome assembly into your exclude coordinates from **Step 12**. To download and merge a blacklist for hg38 provided by ENCODE into a single exclude_coords.bed file, execute:

```
wget https://www.encodeproject.org/files/ENCFF356LFX/@@download/ENCFF356LFX.bed.gz -q -O - | zcat | sort -k 1,1 -k 2,2n GRCh38_masked_cognates.bed - | bedtools merge > exclude_coords.bed
```

TODO: Add note saying the blacklist is necessary for ANTICIPATED RESULTS?

14| Train a 10-label Segway model using 1% of your genome. Create a 10-label model training on 1% of the genome from the archives created from **Steps 7 and 8**, excluding regions from **Steps 12 and 13**, at 10 base pair resolution, using 10 simultaneous training instances by executing:

```
segway train --resolution=10 --num-instances=10 --minibatch-fraction=0.01 --num-labels=10 --max-train-rounds=10 --exclude-coords=exclude_coords.bed ENCFF509XSM_DOHH2_H3K4me1.genomedata ENCFF745GML_DOHH2_H3K4me3.genomedata ENCFF592CSV_DOHH2_H3K27me3.genomedata ENCFF890NAY_DOHH2_H3K27ac.genomedata ENCFF884IIL_DOHH2_CTCF.genomedata train_results
```

Segway prints a log of genomic regions it trains on and individual training jobs run on your cluster or in local mode (Figure 3).

? TROUBLESHOOTING

Annotate the genome using the trained model • Timing < 1.5 h

15| Annotate the genome using the trained model from **Step 14**. The `train_results` directory contains the final model and trained parameters. To annotate the whole genome from our previously trained model, excluding regions from **Steps 12 and 13**, execute:

```
segway annotate --exclude-coords="exclude_coords.bed" --
bigBed=annotate_results/segway.layered.bb ENCFF509XSM_DOHH2_H3K4me1.genomedata
ENCFF745GML_DOHH2_H3K4me3.genomedata ENCFF592CSV_DOHH2_H3K27me3.genomedata
ENCFF890NAY_DOHH2_H3K27ac.genomedata ENCFF884IIL_DOHH2_CTCF.genomedata
train_results annotate_results
```

Segway prints a log of genomic regions it will annotate and individual identification jobs run on your cluster or in local mode (**Figure 4**).

Segway writes its annotation to a BED file inside the “annotate” directory (`annotate_results`), named `segway.bed.gz`. This is a tab-delimited file describing the chromosome regions and their corresponding label number (**Figure 5**).

Analyze the annotation using Segtools • Timing < 30 min

16| Plot the emission parameters learned during the training task performed in the **Step 14**.

```
segtools-gmtk-parameters train_results/params/params.params
```

This creates the gmtk-parameters directory that contains a heatmap (`gmtk_parameters.stats.png`) showing the learned parameters per label-track pairs. You can run this command directly after the training task.

? TROUBLESHOOTING

17| Calculate and plot the length distribution of segments in each label, and the genomic fraction covered by each label using `segtools-length-distribution`:

```
segtools-length-distribution annotate_results/segway.bed.gz
```

This creates the length_distribution directory that contains summary statistics in tab-delimited format (length_distribution.tab and segment_sizes.tab) and two plots. The first plot (length_distribution.png) shows the distribution of segment lengths for each label.

The second plot (segment_sizes.png) shows the fraction of total segments for each label and the fraction of genomic bases covered by each label.

18| Calculate the enrichment of each segment label over a gene annotation using `segtools-aggregation`. Download a gene annotation for your given assembly. In this example, download the GRCh38/hg38 human gene annotation in Gene Transfer Format (GTF) from GENCODE[40]:

```
wget
```

```
ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_36/gencode.v36.annotation.gtf.gz
```

Calculate and plot the aggregation:

```
segtools-aggregation --mode=gene --normalize --outdir aggregate_gene
annotate_results/segway.bed.gz gencode.v36.annotation.gtf.gz
```

This command runs `segtools-aggregation` in “gene” mode and creates two figures showing the enrichment of a segmentation over an idealized transcriptional (`aggregate_gene/feature_aggregation.splicing.png`) and translational (`aggregate_gene/feature_aggregation.translation.png`) gene model.

19 | Create a filtered GENCODE annotation list with only genes

```
zcat gencode.v36.annotation.gtf.gz | sed '/\(gene\t\|^#\)/!d' >
gencode.v36.genes.gtf
```

20| Create a list of GENCODE genes that overlap with each label from the produced annotation in **Step 15**.

```
for label in {0..10}; do zcat annotate_results/segway.bed.gz | awk --assign
label=$label 'BEGIN{OFS="\t"} { if ($4==label) print $1,$2,$3 }' | bedtools
intersect -a gencode.v36.genes.gtf -b stdin | sed 's/.*gene_id "\([^"]\+\).*/\1/' |
uniq > "segway.${label}.gencode.v36.genes.bed" ; done
```

Visualize signal data and segmentation on the UCSC Genome Browser •

Timing < 2 h

To visualize our segmentation and signal data, make a track hub and visualize it on the UCSC Genome Browser[34]

21| Create the directory hierarchy for the track hub:

```
mkdir -p trackhub/hg38
```

This creates the main directory `trackhub` that will contain all the information necessary for the UCSC Genome Browser to locate your data. The `hg38` subdirectory will contain your Umap-filtered signal tracks and segmentations generated based on the GRCh38/hg38 genome assembly.

22| Create a `hub.txt` file in the `trackhub` directory and add the following lines to the beginning of the file:

```
hub DOHH2_ChIP-seq
shortLabel DOHH2 ChIP-seq
longLabel Segway annotation of DOHH2 ChIP-seq data
genomesFile genomes.txt
email your.email@example.com
```

This file describes the general properties of your track hub where the first word of each line is the name of the property and the rest of the line is the value assigned to it.

23| Create a `genomes.txt` file in the `trackhub` directory file describing the genome assembly and the path to the track property file `trackDb.txt` for that genome assembly:

```
genome hg38

trackDb hg38/trackDb.txt
```

24| Copy the signal files in bigWig format to the `trackhub/hg38` directory with the `cp` command:

```
cp *.bigWig trackhub/hg38
```

The command above copies all the files with the extension `bigWig`

25| Move the layered segmentation generated in **Step 15** to the trackhub/hg38 directory.

Execute:

```
mv annotate_results/segway.layered.bb trackhub/hg38/
```

26| Create `trackhub/hg38/trackDb.txt` which describes how to display the tracks. Set the following parameters for an optimal view of segmentation tracks.

```
track DOHH2_Segway

type bigBed 12

bigDataUrl segway.layered.bb

shortLabel DOHH2 segmentation

longLabel segmentation of DOHH2 cell line from ChIP-seq data

itemRgb on

visibility pack
```

Set the following parameters for the signal tracks.

```
track ENCFF509XSM_DOHH2_H3K4me1

type bigWig

bigDataUrl ENCFF509XSM_DOHH2_H3K4me1.bigWig

shortLabel DOHH2 H3K4me1

longLabel ChIP-seq signal in DOHH2

visibility full

maxHeightPixels 100:60:8

viewLimits 0:100


track ENCFF745GML_DOHH2_H3K4me3

type bigWig

bigDataUrl ENCFF745GML_DOHH2_H3K4me3.bigWig

shortLabel DOHH2 H3K4me3

longLabel ChIP-seq signal in DOHH2

visibility full
```

```
maxHeightPixels 100:60:8
viewLimits 0:100


track ENCFF592CSV_DOHH2_H3K27me3
type bigWig
bigDataUrl ENCFF592CSV_DOHH2_H3K27me3.bigWig
shortLabel DOHH2 H3K27me3
longLabel ChIP-seq signal in DOHH2
visibility full
maxHeightPixels 100:60:8
viewLimits 0:100


track ENCFF890NAY_DOHH2_H3K27ac
type bigWig
bigDataUrl ENCFF890NAY_DOHH2_H3K27ac.bigWig
shortLabel DOHH2 H3K27ac
longLabel ChIP-seq signal in DOHH2
visibility full
maxHeightPixels 100:60:8
viewLimits 0:100


track ENCFF884IIL_DOHH2_CTCF
type bigWig
bigDataUrl ENCFF884IIL_DOHH2_CTCF.bigWig
shortLabel DOHH2 CTCF
```

```
longLabel ChIP-seq signal in DOHH2

visibility full

maxHeightPixels 100:60:8

viewLimits 0:100
```

The UCSC Genome Browser provides many other options as described on its website (genome.ucsc.edu/goldenpath/help/trackDb/trackDbHub.html#commonSettings).

27| Upload the track hub directory to a public web space. For example, to copy changes to a remote server named `yourserver` with username `yourname`, execute:

```
rsync -a trackhub yourname@yourserver:/your/publicly/available/space
```

The `-a` option specifies `rsync`'s archive mode, which preserves all file attributes, recursively copying files and directories.

28| Visit the UCSC Genome Browser (https://genome.ucsc.edu) and load your track hub. To do so, select "My data" > "Track hubs" from the top menu and add the direct link to your `hub.txt` file in the "URL" field. Push the "Add Hub" button. This will allow you to visualize your segmentation as well as your signal tracks, if you included them in your track hub.

? TROUBLESHOOTING

? TROUBLESHOOTING

Table 1 contains troubleshooting recommendations.

• Timing

The entire protocol takes < 8 h, with approximately 3 h of configuration and entering commands and approximately 5 h of computation. We took the timings for this protocol from a Grid Engine cluster system where we submitted each job to a compute node running at 2.6 GHz with 4 MB of cache and 32 effective CPU cores.

Step 1, installing the prerequisite software: < 30 min

Steps 2–3, downloading data from ENCODE: < 1.5 h. This step largely depends on the speed of the internet connection used to download the datasets. We downloaded the datasets at 3 MB/s–4 MB/s.

Steps 4–6, filtering uniquely mappable regions: 10 min

Step 7–8, creating the Genomedata archives: 20 min

Steps 9–14, prepare and train Segway with a 10-label model: 30 min

Step 15, annotate the genome with the trained Segway model: 1.5 h

Steps 16–20, analyzing the annotation with Segtools: 30 min

Steps 21–28, create and upload a trackhub of the annotation: 2 h. The timing on these steps depends on the speed of the internet connection used to upload the datasets.

For smaller datasets not in this protocol, such as K562 GRCh37/hg19 with the same ChIP-seq targets from ENCODE, the computation time is substantially reduced. The speed of the protocol

largely depends on the availability of processors Segway can submit jobs to and the speed of processors themselves. Any bottlenecks on a cluster system or running Segway on a limited number of processors will substantially increase the protocol length.

ANTICIPATED RESULTS

Segway produces an annotation for a given cell type. We illustrate the results of Segway's annotation on DOHH2 from **Step 15** by exploring the output of Segtools produced in the **Steps 16–20** and visualizing the segmentation on the UCSC Genome Browser using the track hub produced in **Steps 21–28**.

Exploring the parameters learned during the training task

The command described in **Step 16** creates the `gmtk-parameters/gmtk_parameters.stats.png` file showing the Gaussian parameters learned by Segway during the training task described in **Step 14**. The file contains a heatmap (**Figure 6**) with the data tracks in rows and the labels in columns. For each track-label combination, Segway learns a probability distribution over track values given a label. By default, it uses a Gaussian, or normal distribution, for this probability distribution. The colors on the heatmap represent row-normalized Gaussian means, where dark blue indicates a low mean and dark red indicates a high mean. The sizes of the black rectangles represent the variance parameter of the Gaussian, where larger rectangles indicate a higher variance. For example, label 1

associates with high values for H3K4me3, H3K27ac, and H3K4me1 tracks, shown in red. Low values occur for CTCF and H3K27me3, shown in blue. This observation allows us to hypothesize that label 1 is associated with active genes' transcription start sites.

Exploring the segment length distribution

The command described in **Step 17** creates the `length_distribution` directory. This directory contains summary statistics in tab-delimited format (`length_distribution.tab` and `segment_sizes.tab`). Segtools uses these statistics to generate summary plots. The first plot (`length_distribution.png`) (**Figure 7**) shows the distribution of segment lengths for each label.

The second plot (`segment_sizes.png`) (**Figure 8**) shows the fraction of total segments for each label and the fraction of genomic bases covered by each label. Some segments cover very large regions. In particular, Label 1 has some segments with extremely large length. Since the segmentation included some large assembly gaps, Segway picked the default highest average mean label. This is a result of the protocol using a genome sizes file for describing the reference genome used in the analysis and not a more precise description of the genome such as an Assembly Golden Path[42] (AGP) file where such regions would not be considered for training or annotation.

Exploring the segment enrichment against gene annotation

The command described in **Step 18** generates the file `aggregate_gene/feature_aggregation.splicing.png`, which summarizes the occurrence of each segmentation label (y-axis) relative to an idealized transcriptional gene model separated into 8 components (x-axis). For example, one can see the enrichment of label 1, in red, over components generally found around the 5′ end of a gene (**Figure 9**).

Visualize Segway results on a genome browser

Figure 10 illustrates the track hub generated in **Steps 21–28** loaded on the UCSC Genome Browser. The first track shows the GENCODE annotation of the *CDK1* locus. The subsequent tracks display the signal values from the five bigWig files generated in **Step 3**. Finally, the last track is the segmentation generated with Segway. In this example, labels 1 and 2 are present around the 5′ end of the *CDK1* gene. Labels 5 and 6 cover the middle and end of the gene. These observations are consistent with the enrichments shown in **Figure 9**.

Acknowledgments

We thank Carl Virtanen and Zhibin Lu (University Health Network High Performance Computing Centre and Bioinformatics Core) for technical assistance. We thank Paul Kitts and Valerie Schneider for providing masked regions from the Genome Reference Consortium analysis set. We thank J. Seth Strattan for assistance with locating the Genome Reference Consortium analysis set and the signal datasets used in this protocol. We thank Thomas Carroll for providing support for ChIPQC and for expediting work on ChIPQC's GRCh38 annotation.

This work was supported by the Canadian Cancer Society (703827 to M.M.H.), the Ontario Institute of Cancer Research (OICR) through funding provided by the Government of Ontario (CSC-FR-UHN to John E. Dick), the Ontario Ministry of Research, Innovation and Science (ER15-11-233 to M.M.H.), the Natural Sciences and Engineering Research Council of Canada (RGPIN-2015-03948 to M.M.H. and an Alexander Graham Bell Canada Graduate Scholarship to C.V.), the Ontario Ministry of Training, Colleges and Universities (Ontario Graduate Scholarship to C.V.), the University of Toronto Medicine by Design (C1TPA-2016-04 to M.M.H. and C1TPA-2016-01 to M.M.H.), the McLaughlin Centre (MC-2015-16 to M.M.H.), the National Institutes of Health (GM119550 to J.R.H.), the American Cancer Society (RSG-13-216-01-DMC to J.R.H.), and the Princess Margaret Cancer Foundation.

Author Contributions

E.G.R. wrote the introduction steps 1–4, 6–13 and Boxes 1 and 5.

M.M. wrote steps 5 and 14–24 of the procedure and the anticipated results.

E.G.R. and R.C. tested the protocol and improved the software.

M.K. and D.C. provided additional troubleshooting and testing.

A.K., R.A, R.C, and J.R.H wrote Box 2.

M.K. wrote Box 3.

C.V. wrote Box 4

R.A. and J.H. wrote Box 6.

E.G.R. and M.M.H. designed and coordinated the production of the overall protocol.

Competing Financial Interests

The authors declare that they have no competing financial interests.

Boxes

Box 1 | Installing Prerequisite Software

Segway is only supported on Linux.

There are two distinct methods to install Segway on your machine.

1. Install Segway as an administrator, for all users on the system. This is the easier method, if you have the administrator privileges.
2. Install Segway without any special privileges, for one user.

For either method you need to install:

- Python (2.7)
- Hierarchical Data Format 5 (HDF5) (1.8.17)
- Graphical Models Toolkit (GMTK) (1.4.4)
- Segway (2.0.2)
- R (3.3)
- Segtools (1.14)
- bedtools (2.26.0)
- bigWigToBedGraph
- bedGraphToBigWig (4)
- bedToBigBed (2.7)

- GNU Parallel (20180522)

Installing Segway as administrator

1. Install Python (with pip), HDF5, R, and GNU Parallel with your system package manager

Ubuntu or Debian 8:

```
sudo apt-get install python2.7 libhdf5-serial-dev hdf5-tools r-base
bedtools python-pip parallel
```

CentOS 7, Red Hat Enterprise Linux 7, Fedora:

```
sudo yum -y install hdf5 hdf5-devel R BEDTools readline-devel python-devel
parallel
```

For Red Hat Systems we recommend using the existing installed version of Python 2.7. Upgrading the system Python can break `yum`. **For Fedora 22+ only:** replace `'yum'` with `'dnf'`.

Install R dependencies:

From an interactive R environment (with sudo access):

```
install.packages(c("latticeExtra", "reshape2"), repos='http://cloud.r-
project.org/')
```

All Linux Distributions:

GMTK

```
wget http://melodi.ee.washington.edu/downloads/gmtk/gmtk-1.4.4.tar.gz

tar xf gmtk-1.4.4.tar.gz

cd gmtk-1.4.4

./configure

make

make install
```

Segtools

```
sudo pip install segtools
```

Segway

On Debian only:

```
    export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:"/usr/lib/x86_64-linux-
gnu/hdf5/serial"

    export C_INCLUDE_PATH=$C_INCLUDE_PATH:"/usr/include/hdf5/serial"

    sudo pip install segway
```

2. Install UCSC Genome Browser BigWig and BigBed tools[43]

The remaining software prerequisites are utilities distributed as standalone binaries. These utilities to convert genomic signal and annotation data between different formats. Ensure your PATH environment variable contains the location of your downloaded binaries. Download remaining utilities with the following commands:

bigWigToBedGraph

```
wget http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/bigWigToBedGraph
chmod +x bigWigToBedGraph
```

bedGraphToBigWig

```
wget http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/bedGraphToBigWig
chmod +x bedGraphToBigWig
mv bedGraphToBigWig "${HOME}/.local/bin/"
```

bedToBigBed

```
wget http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/bedToBigBed
chmod +x bedToBigBed
```

Installing Segway without administrator privileges

We highly recommend, when possible, getting an administrator to install the necessary software. Alternatively, we recommend using Bioconda[44] to install all the necessary software for this protocol. Bioconda is a suite of software pacakges that specialize in Bioinformatics. To use the packages available on Bioconda, the conda package and environment manager must be installed. If you have Anaconda (https://anaconda.org/) or Miniconda (https://conda.io/miniconda.html) already installed, conda is already installed on your system. If you do not have either, we recommend downloading and installing Miniconda. Miniconda runs on top of either Python 2 or Python 3. To determine which version of python version your system is running, run the command:

```
python --version
```

If you do not have Python installed, if possible ask your system adminstrator to install it for you. After determining which version of Python you have installed, install either a Python 2 or a Python 3 based Miniconda.

Installing Python 2 based Miniconda:

```
wget  https://repo.continuum.io/miniconda/Miniconda2-latest-Linux-x86_64.sh

chmod +x Miniconda2-latest-Linux-x86_64.sh

./Miniconda2-latest-Linux-x86_64.sh
```

Installing Python 3 based Miniconda:

```
wget  https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh

chmod +x Miniconda3-latest-Linux-x86_64.sh
```

```
./Miniconda3-latest-Linux-x86_64.sh
```

To add Bioconda packages for installation with conda, add the Bioconda channel:

```
conda config --add channels defaults
conda config --add channels conda-forge
conda config --add channels bioconda
```

The conda packages "segway" and "segtools" are the only two packages required for this protocol. We recommend setting up a new conda environment and installing the packages to this environment. To setup an environment named "segway" with the exact packages necessary for this protocol run the following command:

```
conda create –-name segway segway=2.0.2 segtools=1.1.14 parallel=20180522
```

To activate the environment and gain access to the software necessary to the protocol, enter the command:

```
conda activate segway
```

To return to back to your old environment enter the command:

```
conda deactivate
```

If any installation step fails, refer to **? TROUBLESHOOTING**.

Box 2 | Generate ChIP-seq Signal Files Using MACS2

Segway takes its input from signal files—normalized representations of ChIP-seq reads within a genomic region. You can convert aligned ChIP-seq data in BAM format to signal files in bedGraph or bigWig format. For visualization, bigWig is necessary. The bedGraph format may also be used in this protocol.

You should filter your aligned reads and compute predominant fragment lengths for the ChIP-seq data prior to generating the fold-enrichment signal files using the MACS2 software. We compare fragment-lengths to read-lengths in order to provide estimates about the amount of background signal in the ChIP-seq data.

To install the software necessary for generating ChIP-seq signal demonstrated here, in a Bioconda environment execute:

```
conda install phantompeakqualtools sambamba bedtools macs2 ucsc-bedclip
ucsc-bedgraphtobigwig
```

Estimate predominant fragment length

The program phantompeakqualtools[45–47] (v1.1) will calculate the predominant insert-size (or fragment length) based on strand cross-correlation analysis

Filtering aligned reads to generate a tagAlign file

The Sambamba[48] program is used to process BAM files. If your files are in tagAlign format then you do not need to use Sambamba.

Prior to generating the fragment length estimation, execute the following to filter a ChIP-seq

BAM file and generate a tagAlign file for the phantompeakqualtools and MACS2 program. For example, to generate a tagAlign file named `chip_TA.tagAlign.gz` from `mycellchipseq.bam`, execute:

```
sambamba view --nthreads "$NUM_THREADS" --filter 'not(unmapped or
mate_is_unmapped or failed_quality_control or secondary_alignment or
duplicate) and mapping_quality >=30' –format=bam mycellchipseq.bam |
bedtools bamtobed -i stdin | awk 'BEGIN{OFS="\t"}{$4="N";$5="1000";print
$0}' | gzip -c > chip_TA.tagAlign.gz
```

You are now ready to estimate the predominant fragment lengths using cross-correlation analysis. The input file can be in tagAlign or BED format.

Running phantompeakqualtools

From the command line, execute the following:

1. Run the phantompeakqualtools program from the command line in order to use cross-correlation analysis to estimate the predominant fragment lengths. To use multiple threads with phantompeakqualtools, use the `-p` option. For example, to run phantompeakqualtools with `$NUM_THREADS` threads, execute:

```
run_spp.R -c=chip_TA.tagAlign.gz -p="$NUM_THREADS" -filtchr=chrM -
savp=chip_TA.cc.plot.pdf -out=chip_TA.cc.qc
```

2. Execute the following command to write only the first value for estimated fragment length into the output file. This value (in almost all cases) is the best estimate of predominant fragment length.

```
sed -i -r 's/,[^\t]+//g' chip_TA.cc.qc
```

The output file `chip_TA.cc.qc` contains NSC/RSC results in a tab-delimited file of 11

columns. The columns are filename, number of reads, estimated fragment length, strand cross-correlation at estimated fragment length, read length, strand cross correlation at read length, strand shift with minimum cross-correlation, minimum cross-correlation, normalized strand cross-correlation coefficient NSC, relative strand cross-correlation coefficient RSC, and Quality Tag.

Notably, the estimated fragment length (in column 3) can contain multiple comma-separated values. We recommend using the first value, as this value is the best estimate of predominant fragment length in almost all cases.

NSC values less than 1.05 and RSC values substantially less than 1 have high background signal or low signal to noise ratios, which indicates poor quality data or low abundance of DNA-protein binding events[46]. In our example, the '`chip_TA.cc.plot.pdf`' output file contains the cross-correlation plot.

Generate fold enrichment coverage tracks using MACS2

The normalized signal track generation requires the use of MACS2 (https://github.com/taoliu/MACS/).

MACS will use as input a tagAlign file and a control ChIP-seq sample tagAlign file**.** Use the same procedure described above to convert your control file into tagAlign format if starting with a BAM file. The MACS2 program will ultimately produce a fold-enrichment file in the bigWig format. To use MACS2 to produce this output, execute the following from the command line:

1. Create the output directory for the MACS2 results:

```
mkdir -p peak_output
```

2. Using MACS2, generate the narrow peaks and preliminary signal tracks using the tagAlign file generated directly from the BAM file. The `--gsize` parameter passes the effective genome size to MACS2. Using `bedtools genomecov` and a mappability track for a read-length of 50, we calculate the effective genome size for hg38 to be `2.8e9`. MACS2 uses the effective genome size to calculate the background Poisson λ. You should always use your input ChIP-seq data's fragment length for MACS2. For example, to generate the signal tracks for tagAlign file `chip_TA.tagAlign.gz` and control ChIP-seq sample tagAlign file `control_TA.tagAlign.gz` with a fragment length (specified with `--extsize`) of 250 and a p-value cutoff (specified with `--pvalue`) of 0.01, execute:

```
macs2 callpeak --treatment chip_TA.tagAlign.gz --control
control_TA.tagAlign.gz --format BED --name peak_output/chip_TA --gsize
2.8e9 --pvalue 1e-2 --nomodel --extsize 250 --keep-dup all --bdg --SPMR --
shift 0
```

The --keep-dup all option specifies that MACS2 should keep all duplicate tags at the exact same location. The --bdg option produces `peak_output/chip_TA_treat_pileup.bdg`, which we use for noise removal. The `--shift 0` option specifies that there should be no arbitrary shift.

3. Using MACS2, generate the final fold-enrichment signal tracks. To generate the signal tracks for our example, execute:

```
macs2 bdgcmp --tfile peak_output/chip_TA_treat_pileup.bdg --cfile
peak_output/chip_TA_control_lambda.bdg --outdir peak_output --ofile
```

```
chip_TA_FE.bdg --method FE
```

4. Using `bedClip`, remove coordinates outside those specified in your chromosome sizes file, and generate a sorted bedGraph file. You will need a chromosome size file. This is a tab-delimited file with two columns; chromosome name (column 1), and chromosome size in base pairs (column 2). To download the chromosome sizes file for your genome see the bedClip command line help text. For hg38, a sizes file can be downloaded from http://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/hg38.chrom.sizes. For example, to remove any coordinates in `chip_TA_FE.bdg` that are outside of those specified in `hg38.chrom.sizes`, and generate and sort the resulting bedGraph file, execute:

```
bedClip -truncate peak_output/chip_TA_FE.bdg hg38.chrom.sizes
peak_output/chip_TA.fc.signal.bedGraph
sort -k 1,1 -k2,2n chip_TA.fc.signal.bedGraph &>
chip_TA.fc.signal.sorted.bedGraph
```

5. (Optional) Using bedGraphToBigWig, convert the resulting bedGraph file to bigWig format. Segway can directly use bedGraph files as signal tracks. However, bigWig format enables more efficient visualization on the UCSC Genome Browser of large, dense, and continuous data. For example, to convert `chip_TA.fc.signal.sorted.bedGraph` to bigWig format:

```
bedGraphToBigWig peak_output/chip_TA.fc.signal.sorted.bedGraph
GRCh38_EBV.chrom.sizes.tsv peak_output/chip_TA.fc.signal.sorted.bigWig
```

6. (Optional) Remove intermediate files

```
rm -f peak_output/chip_TA_peaks.xls
rm -f peak_output/chip_TA_peaks.narrowPeak
```

```
rm -f peak_output/chip_TA_summits.bed
rm -f peak_output/chip_TA_FE.bdg
rm -f peak_output/chip_TA.fc.signal.bedGraph
rm -f peak_output/chip_TA_treat_pileup.bdg
rm -f peak_output/chip_TA_control_lambda.bdg
```

For problems encountered in this Box, refer to **? TROUBLESHOOTING**

Box 3 | Downloading ChIP-seq data from ENCODE

The ENCODE Project (https://www.encodeproject.org/) provides raw and processed ChIP-seq data for transcription factors and histone modifications on its website. Locate data of interest through this website's search or the ENCODE data matrix (https://www.encodeproject.org/matrix). The data matrix makes it easy to explore all available experiments for your cell type of interest.

For example, we want to acquire ChIP-seq data for H3K4me1 in the DOHH2 cell type. This cell type is derived from the pleural effusion of a B cell lymphoma patient. We will therefore use the search box to find it directly.

1. Open https://www.encodeproject.org/ in your browser.
2. In the search box at the upper right corner of the page, type in "DOHH2 H3K4me1 ChIP-seq" and press Enter (Figure B3.1).
2.
3. Click on any of the results that match your preferences to visit the experiment summary page. If you plan to use data from several ChIP-seq experiments, consider that different labs may have generated the data. These labs might also use different laboratory protocols. You can read these details in the experiment summary page (Figure B3.2).

4. In the "File" section, under the "File details" section there are two panels for accessing raw and processed data (Figure B3.3). Each panel provides detailed information on

each file. The “File type” and “Mapping assembly” columns guide you to the format you need for your analysis. If the “Biological replicate” column is empty or “1,2”, this indicates replicates are merged. Find the file type and genome assembly you need.

5. Click the download icon in the “Accession” column in the right of accession numbers to download a file immediately. Alternatively, right click on the icon and copy the link location to use with `"wget"` command later. In the example below, we download the signal file of H3K4me1 ChIP-seq experiment in DOHH2:

```
URL="https://www.encodeproject.org"

ACCESSION="ENCFF509XSM"

FORMAT="bigWig"

CELL="DOHH2"

MARK="H3K4me1"

wget "$URL/files/$ACCESSION/@@download/$ACCESSION.$FORMAT" \

    -O "${ACCESSION}_${CELL}_${MARK}.$FORMAT"
```

ENCODE project website has a detailed application program interface (API) documentation (https://www.encodeproject.org/help/rest-api/). The ENCODE project API allows you to automate your search queries and downloads. The preferred input for Segway is the “fold change over control” bigWig signal file, because it is already processed and normalized. It is possible to generate the signal file from raw files of any ChIP-seq experiment using `MACS2` (see **Box 2**).

For the examples in this manuscript, we download the signal files of DOHH2 ChIP-seq datasets for H3K4me1 (ENCFF509XSM), H3K4me3 (ENCFF745GML), H3K27ac (ENCFF890NAY), H3K27me3 (ENCFF592CSV) and CTCF (ENCFF884IIL). These bigWig files average approximately 1 GiB each in size. To download all of these files, execute the following script in a bash terminal:

```
URL="https://www.encodeproject.org"

FORMAT=bigWig

CELL=DOHH2

MARKS=(H3K4me1 H3K4me3 H3K27ac H3K27me3 CTCF)

ACCESSIONS=(ENCFF509XSM ENCFF745GML ENCFF890NAY ENCFF592CSV ENCFF884IIL)

for i in ${!MARKS[@]}

do

    ACCESSION=${ACCESSIONS[$i]}

    MARK=${MARKS[$i]}

    wget "$URL/files/$ACCESSION/@@download/$ACCESSION.$FORMAT" \

        -O "${ACCESSION}_${CELL}_${MARK}.${FORMAT}"

done
```

Box 4 | Quality control for ChIP-seq data

For Segway to produce high-quality segmentations, input chromatin immunoprecipitation sequencing (ChIP-seq) data must have sufficient quality. Quality control methods for ChIP-seq data can ensure this.

Several technical factors affect the quality of ChIP-seq data.[49] Often, most important is the specificity of the antibody employed. Also key is using a sufficient number of cells and appropriate controls. Fragmentation, library construction, and sequencing protocols used also influence data quality.[49] Finally, the alignment software used influences signal mapping quality.

This box explains how to compute different quality control metrics using FastQC,[50] ChIPQC,[51,52] and NGS-QC.[53] We provide general guidelines, which provide a clear means of assessing data quality. There are, however, a number of different ways of assessing quality and many methods depend greatly upon the particulars of the assessed experiment. There is not yet a consensus on a general and optimal means of assessing ChIP-seq data quality and we suggest that interested readers consult some of the broad literature on this topic.[45,46,51,53–57]

Installation of QC tools and dependencies

Installing with Bioconda

In an environment of your choice, install the fastqc, sambamba, and ChIPQC Bioconductor packages:

```
conda install fastqc sambamba bioconductor-chipqc
```

Installing without Bioconda

FastQC[50]: follow instructions at http://www.bioinformatics.babraham.ac.uk/projects/fastqc/INSTALL.txt.

Sambamba[48]: follow instructions at http://lomereiter.github.io/sambamba or our instructions in Box 2.

ChIPQC:[51,52]

1. Install: R[58] 3.4.3
2. Install Bioconductor[59] 3.6 and ChIPQC version 1.14.0, by executing the following within the R environment:

```
source("http://bioconductor.org/biocLite.R")biocLite()  # install Bioconductor
biocLite("DiffBind")  # install latest version of dependency
biocLite("ChIPQC")  # install ChIPQC
biocLite("TxDb.Hsapiens.UCSC.hg38.knownGene")  # for GRCh38/hg38
```

Sequence data quality control with FastQC

FastQC[50] reports on potentially problematic aspects of the sequencing itself. This includes base qualities, G+C bias, systematic overrepresentation of sequences, and several other metrics. To run FastQC on BAM files, such as the ENCODE CTCF ChIP-seq samples, whose signal files were downloaded in Box 3, execute:

```
fastqc ENCFF863PSQ.bam ENCFF092CZO.bam
```

One useful metric is library complexity—a measure of the number of distinct molecules in the library. Low complexity often results in repeated sequencing of duplicates, yielding little information.[60] You can estimate library complexity from FastQC's duplicate sequence plot. More detailed analyses can be performed, if necessary, using preseq.[61] A large fraction of duplicate sequences is often a result of insufficient sequence diversity, which can suggest an inherent experimental bottleneck, such as an insufficient number of input cells.[55,60]

Additional considerations include checking the quality of read mapping, the proportion of reads aligning to a genomic position, the number passing a mapping quality threshold, and the abundance of duplicate reads.

Assessing ChIP-seq data quality

To assess ChIP-seq data quality; you should perform overall assessment of both ChIP-seq reads and the effects of random sub-sampling, when possible. You should use frequency of reads in peaks (FRiP) with ChIP-seq peak calls, but we do not focus on this approach since Segway operates directly upon signal. It is useful to assess the consistency across ChIP-seq replicates, such as via an irreproducible discovery rate[62]. ENCODE conducts this analysis for all of its ChIP-seq datasets.[46] This is currently non-trivial, however, to use in one's own workflow.

Testing ChIP-seq quality with ChIPQC

ChIPQC[51,52] is an R[58] Bioconductor[59] package that evaluates metrics of mapping, filtering, and duplication rates, as well as ChIP-seq signal distribution and structure. While ChIPQC usually also computes FRiP, from called peaks, here we will run it using only signal data.

1. Define or use the previously exported `NUM_THREADS` environment variable.
2. Determine the maximum memory for ChIPQC to use in GiB:

```
MAX_M_USE=$(($(free -g | head -3 | tail -1 | tr -s ' ' | cut -d ' ' -f 4) -
1))
```

On a cluster, use instead 1 GiB less than the amount allocated to your job.

3. Create a sorted, indexed, and duplicate marked BAM file, using Sambamba[48]. For example to do so for the `ENCFF863PSQ.bam`, run the following:
    a.
```
sambamba sort --memory-limit "${MAX_M_USE}GiB" --nthreads
"$NUM_THREADS" --compression-level 0 --out "${TMPDIR:-
/tmp}/ENCFF863PSQ.sorted.bam" ENCFF863PSQ.bam
```
    b.
```
sambamba markdup --nthreads "$NUM_THREADS" --compression-level
9 --tmpdir="$TMPDIR" "${TMPDIR:-/tmp}/ENCFF863PSQ.sorted.bam"
/dev/stdout | tee ENCFF863PSQ.sorted.markeddup.bam | sambamba
index --nthreads "$NUM_THREADS" /dev/stdin
ENCFF863PSQ.sorted.markeddup.bam.bai
```
    c.
```
rm -f "${TMPDIR:-/tmp}"/ENCFF863PSQ.sorted.bam*
```
4. Create an experiment description file for ChIPQC. This file describes the ChIP-seq samples that ChIPQC will analyze. ChIPQC operates on each input file, which might be a single technical replicate, with pooled sequencing lanes, or a single biological replicate, merged from multiple technical replicates. The content of a single unit of quality assessment—a single file—depends upon your experimental setup and downstream experimental goals. Each row corresponds to one ChIP-seq file, while each column describes the data associated with that file. We highlight a common use-case; refer to the package documentation for a detailed description of all available

fields. You should specify the following columns: `SampleID`, `Tissue`, `Factor`, `Replicate`, `bamReads`, `ControlID`, `bamControl`, and `Peaks`. These fields act merely as annotations of your data and do not alter ChIPQC's operation, with two exceptions. Two fields must contain valid file paths: `bamReads`, which must contain the full path to the above sorted and duplicate marked BAM file of the ChIP-seq experiment. The other field, `bamControl`, must contain the full path to a corresponding set of control reads, in a sorted and duplicate marked BAM, such as from an input (antibody-free) experiment. In this case, without peak calls, set `Peaks` to `NA`. This will cause ChIPQC to compute all metrics that do not depend upon a peak set. If you have a peak file, set `Peaks` to the file name and additionally specify a `PeakFormat` column, if the file is not in BED format. Even if you are only analyzing a single replicate, you must still specify the `Replicate` column. You may set it to `1` in this case.

Name this file `QCexperiment.csv` and delimit its columns with tabs. For example, a single replicate may result in a file like this:

```
SampleID        Tissue  Factor  Replicate  bamReads        ControlID
bamControl      Peaks

ENCFF863PSQ       DOHH2    CTCF  1  ENCFF863PSQ.sorted.markeddup.bam
ENCFF631ENA ENCFF631ENA.sorted.markeddup.bam     NA
```

3. Execute the following within the R environment:

```
library("TxDb.Hsapiens.UCSC.hg38.knownGene")
library("ChIPQC")

samples <- read.delim("QCexperiment.csv", stringsAsFactors=FALSE)
```

```
experiment <- ChIPQC(samples, annotation="hg38")
```

Specify the assembly employed via the `annotation` parameter. We used GRCh38/hg38 above, but you can also, for example, use GRCh37/hg19 instead via `annotation="hg19"`.

4. Generate the output report, summary, and plots of interest by executing in the R environment:

```
# disable faceting when using only a single sample

facet <- ifelse(nrow(samples) > 1, TRUE, FALSE)



write.table(QCmetrics(experiment), file='QCmetrics.csv')



ChIPQCreport(experiment, facet=facet)
```

The ChIPQC documentation contains additional details, including other available plots.[52]

? TROUBLESHOOTING

Interpreting ChIPQC results

Assessing read mapping quality

Verify that ChIP-seq results have a substantial portion of uniquely mapped reads, without an unexpectedly high proportion of reads filtered out due to insufficient quality. Generally, at least 50% of reads in an experiment should map uniquely, with lower values expected for input data, which lacks a targeting antibody. This varies greatly, however, and depends on the

species analyzed.[55,56] In human and mouse ChIP-seq samples, expect over 70% of reads to map uniquely.[55,56] You should also assess the duplication rate—a ratio of unique to total reads. This varies greatly, but you should expect for it to be much less in control samples and it should generally not exceed 50%.[52]

Assessing ChIP-seq signal distribution

You should also assess the read cross-correlation, as a metric of ChIP-seq data quality. The ChIPQC cross-correlation plot should have a clear peak at the fragment length in successfully-enriched samples.[52] The normalized strand cross-correlation coefficient within the QC metrics file should be greater than 1.05, while the relative strand cross-correlation coefficient should be greater than 0.8.[46] Bailey et al.[55] describes how to use and interpret this metric further (Box 2 of Bailey et al.[55]).

Additionally, evaluate ChIPQC's coverage histograms and their standardized standard deviation (SSD), normalized to read depth. Expect the coverage histogram to have a non-negligible "tail" and for SSD values to be greater than 1, generally above 1.5. Expect controls to have SSD values of around 1. Control SSDs greater than 1 might indicate aberrant enrichment.[52]

Subsampling to assess ChIP-seq quality with NGS-QC

Random sub-sampling on ChIP-seq profiles provides a means of computing quality metrics that do not depend upon peak calling. Importantly, such metrics are comparable between sharp (transcription factors) and broad (histone modifications) peaks,[53] both of which are frequently used together by Segway. The Next Generation Sequencing Quality Control Generator (NGS-QC)[53,63,64] provides these metrics for a wide array of ChIP-seq datasets. It

uses measurements of global deviations of random read subsets with respect to the full set of aligned reads to assign quality scores.[64] NGS-QC randomly subsamples 90%, 70%, and 50% of reads and counts reads in subsampled and full datasets in 500–base-pair bins. NGS-QC measures the variance from the expectation of the same percentage decrease in read counts. It quantifies the proportion of these bins that are below 2.5%, 5%, or 10% of this expected fraction, and each threshold forms a component of the quality score. These scores form labels for ChIP-seq data quality from A–D, for each threshold. Therefore, the highest rating is "AAA", while the lowest is "DDD".[64] NGS-QC has pre-computed quality metrics for many public datasets, including some ENCODE data. Consult these if available. Otherwise, compare your data to them, as outlined below.

Use NGS-QC via its Galaxy[65] instance, as follows:

1. Navigate to: http://galaxy.ngs-qc.org/
2. Create an account.
3. Upload BAM files, using FTP.
    a. From the directory containing the BAM files to upload on the command line, run
       `ftp galaxy.ngs-qc.org`
    b. Login using the previously created credentials.
    c. Execute:
        i. `prompt` (disables confirmation of each individual upload)
        ii. `mput` <files>
        iii. `exit`
    d. Navigate to the Galaxy instance, and select "Get Data" > "Upload file" from the left panel.

e. Select the files uploaded via FTP, using the “Choose FTP file” option.

4. Access the NGS-QC command under “NGS-QC” > “NGS-QC Generator“. Fill in the provided form to assess the uploaded data, following the NGS-QC guidelines (http://www.ngs-qc.org/tutorial.php#part2). We recommend selecting three replicates of the QC computation, to mitigate against sampling bias. Specify the genome to which the uploaded data was aligned, from the available list. At this time, GRCh38/hg38 is not available. For the other parameters, use the defaults.
5. Compare against existing public datasets, selecting the target of the ChIP-seq experiment at the bottom of the “NGS-QC Generator” interface, when it lists your target.
6. Once Galaxy indicates that the tasks have completed (turning green), select the “View Data” button for the task starting with “Results”. Then inspect each replicate, especially its quality rating and percent uniquely mapped reads. Evaluate them using the criteria above.

Mendoza-Parra et al.[63] and the NGS-QC tutorial (http://www.ngs-qc.org/tutorial.php#part4) provide details on how to interpret the results.

Box 5 | Accelerating Segway using a compute cluster

We designed Segway to run on a large variety of cluster systems. Segway uses the Distributed Resource Management and Application API[66] (DRMAA) version 1 for submitting its jobs to a cluster. This interface has been implemented for a variety of cluster systems including Grid Engine[67] (GE), Condor[68], Portable Batch System[69] (PBS / Torque), and Platform Load Sharing Facility[70] (LSF).

Installing DRMAA

We recommend that a cluster administrator installs DRMAA. With the exception of Grid Engine, which has DRMAA installed by default, each cluster system has its own specific installation procedure. The general steps for installing DRMAA are as follows:

1. Download and build the DRMAA implementation for your cluster system:

Grid Engine

Not applicable: installed by default

Torque or PBS Pro

https://sourceforge.net/projects/pbspro-drmaa/

Platform Load Sharing Facility

https://sourceforge.net/projects/lsf-drmaa/

Slurm workload manager

https://github.com/natefoo/slurm-drmaa/releases

2. Set the environment variable `DRMAA_LIBRARY_PATH` to the full name and path of the DRMAA library. For example, if you have GE installed in `/sge/lib/linux-x64` execute:

```
export DRMAA_LIBRARY_PATH="/sge/lib/linux-x64/libdrmaa.so"
```

Computer cluster memory settings

For Grid Engine only, in order to improve Segway's handling of memory usage, a "mem_requested" resource must be setup by the system adminstrator for Segway to use. After Segway has been installed, this can be achieved by running:

python –m segway.cluster.sge_setup

Compute cluster settings in Segway

Some cluster configurations require submitted jobs to have specific settings. Segway can specify cluster specific settings with the `--cluster-opt` option. This option passes on what would normally be set as options from your native job submission command (such as `qsub` from Grid Engine or Torque). For example, for Segway to submit jobs to a Grid Engine queue named `bioinformatics` use the option:

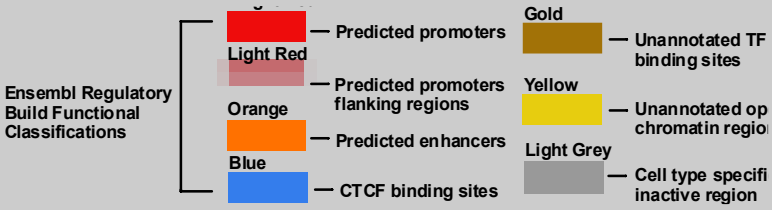
```
--cluster-opt="-q bioinformatics"
```

Verifying job submission by Segway

When Segway submits jobs, it logs each submission to the console. If the log entry starts with “queued” then DRMAA is running successfully. If the log entry starts with “running locally”, then Segway is not submitting the jobs to the cluster and they are instead running “locally” on the same machine that contains the Segway process.

Box 6 | Methods to compare and analyze segmentations

Comparing a new segmentation with previously annotated segmentations can be used to help assign biological meaning to the numbered labels generated by Segway (e.g., patterns associated with TSS, promoters, or enhancers). One strategy for making these comparisons is to visualize a new segmentation alongside a published segmentation in the UCSC Genome Browser[29] and compare the un-annotated labeled regions to previously annotated labels. For example, we can compare the DOHH2, a B-cell lymphoma cancer cell line, segmentation to a published segmentation of a lymphoblastic B-cell line, GM12878, from the Ensembl Regulatory Build.[28] The Ensembl Regulatory Build partitioned the genome into regulatory regions and 18 cell lines are available for through the UCSC Genome Browser in the hg38 assembly. The regulatory segmentation employs color-coding to identify the regulatory elements represented by each label (as illustrated below).

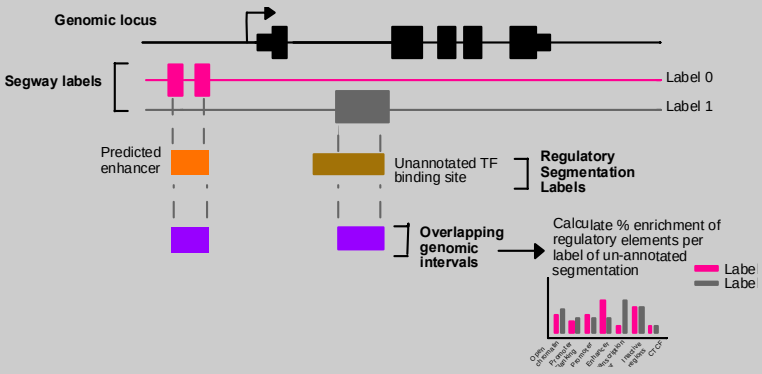


In a UCSC track hub, we can visually compare the DOHH2 segmentation and ChIP-seq data to the GM12878 regulatory segmentation in order to manually assign labels by their association with regulatory segments, ChIP-seq signal, and gene models. To add the Ensembl Regulatory Segmentation to a hub containing an un-annotated segmentation, in the UCSC browser select “My Data” > “Track Hubs.” Under “Public Hubs”, connect the “Ensembl Regulatory Build.” After loading the hub, open the genome browser to the hg38 assembly and the regulatory track hub will appear in the window. To visualize only the GM12878 segmentation, right click on the segmentation in the browser window and click “Configure Cell Type Activity.” Select only the GM12878 track and submit. Use the “zoom in” tool on the

browser for a closer view of the segments. Reorder the segmentation and signal tracks in the browser to facilitate visual comparison by clicking on the left-hand side of the track and dragging up or down to a new position. Adjust the configuration settings for each track by right clicking and selecting “Configure” to manipulate density of the display, axis display height, and color display. These settings will also display subsets of the regulatory segmentation. For example, a summary track from the regulatory segmentation can be included that identifies transcription factor binding sites from all cell types.

In figure XX.X, the CDKN1A gene locus was selected for visual comparison of the DOHH2 segmentation and the GM12878 published regulatory segmentation. This comparison included the ChIP-seq signal tracks, which may provide additional information about genomic features captured by segments and labels. The CDKN1A gene codes for a cyclin-dependent kinase inhibitor that is upregulated in multiple myeloma and is a potential therapeutic target.[71,72] Specific genes or genomic regions can be targeted for comparison based on cell type and biological process of interest to provide new insights into gene regulation.

Another strategy is to determine the similarity of intervals associated with a numbered label in a new segmentation to regions with an annotated label from a published segmentation. This type of analysis can be performed using bioinformatics tools to determine which regulatory genomic regions are enriched in the un-annotated segmentation labels, as illustrated below. This will help to assign biologically meaningful labels to new segmentations.



This analysis can be performed entirely from the command line

or it may be performed using the R programming language in Rstudio. Scripts to perform the analyses are available (wwww.xxxx.com). In order to perform either analysis, the Ensembl Regulatory Segmentation must be downloaded and converted to bed file format. Navigate to segway.hoffmanlab.org and click on the link "Ensembl Regulatory Build for GRCh38 (hg38)." Select "segmentations/" and in the files section of the new page copy the link location for the GM12878. Use the "wget" command to download the bigBed file for this cell line. Install "bigBedToBed (http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/bigBedToBed) and convert the published segmentation file in bigBed file format to bed file format using the command `bigBedToBed GM12878.bb GM12878.regulatory.bed`.

The first bioinformatics approach uses the `bedtools intersect` command, which will generate a new bed file reporting the overlapping genomic regions between the intersected bed files.[69] Using the script `bed_enrichment.sh`, the intersection bed file can be filtered to calculate a percent enrichment for the number of times a regulatory element overlaps a segment in a specific label.

The second bioinformatics approach uses packages available through `RStudio` (download here: https://www.rstudio.com). In addition to calculating enrichment of regulatory elements within labels, this approach facilitates more robust analysis and visualization of the segmentations (e.g, statistics for relationships between intervals, easy-to-use plotting functions).

Functional profiling of Segway annotation with gProfiler

In this section we show how Segway annotations can be used to identify the biological processes that are specific to the cell type of interest. This section of the protocol involves installing libraries that are not required by neither Segway nor Segtools, therefore, we estimate that it is beyond the scope of this manuscript to have as detailed instruction as in the

main text. Instead, we focus on explaining the main concepts, provide the scripts and the instructions to create a virtual environment to reproduce the analysis.

The approach described here consist of defining a list of DOHH2 specific genes and use it as an input to the gProfiler tool.

To define a list of DOHH2 promoter region, first identify the segment label that is the most enriched at promoter regions using the figure generated previously. Specifically, search in the the **Figure 7** generated in **step 14** for the label with the highest theoretical signals from the H3K4me3 track which is known as a mark of active promoter [PMID:15123803]. For DOHH2 the label 8 seems the most relevant. You can confirm this hypothesis by looking at the most frequent location of this label in the **Figure 9** generated in **step 17** that shows that label 8 is enriched at gene starts.

To obtain a list of gene ids expressed in DOHH2, get the coordinates of the segments with the TSS label and intersect it with gencode gene annotations. Run:

```
segment_to_gencode_gene_list.py segway.bed.gz gencode.v25.annotation.gtf.gz –label 8 –output DOHH2.genes.csv
```

[MMM-source]

To identify the biological processed specific to the DOHH2 cell lines, the list of genes obtained above need to be filtered for relevant genes. It is at user's discretion to determine what relevant genes are. Here we illustrate how to achieve this by subtracting to the DOHH2 genes, the genes obtained from running the steps 1 to 17 of this protocol on the H1 Human embryonic stem cells (H1-hESc) and GM12878 cells from ENCODE, with the same histone marks (Table 2).

After obtaining the list H1-hESc and GM12878 expressed genes by following the instructions

in this section, subtract these genes to the DOHH2 genes and run gProfiler:

```
subtract_genes_and_run_gprofiler.py   DOHH2.genes.csv   H1hESC.genes.csv   –
output DOHH2-H1hESC.enrichment.tsv

subtract_genes_and_run_gprofiler.py   DOHH2.genes.csv   GM12878.genes.csv   –
output DOHH2-GM12878.enrichment.tsv
```

Interpretation of the enrichment analysis

DOHH2 is an EBV-negative non-Hodgekin's B-cell lymphoma cell line [PMID:1849602]. To identify epigenomic signatures of lymphoma that do not exist in the human embryonic stem cell (H1-hESC) or the EBV-positive lymphoblastoid cell line GM12878, we identified unsupervised epigenomic signatures of each cell type using H3K27ac, H3K4me3, H3K27me1, and CTCF ChIP-seq data and Segway.

Segway identified 2966 promoters in DOHH2 that did not exist in H1-hESC. Gene ontology terms such as immune system process, leukocyte activation, immune effector process, adaptive immune response, and other immune related pathways had a significant enrichment in DOHH2-specific promoters (Fisher's exact test FDR < 0.01). This is in line with the nature of the cell of origin for non-Hodgekin's lymphomas [PMID:1849602].

Segway also identified 3292 promoters identified only in the EBV-negative DOHH2 but not in the EBV-positive lymphoblastoid cell line GM12878. Gene ontology terms such as cellular metabolic process and cellular biosynthetic processes had significant enrichment in DOHH2-specific promoters. This agrees with previous evidence of metabolic shift and hypoxic stress in non-Hodgkin's lymphoma [PMID:25158954,PMID:PMC4591764].

References

1. Hoffman, M. M. *et al.* Unsupervised pattern discovery in human chromatin structurethrough genomic segmentation. *Nat. Methods* **9**, 473–476 (2012).

2. Hoffman, M. M. *et al.* Integrative annotation of chromatin elements from ENCODEdata. *Nucleic Acids Res.* **41**, 827–841 (2013).

3. Song, L. & Crawford, G. E. DNase-seq: a high-resolution technique for mapping activegene regulatory elements across the genome from mammaliancells. *Cold Spring Harb. Protoc.* **2010**, db.prot5384 (2010).

4. Dorschner, M. O. *et al.* High-throughput localization of functional elements byquantitative chromatin profiling. *Nat. Methods* **1**, 219–225 (2004).

5. Buenrostro, J. D., Giresi, P. G., Zaba, L. C., Chang, H. Y. & Greenleaf, W. J. Transposition of native chromatin for fast and sensitiveepigenomic profiling of open chromatin, DNA-binding proteinsand nucleosome position. *Nat. Methods* **10**, 1213–1218 (2013).

6. Barski, A. *et al.* High-resolution profiling of histone methylations in the humangenome. *Cell* **129**, 823–837 (2007).

7. Zhang, Y. *et al.* Model-based Analysis of ChIP-Seq (MACS). *Genome Biol.* **9**, R137 (2008).

8. Feng, J., Liu, T., Qin, B., Zhang, Y. & Liu, X. S. Identifying ChIP-seq enrichment using MACS. *Nat. Protoc.* (2012).

9. Rozowsky, J. *et al.* PeakSeq enables systematic scoring of ChIP-seq experiments

relative to controls. *Nat. Biotechnol.* **27**, 66–75 (2009).

10. ENCODE Project Consortium *et al.* Identification and analysis of functional elements in 1% of thehuman genome by the ENCODE pilot project. *Nature* **447**, 799–816 (2007).

11. Day, N., Hemmaplardh, A., Thurman, R. E., Stamatoyannopoulos, J. A. & Noble, W. S. Unsupervised segmentation of continuous genomic data. *Bioinformatics* **23**, 1424–1426 (2007).

12. Thurman, R. E., Day, N., Noble, W. S. & Stamatoyannopoulos, J. A. Identification of higher-order functional domains in the human ENCODE regions. *Genome Res.* **17**, 917–927 (2007).

13. Ernst, J. & Kellis, M. ChromHMM: automating chromatin-state discovery andcharacterization. *Nat. Methods* **9**, 215–216 (2012).

14. Lystig, T. C. & Hughes, J. P. Exact computation of the observed information matrix for hiddenmarkov models. *J. Comput. Graph. Stat.* **11**, 678–689 (2002).

15. Schliep, A., Schönhuth, A. & Steinhoff, C. Using hidden Markov models to analyze gene expression timecourse data. *Bioinformatics* **19 Suppl 1**, i255–63 (2003).

16. Jiang, K., Thorsen, O., Peters, A., Smith, B. & Sosa, C. P. An Efficient Parallel Implementation of the hidden Markov methodsfor genomic sequence-search on a massively parallel system. *IEEE Trans. Parallel Distrib. Syst.* **19**, 15–23 (2008).

17. Sohn, K.-A. *et al.* hiHMM: Bayesian non-parametric joint inference of chromatinstate maps. *Bioinformatics* **31**, 2066–2074 (2015).

18. Mammana, A. & Chung, H.-R. Chromatin segmentation based on a probabilistic model for readcounts explains a large portion of the epigenome. *Genome Biol.* **16**, 151 (2015).

19. Rhee, H. S. & Pugh, B. F. Comprehensive genome-wide protein-DNA interactions detected at single-nucleotide resolution. *Cell* **147**, 1408–1419 (2011).

20. He, Q., Johnston, J. & Zeitlinger, J. ChIP-nexus enables improved detection of in vivo transcription factor binding footprints. *Nat. Biotechnol.* **33**, 395–401 (2015).

21. Chen, X., Hoffman, M. M., Bilmes, J. A., Hesselberth, J. R. & Noble, W. S. A dynamic Bayesian network for identifying protein-bindingfootprints from single molecule-based sequencing data. *Bioinformatics* **26**, i334–i342 (2010).

22. Chan, R. C. W. *et al.* Segway 2.0: Gaussian mixture models and minibatch training. *Bioinformatics* 1–3 (2017) doi:10.1093/bioinformatics/btx603.

23. Ernst, J. & Kellis, M. Chromatin-state discovery and genome annotation with ChromHMM. *Nat. Protoc.* **12**, 2478–2492 (2017).

24. Song, J. & Chen, K. C. Spectacle: Fast chromatin state annotation using spectral learning. *Genome Biol.* **16**, 33 (2015).

25. Libbrecht, M. W. *et al.* A unified encyclopedia of human functional DNA elements through fully automated annotation of 164 human cell types. *Genome Biol.* **20**, 1–14 (2019).

26. UCSC Genome Browser: BedGraph Track Format.

27. Hoffman, M. M., Buske, O. J. & Noble, W. S. The Genomedata format for storing large-scale functional genomics data. *Bioinformatics* **26**, 1458–1459 (2010).

28. UCSC Genome Bioinformatics: FAQ.

29. Kluin-Nelemans, H. C. *et al.* A new non-Hodgkin's B-cell line (DoHH2) with a chromosomaltranslocation t(14;18)(q32;q21). *Leukemia* **5**, 221–224 (1991).

30. Karimzadeh, M., Ernst, C., Kundaje, A. & Hoffman, M. M. *Umap and Bismap: quantifying genome and methylome mappability*.

31. Hoffman, M. M., Buske, O. J. & Noble, W. S. The Genomedata format for storing large-scale functionalgenomics data. *Bioinformatics* **26**, 1458–1459 (2010).

32. Zerbino, D. R., Wilder, S. P., Johnson, N., Juettemann, T. & Flicek, P. R. The ensembl regulatory build. *Genome Biol.* **16**, 56 (2015).

33. Yates, A. *et al.* Ensembl 2016. *Nucleic Acids Res.* **44**, D710–D716 (2016).

34. Kent, W. J. *et al.* The Human Genome Browser at UCSC. *Genome Res.* **12**, 996–1006 (2002).

35. Bilmes, J. & Zweig, G. The Graphical Models Toolkit: An open source softwaresystem for speech and time-series processing. in *IEEE International Conference on Acoustics Speech andSignal Processing* IV–3916–3919.

36. Chen, X., Hoffman, M. M., Bilmes, J. A., Hesselberth, J. R. & Noble, W. S. A dynamic Bayesian network for identifying protein-binding footprints from single molecule-based sequencing data. *Bioinformatics* **26**, i334-42 (2010).

37. Oracle Grid Engine.

38. Dempster, A. P., Laird, N. M. & Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **39**, 1–38 (1977).

39. Buske, O. J., Hoffman, M. M., Ponts, N., LeRoch, K. G. & Noble, W. S. Exploratory analysis of genomic segmentations with Segtools. *BMC Bioinformatics* **12**, 415 (2011).

40. Harrow, J. *et al.* GENCODE: the reference human genome annotation for theENCODE

Project. *Genome Res.* **22**, 1760–1774 (2012).

41. Raney, B. J. *et al.* Track data hubs enable visualization of user-definedgenome-wide annotations on the UCSC Genome Browser. *Bioinformatics* **30**, 1003–1005 (2014).

42. AGP Specification 2.0.

43. Kent, W. J., Zweig, A. S., Barber, G., Hinrichs, A. S. & Karolchik, D. BigWig and BigBed: Enabling browsing of large distributed datasets. *Bioinformatics* **26**, 2204–2207 (2010).

44. Grüning, B. *et al.* Bioconda: A sustainable and comprehensive software distribution for the life sciences. *bioRxiv* 207092 (2017) doi:10.1101/207092.

45. Marinov, G. K., Kundaje, A., Park, P. J. & Wold, B. J. Large-Scale Quality Analysis of Published ChIP-seq Data. *G3 Genes|Genomes|Genetics* **4**, 209–223 (2014).

46. Landt, S. G. *et al.* ChIP-seq guidelines and practices of the ENCODE andmodENCODE consortia. *Genome Res.* **22**, 1813–1831 (2012).

47. Kharchenko, P. V, Tolstorukov, M. Y. & Park, P. J. Design and analysis of ChIP-seq experiments for DNA-bindingproteins. *Nat. Biotechnol.* **26**, 1351–1359 (2008).

48. Tarasov, A., Vilella, A. J., Cuppen, E., Nijman, I. J. & Prins, P. Sambamba: fast processing of NGS alignment formats. *Bioinformatics* **31**, 2032–2034 (2015).

49. Kidder, B. L., Hu, G. & Zhao, K. ChIP-Seq: technical considerations for obtaining high-qualitydata. *Nat. Immunol.* **12**, 918–922 (2011).

50. Andrews, S. FastQC: A quality control tool for high throughput sequencedata. (2010).

51. Carroll, T. S., Liang, Z., Salama, R., Stark, R. & de Santiago, I. Impact of artifact removal on ChIP quality metrics in ChIP-seq and ChIP-exo data. *Front. Genet.* **5**, (2014).

52. Carroll, T. S. & Stark, R. *Assessing ChIP-seq sample quality with ChIPQC*. (2016).

53. Mendoza-Parra, M.-A., Van Gool, W., MohamedSaleem, M. A., Ceschin, D. G. & Gronemeyer, H. A quality control system for profiles obtained by ChIPsequencing. *Nucleic Acids Res.* **41**, e196 (2013).

54. Stark, R. & Hadfield, J. Characterization of DNA-Protein Interactions: Design andAnalysis of ChIP-Seq Experiments. in *Field Guidelines for Genetic Experimental Designs inHigh-Throughput Sequencing* (eds. Aransay, M. A. & Lav\'\in Trueba, L. J.) 223–260 (Springer International Publishing, 2016).

55. Bailey, T. *et al.* Practical guidelines for the comprehensive analysis ofChIP-seq data. *PLOS Comput. Biol.* **9**, e1003326 (2013).

56. Ho, J. W. K. *et al.* ChIP-chip versus ChIP-seq: lessons for experimental designand data analysis. *BMC Genomics* **12**, 134 (2011).

57. Diaz, A., Nellore, A. & Song, J. S. CHANCE: comprehensive software for quality control andvalidation of ChIP-seq data. *Genome Biol.* **13**, R98 (2012).

58. R Core Team. R: A Language and Environment for Statistical Computing. (2016).

59. Huber, W. *et al.* Orchestrating high-throughput genomic analysis with Bioconductor. *Nat. Methods* **12**, 115–121 (2015).

60. Daley, T. & Smith, A. D. Predicting the molecular complexity of sequencing libraries. *Nat. Methods* **10**, 325–327 (2013).

61. Daley, T. & Smith, A. D. Modeling genome coverage in single-cell sequencing. *Bioinformatics* **30**, 3159–3165 (2014).

62. Li, Q., Brown, J. B., Huang, H. & Bickel, P. J. Measuring reproducibility of high-throughput experiments. *Ann. Appl. Stat.* **5**, 1752–1779 (2011).

63. Mendoza-Parra, M. A., Saleem, M.-A. M., Blum, M., Cholley, P.-E. & Gronemeyer, H. NGS-QC Generator: A Quality Control System for ChIP-Seq andRelated Deep Sequencing-Generated Datasets. in *Statistical Genomics: Methods and Protocols* (eds. Mathé, E. & Davis, S.) vol. 1418 243–265 (Springer New York, 2016).

64. Mendoza-Parra, M.-A. *et al.* Antibody performance in ChIP-sequencing assays: From qualityscores of public data sets to quantitative certification. *F1000Res.* **5**, 54 (2016).

65. Afgan, E. *et al.* The Galaxy platform for accessible, reproducible andcollaborative biomedical analyses: 2016 update. *Nucleic Acids Res.* (2016).

66. Troger, P., Peter, T., Hrabri, R., Andreas, H. & Piotr, D. Standardization of an API for Distributed Resource ManagementSystems. in *Seventh IEEE International Symposium on Cluster Computing andthe Grid (CCGrid '07)* (2007).

67. Gentzsch, W. Sun Grid Engine: towards creating a compute power grid. in *Proceedings First IEEE/ACM International Symposium on ClusterComputing and the Grid*.

68. Thain, D., Douglas, T., Todd, T. & Miron, L. Distributed computing in practice: the Condor experience. *Concurr. Comput.* **17**, 323–356 (2005).

69. Henderson, R. L. Job scheduling under the Portable Batch System. in *Lecture Notes in Computer Science* 279–294 (1995).

70. Zhou, S., Songnian, Z., Xiaohu, Z., Jingwen, W. & Pierre, D. Utopia: A load sharing facility for large, heterogeneousdistributed computer systems. *Softw. Pr. Exp.* **23**, 1305–1336 (1993).

71. Zhan, F. *et al.* Global gene expression profiling of multiple myeloma, monoclonal gammopathy of undetermined significance, and normal bone marrow plasma cells. *Blood* **99**, 1745–1757 (2002).

72. Han, S. S. *et al.* CDKN1A and FANCD2 are potential oncotargets in Burkitt lymphoma and multiple myeloma. *Exp. Hematol. Oncol.* **4**, 1–10 (2015).

Tables

Table 1 | Troubleshooting Table

Step	Problem	Possible Reason	Solution
8	Segway produces identical results each time it is run	Accidentally have the random seed for Segway set	Remove the Segway random seed with the command `unset SEGWAY_RAND_SEED`
8	Segway produces different results each time it is run	No random seed has been set or not enough instances have been run	If you're attempting to reproduce results consistently, consider using a random seed as described in **Step 8**. If you're not using a seed but getting significantly different results every time Segway is run, increasing your --num-instances will allow Segway to create a more consistent trained model since it gains a larger trained sample size
12	Need to recover from a crashed Segway process	This could be a cluster issue or configuration issue. Check the error log output by Segway for the reason your training/identify run crashed for details	To recover your Segway run, rename your old (train or identify) results directory to a new name which Segway can use to recover from. Specify the new directory name with the `--recover` option. Recovery only works with identical

			parameters
12	Not enough disk space or temporary directory issues while or after running Segway	Segway creates temporary observation files as part of its process.	Set the `TMPDIR` environment variable to a directory where you have write access and sufficient disk space
14	Segtools does not produce PNG images	To create PNG images, R requires an X11 display which will not be present on a headless node	Copy the necessary files to a machine that has Segtools and an X11 display available and run Segtools there. Alternatively Install the R package, Cairo, on your machine. This may require a number of additional dependencies to be installed
24	Track data from the uploaded Track hub is not displaying correctly on the UCSC Genome Browser	The server hosting the track data does not support byte serving or does not advertise it with an “Accept-Ranges” HTTP response header	Contact the hosting administrator to enable byte serving for the track hub track datasets or find an alternative web host that supports byte serving
Box 1	I have Python 2.7 but my system reports that pip is not installed	Pip does not automatically come bundled with some distributions of Python 2.7, or older versions of Python	If possible, ask your administrator to install the necessary python package containing pip. If running Python 2.7.9 or later, it is possible to install pip through the ensurepip module with ‘`python -m ensurepip --upgrade`’. For all other cases refer to the pip

			documentation
Box 1	There is a Unicode-related error when attempting to install numpy with `pip`	Older versions of pip cannot install numpy correctly due to a unicode related bug	Upgrade your pip version using '`pip install --upgrade pip`'
Box 1	The computer to run Segway on doesn't have an internet connection	Some cluster systems have strict policies regarding internet connectivity	On a machine with an internet connection download Segway and all dependencies without installation by executing '`pip install --download segway_packages segway`'. Copy the `segway_packages` directory to your target machine without an internet connection. To install Segway from the directory of python packages, execute: '`pip install --no-index --find-links segway_packages segway`'
Box 2	MACS2 produces a segfault error, for example, after attempting to 'Call peaks for each chromosome'	MACS2 will not create the necessary output directories for you	Create the directory manually using '`mkdir`' and run your MACS2 command again
Box 2	SPP installation fails with the error: 'configure: error: cannot find Boost headers version >= 1.41.0'	SPP is unable to find your Boost C++ installation	Ensure that you have properly set your `BOOST_ROOT` environment variable to include your Boost C++ installation. For example, if your Boost installation is located in directory

			`sampledir`, execute: ‘`export BOOST_ROOT=sampledir`’.
Box 4	`ChIPQCreport` reports X11 errors	To create images, `ChIPQCreport` requires an X11 display which will not be present on a headless node	Copy the necessary files to a machine that has Segtools and an X11 display available and run `ChIPQCreport` there. Alternatively, on a headless node install the R package, Cairo, on your machine. This may require a number of additional dependencies to be installed. Place the necessary commands in a R script file, `x.R`, and run it via `xvfb-run –a -s "-screen 0 1600x1200x24+32" Rscript x.R` It may be necessary to also set `options(bitmapType='cairo')` in your R script file

Table 2 | Input signal ENCODE accession IDs

Target	DOHH2	H1-hESc	GM12878
H3K4me1	ENCFF509XSM	ENCFF591KWL	ENCFF831ZHL
H3K4me3	ENCFF745GML	ENCFF372YWG	ENCFF776DPQ
H3K27ac	ENCFF890NAY	ENCFF423TVA	ENCFF340JIF
H3K27me3	ENCFF592CSV	ENCFF043JFV	ENCFF313LYI
CTCF	ENCFF884IIL	ENCFF520THR	ENCFF279CYY

Figure Legends

Figure 1: Segway workflow for producing and analysing annotations.

Figure 2: Sample output of “cat /proc/cpuinfo” on a Linux operating system. This particular CPU has two physical cores and hyperthreading (ht), marked in bold above. Hyperthreading allows software to treat one physical core as two effective cores for the operating system. As a result, this machine has four effective cores, marked in bold above.

Figure 3: Sample of the expected output from Segway training. The first lines are windows saved for consideration while training the Segway model. The last lines are the EM training (EMT) jobs submitted to a cluster system. Segway determines individual EMT job names using numbers of training instance, EMT round, and window.

Figure 4: Sample of expected output from the Segway identify task. The first lines are indexed genomic windows saved for subsequent annotation. The last lines are Viterbi jobs to run on a

cluster system. Segway numbers the Viterbi job names based on the previously indexed windows.

Figure 5: Sample of the resulting segmentation produced by Segway. After a BED file header line, each line contains information on the chromosome, region, and label assigned to each region.

Figure 6: Gaussian emission parameters learned by training a 10-label model on 5-signal dataset.

Figure 7: Distribution of the segment sizes for each label shown as a violin plot.

Figure 8: Segment counts and genome coverage per label. In blue: the fraction of segments for each label. In red: the fraction of the genome covered for each label.

Figure 9: Segment labels' enrichment relative to an idealized gene model derived from GENCODE 25. Color indicates enrichment (red) or depletion (blue).

Figure 10: Example of signal tracks and segmentations on the UCSC Genome Browser for the CDK1 locus.

Figure B3.1. Using the ENCODE DCC search panel to find a particular experiment.

Figure B3.2. Example of the type of results you would get when using ENCODE DCC website search option.

Figure B3.3. When you select the results of a search on the ENCODE DCC website, the website will direct you to a webpage on the dataset. In this webpage, you can find detailed information on aspects of the experiment. You can also download raw or processed data files (e.g. bigWig or BAM files) directly from this webpage.

Figure B6.1: Classification of regulatory elements for labels in the Ensembl Regulatory Build.

Figure B6.2: Loading cell type specific segmentation into the UCSC Genome Browser session

Figure B6.3: Comparison of Ensemble Regulatory segmentation for GM12878 cell line and DOHH2 segmentation at the MEN1 locus.

Figure B6.4: Identification of enhancer regions at the MEN1 locus. The regulatory segmentation also identifies 2 distal enhancer regions at the MEN1 gene. This figure shows segments in labels 3, 5, and 9 that overlap with 1 of these identified enhancer regions. This type of visual comparison at other genes may identify one of these labels as highly correlated to overlap CTCF binding sites or enhancer regions identified by the regulatory segmentation.

Figure B6.5: Configuring tracks in track hub for comparison of segmentations.

Figure B6.6: Percentage of regulatory elements from Ensembl segmentation that overlap with labels 1 and 9 in the DOHH2 segmentation.