Unsupervised pattern discovery in human chromatin structure through genomic segmentation

Michael M Hoffman[1], Orion J Buske[1,5], Jie Wang[2], Zhiping Weng[2], Jeff A Bilmes[3] & William Stafford Noble[1,4]

We trained Segway, a dynamic Bayesian network method, simultaneously on chromatin data from multiple experiments, including positions of histone modifications, transcription-factor binding and open chromatin, all derived from a human chronic myeloid leukemia cell line. In an unsupervised fashion, we identified patterns associated with transcription start sites, gene ends, enhancers, transcriptional regulator CTCF-binding regions and repressed regions. Software and genome browser tracks are at http://noble.gs.washington.edu/proj/segway/.

Recently, the genomics community has seen an explosion in the availability of large-scale functional genomics data. Researchers have produced genome-wide data sets on the locations of transcription-factor binding, histone modifications, open chromatin and RNA transcription with sequence census assays that measure properties genome-wide by analyzing the location and count of sequenced tags. Consequently, our representation of the whole human genome has expanded from a sequence of nucleotides, occasionally annotated with discrete features, to a collection of numerical data tracks with values at almost every part of every chromosome. We will soon have access to data from dozens of cell types[1]. How can we make sense out of this multitude of data?

A segmentation procedure provides a conceptually simple approach to finding patterns in genomic data[2–7]. The segmentation task involves finding segment boundaries while simultaneously assigning labels to the segments. One partitions the genome into nonoverlapping contiguous segments, assigning one of a finite set of labels (such as 'promoter' or 'enhancer') to each, such that regions sharing the same segment label have in common certain properties in observed data. Occam's razor implies the desirability of making the set of segment labels as small as possible, while still retaining their capacity to accurately model the observations.

Because the label at one position is unknown but is influenced by the label at the previous position, a hidden Markov model (HMM) provides a natural solution for the segmentation task. Use of an HMM, however, has some limitations. For example, HMMs handle missing data poorly, requiring interpolation and smoothing to process regions where data are not reported. Similarly, hard or soft constraints on segment lengths prove complex and difficult to implement with a simple HMM.

Dynamic Bayesian networks (DBNs), in contrast, provide a powerful framework for modeling the complex hidden relationships that explain observed data sampled at regular intervals along some axis, such as physical positions on a genome. Automatic speech recognition researchers have long used DBNs[8], and now scientists have started using them to solve biological problems[9].

A DBN is a graphical structure that depicts the conditional independence properties of multiple random variables. We can represent a standard HMM by a DBN with a hidden random variable for the HMM's hidden state and an observed random variable for the observations. With a DBN, however, we can easily incorporate complex relationships among variables at the current or nearby positions in the genome. The DBN therefore allows us to include multiple hidden variables and model their interrelationships without flattening them into a single state variable and thereby ignoring the properties implied by these interrelationships. For example, the DBN can incorporate a structured state space in which superlabels and sublabels augment a simple label set: the superlabel might represent a region of active euchromatin, and the sublabel might represent the 3′ end of an active gene in active euchromatin. In general, incorporating various types of prior knowledge and interpreting a trained model proves much easier with a DBN than with an HMM. For example, our model includes a principled mechanism for handling heterogeneous missing data. This mechanism is necessary because functional genomics assays may target only a subset of the genome or because repetitive sequences may lead to different patterns of missing data in different experiments (**Supplementary Discussion**, **Supplementary Fig. 1** and **Supplementary Table 1**).

Here we describe a segmentation method, Segway, that uses DBN techniques to simultaneously segmentat and cluster genomic data. We describe the application of Segway to human chromatin immunoprecipitation sequencing (ChIP-seq), DNase I hypersensitive site sequencing (DNase-seq) and formaldehyde-assisted isolation of regulatory elements sequencing (FAIRE-seq) data from the Encyclopedia of DNA elements (ENCODE) Project[1].

[1]Department of Genome Sciences, University of Washington, Seattle, Washington, USA. [2]Program in Bioinformatics and Integrative Biology, University of Massachusetts Medical School, Worcester, Massachusetts, USA. [3]Department of Electrical Engineering, University of Washington, Seattle, Washington, USA. [4]Department of Computer Science and Engineering, University of Washington, Seattle, Washington, USA. [5]Present address: Department of Computer Science, University of Toronto, Toronto, Ontario, Canada. Correspondence should be addressed to W.S.N. (william-noble@uw.edu).

RECEIVED 1 JULY 2011; ACCEPTED 14 FEBRUARY 2012; PUBLISHED ONLINE 18 MARCH 2012; DOI:10.1038/NMETH.1937

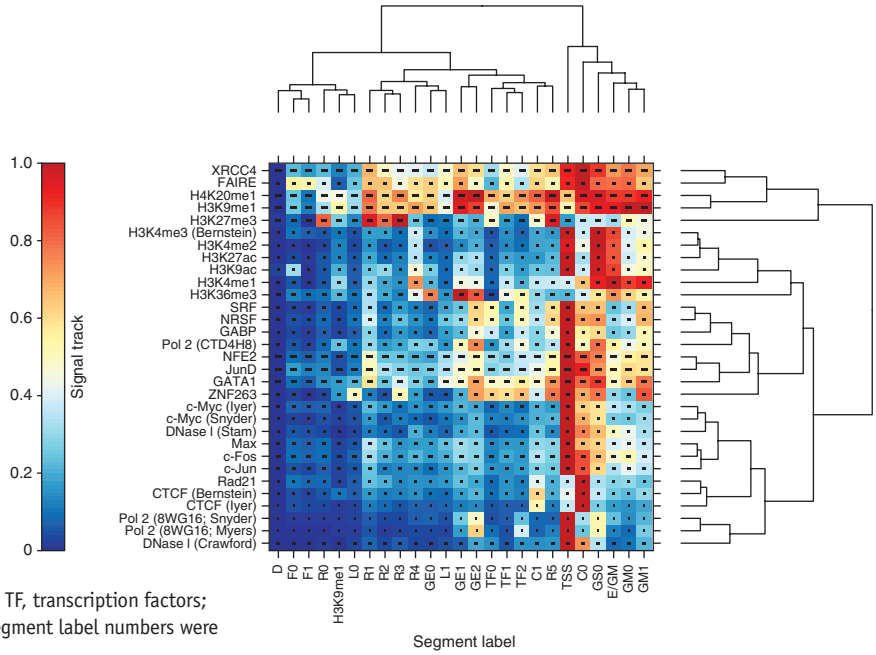

Figure 1 | Heat map of discovered Gaussian parameters in an unsupervised 25-label segmentation trained on 31 tracks of histone modification, transcription-factor binding and open chromatin signal data in 1% of the human genome. Row labels include last names of the principal investigator in whose laboratory data were generated, when assays were conducted in multiple laboratories (Stam, Stamatoyannopoulos). Each row contains parameters for one signal track, and each column contains parameters for one segment label. Within each row, we did an affine transformation, such that the largest mean was 1 and the smallest 0. The color in each cell indicates the transformed mean parameter μ according to the color bar on the left. The width of the black inner boxes is proportional to the square root of the variance parameter σ^2, after multiplying by the linear factor used in the transformation of μ. Dendrogram show a hierarchical clustering by both rows and columns. Functional categories manually assigned to segment labels: D, dead; F, FAIRE; R, repression; H3K9me1, histone 3 lysine 9 onomethylation; L, low; GE, gene end; TF, transcription factors; C, CTCF; GS, gene start; E, enhancer; GM, gene middle; segment label numbers were assigned arbitrarily.

We carried out unsupervised training on 1% of the human genome with the Segway model and 31 ENCODE signal tracks that showed locations of histone modifications, transcription-factor binding and open chromatin. Whereas large numbers of labels may have statistical support, we arbitrarily fixed the number of labels at 25 so that the set of labels would be sufficiently small to remain interpretable by biologists. Our method aims to reduce complex data for easier interpretation, and using a small number of labels served this purpose. The discovered parameters characterized a diverse set of biological patterns (**Fig. 1**). We then identified the most probable path of segment labels given the trained parameters and observed signal.

We refer to the resulting assignment of labels to nonoverlapping genomic regions as a segmentation (**Supplementary Results**, **Supplementary Fig. 2** and **Supplementary Table 2**). The labels allow one to easily identify other regions that show similar signal patterns in the observation tracks. By examining the learned parameters directly (**Fig. 1** and **Supplementary Fig. 3**) and comparing the segmentation to public annotations (**Fig. 2** and **Supplementary Figs. 3–5**), we assigned functional categories to groups of segment labels on the basis of notable features. Many of these labels recapitulated known patterns in the chromatin literature, and some represented new patterns.

Notably, Segway 'rediscovered' protein-coding genes, as the unsupervised segmentation included chromatin states associated with the starts, middles and ends of genes (**Fig. 2**) found without direct recourse to information traditionally used to find genes, such as primary sequence, similarity to mRNAs or genome sequences of other species. Several labels tended to reside in particular locations in protein-coding genes, and the discovered transition parameters of the model increased the probability of moving from the labels that tended to reside in 5′ ends of genes to the labels found more often in 3′ ends of genes. We also found expected patterns of transcription factor binding near the transcription start site (TSS), histone H2A.Z associated with the TSS[10], and histone mark H3K79me2 associated with the gene start[10] (**Supplementary Fig. 6**). The patterns of chromatin structure in protein-coding genes fit well with existing knowledge (**Supplementary Results** and **Supplementary Figs. 3** and **7**).

Many of the chromatin states that seemed, at first glance, to be associated with protein-coding genes turned out to be associated only with genes active in the tissue type assayed for the data used in the segmentation (**Supplementary Results**). Segway also discovered patterns associated with specific functional elements, such as enhancers, insulators, regions of repressed gene expression and regions of no or very low biochemical activity ('dead' regions) (**Supplementary Results**, **Supplementary Figs. 5** and **8** and **Supplementary Table 3**).

We used Segway to generate hypotheses about the role of individual sequences in promoting transcription that we then tested experimentally. We began by identifying expressed and unexpressed genes using RNA data. To determine whether specific sequences identified by segmentation labels correctly identified these genes as being expressed or not expressed, we used transient transfection followed by luciferase assays to test transcription of small (~1-kilobase) constructs that overlapped the gene's TSS and either a segmentation TSS label (24 predictions of expression, referred to as 'positive predictions') or an 'R' (repressed) label (as defined by 67 predictions of no expression, referred to as 'negative predictions'). Every positive prediction resulted in substantial expression of a reporter, a majority of the negative predictions resulted in no substantial expression, and a larger majority of negative predictions not overlapping CpG islands resulted in no substantial expression (**Supplementary Fig. 9**). Positive predictions showed higher median expression activity than negative predictions.

Segway differs substantially from several previously described methods for jointly analyzing chromatin data (**Supplementary Discussion**). For example, Segway solves a fundamentally different problem than ChromaSig[11], which does not attempt to fully partition the genome or integrate arbitrary combinations of functional genomics data. Segway is most similar to an HMM-based method called ChromHMM[5]. However, Segway and ChromHMM differ in several respects, chiefly in the relative resolution. Segway operates

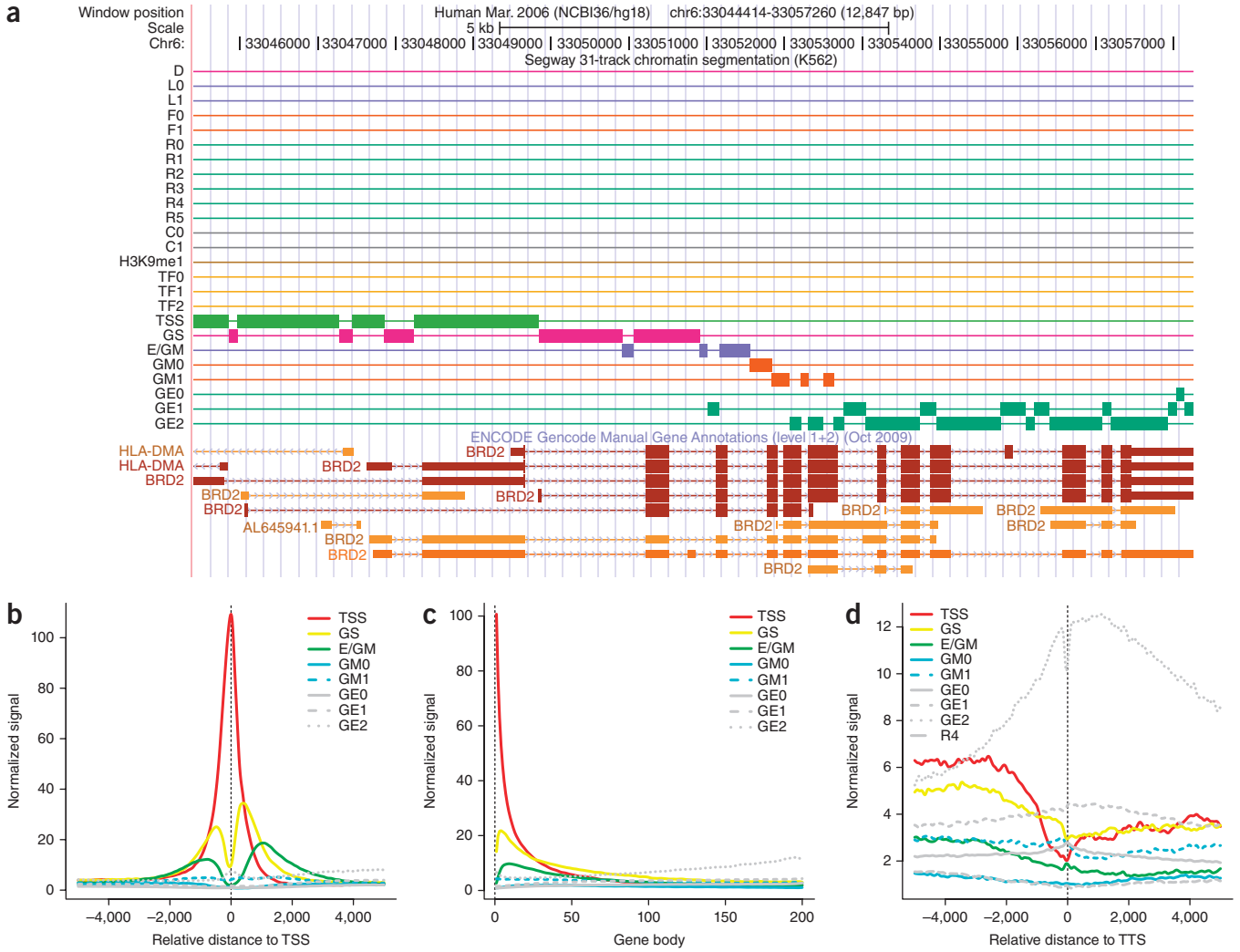

Figure 2 | Gene structure in Segway labels. (**a**) Segmentation of the human *BRD2* locus, as shown in the University of California, Santa Cruz (UCSC) Genome Browser. The first three rows show the position along chromosome 6 and a scale bar. The next 25 rows correspond to the labels of the segmentation, with a thick bar present at any position where the segmentation has the corresponding label[5]. Labels are colored by functional category. The bottom rows show GENCODE manual gene annotations. (**b**–**d**) Aggregation of gene-related Segway labels with respect to the starts, middles and ends of protein-coding genes from GENCODE v7. Each panel plots the mean density of a set of gene-related Segway labels as a function of genomic location relative to the TSS (**b**), the gene body, extending from the TSS to the transcription termination site (TTS) (**c**) and the TTS (**d**). The frequency of each label was normalized by the relative proportion of the genome that is covered by that label. Genomic distances in **c** were scaled to a proportion of gene length.

on the full data set, whereas ChromHMM works at 200-base-pair (bp) resolution and condenses each data track to a single binary datum within each 200-bp window. Consequently, Segway provides a finer-grained segmentation. Whereas the segments in a published ChromHMM segmentation[5] had a minimum length of 200 bp, a mean of 4,862 bp and a median of 800 bp, Segway segments had a mean length of 168 bp and a median of 124 bp.

We compared the behavior of Segway and ChromHMM genome-wide and, specifically, at three separate loci (**Supplementary Results** and **Supplementary Fig. 10**). The Segway methodology resulted in several distinct advantages, such as the ability to detect elements at subnucleosomal resolution, better precision in finding known elements and superior handling of missing data.

We found that Segway reduced heterogeneous genomic data sets into understandable patterns with clear biological implications. The flexibility of the DBN suggests a potential solution to the problem of learning large numbers of segment labels while retaining comprehensibility. Extension to a hierarchical segmentation would allow the learning of many sublabel patterns, while keeping a higher-level, yet smaller-order, structure that a researcher could analyze and understand more readily. Whereas effective hyperparameter setting for hierarchical segmentation requires additional research, we have already implemented the capability for hierarchical training and identification in Segway.

METHODS

Methods and any associated references are available in the online version of the paper at http://www.nature.com/naturemethods/.

Note: Supplementary information is available on the Nature Methods website.

ACKNOWLEDGMENTS

We thank P.J. Collins for assistance with transient transfection assays, S. Djebali for processing data, C.E. Grant for motif analysis, A. Kundaje for helpful

suggestions, and members of the ENCODE Project Consortium, the ENCODE Data Coordination Center and the US National Human Genome Research Institute for providing early public access to the unpublished data used in this work. This work used data produced in the laboratories of B.E. Bernstein (Broad Institute of the Massachusetts Institute of Technology and Harvard University), M.P. Snyder (Stanford University), R.M. Myers (HudsonAlpha Institute for Biotechnology), P.J. Farnham (University of Southern California), V.R. Iyer (University of Texas at Austin), G.E. Crawford (Duke University), J.D. Lieb and T.S. Furey (University of North Carolina at Chapel Hill), J.A. Stamatoyannopoulos (University of Washington), P. Carninci (RIKEN), T.R. Gingeras (Cold Spring Harbor Laboratory), and A. Sidow (Stanford University). This publication was made possible by grants 004695, 004561 and 006259 from National Human Genome Research Institute.

AUTHOR CONTRIBUTIONS

M.M.H., W.S.N. and J.A.B. conceived of the project; M.M.H., W.S.N. and Z.W. designed computational and biological experiments. M.M.H., J.A.B., O.J.B. and J.W. developed software used in this work; M.M.H., O.J.B. and J.W. conducted computational experiments and analyzed data; and M.M.H., W.S.N., Z.W., J.A.B., O.J.B. and J.W. wrote the manuscript.

COMPETING FINANCIAL INTERESTS

The authors declare no competing financial interests.

Published online at http://www.nature.com/naturemethods/.
Reprints and permissions information is available online at http://www.nature.com/reprints/index.html.

1. ENCODE Project Consortium. *PLoS Biol.* **9**, e1001046 (2011).
2. Day, N., Hemmaplardh, A., Thurman, R.E., Stamatoyannopoulos, J.A. & Noble, W.S. *Bioinformatics* **23**, 1424–1426 (2007).
3. Erdman, C. & Emerson, J.W. *Bioinformatics* **24**, 2143–2148 (2008).
4. Jaschek, R. & Tanay, A. in *Research in Computational Molecular Biology, Lecture Notes in Computer Science* Vol. 5541 (ed. Batzoglou, S.) 170–183 (Springer, Berlin, 2009).
5. Ernst, J. & Kellis, M. *Nat. Biotechnol.* **28**, 817–825 (2010).
6. Filion, G.J. *et al. Cell* **143**, 212–224 (2010).
7. Kharchenko, P.V. *et al. Nature* **471**, 480–485 (2011).
8. Bilmes, J. & Bartels, C. *IEEE Signal Process. Mag.* **22**, 89–100 (2005).
9. Reynolds, S.M., Käll, L., Riffle, M.E., Bilmes, J.A. & Noble, W.S. *PLOS Comput. Biol.* **4**, e1000213 (2008).
10. Wang, Z., Schones, D.E. & Zhao, K. *Curr. Opin. Genet. Dev.* **19**, 127–134 (2009).
11. Hon, G., Ren, B. & Wang, W. *PLOS Comput. Biol.* **4**, e1000201 (2008).

ONLINE METHODS

Signal generation. We selected data from 31 ChIP-seq, DNase-seq and FAIRE-seq assays carried out in the chronic myeloid leukemia cell line K562 by the ENCODE Project[1]. We downloaded tagAlign files from the ENCODE Data Coordination Center[12] (**Supplementary Table 4**). We used Wiggler (http://code.google.com/p/align2rawsignal/) to convert tag alignment positions into an extended reads per base signal for each position of the genome, pooling multiple replicates when available. We then loaded the signal data into a Genomedata[13] archive. We include the URL for each of the tagAlign and signal files used in **Supplementary Table 3**.

Signal transformation. To reduce the distorting effects of high data values in sequence census assay data, Segway first transforms all data values with the inverse hyperbolic sine function $\operatorname{asinh} x = \ln(x + \sqrt{(x^2 + 1)})$. This function has the compressing effect of a function like $\ln x$ for large values of x but has much less of a compressing effect for small values. Additionally, it is defined on the entire real number line, preserving the input data values as positive or negative. One therefore avoids the discontinuities inherent in the use of a simple logarithmic transformation and the necessity of first adding some offset to deal with zero values[14].

The core model. Segway carries out training and inference on a DBN model designed for genomic segmentation (**Supplementary Fig. 11**). The core of the default Segway DBN has some similarity to an HMM, with multiple observation tracks and a number of discrete hidden variables. The default Segway DBN has a segment label backbone defined for each base pair of a genomic region. This hidden variable emits an observation variable for each observation track present at that position. An observation track is a sequence of numerical observations, such as the number of sequence tags overlapping successive genomic positions, or the intensity of a microarray probe associated with a position. In the default model, the ith observation track is represented by the sequence of random variables $X_{1:T}^{(i)} = (X_1^{(i)}, \ldots, X_T^{(i)})$. Some positions t may not correspond to a defined value of $X_t^{(i)}$. Examples include unmappable sequence in sequence census assays, or sequence not represented by a probe in a microarray assay. To explicitly model these missing data, we use an indicator variable to mark whether $X_t^{(i)}$ is defined ($\mathring{X}_t^{(i)} = 1$) or undefined ($\mathring{X}_t^{(i)} = 0$). The observation variables at every position have dependency only on the hidden segment label variable at that position Q_t (**Supplementary Fig. 11**). When the indicator variable ($\mathring{X}_t^{(i)} = 0$, then $X_t^{(i)}$ does not have a dependence on Q_t. In that case, the conditional parent arc from Q_t to $X_t^{(i)}$ is, for all intents and purposes, nonexistent in the DBN[15]. Segway models the asinh-transformed data with an $m \times n$ matrix of scalar Gaussians, with one Gaussian for each combination of n observation tracks and m segment labels. Each Gaussian has a mean parameter μ, and variance parameter σ^2, which control the signal distribution that the Gaussian emits. Segway ties variance parameters such that they are all equal for a given observation track.

Weighting. Segway weights the contribution of each observation variable such that tracks with different numbers of data points still have roughly the same contribution to the overall likelihood of the model. It does this by exponentiating during inference calculations every occurrence of an original probability $P(X_t^{(i)} \mid Q_t)$ for some track i by the number of data points for that track $N^{(i)} = \Sigma_t(\mathring{X}_t^{(i)})$ divided by the maximum number of data points for any particular track $N^\star = \max_j N^{(j)}$. In other words, it replaces the original probability with a new probability:

$$P'(X_t^{(i)} \mid Q_t) = (P(X_t^{(i)} \mid Q_t))^{\frac{N^{(i)}}{N^\star}}$$

Because probabilities are always in the range [0, 1], an increase in weight will result in a decreased probability of the model overall, but when the weight is held constant, changes in the probabilities exponentiated to higher weights will affect the overall probability more than those exponentiated to lower weights.

The duration model. The Segway model includes additional random variables that allow tuning beyond the simple HMM-like core model. A discrete countdown variable C_t allows the specification of minimum or maximum segment length. This variable begins with an initial value dependent on Q_t, and decreases by 1 at every position where the ruler marker variable M_t is 1 ('ruler marks'). Decreasing the countdown variable only upon the occurrence of an occasional ruler mark decreases the countdown variable state space while still allowing longer minimum or maximum segment lengths. This heuristic greatly decreases computational complexity. A binary segment transition variable J_t can either force the segment label to change at the current position ($J_t = 1$), or prevent it from changing ($J_t = 0$). Segway generates a conditional probability table $P(J_t = 1 \mid Q_{t-1}, C_{t-1})$ that maps each (Q_{t-1}, C_{t-1}) tuple to one of three rules that determine the value of J_t: (i) force, $P(J_t = 1) = 1$; (ii) prevent, $P(J_t = 0) = 1$; or (iii) allow, $P(J_t = 1) = 1/(1 + L)$. The duration in which the segment label is unchanged following the point where the 'allow' rule becomes active follows a geometric distribution with an expected length of the L value[9,16]. In this study, we set $C_t = 10$ and the 'force' rule at the beginning of each segment. We also set the ruler marker every ten positions. This setting allows the enforcement of large minimum segment lengths with only a small increase in state space and computational complexity. When C_t decreases to 0, we set the 'allow' rule with $L = 100{,}000$ to model broader behavior when the observation variables do not control the segment label as they usually do. Despite the enforcement of a minimum duration before allowing a change in segment label, it is still possible to have very short segments at the end of input sequences, such as at the 3′ ends of genome assembly scaffolds.

DBN inference. Segway uses the graphical models toolkit (GMTK)[17] to do expectation-maximization training[18] and Viterbi decoding[19] on the Segway models. Segway uses model structure files in the GMTKL specification language, and users may modify them or replace them with a DBN of arbitrary complexity. For decoding, Segway divides large sequences into nonoverlapping windows of no more than 2 Mb in size each so that the necessary computation can fit into available memory.

Training and parameters. We carried out ten separate instances of expectation-maximization training, each with different initial parameters, and each instance had no more than 100 iterations. Segway picks an initial mean parameter μ for each track

uniformly from the values in $\tilde{\mu} \pm 0.2\ \tilde{\sigma}$ for the empirical mean $\tilde{\mu}$ and s.d. $\tilde{\sigma}$ of the training data. It sets variance parameters σ^2 to the difference between the maximum and minimum data values within a particular observation track. We did training on the ENCODE pilot regions (http://hgdownload.cse.ucsc.edu/goldenPath/hg18/database/encodeRegions.txt.gz), which represent 1% of the human genome (30 Mb). Segway sets the probabilities $P(Q_1 = q)$ of starting a sequence with each segment label $q \in [1, n]$ to $1/n$, and does not adjust these probabilities during training. The conditional probability table $P(Q_t \mid Q_{t-1})$, which Segway only uses when the segment transition variable $J_t = 1$, has only zeroes in its diagonal. This prevents a 'transition' that does not change the segment label. Segway initially sets the other values equally, but expectation-maximization training will adjust them. A training instance continues until the difference between the difference between the current likelihood L_n and the likelihood from the previous round L_{n-1} diminishes such that:

$$\left| \frac{\log L_n - \log L_{n-1}}{\log L_{n-1}} \right| < 10^{-5}$$

We designated the training instance with the highest final likelihood as the winning training instance and used its parameters to create the primary segmentation discussed here.

Segment identification and visualization. We carried out Viterbi decoding on 92% of the genome (2,828 Mb), excepting excluded blacklist regions downloaded from the ENCODE Data Coordination Center[12] (http://hgdownload.cse.ucsc.edu/goldenPath/hg18/encodeDCC/wgEncodeMapability/wgEncodeDukeRegionsExcluded.bed6.gz). These regions contain repetitive elements that cause artifactually high signal in sequence census assays. We created the primary segmentation discussed here with the winning training instance parameters but did Viterbi decoding with parameters from the other nine training instances for comparison purposes. Segway produces segmentations in BED format, designed for easy uploading and display on the UCSC Genome Browser[20].

Distributed computing. Segway uses the Grid Engine or Platform LSF distributed computing systems through the Distributed Resource Management Application API (http://www.drmaa.org/) interface. Using these systems, Segway trains or decodes on multiple sequences and multiple initial parameter values at once, leading to a considerable savings in wall clock time. Segway automatically tunes memory usage by first queuing jobs with a small amount of memory (such as 2 gigabytes) and then repeating with ever larger amounts of memory if GMTK cannot complete inference with the smaller amount. This approach allows efficient resource management even when one cannot easily predict the memory usage in advance.

Genome assembly and annotations. We did all training, segmentation, and analysis on the NCBI36 assembly of the human genome (hg18). We used GENCODE[21] version 3c gene annotations (http://www.gencodegenes.org/releases/3c.html) for aggregation and overlap analyses, as well as track display. We used ENCODE Project CAGE[22] data (ftp://genome.crg.es/pub/Encode/data_analysis/TSS/Mar-July2010/Gencodev3c_TSS.gff) for overlap analyses. We downloaded PhastCons scores for multiple alignments of 45 vertebrates to the human genome[23] from the UCSC Genome Browser. We downloaded nucleosome positioning signal data collected in K562 by micrococcal nuclease digestion and high-depth sequencing from the UCSC Genome Browser (http://genome.ucsc.edu/cgi-bin/hgFileUi?db=hg19&g=wgEncodeSydhNsome). We produced genome-wide visualizations and summary statistics using Segtools[24]. We based precision and recall statistics on the overlap of the bases within a segment against some annotation, where true positives (TP) are positions where the segment label overlaps the annotation, false positives (FP) are positions where the segment label does not overlap the annotation, and false negatives (FN) are positions where the annotation does not overlap the segment label. We expanded point annotations (such as TSSs) by 500 bp on each side before calculating these statistics. Precision (also known as positive predictive value) is defined as TP/(TP + FP). Recall (also known as sensitivity) is defined as TP/(TP + FN). We used precision and recall for evaluation instead of receiver operating characteristic curves, because precision and recall can provide a more informative measurement of performance[25]. We defined TSSs with reproducible CAGE support as those with at least two cytoplasmic CAGE tags.

Transient transfection luciferase assays. We predicted the promoter activity of sequences in the segmentation selected by the following procedure. We started with the segments that overlapped both GENCODE manually annotated protein-coding TSSs and a catalog of existing constructs from SwitchGear Genomics. We then excluded the segments that overlapped RepeatMasker 3.27 regions downloaded from the UCSC Genome Browser (http://hgdownload.cse.ucsc.edu/goldenPath/hg18/database/). We defined positive predictions as those segments with a TSS label that overlapped a GENCODE TSS with at least two K562 cytosolic poly(A)$^+$ CAGE tags and a GENCODE TSS for a gene with at least K562 RNA-seq reads per kilobase per million reads (RPKM) above the 90th percentile. We defined negative predictions as those segments with R labels that overlapped a GENCODE TSS with no CAGE tags in any K562 cellular compartment, and a GENCODE TSS for a gene with 0.0 RPKM K562 RNA-seq reads. We further divided negative predictions by whether they had GM12878 RNA-seq read RPKM over the 10th percentile or not. This process resulted in a candidate set of 26 positive predictions and 74 negative predictions. SwitchGear Genomics did transient transfection luciferase assays in triplicate on 24 positive predictions, 67 negative predictions, and 5 control constructs, narrowing from the candidate set on the basis of constructs in stock. They mixed 100 ng luciferase reporter DNA, 0.4 µm Lipofectamine LTX (Invitrogen), and 0.1 µl PLUS Reagent (Invitrogen), and incubated it for 30 min. SwitchGear then thoroughly mixed this suspension with 25,000 freshly counted K562 cells resuspended in warm, complete media. They dispensed 100 µl of the combined suspension into replicate white 96-well assay plates and incubated them at 37 °C for 24 h. They then removed the plates from the incubator, added SteadyGlo reagent (Promega) to each well, and kept the plates in the dark for >30 min. They then read the plates on an LmaxII-384 luminometer (Molecular Devices). We considered the resulting data points substantial when their activity exceeded a threshold of the mean control activity plus six times the s.d. of control activity.

Gene Ontology enrichment analysis. We analyzed the sets of GENCODE TSSs that overlapped with each segment label for

doi:10.1038/nmeth.1937

the enrichment of GO terms. We used the Ensembl[26] Perl API to convert GENCODE transcript identifiers to UniProt[27] identifiers. We then used FuncAssociate[28] to find enrichment in 25 resulting gene sets, using 10,000 resamplings for each to determine hypothesis-wise *P* values. We then multiplied these *P* values by 25 as a Bonferroni correction for testing multiple hypotheses.

Transcription factor motif analysis. We obtained position specific scoring matrices (PSSMs) from the TRANSFAC 10.2 (ref. 29) and JASPAR CORE 2009 (ref. 30) databases. Using FIMO[31], we scanned the human genome for significant matches to the PSSM ($q < 0.1$). From these candidate motifs, we selected known vertebrate *cis*-regulatory motifs with at least 1,000 significant matches in the genome. We used the genome structure correction marginal region overlap test in block_bootstrap[32], as implemented in Statmap (http://www.statmap-bio.org/), to test each segment label for enrichment in matches to each of the selected motifs. We then applied the Bonferroni correction to the resulting *P* values.

Statistical analysis. We carried out statistical analysis with R 2.11 (http://www.rproject.org/).

12. Raney, B.J. *et al. Nucleic Acids Res.* **39**, D871–D875 (2011).
13. Hoffman, M.M., Buske, O.J. & Noble, W.S. *Bioinformatics* **26**, 1458–1459 (2010).
14. Johnson, N.L. *Biometrika* **36**, 149–176 (1949).
15. Bilmes, J. in *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence* (eds. Boutilier, C. & Goldszmidt, M.) 38–45 (Morgan Kaufmann, San Francisco, 2000).
16. Grundy, W.N., Bailey, T.L., Elkan, C.P. & Baker, M.E. *Comput. Appl. Biosci.* **13**, 397–406 (1997).
17. Bilmes, J. & Bartels, C. in *UAI '03, Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence* (eds. Meek, C. & Kjærulff, U.) 47–56 (Morgan Kaufmann Publishers, San Francisco, 2003).
18. Dempster, A.P., Laird, N.M. & Rubin, D.B. *J. Royal Stat. Soc. B* **39**, 1–22 (1977).
19. Viterbi, A.J. *IEEE Trans. Inf. Theory* **13**, 260–269 (1967).
20. Fujita, P.A. *et al. Nucleic Acids Res.* **39**, D876–D882 (2011).
21. Harrow, J. *et al. Genome Biol.* **7**, S4.1–S4.9 (2006).
22. Takahashi, H., Kato, S., Murata, M. & Carninci, P. *Methods Mol. Biol.* **786**, 181–200 (2012).
23. Siepel, A. *et al. Genome Res.* **15**, 1034–1050 (2005).
24. Buske, O.J., Hoffman, M.M., Ponts, N., Roch, K.G.L. & Noble, W.S. *BMC Bioinformatics* **12**, 415 (2011).
25. Davis, J. & Goadrich, M. in *Proceedings of the 23rd International Conference on Machine Learning* 233–240 (ACM, New York, 2006).
26. Flicek, P. *et al. Nucleic Acids Res.* **39**, D800–D806 (2011).
27. UniProt Consortium. *Nucleic Acids Res.* **39**, D214–D219 (2011).
28. Berriz, G.F., Beaver, J.E., Cenik, C., Tasan, M. & Roth, F.P. *Bioinformatics* **25**, 3043–3044 (2009).
29. Wingender, E. *et al. Nucleic Acids Res.* **28**, 316–319 (2000).
30. Sandelin, A., Alkema, W., Engstrom, P., Wasserman, W. & Lenhard, B. *Nucleic Acids Res.* **32**, D91–D94 (2004).
31. Grant, C.E., Bailey, T.L. & Noble, W.S. *Bioinformatics* **27**, 1017–1018 (2011).
32. Bickel, P.J., Boley, N., Brown, J.B., Huang, H. & Zhang, N.R. *Ann. Appl. Stat.* **4**, 1660–1697 (2010).