Segway and the Graphical Models Toolkit: A framework for probabilistic genomic inference

Michael M. Hoffman

Princess Margaret Cancer Centre

Vector Institute

Department of Medical Biophysics
Department of Computer Science
University of Toronto

https://hoffmanlab.org/

@michaelhoffman

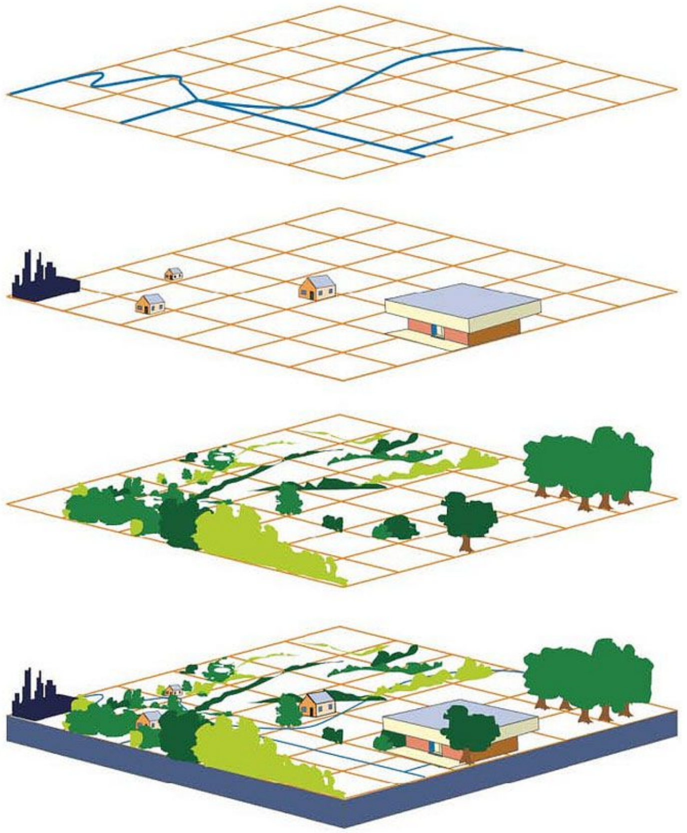
Geographical maps have…

street data

+ buildings data

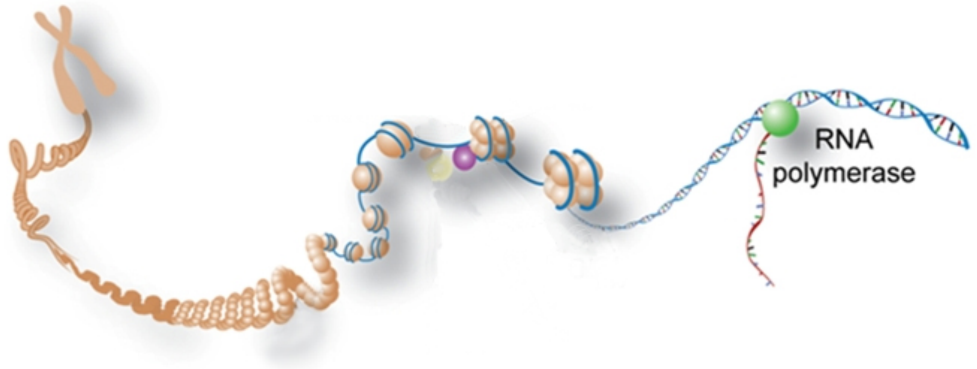
+ vegetation data

= integrated map

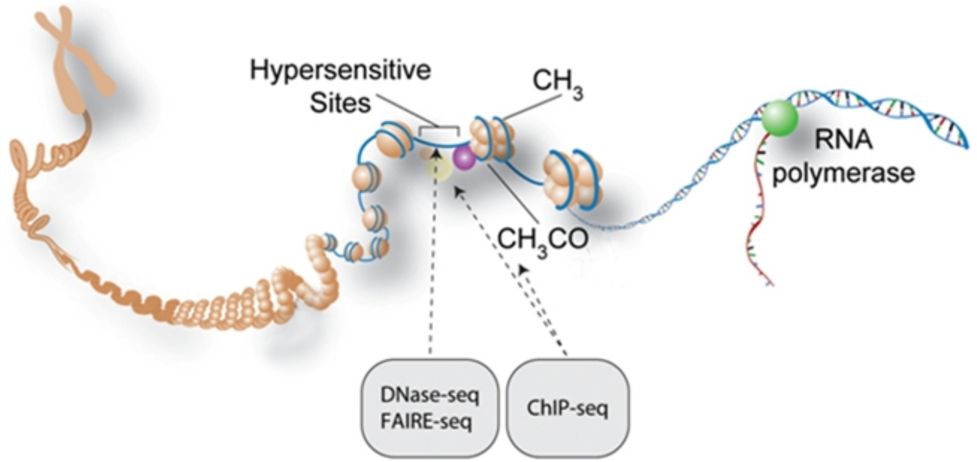


Rachel Chan
Figures from 1) National Geographic Society (2011) GIS.

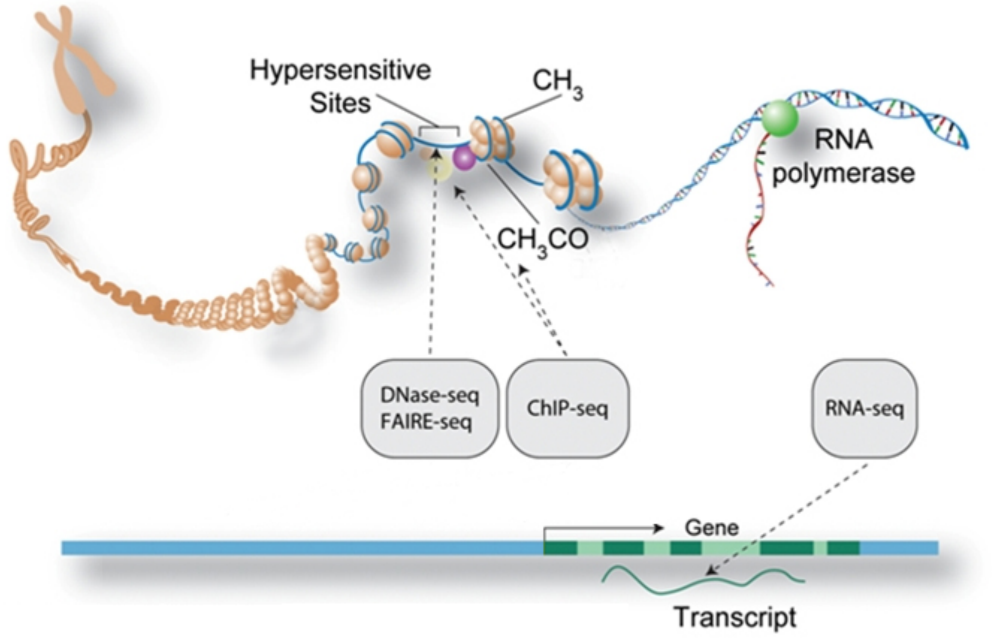
Functional genomics



Functional genomics

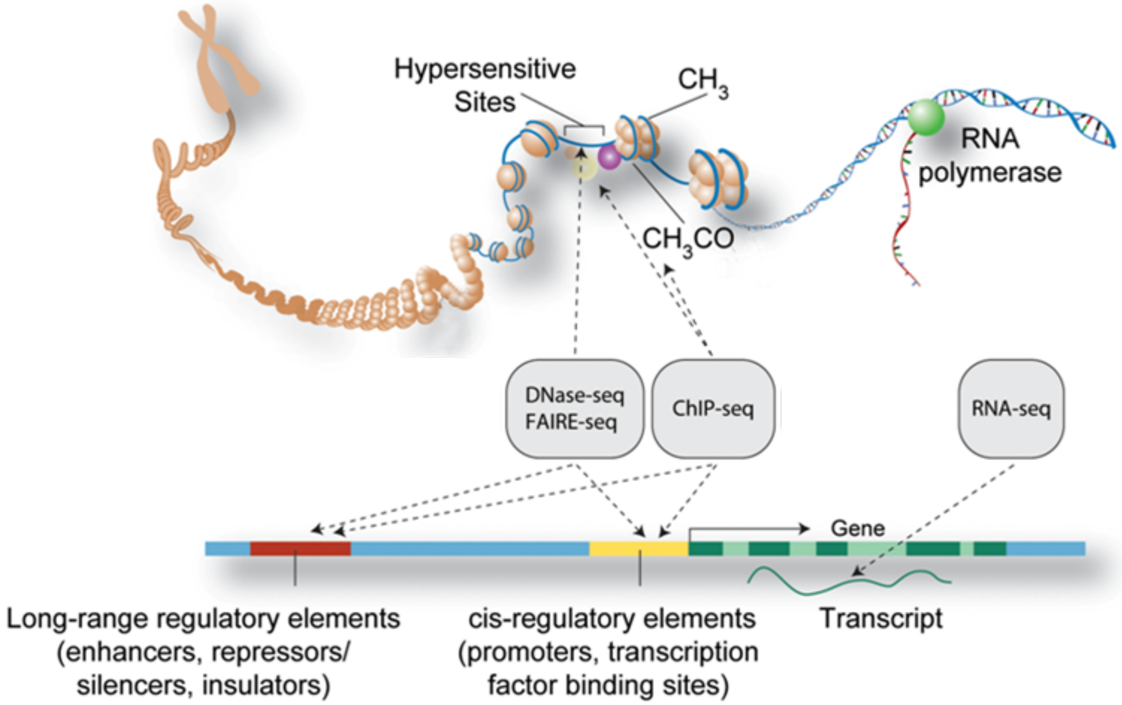


Functional genomics



ENCODE Project Consortium 2011. *PLoS Biol* 9:e1001046.

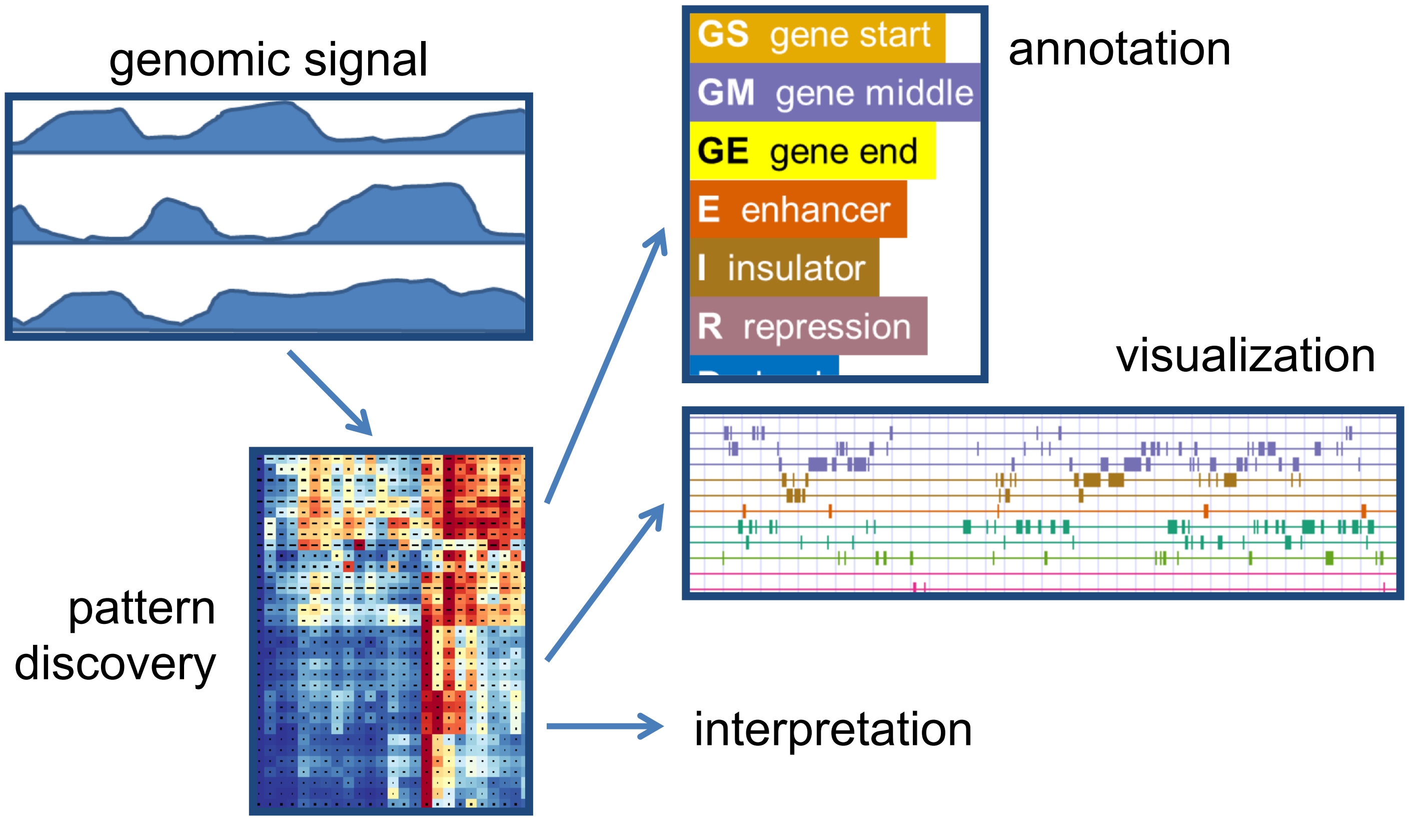
Functional genomics



Hypersensitive Sites
CH_3
CH_3CO
RNA polymerase
DNase-seq FAIRE-seq
ChIP-seq
RNA-seq
Gene
Transcript
Long-range regulatory elements (enhancers, repressors/ silencers, insulators)
cis-regulatory elements (promoters, transcription factor binding sites)

ENCODE Project Consortium 2011. *PLoS Biol* 9:e1001046.

Semi-automated genome annotation

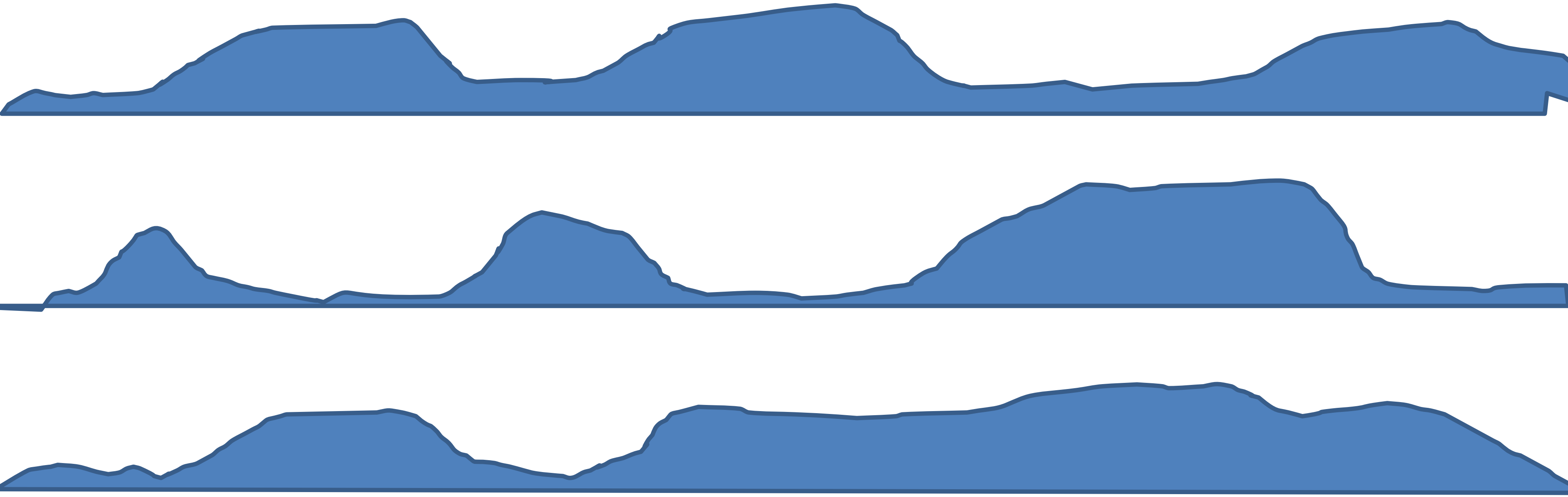


Segway

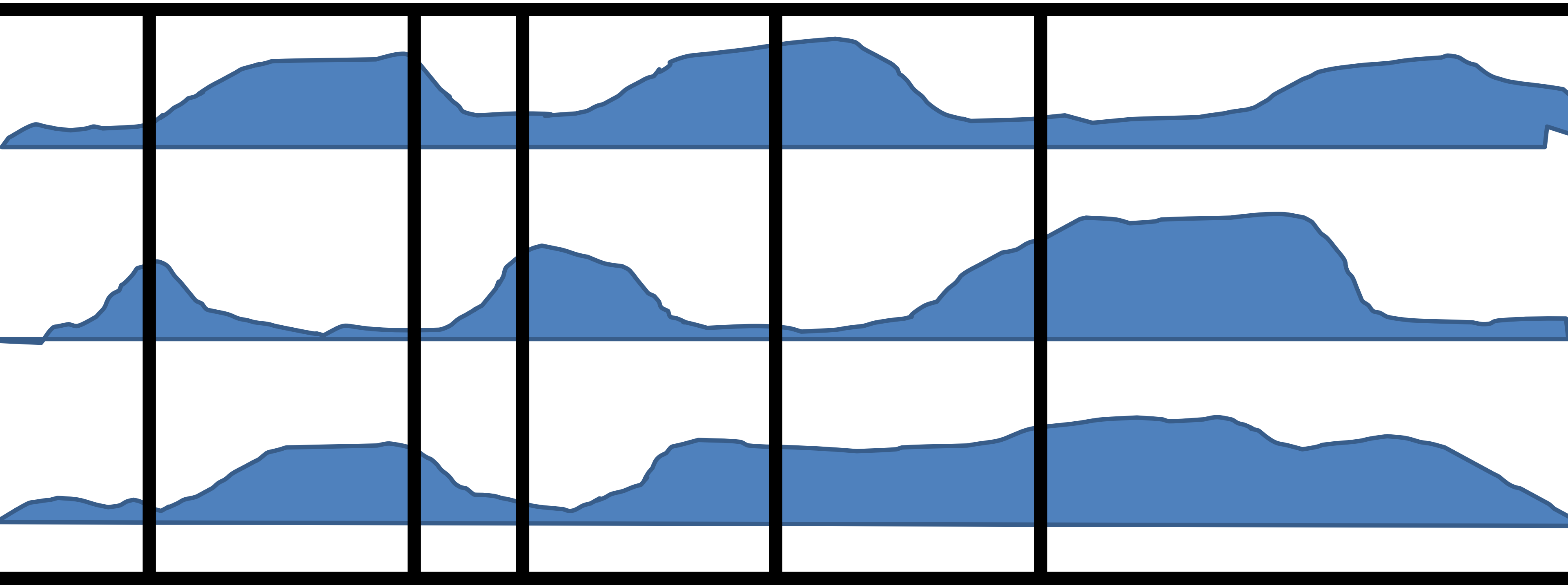
A way to segment the genome

https://segway.hoffmanlab.org/
Hoffman MM *et al*. 2012. *Nat Methods* 9:473.

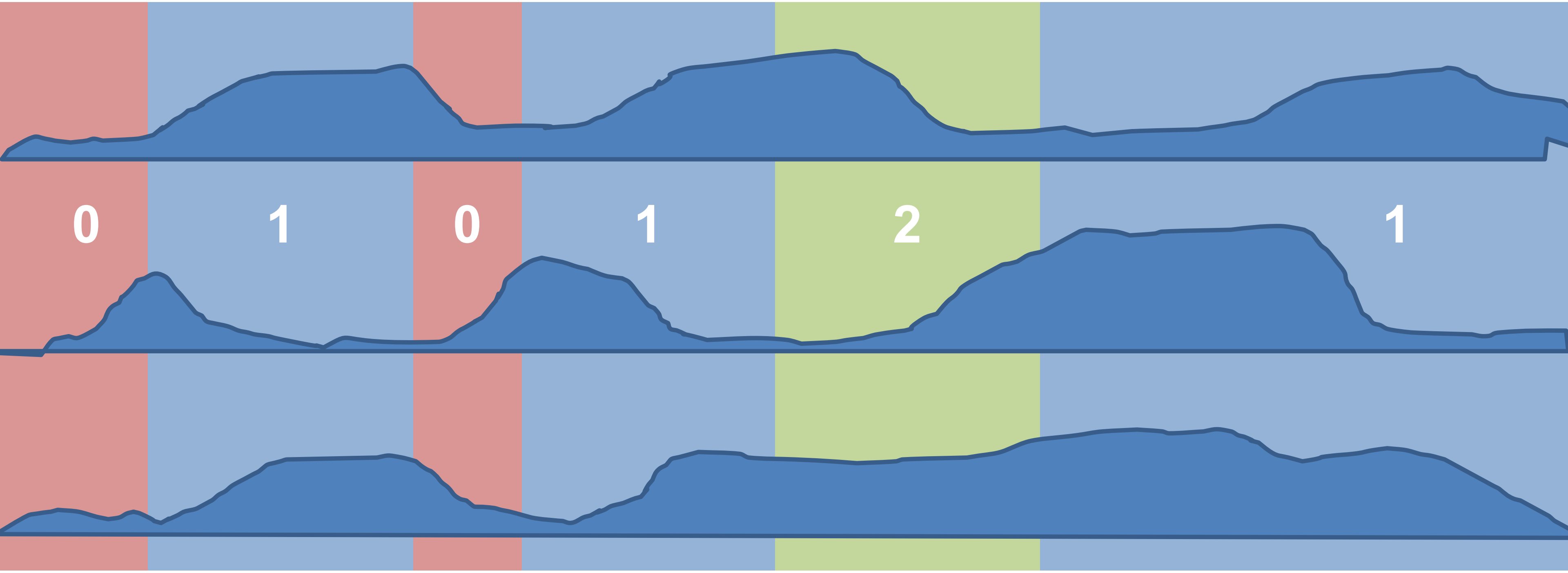
Genomic segmentation



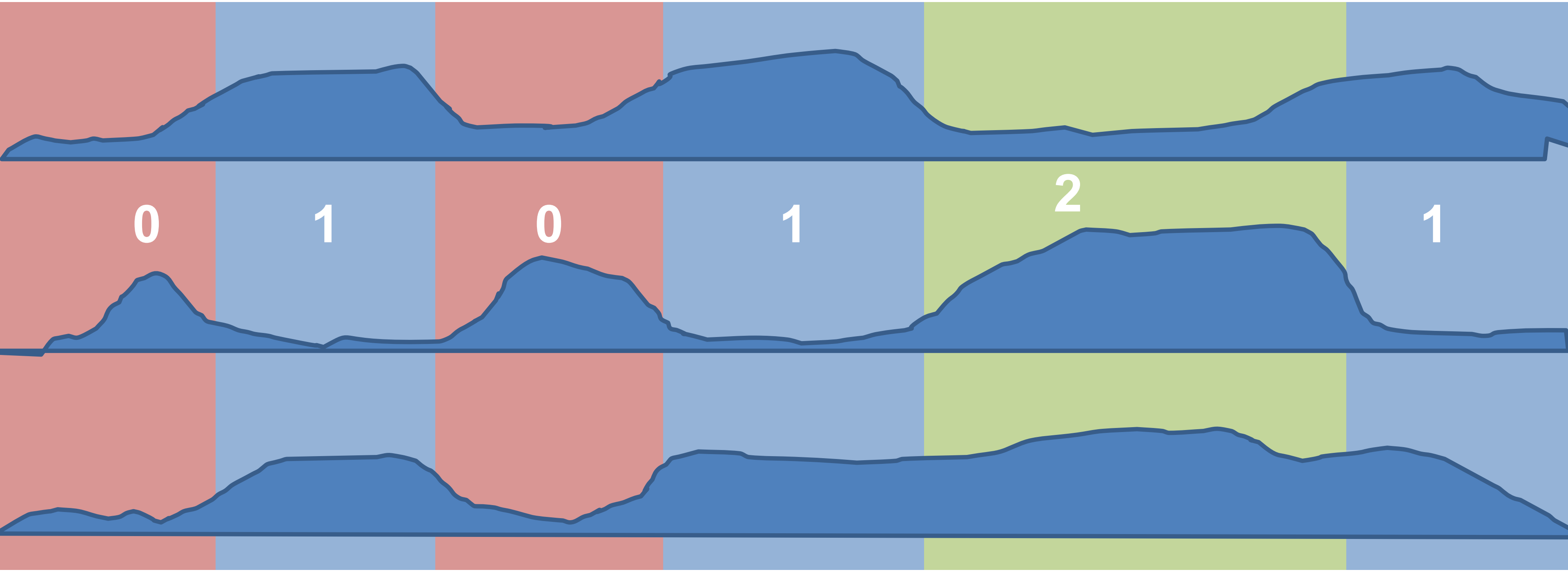
Nonoverlapping segments



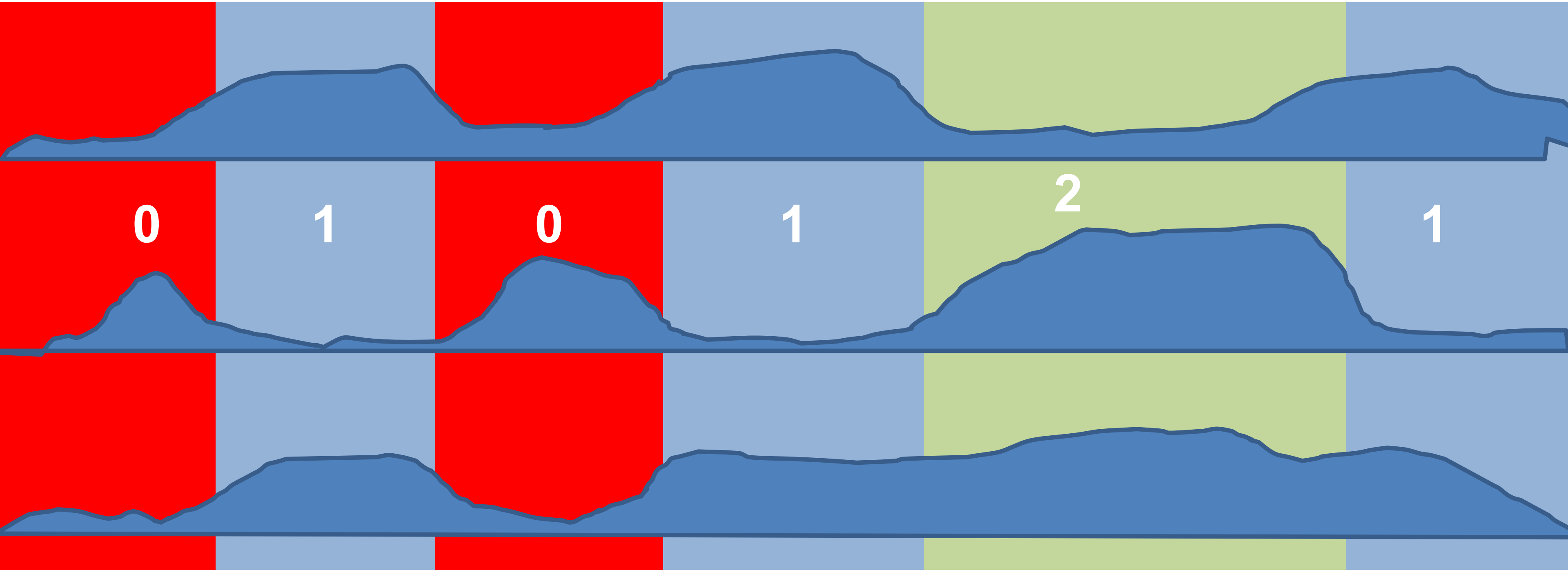
Finite number of labels



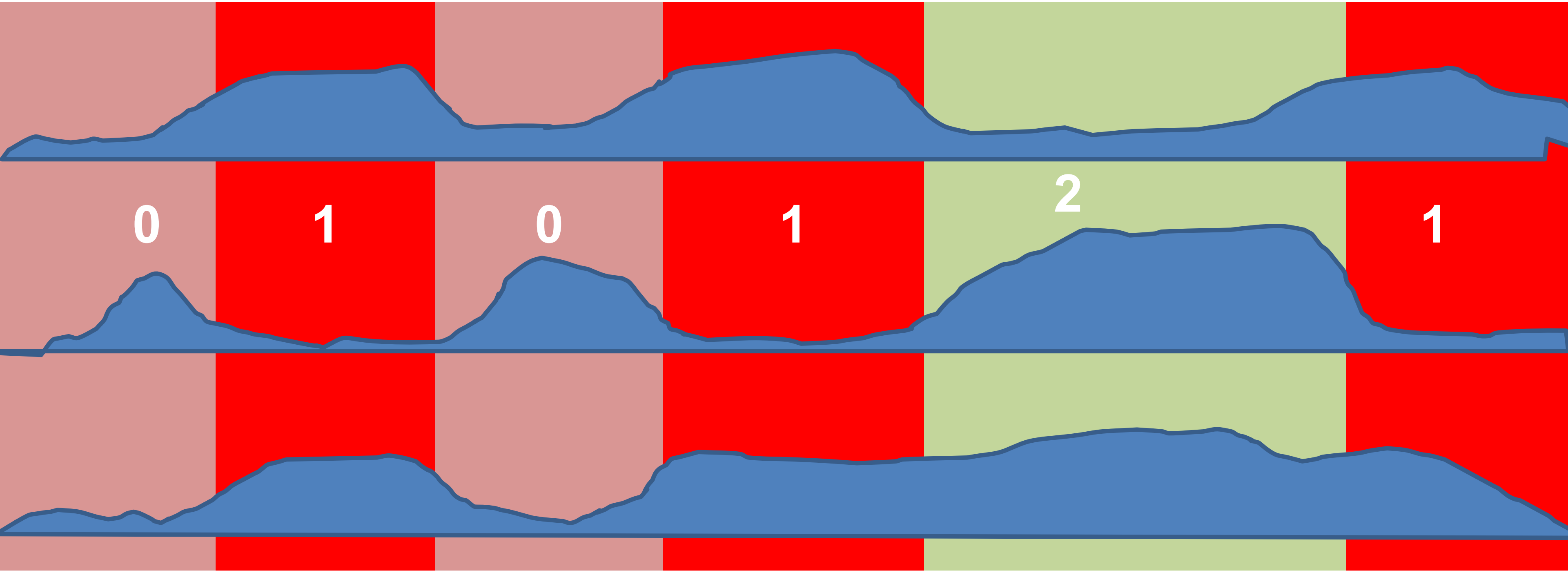
Maximize similarity in labels

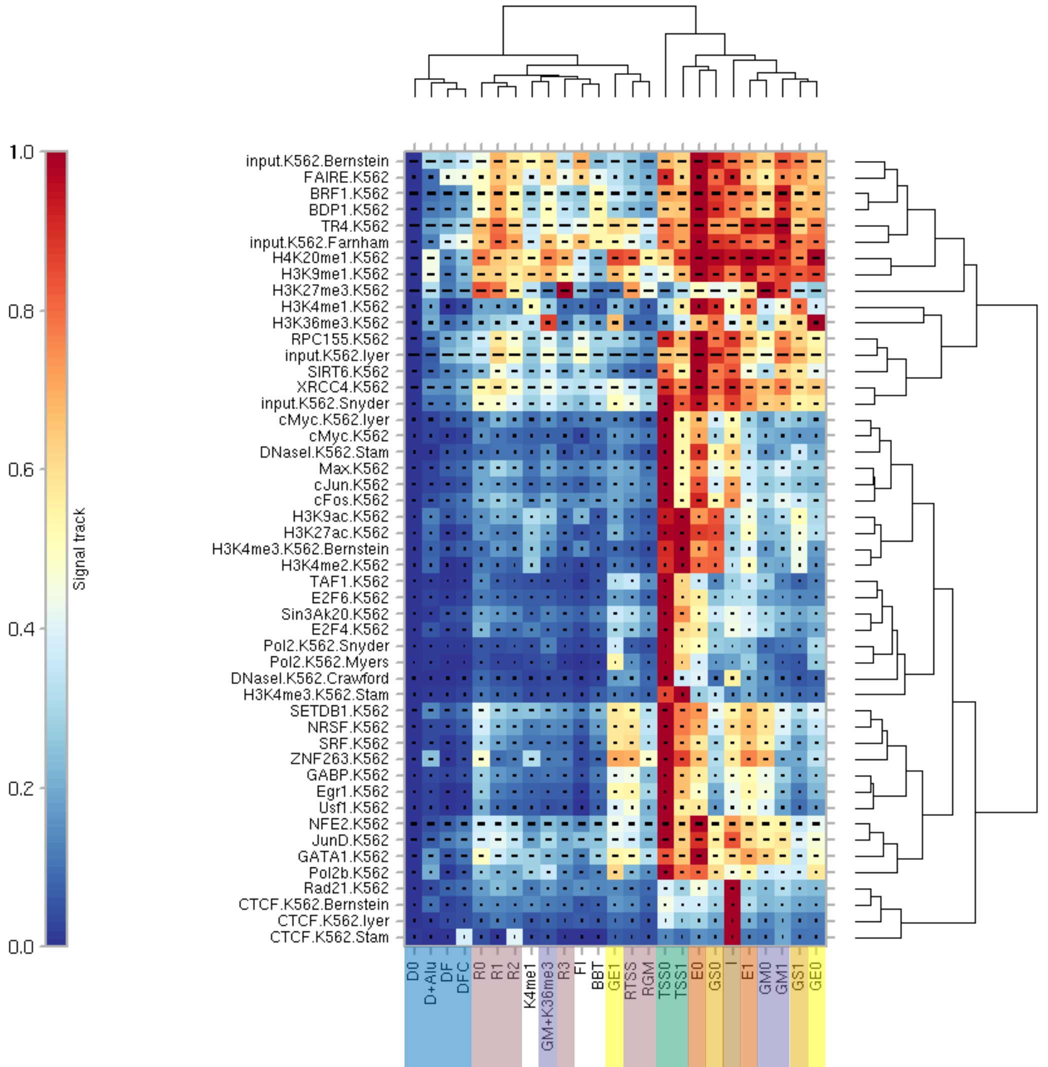


Maximize similarity in labels



Maximize similarity in labels





TSS transcription start site

GS gene start

GM gene middle

GE gene end

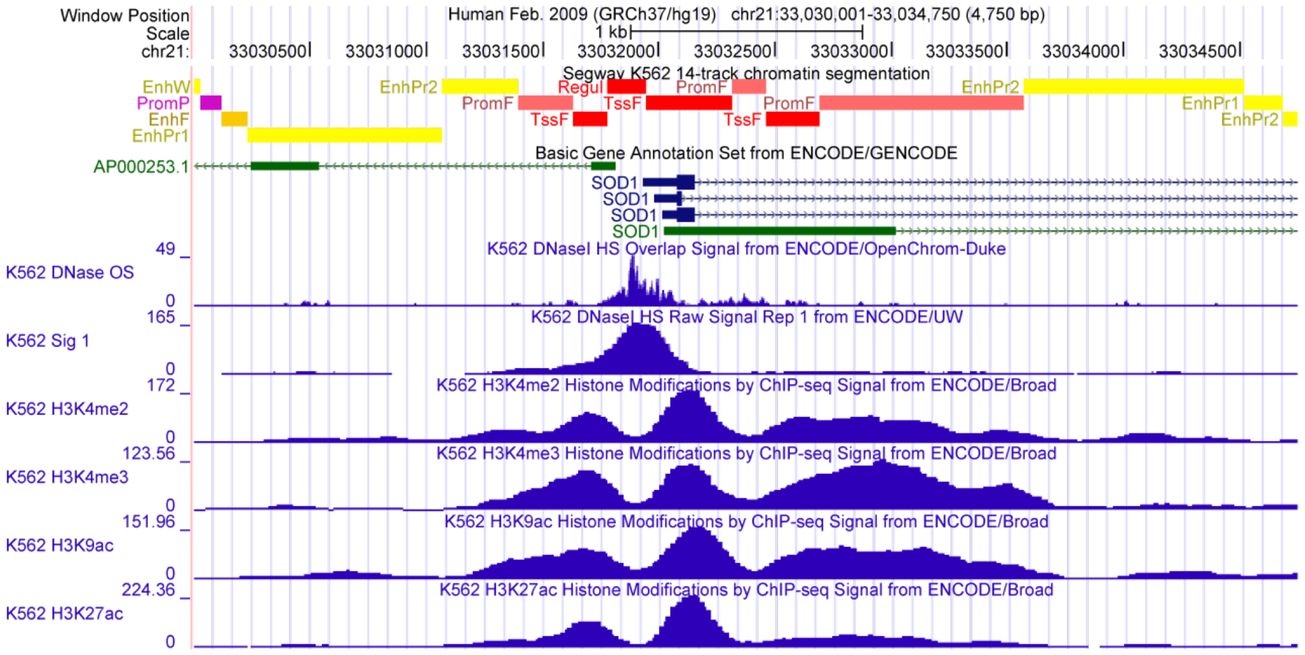
E enhancer

I distal CTCF

R repression

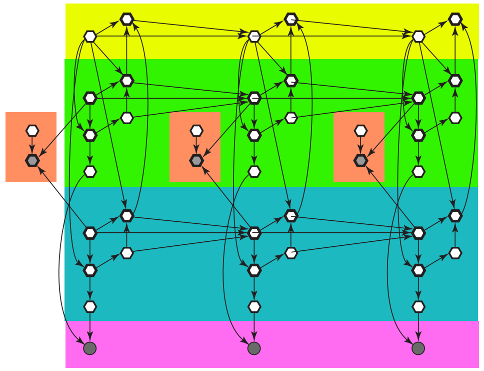
D dead

Transcription start site (TSS)


Window Position
Scale
chr21:
Human Feb. 2009 (GRCh37/hg19) chr21:33,030,001-33,034,750 (4,750 bp)
1 kb
33030500
33031000
33031500
33032000
33032500
33033000
33033500
33034000
33034500
Segway K562 14-track chromatin segmentation
EnhW
PromP
EnhF
EnhPr1
EnhPr2
Regul
PromF
TssF
PromF
TssF
PromF
TssF
EnhPr2
EnhPr1
EnhPr2
Basic Gene Annotation Set from ENCODE/GENCODE
AP000253.1
SOD1
SOD1
SOD1
SOD1
49
K562 DNase OS
0
K562 DNaseI HS Overlap Signal from ENCODE/OpenChrom-Duke
165
K562 Sig 1
0
K562 DNaseI HS Raw Signal Rep 1 from ENCODE/UW
172
K562 H3K4me2
0
K562 H3K4me2 Histone Modifications by ChIP-seq Signal from ENCODE/Broad
123.56
K562 H3K4me3
0
K562 H3K4me3 Histone Modifications by ChIP-seq Signal from ENCODE/Broad
151.96
K562 H3K9ac
0
K562 H3K9ac Histone Modifications by ChIP-seq Signal from ENCODE/Broad
224.36
K562 H3K27ac
0
K562 H3K27ac Histone Modifications by ChIP-seq Signal from ENCODE/Broad

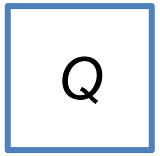
Hoffman MM *et al*. 2013. *Nucleic Acids Res* 41:827.

Graphical Models Toolkit (GMTK)



- General purpose toolkit for probabilistic inference in temporal signals (speech, language, activity recognition, genomics)
- Written in highly optimized scalable C++
- Supports many machine learning algorithms, inference procedures, and probability models.
- Can express arbitrary structured dynamic graphical models
- http://melodi.ee.washington.edu/gmtk
- `conda install -c bioconda gmtk`

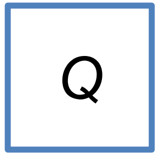
Jeff Bilmes



Graphical

○ hidden random variable

■ discrete



Graphical

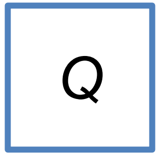
realization **parameters**

$$P(q \mid \theta) = P(Q = q)$$

Equation

☁ hidden random variable

■ discrete



Graphical

```
variable: state {
  type: discrete hidden cardinality 2;
  conditionalparents: nil using DenseCPT("start_state");
}
```

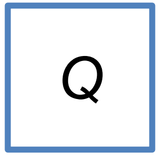
GMTKL

$$P(q \mid \Theta) \\ = P(Q = q)$$

Equation

○ hidden random variable

■ discrete



Graphical

○ hidden random variable

■ discrete

```
variable: state {
  type: discrete hidden cardinality 2;
  conditionalparents: nil using DenseCPT("start_state");
}
```

GMTKL

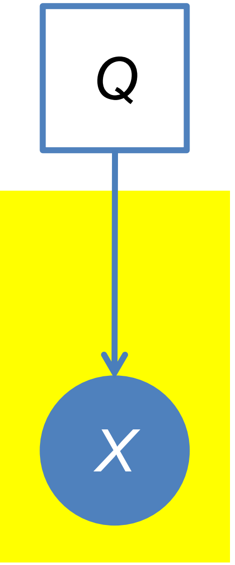
```
DENSE_CPT_IN_FILE inline % conditional probability tables
1                        % total number of CPTs = 1

0 start_state            % CPT #0, "start_state"
0                        % 0 parents
2                        % output cardinality 2
0.25                     % P(Q = 0) = 0.25
0.75                     % P(Q = 1) = 0.75
```

Parameters

$$P(q \mid \theta) = P(Q = q)$$

Equation



○ hidden random variable

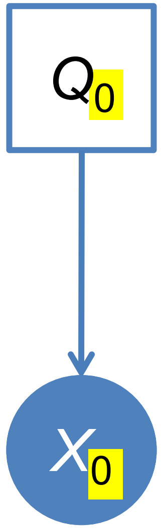
● observed random variable

■ discrete ● continuous

↘ conditional relationship

```
variable: state {
  type: discrete hidden cardinality 2;
  conditionalparents: nil using DenseCPT("start_state");
}
variable: obs {
  type: continuous observed 0:0;
  conditionalparents: state(0) using mapping("state_obs");
}
```

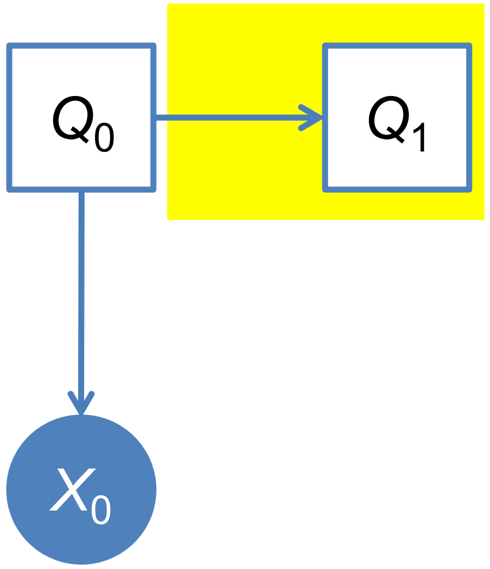
$$P(q,x \mid \theta) \\ = P(Q=q)P(X=x \mid Q=q)$$



○ hidden random variable
● observed random variable
■ discrete ● continuous
↘ conditional relationship

```
frame: 0 {
  variable: state {
    type: discrete hidden cardinality 2;
    conditionalparents: nil using DenseCPT("start_state");
  }
  variable: obs {
    type: continuous observed 0:0;
    conditionalparents: state(0) using mapping("state_obs");
  }
}
```

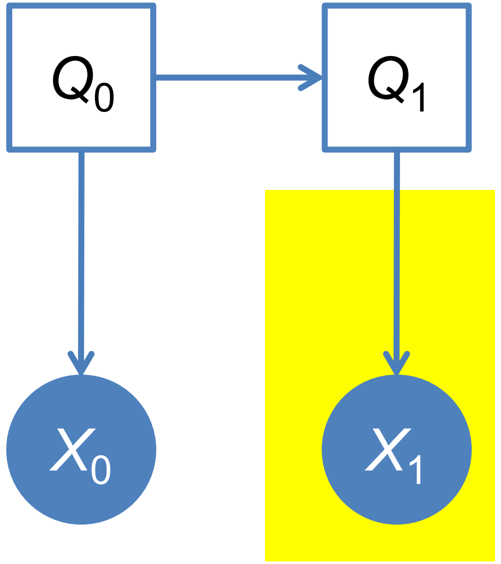
$$P(q_0, x_0 | \Theta)$$
$$= P(Q_0 = q_0) P(X_0 = x_0 | Q_0 = q_0)$$



```
frame: 0 {
  variable: state {
    type: discrete hidden cardinality 2;
    conditionalparents: nil using DenseCPT("start_state");
  }
  variable: obs {
    type: continuous observed 0:0;
    conditionalparents: state(0) using mapping("state_obs");
  }
}
frame: 1 {
  variable: state {
    type: discrete hidden cardinality 2;
    conditionalparents: state(-1) using DenseCPT("state_state");
  }
}
```

○ hidden random variable
● observed random variable
■ discrete ● continuous
↘ conditional relationship

$$P(q_{0:1}, x_0 \mid \Theta)$$
$$= P(Q_0 = q_0)P(X_0 = x_0 \mid Q_0 = q_0)P(Q_1 = q_1 \mid Q_0 = q_0)$$



```
frame: 0 {
  variable: state {
    type: discrete hidden cardinality 2;
    conditionalparents: nil using DenseCPT("start_state");
  }
  variable: obs {
    type: continuous observed 0:0;
    conditionalparents: state(0) using mapping("state_obs");
  }
}
frame: 1 {
  variable: state {
    type: discrete hidden cardinality 2;
    conditionalparents: state(-1) using DenseCPT("state_state");
  }
  variable: obs {
    type: continuous observed 0:0;
    conditionalparents: state(0) using mapping("state_obs");
  }
}
```

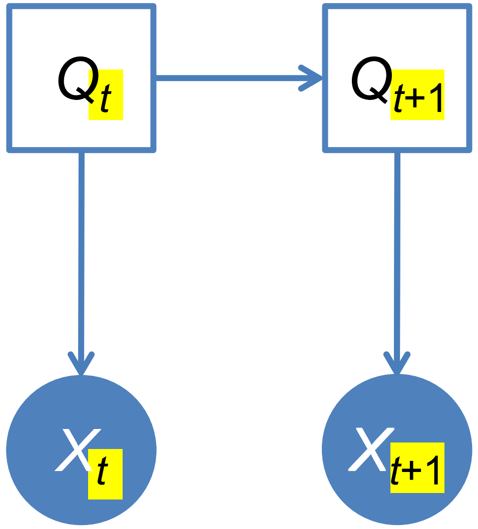
◌ hidden random variable

● observed random variable

■ discrete ● continuous

↘ conditional relationship

$$P(q_{0:1}, x_{0:1} \mid \Theta)$$
$$= P(Q_0 = q_0)P(X_0 = x_0 \mid Q_0 = q_0)P(Q_1 = q_1 \mid Q_0 = q_0)P(X_1 = x_1 \mid Q_1 = q_1)$$

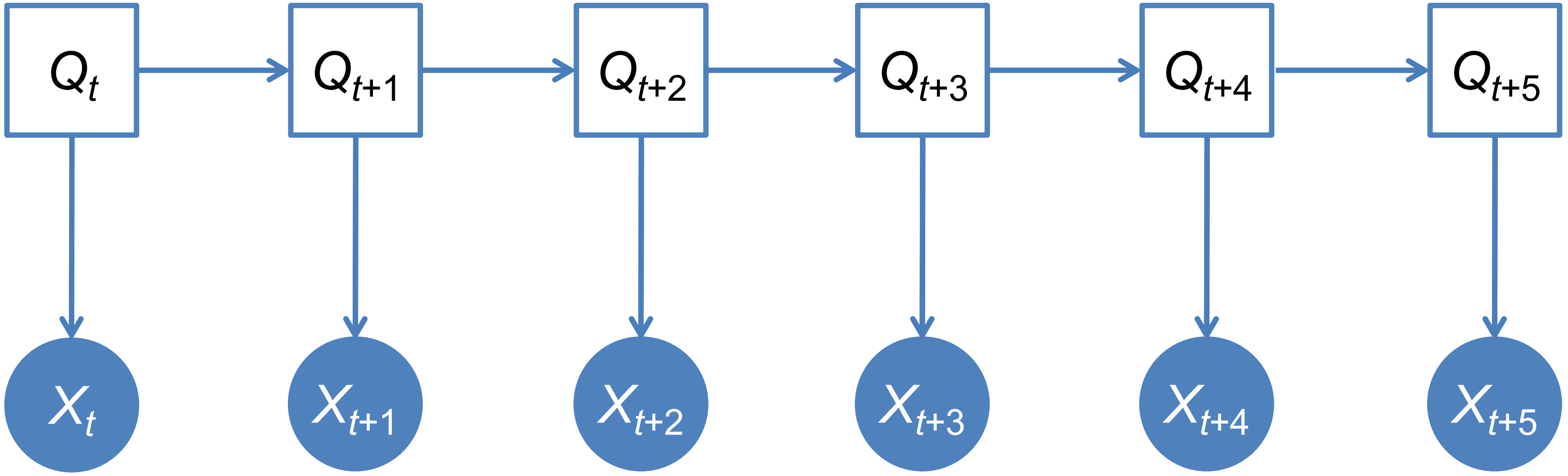


○ hidden random variable
● observed random variable
■ discrete ● continuous
↘ conditional relationship

```
frame: 0 {
  variable: state {
    type: discrete hidden cardinality 2;
    conditionalparents: nil using DenseCPT("start_state");
  }
  variable: obs {
    type: continuous observed 0:0;
    conditionalparents: state(0) using mapping("state_obs");
  }
}
frame: 1 {
  variable: state {
    type: discrete hidden cardinality 2;
    conditionalparents: state(-1) using DenseCPT("state_state");
  }
  variable: obs {
    type: continuous observed 0:0;
    conditionalparents: state(0) using mapping("state_obs");
  }
}
chunk 1:1
```

$$P(q_{0:T}, x_{0:T} \mid \Theta)$$
$$= P(Q_0 = q_0) P(X_0 = x_0 \mid Q_0 = q_0) \, \Pi_{t=1}^{T} P(Q_t = q_t \mid Q_{t-1} = q_{t-1}) P(X_t = x_t \mid Q_t = q_t)$$

Dynamic Bayesian network (hidden Markov model)



◌ hidden random variable

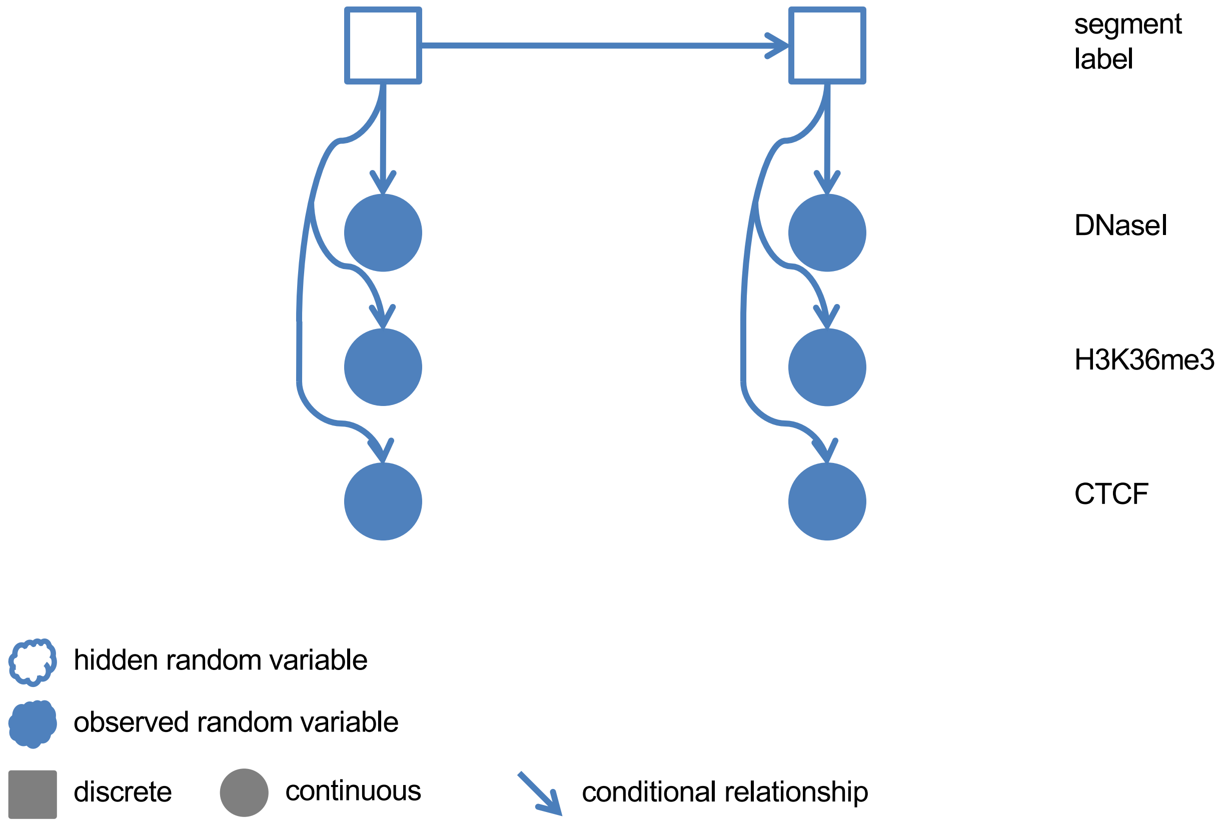
● observed random variable

■ discrete ● continuous

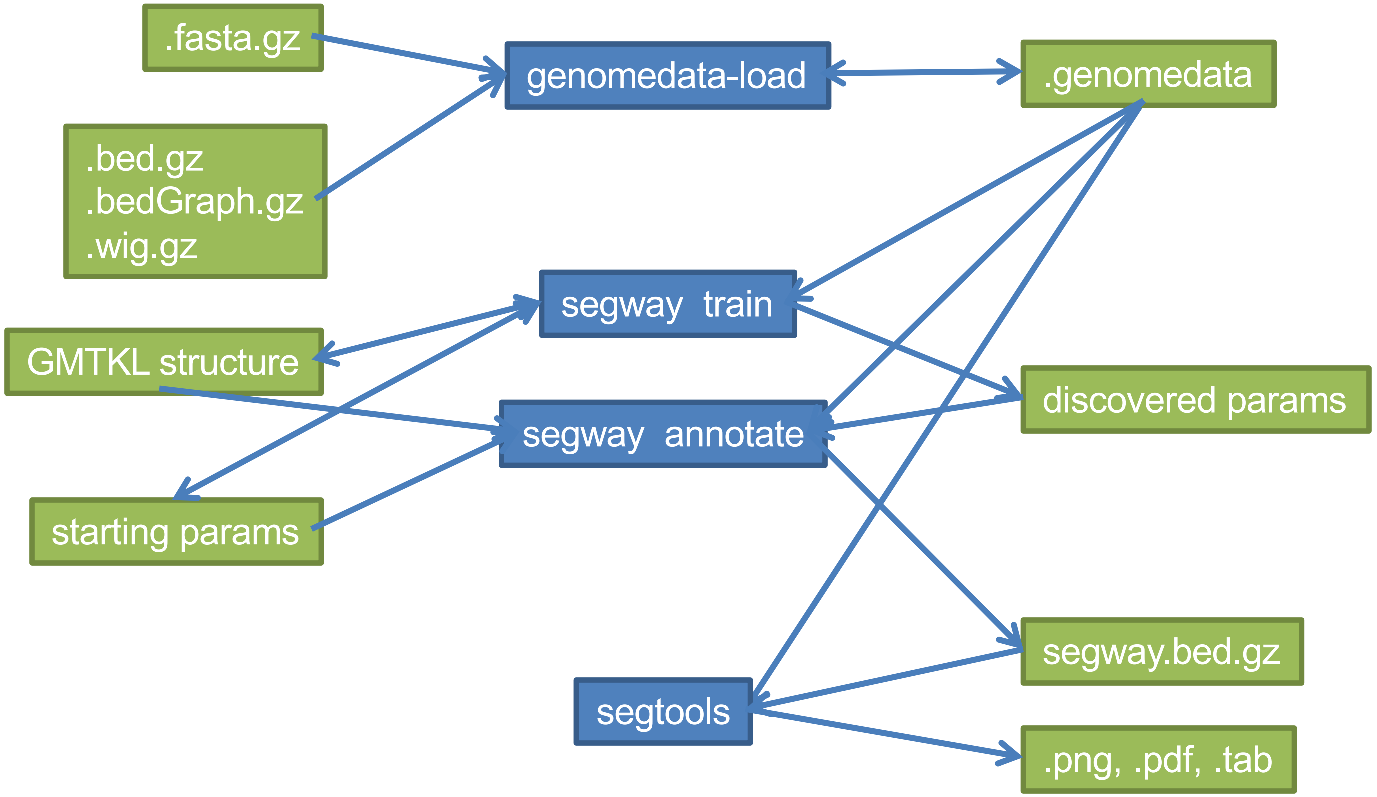
↘ conditional relationship

$$P(q_{0:T}, x_{0:T} \mid \Theta) = P(Q_0 = q_0) P(X_0 = x_0 \mid Q_0 = q_0) \prod_{t=1}^{T} P(Q_t = q_t \mid Q_{t-1} = q_{t-1}) P(X_t = x_t \mid Q_t = q_t)$$

Dynamic Bayesian Network for segmentation



Segway work flow



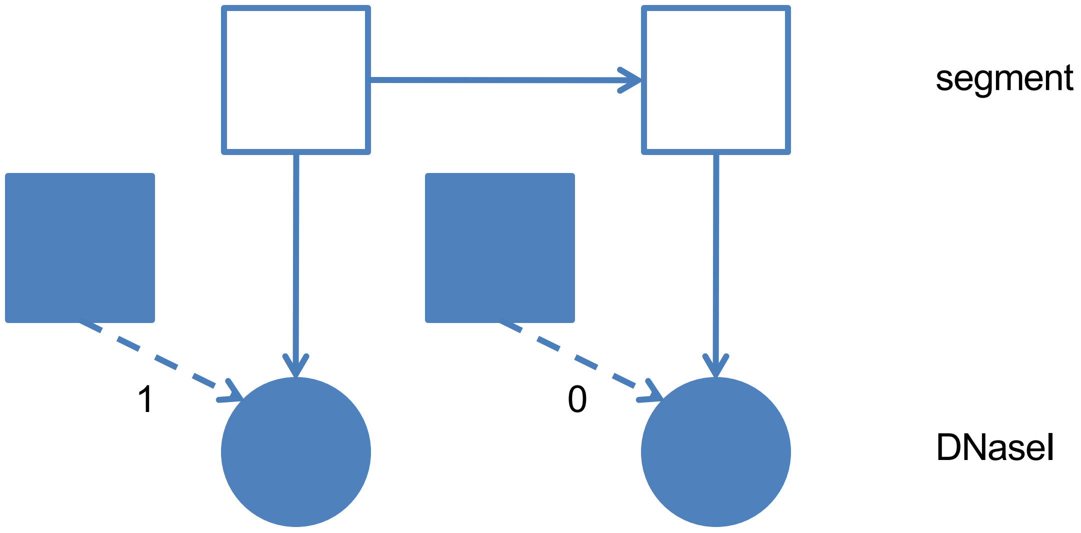
What does Segway do, really?

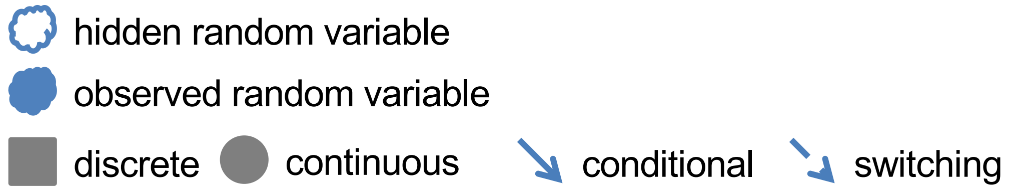
1. Creates GMTKL structure and parameter files for semi-automated genome annotation
2. Converts genomic data to a GMTK binary observation format
3. Runs GMTK to perform EM training, Viterbi decoding, posterior inference
4. Manages job execution via cluster (SGE, LSF, PBS, Slurm), or local multiprocessing
5. Converts GMTK output to genomics formats (BED, wiggle)

Segway modular API (pull request #91)

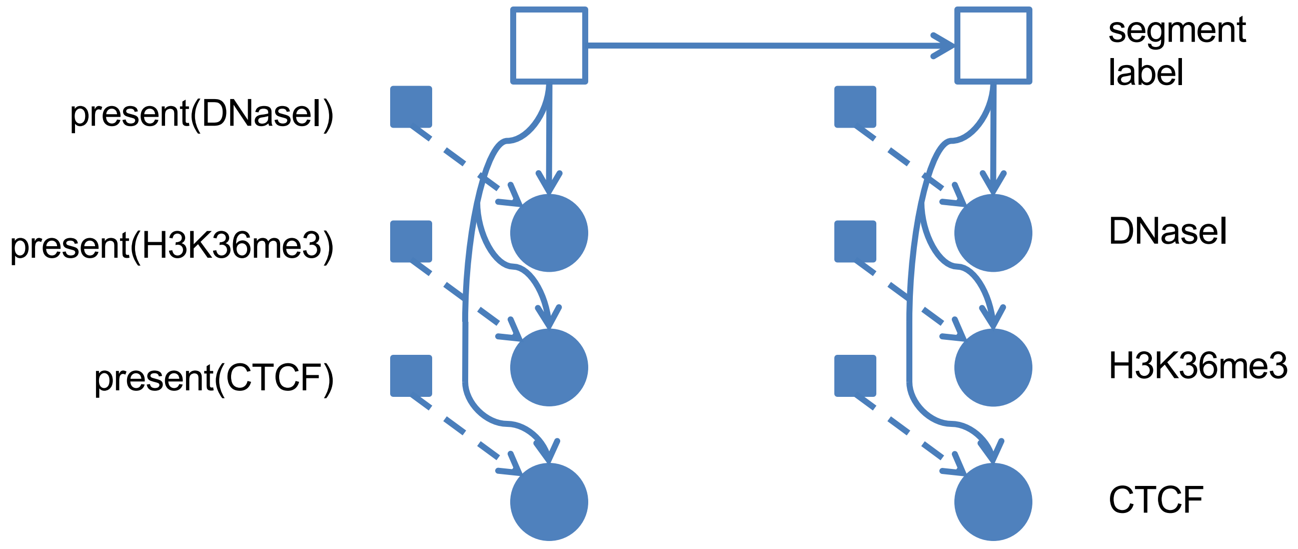
- segway train
  - segway train-init
  - segway train-run
  - segway train-run-round
  - segway train-finish
- segway annotate
  - segway annotate-init
  - segway annotate-run
  - segway annotate-finish
- segway posterior
  - segway posterior-init
  - segway posterior-run
  - segway posterior-finish

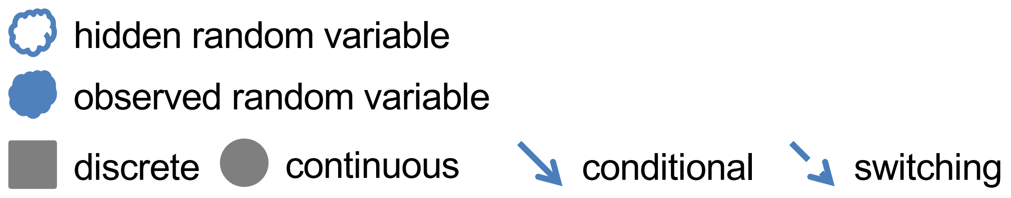
Handling missing data



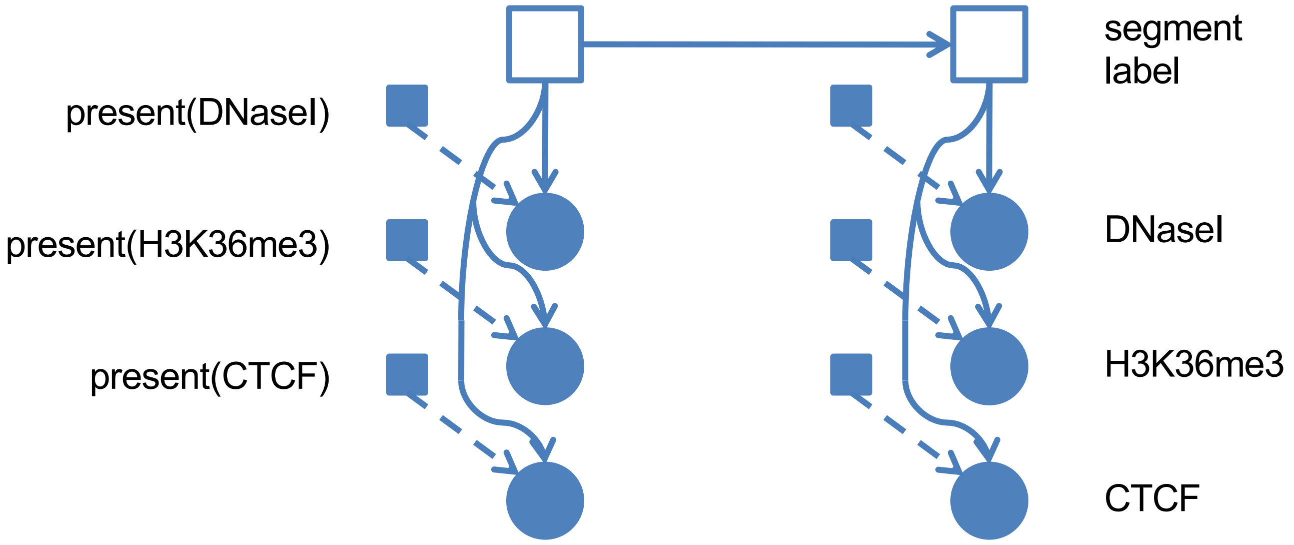


Handling missing data

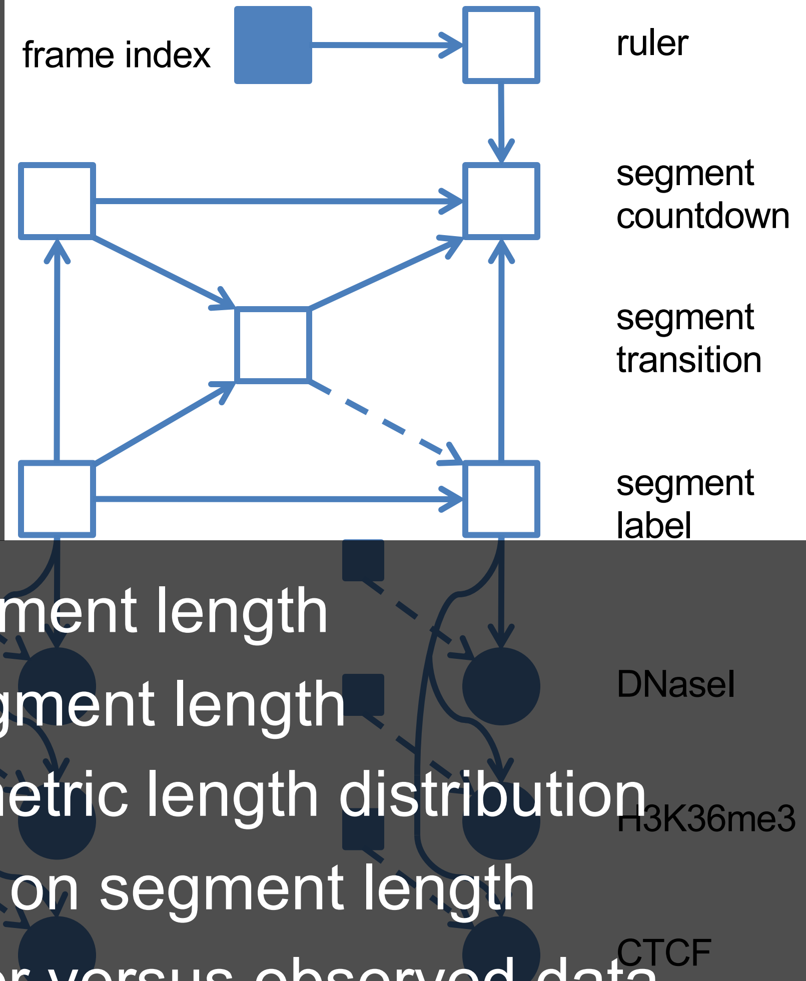




Length distribution

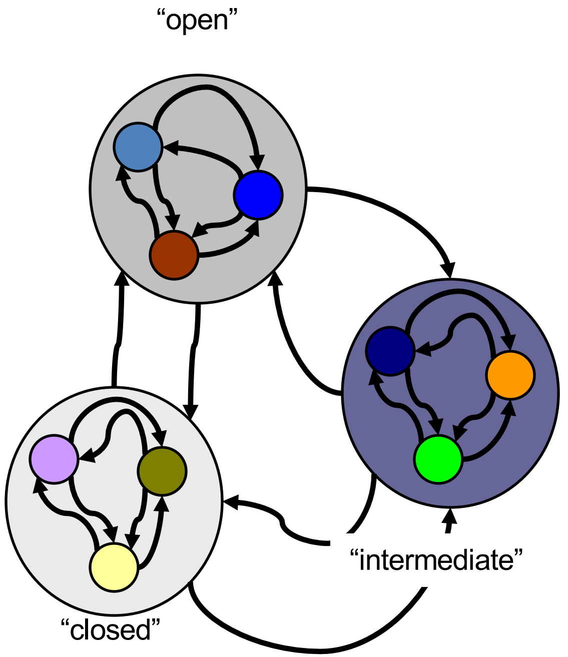


Length distribution

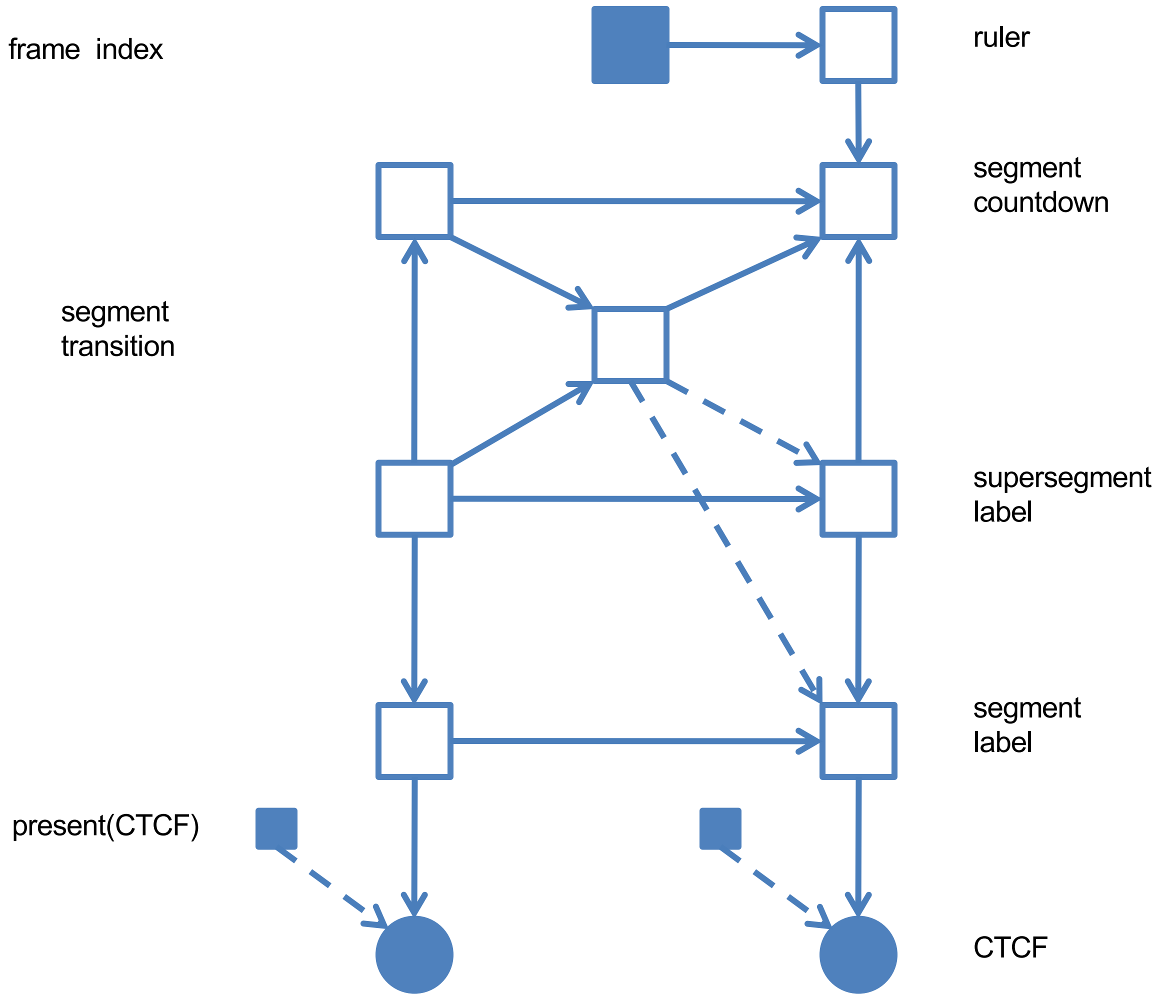


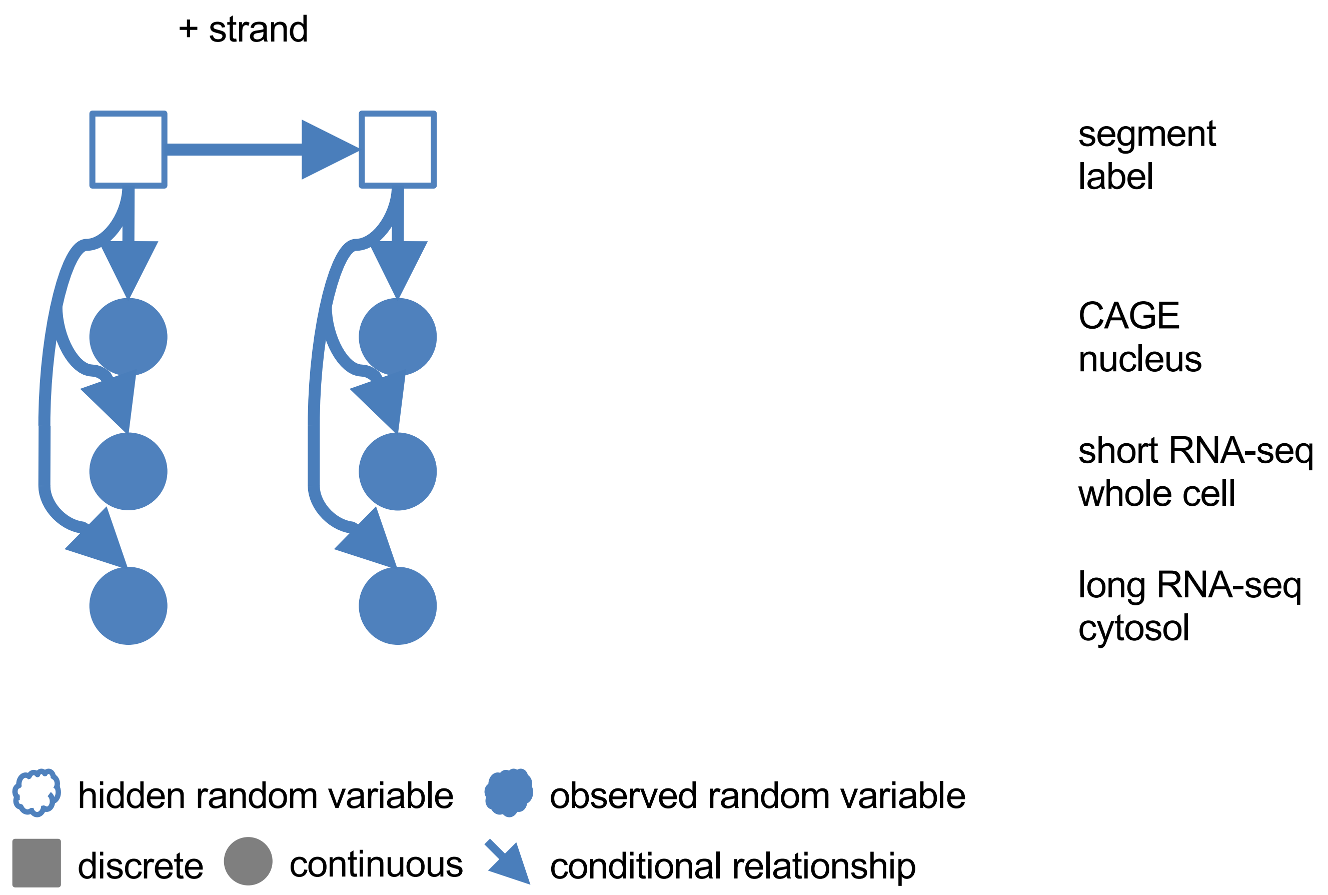
- Minimum segment length
- Maximum segment length
- Trained geometric length distribution
- Dirichlet prior on segment length
- Weight of prior versus observed data

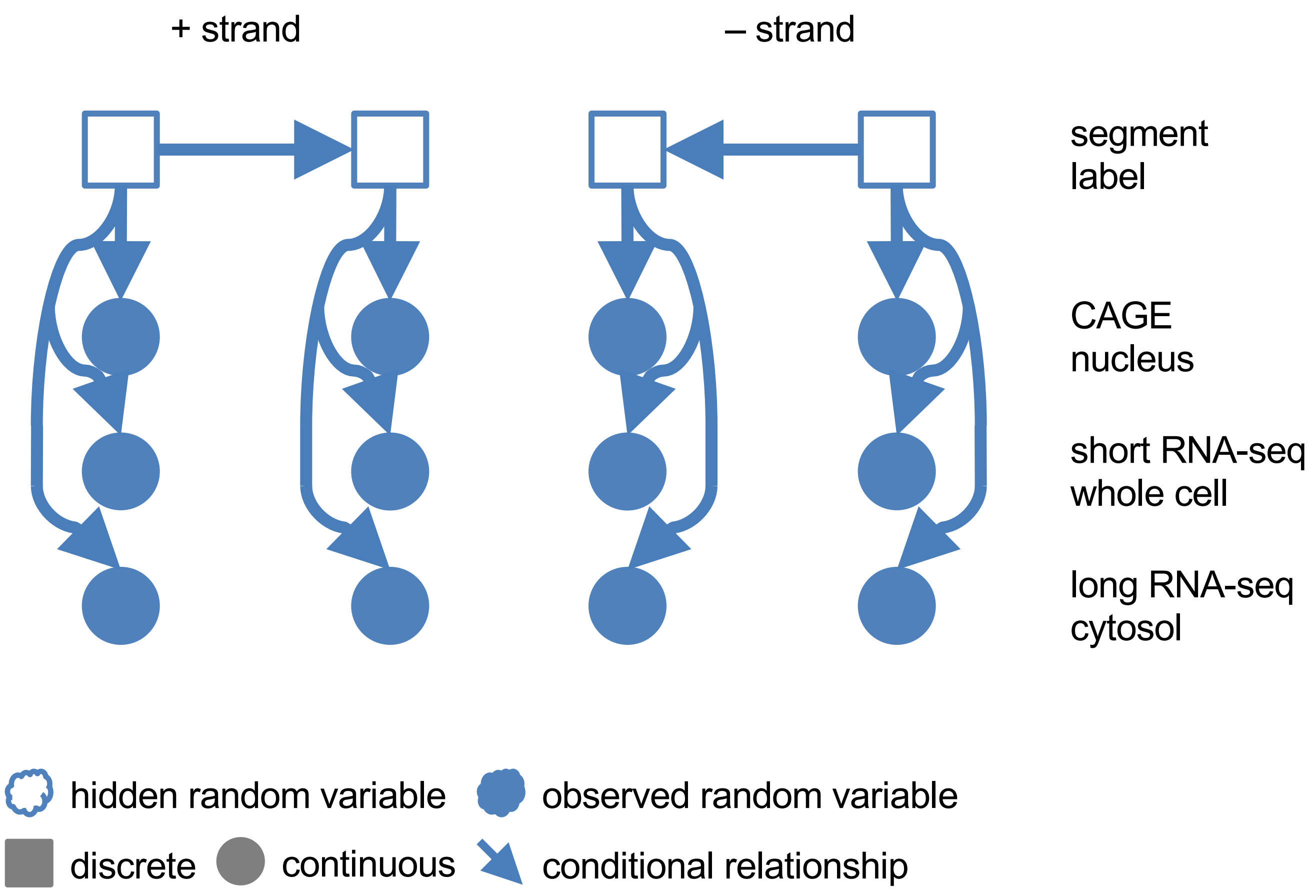
Hierarchical models



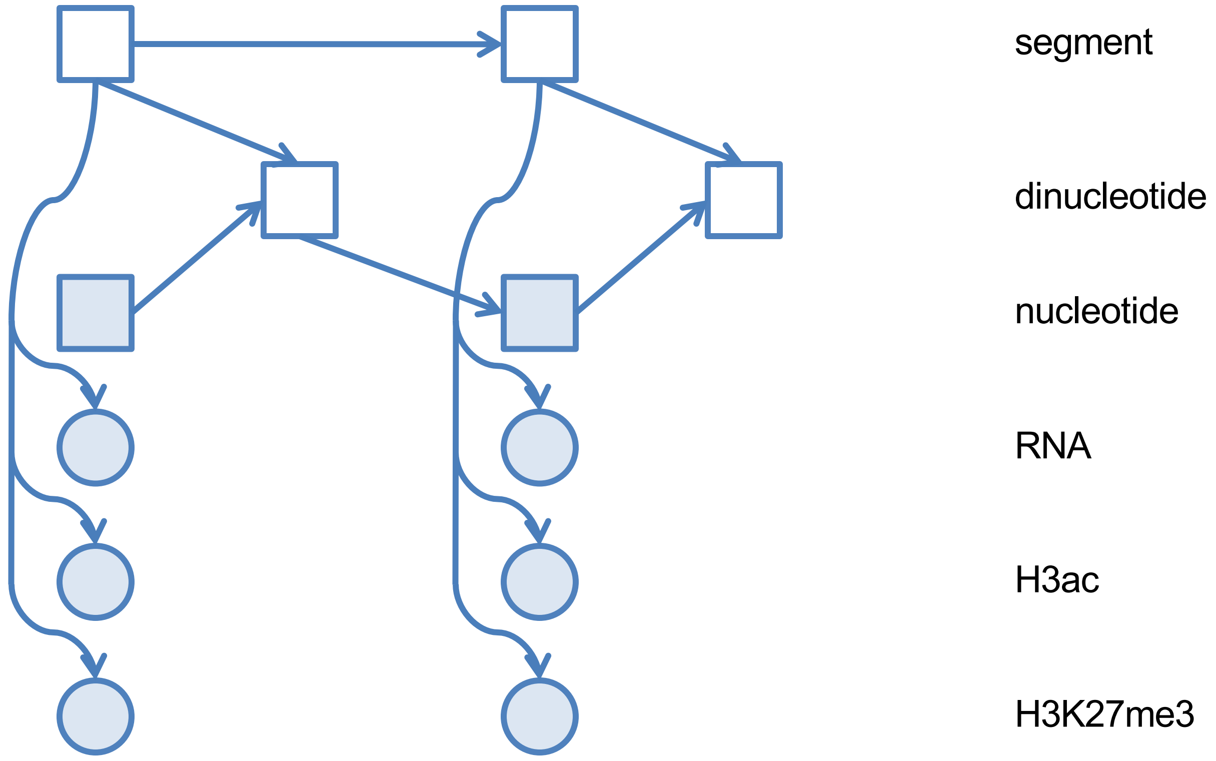
- Two-level model topology
- Mixture of large-scale and punctate labels


frame index
ruler
segment countdown
segment transition
supersegment label
segment label
present(CTCF)
CTCF

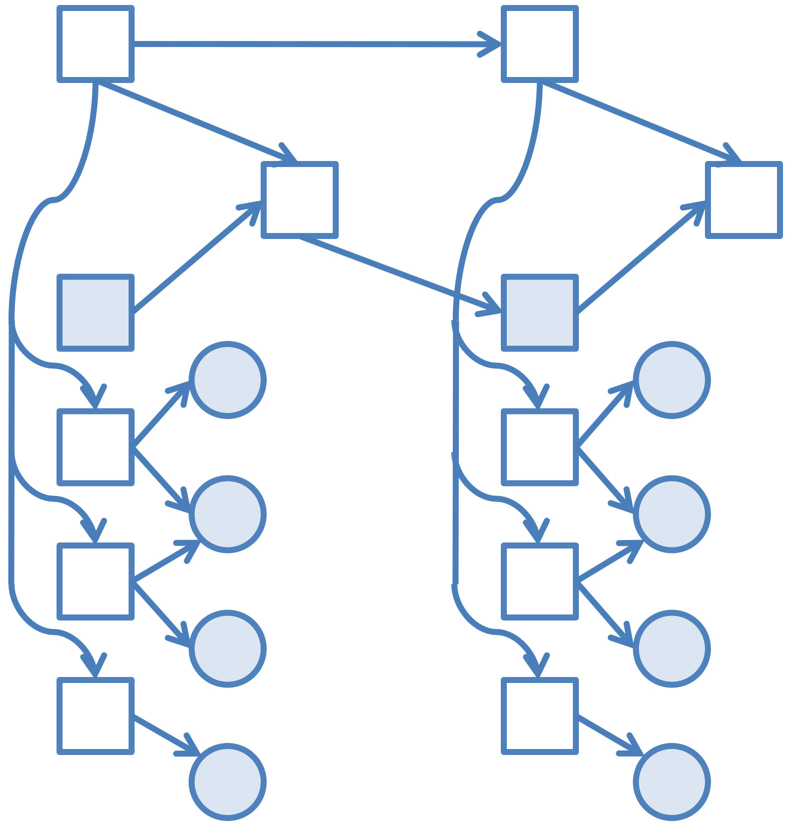

+ strand
segment label
CAGE nucleus
short RNA-seq whole cell
long RNA-seq cytosol
hidden random variable
observed random variable
discrete
continuous
conditional relationship


+ strand
– strand
segment label
CAGE nucleus
short RNA-seq whole cell
long RNA-seq cytosol
hidden random variable
observed random variable
discrete
continuous
conditional relationship

Adding genome sequence



Adding data relationships



segment

dinucleotide

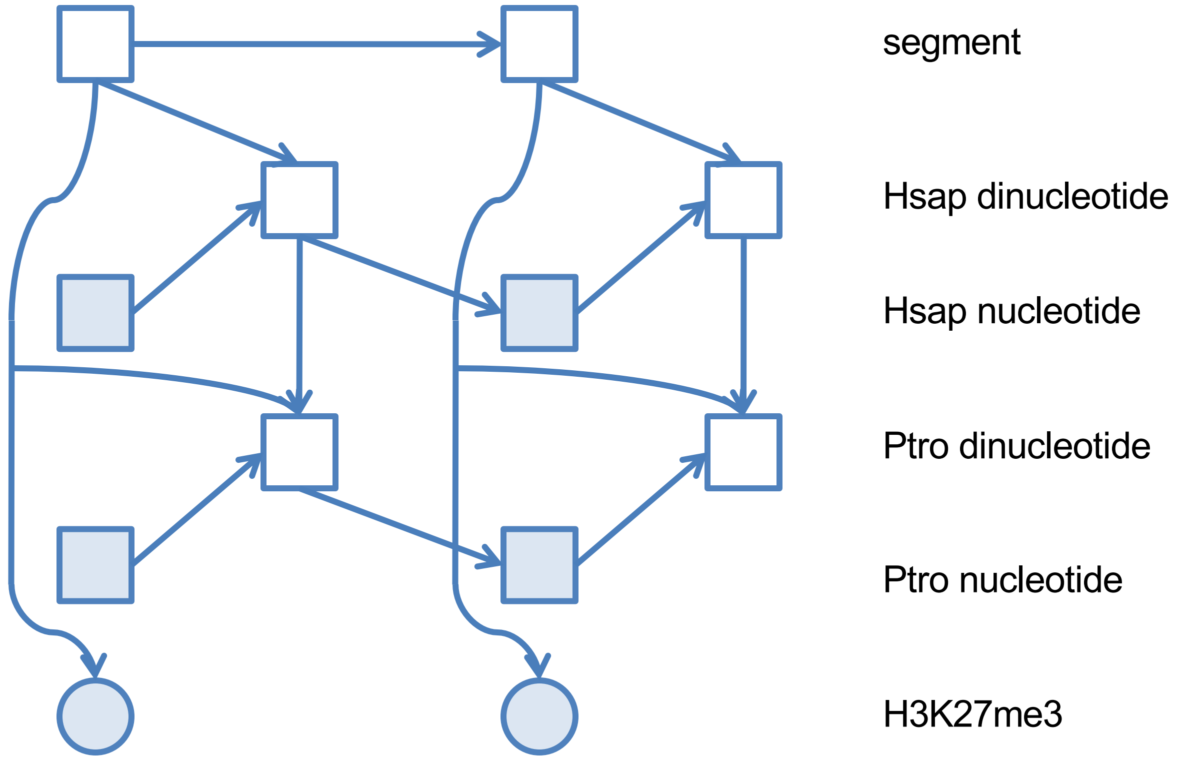
nucleotide

HeLa

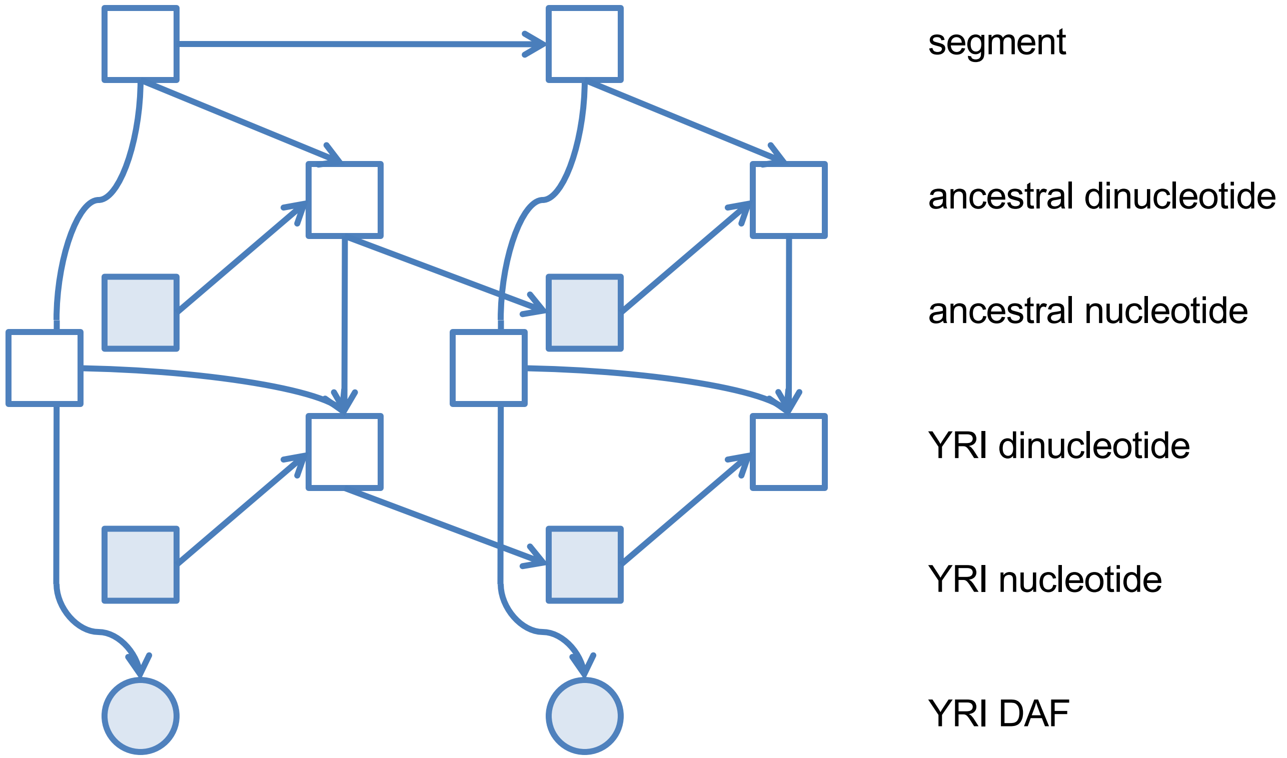
histone

GM12878

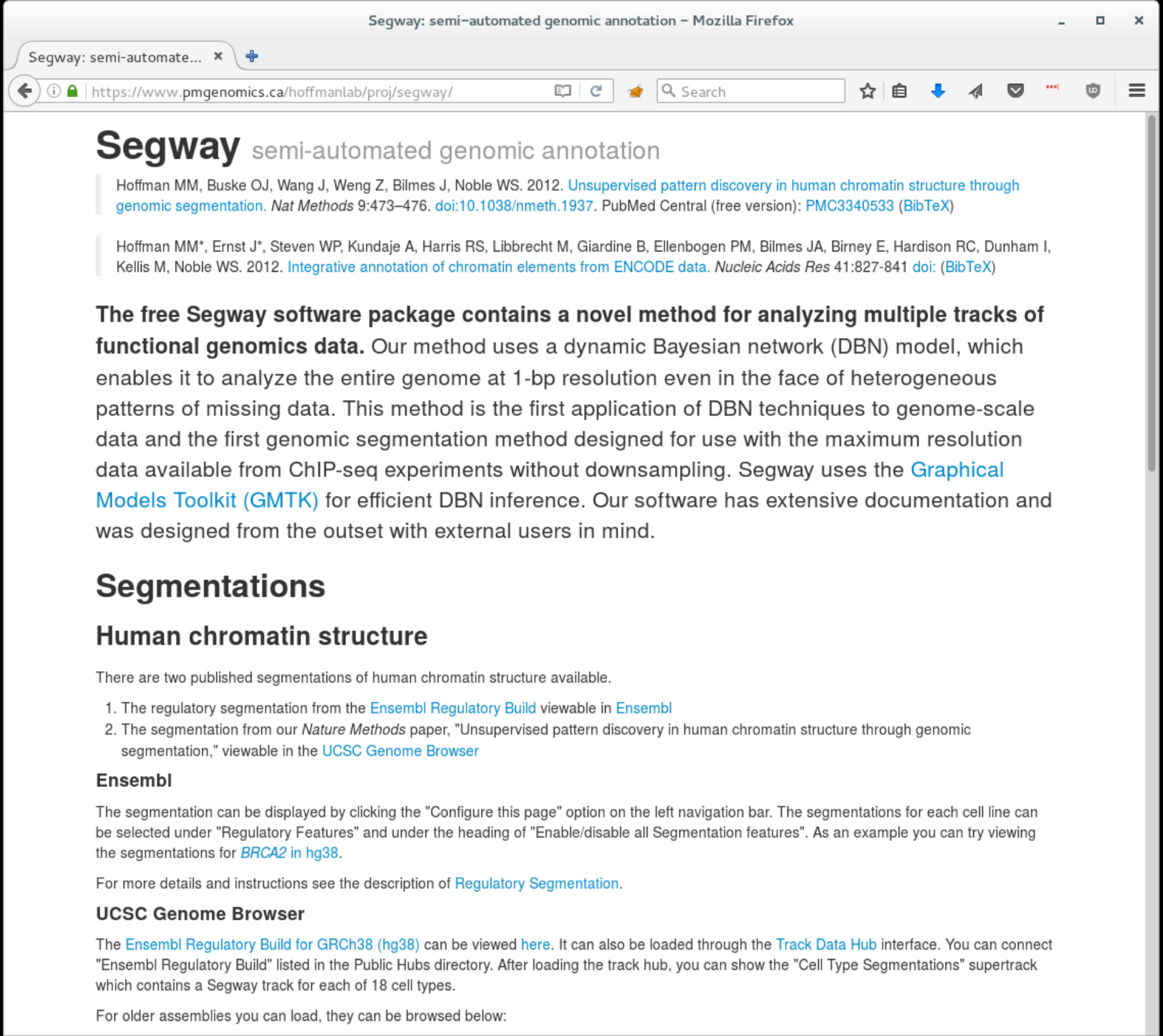
Adding evolution



Adding variation



https://segway.hoffmanlab.org/



Segway: semi-automated genomic annotation – Mozilla Firefox

https://www.pmgenomics.ca/hoffmanlab/proj/segway/

Segway semi-automated genomic annotation

Hoffman MM, Buske OJ, Wang J, Weng Z, Bilmes J, Noble WS. 2012. Unsupervised pattern discovery in human chromatin structure through genomic segmentation. *Nat Methods* 9:473–476. doi:10.1038/nmeth.1937. PubMed Central (free version): PMC3340533 (BibTeX)

Hoffman MM*, Ernst J*, Steven WP, Kundaje A, Harris RS, Libbrecht M, Giardine B, Ellenbogen PM, Bilmes JA, Birney E, Hardison RC, Dunham I, Kellis M, Noble WS. 2012. Integrative annotation of chromatin elements from ENCODE data. *Nucleic Acids Res* 41:827-841 doi: (BibTeX)

The free Segway software package contains a novel method for analyzing multiple tracks of functional genomics data. Our method uses a dynamic Bayesian network (DBN) model, which enables it to analyze the entire genome at 1-bp resolution even in the face of heterogeneous patterns of missing data. This method is the first application of DBN techniques to genome-scale data and the first genomic segmentation method designed for use with the maximum resolution data available from ChIP-seq experiments without downsampling. Segway uses the Graphical Models Toolkit (GMTK) for efficient DBN inference. Our software has extensive documentation and was designed from the outset with external users in mind.

Segmentations

Human chromatin structure

There are two published segmentations of human chromatin structure available.

1. The regulatory segmentation from the Ensembl Regulatory Build viewable in Ensembl
2. The segmentation from our *Nature Methods* paper, "Unsupervised pattern discovery in human chromatin structure through genomic segmentation," viewable in the UCSC Genome Browser

Ensembl

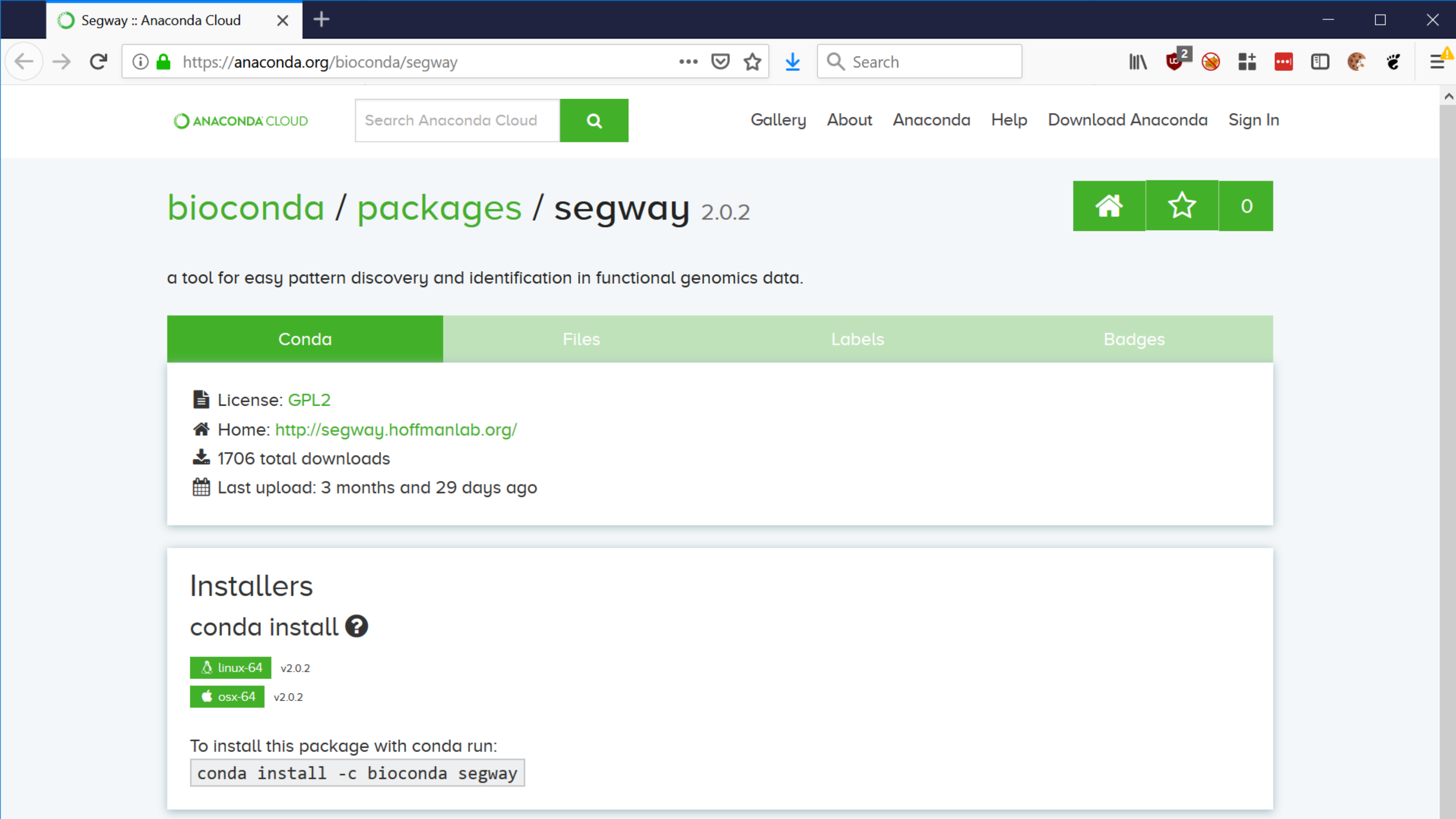
The segmentation can be displayed by clicking the "Configure this page" option on the left navigation bar. The segmentations for each cell line can be selected under "Regulatory Features" and under the heading of "Enable/disable all Segmentation features". As an example you can try viewing the segmentations for *BRCA2* in hg38.

For more details and instructions see the description of Regulatory Segmentation.

UCSC Genome Browser

The Ensembl Regulatory Build for GRCh38 (hg38) can be viewed here. It can also be loaded through the Track Data Hub interface. You can connect "Ensembl Regulatory Build" listed in the Public Hubs directory. After loading the track hub, you can show the "Cell Type Segmentations" supertrack which contains a Segway track for each of 18 cell types.

For older assemblies you can load, they can be browsed below:



Segway :: Anaconda Cloud

https://anaconda.org/bioconda/segway

bioconda / packages / segway 2.0.2

a tool for easy pattern discovery and identification in functional genomics data.

License: GPL2

Home: http://segway.hoffmanlab.org/

1706 total downloads

Last upload: 3 months and 29 days ago

Installers

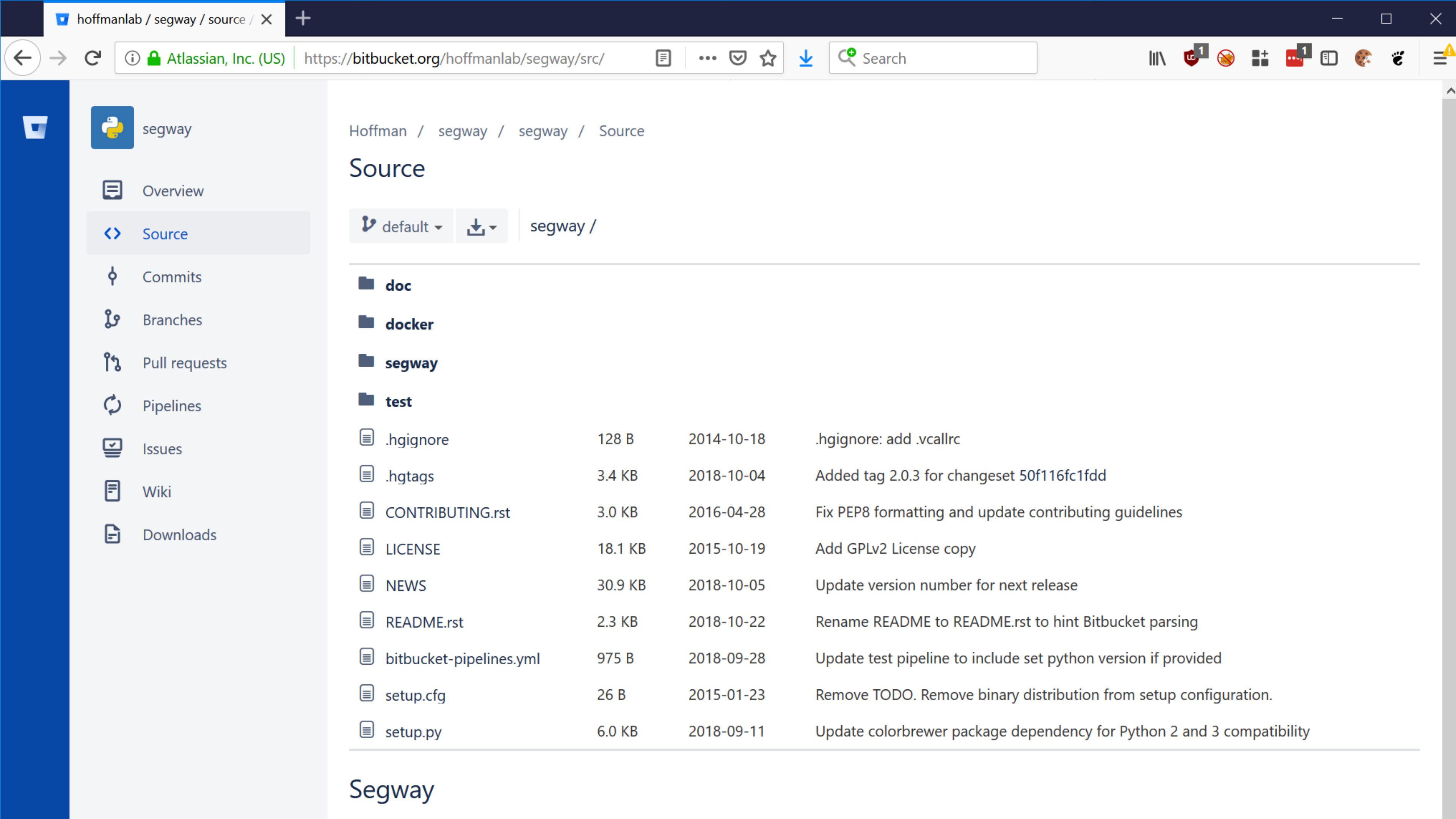
conda install

linux-64 v2.0.2

osx-64 v2.0.2

To install this package with conda run:

```
conda install -c bioconda segway
```



hoffmanlab / segway / source /

Atlassian, Inc. (US) https://bitbucket.org/hoffmanlab/segway/src/

Search

segway

Overview
Source
Commits
Branches
Pull requests
Pipelines
Issues
Wiki
Downloads

Hoffman / segway / segway / Source

Source

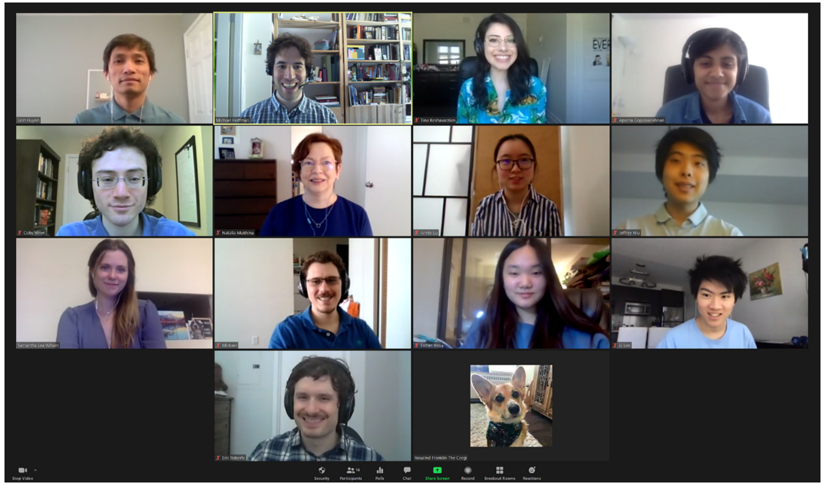
default segway /

doc			
docker			
segway			
test			
.hgignore	128 B	2014-10-18	.hgignore: add .vcallrc
.hgtags	3.4 KB	2018-10-04	Added tag 2.0.3 for changeset 50f116fc1fdd
CONTRIBUTING.rst	3.0 KB	2016-04-28	Fix PEP8 formatting and update contributing guidelines
LICENSE	18.1 KB	2015-10-19	Add GPLv2 License copy
NEWS	30.9 KB	2018-10-05	Update version number for next release
README.rst	2.3 KB	2018-10-22	Rename README to README.rst to hint Bitbucket parsing
bitbucket-pipelines.yml	975 B	2018-09-28	Update test pipeline to include set python version if provided
setup.cfg	26 B	2015-01-23	Remove TODO. Remove binary distribution from setup configuration.
setup.py	6.0 KB	2018-09-11	Update colorbrewer package dependency for Python 2 and 3 compatibility

Segway

Acknowledgments

The Hoffman Lab



Samantha Wilson
Eric Roberts
Mickaël Mendez
Annie Lu
Leo Li

Linh Huynh
Coby Viner
Jeffrey Niu
Aparna Gopalakrishnan
Esther Yu

Natalia Mukhina

Jeff Bilmes
William Noble
Max Libbrecht

Funding:
Princess Margaret Cancer Foundation
Canadian Institutes of Health Research
Canadian Cancer Society
Natural Sciences and Engineering Research Council
Ontario Institute for Cancer Research
Ontario Ministry of Research, Innovation and Science
Medicine by Design
McLaughlin Centre

Slides: https://tinyurl.com/segwaygmtk

Postdoctoral, MSc, PhD positions available in my research lab at the

Princess Margaret Cancer Centre

Dept of Medical Biophysics
Dept of Computer Science
University of Toronto

Please approach me for details.

Michael Hoffman
https://hoffmanlab.org/

michael.hoffman@utoronto.ca

@michaelhoffman

Software availability

- Segway
  - data tracks → segmentation
  - Hoffman MM *et al.* 2012. *Nat Methods* 9:473.
  - https://segway.hoffmanlab.org/
- Segtools
  - segmentation → plots and summary statistics
  - Buske OJ *et al.* 2011. *BMC Bioinformatics* 12:415
  - https://segtools.hoffmanlab.org/
- Genomedata
  - efficient access to numeric data anchored to genome
  - Hoffman MM *et al*. 2010. *Bioinformatics* 26:1458.
  - https://genomedata.hoffmanlab.org/
- Umap and Bismap
  - Identify unmappable regions and corresponding missing data
  - Karimzadeh *et al.* 2018. *Nucleic Acids Res,* in press. https://doi.org/10.1093/nar/gky677
  - https://bismap.hoffmanlab.org/

GMTK Core Features

1. Textual Graph Language
2. Switching Parent Functionality
3. Forwards and Backwards time links
4. Multi-rate models with extended templates.
5. Linear Dependencies on observations
6. Non-linear (Deep Many-Layered) Unary Potentials
7. Arbitrary low-level parameter sharing (EM/GEM training)
8. Gaussian Vanishing/Splitting algorithm.
9. Decision-Tree and C++ Based implementations of dependencies (deterministic, sparse, formula leaf nodes)
10. Full inference, single pass decoding, online inference
11. Linear and Island Algorithm (O(logT)) Exact Inference

Graphical Models Toolkit (GMTK)

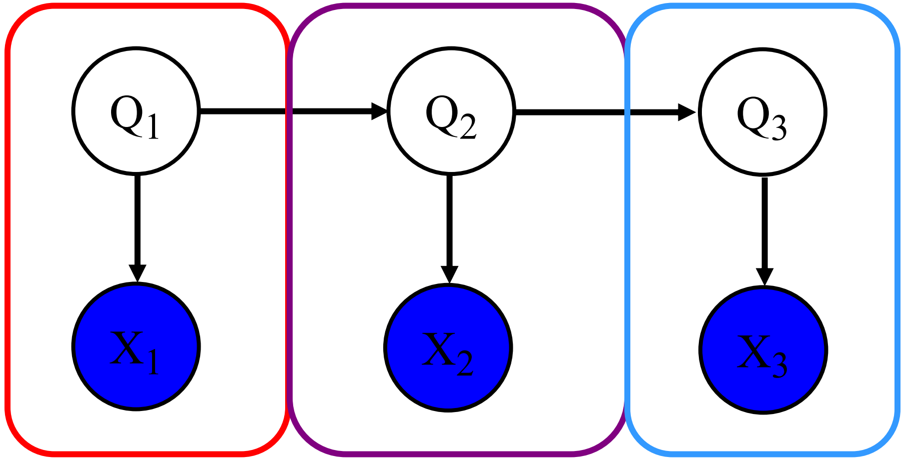
1. Textual Graph Language
2. Switching Parent Functionality
3. Forwards and Backwards time directed edges
4. Edges across multiple frames and chunks
5. Multi-frame and multi-rate models with extended DBN templates.
6. Disconnected Networks
7. Static Networks (in addition to dynamic)
8. Native ARPA Language Models
9. Full HTK lattice support with additional support for soft bounded node range determination.
10. Native Factored Language Models (compatible with all SRILM-FLM generalized backoff options)
11. Linear Dependencies on observations
12. Feature-file processing (up/down sampling, merging, subset selection, etc.)
13. Arbitrary low-level parameter sharing (EM/GEM training)
14. Gaussian Vanishing/Splitting algorithm.
15. Decision-Tree-Based implementations of dependencies
16. Rich and arbitrary integer formula DT leaf nodes.
17. Full inference, single pass decoding
18. Sampling Methods
19. Linear and Island Algorithm (O(logT)) Exact Inference
20. Native Virtual Evidence (Hybrid {SVM,ANN}/DBN)
21. Virtual Evidence Separators (speeds VE)
22. Switching Weights (penalties, scales, and shifts)
23. Separate GMTK Triangulation Engine (*many* heuristics supported)
24. Anytime triangulation search
25. Graph visualization tool (gmtkViz)
26. Tieing program (gmtkTie), Simon King
27. Fancy debugging/trace output to help debugging your graph.
28. Much much faster! (up to 800 times!!)

… under the hood:

- Algorithmic features:
    1. triangulation of DBN partitions rather than entire unrolled graph
    2. Optimal boundary/separator determination
    3. Max-flow (and submodular) based boundary determination
    4. Big cliques can be fast cliques (when determinism abounds). Non-elimination based triangulation.
    5. Separator (rather than clique) driven inference to quickly and dynamically remove zeros. Yields exact and approximate inference (via dynamic pruning)
    6. Virtual Evidence Separators (utilize constraints)
- System level features:
    1. Topological and ascending cardinality ordered variable iteration within cliques
    2. Almost optimal packed representations
    3. DBN clique-specific value sharing
    4. Anytime triangulation search
    5. No C++ STL for inner loops (custom data structures, PHiPAC-style optimizations, and memory management)

GMTK Structure file for HMM

- Structure file defines a prologue P, chunk C, and epilog E. E.g., for the basic HMM:



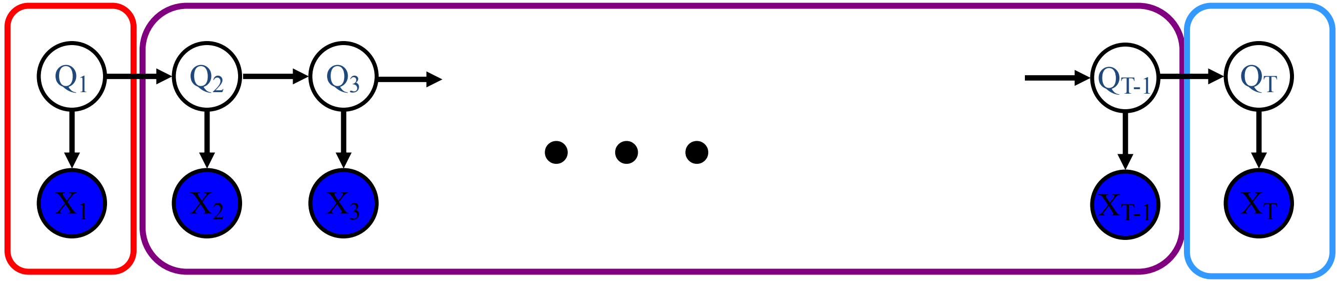
Prologue, first Group of frames

Chunk, Repeated Until T frames

Epilogue, last Group of frames

GMTK Unrolled structure

- Chunk is unrolled T-size(prologue)-size(epilog) times (if 1 frame in chunk)



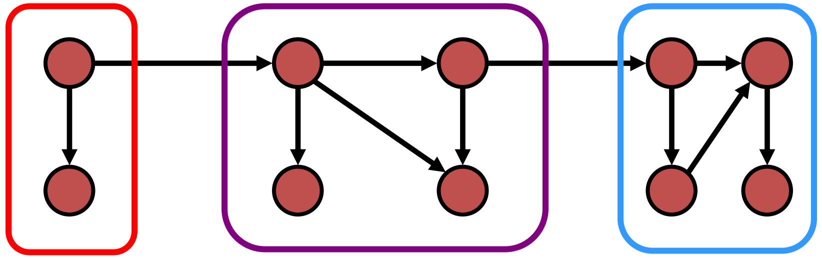
Prologue, first group of frames

Chunk, Repeated until T frames is obtained.

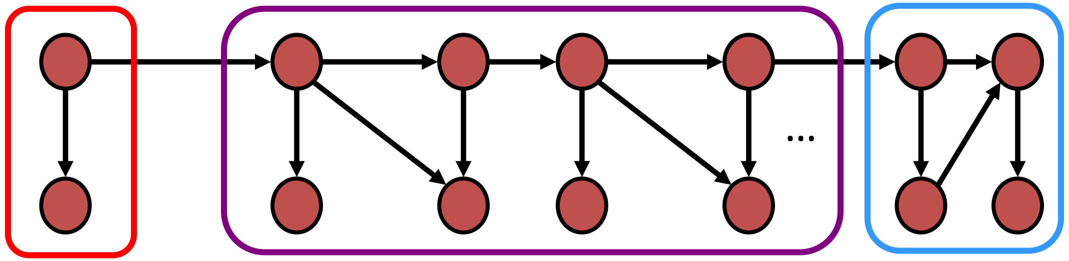
Epilog, last group of frames

Multiframe Repeating Chunks

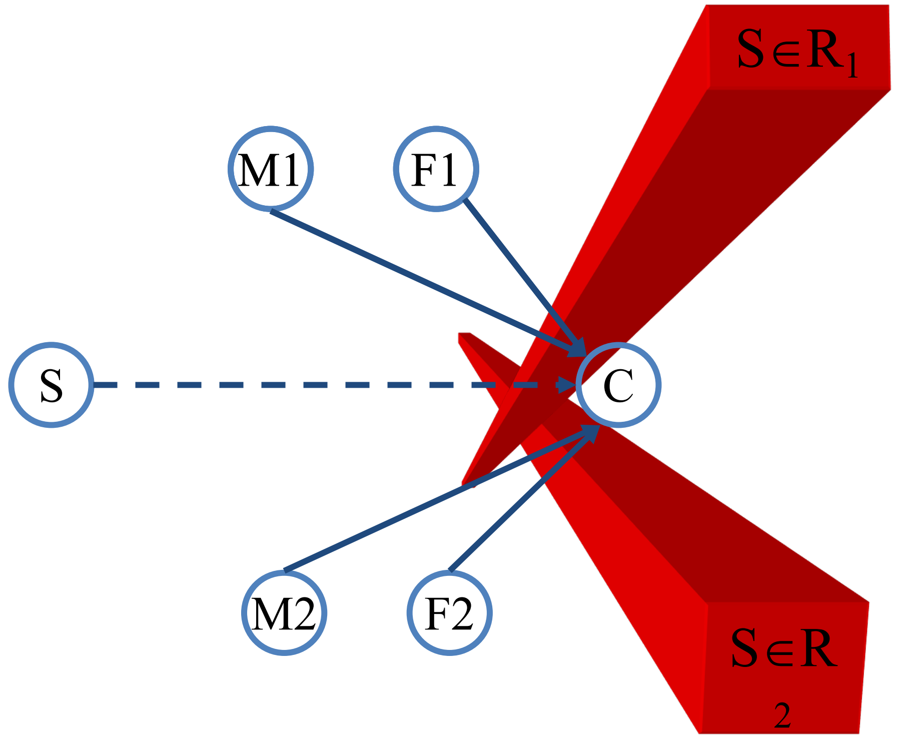
Prologue P Repeating Chunk C Epilogue E



Prologue Chunk Unrolled 1 time Epilogue



Switching Parents



$$P(C \mid M1, F1, M2, F2) = \sum_i P(C \mid M_i, F_i, S \in R_i) P(S \in R_i)$$

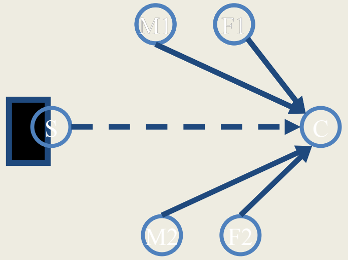
Switching Parents

- Switching parents can also be used to switch CPT implementations
- Here, depending on the value of wordTransistion(-1), word(0) will use either a “copy” or will use a “wordBigram” as the CPT.

```
variable : word {
    type: discrete hidden cardinality VOCAB_SIZE;
    switchingparents: wordTransition(-1)
      using mapping("directMappingWithOneParent");
    conditionalparents:
        word(-1) using DeterministicCPT("copyCPT")
      | word(-1) using DenseCPT("wordBigram");
 }
```

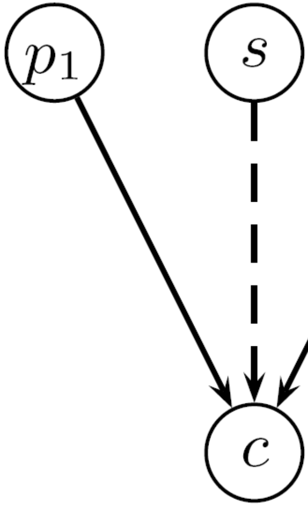
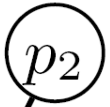
GMTK Switching Structure

```
variable : S {
        type : discrete hidden cardinality 100;
        switchingparents : nil;
        conditionalparents : nil using DenseCPT("pi");
}
variable : M1 {...}
variable : F1 {...}
variable : M2 {...}
variable : F2 {...}
variable : C {
   type : discrete hidden cardinality 30;
   switchingparents : S(0) using mapping("S-mapping");
   conditionalparents :
        M1(0),F1(0) using DenseCPT("M1F1")
      | M2(0),F2(0) using DenseCPT("M2F2");
}
```



Switching Parents

```
    variable : C {
      type: discrete hidden cardinality STATES;
      switchingparents:
          S(0)using
   mapping("directMappingWithOneParent");
      conditionalparents:
          p1(0) using DenseCPT("probGivenP1") |
          p2(0) using DenseCPT("probGivenP2");
   }
```

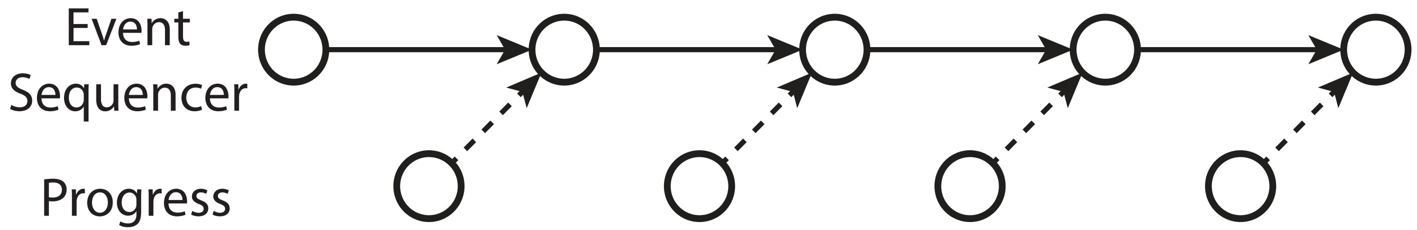



$$\begin{tikzcd}
p_1 \arrow[dr] & s \arrow[d, dashed] & p_2 \arrow[dl] \\
 & c &
\end{tikzcd}$$

Switching parents used here
to switch between parents but
this is not all that they can do …

Stochastic Event Sequencer w. Switching Parents

- Event sequencer, sequences through an order of events.
- Binary progress random variable controls progress.

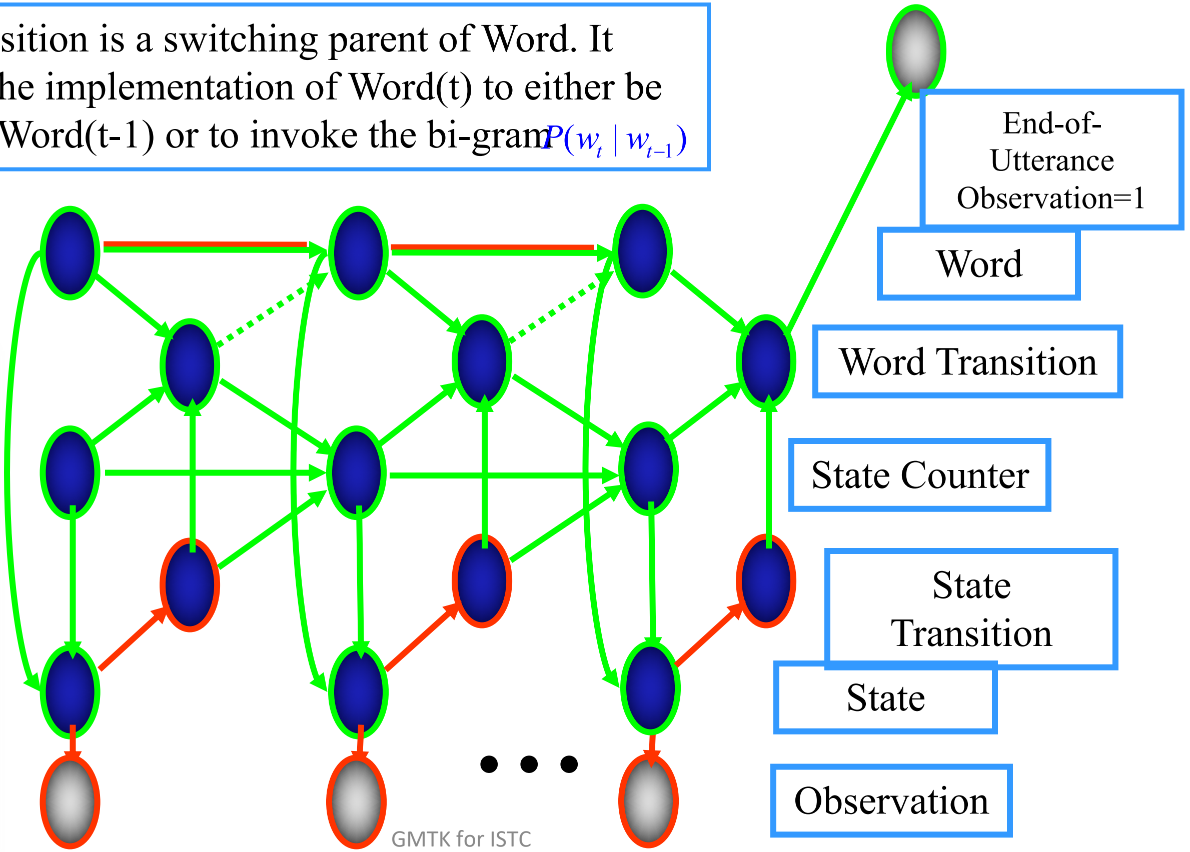


Explicit bi-gram Decoder

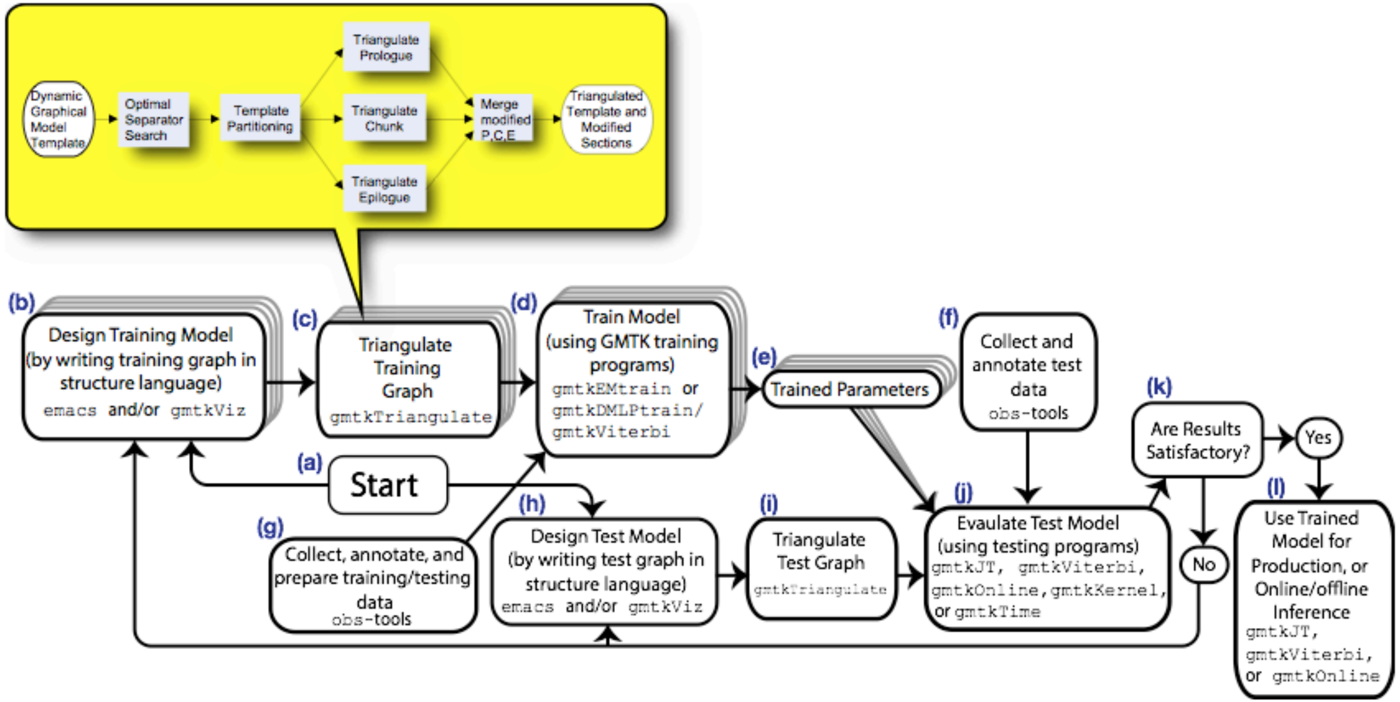
WordTransition is a switching parent of Word. It switches the implementation of Word(t) to either be a copy of Word(t-1) or to invoke the bi-gram $P(w_t \mid w_{t-1})$

Nodes & Edges:

- Red ⇔ RANDOM
- Green ⇔ Deterministic
- Dashed line ⇔ Switching Parent



GMTK Model Design/Use Workflow



Dynamic Graphical Model Template
Optimal Separator Search
Template Partitioning
Triangulate Prologue
Triangulate Chunk
Triangulate Epilogue
Merge modified P,C,E
Triangulated Template and Modified Sections
(b) Design Training Model (by writing training graph in structure language) emacs and/or gmtkViz
(c) Triangulate Training Graph gmtkTriangulate
(d) Train Model (using GMTK training programs) gmtkEMtrain or gmtkDMLPtrain/ gmtkViterbi
(e) Trained Parameters
(f) Collect and annotate test data obs-tools
(k) Are Results Satisfactory?
Yes
No
(a) Start
(g) Collect, annotate, and prepare training/testing data obs-tools
(h) Design Test Model (by writing test graph in structure language) emacs and/or gmtkViz
(i) Triangulate Test Graph gmtkTriangulate
(j) Evaulate Test Model (using testing programs) gmtkJT, gmtkViterbi, gmtkOnline, gmtkKernel, or gmtkTime
(l) Use Trained Model for Production, or Online/offline Inference gmtkJT, gmtkViterbi, or gmtkOnline

Parameter Definitions - MasterFile

- Parameters are stored either via a MasterFile or via a “trainable file”
- For each parameter either
  - It is stored directly in the MasterFile
  - MasterFile gives a pointer to the location of the file where the parameters are stored
  - The master file can include all the parameters that could be trained.
- Input/Output Trainable parameters
  - The trainable file is a short-cut to save/restore parameters that might change during training.

Parameter Definitions - CPTs

- There are many types of CPTs in GMTK, which define the values used for p(child|parents).
- The types of CPTs used in the TIMIT tutorial include:
  - Dense – all values of p(a|b) are included, leading to O(r^K) sized table
  - Deterministic – ‘a’ is deterministically related to its parents, so what is specified is that functional relationship
- There are many others as well, including
  - Sparse – ‘a’ is such that only a sub-set of ‘a’s possible values are specified, i.e., the CPT is sparse.
  - Language/factored language model, virtual evidence
  - Deep (many-layered neural network) CPTs

Deterministic CPT

- Value of child given its parents is a deterministic function
- A=f(B), p(A|B)=1 or p(A|B)=0
- Decision trees (with functional leaves) are used for specifying the deterministic function f(B) (these will be mentioned soon)

```
wordPosition
3       % number of parents
2 2 13 13       % cardinalities of parents, self
wordPosition_DT % decision tree
```

GMTK's Decision Trees

- Decision trees - for specifying arbitrary deterministic functions: integer to integer mappings
- Leaf-node formulas (including conditionals and functions).
- Can also be expressed directly in C++ for efficiency.
- Example:

```
wordPositionDT  % Name
3               % Number of Parents
1 2 0 default
    -1 { p2+1 }
    0 2 1 default
        -1 0
        -1 7

phoneCounterDT  % Name
2               % Number of Parents
-1 { min(p0+1, 7*(p1/4)-3, mc) }
```

- Can also be expressed directly in C++ for efficiency.

Gaussian and Conditional Gaussian Parameters

- Mixture of components
  - Gives list of components
- Component
  - Specifies mean and variance if Gaussian
- Mean, Varience, DPMF (mixture weights)
- Also, dlinks for sparse autoregressive multivariate Gaussians

➢ Defined in this way so that anything can be shared

EM Training

- The EM (expectation/maximization) algorithm is used for maximum-likelihood parameter training.
- Useful when there is missing values.
- gmtkEMtrain
  - The program that reads in a graph and initial set of parameters, and iterates via the EM (expectation-maximization) algorithm.
- Discriminative training also possible (but much slower, not part of tutorial currently)

EM Training

- Gaussian Splitting and Vanishing
  - Gaussians are initialized as single component, zero mean, unit variance
  - Single components trained
  - Single components are split into two components, then trained
  - Repeat
- Optionally split only components with high mixture weights
- Optionally vanish components with low mixture weights

Viterbi Decoding

- gmtkViterbi
  - Also called MPE (most probable explanation)
  - This is what is used for decoding, you run this to get, say, a list of activities/phones/etc. based on what the input observations
  - Implements:

$$(h^*_{1:T}, y^*_{1:T}) \in \underset{h_{1:T}, y_{1:T}}{\operatorname{argmax}} \Pr(\bar{x}_{1:T}, h_{1:T}, y_{1:T})$$

ChromHMM vs. Segway

	ChromHMM	Segway
Modeling framework	Hidden Markov model	Dynamic Bayesian network
Genomic resolution	200 bp	1 bp
Data resolution	Boolean	Real value
Handling missing data	Interpolation	Marginalization
Emission modeling	Bernoulli distribution	Gaussian distribution
Length modeling	Geometric distribution	Geometric distribution plus hard and soft constraints

Hoffman MM *et al*. 2013. *Nucleic Acids Res* 41:827.