MIT 18.700/18.701/6.047/6.878/HST.507
Computational Biology: Genomes, Networks, Evolution

Lecture 5 – Epigenomics, HMMs

read mapping – peak calling – multivariate HMMs
HMM Foundations, Generating, Parsing, Decoding, Learning

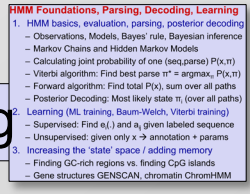
Prof. Manolis Kellis – manoli@mit.edu – MIT Computer Science & AI Lab

MLCB - Machine Learning in Computational Biology - Fall 2024 - Profs. Manolis Kellis + Eric Alm - 6.8700/6.8701/20.s900/20.s948/HST.507

Homeworks	Project / Mentoring (Fri)	Wk	Date	Lec	Topic
			Introduction: Machine Learning, Deep Learning, Generative AI, and the Unification of Biology		
HW0 out Thu 9/5		1	Thu-Sep-05	L1	Course Overview, Machine Learning, Deep Learning, Inference, Genome, Proteins, Chemistry, Imaging
			Module 1: Genomics, Epigenomics, Single-Cell, Networks, Circuitry		
HW0 due Wed 9/11		2	Tue-Sep-10	L2	Expression Analysis, Clustering/Classification, Gaussian Mixture Models, K-means, Bayesian Inf, Gen-vs-DiscrML
HW1 out Thu 9/12	0=Self Introductions	2	Thu-Sep-12	L3	Single-cell genomics, sc-mutli-omics, non-linear embeddings, spatial transcriptomics, next-gen technologies
		3	Tue-Sep-17	L4	Sequential Data, Alignment, DynProg, Hidden Markov Models, Parsing, Posterior Decoding, HMM architectures
		3	Thu-Sep-19	L5	Epigenomics: Signal Modeling, Peak calling, Chromatin states, 3D structure, Hi-C, Genome Topology
		4	Tue-Sep-24	L6	Regulatory Genomics: Motifs, Information, ChIP, Gibbs Sampling, EM, CNNs for Genome Parsing
HW1 due Mon 9/30	1=Select previous paper(s)	4	Thu-Sep-26	L7	Regulatory Networks: Graphs, Linear Algebra, PCA, SVD, Dimentionality Reduction, TF-enhancer-gene circuitry
			Module 2: Protein Structure, Protein Language Models, Geometric Deep Learning		
HW2 out Thu 10/3		5	Tue-Oct-01	L8	Intro to structural biology
		5	Thu-Oct-03	L9	Protein structure and folding: Diffusion models, Cryo-EM, Protein design
		6	Tue-Oct-08	L10	Intro to transformers and Large Language Models LLMs
	2=Proposal+Feasibility	6	Thu-Oct-10	L11	Protein Language Models PLMs and Transfer Learning
		7	Tue-Oct-15	-	-- No Class -- Student holiday
HW2 due Mon 10/21		7	Thu-Oct-17	L12	DNA language models: Chromatin Structure
			Module 3: Chemistry, Therapeutics, Graph Neural Networks		
HW3 out Thu 10/24		8	Tue-Oct-22	L13	Overview of drug development
	3=OffHrs|Update|Feedback	8	Thu-Oct-24	L14	Intro to small molecules
		9	Tue-Oct-29	L15	Representation of small molecules: Graphs, GNNs, Transformers, RDKit
		9	Thu-Oct-31	L16	Docking: Small molecule - proteins docking
		10	Tue-Nov-05	L17	Disease Association Mapping, genetics, GWAS, linkage analysis, disease circuitry, variant-to-function
HW3 due Tue 11/12	4=OffHrs|Update|Feedback	10	Thu-Nov-07	L18	Quantitative trait mapping, molecular traits, eQTLs, mediation analysis, iMWAS, multi-modal QTLs
			Module 4: Electronic Health Records, Imaging, Evolution, Metabolism		
No HW		11	Tue-Nov-12	L19	Electronic Health Records, AllOfUs, UKBioBank, Medical Genomics, Pop-Scale Cohorts, Multi-Ancestry [not quiz'd]
		11	Thu-Nov-14	L20	**-- In-class Quiz**
		12	Tue-Nov-19	L21	Imaging methods for biological applications
	5=Midcourse report	12	Thu-Nov-21	L22	Comparative genomics, Conservation, Evolutionary signatures, PhyloCSF, RNA structure, Motif BLS2conf
		13	Tue-Nov-26	L23	Evolution, Phylogenetics, Phylogenomics, Duplication, RNA world, RNA folding, lincRNAs, RNA modifications, m6A
		13	Thu-Nov-28	-	-- No Class -- Thanksgiving Holiday
		14	Tue-Dec-03	L24	Modeling metabolism: Flux balance analysis
	6=WriteUp, Slides Due	14	Thu-Dec-05	L25	Measuring metabolism: Metabolomics and Deep Learning
			Final Projects		
	7=In Class Presentations	15	Tue-Dec-10	L26	**Project Presentations (6-8 mins/team). Report due Fri@11.59p, Slides due Mon@11.59p, Present Live Tue**

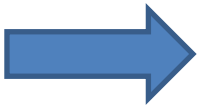
Epigenomics, ChIP, BWT, HMMs, EM, Hi-C

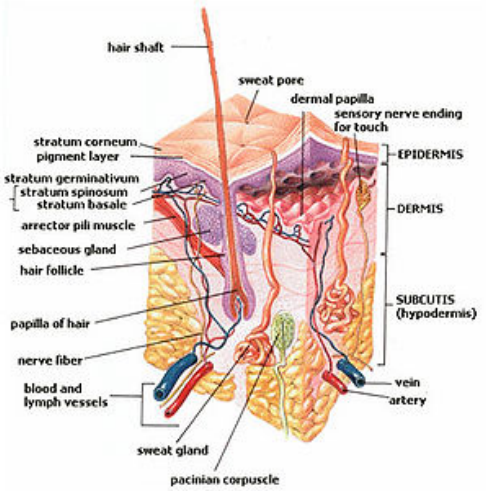
1. Introduction to Epigenomics
   - Overview of epigenomics, Diversity of Chromatin modifications
   - Antibodies, ChIP-Seq, data generation projects, raw data
2. Primary data processing: Read mapping, Peak calling
   - Read mapping: Hashing, Suffix Trees, Burrows-Wheeler Transform
   - Quality Control, Cross-correlation, Peak calling, IDR (similar to FDR)
3. Discovery and characterization of chromatin states
   - HMM Foundations, Generating, Parsing, Decoding, Learning
   - ChromHMM: Multi-variate HMM for chromatin state learning
4. Model complexity: selecting the number of states/marks
   - Selecting the number of states, selecting number of marks
   - Capturing dependencies and state-conditional mark independence
5. Learning chromatin states jointly across multiple cell types
   - Stacking vs. concatenation approach for joint multi-cell type learning
   - Defining activity profiles for linking enhancer regulatory networks

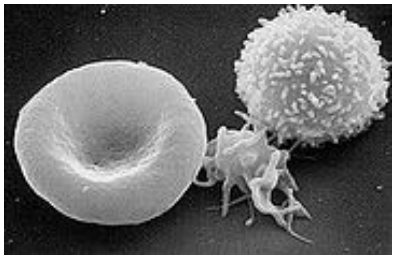


One Genome – Many Cell Types

ACCAGTTACGACGGTCA
GGGTACTGATACCCCAA
ACCGTTGACCGCATTTA
CAGACGGGGTTTGGGTT
TTGCCCCACACAGGTAC
GTTAGCTACTGGTTTAG
CAATTTACCGTTACAAC
GTTTACAGGGTTACGGT
TGGGATTTGAAAAAAAG
TTTGAGTTGGTTTTTTC
ACGGTAGAACGTACCGT
TACCAGTA







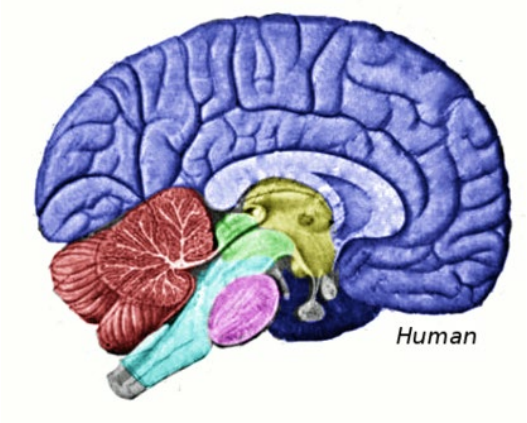
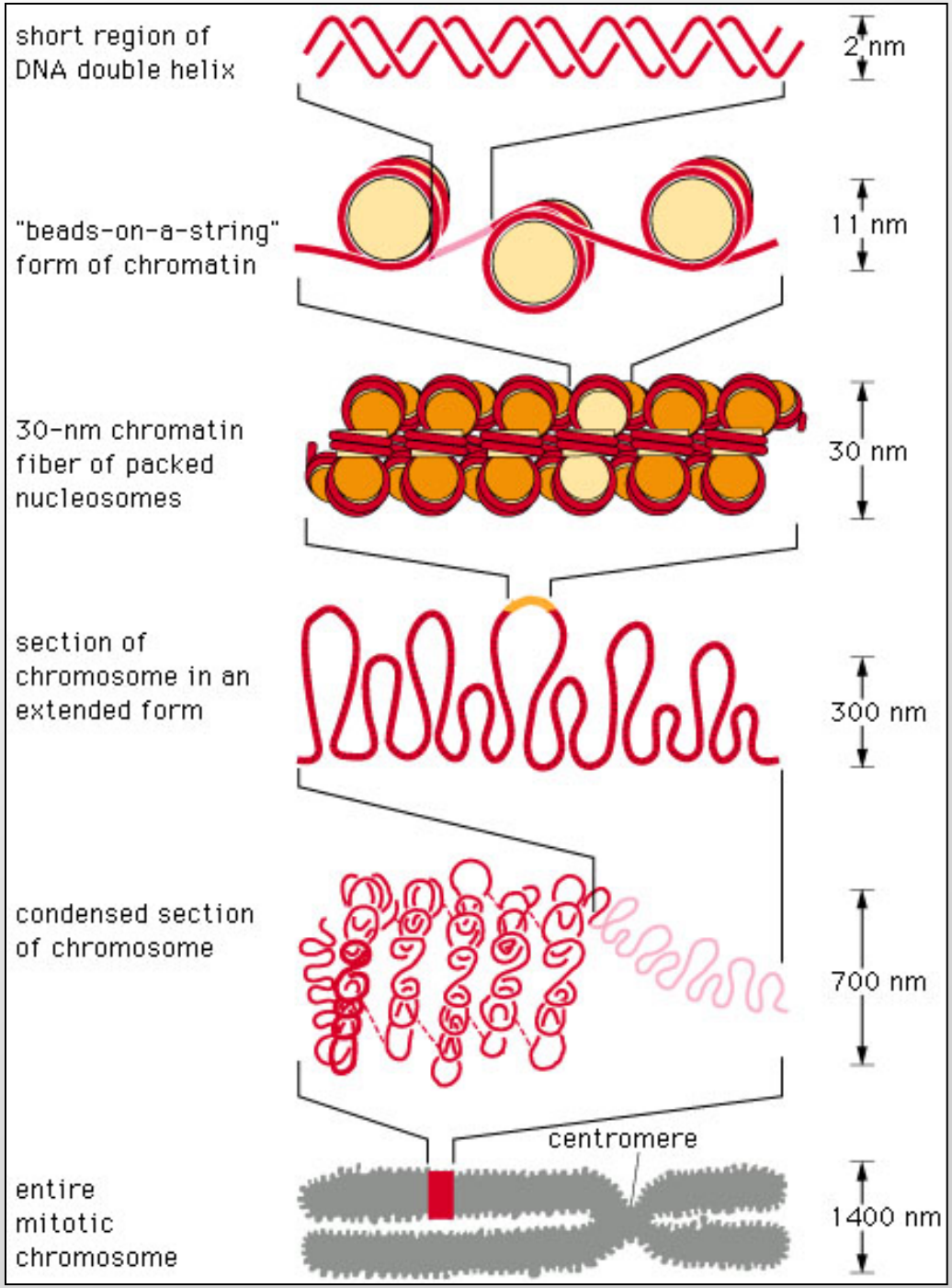


Image Source wikipedia

DNA packaging

- Why packaging
  - DNA is very long
  - Cell is very small
- Compression
  - Chromosome is 50,000 times shorter than extended DNA
- Using the DNA
  - Before a piece of DNA is used for anything, this compact structure must open locally
- Now emerging:
  - Role of accessibility
  - State in chromatin itself
  - Role of 3D interactions



Three types of epigenetic modifications

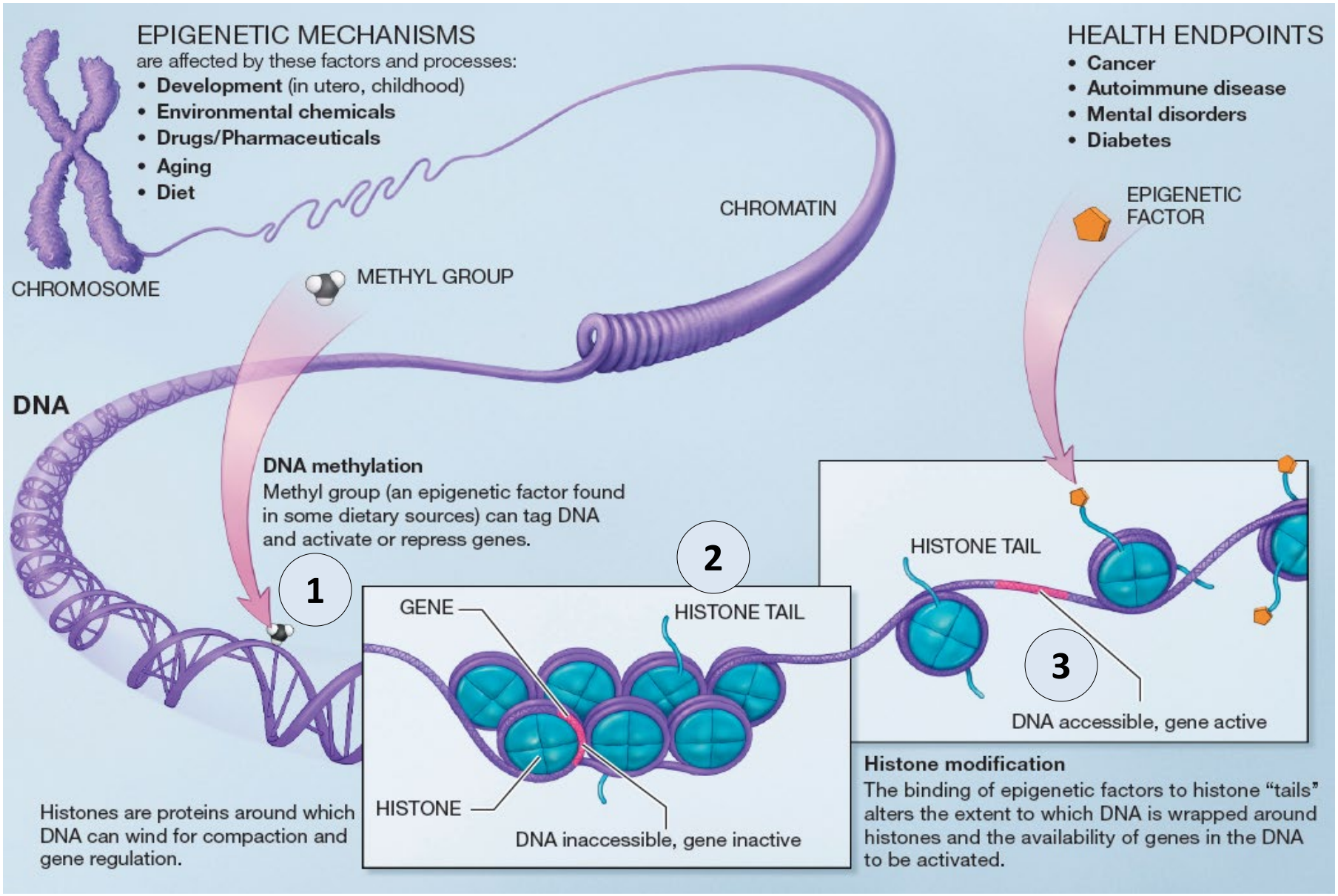
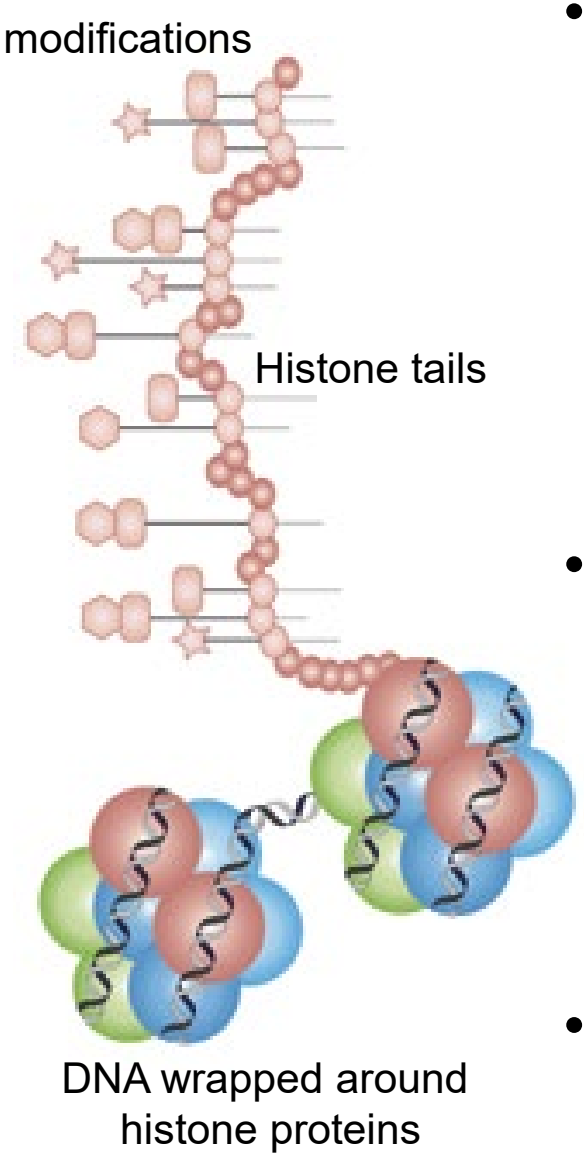


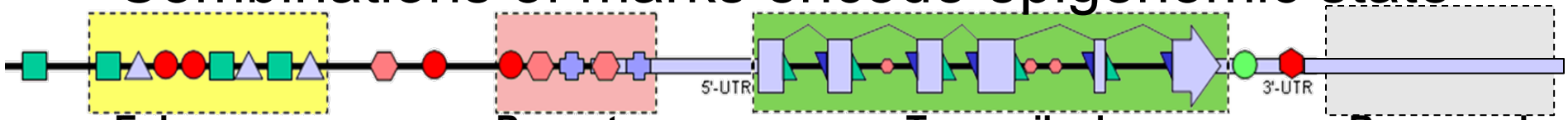
Image source: http://nihroadmap.nih.gov/epigenomics/

100s of histone tail modifications



- 100+ different histone modifications
  - Histone protein → H3/H4/H2A/H2B
  - AA residue → Lysine4(K4)/K36…
  - Chemical modification → Met/Pho/Ubi
  - Number → Me-Me-Me(me3)
  - Shorthand: H3K4me3, H2BK5ac
- In addition:
  - DNA modifications
  - Methyl-C in CpG / Methyl-Adenosine
  - Nucleosome positioning
  - DNA accessibility
- The constant struggle of gene regulation
  - TF/histone/nucleo/GFs/Chrom compete

Combinations of marks encode epigenomic state

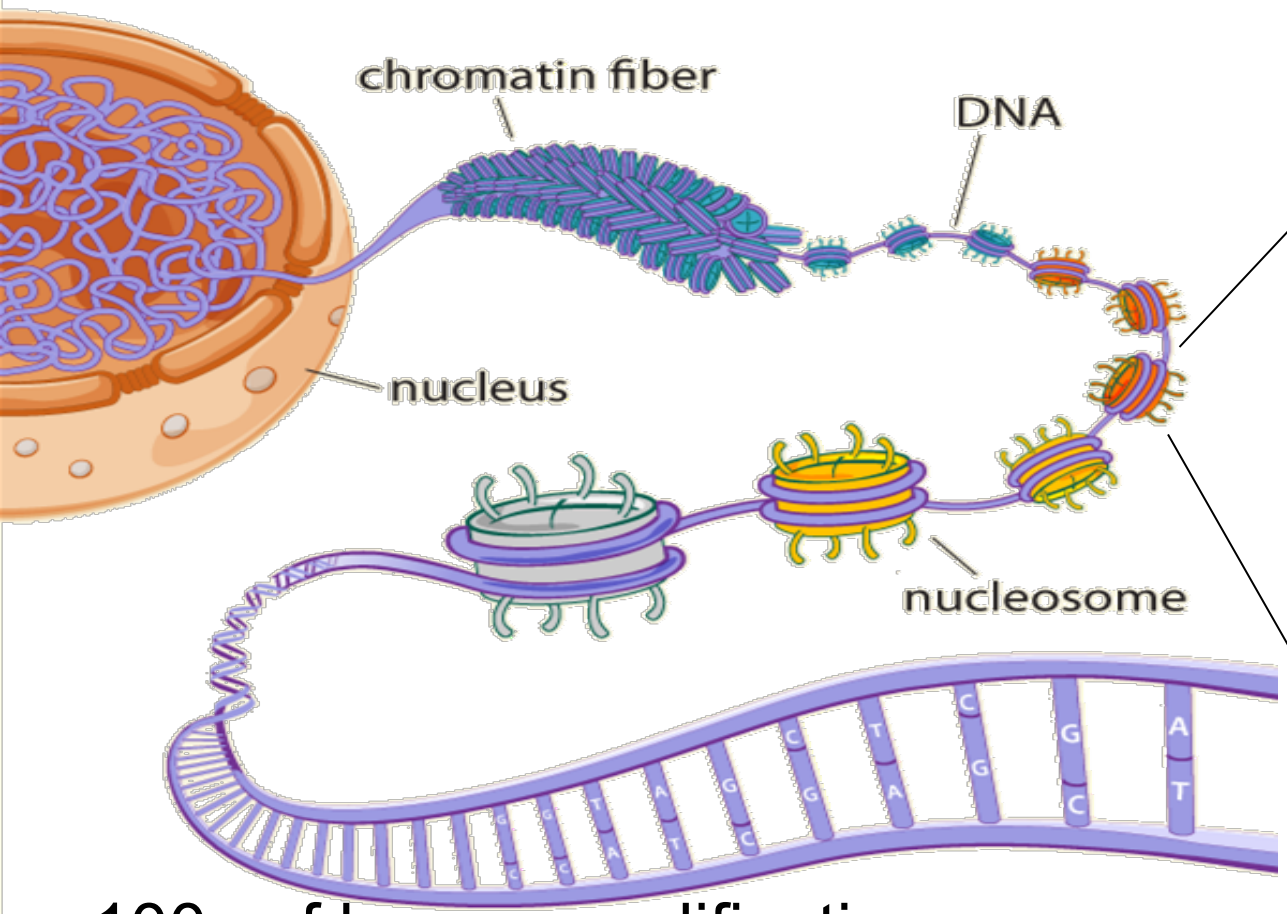


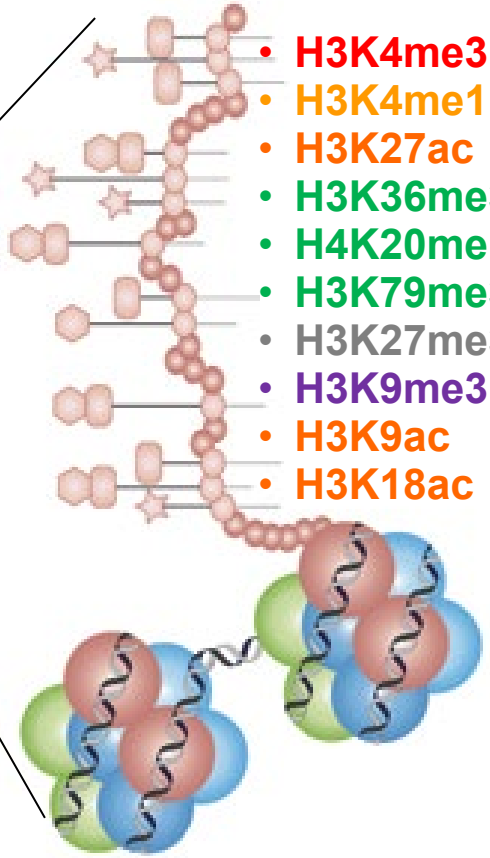
Enhancers
- H3K4me1
- H3K27ac
- DNase

Promoters
- H3K4me3
- H3K9ac
- DNase

Transcribed
- H3K36me3
- H3K79me2
- H4K20me1

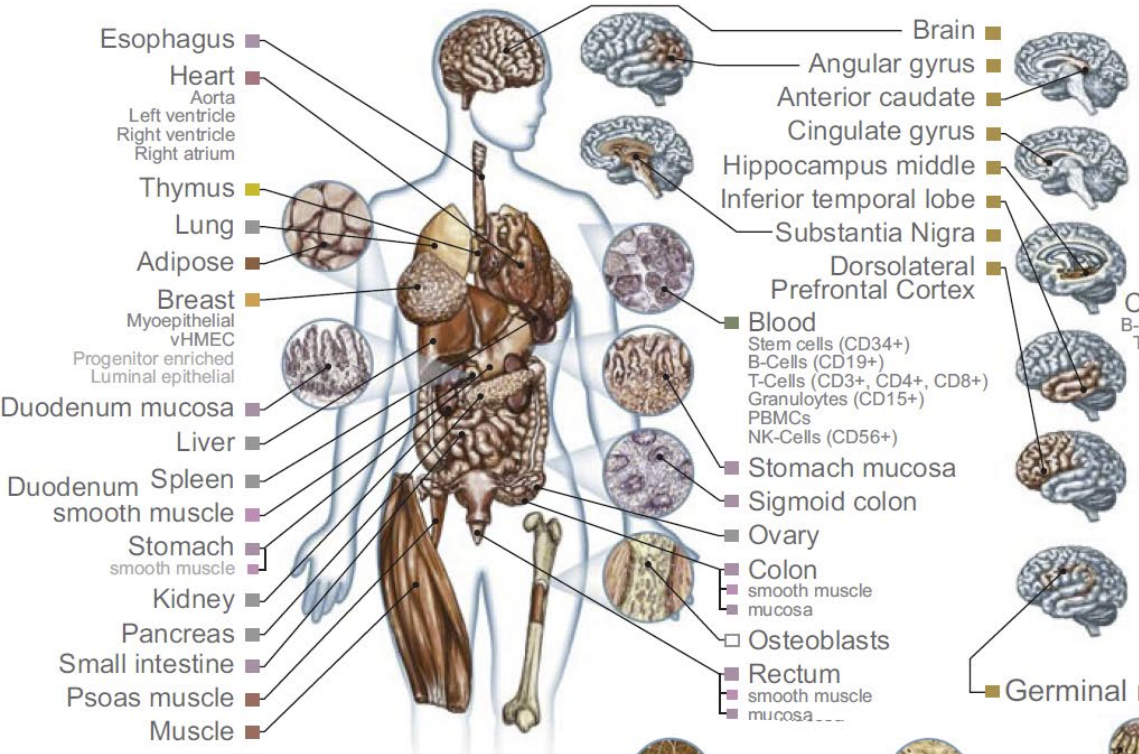
Repressed
- H3K9me3
- H3K27me3
- DNAmethyl

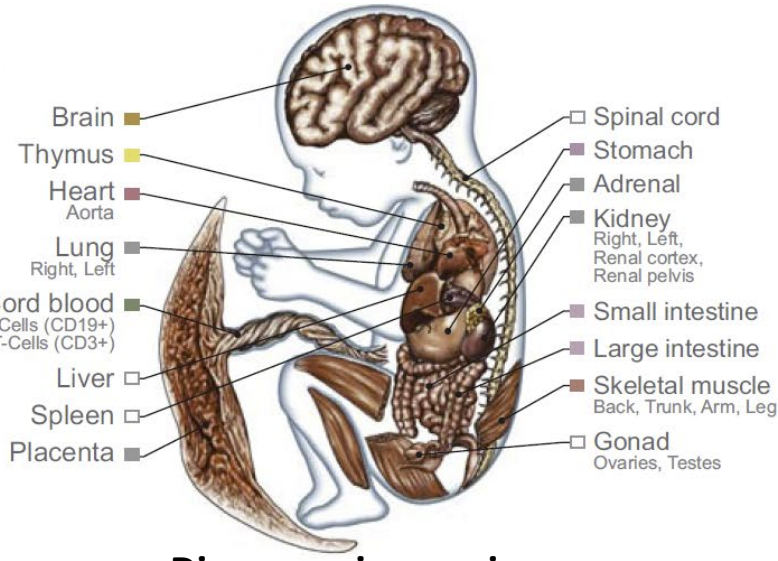




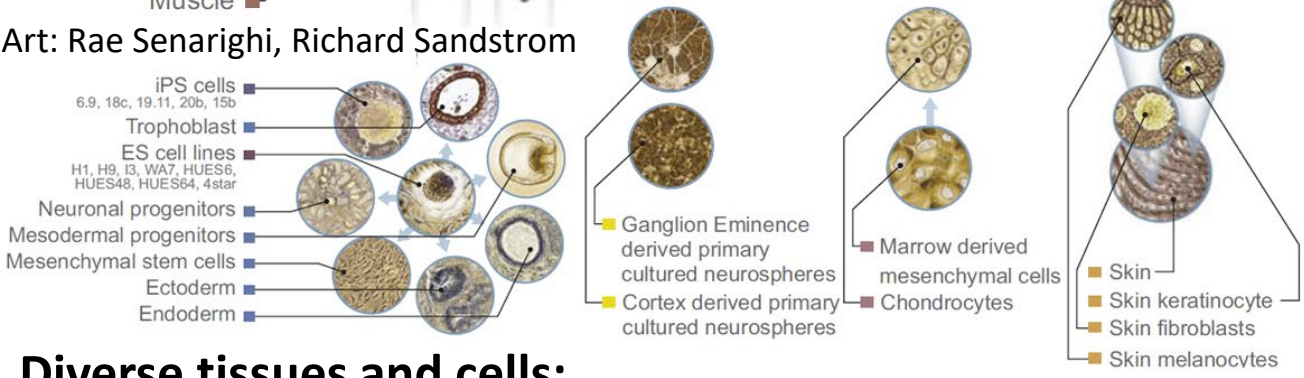
- 100s of known modifications, many new still emerging
- Systematic mapping using ChIP-, Bisulfite-, DNase-Seq

Epigenomics Roadmap across 100+ tissues/cell types





Art: Rae Senarighi, Richard Sandstrom



Diverse epigenomic assays:

1. **Histone modifications**
   - **H3K4me3, H3K4me1**
   - **H3K36me3**
   - **H3K27me3, H3K9me3**
   - **H3K27/9ac, +20 more**
2. **Open chromatin**:
   - **DNase**
3. **DNA methylation**:
   - **WGBS, RRBS, MRE/MeDIP**
4. **Gene expression**
   - **RNA-seq, Exon Arrays**

Diverse tissues and cells:

1. **Adult tissues and cells** (brain, muscle, heart, digestive, skin, adipose, lung, blood…)
2. **Fetal tissues** (brain, skeletal muscle, heart, digestive, lung, cord blood…)
3. **ES cells, iPS, differentiated cells** (meso/endo/ectoderm, neural, mesench, trophobl)

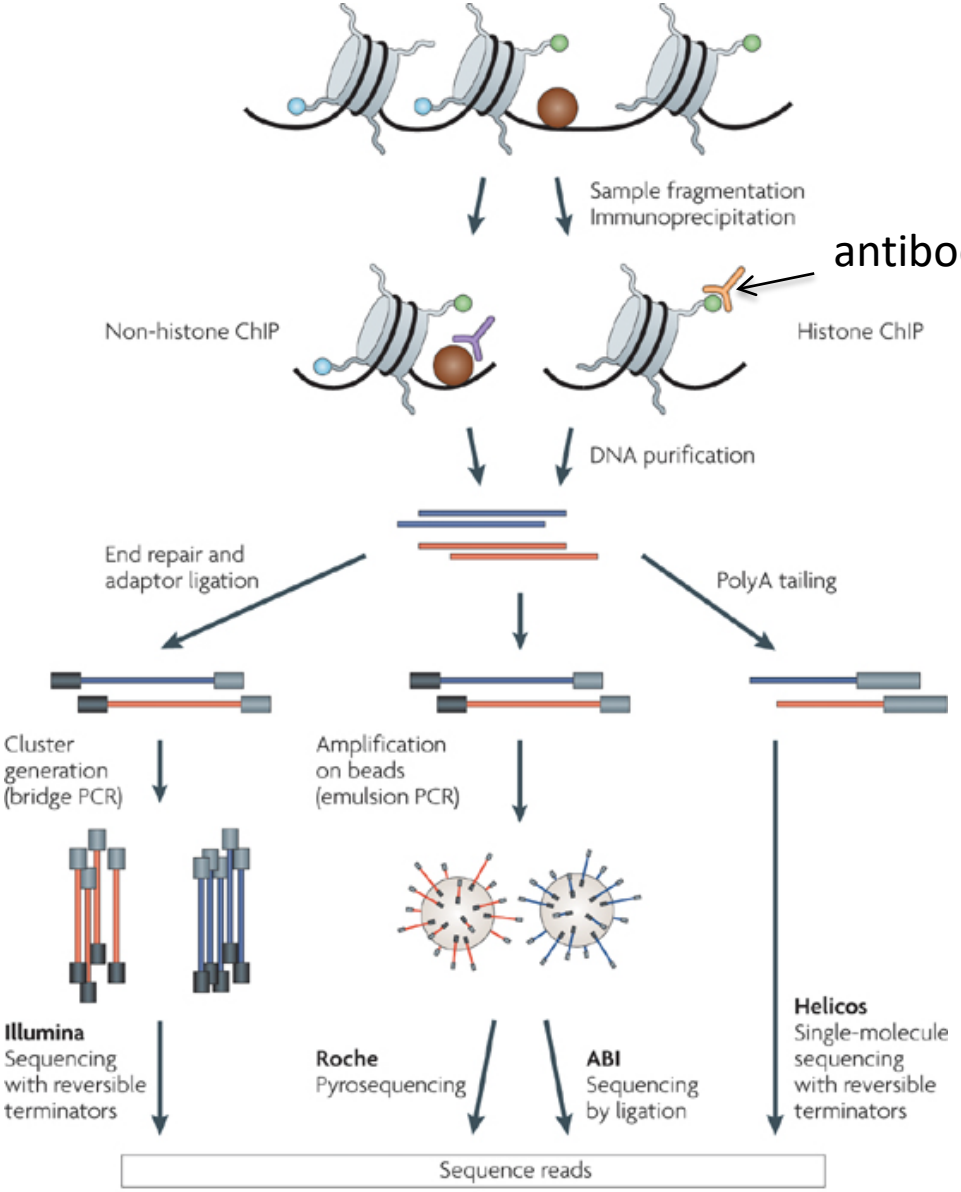
Ongoing epigenomic mapping projects

- Mapping multiple modifications
- In multiple cell types
- In multiple individuals
- In multiple species
- In multiple conditions
- With multiple antibodies
- Across the whole genome

- First waves published
- Lots more in pipeline
- Time for analysis!

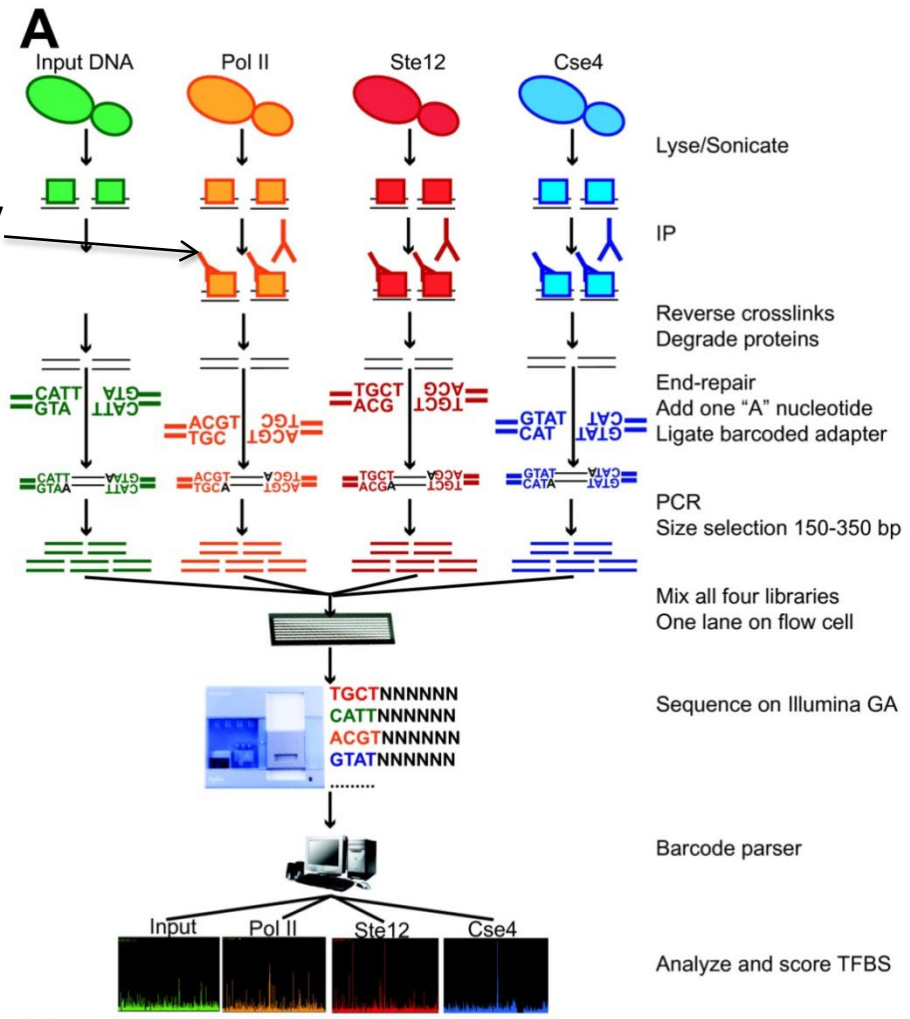
ChIP-seq review

(Chromatin immunoprecipitation followed by sequencing)



Nature Reviews | Genetics

antibody



Bar-coded multiplexed sequencing

ChIP-chip and ChIP-Seq technology overview

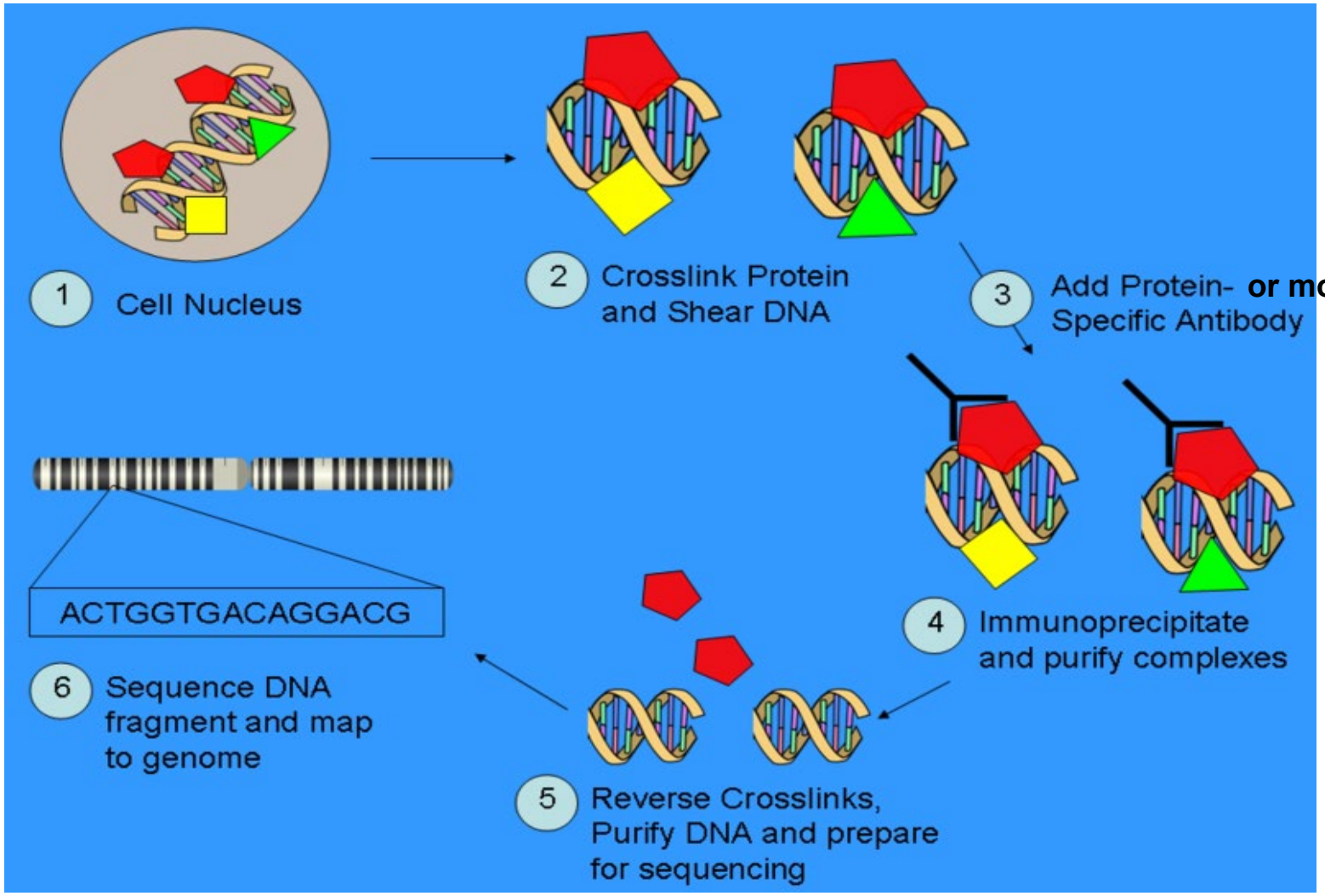


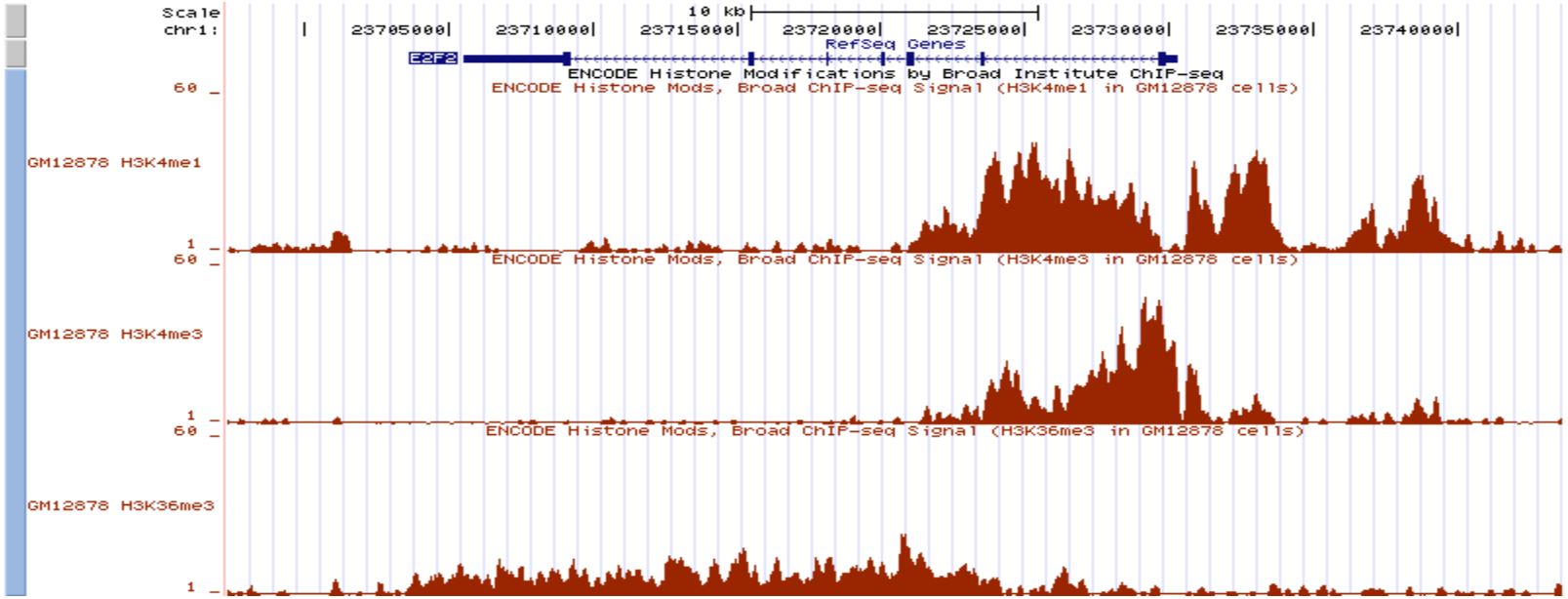
Image adapted from Wikipedia

Modification-specific antibodies → Chromatin Immuno-Precipitation

followed by: ChIP-chip: array hybridization

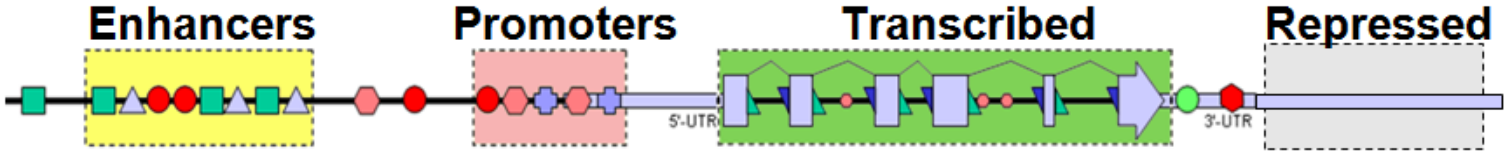
ChIP-Seq: Massively Parallel Next-gen Sequencing

ChIP-Seq Histone Modifications: What the raw data looks like



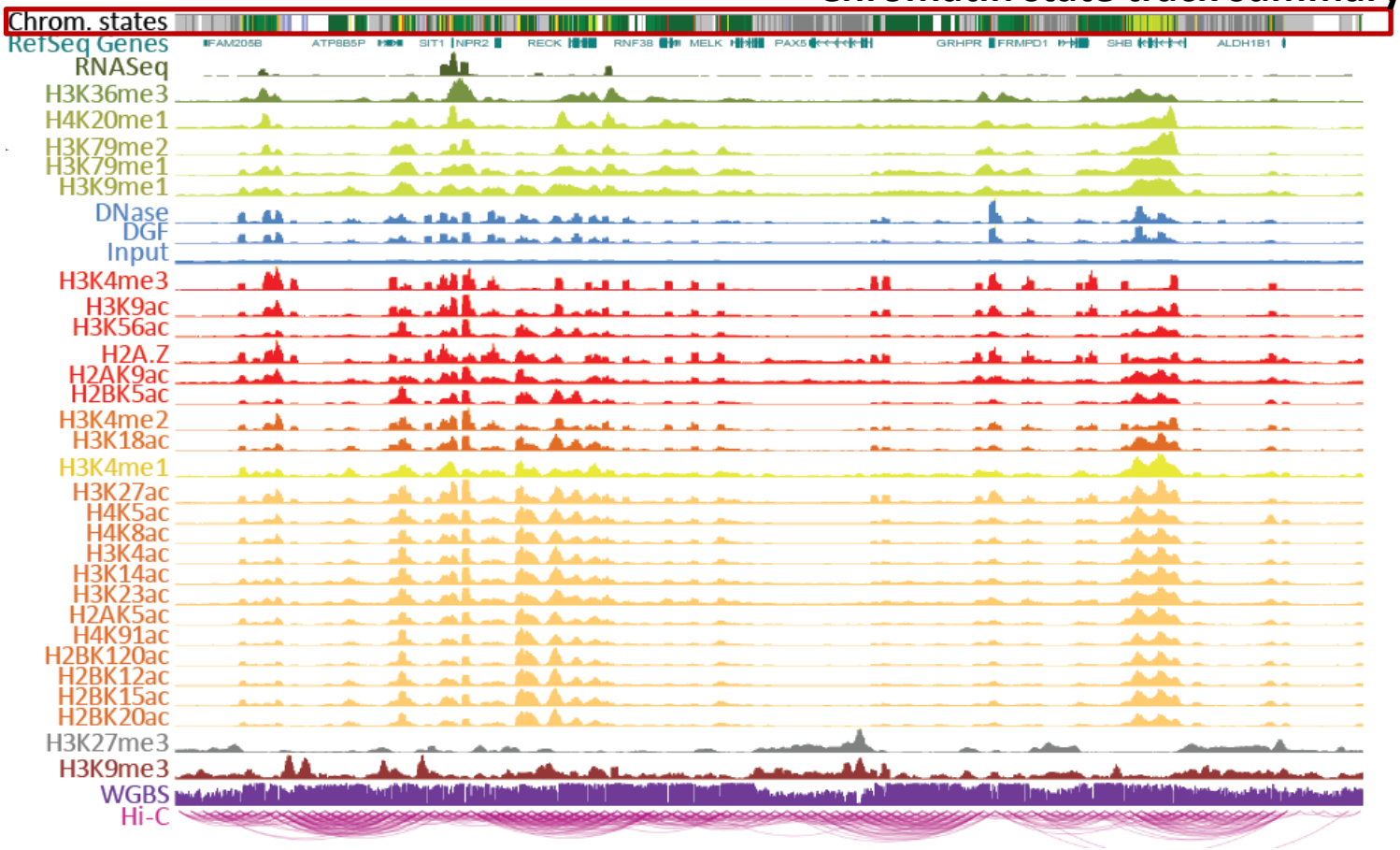
- Each sequence tag is 30 base pairs long
- Tags are mapped to unique positions in the ~3 billion base reference genome
- Number of reads depends on sequencing depth. Typically on the order of 10 million mapped reads.

Summarize multiple marks into chromatin states



Chromatin state track summary

30+ epigenomics marks

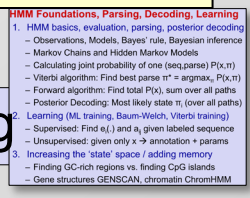


WashU Epigenome Browser

ChromHMM: multi-variate hidden Markov model

Goals for today: Computational Epigenomics

1. Introduction to Epigenomics
   - Overview of epigenomics, Diversity of Chromatin modifications
   - Antibodies, ChIP-Seq, data generation projects, raw data
2. Primary data processing: Read mapping, Peak calling
   - Read mapping: Hashing, Suffix Trees, Burrows-Wheeler Transform
   - Quality Control, Cross-correlation, Peak calling, IDR (similar to FDR)
3. Discovery and characterization of chromatin states
   - HMM Foundations, Generating, Parsing, Decoding, Learning
   - ChromHMM: Multi-variate HMM for chromatin state learning
4. Model complexity: selecting the number of states/marks
   - Selecting the number of states, selecting number of marks
   - Capturing dependencies and state-conditional mark independence
5. Learning chromatin states jointly across multiple cell types
   - Stacking vs. concatenation approach for joint multi-cell type learning
   - Defining activity profiles for linking enhancer regulatory networks



Mapping millions of short reads to the genome

Traditional Hashing Schemes

Burrows-Wheeler Transform (BWT)

Mapping Reads to the Genome

- Assign reads to best matching location in reference genome
- 10,000,000s of reads, ~30 bases long
- Example: CAGGGCTGATTGAGGACATTCATCACG
- Allow mismatches: sequencing errors, or SNPs
- Algorithmic and memory efficiency is critical

How would you do it:

- Sequence alignment: O(m*n)
- Hashing / BLAST: O(m+n)
  - Solution until 2008 (e.g. MAQ, Li et al, GR 2008)
- Other advanced algorithms:
  - Linear-time string matching: O(m+n). L3 addendum
  - Suffix trees and suffix arrays: O(m). L13 addendum
- Challenge: memory requirements
  - Hash table, suffix tree/array require O(m*n) space
- Today: Burrows-Wheeler transformation O(m)
  - Ultrafast/memory efficient. New norm since 2009.
  - Introduced in: Bowtie (Langmead GB 2009).

Second Generation Mappers have Leveraged the Burrows Wheeler Transformation

Software

Open Access

Ultrafast and memory-efficient alignment of short DNA sequences to the human genome

Ben Langmead, Cole Trapnell, Mihai Pop and Steven L Salzberg

Address: Center for Bioinformatics and Computational Biology, Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA.

Correspondence: Ben Langmead. Email: langmead@cs.umd.edu

Published: 4 March 2009

Genome Biology 2009, **10**:R25 (doi:10.1186/gb-2009-10-3-r25)

The electronic version of this article is the complete one and can be found online at http://genomebiology.com/2009/10/3/R25

Received: 21 October 2008
Revised: 19 December 2008
Accepted: 4 March 2009

Abstract

Bowtie is an ultrafast, memory-efficient alignment program for aligning short DNA sequence reads to large genomes. For the human genome, Burrows-Wheeler indexing allows Bowtie to align more than 25 million reads per CPU hour with a memory footprint of approximately 1.3 gigabytes. Bowtie extends previous Burrows-Wheeler techniques with a novel quality-aware backtracking algorithm that permits mismatches. Multiple processor cores can be used simultaneously to achieve even greater alignment speeds. Bowtie is open source http://bowtie.cbcb.umd.edu.

"…35 times faster than Maq and 300 times faster than SOAP under the same conditions"

BIOINFORMATICS APPLICATIONS NOTE

Vol. 24 no. 5 2008, pages 713–714
doi:10.1093/bioinformatics/btn025

Sequence analysis

SOAP: short oligonucleotide alignment program

Ruiqiang Li[1,2], Yingrui Li[1], Karsten Kristiansen[2] and Jun Wang[1,2,*]

[1]Beijing Genomics Institute at Shenzhen, Shenzhen 518083, China and [2]Department of Biochemistry and Molecular Biology, University of Southern Denmark, Odense M, DK-5230, Denmark

Received on November 10, 2007; revised on December 20, 2007; accepted on January 14, 2008

Advance Access publication January 28, 2008

Associate Editor: Keith Crandall

ABSTRACT

Summary: We have developed a program SOAP for efficient gapped and ungapped alignment of short oligonucleotides onto reference sequences. The program is designed to handle the huge amounts of short reads generated by parallel sequencing using the new generation Illumina-Solexa sequencing technology. SOAP is compatible with numerous applications, including single-read or pair-end resequencing, small RNA discovery and mRNA tag sequence mapping. SOAP is a command-driven program, which supports multi-threaded parallel computing, and has a batch module for multiple query sets.

Availability: http://soap.genomics.org.cn

Contact: soap@genomics.org.cn

SOAP will allow either a certain number of mismatches or one continuous gap for aligning a read onto the reference sequence. The best hit of each read which has minimal number of mismatches or smaller gap will be reported. For multiple equal-best hits, the user can instruct the program to report all, or randomly report one, or disregard all of them. Since the typical read length is 25–50 bp, hits with too many mismatches are unreliable which are hard to distinguish with random matches. By default, the program will allow at most two mismatches. Between two haplotype genome sequences, occurrence of single nucleotide polymorphism is much higher than that of small insertions or deletions, so ungapped hits have precedence over gapped hits. For gapped alignment only one continuous gap with a size ranging from 1 to 3 bp is accepted, while no

BIOINFORMATICS APPLICATIONS NOTE

Vol. 25 no. 15 2009, pages 1966–1967
doi:10.1093/bioinformatics/btp336

Sequence analysis

SOAP2: an improved ultrafast tool for short read alignment

Ruiqiang Li[1,2,†], Chang Yu[1,†], Yingrui Li[1], Tak-Wah Lam[3], Siu-Ming Yiu[3], Karsten Kristiansen[2] and Jun Wang[1,2,*]

[1]Beijing Genomics Institute at Shenzhen, Shenzhen, 518083, China, [2]Department of Biochemistry and Molecular Biology, University of Southern Denmark, Odense M, DK-5230, Denmark and [3]Department of Computer Science, University of Hong Kong, Hong Kong, China

Received on January 23, 2009; revised on April 27, 2009; accepted on May 24, 2009

Advance Access publication June 3, 2009

Associate Editor: Joaquin Dopazo

ABSTRACT

Summary: SOAP2 is a significantly improved version of the short oligonucleotide alignment program that both reduces computer memory usage and increases alignment speed at an unprecedented rate. We used a Burrows Wheeler Transformation (BWT) compression index to substitute the seed strategy for indexing the reference sequence in the main memory. We tested it on the whole human genome and found that this new algorithm reduced memory usage from 14.7 to 5.4 GB and improved alignment speed by 20–30 times. SOAP2 is compatible with both single- and paired-end reads. Additionally, this tool now supports multiple text and compressed file formats. A consensus builder has also been developed for consensus assembly and SNP detection from alignment of short reads on a reference genome.

Availability: http://soap.genomics.org.cn

Contact: soap@genomics.org.cn

variation map, will generate about 15 Tb of sequence using next-generation sequencing technologies. With even the fastest programs currently available, one would need ~1000 CPU months to align these short reads onto the human reference genome. Additionally, new methods are now needed to support longer reads as the existing methods were primarily designed for very short reads with typical lengths shorter than 50 bp. With improvements in sequencing chemistry and data processing algorithms, the Illumina Genome Analyzer can now generate up to 75–100 bp high-quality reads, and longer reads are expected in the near future.

Here, we have developed an improved version of SOAP, called SOAP2. The new program uses the Burrows Wheeler Transformation (BWT) compressed index instead of the seed algorithm that was used in the previous version for indexing the reference sequence in the main memory. Use of BWT substantially improved alignment speed; additionally, it significantly reduced memory usage.

Mapping short DNA sequencing reads and calling variants using mapping quality scores

Heng Li,[1] Jue Ruan,[2] and Richard Durbin[1,3]

[1]*The Wellcome Trust Sanger Institute, Hinxton CB10 1SA, United Kingdom;* [2]*Beijing Genomics Institute, Chinese Academy of Science, Beijing 100029, China*

New sequencing technologies promise a new era in the use of DNA sequence. However, some of these technologies produce very short reads, typically of a few tens of base pairs, and to use these reads effectively requires new algorithms and software. In particular, there is a major issue in efficiently aligning short reads to a reference genome and handling ambiguity or lack of accuracy in this alignment. Here we introduce the concept of *mapping quality*, a measure of the confidence that a read actually comes from the position it is aligned to by the mapping algorithm. We describe the software MAQ that can build assemblies by mapping shotgun short reads to a reference genome, using quality scores to derive genotype calls of the consensus sequence of a diploid genome, e.g., from a human sample. MAQ makes full use of mate-pair information and estimates the error probability of each read alignment. Error probabilities are also derived for the final genotype calls, using a Bayesian statistical model that incorporates the mapping qualities, error probabilities from the raw sequence quality scores, sampling of the two haplotypes, and an empirical model for correlated errors at a site. Both read mapping and genotype calling are evaluated on simulated data and real data. MAQ is accurate, efficient, versatile, and user-friendly. It is freely available at http://maq.sourceforge.net.

[Supplemental material is available online at www.genome.org. Short-read sequences have been deposited in the European Read Archive (ERA) under accession no. ERA000012 (ftp://ftp.era.ebi.ac.uk/ERA000012/).]

18:1851–1858 ©2008 by Cold Spring Harbor Laboratory Press; ISSN 1088-9051/08; www.genome.org

BIOINFORMATICS ORIGINAL PAPER

Vol. 25 no. 14 2009, pages 1754–1760
doi:10.1093/bioinformatics/btp324

Sequence analysis

Fast and accurate short read alignment with Burrows–Wheeler transform

Heng Li and Richard Durbin*

Wellcome Trust Sanger Institute, Wellcome Trust Genome Campus, Cambridge, CB10 1SA, UK

Received on February 20, 2009; revised on May 6, 2009; accepted on May 12, 2009

Advance Access publication May 18, 2009

Associate Editor: John Quackenbush

ABSTRACT

Motivation: The enormous amount of short reads generated by the new DNA sequencing technologies call for the development of fast and accurate read alignment programs. A first generation of hash table-based methods has been developed, including MAQ, which is accurate, feature rich and fast enough to align short reads from a single individual. However, MAQ does not support gapped alignment for single-end reads, which makes it unsuitable for alignment of longer reads where indels may occur frequently. The speed of MAQ is also a concern when the alignment is scaled up to the resequencing of hundreds of individuals.

Results: We implemented Burrows-Wheeler Alignment tool (BWA), a new read alignment package that is based on backward search with Burrows–Wheeler Transform (BWT), to efficiently align short sequencing reads against a large reference sequence such as the human genome, allowing mismatches and gaps. BWA supports both base space reads, e.g. from Illumina sequencing machines, and color space reads from AB SOLiD machines. Evaluations on both simulated and real data suggest that BWA is ~10–20× faster than MAQ, while achieving similar accuracy. In addition, BWA outputs alignment in the new standard SAM (Sequence Alignment/Map) format. Variant calling and other downstream analyses after the alignment can be achieved with the open source SAMtools software package.

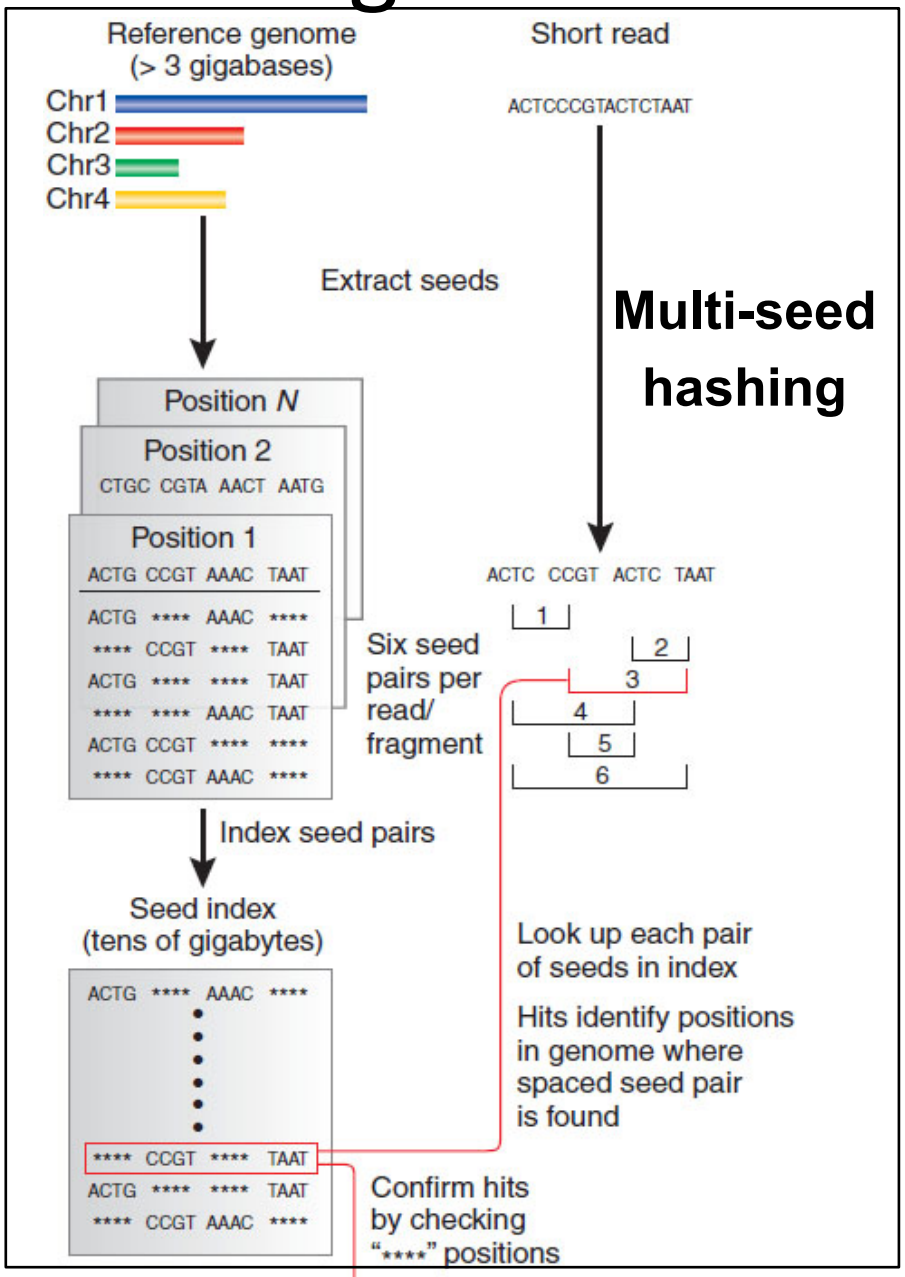
Availability: http://maq.sourceforge.net

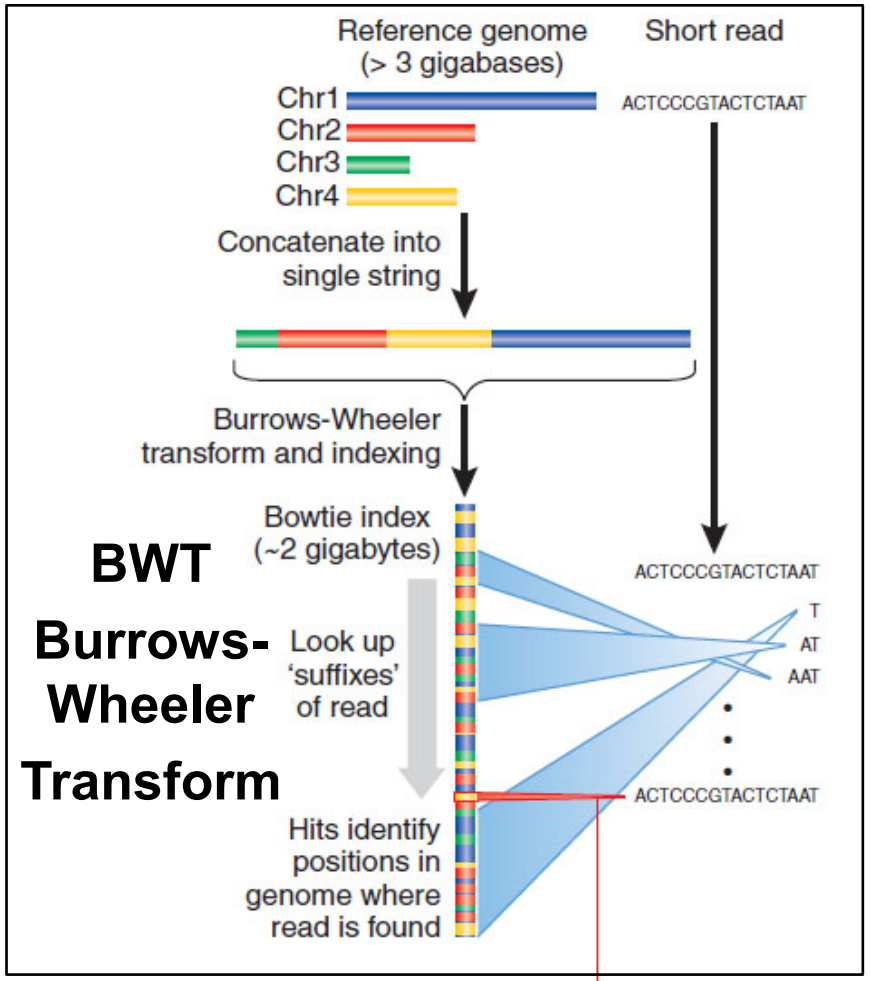
Contact: rd@sanger.ac.uk

of scanning the whole genome when few reads are aligned. The second category of software, including SOAPv1 (Li *et al.*, 2008b), PASS (Campagna *et al.*, 2009), MOM (Eaves and Gao, 2009), ProbeMatch (Jung Kim *et al.*, 2009), NovoAlign (http://www.novocraft.com), ReSEQ (http://code.google.com/p/re-seq), Mosaik (http://bioinformatics.bc.edu/marthlab/Mosaik) and BFAST (http://genome.ucla.edu/bfast), hash the genome. These programs can be easily parallelized with multi-threading, but they usually require large memory to build an index for the human genome. In addition, the iterative strategy frequently introduced by these software may make their speed sensitive to the sequencing error rate. The third category includes slider (Malhis *et al.*, 2009) which does alignment by merge-sorting the reference subsequences and read sequences.

Recently, the theory on string matching using Burrows–Wheeler Transform (BWT) (Burrows and Wheeler, 1994) has drawn the attention of several groups, which has led to the development of SOAPv2 (http://soap.genomics.org.cn/), Bowtie (Langmead *et al.*, 2009) and BWA, our new aligner described in this article. Essentially, using backward search (Ferragina and Manzini, 2000; Lippert, 2005) with BWT, we are able to effectively mimic the top-down traversal on the prefix trie of the genome with relatively small memory footprint (Lam *et al.*, 2008) and to count the number of exact hits of a string of length m in $O(m)$ time independent of the size of the genome. For inexact search, BWA samples from the implicit prefix trie the distinct substrings that are less than k edit distance

Hashing vs. Burrows Wheeler Transform





Today: How does the BW transform actually work?

Convert each hit back to genome location

Report alignment to user

Burrows-Wheeler Transform (BWT)

http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-124.pdf

- Transform: ^BANANA@ INTO: BNN^AA@A

```
function BWT (string s)
    create a table, rows are all possible rotations of s
    sort rows alphabetically
    return (last column of the table)
```

All Rotations	Sorted List of Rotations	Output Last Column
^BANANA@	ANANA@^B	
@^BANANA	ANA@^BAN	
A@^BANAN	A@^BANAN	
NA@^BANA	BANANA@^	
ANA@^BAN	NANA@^BA	
NANA@^BA	NA@^BANA	
ANANA@^B	^BANANA@	
BANANA@^	@^BANANA	
^BANANA@		BNN^AA@A

- Reversible

```
function inverseBWT (string s)
    create empty table
    repeat length(s) times
        insert s as a column of table before first column of the table // first insert creates first column
        sort rows of the table alphabetically
    return (row that ends with the 'EOF' character)
```

Last column only suffices to reconstruct **entire matrix**, and thus recover **original string**

Add 1	Sort 1	Add 2	Sort 2	Add 3	Sort 3	Add 4	Sort 4	Add 5	Sort 5	Add 6	Sort 6	Add 7	Sort 7	Add 8	Sort 8
B	A	BA	AN	BAN	ANA	BANA	ANAN	BANAN	ANANA	BANANA	ANANA@	BANANA@	ANANA@^	BANANA@^	ANANA@^B
N	A	NA	AN	NAN	ANA	NANA	ANA@	NANA@	ANA@^	NANA@^	ANA@^B	NANA@^B	ANA@^BA	NANA@^BA	ANA@^BAN
N	A	NA	A@	NA@	A@^	NA@^	A@^B	NA@^B	A@^BA	NA@^BA	A@^BAN	NA@^BAN	A@^BANA	NA@^BANA	A@^BANAN
^	B	^B	BA	^BA	BAN	^BAN	BANA	^BANA	BANAN	^BANAN	BANANA	^BANANA	BANANA@	^BANANA@	BANANA@^
A	N	AN	NA	ANA	NAN	ANAN	NANA	ANANA	NANA@	ANANA@	NANA@^	ANANA@^	NANA@^B	ANANA@^B	NANA@^BA
A	N	AN	NA	ANA	NA@	ANA@	NA@^	ANA@^	NA@^B	ANA@^B	NA@^BA	ANA@^BA	NA@^BAN	ANA@^BAN	NA@^BANA
@	^	@^	^B	@^B	^BA	@^BA	^BAN	@^BAN	^BANA	@^BANA	^BANAN	@^BANAN	^BANANA	@^BANANA	^BANANA@
A	@	A@	@^	A@^	@^B	A@^B	@^BA	A@^BA	@^BAN	A@^BAN	@^BANA	A@^BANA	@^BANAN	A@^BANAN	@^BANANA
last	1st col	pairs	2nd col	triples	3rd col	4mers	4th col	5mers	5th col	6-mers	6th col	7-mers	7th col	8-mers	Full matrix

Searching for an Exact Match

e.g. Searching for OLIS
In MANOLISKELLIS
For simplicity (here):
- only exact matches
- Show entire matrix

In practice: only pointers

Algorithm 3 EXACTMATCH$(P[1,p])$

1: $c \Leftarrow P[p]$ c=last character of query P
2: $sp \Leftarrow C[c]+1$ sp=start pointer
3: $ep \Leftarrow C[c+1]+1$ ep=end pointer
4: $i \Leftarrow p-1$ i=iterate chars from end of query
5: **while** $sp < ep$ and $i \geq 1$ **do**
6: $c \Leftarrow P[i]$ pull next character
7: $sp \Leftarrow C[c] + \mathrm{Occ}(c, sp) + 1$
8: $ep \Leftarrow C[c] + \mathrm{Occ}(c, ep) + 1$
9: $i \Leftarrow i-1$ include earlier character
10: **end while**
11: **return** sp, ep

Last-First (LF) property =
k[th] occurrence of character X in the **last** column is
k[th] occurrence of character X in the **first** column

P is the query substring
C[c] – is how many characters occur before c lexicographically in the genome
Occ(c,sp) is the number of occurrences of character c in the far right column starting at the current position **sp**
[and similarly at current position **ep**]

Letter	Index	Run
$	1	1
A	2	1
E	3	1
I	4	2
L	6	2
K	9	1
M	10	2
N	11	1
O	12	1
S	13	2

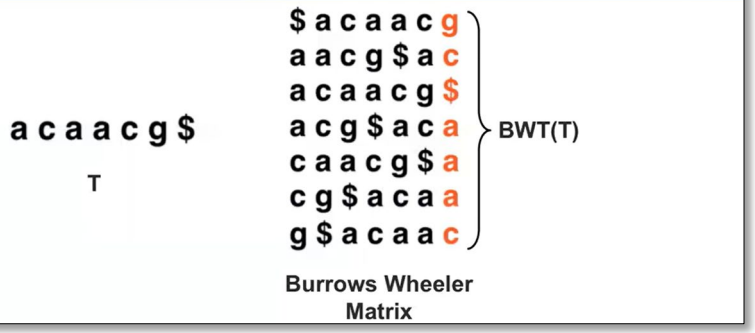
OLIS	OLIS	OLIS	OLIS
1. $MANOLISKELLIS	1. $MANOLISKELLIS	1. $MANOLISKELLIS	1. $MANOLISKELLIS
2. ANOLISKELLIS$M	2. ANOLISKELLIS$M	2. ANOLISKELLIS$M	2. ANOLISKELLIS$M
3. ELLIS$MANOLISK	3. ELLIS$MANOLISK	3. ELLIS$MANOLISK	3. ELLIS$MANOLISK
4. IS$MANOLISKELL	4. IS$MANOLISKELL	4. IS$MANOLISKELL	4. IS$MANOLISKELL
5. ISKELLIS$MANOL	5. ISKELLIS$MANOL	5. ISKELLIS$MANOL	5. ISKELLIS$MANOL
6. LIS$MANOLISKEL	6. LIS$MANOLISKEL	6. LIS$MANOLISKEL	6. LIS$MANOLISKEL
7. LISKELLIS$MANO	7. LISKELLIS$MANO	7. LISKELLIS$MANO	7. LISKELLIS$MANO
8. LLIS$MANOLISKE	8. LLIS$MANOLISKE	8. LLIS$MANOLISKE	8. LLIS$MANOLISKE
9. KELLIS$MANOLIS	9. KELLIS$MANOLIS	9. KELLIS$MANOLIS	9. KELLIS$MANOLIS
10.MANOLISKELLIS$	10.MANOLISKELLIS$	10.MANOLISKELLIS$	10.MANOLISKELLIS$
11.NOLISKELLIS$MA	11.NOLISKELLIS$MA	11.NOLISKELLIS$MA	11.NOLISKELLIS$MA
12.OLISKELLIS$MAN	12.OLISKELLIS$MAN	12.OLISKELLIS$MAN	12.OLISKELLIS$MAN
13.S$MANOLISKELLI	13.S$MANOLISKELLI	13.S$MANOLISKELLI	13.S$MANOLISKELLI
14.SKELLIS$MANOLI	14.SKELLIS$MANOLI	14.SKELLIS$MANOLI	14.SKELLIS$MANOLI

sp=start pointer
ep=end pointer

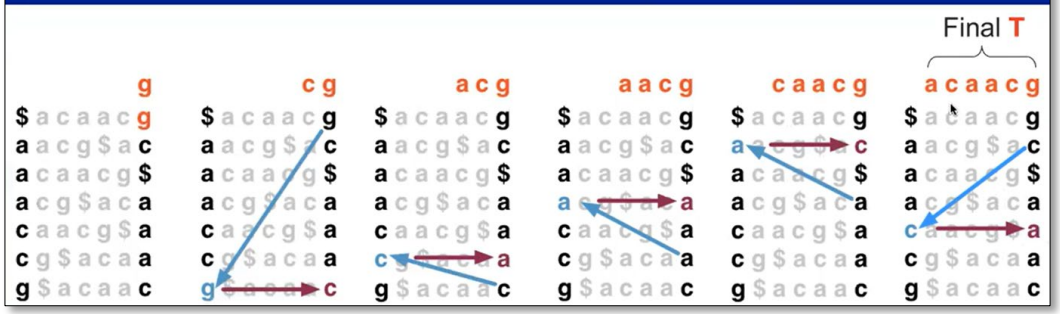
Pseudocode from Langmead et al, 2009. Example by Jason Ernst.

Why BWT search works: entire substrings are sorted

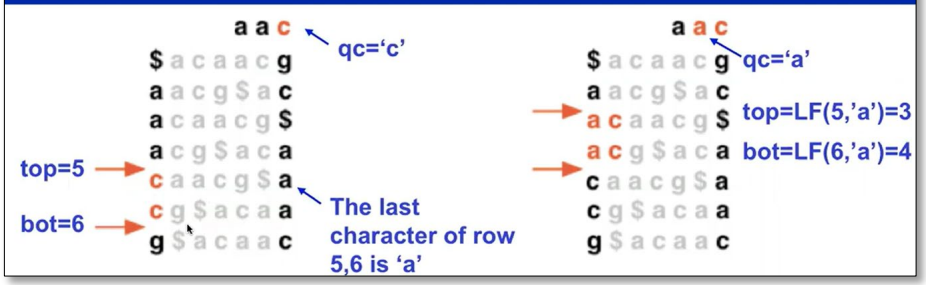
- Property that makes BWT(T) reversible is "LF Mapping".
  - i^{th} occurrence of a character in **L**ast column is the same *text* occurrence as the i^{th} occurrence in **F**irst column.



- To recreate T from BWT(T), repeatedly apply rule:
  T = BWT[LF(i)] + T; i = LF(i)
  - Where LF(i) maps row i to row whose first character corresponds to i's last per LF Mapping.

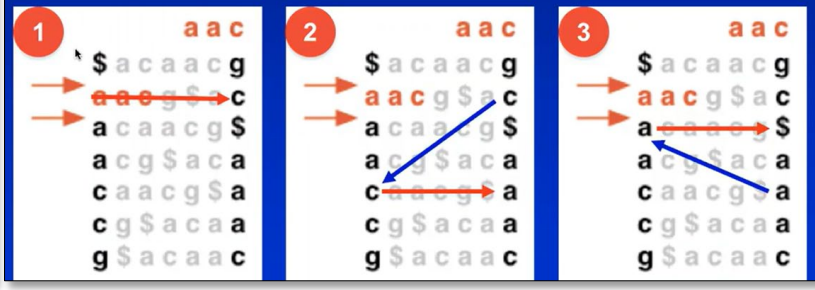


- Query Q = aac / DB T = acaacg$ / BWT(T) = gc$aaac
- To match Q in T using BWT(T), repeatedly apply rule:
  top = LF(top, qc); bot = LF(bot, qc)
  - Where qc is the next character in Q (right-to-left) and LF(i, qc) maps row i to the row whose first character corresponds to i's last character *as if it were qc*.



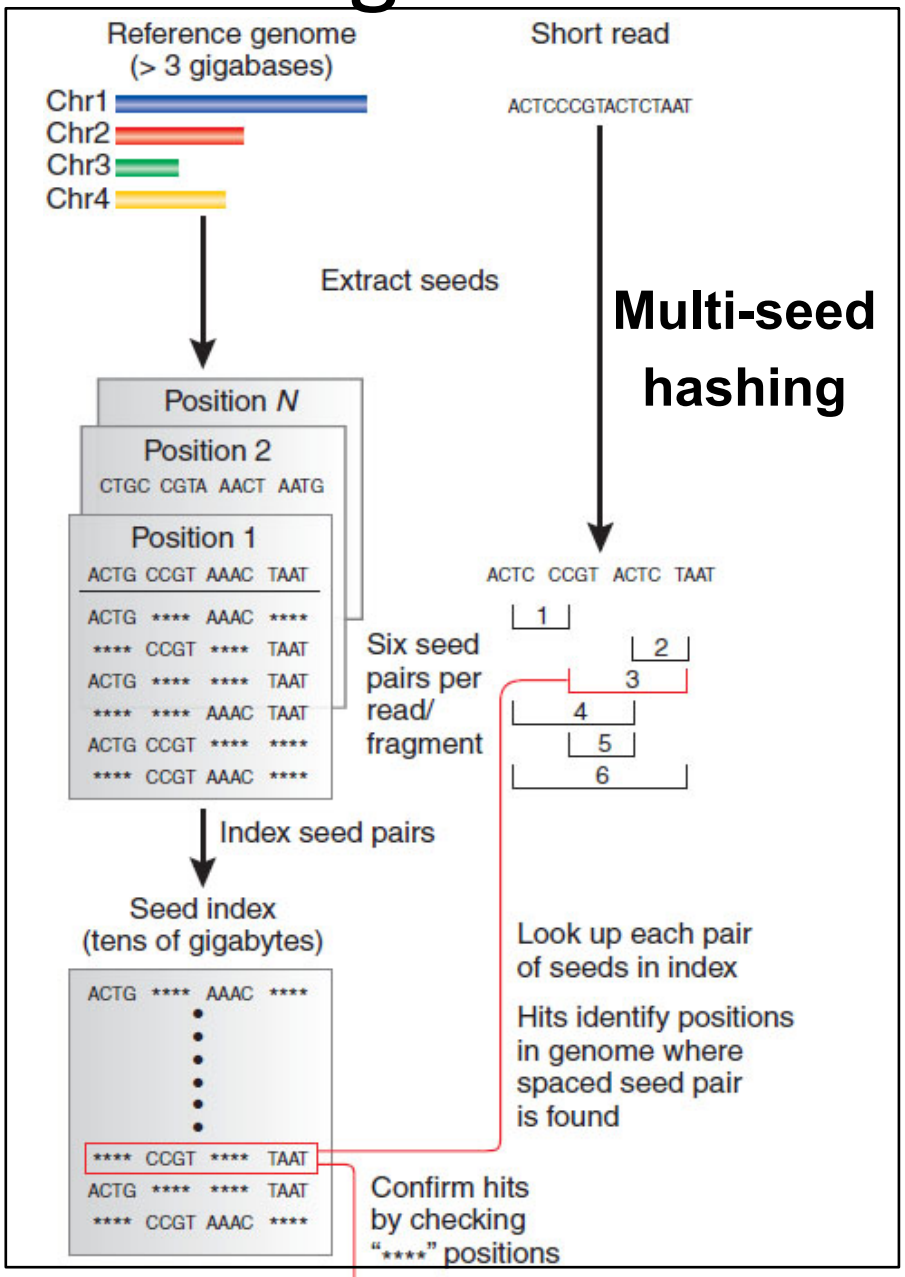
- How to recover the query sequence (Q) alignment **position** in the reference sequence T?
  - LF mapping!

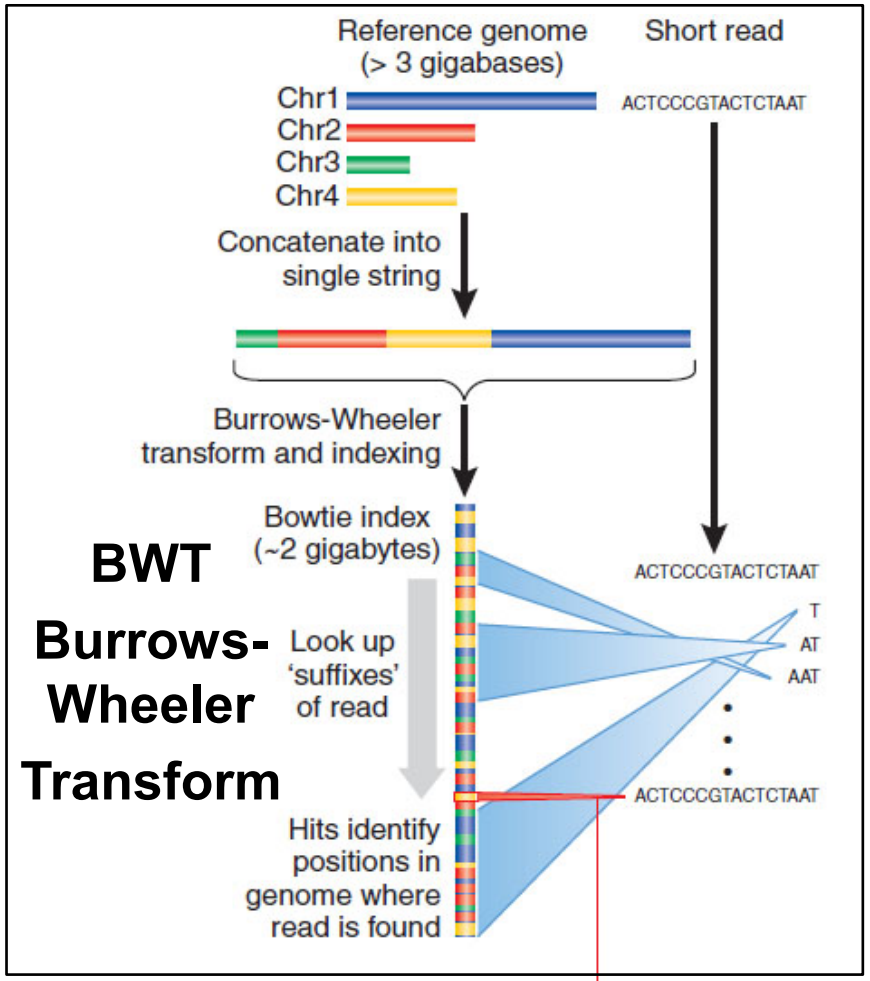
T = acaacg$ (3)
Q = aac



Slides credit: Xiaole Liu

Hashing vs. Burrows Wheeler Transform





Today: How does the BW transform actually work?

Convert each hit back to genome location

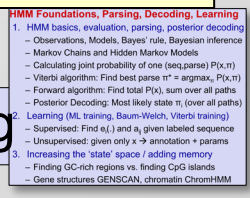
Report alignment to user

Key properties of Burrows-Wheeler Transform

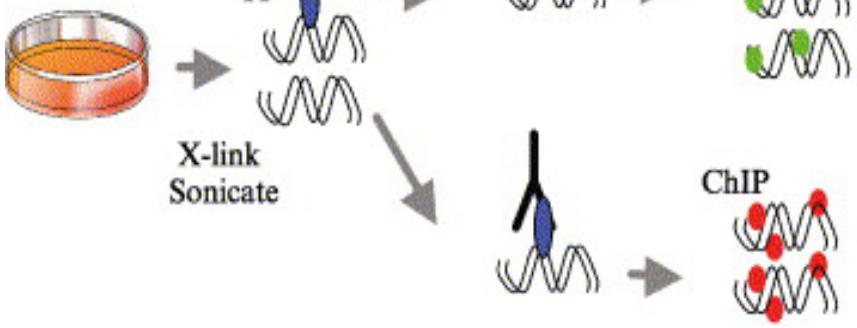
- **Very little memory usage. Same as input (or less)**
  - Don't represent matrix, or strings, just pointers
  - Encode: Simply sort pointers. Decode: follow pointers
- **Original application: string compression (bZip2)**
  - Runs of letters compressed into (letter, runlength) pairs
- **Bioinformatics applications: substring searching**
  - Achieve similar run time as hash tables, suffix trees
  - But: very memory efficient → practical speed gains
- **Mapping 100,000s of reads: only transform once**
  - Pre-process once; read counts in transformed space.
  - Reverse transform once, map counts to genome coords

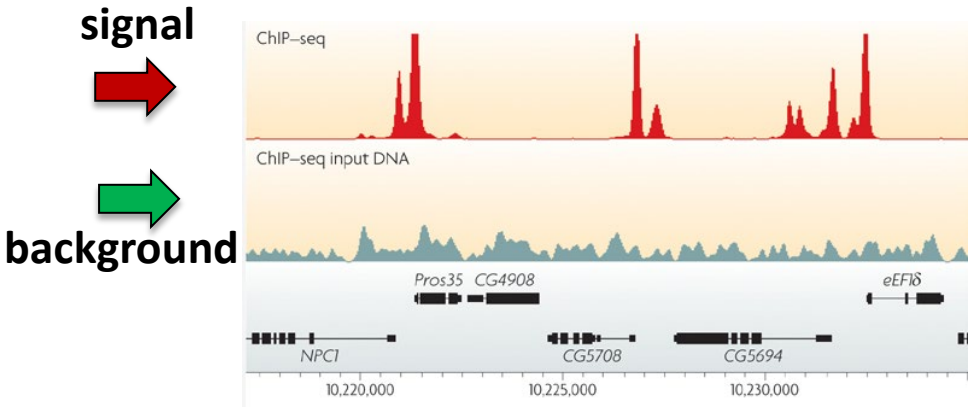
Goals for today: Computational Epigenomics

1. Introduction to Epigenomics
   - Overview of epigenomics, Diversity of Chromatin modifications
   - Antibodies, ChIP-Seq, data generation projects, raw data
2. Primary data processing: Read mapping, Peak calling
   - Read mapping: Hashing, Suffix Trees, Burrows-Wheeler Transform
   - Quality Control, Cross-correlation, Peak calling, IDR (similar to FDR)
3. Discovery and characterization of chromatin states
   - HMM Foundations, Generating, Parsing, Decoding, Learning
   - ChromHMM: Multi-variate HMM for chromatin state learning
4. Model complexity: selecting the number of states/marks
   - Selecting the number of states, selecting number of marks
   - Capturing dependencies and state-conditional mark independence
5. Learning chromatin states jointly across multiple cell types
   - Stacking vs. concatenation approach for joint multi-cell type learning
   - Defining activity profiles for linking enhancer regulatory networks



QC1: Use of input DNA as control dataset



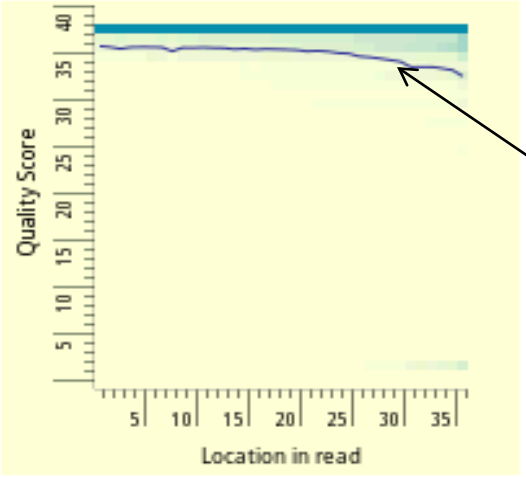


- **Challenge:**
  - Even without antibody: Reads are **not** uniformly scattered
- **Sources of bias in input dataset scatter:**
  - Non-uniform fragmentation of the genome
  - Open chromatin fragmented more easily than closed regions
  - Repetitive sequences over-collapsed in the assembled genome.
- **How to control for these biases:**
  - Remove portion of DNA sample before ChIP step
  - Carry out control experiment without an antibody (input DNA)
  - Fragment input DNA, sequence reads, map, use as background

QC2: Read-level sequencing quality score Q>10

Read quality histograms

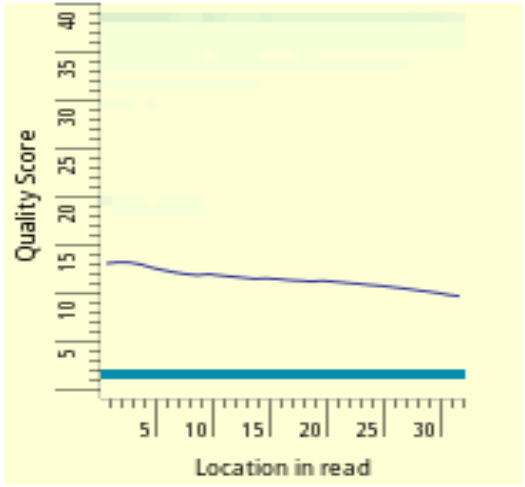
High quality reads



average base score per position

- Each column is a color-coded histogram
- Encodes fraction of all mapped reads that have base score Q (y-axis) at each position (x-axis)
- Darker blue = higher density

Low quality reads



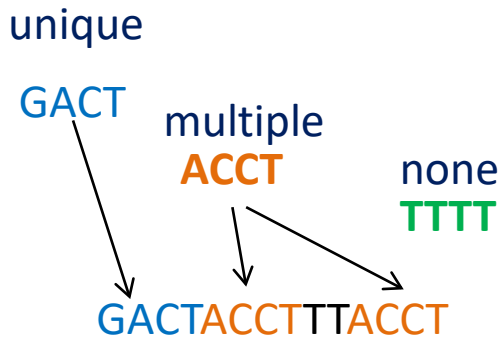
- Read quality tends to drop towards the ends of reads
- Low average per base score implies greater probabilty of mismappings.

- Typically, reject reads whose average score Q < 10

QC3: Fraction of short reads mapped >50%

Reads can map to:

- exactly one location (uniquely mapping)
- multiple locations (repetitive or multi-mapping)
- no locations (unmappable)



Dealing with multiply-mapping reads:

- Conservative approach: do not assign to any location
- Probabilistic approach: assign fractionally to all locations
- Sampling approach: pick one location at random, averages across many reads
- EM approach: map according to density, estimated from unambiguous reads
- Pair-end approach: use paired end read to resolve ambiguities in repeat reads

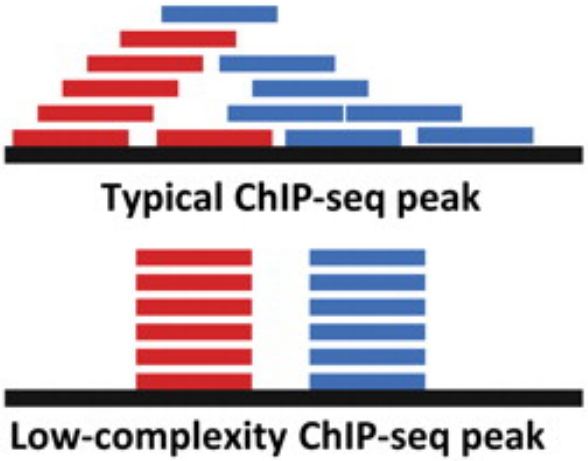
Absence of reads in a region could be due to:

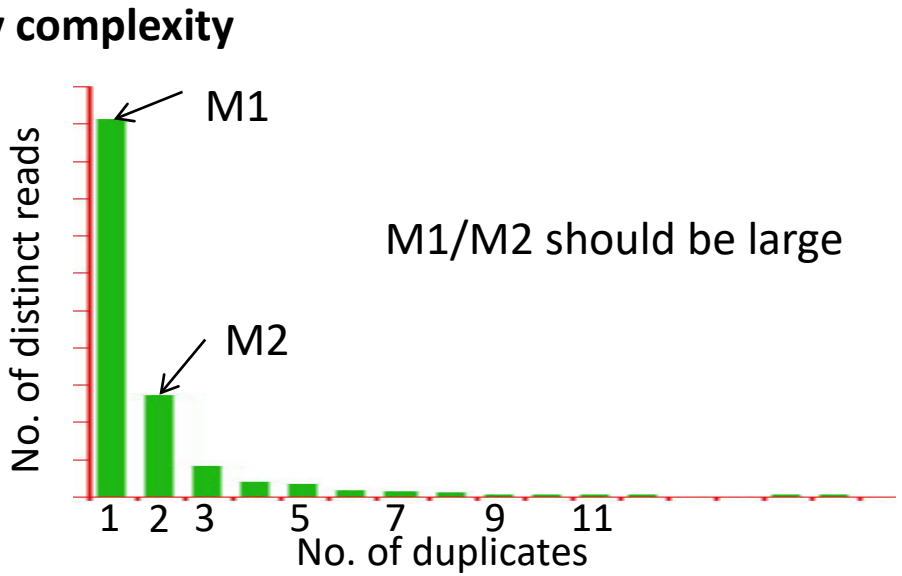
- No assembly coverage in that region (e.g. peri-centromeric region)
- Too many reads mapping to this location (e.g. repetitive element)
- No activity observed in this location (e.g. inactive / quiescent / dead regions)

Dealing with mappability biases:

- ‘Black-listed’ regions, promiscuous across many datasets
- ‘White-listed’ regions, for which at least some dataset has unique reads
- Treat unmappable regions as missing data, distinguish from ‘empty’ regions

QC4: Library complexity: non-redundant fraction





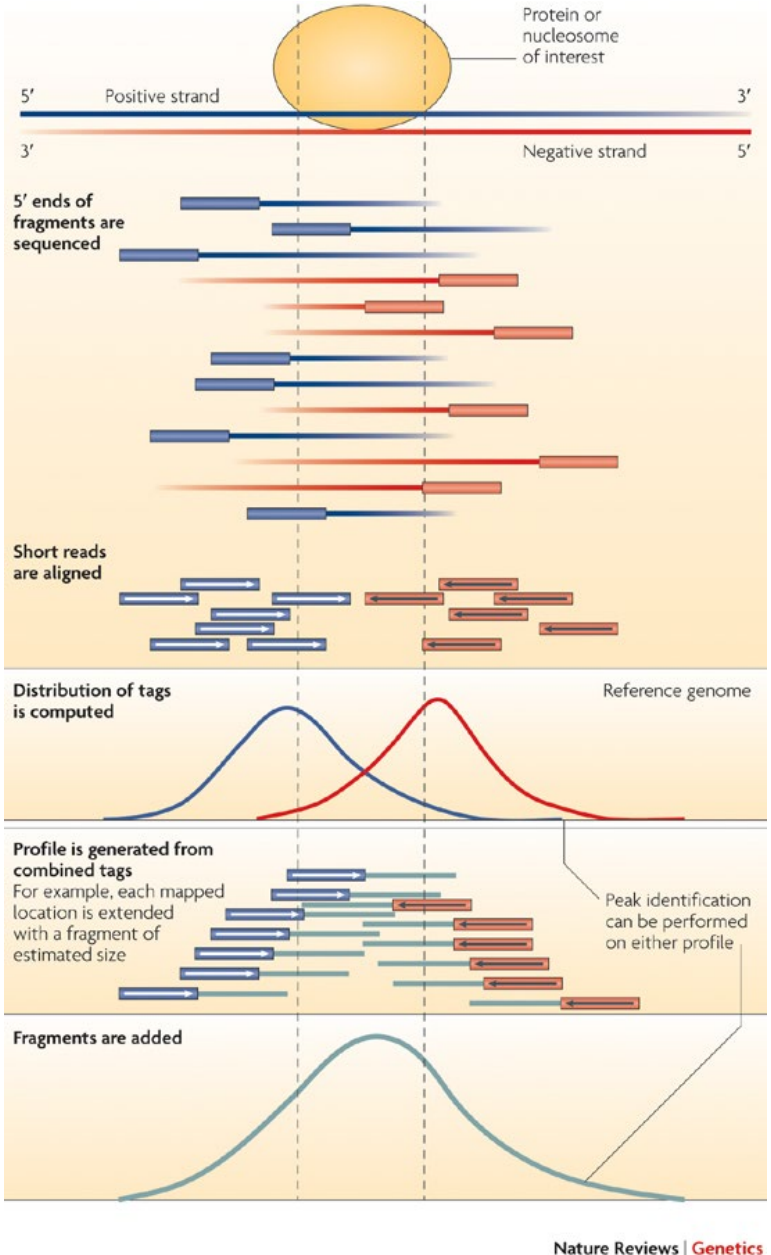
How many distinct uniquely mapping read? How many duplicates?

If your sample does not contain sufficient DNA and/or you over-sequence, you will simply be repeatedly sequencing PCR duplicates of a restricted pool of distinct DNA fragments. This is known a **low-complexity library** and is not desirable.

- **Histogram of no. of duplicates**
- Non-redundant fraction (NRF) = $\frac{\text{No. of 'distinct' unique-mapping reads}}{\text{No. of unique-mapping reads}}$
- NRF should be > 0.8 when 10M < #reads < 80M unique-mapping reads

QC5: exploit ChIP forward vs. reverse reads

(Chromatin immunoprecipitation followed by sequencing)



Multiple IP fragments are obtained corresponding to each binding event

Ends of the fragments are sequenced i.e. "Short-reads/tags"

- Typically ~36 bp, 50 bp, 76 bp or 101 bp

Single-end (SE) sequencing

- Randomly sequence one of the ends of each fragment

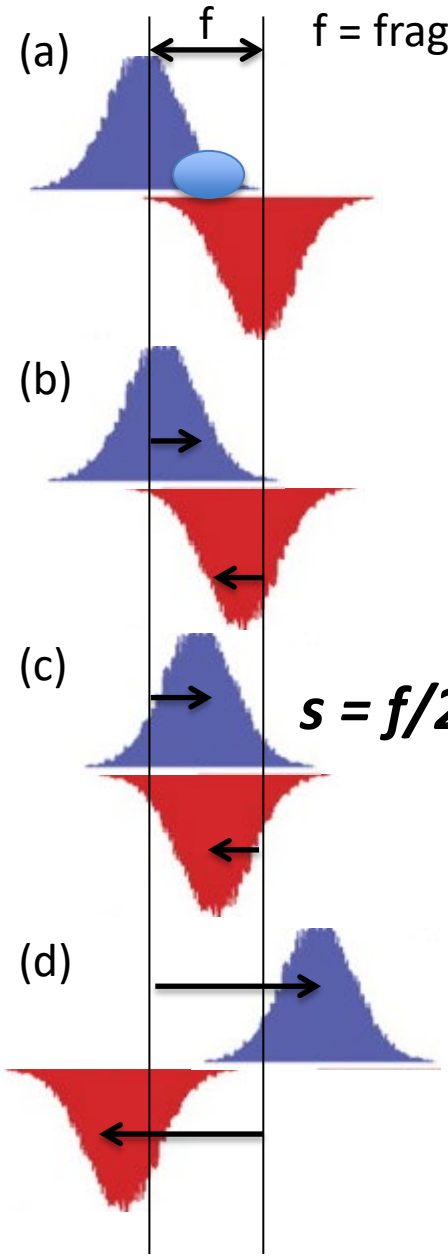
Paired-end (PE) sequencing

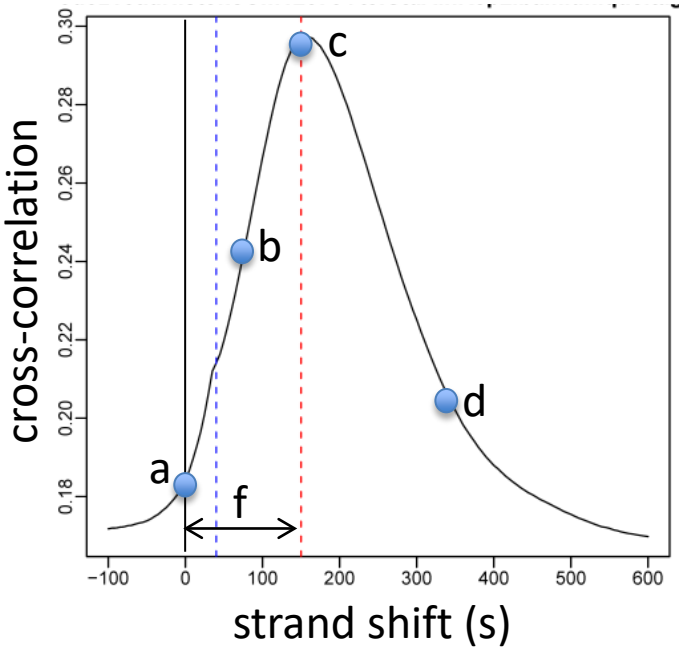
- sequence both ends of each fragment

Canonical "stranded mirror distribution of short-reads" after mapping reads to genome

- Heaps of reads on the **+ strand** and **– strand** separated by a distance ~= fragment length

QC5: Strand cross-correlation (CC) analysis



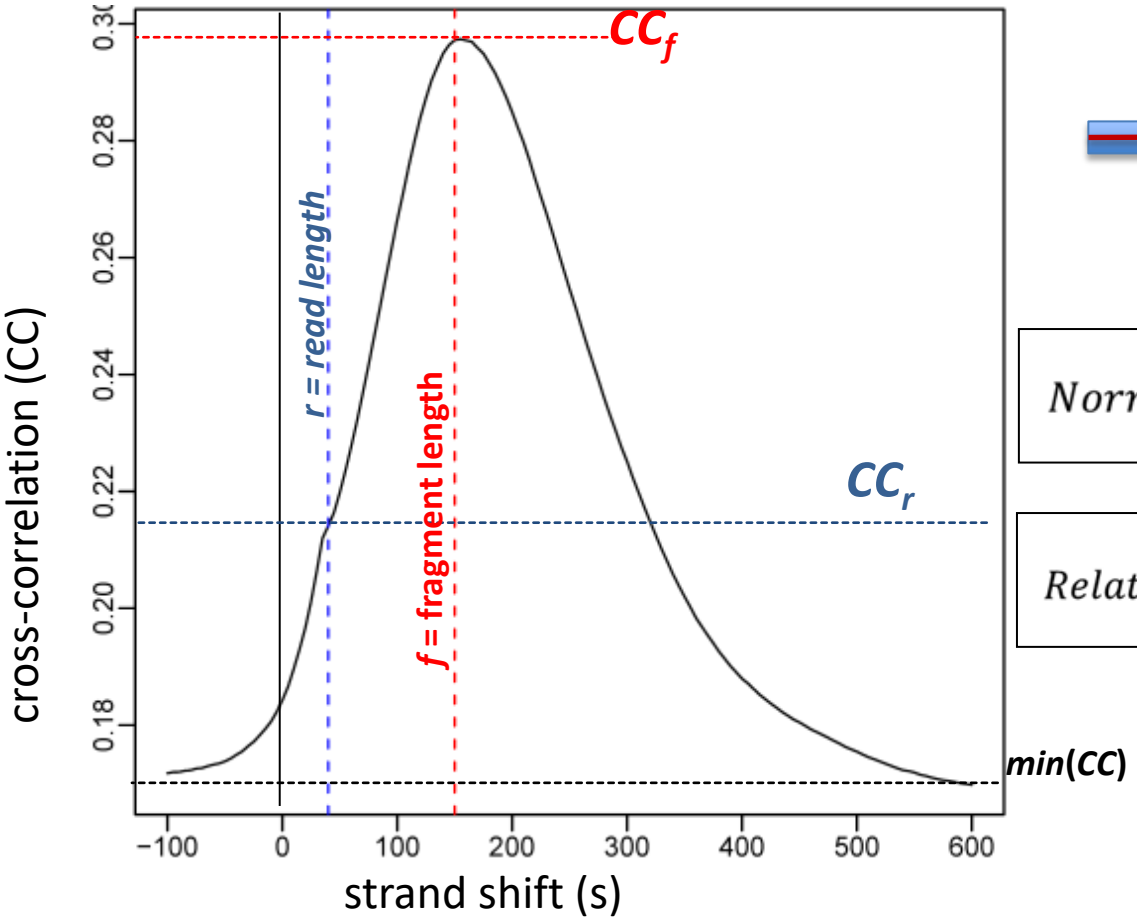


s = f/2 + f/2

1. Calculate forward and reverse strand signals
2. Shift both by specified offset towards each other
3. Calculate correlation of two signals at that shift
4. Correlation peaks at **fragment length** offset ***f***

f is the length at which ChIP DNA is fragmented

Cross-correlation at *read* vs. *fragment* length

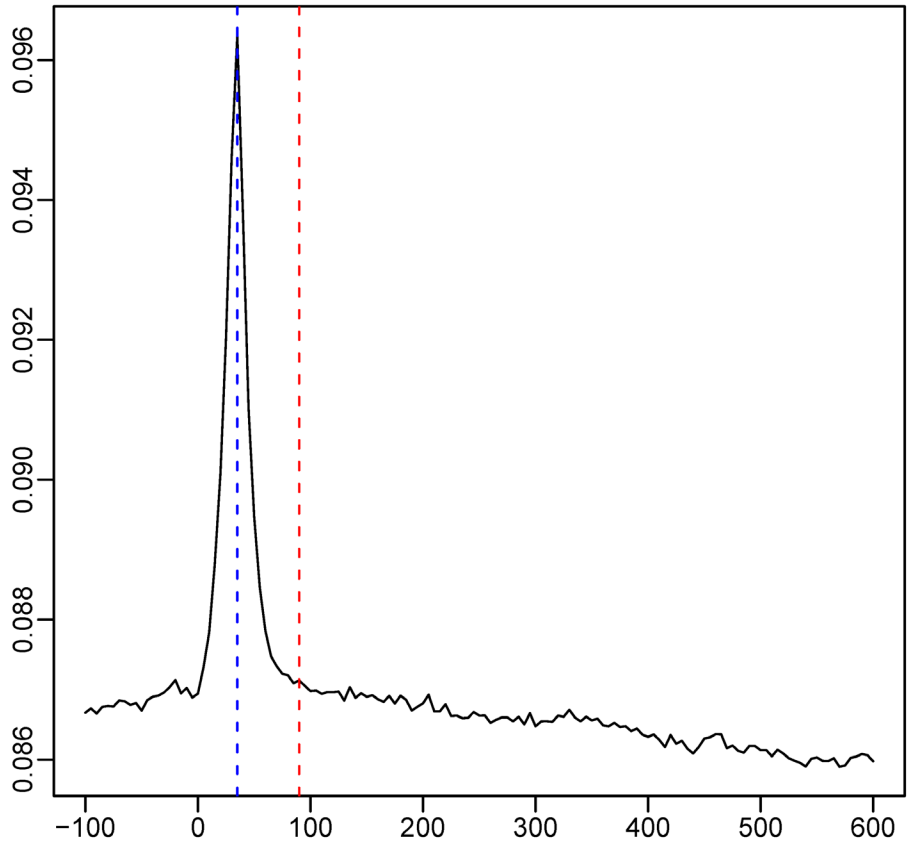


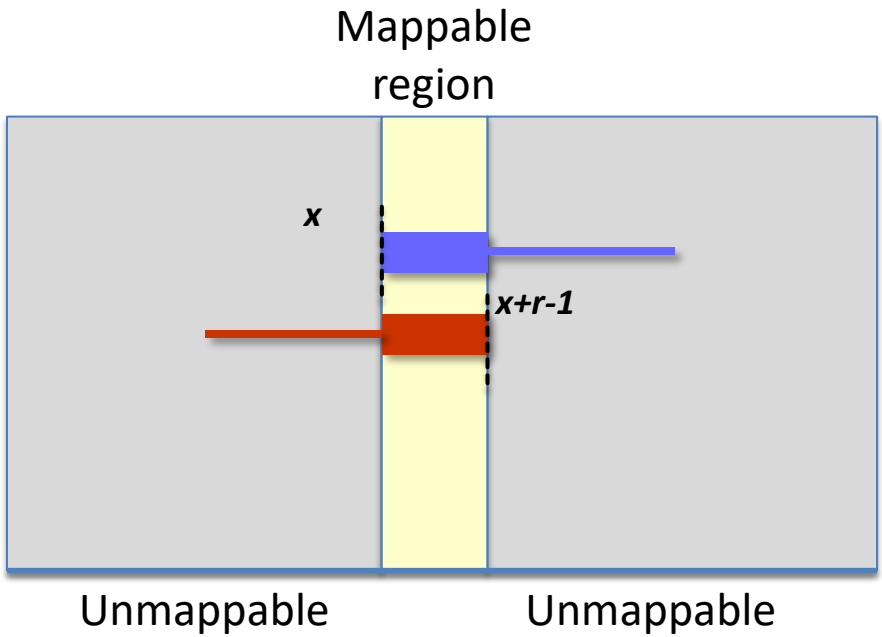
$$Normalized\ strand\ CC(NSC) = \frac{CC_f}{\min(CC)}$$

$$Relative\ strand\ CC(RSC) = \frac{(CC_f - \min(CC))}{(CC_r - \min(CC))}$$

- Sign of a good dataset:
  - High absolute cross-correlation at *fragment* length (NSC)
  - High *fragment* length CC relative to *read* length CC (RSC)

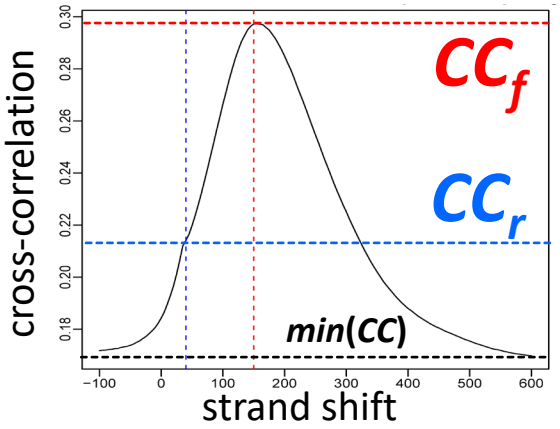
Where does *read* cross-correlation come from?



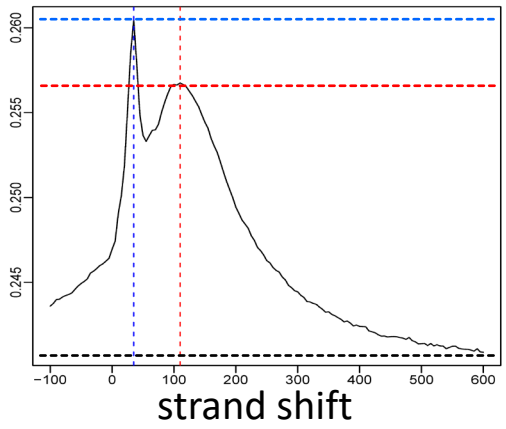


- Input dataset (no ChIP) shows ‘phantom’ peak at ***read*** length only
- Due to read mappability:
  - If position ‘x’ is uniquely mappable on + strand
  - Then position ‘x+r-1’ is uniquely mappable on – strand
- *Fragment*-length peak should always dominate the read-length peak

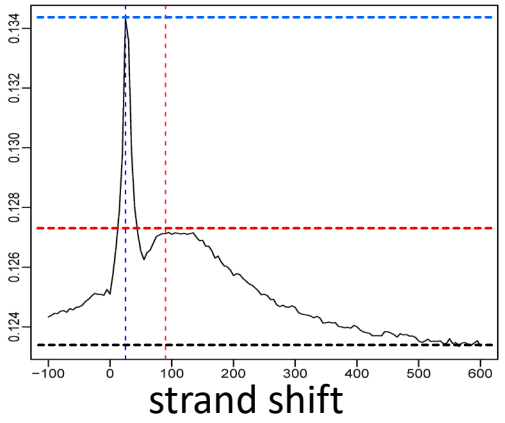
Example of good, medium, bad CC datasets



Highly quality



Medium quality



Low quality

$$Normalized\ strand\ CC(NSC) = \frac{CC_f}{\min(CC)}$$

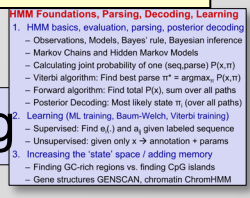
$$Relative\ strand\ CC(RSC) = \frac{(CC_f - \min(CC))}{(CC_r - \min(CC))}$$

For highly enriched datasets, fragment length cross-correlation peak should be able to beat read-length phantom peak

RSC should be > 1

Goals for today: Computational Epigenomics

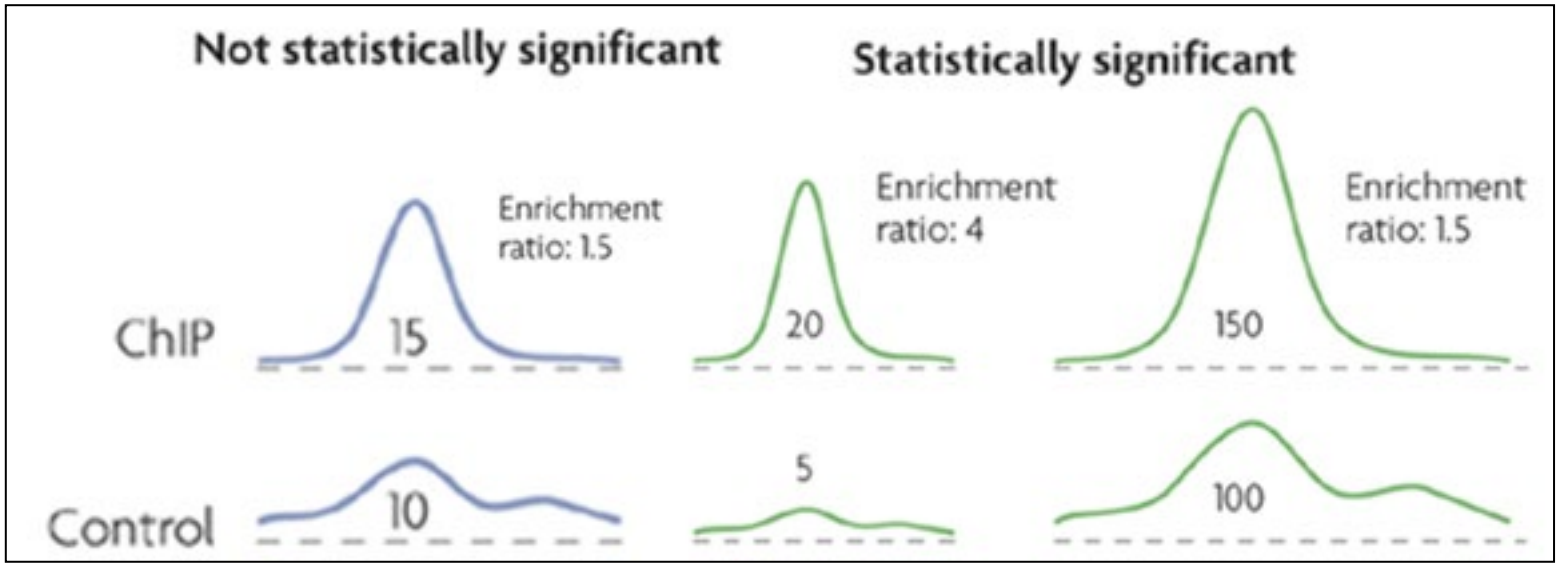
1. Introduction to Epigenomics
   - Overview of epigenomics, Diversity of Chromatin modifications
   - Antibodies, ChIP-Seq, data generation projects, raw data
2. Primary data processing: Read mapping, Peak calling
   - Read mapping: Hashing, Suffix Trees, Burrows-Wheeler Transform
   - Quality Control, Cross-correlation, Peak calling, IDR (similar to FDR)
3. Discovery and characterization of chromatin states
   - HMM Foundations, Generating, Parsing, Decoding, Learning
   - ChromHMM: Multi-variate HMM for chromatin state learning
4. Model complexity: selecting the number of states/marks
   - Selecting the number of states, selecting number of marks
   - Capturing dependencies and state-conditional mark independence
5. Learning chromatin states jointly across multiple cell types
   - Stacking vs. concatenation approach for joint multi-cell type learning
   - Defining activity profiles for linking enhancer regulatory networks



Peak Calling

Continuous signal → Intervals

Peak calling: detect regions of enrichment



Goal: Transform read counts into **normalized intensity signal**

Steps:

1. Estimate fragment-length *f* using strand cross-correlation analysis
2. Extend each read from 5' to 3' direction to fragment length *f*
3. Sum intensity for each base in 'extended reads' from both strands
4. Perform same operation on input-DNA control data (correct for sequencing depth differences)
5. Calculate enrichment ratio value for every position in the genome

Result: Enrichment fold difference for ChIP / control signal

Peak calling: identify discrete intervals

Program	Reference	Version	Graphical user interface?	Window-based scan	Tag clustering	Gaussian kernel density estimator	Strand-specific scoring	Peak height or fold enrichment (FE)	Background subtraction	Compensates for genomic duplications or deletions	False Discovery Rate	Compare to normalized control data (FE)	Compare to statistical model fitted with control data	Statistical model or test
CisGenome	28	1.1	X*	X				X	X		X		X	conditional binomial model
Minimal ChipSeq Peak Finder	16	2.0.1			X			X				X		
E-RANGE	27	3.1			X			X				X	X	chromsome scale Poisson dist.
MACS	13	1.3.5		X				X			X		X	local Poisson dist.
QuEST	14	2.3				X		X			X**		X	chromsome scale Poisson dist.
HPeak	29	1.1		X				X					X	Hidden Markov Model
Sole-Search	23	1	X	X				X		X			X	One sample t-test
PeakSeq	21	1.01			X			X					X	conditional binomial model
SISSRS	32	1.4		X			X					X		
spp package (wtd & mtc)	31	1.7		X			X		X	X'	X			
				Generating density profiles			Peak assignment		Adjustments w. control data		Significance relative to control data			

X* = Windows-only GUI or cross-platform command line interface

X** = optional if sufficient data is available to split control data

X' = method exludes putative duplicated regions, no treatment of deletions

Wilbanks EG, Facciotti MT (2010) Evaluation of Algorithm Performance in ChIP-Seq Peak Detection. PLoS ONE 5(7): e11471. doi:10.1371/journal.pone.0011471
http://www.plosone.org/article/info:doi/10.1371/journal.pone.0011471

Peak calling thresholds

Poisson p-value thresholds

- Read count model: Locally-adjusted' Poisson distribution

$$P(count = x) = \frac{\lambda_{local}^{x} \exp(-\lambda_{local})}{x!}$$

- $\lambda_{local} = \max(\lambda_{BG}, [\lambda_{1k},] \lambda_{5k}, \lambda_{10k})$ estimated from control data
  - Poisson *p*-value = $P(count \geq x)$
  - *q*-value : Multiple hypothesis correction

Peaks: Genomic locations that pass a user-defined *p*-value (e.g. 1e-5) or *q*-value (e.g. 0.01) threshold

Empirical False discovery rates

- Swap ChIP and input-DNA tracks
  - Recompute*p*-values
- At each *p*-value, eFDR = Number of control peaks / Number of ChIPpeaks
  - Use an FDR threshold to call peaks

Issues with peak calling thresholds

Cannot set a universal threshold for empirical FDRs and p-values

- Depends on ChIP and input sequencing depth
- Depends on binding ubiquity of factor
- Stronger antibodies get an advantage

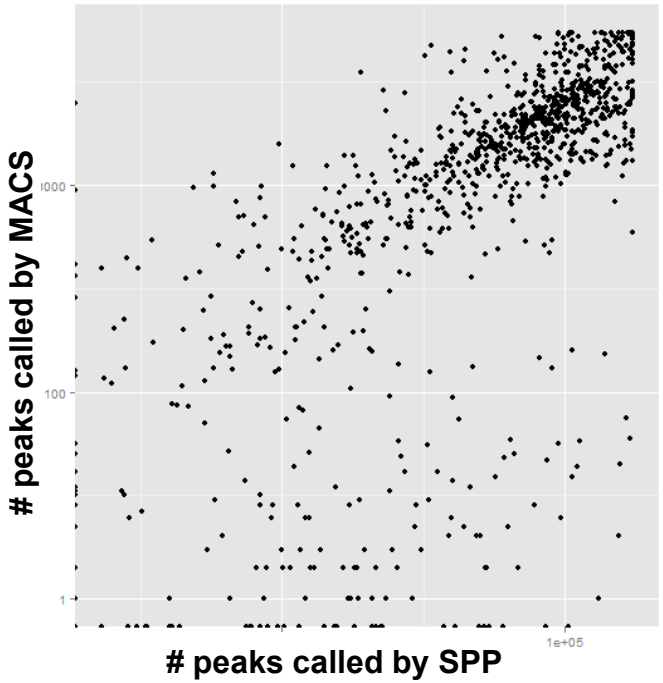
FDRs quite unstable

- Small changes in threshold => massive changes in peak numbers

Difficult to compare results across peak callers with a fixed threshold

- Different methods to compute eFDR or q-values

(at FDR = 1% cutoff)



Select ‘real’ peaks w/ replicates

Use peak ranks in replicate experiments

IDR: Irreproducible Discovery Rate

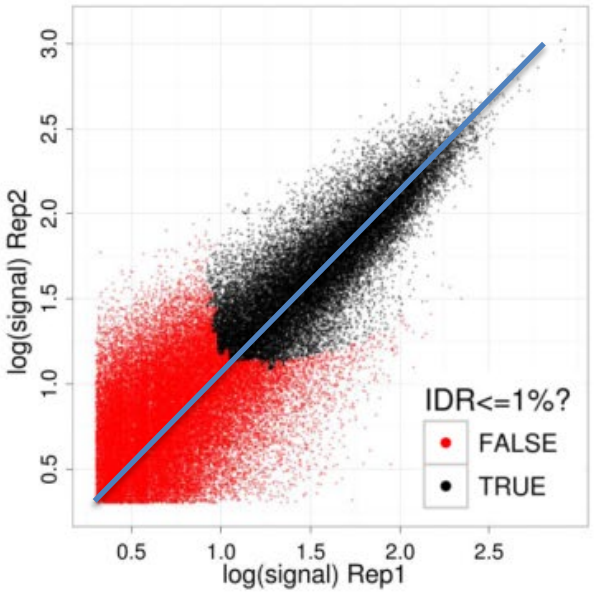
http://anshul.kundaje.net/projects/idr

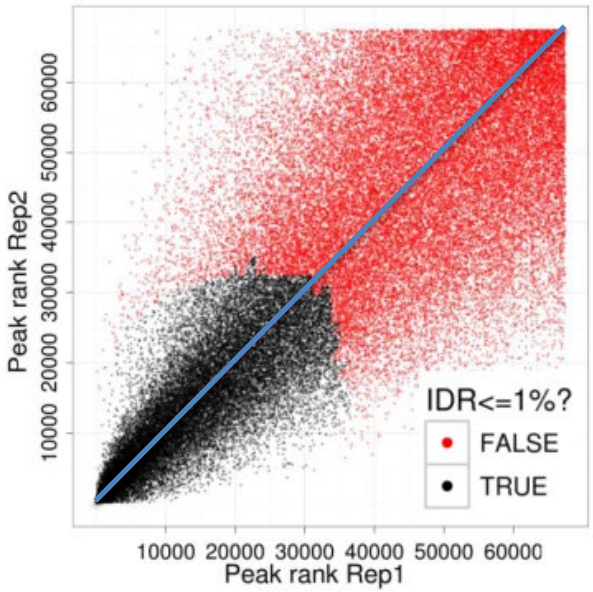
A. Kundaje, Q. Li , B. Brown, J. Rozowsky, S. Wilder, M. Gerstein, I. Dunham, E. Birney, P. Bicke

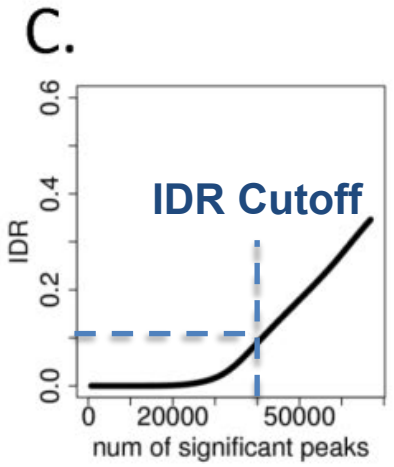
How to combine two replicates

- Challenge:
  - Replicates show small differences in peak heights
  - Many peaks in common, but many are unique
- Problem with simple solutions:
  - Union: too lenient, keeps garbage from both
  - Intersection: too stringent, throws away good peaks
  - Sum: does not exploit independence of two datasets

IDR idea: Exploit peak rank similarity in replicates







- Key idea: True peaks will be highly ranked in both replicates
  - Keep going down rank list, until ranks are no longer correlated
    - This cutoff could be different for the two replicates
  - The actual peaks included may differ between replicates
    - Adaptively learn optimal peak calling threshold
  - **F**DR threshold of 10% ➔ 10% of peaks are false (widely used)
- **I**DR threshold of 10% ➔ 10% of peaks are not reproducible

The IDR model: A two component mixture model

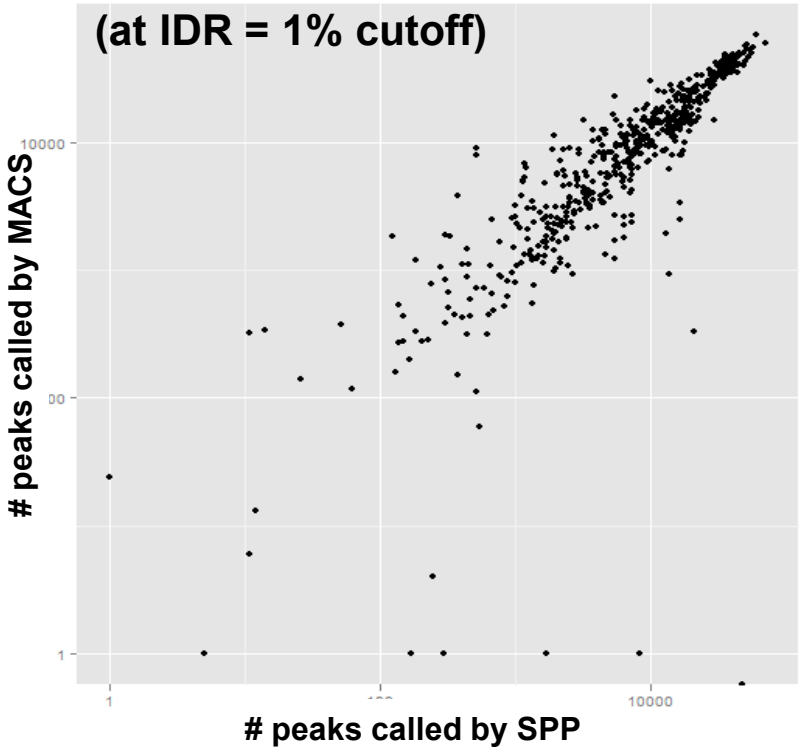
- Looking only at ranks means that the marginals are uniform, so all the information is encoded in the joint distribution.
- Model the joint distribution of ranks as though it came from a two component Gaussian mixture model:

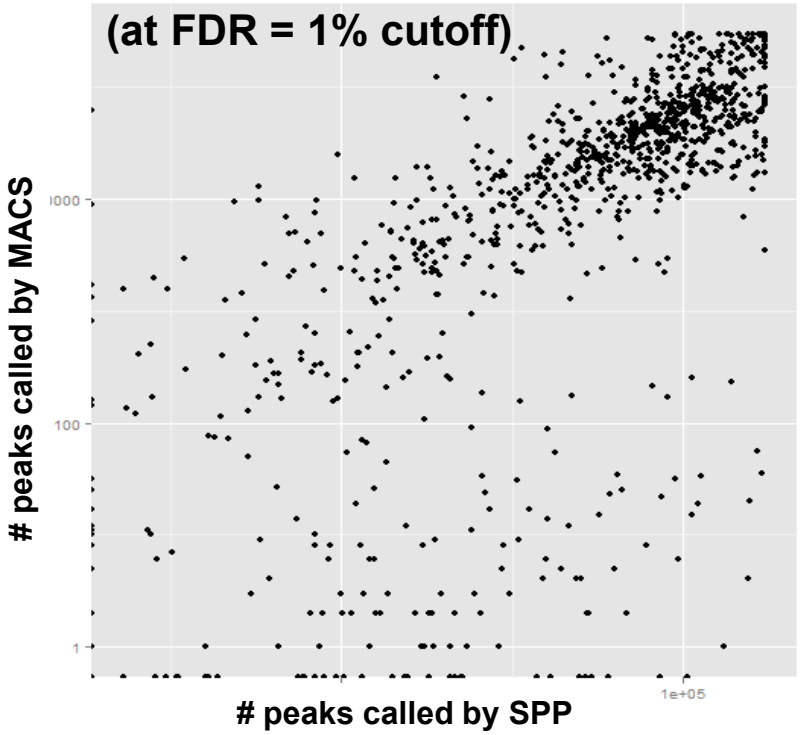
$$(x, y) \sim pN(\mu, \mu, \sigma, \sigma, \rho) + (1 - p)N(0,0,1,1,0)$$

- This can be fit via an EM-like algorithm.

IDR leads to higher consistence between peak callers

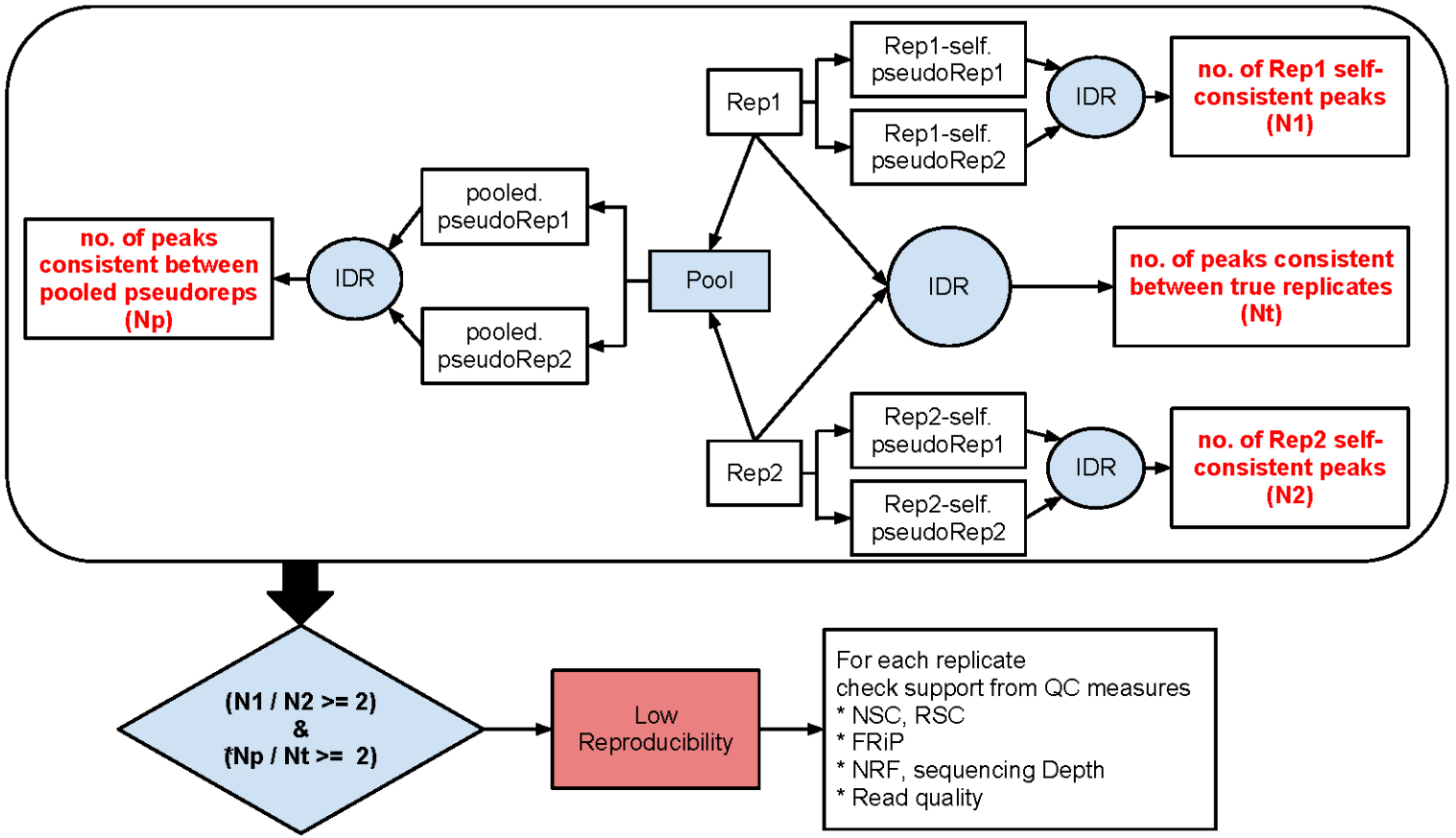
IDR = Irreproducible Discovery Rate FDR = False Discovery Rate





- Compare number of peaks found by two different peak callers
- IDR thresholds are far more robust and comparable than FDR
- FDR only relies on enrichment over input, IDR exploits replicates

What if we don't have good replicates?

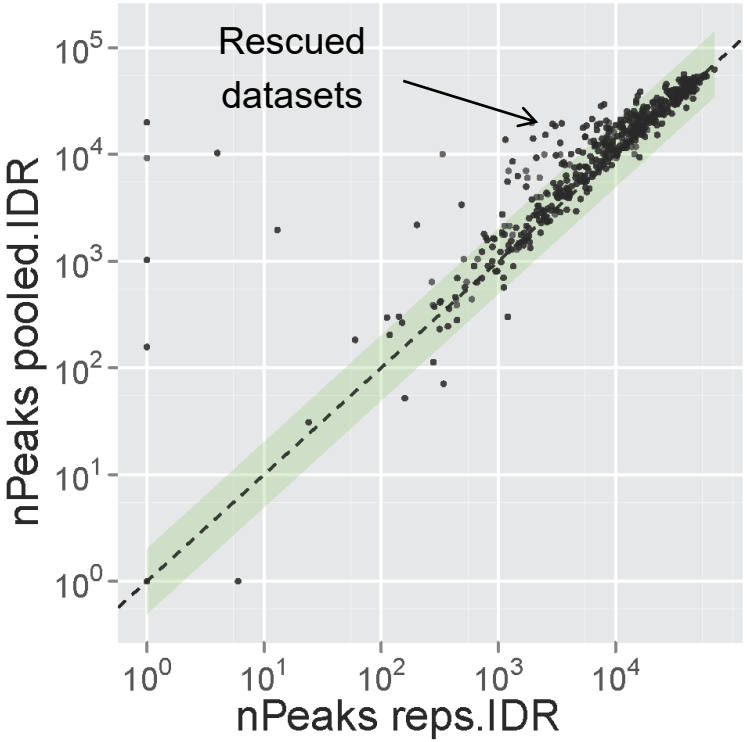


- IDR pipeline uses replicates when they are available
- IDR pipeline also evaluates each replicate individually
  - Pooling strategy to generate pseudo-replicates

➔ Can pin-point 'bad' replicates that may lead to low reproducibility

➔ Can estimate IDR thresholds when replicates are not available

Only one good replicate: Pseudo-replicates



- IDR pipeline can be used to rescue datasets with only one good replicate (using pseudo-replicates)
- IDR pipeline can also be used to call optimal thresholds on a dataset with a single replicate (e.g. when there isn't enough material to perform multiple reps)

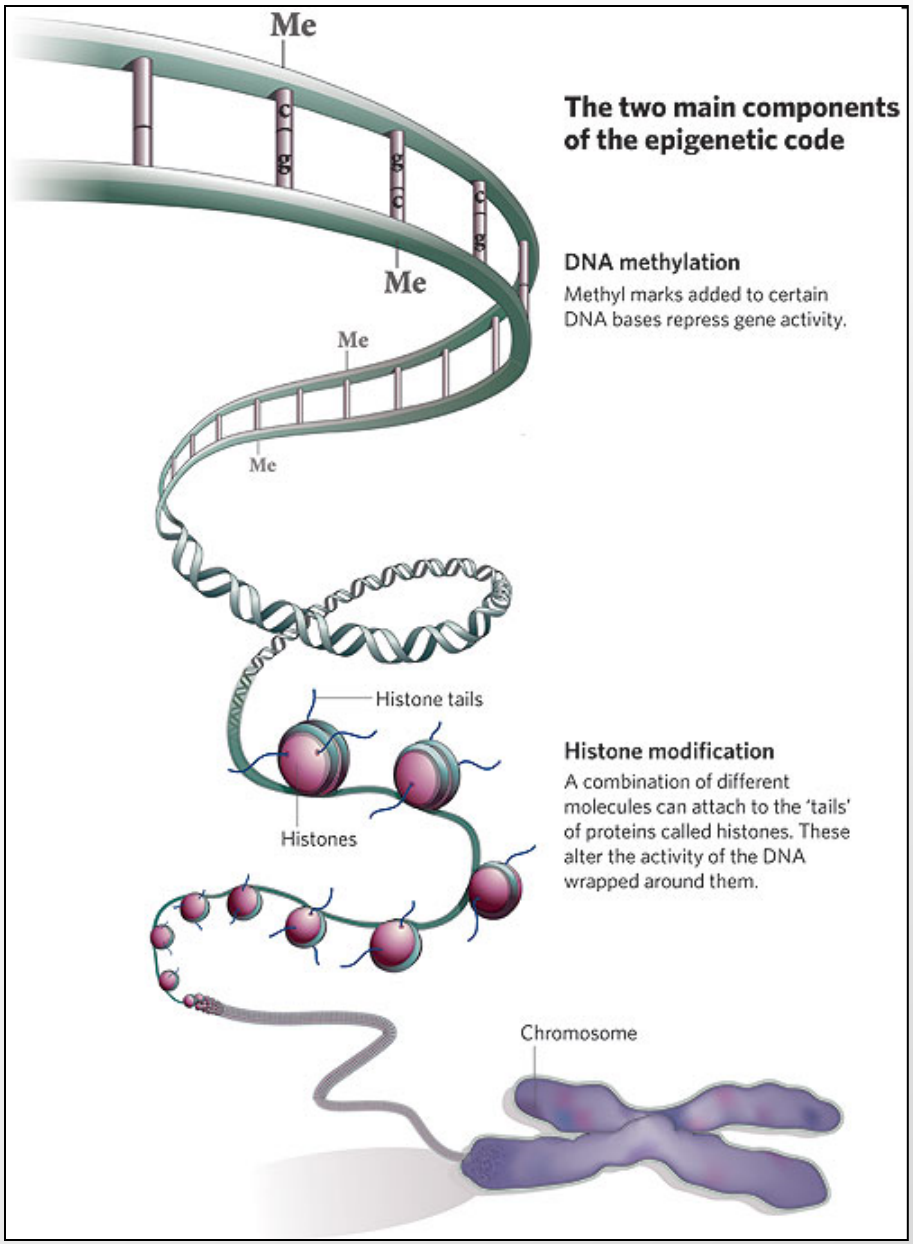
Goals for today: Computational Epigenomics

1. Introduction to Epigenomics
   - Overview of epigenomics, Diversity of Chromatin modifications
   - Antibodies, ChIP-Seq, data generation projects, raw data
2. Primary data processing: Read mapping, Peak calling
   - Read mapping: Hashing, Suffix Trees, Burrows-Wheeler Transform
   - Quality Control, Cross-correlation, Peak calling, IDR (similar to FDR)
3. Discovery and characterization of chromatin states
   - HMM Foundations, Generating, Parsing, Decoding, Learning
   - ChromHMM: Multi-variate HMM for chromatin state learning
4. Model complexity: selecting the number of states/marks
   - Selecting the number of states, selecting number of marks
   - Capturing dependencies and state-conditional mark independence
5. Model complexity: selecting the number of states/marks
   - Selecting the number of states, selecting number of marks
   - Capturing dependencies and state-conditional mark independence
6. Learning chromatin states jointly across multiple cell types
   - Stacking vs. concatenation approach for joint multi-cell type learning
   - Defining activity profiles for linking enhancer regulatory networks

HMM Foundations, Parsing, Decoding, Learning

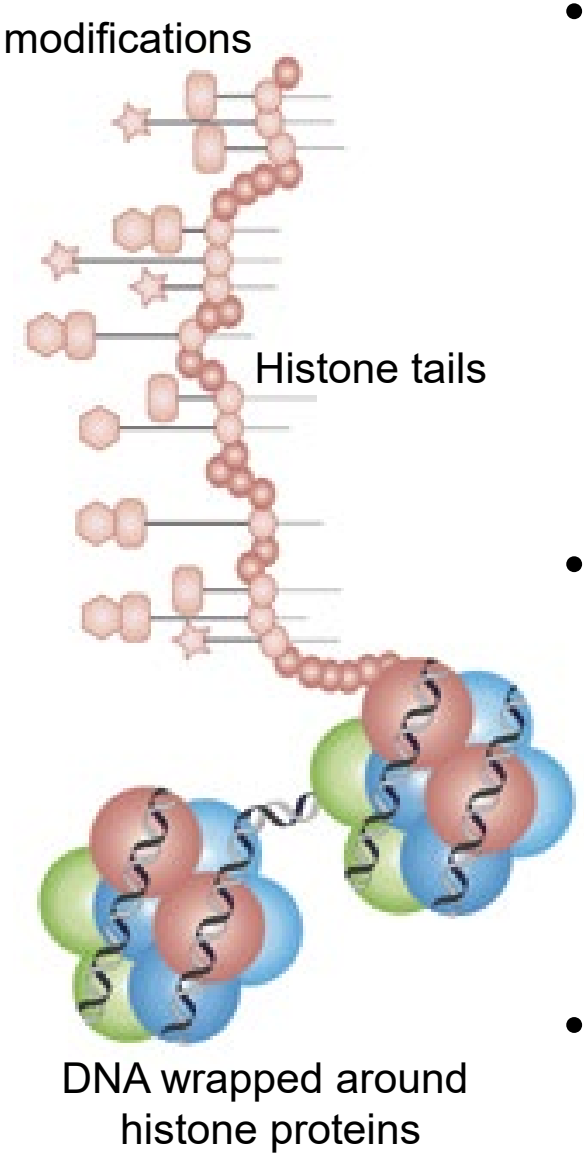
1. HMM basics, evaluation, parsing, posterior decoding
   - Observations, Models, Bayes' rule, Bayesian inference
   - Markov Chains and Hidden Markov Models
   - Calculating joint probability of one (seq,parse) $P(x,\pi)$
   - Viterbi algorithm: Find best parse $\pi^* = \text{argmax}_\pi P(x,\pi)$
   - Forward algorithm: Find total $P(x)$, sum over all paths
   - Posterior Decoding: Most likely state π_i (over all paths)
2. Learning (ML training, Baum-Welch, Viterbi training)
   - Supervised: Find $e_i(.)$ and a_{ij} given labeled sequence
   - Unsupervised: given only x → annotation + params
3. Increasing the 'state' space / adding memory
   - Finding GC-rich regions vs. finding CpG islands
   - Gene structures GENSCAN, chromatin ChromHMM

Chromatin signatures for genome annotation



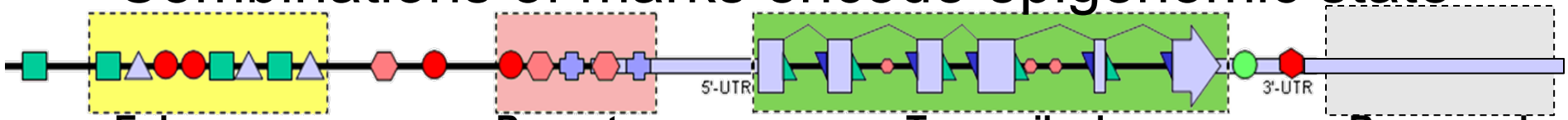
- **Challenges**
  - Dozens of marks
  - Complex combinatorics
  - Diversity and dynamics
- **Histone code hypothesis**
  - Distinct function for distinct combinations of marks?
  - Both additive and combinatorial effects
- **How do we find biologically relevant ones?**
  - Unsupervised approach
  - Probabilistic model
  - Explicit combinatorics

100s of histone tail modifications



- 100+ different histone modifications
  - Histone protein → H3/H4/H2A/H2B
  - AA residue → Lysine4(K4)/K36…
  - Chemical modification → Met/Pho/Ubi
  - Number → Me-Me-Me(me3)
  - Shorthand: H3K4me3, H2BK5ac
- In addition:
  - DNA modifications
  - Methyl-C in CpG / Methyl-Adenosine
  - Nucleosome positioning
  - DNA accessibility
- The constant struggle of gene regulation
  - TF/histone/nucleo/GFs/Chrom compete

Combinations of marks encode epigenomic state

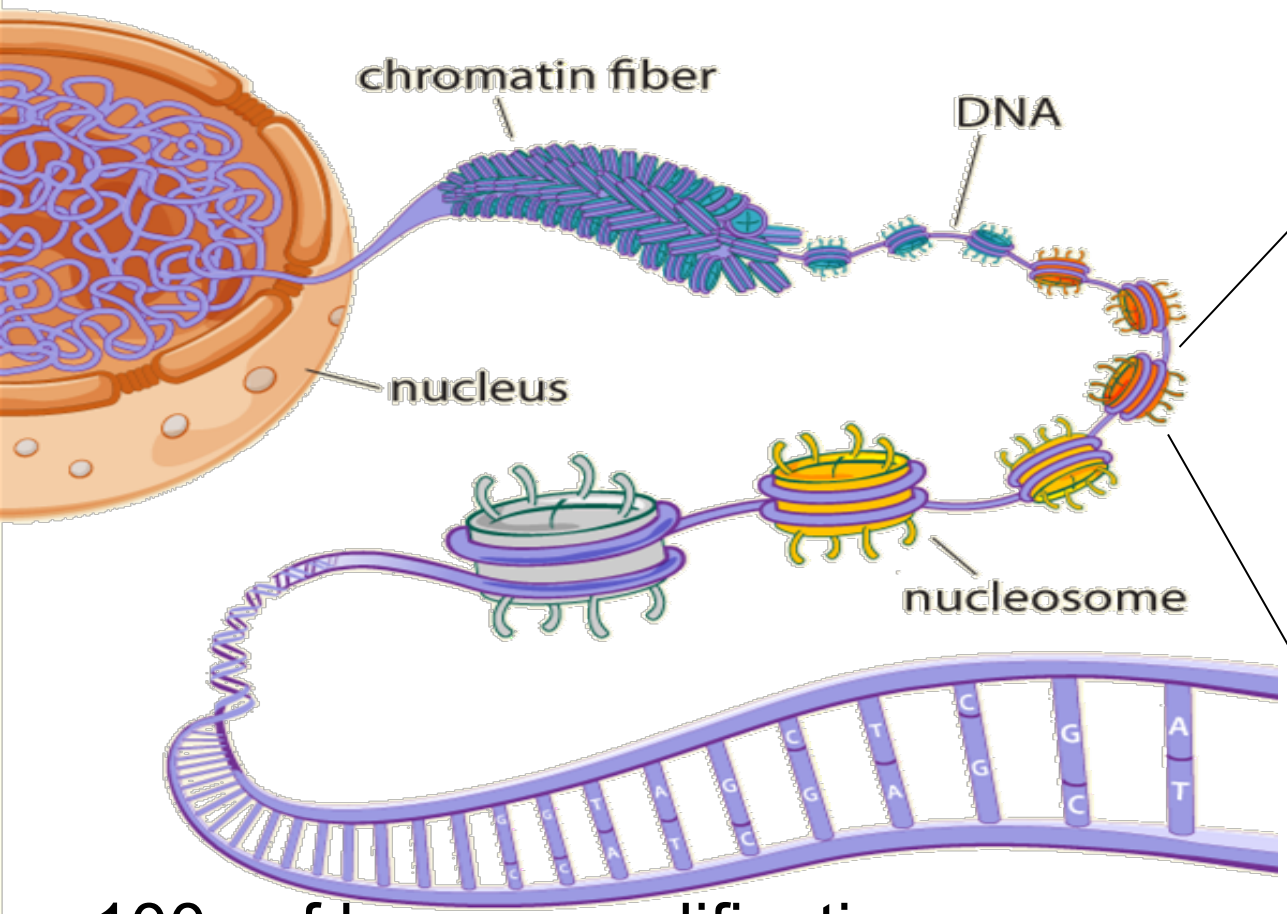


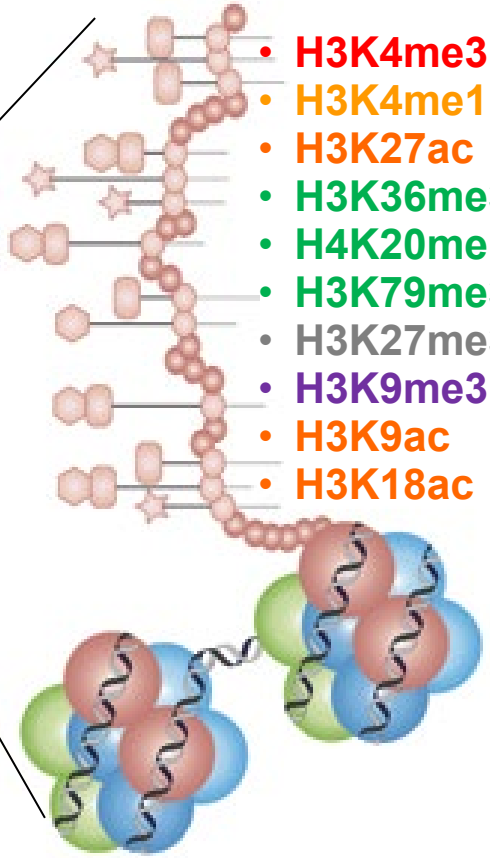
Enhancers
- H3K4me1
- H3K27ac
- DNase

Promoters
- H3K4me3
- H3K9ac
- DNase

Transcribed
- H3K36me3
- H3K79me2
- H4K20me1

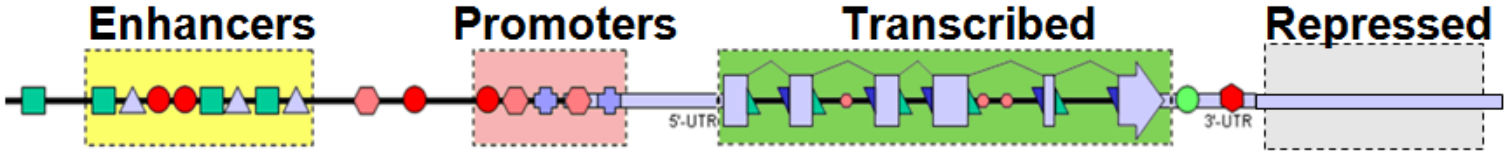
Repressed
- H3K9me3
- H3K27me3
- DNAmethyl





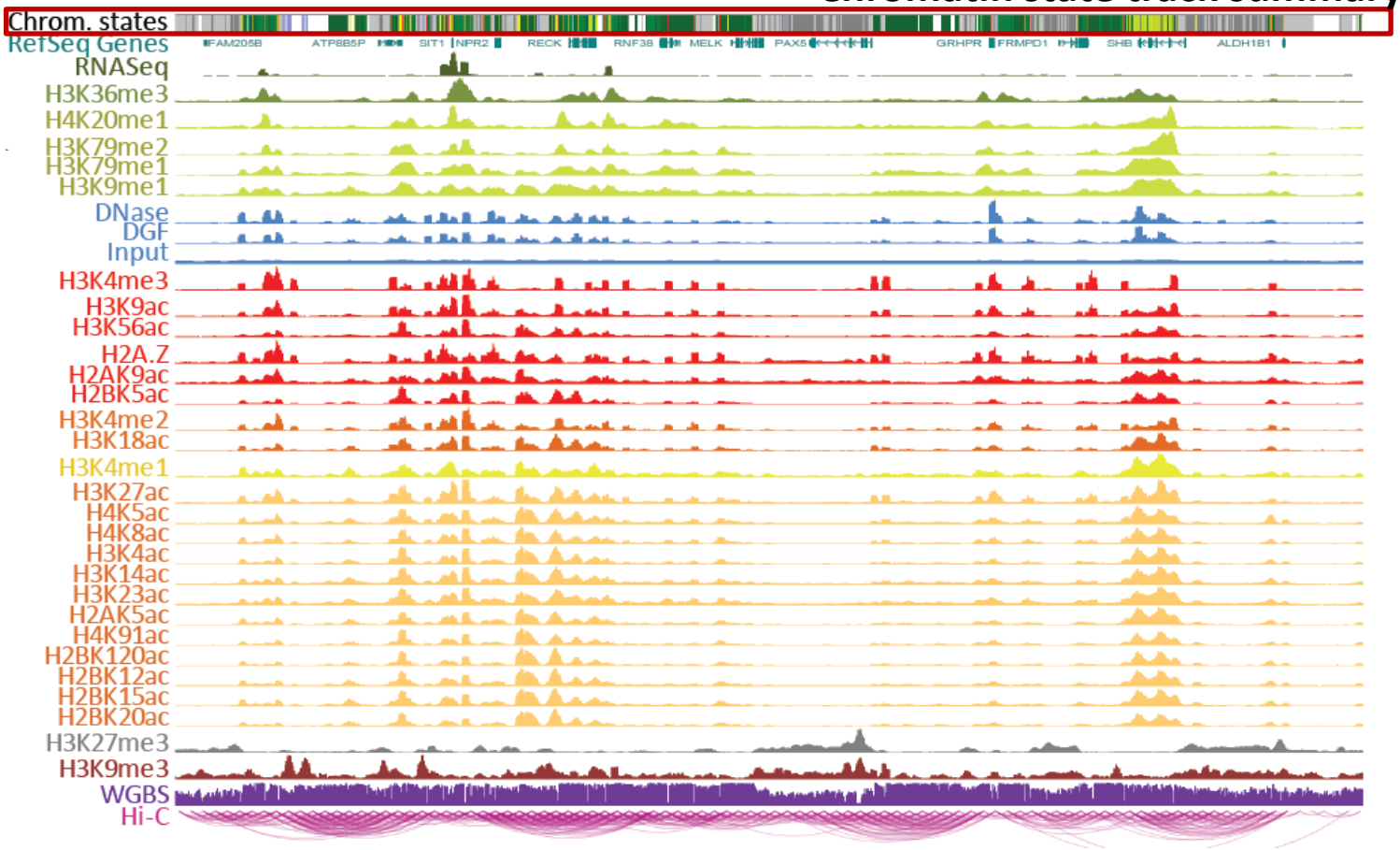
- 100s of known modifications, many new still emerging
- Systematic mapping using ChIP-, Bisulfite-, DNase-Seq

Summarize multiple marks into chromatin states



Chromatin state track summary

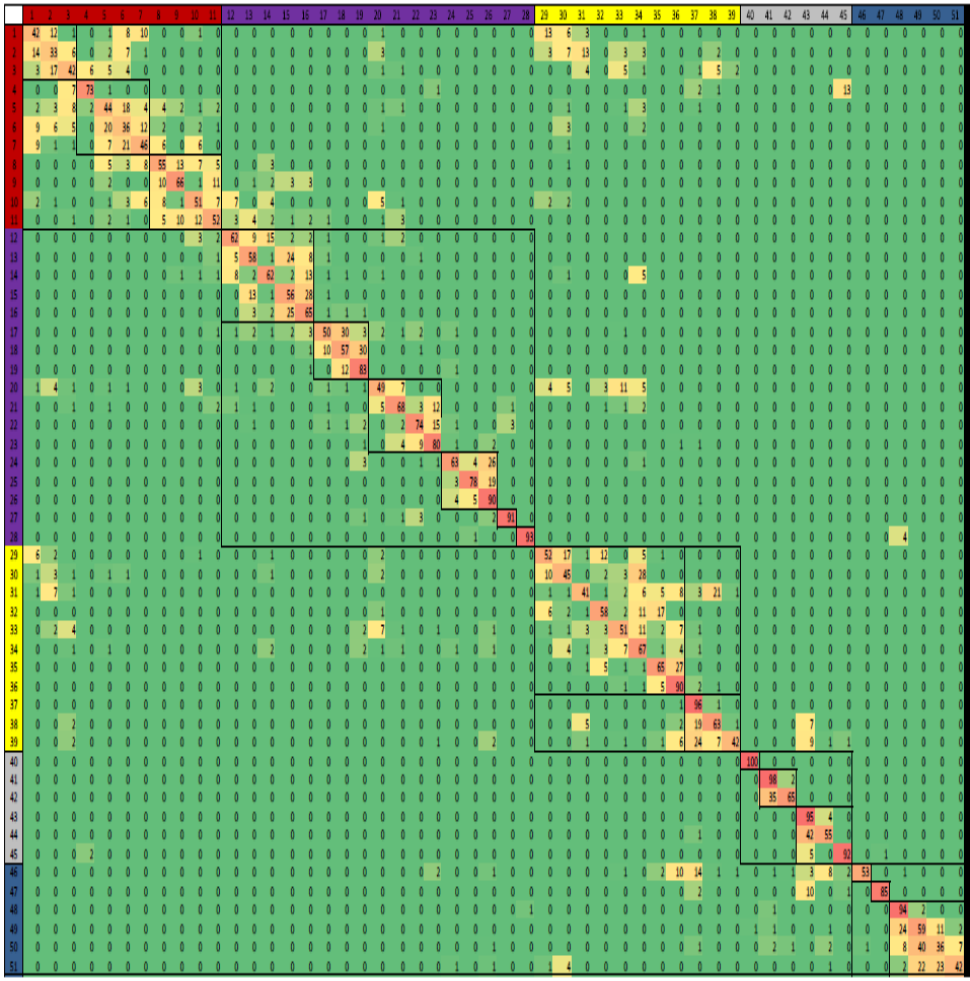
30+ epigenomics marks



WashU Epigenome Browser

ChromHMM: multi-variate hidden Markov model

Solution: Multi-Variariate HMMs, Chromatin States



- **Emission matrix $e_k(x_i)$**
  - Multi-variate HMM
  - Emits vector of values
- **Transition matrix a_{kl}**
  - Learn spatial relationships
  - No a-priori ‘gene’ structure

Ernst and Kellis, Nature Biotech 2010, Nature 2011, Nature Methods 2012

Goals for today: Computational Epigenomics

1. Introduction to Epigenomics
   - Overview of epigenomics, Diversity of Chromatin modifications
   - Antibodies, ChIP-Seq, data generation projects, raw data
2. Primary data processing: Read mapping, Peak calling
   - Read mapping: Hashing, Suffix Trees, Burrows-Wheeler Transform
   - Quality Control, Cross-correlation, Peak calling, IDR (similar to FDR)
3. Discovery and characterization of chromatin states
   - HMM Foundations, Generating, Parsing, Decoding, Learning
   - ChromHMM: Multi-variate HMM for chromatin state learning
4. Model complexity: selecting the number of states/marks
   - Selecting the number of states, selecting number of marks
   - Capturing dependencies and state-conditional mark independence
5. Model complexity: selecting the number of states/marks
   - Selecting the number of states, selecting number of marks
   - Capturing dependencies and state-conditional mark independence
6. Learning chromatin states jointly across multiple cell types
   - Stacking vs. concatenation approach for joint multi-cell type learning
   - Defining activity profiles for linking enhancer regulatory networks

HMM Foundations, Parsing, Decoding, Learning

1. HMM basics, evaluation, parsing, posterior decoding
   - Observations, Models, Bayes' rule, Bayesian inference
   - Markov Chains and Hidden Markov Models
   - Calculating joint probability of one (seq,parse) $P(x,\pi)$
   - Viterbi algorithm: Find best parse $\pi^* = \text{argmax}_\pi P(x,\pi)$
   - Forward algorithm: Find total $P(x)$, sum over all paths
   - Posterior Decoding: Most likely state π_i (over all paths)
2. Learning (ML training, Baum-Welch, Viterbi training)
   - Supervised: Find $e_i(.)$ and a_{ij} given labeled sequence
   - Unsupervised: given only x → annotation + params
3. Increasing the 'state' space / adding memory
   - Finding GC-rich regions vs. finding CpG islands
   - Gene structures GENSCAN, chromatin ChromHMM

HMM Foundations, Parsing, Decoding, Learning

1. HMM basics, evaluation, parsing, posterior decoding
   - Observations, Models, Bayes' rule, Bayesian inference
   - Markov Chains and Hidden Markov Models
   - Calculating joint probability of one (seq,parse) $P(x,\pi)$
   - Viterbi algorithm: Find best parse $\pi^* = \text{argmax}_\pi P(x,\pi)$
   - Forward algorithm: Find total $P(x)$, sum over all paths
   - Posterior Decoding: Most likely state π_i (over all paths)
2. Learning (ML training, Baum-Welch, Viterbi training)
   - Supervised: Find $e_i(.)$ and a_{ij} given labeled sequence
   - Unsupervised: given only x → annotation + params
3. Increasing the 'state' space / adding memory
   - Finding GC-rich regions vs. finding CpG islands
   - Gene structures GENSCAN, chromatin ChromHMM

We have learned how to align sequences to other sequences

- Sequence alignment
  - Dynamic programming, duality path ⇔ alignment
  - Global / local alignment, general gap penalties
- Rapid string search
  - Exact string match, semi-numerical matching
  - Database search: Hashing, BLAST, variations
- Comparative genomics: evolutionary signatures
  - Tell me how you evolve, I'll tell you what you are
  - Identifying conserved elements through evolution
- Whole-genome assembly/alignment/duplication:
  - Finding all common substrings within/across species
  - Contigs/scaffolds, string graphs, glocal alignmt paths
- Problem set 1 due next Tues, project planning

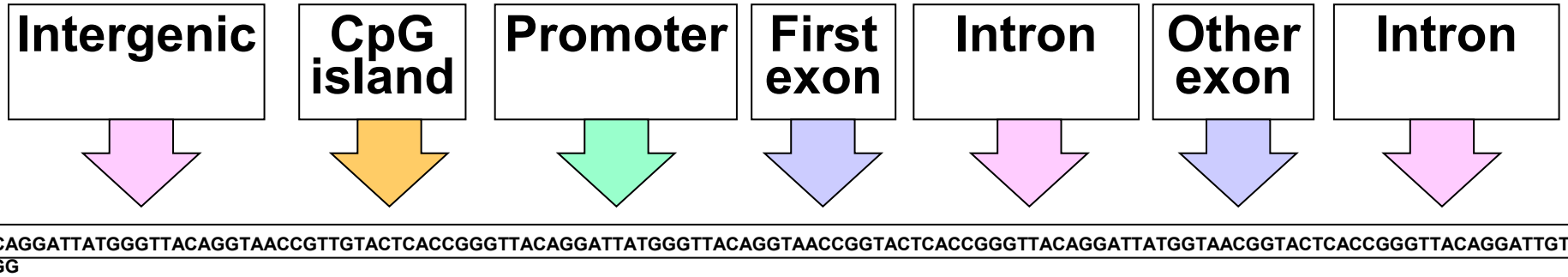
Today: apply these ideas to model DNA sequences

…GTACTCACCGGGTTACAGGATTATGGGTTACAGGTAACCGTT…

- What to do with a completely new piece of DNA
  - Align it to things we know about (database search)
  - Align it to things we don't know about (assembly)
- Stare at it
  - Non-standard nucleotide composition?
  - Interesting k-mer frequencies?
  - Recurrent patterns?
- Model it
  - Make some hypotheses about it
  - Build a 'generative model' to describe it
  - Find sequences of similar ***type***

➔ How do we model DNA sequences?

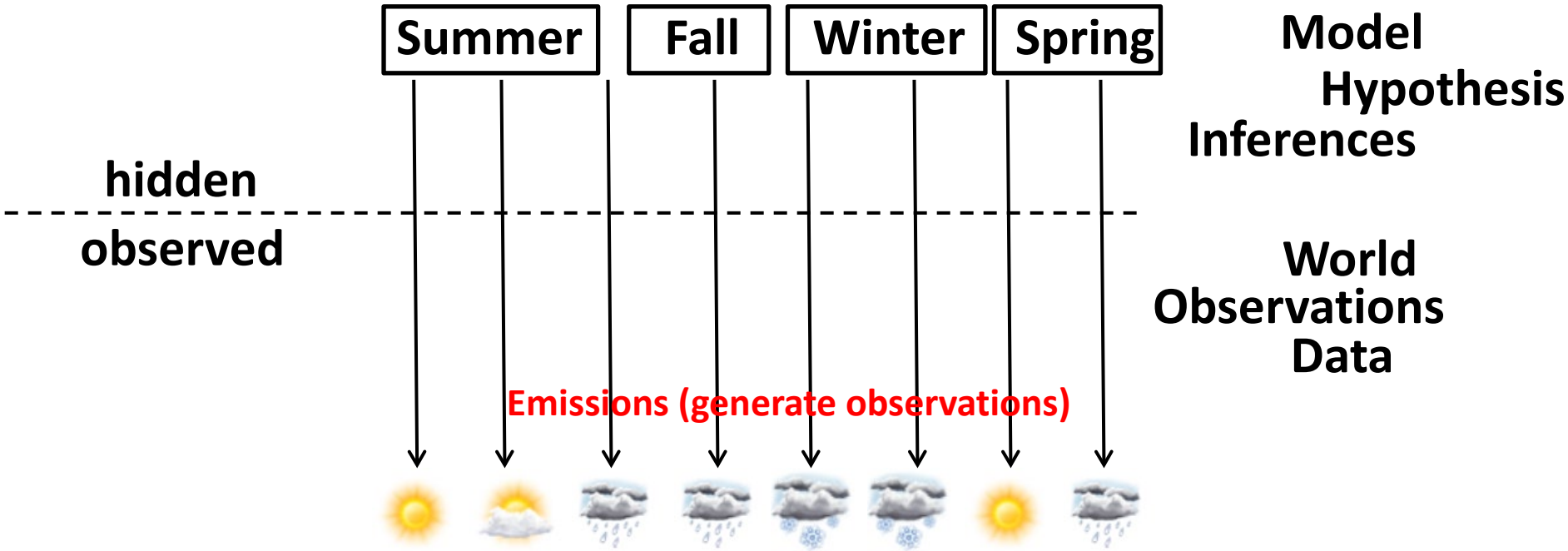
Modeling biological sequences with HMMs
(a.k.a. What to do with big unlabeled regions of DNA)



- Ability to **emit** DNA sequences of a certain ***type***
  - Not exact alignment to previously known gene
  - Preserving ‘properties’ of **type**, not identical sequence
- Ability to **recognize** DNA sequences of a certain type (state)
  - What (hidden) state is most likely to have generated observations
  - Find set of states and transitions that generated a long sequence
- Ability to **learn** distinguishing characteristics of each state
  - Training our generative models on large datasets
  - Learn to classify unlabeled data

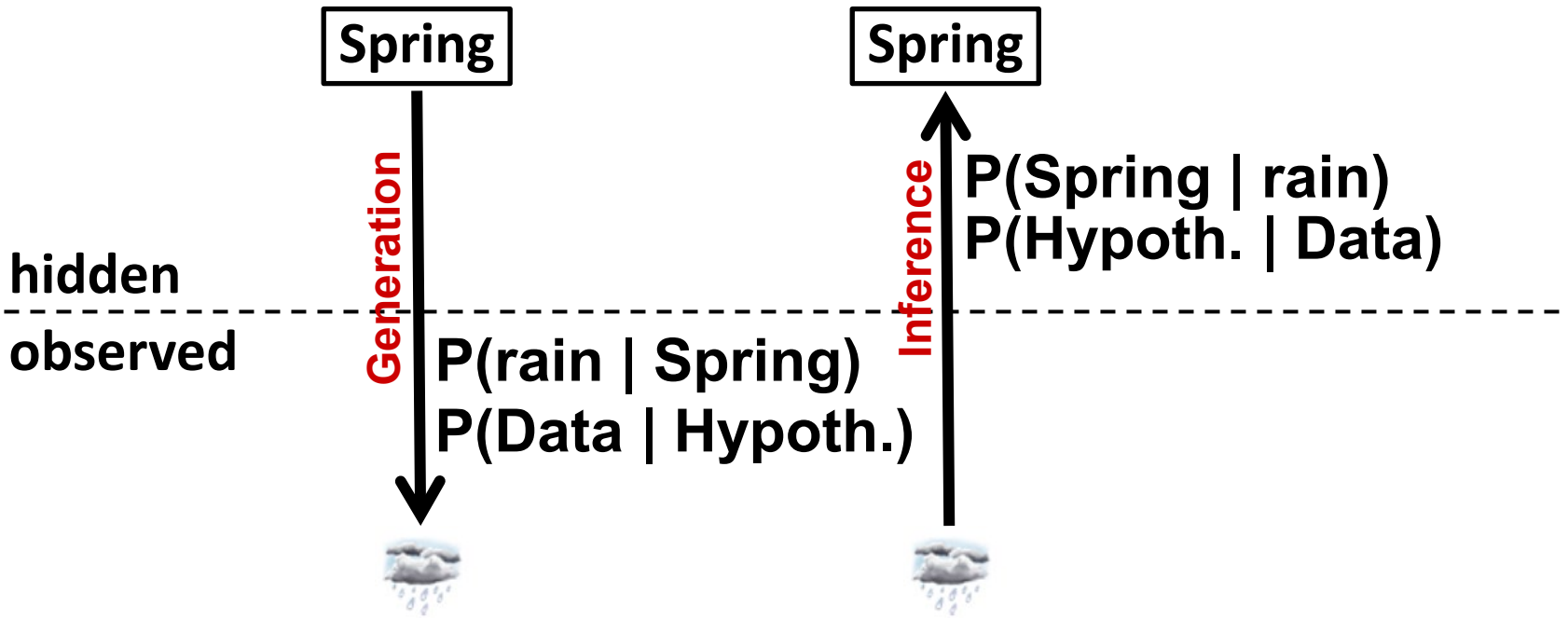
Generative models, Inference, "Reversing" the arrows

- Generative models:
  Express **forward** probability of an event,
  given the **hidden** state of the world



- We can measure/estimate:
  - P(snow | winter), P(observation | model) ➔ Generation
- We can infer:
  - P(winter | snow), P(model | observation) ➔ Inference

Bayesian Inference: "Reversing the arrows"



- Goal: P(D|H) ➔ P(H|D)
- Bayes' Rule allows us to do this:

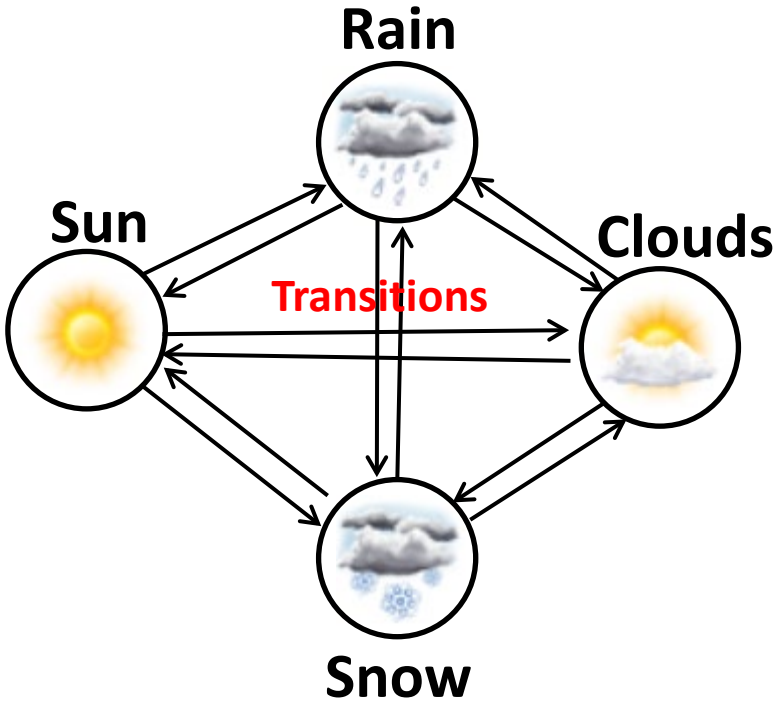
$$\underbrace{P(H|D)}_{\text{Posterior}} = \frac{\overbrace{P(D|H)}^{\text{Likelihood}}\ \overbrace{P(H)}^{\text{Prior}}}{\underbrace{P(D)}_{\text{Marginal likelihood aka. Model evidence}}}$$

HMM Foundations, Parsing, Decoding, Learning

1. HMM basics, evaluation, parsing, posterior decoding
   - Observations, Models, Bayes' rule, Bayesian inference
   - Markov Chains and Hidden Markov Models
   - Calculating joint probability of one (seq,parse) $P(x,\pi)$
   - Viterbi algorithm: Find best parse $\pi^* = \text{argmax}_\pi P(x,\pi)$
   - Forward algorithm: Find total $P(x)$, sum over all paths
   - Posterior Decoding: Most likely state π_i (over all paths)
2. Learning (ML training, Baum-Welch, Viterbi training)
   - Supervised: Find $e_i(.)$ and a_{ij} given labeled sequence
   - Unsupervised: given only x → annotation + params
3. Increasing the 'state' space / adding memory
   - Finding GC-rich regions vs. finding CpG islands
   - Gene structures GENSCAN, chromatin ChromHMM

Markov chains and Hidden Markov Models (HMMs)

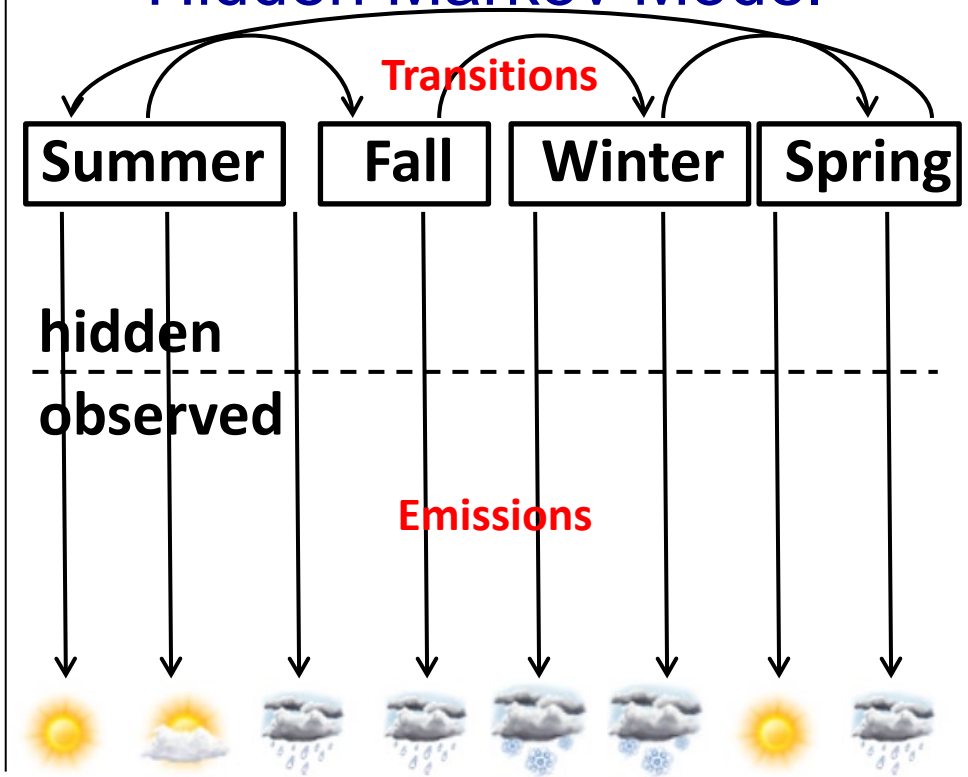
- Markov Chain



All observed

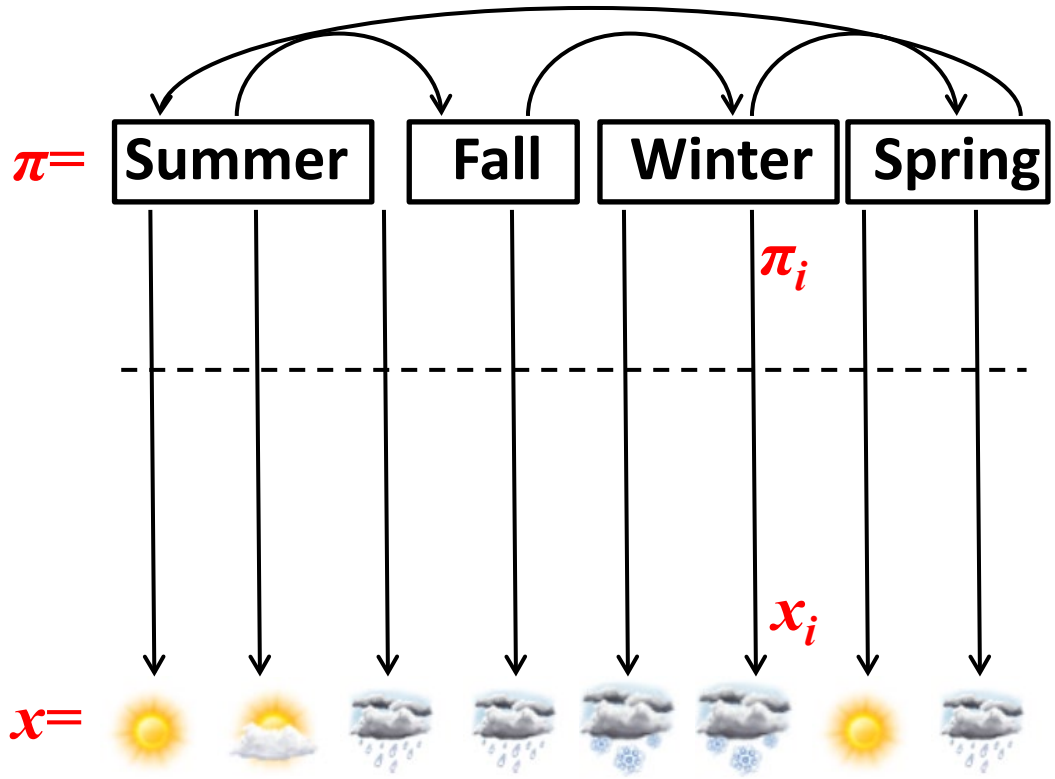
- What you see is what you get: next state only depends on current state (no memory)

- Hidden Markov Model



- Hidden state of the world (e.g. storm system) determines emission probabilities
- State transitions governed by a Markov chain

HMM nomenclature for this course



Transitions: $a_{kl}=P(\pi_i=l|\pi_{i-1}=k)$

Transition probability from state k to state l

Emissions: $e_k(x_i)=P(x_i|p_i=k)$

Emission probability of symbol x_i from state k

- Vector x = Sequence of observations
- Vector π = Hidden path (sequence of hidden states)
- Transition matrix A=a_{kl}=probability of $k \rightarrow l$ state transition
- Emission vector $E=e_k(x_i)$ = prob. of observing x_i from state k
- Bayes's rule: Use $P(x_i|\pi_i=k)$ to estimate $P(\pi_i=k|x_i)$

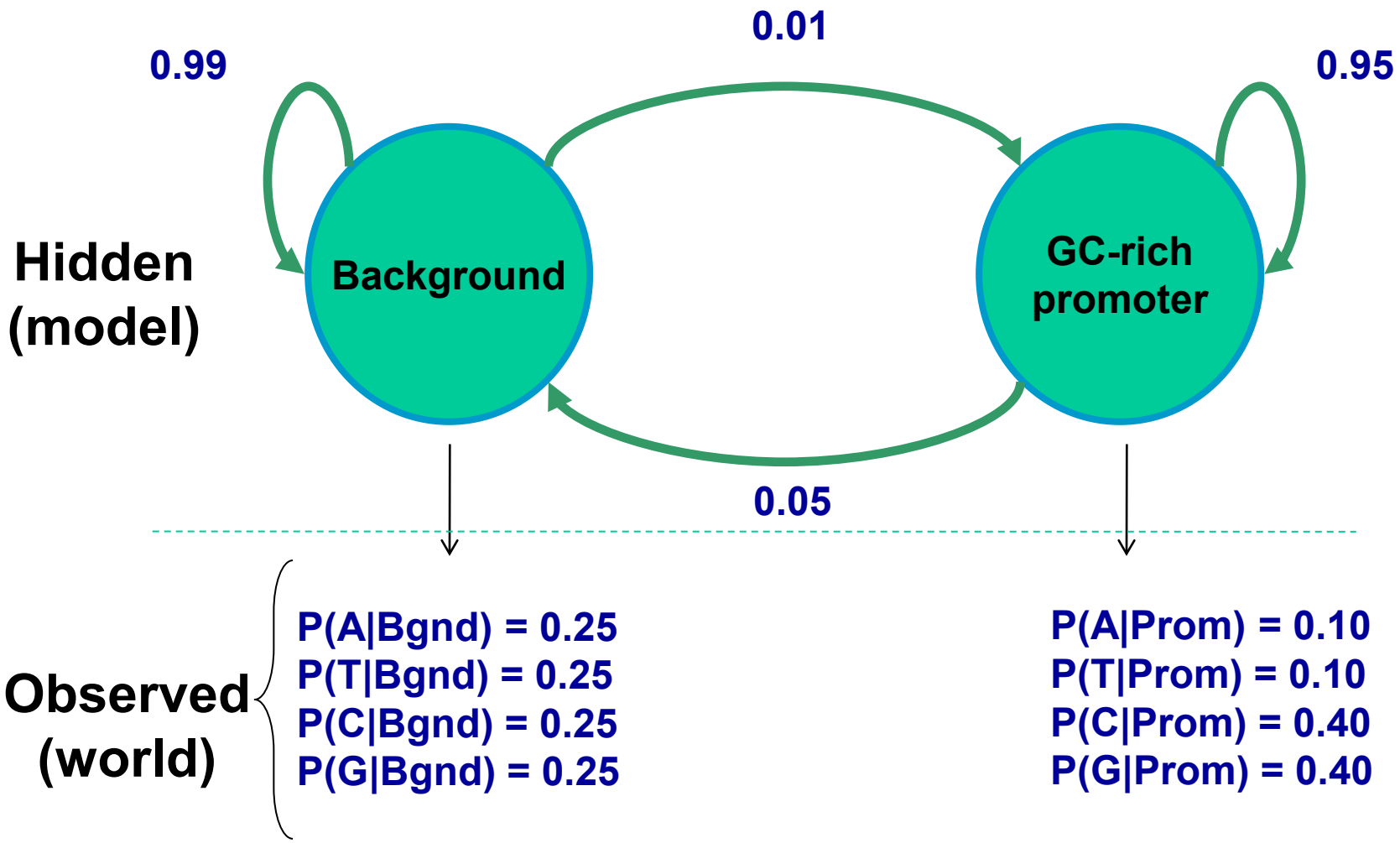
HMM Foundations, Parsing, Decoding, Learning

1. HMM basics, evaluation, parsing, posterior decoding
   - Observations, Models, Bayes' rule, Bayesian inference
   - Markov Chains and Hidden Markov Models
   - Calculating joint probability of one (seq,parse) $P(x,\pi)$
   - Viterbi algorithm: Find best parse $\pi^* = \text{argmax}_\pi P(x,\pi)$
   - Forward algorithm: Find total $P(x)$, sum over all paths
   - Posterior Decoding: Most likely state π_i (over all paths)
2. Learning (ML training, Baum-Welch, Viterbi training)
   - Supervised: Find $e_i(.)$ and a_{ij} given labeled sequence
   - Unsupervised: given only x → annotation + params
3. Increasing the 'state' space / adding memory
   - Finding GC-rich regions vs. finding CpG islands
   - Gene structures GENSCAN, chromatin ChromHMM

The six algorithmic settings for HMMs

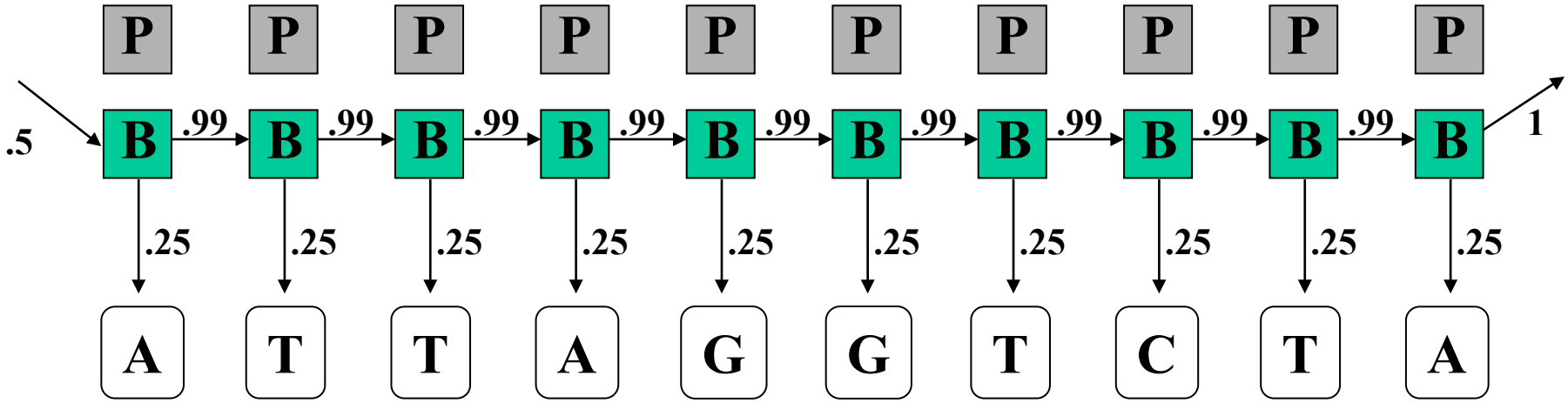
	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \sum_{\pi} P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_{\pi} P(x,\pi)$ Most likely path	4. Posterior decoding $\pi\text{^} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_{\pi} P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_{\Lambda} P(x,\pi\mid\Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_{\Lambda} \max_{\pi} P(x,\pi\mid\Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_{\Lambda} \sum_{\pi} P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths

Detecting GC-rich regions: HMM architecture



Slide credit: Serafim Batzoglou

Score annotation (path π) given a seq (obs x): all-B



What is the joint probability of observing x and a specific path π:

π = Bgnd, Bgnd, Bgnd, Bgnd, Bgnd, Bgnd, Bgnd, Bgnd, Bgnd, Bgnd

and sequence (observations):

x = A , T, T, A, G, G, T, C, T, A

Joined probability P(x,π)=P(x|π)P(π)=P(emissions|path)*P(path)

emission transition emission transition emission

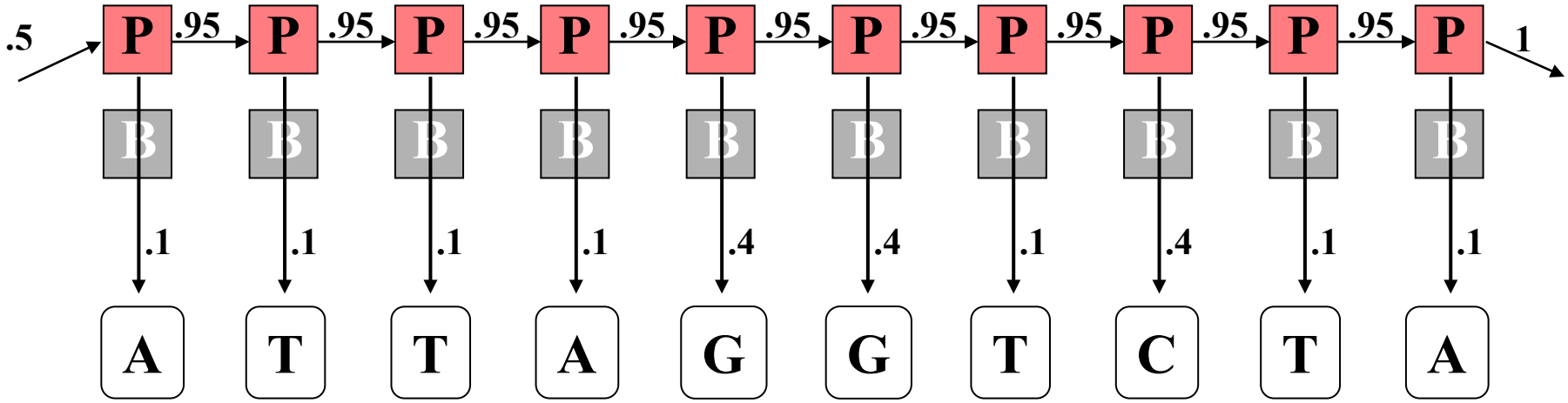
$p = ½ \times P(A \mid Bgnd)\, P(Bgnd_{i+1} \mid Bgnd_i)\, P(T \mid Bgnd)\, P(Bgnd \mid Bgnd) \ldots P(A \mid Bgnd)$

$= ½ \times (.25)^{10} \times (0.99)^{9}$

$= 5.2 \times 10^{-9}$

Why is p so small?

Score annotation (path π) given a seq (obs x): all-P



What is the likelihood of

π = Prom, Prom, Prom, Prom, Prom, Prom, Prom, Prom, Prom, Prom

and sequence

x = A, T , T , A, G, G, T, C, T, A

emission transition emission transition emission

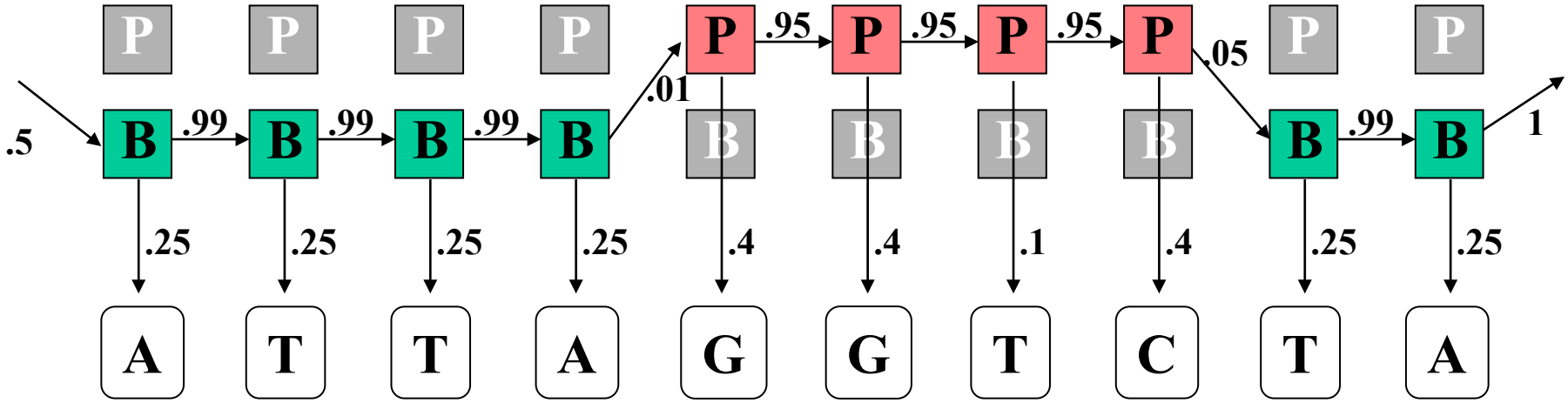
$p = ½ \times P(A \mid Prom)\, P(Prom_{i+1} \mid Prom_i)\, P(T \mid Prom)\, P(Prom|Promoad) \ldots P(G \mid Bgnd)$

$= ½ \times (.1)^7 \times (.4)^3\, (0.95)^9$

$= 2.0 \times 10^{-9}$

Compare the two!

Score annotation (path π) given a seq (obs x): B-P-B



What is the likelihood of

π = Bgnd, Bgnd, Bgnd, Bgnd, Prom, Prom, Prom, Prom, Bgnd, Bgnd

and sequence

x = A , T, T, A, G, G, T, C, T, A

emission transition emission transition emission

$p = ½ \times P(A \mid Bgnd)\, P(Bgnd_{i+1} \mid Bgnd_i)\, P(T \mid Bgnd)\, P(Bgnd \mid Bgnd) \ldots P(A \mid Bgnd)$

$= ½ \times (.1)^1 \times (.4)^3 \times (.25)^6 \times (0.99)^4 \times (0.01)^1 \times (0.95)^3 \times (0.05)^1$

$= 6.4 \times 10^{-10}$

Much less likely, due to high cost of transitions

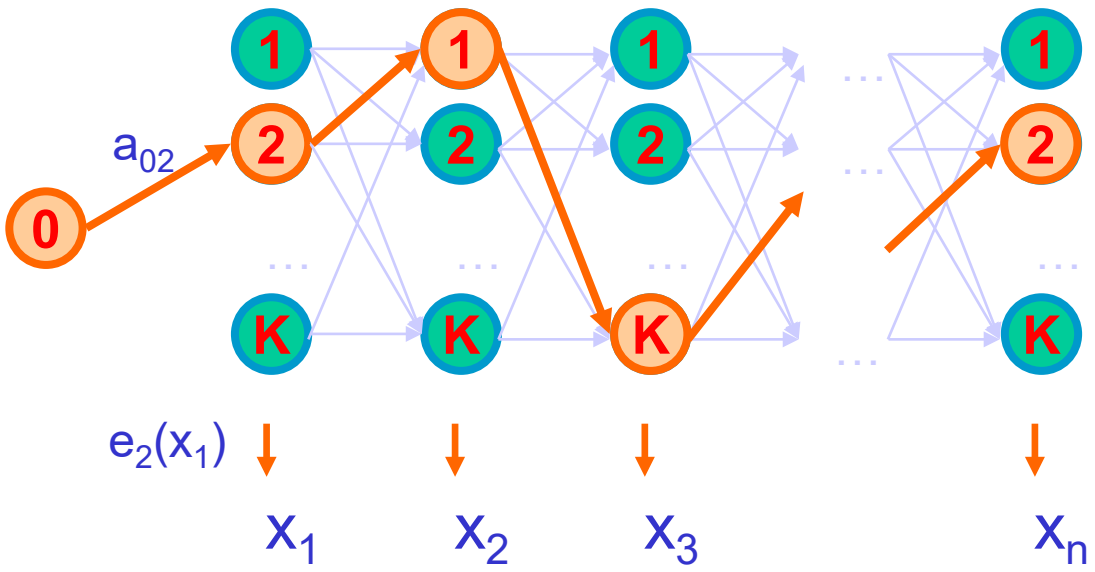
HMM Foundations, Parsing, Decoding, Learning

1. HMM basics, evaluation, parsing, posterior decoding
   - Observations, Models, Bayes' rule, Bayesian inference
   - Markov Chains and Hidden Markov Models
   - Calculating joint probability of one (seq,parse) $P(x,\pi)$
   - Viterbi algorithm: Find best parse $\pi^* = \text{argmax}_\pi P(x,\pi)$
   - Forward algorithm: Find total $P(x)$, sum over all paths
   - Posterior Decoding: Most likely state π_i (over all paths)
2. Learning (ML training, Baum-Welch, Viterbi training)
   - Supervised: Find $e_i(.)$ and a_{ij} given labeled sequence
   - Unsupervised: given only x → annotation + params
3. Increasing the 'state' space / adding memory
   - Finding GC-rich regions vs. finding CpG islands
   - Gene structures GENSCAN, chromatin ChromHMM

The six algorithmic settings for HMMs

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \sum_{\pi} P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_{\pi} P(x,\pi)$ Most likely path	4. Posterior decoding $\pi\text{^} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_{\pi} P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_{\Lambda} P(x,\pi\mid\Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_{\Lambda} \max_{\pi} P(x,\pi\mid\Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_{\Lambda} \sum_{\pi} P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths

Simple: Given the model, generate some sequence x



Given a HMM, we can generate a sequence of length n as follows:

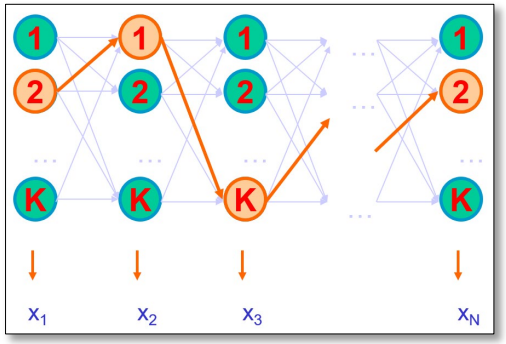
1. Start at state π_1 according to prob $a_{0\pi1}$
2. Emit letter x_1 according to prob $e_{\pi1}(x_1)$
3. Go to state π_2 according to prob $a_{\pi1\pi2}$
4. … until emitting x_n

We have some sequence x that can be emitted by p. Can calculate its likelihood.
However, in general, many different paths may emit this same sequence x.
How do we find the total probability of generating a given x, over any path?

Slide credit: Serafim Batzoglou

Finding the optimal path

- Find path π^* that maximizes total joint probability $P[x, \pi]$

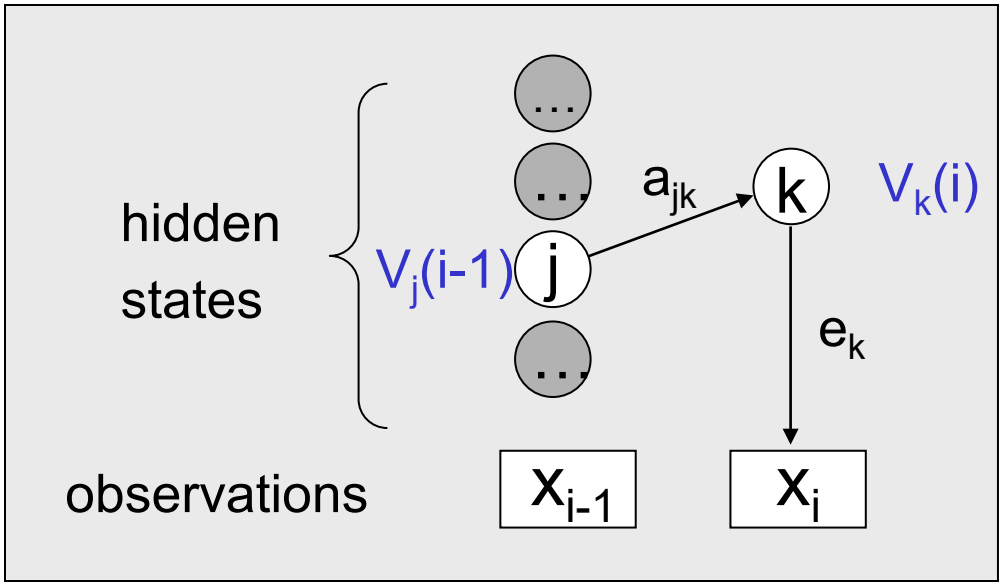


- $P(x,\pi) = a_{0\pi_1} * \prod_i e_{\pi_i}(x_i) \times a_{\pi_i\pi_{i+1}}$

start emission transition

- Can evaluate any path through hidden states, given emitted seqs
- How do we find the best path?
  - Try all possibilities? Not practical, exponential number of paths
  - 1 human gene ~ 100,000 nucleotides ➔ $2^{100,000}$ paths for 2 states!
- Instead: Dynamic programming ➔ Viterbi algortithm
  - Store partial computation (max score to position i through state k)
    ➔ Define $V_k(i)$ = Probability of most likely path through state π_i=k
  - Use it to compute max score to position i+1 through each state k'
    ➔ Compute $V_{k'}(i+1)$ as a function of $\max_k \{ V_k(i) \}$
  - Simple computation, just include emission score + cost of transition
    ➔ $V_k(i+1) = e_k(x_{i+1}) * \max_j a_{jk} V_j(i)$
- DP works because of optimal sub-structure:
  - Best path through given state is: (1) Best path to previous state. (2) Best transition from previous state to this state. (3) Best path to the end state

Calculate maximum P(x,π) recursively



- Assume we know V_j for the previous time step (i-1)
- Calculate $V_k(i) = e_k(x_i) * \max_j (V_j(i-1) \times a_{jk})$

current max

this emission

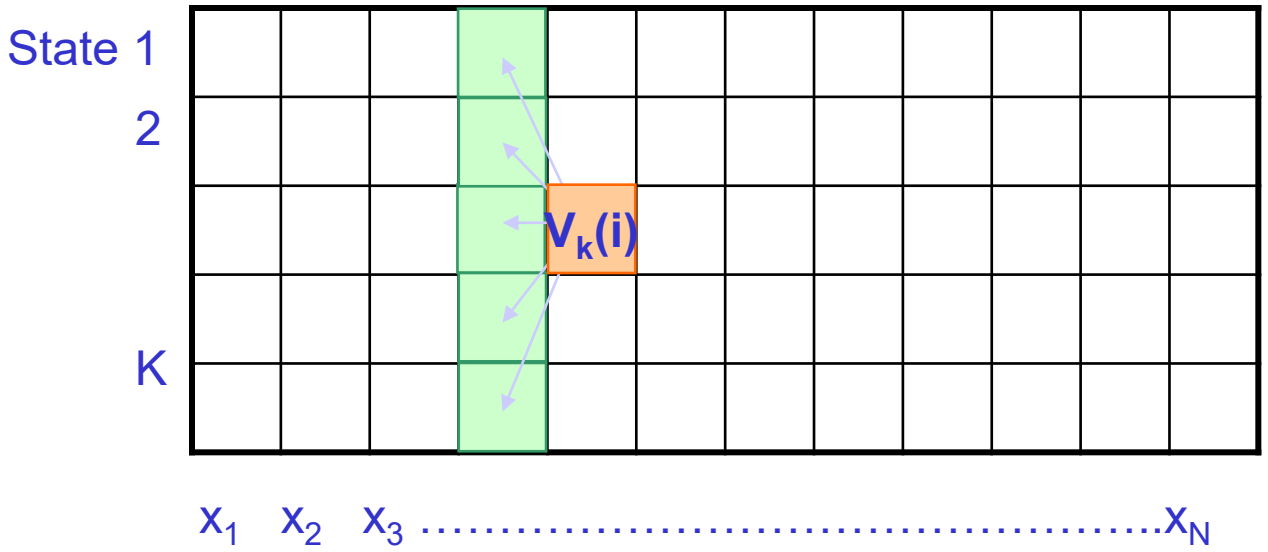
max ending in state j at step i

Transition from state j

all possible previous states j

Slide credit: Serafim Batzoglou

The Viterbi Algorithm



Input: x = x1……xN

Initialization:

$V_0(0)=1$, $V_k(0) = 0$, for all $k > 0$

Iteration:

$V_k(i) = e_K(x_i) \times \max_j a_{jk} V_j(i-1)$

Termination:

$P(x, \pi^*) = \max_k V_k(N)$

Traceback:

Follow max pointers back

Similar to aligning states to seq

In practice:

Use log scores for computation

Running time and space:

Time: $O(K^2N)$

Space: $O(KN)$

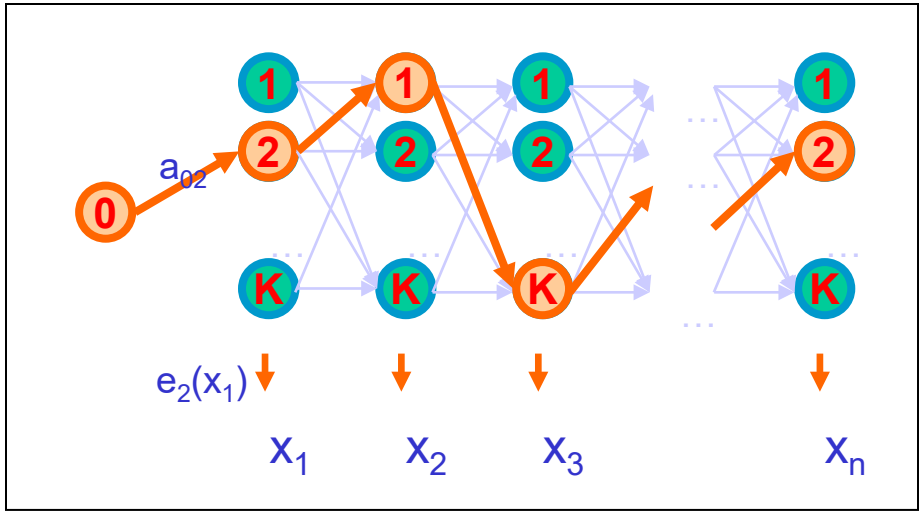
HMM Foundations, Parsing, Decoding, Learning

1. HMM basics, evaluation, parsing, posterior decoding
   - Observations, Models, Bayes' rule, Bayesian inference
   - Markov Chains and Hidden Markov Models
   - Calculating joint probability of one (seq,parse) $P(x,\pi)$
   - Viterbi algorithm: Find best parse $\pi^* = \text{argmax}_\pi P(x,\pi)$
   - Forward algorithm: Find total $P(x)$, sum over all paths
   - Posterior Decoding: Most likely state π_i (over all paths)
2. Learning (ML training, Baum-Welch, Viterbi training)
   - Supervised: Find $e_i(.)$ and a_{ij} given labeled sequence
   - Unsupervised: given only x → annotation + params
3. Increasing the 'state' space / adding memory
   - Finding GC-rich regions vs. finding CpG islands
   - Gene structures GENSCAN, chromatin ChromHMM

The six algorithmic settings for HMMs

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions ✓	2. Scoring x, all paths $P(x) = \sum_{\pi} P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_{\pi} P(x,\pi)$ Most likely path ✓	4. Posterior decoding $\pi\text{^} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_{\pi} P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_{\Lambda} P(x,\pi\mid\Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_{\Lambda} \max_{\pi} P(x,\pi\mid\Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_{\Lambda} \sum_{\pi} P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths

Complex: Given x, was it generated by the model?



Given a sequence x,

What is the probability that x was generated by the model (using any path)?

- $P(x) = \sum_{\pi} P(x,\pi) = \sum_{\pi} P(x|\pi)\, P(\pi)$
- (weighted average of conditional probability, summed over all paths, weighted by each path's probability)

- Challenge: exponential number of paths

Slide credit: Serafim Batzoglou

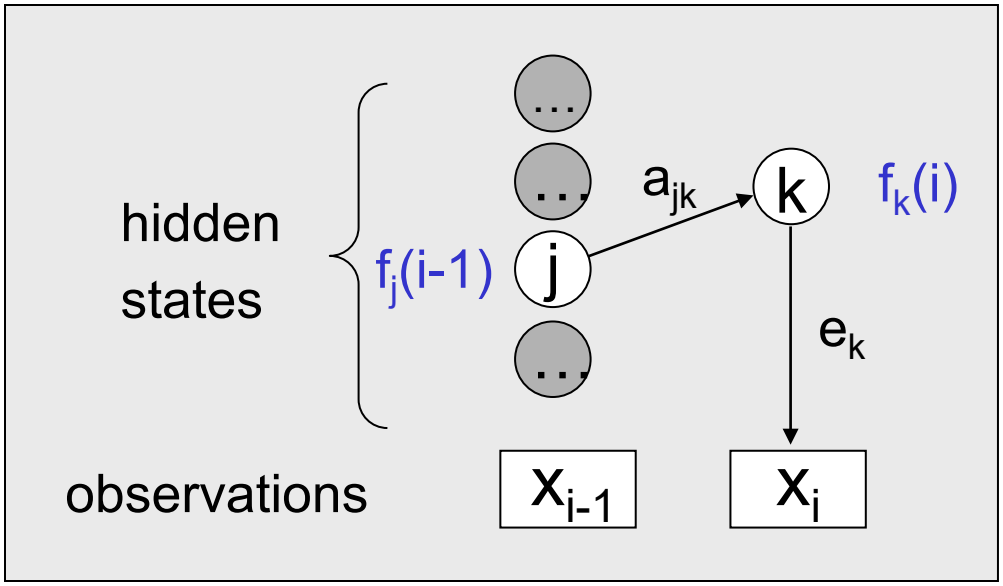
Calculate probability of emission over all paths

- Each path has associated probability
  - Some paths are likely, others unlikely: sum them all up

  → Return total probability that emissions are observed, summed over all paths
  - Viterbi path is the most likely one
    - How much 'probability mass' does it contain?
- (cheap) alternative:
  - Calculate probability over maximum (Viterbi) path π^*
  - Good approximation if Viterbi has highest density
  - BUT: incorrect
- (real) solution
  - Calculate the exact sum iteratively
    - $P(x) = \sum_{\pi} P(x,\pi)$
  - Can use dynamic programming

Slide credit: Serafim Batzoglou

Calculate total probability $\Sigma_\pi P(x,\pi)$ recursively



- Assume we know f_j for the previous time step (i-1)
- Calculate $f_k(i) = e_k(x_i) * \text{sum}_j (f_j(i-1) \times a_{jk})$

updated sum

this emission

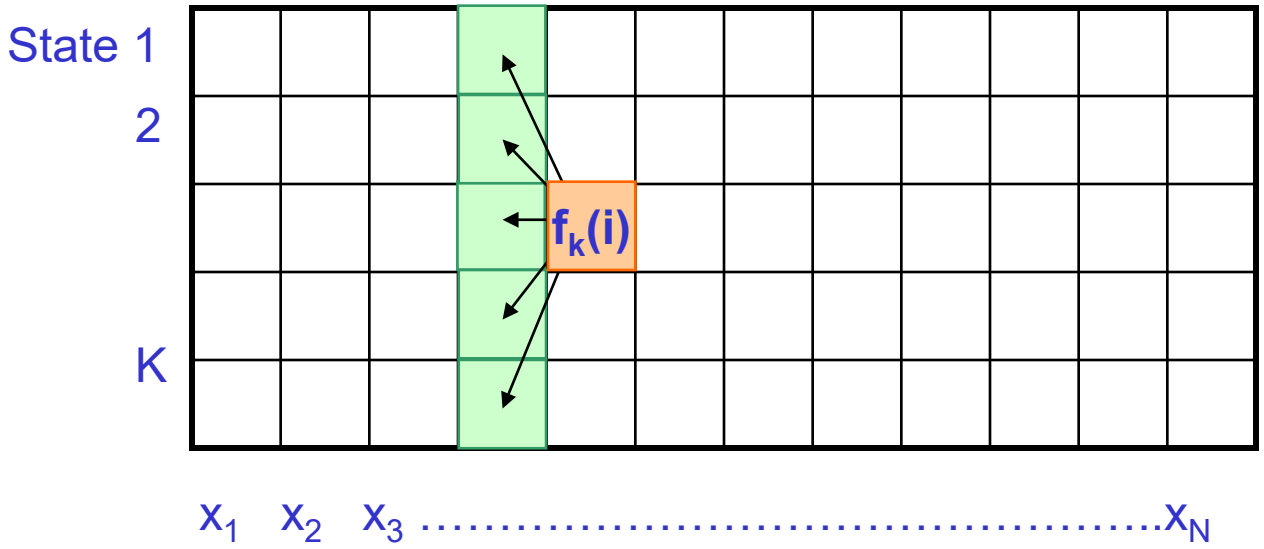
sum ending in state j at step i

transition from state j

every possible previous state j

Slide credit: Serafim Batzoglou

The Forward Algorithm



Input: x = x1……xN

Initialization:

$f_0(0)=1$, $f_k(0) = 0$, for all $k > 0$

Iteration:

$f_k(i) = e_K(x_i) \times \text{sum}_j\ a_{jk}\ f_j(i\text{-}1)$

Termination:

$P(x, \pi^*) = \text{sum}_k\ f_k(N)$

In practice:

Sum of log scores is difficult

→ approximate exP(A+p+q)

→ scaling of probabilities

Running time and space:

Time: $O(K^2N)$

Space: $O(KN)$

Slide credit: Serafim Batzoglou

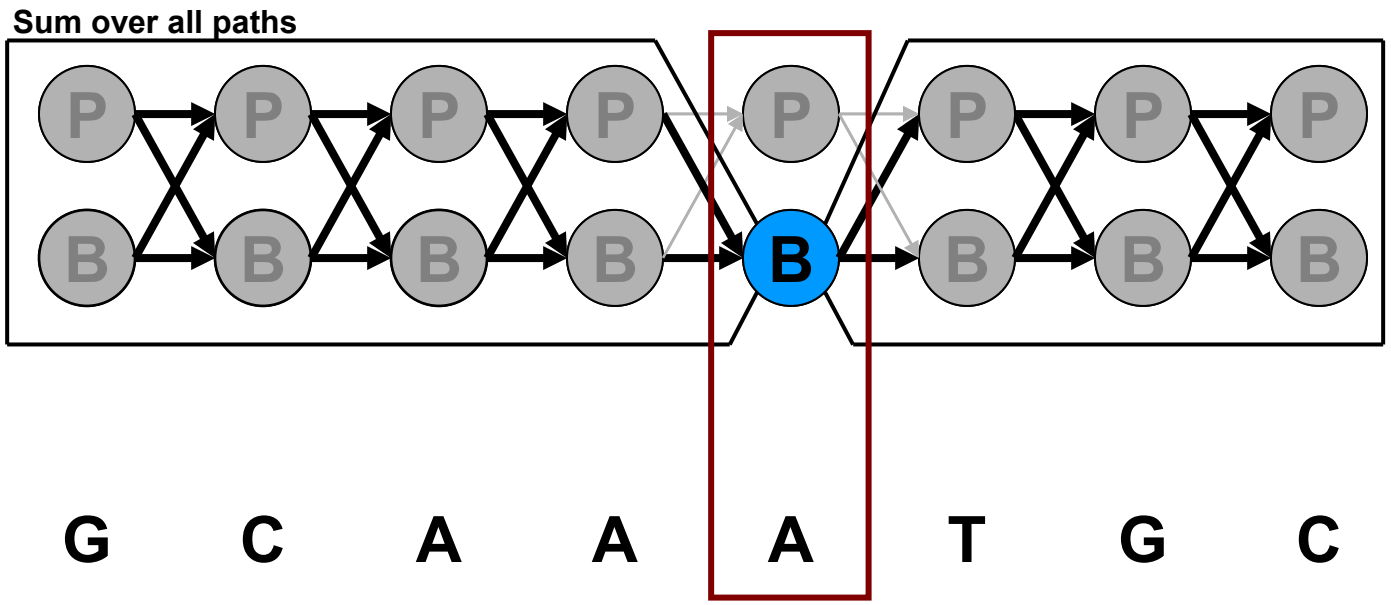
HMM Foundations, Parsing, Decoding, Learning

1. HMM basics, evaluation, parsing, posterior decoding
   - Observations, Models, Bayes' rule, Bayesian inference
   - Markov Chains and Hidden Markov Models
   - Calculating joint probability of one (seq,parse) $P(x,\pi)$
   - Viterbi algorithm: Find best parse $\pi^* = \text{argmax}_\pi P(x,\pi)$
   - Forward algorithm: Find total $P(x)$, sum over all paths
   - Posterior Decoding: Most likely state π_i (over all paths)
2. Learning (ML training, Baum-Welch, Viterbi training)
   - Supervised: Find $e_i(.)$ and a_{ij} given labeled sequence
   - Unsupervised: given only x → annotation + params
3. Increasing the 'state' space / adding memory
   - Finding GC-rich regions vs. finding CpG islands
   - Gene structures GENSCAN, chromatin ChromHMM

The six algorithmic settings for HMMs

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions ✓	2. Scoring x, all paths $P(x) = \sum_{\pi} P(x,\pi)$ Prob of emissions, over all paths ✓
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_{\pi} P(x,\pi)$ Most likely path ✓	4. Posterior decoding $\pi\hat{} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_{\pi} P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_{\Lambda} P(x,\pi\mid\Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_{\Lambda} \max_{\pi} P(x,\pi\mid\Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_{\Lambda} \sum_{\pi} P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths

Calculate most probable label at a single position



- Calculate most probable label, L^*_i , at each position i
- Do this for all N positions gives us $\{L^*_1, L^*_2, L^*_3 \ldots L^*_N\}$
- How much information have we observed? Three settings:
  - **Observed nothing: Use prior information**
  - **Observed only character at position i: Prior + emission probability**
  - **Observed entire sequence: Posterior decoding**

Calculate $P(\pi_7 = \text{CpG+} \mid x_7 = G)$

- With no knowledge (no characters)
  - Simply time spent in markov chain states
  - $P(\pi_i = k)$ = most likely state (**prior**)

- With very little knowledge (just that character)
  - Time spent, adjusted for different emission probs.
  - Use Bayes rule to change inference directionality
  - $P(\pi_i = k \mid x_i = G) = P(\pi_i = k) * P(x_i = G \mid \pi_i = k) / P(x_i = G)$

- With knowledge of entire sequence (all characters)
  - $P(\pi_i = k \mid x = \text{AGCGCG…GATTATCGTCGTA})$
  - Sum over all paths that emit ‘G’ at position 7

  ➔ **Posterior** decoding

Motivation for the Backward Algorithm

We want to compute

$P(\pi_i = k \mid x)$, the probability distribution on the i^{th} position, given x

We start by computing

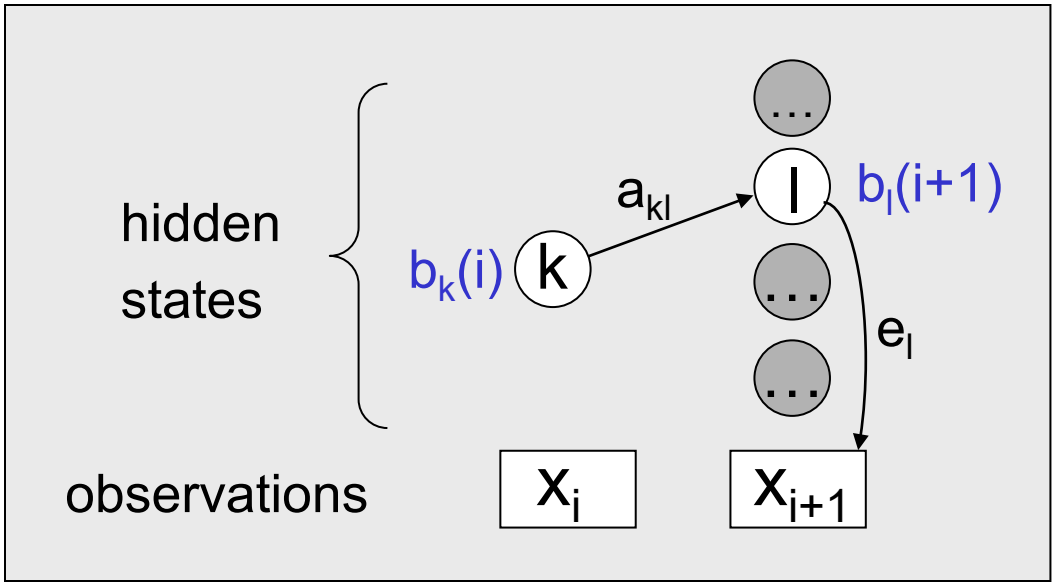
$$P(\pi_i = k, x) = P(x_1 \ldots x_i, \pi_i = k, x_{i+1} \ldots x_N)$$
$$= P(x_1 \ldots x_i, \pi_i = k)\, P(x_{i+1} \ldots x_N \mid x_1 \ldots x_i, \pi_i = k)$$
$$= P(x_1 \ldots x_i, \pi_i = k)\, P(x_{i+1} \ldots x_N \mid \pi_i = k)$$

Forward, $f_k(i)$ **Backward, $b_k(i)$**

Derivation: Define the backward probability recursively, as f(➔):

$$b_k(i) = P(x_{i+1} \ldots x_N \mid \pi_i = k)$$
$$= \Sigma_{\pi i+1 \ldots \pi N} P(x_{i+1}, x_{i+2}, \ldots, x_N, \pi_{i+1}, \ldots, \pi_N \mid \pi_i = k)$$
$$= \Sigma_l \Sigma_{\pi i+1 \ldots \pi N} P(x_{i+1}, x_{i+2}, \ldots, x_N, \pi_{i+1} = l, \pi_{i+2}, \ldots, \pi_N \mid \pi_i = k)$$
$$= \Sigma_l e_l(x_{i+1})\, a_{kl}\, \Sigma_{\pi i+1 \ldots \pi N} P(x_{i+2}, \ldots, x_N, \pi_{i+2}, \ldots, \pi_N \mid \pi_{i+1} = l)$$
$$= \Sigma_l e_l(x_{i+1})\, a_{kl}\, b_l(i+1)$$

Calculate total end probability recursively



- Assume we know b_l for the next time step (i+1)
- Calculate $b_k(i)$ = sum_l ($e_l(x_{i+1})$ × a_{kl} × $b_l(i+1)$)

$b_k(i)$: current max

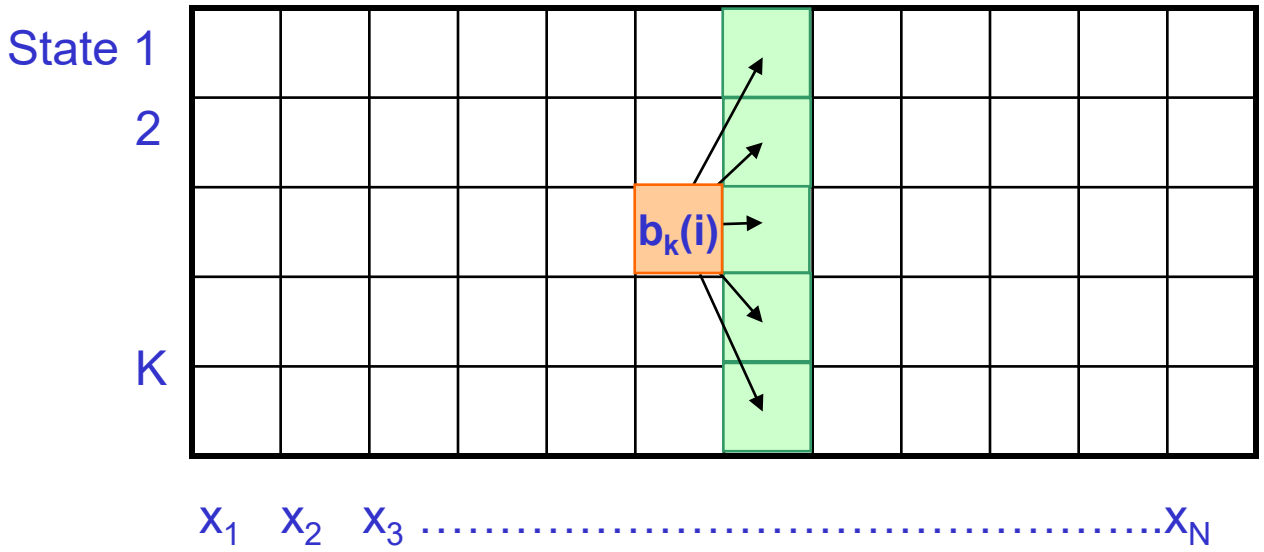
$e_l(x_{i+1})$: next emission

a_{kl}: transition to next state

$b_l(i+1)$: prob sum from state l to end

sum_l: sum over all possible next states

The Backward Algorithm



Input: x = x1……xN

Initialization:

$b_k(N) = a_{k0}$, for all k

Iteration:

$b_k(i) = \Sigma_l\, e_l(x_{i+1})\, a_{kl}\, b_l(i+1)$

Termination:

$P(x) = \Sigma_l\, a_{0l}\, e_l(x_1)\, b_l(1)$

In practice:

Sum of log scores is difficult

→ approximate exp(1+p+q)

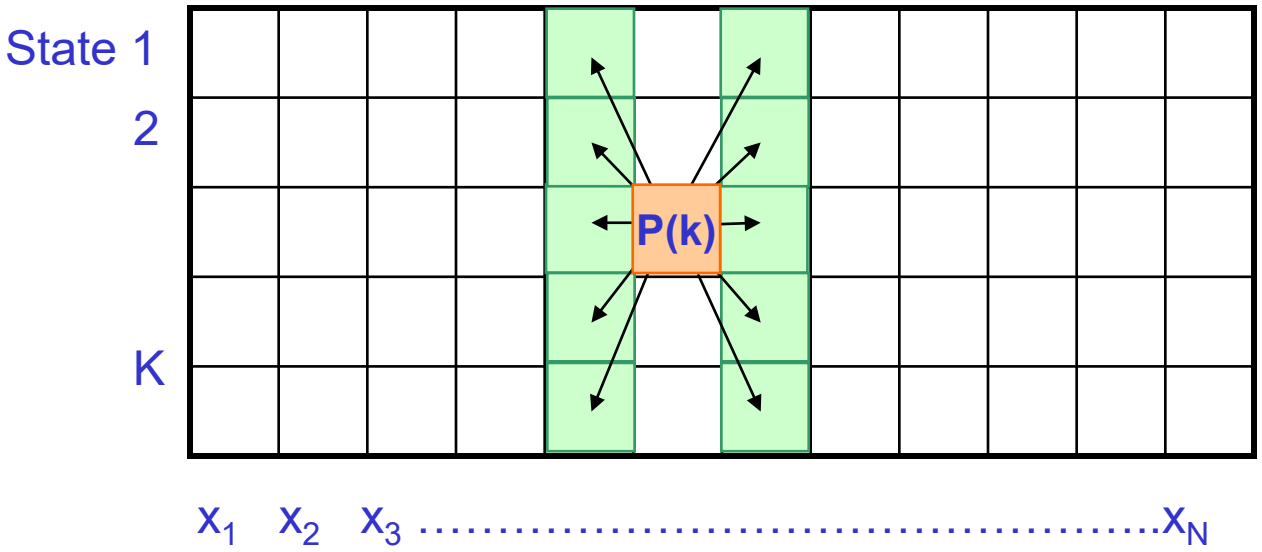
→ scaling of probabilities

Running time and space:

Time: $O(K^2N)$

Space: $O(K)$

Putting it all together: Posterior decoding



- $P(k) = P(\pi_i=k \mid x) = f_k(i)*b_k(i) / P(x)$
  - Probability that i^{th} state is k, given all emissions x
- Posterior decoding
  - Find the most likely state at position i over all possible hidden paths given the observed sequence x
  - $\hat{\pi}_i = \text{argmax}_k P(\pi_i = k \mid x)$
- Posterior decoding 'path' $\hat{\pi}_i$
  - For classification, more informative than Viterbi path π^*
    - More refined measure of "which hidden states" generated x
  - However, it may give an invalid sequence of states
    - Not all j→k transitions may be possible

HMM Foundations, Parsing, Decoding, Learning

1. HMM basics, evaluation, parsing, posterior decoding
   - Observations, Models, Bayes' rule, Bayesian inference
   - Markov Chains and Hidden Markov Models
   - Calculating joint probability of one (seq,parse) $P(x,\pi)$
   - Viterbi algorithm: Find best parse $\pi^* = \text{argmax}_\pi P(x,\pi)$
   - Forward algorithm: Find total $P(x)$, sum over all paths
   - Posterior Decoding: Most likely state π_i (over all paths)
2. Learning (ML training, Baum-Welch, Viterbi training)
   - Supervised: Find $e_i(.)$ and a_{ij} given labeled sequence
   - Unsupervised: given only x → annotation + params
3. Increasing the 'state' space / adding memory
   - Finding GC-rich regions vs. finding CpG islands
   - Gene structures GENSCAN, chromatin ChromHMM

The six algorithmic settings for HMMs

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions ✓	2. Scoring x, all paths $P(x) = \sum_{\pi} P(x,\pi)$ Prob of emissions, over all paths ✓
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_{\pi} P(x,\pi)$ Most likely path ✓	4. Posterior decoding $\pi\hat{} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_{\pi} P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_{\Lambda} P(x,\pi\mid\Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_{\Lambda} \max_{\pi} P(x,\pi\mid\Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_{\Lambda} \sum_{\pi} P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths

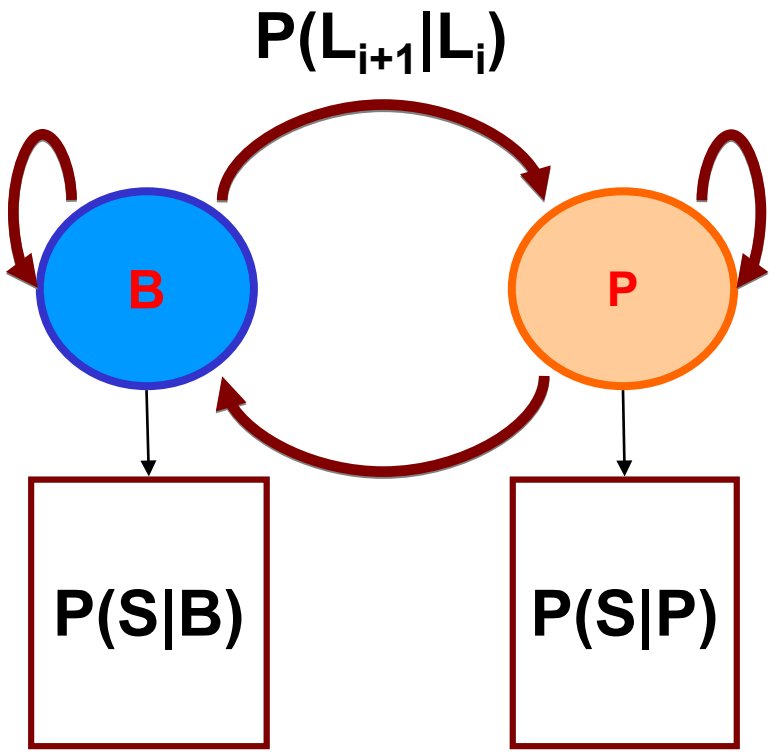
Learning: How to train an HMM

Transition probabilities

e.g. $P(P_{i+1}|B_i)$ – the probability of entering a pathogenicity island from background DNA

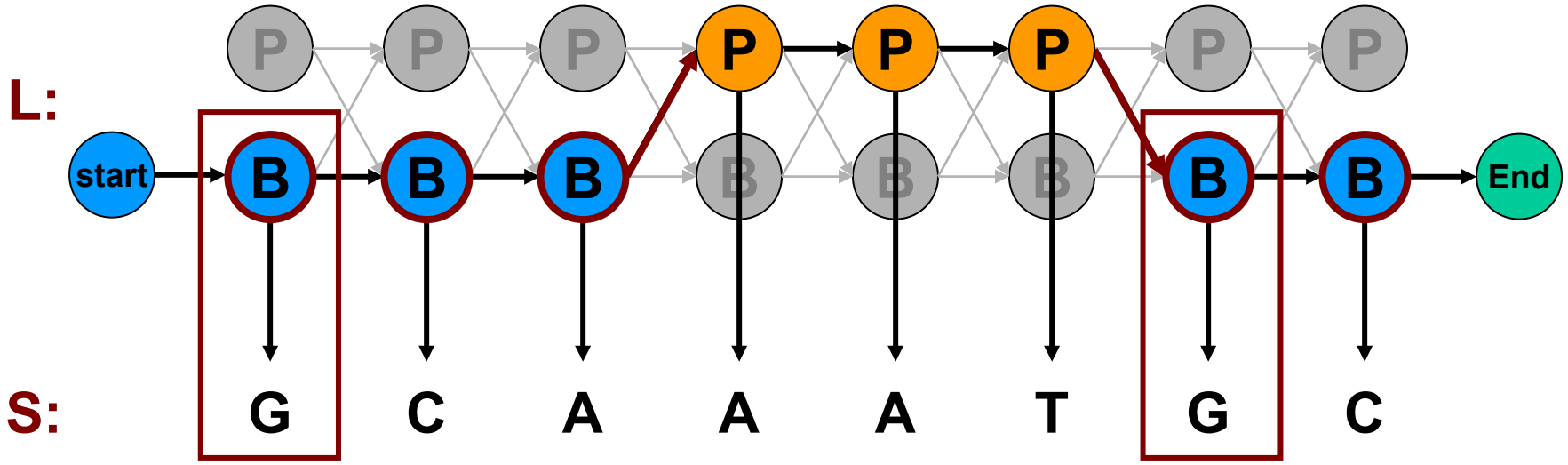
Emission probabilities

i.e. the nucleotide frequencies for background DNA and pathogenicity islands



Learning From labeled Data
→ Maximum Likelihood Estimation

If we have a sequence that has islands marked, we can simply count



$P(L_{i+1}|L_i)$

	B_{i+1}	P_{i+1}
B_i		
P_i		

P(S|B)

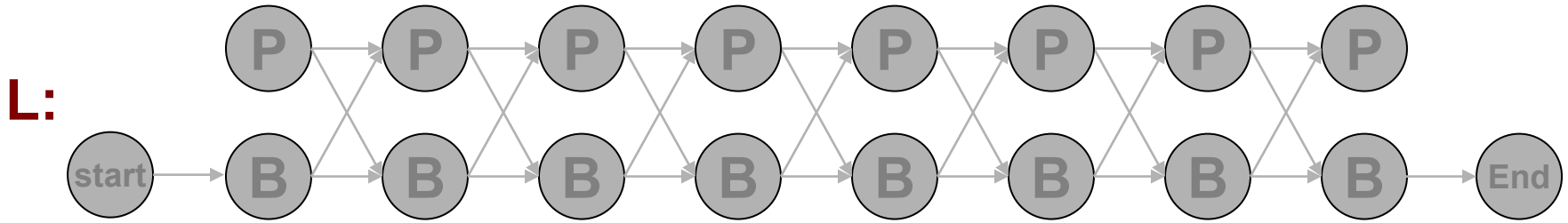
A:
T:
G:
C:
!

P(S|P)

A:
T:
G:
C:
ETC..

Unlabeled Data

How do we know how to count?



S: G C A A A T G C

?

$P(L_{i+1}|L_i)$

	B_{i+1}	P_{i+1}	End
B_i			
P_i			
Start			

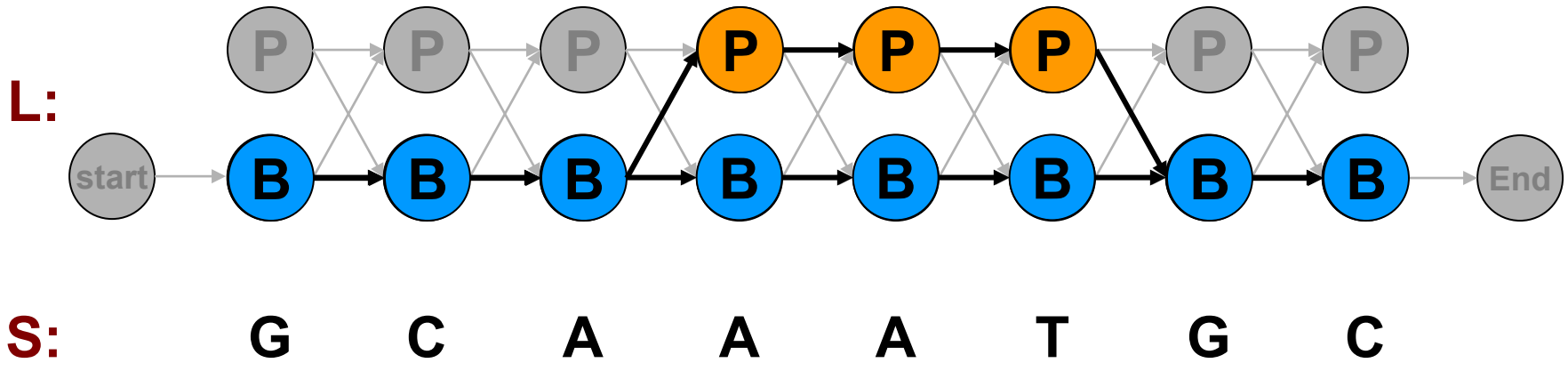
P(S|B)

A:
T:
G:
C:

P(S|P)

A:
T:
G:
C:

Unlabeled Data



An idea:

1. Imagine we start with some parameters
2. We *could* calculate the most likely path, P*, given those parameters and S
3. We *could* then use P* to update our parameters by maximum likelihood
4. And iterate (to convergence)

~~$P(L_{i+1}|L_i)^0$ $P(S|B)^0$ $P(S|P)^0$~~

~~$P(L_{i+1}|L_i)^1$ $P(S|B)^1$ $P(S|P)^1$~~

$P(L_{i+1}|L_i)^2$ $P(S|B)^2$ $P(S|P)^2$

…

$P(L_{i+1}|L_i)^K$ $P(S|B)^K$ $P(S|P)^K$

Simple case: Viterbi Training

Initialization:

Pick the best-guess for model parameters

(or arbitrary)

Iteration:

1. Perform Viterbi, to find π^*
2. Calculate A_{kl}, $E_k(b)$ according to π^* + pseudocounts
3. Calculate the new parameters a_{kl}, $e_k(b)$

Until convergence

Notes:

- Convergence to local maximum guaranteed. Why?
- Does not maximize $P(x \mid \theta)$
- In general, worse performance than Baum-Welch

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions ✓	2. Scoring x, all paths $P(x) = \sum_\pi P(x,\pi)$ Prob of emissions, over all paths ✓
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_\pi P(x,\pi)$ Most likely path ✓	4. Posterior decoding $\pi\text{^} = \{\pi_i \mid \pi_i = \text{argmax}_k \Sigma_\pi P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point. ✓
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_\Lambda P(x,\pi\mid\Lambda)$ ✓ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_\Lambda \max_\pi P(x,\pi\mid\Lambda)$ ✓ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_\Lambda \Sigma_\pi P(x,\pi\mid\Lambda)$ Baum-Welch training, over all paths

Expectation Maximization (EM)

Widely used in Comp Bio. This course alone:
- **HMMs: Baum Welch**
- **Expression Clustering: Fuzzy K-Means**
- **Regulatory motif discovery: MEME**
- **Systems genetics: GWAS priors/enrichments**
- **Phylogenomics: Estimate branch lengths/rates**

The basic idea is always the same:

1. Use model to estimate missing data (E step)

2. Use estimate to update model (M step)

3. Repeat until convergence

EM is a general approach for learning models (ML estimation) when there is "missing data"

1. **Initialize parameters randomly**
2. **E Step Estimate expected probability of hidden labels, Q, given current (latest) parameters and observed (unchanging) sequence**

$$Q = P(Labels | S, params^{t-1})$$

3. **M Step Choose new maximum likelihood parameters over probability distribution Q, given current probabilistic label assignments**

$$params^{t} = \arg\max_{params} E_{Q}\left[\log P(S, labels \mid params^{t-1})\right]$$

4. **Iterate**

P(S|Model) *guaranteed* to increase each iteration

EM for HMM Learning: Annotation ⇔ Parameters

Starting with our best guess of a model M, parameters θ:

Given $x = x_1 \ldots x_N$

for which the true $\pi = \pi_1 \ldots \pi_N$ is unknown,

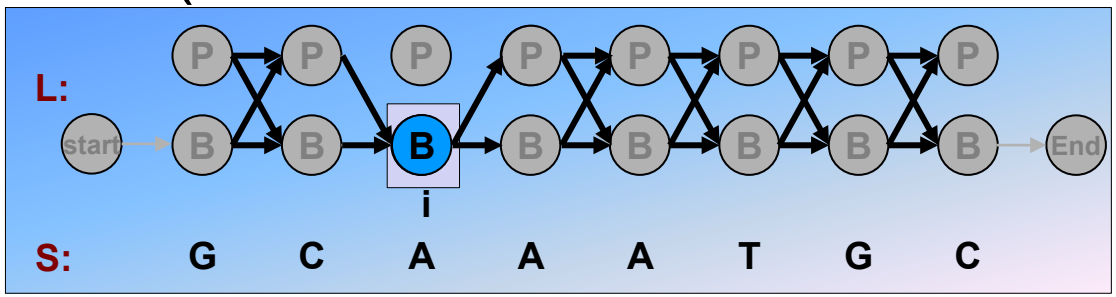
We can get to a provably more likely parameter set θ

Principle: EXPECTATION MAXIMIZATION

1. **Expected** annotations all-paths parse w/ current params (**E step**)
2. **Max**-likelihood params A_{kl}, E_k using this all-paths parse (**M step**)
3. Repeat 1 & 2, until convergence

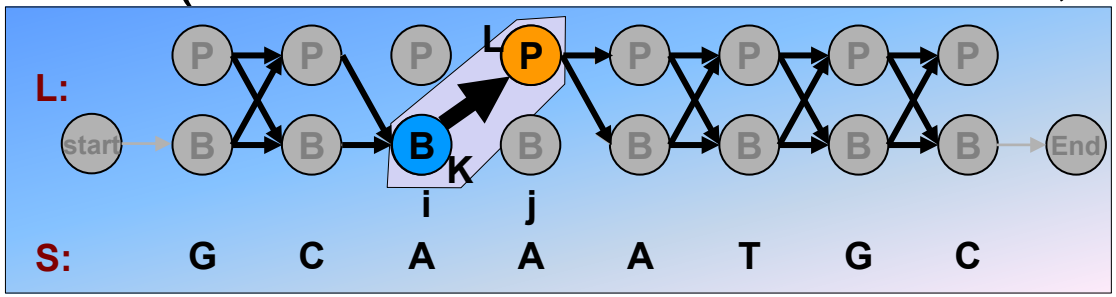
Max likelihood parameters | all-paths parse (M step)

(Sum over all emissions from k, at any time step i)



$$E_k(b) = [1/P(x)] \sum_{\{i \mid x_i = b\}} f_k(i)\, b_k(i)$$

(Sum over all k→l transitions, at any time step i)

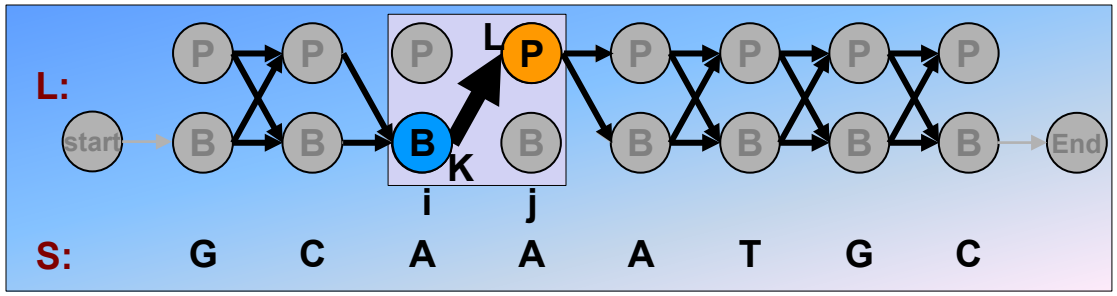


$f_k(i)$ a_{kl} $e_l(x_{i+1})$ $b_l(i+1)$

$$A_{kl} = \sum_i P(\pi_i = k, \pi_{i+1} = l \mid x, \theta) = \sum_i \frac{\quad}{P(x \mid \theta)}$$

Max likelihood parameters | all-paths parse (M step)

Derivation:
To estimate $\mathbf{A_{kl}}$:
At each position i:



Find probability transition k→l is used:

$$P(\pi_i = k, \pi_{i+1} = l \mid x) = [1/P(x)] \times P(\pi_i = k, \pi_{i+1} = l, x_1 \ldots x_N) = Q/P(x)$$

where $Q = P(x_1 \ldots x_i, \pi_i = k, \pi_{i+1} = l, x_{i+1} \ldots x_N) =$

$= P(\pi_{i+1} = l, x_{i+1} \ldots x_N \mid \pi_i = k)\, P(x_1 \ldots x_i, \pi_i = k) =$

$= P(\pi_{i+1} = l, x_{i+1} x_{i+2} \ldots x_N \mid \pi_i = k)\, f_k(i) =$

$= P(x_{i+2} \ldots x_N \mid \pi_{i+1} = l)\, P(x_{i+1} \mid \pi_{i+1} = l)\, P(\pi_{i+1} = l \mid \pi_i = k)\, f_k(i) =$

$= b_l(i+1)\, e_l(x_{i+1})\, a_{kl}\, f_k(i)$

So:
$$\mathbf{P(\pi_i = k, \pi_{i+1} = l \mid x, \theta) = \frac{f_k(i)\, a_{kl}\, e_l(x_{i+1})\, b_l(i+1)}{P(x \mid \theta)}}$$

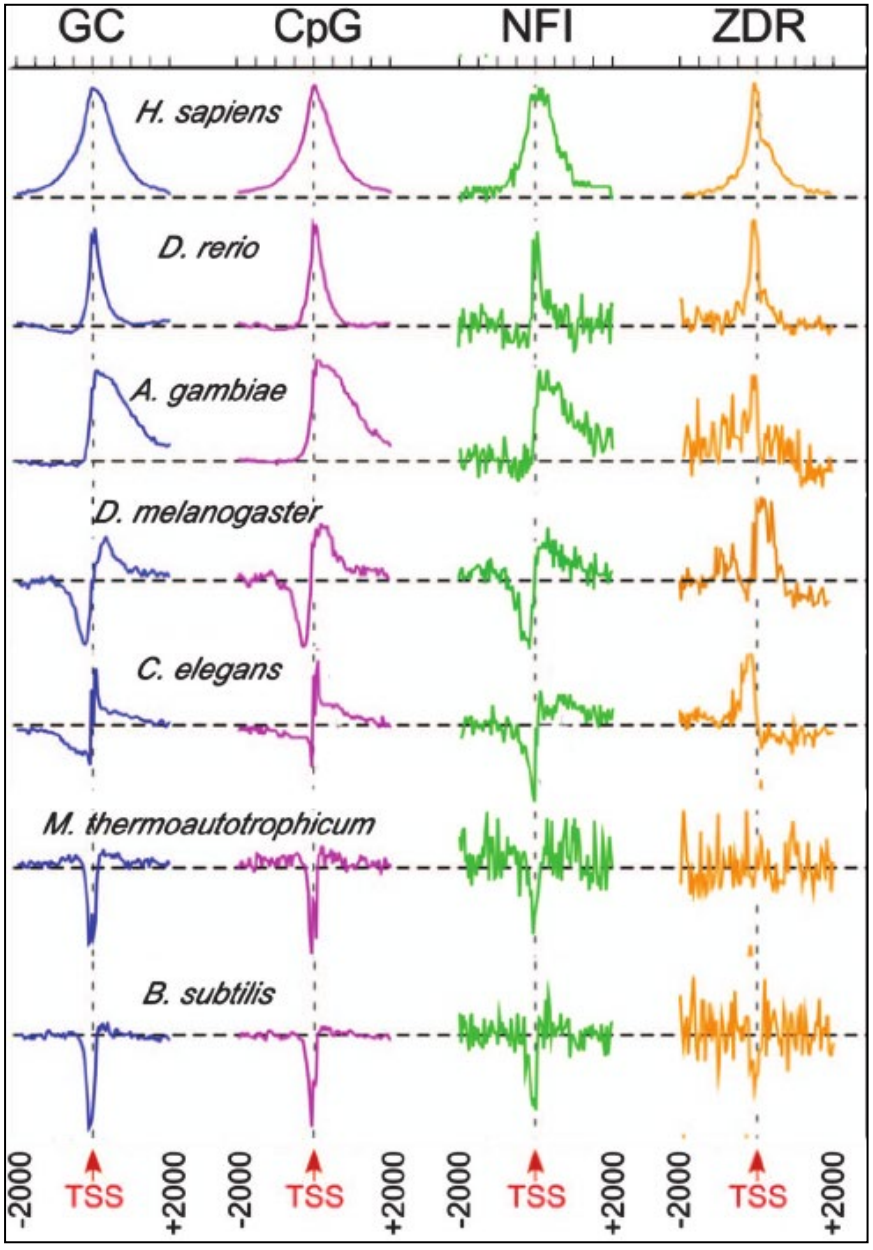
(For one such transition, at time step i→i+1)

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ ✓ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \sum_{\pi} P(x,\pi)$ ✓ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_{\pi} P(x,\pi)$ ✓ Most likely path	4. Posterior decoding $\pi\hat{} = \{\pi_i \mid \pi_i = \text{argmax}_k \sum_{\pi} P(\pi_i = k \mid x)\}$ ✓ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_{\Lambda} P(x,\pi \mid \Lambda)$ ✓ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_{\Lambda} \max_{\pi} P(x,\pi \mid \Lambda)$ ✓ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_{\Lambda} \sum_{\pi} P(x,\pi \mid \Lambda)$ ✓ Baum-Welch training, over all paths

HMM Foundations, Parsing, Decoding, Learning

1. HMM basics, evaluation, parsing, posterior decoding
   - Observations, Models, Bayes' rule, Bayesian inference
   - Markov Chains and Hidden Markov Models
   - Calculating joint probability of one (seq,parse) $P(x,\pi)$
   - Viterbi algorithm: Find best parse $\pi^* = \text{argmax}_\pi P(x,\pi)$
   - Forward algorithm: Find total $P(x)$, sum over all paths
   - Posterior Decoding: Most likely state π_i (over all paths)
2. Learning (ML training, Baum-Welch, Viterbi training)
   - Supervised: Find $e_i(.)$ and a_{ij} given labeled sequence
   - Unsupervised: given only x → annotation + params
3. Increasing the 'state' space / adding memory
   - Finding GC-rich regions vs. finding CpG islands
   - Gene structures GENSCAN, chromatin ChromHMM

Example: Detecting GC-rich regions: motivation



Model genome as two states:

- P: promoter
- B: background

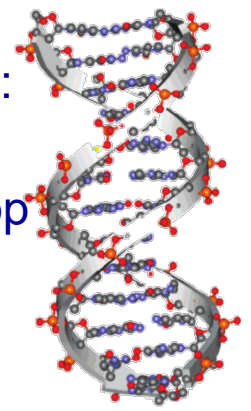
Model different nucleotide compositions

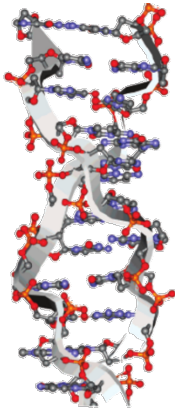
- Background:
  P(A) = P(T) = P(C) = P(G) = 25%
- Promoters
  P(A) = P(T) = 10%
  P(G) = P(C) = 40%

Note: generative model, P(A|promoter) etc
Then: reverse probabilities using Baye's rule

Model length distribution:

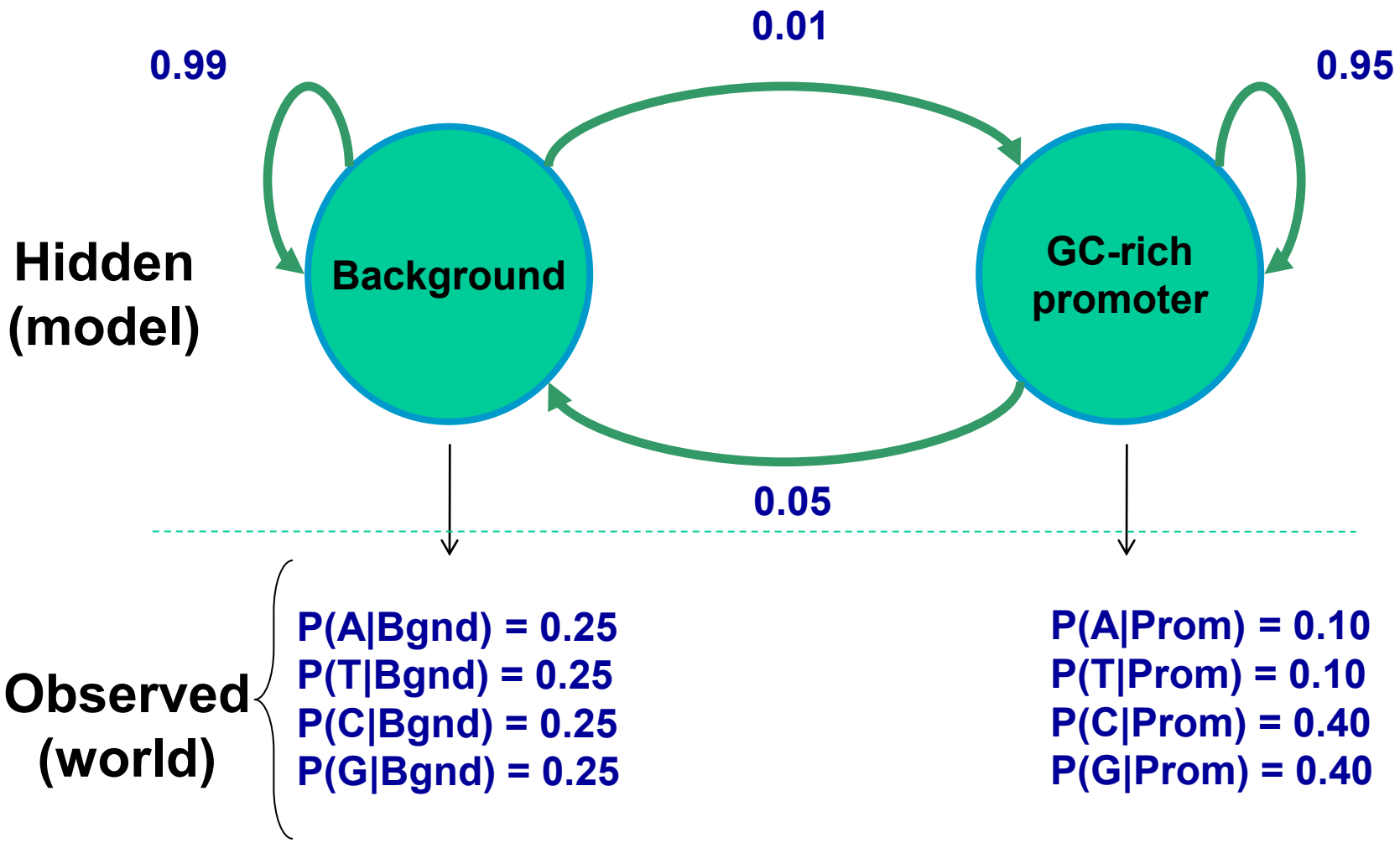
- Promoter: 20 bp
- Non-promoter: 100 bp


B-DNA


Z-DNA

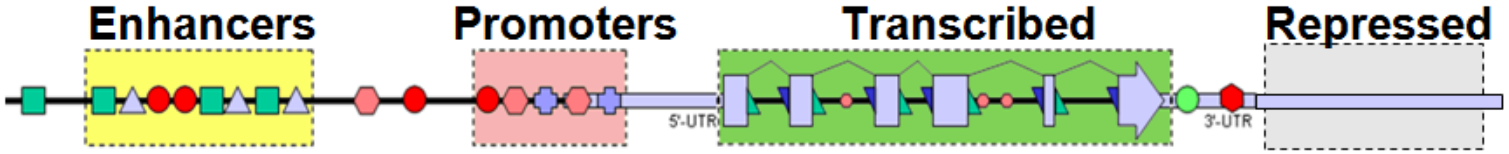
Modified from: Khuu et al PNAS 2007; NFI: Nuclear Factor I; ZDRs: Z-DNA Forming Regions

Detecting GC-rich regions: HMM architecture



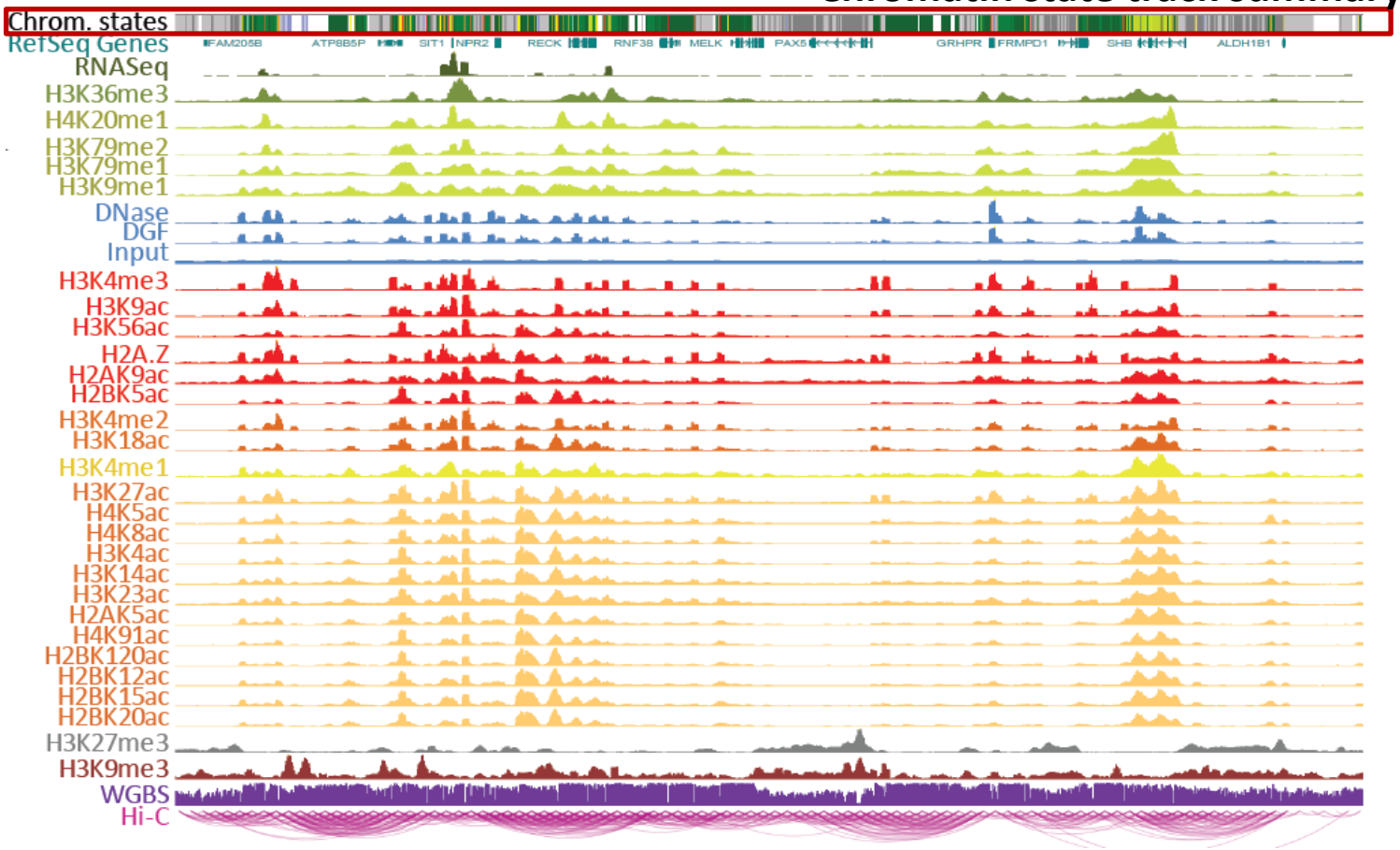
Slide credit: Serafim Batzoglou

Summarize multiple marks into chromatin states



Chromatin state track summary

30+ epigenomics marks



WashU Epigenome Browser

ChromHMM: multi-variate hidden Markov model

HMMs are used broadly for genome annotation

Application	Detection of GC-rich regions	Detection of conserved regions	Detection of protein-coding exons	Detection of protein-coding conservation	Detection of protein-coding gene structures	Detection of chromatin states
Topology / Transitions	2 states, different nucleotide composition	2 states, different conservation levels	2 states, different tri-nucleotide composition	2 states, different evolutionary signatures	~20 states, different composition/ conservation, specific structure	40 states, different chromatin mark combinations
Hidden States / Annotation	GC-rich / AT-rich	Conserved / non-conserved	Coding exon / non-coding (intron or intergenic)	Coding exon / non-coding (intron or intergenic)	First/last/middle coding exon,UTRs, intron.4/3, intergenic, *(+/- strand)	Enhancer / promoter / transcribed / repressed / repetitive
Emissions / Observations	Nucleotides	Level of conservation	Triplets of nucleotides	Nucleotide triplets, conservation levels	Codons, nucleotides, splice sites, start/stop codons	Vector of chromatin mark frequencies

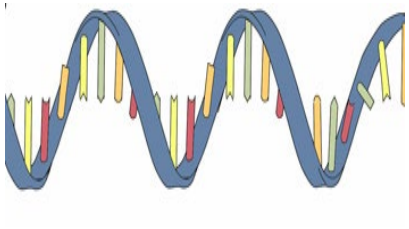
Examples of HMMs for genome annotation

Detection of conserved regions	Detection of GC-rich regions	Detection of CpG-rich regions	Detection of protein-coding exons	Detection of protein-coding gene structures	Detection of chromatin states	Detection of protein-coding conservation
2 states, different conservation levels	2 states, different nucleotide composition	8 states, 4 each +/-, different transition probabilities	2 states, different tri-nucleotide composition	~20 states, different composition/ conservation, specific structure	40 states, different chromatin mark combinations	2 states, different evolutionary signatures
Conserved / non-conserved	GC-rich / AT-rich	CpG-rich / CpG-poor	Coding exon / non-coding (intron or intergenic)	First/last/middle coding exon,UTRs, intron1/2/3, intergenic, *(+/- strand)	Enhancer / promoter / transcribed / repressed / repetitive	Coding exon / non-coding (intron or intergenic)
Level of conservation	Nucleotides	Di-Nucleotides	Triplets of nucleotides	Codons, nucleotides, splice sites, start/stop codons	Vector of chromatin mark frequencies	64x64 matrix of codon substitution frequencies
L2:alignmnt	L4:HMMs1	L5:HMMs2	L5:HMMs2	L5:HMMs2	L8:Epignmcs	L17:CompG

Genome Annotation

Genome Sequence

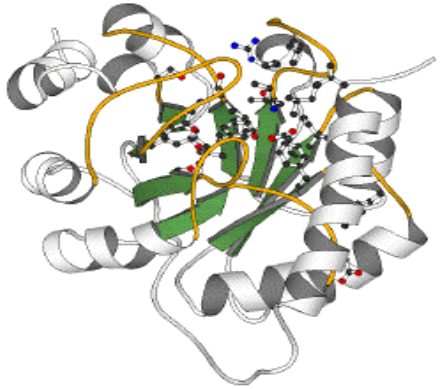
> HSCKIIBE, Human gene for casein kinase II subunit beta (EC 2.7.1.37).
ggggctgagatgtaaattagaggagctggagaggagtgcttcagagtttgggttgctttaagaaagggt
ggttccgaattctcccgtggttggagggccgaatgtgggaggagggaggataccagaggcagggaagga
gaacttgagctttactgacactgttctttttctagctgacgtgaagatgagcagctcagaggaggtgtc
ctggatttcctggttctgtgggctccgtggcaatgaattcttctgtgaagtgagttctcttcaacctcc
ctacttgccagcttcacatatcttcccaccagacgttccttcacatattccacttctacactgttctct
aaagcttttatgggagagagtgtaggtgaactagggagagacacaagtacttctgctgagttgggagtg
agaaacaagcacaacagatgcagttgtgttgatgataaggcatcacttagagcattttgcccaggtcaa
agatgaggattttgatatgggttccctcttggcttccatgtcctgacaggtggatgaagactacatcca
ggacaaatttaatcttactggactcaatgagcaggtccctcactatcgacaagctctagacatgatctt
ggacctggagcctggtgaggcaccctcagggttgttttgtgtgtgtgcgtgcactatttttctcttcaa
atctctattcacttgcctgaattttgccaaatttcctttggttctctgatttctttaaccccaaattca
tgctttattttgatcctccacctgactcttgtctagttttgtgacgtatatcacttgttctcatgtttt
tgcaagggtcagaagcccaggtttctgggtcccatgcccagatgttggatggggtaaggcccaaaagta
ggtgctaggcaaactgaatagcccgcagcccctggatatgggcagggcacctaggaaagctgaaaaaca
agtagttgcatttggccgggctgtggttcagatgaagaactggaagacaaccccaaccagagtgacctg
attgagcaggcagccgagatgctttatggattgatccacgcccgctacatcctaccaaccgtggcatc
gcccagatggtgaggcctctctgctcctacctgcctccttctgagcagtaagagacacaggttcctgca
gcaagaagtcatgtttaagccctgtttaaggaagctagctgagaagaggggaagaaccccagaacttgg
ccctgccctaatttggaagaaaggcaacacagaagtttgagagcccatctagtccagagaaggggggcct
ctggacagagttggaaggagtgccgacagagttggtatgggttgggctgcgaagggagttgcctcttct
ttacatctacctgccaacccctccattgtattcacctcagttggaaaagtaccagcaaggagactttg
gttactgtcctcgtgtgtactgtgagaaccagccaatgcttcccattGgtgagtgttgaagaagggaaa
ggaaagcaccgtgtggcagtcttatgggaaggagttggggctcaacacattggagcctgagtcctgagg
ggaggttaggtaggaatagggggatacctggcctgctgagtctggctgtctcccaggcctttcagacat
cccaggtgaagccatggtgaagctctactgccccaagtgcatggatgtgtacacacccaagtcatcaag
acaccatcacacggatggcgcctacttcggcactggtttccctcacatgctcttcatggtgcatcccga
gtaccggcccaagagacctgccaaccagtttgtgcccaggtagggagcagggagagtcattaagggtca
aaggaaaggcccaagatcccccagagagggggaggacagggcatggccctttcttgaggtctgcttctcc
cagaatcagggcatctccctgctgagtgactgtgggaaagttatttgattatctgtgcttgagttacct
tattgtagaatgttcttgagctgagaagttgggaaccacgaggctttagctctgagcaggtccatagag
gagctcaggtgggagaggtgggaatgcaggtgactggcagggcctggatggggctcatgctgctgcctct
ctgacctctgccctggcctaggctctacggtttcaagatccatccgatggcctaccagctgcagctcca
agccgccagcaacttcaagagcccagtcaagacgattcgctgattccctccccccacctgtcctgcagtc
tttgtcttttccttttcttttttgccacccctttcaggaaccctgtatggtttttagtttaaattaaagga
gtcgttatcgtggtgggaatatgaaataaagtagaagaaaaggccatgagctagtctgctggtgcttgc
tggaaggggtggagcgtggccatggaaatcgggctccacggcccagggatgg



RNA

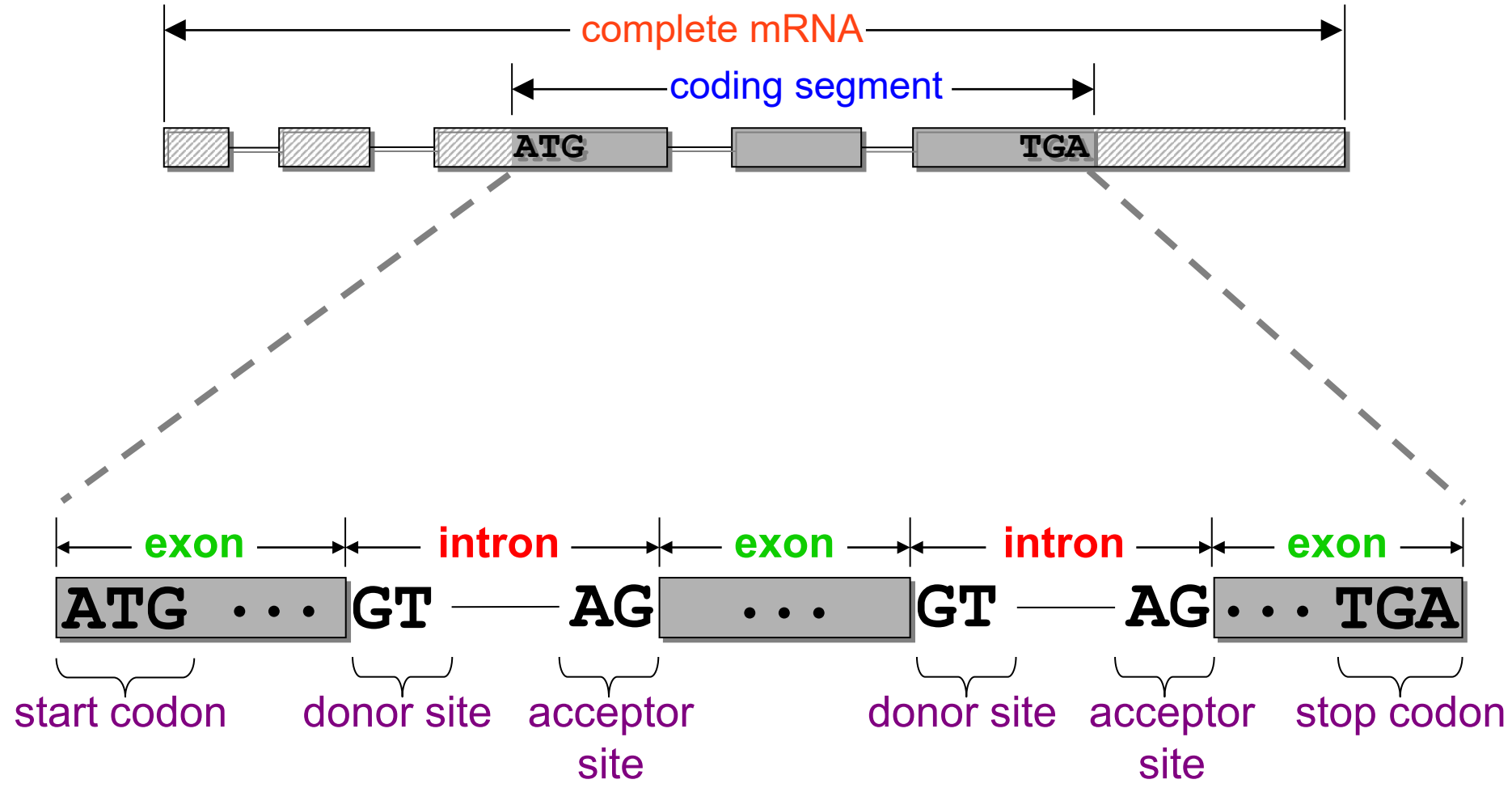
Transcription

Translation

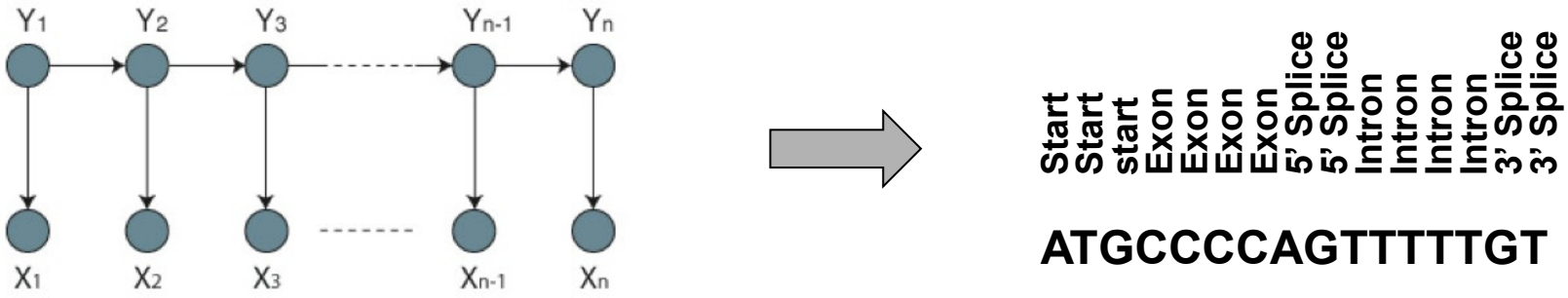


Protein

Eukaryotic Gene Structure



Gene Prediction with HMM

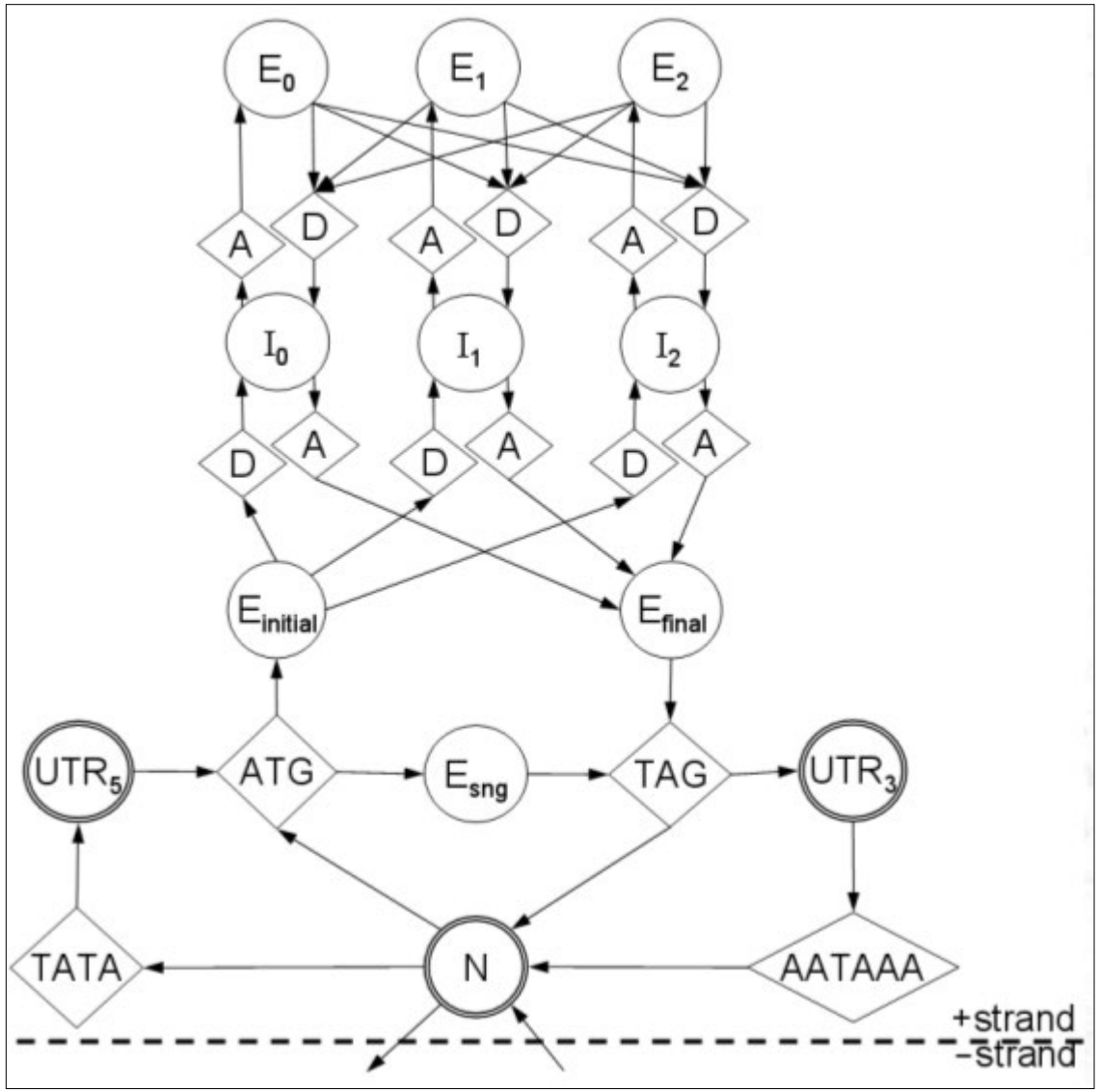


Model of joint distribution P(Y,X) = P(Labels,Seq)

For gene prediction, we are given X…

How do we select a Y efficiently?

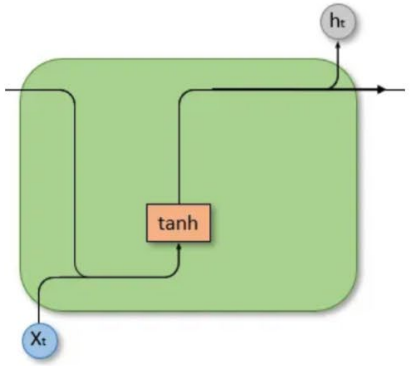
HMM architecture matters: Protein-coding genes



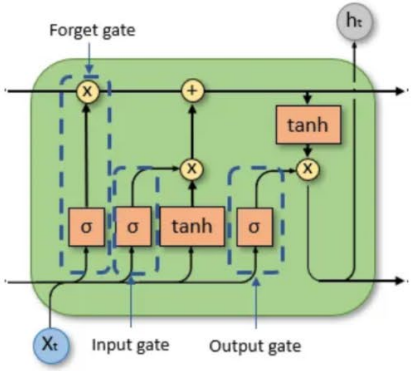
- Gene vs. Intergenic
- Start & Stop in/out
- UTR: 5' and 3' end
- Exons, Introns
- Remembering frame
  - E0,E1,E2
  - I0,I1,I2
- Sequence patterns to transition between states:
  - ATG, TAG, Acceptor/Donor, TATA, AATAA

Sequential models in the age of deep learning

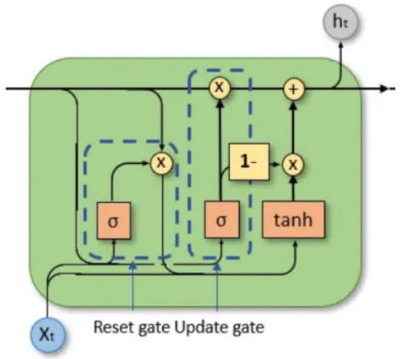
RNN



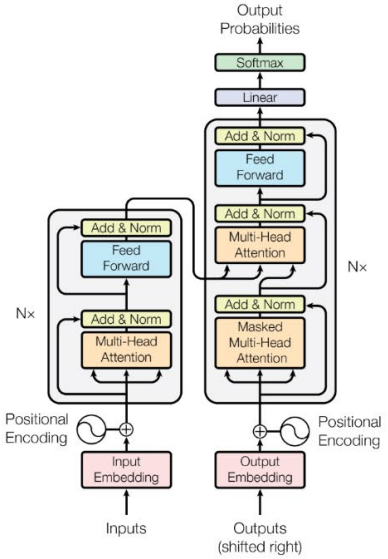
LSTM



GRU

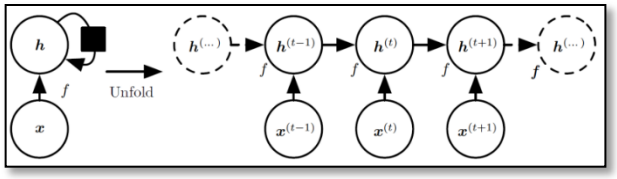


Transformers

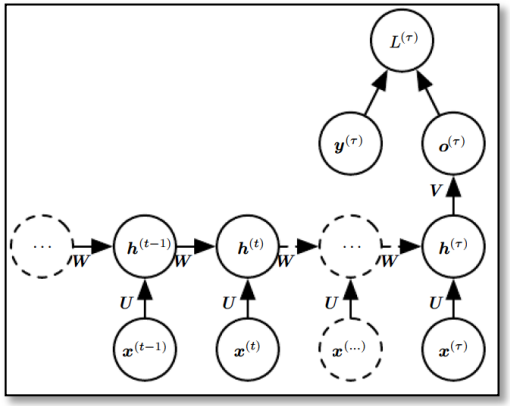


Recurrent Neural Network (RNN)	Long Short Term Memory (LSTM)	Gated Recurrent Unit (GRU)	Transformers
RNNs are foundational sequence models that process sequences iteratively, using the output from the previous step as an input to the current step.	LSTMs are an enhancement over standard RNNs, designed to better capture long-term dependencies in sequences.	GRUs are a variation of LSTMs with a simplified gating mechanism.	Transformers move away from recurrence and focus on self-attention mechanisms to process data in parallel

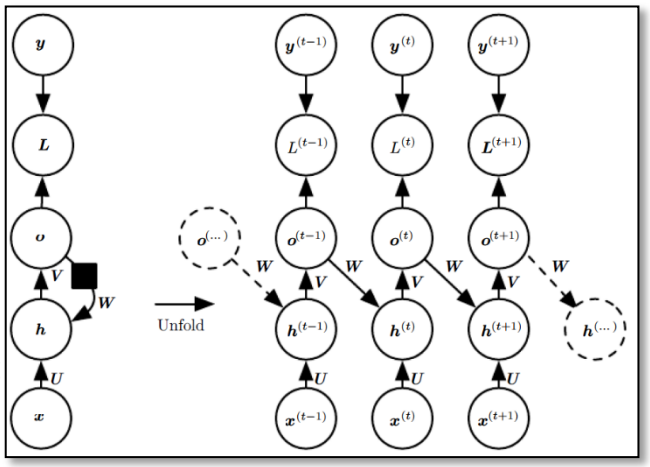
Different RNN remembering architectures



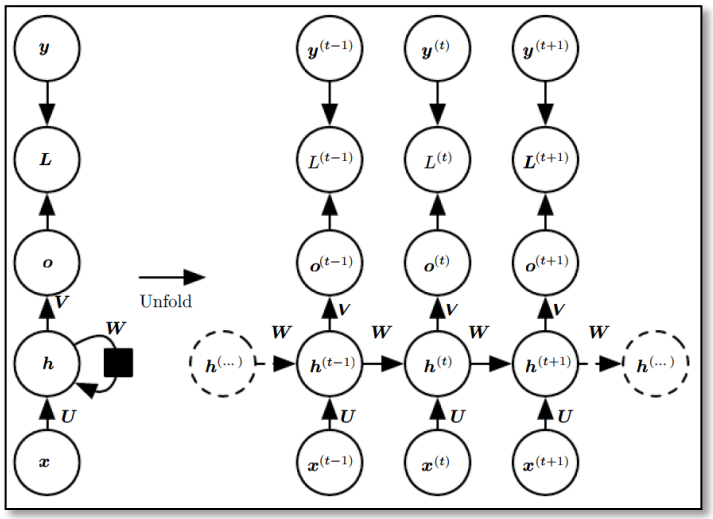
Recurrent network with no outputs



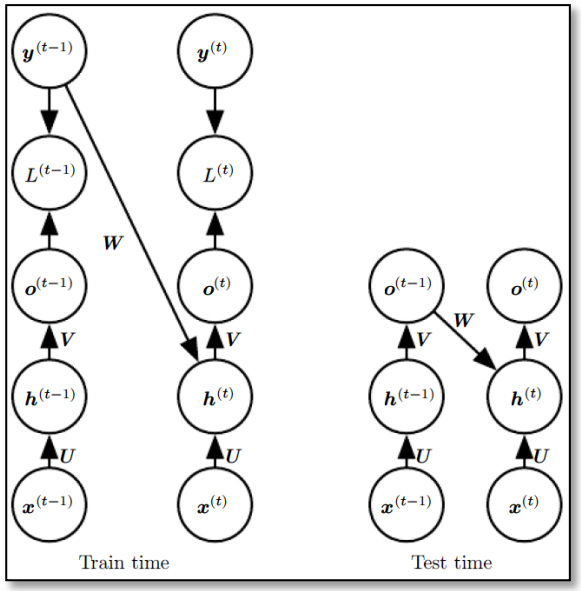
Single output after entire sequence



o: output, y: target, L: loss
Memory: $o^{(t-1)} \rightarrow h^{(t)}$.
Only train sequentially



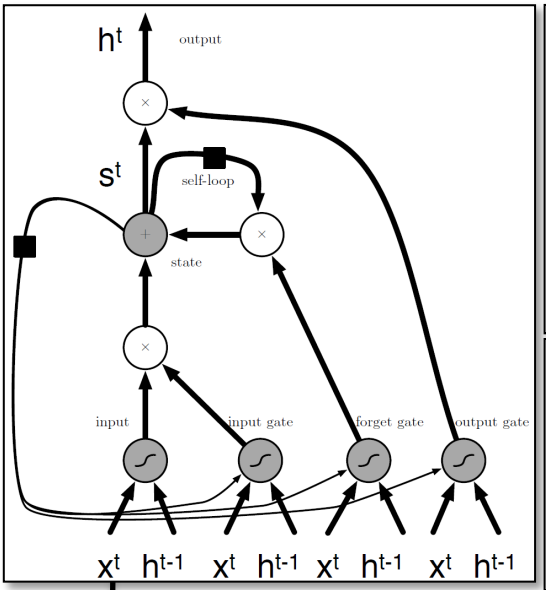
o: output, y: target, L: loss
Memory: $h^{(t-1)} \rightarrow h^{(t)}$



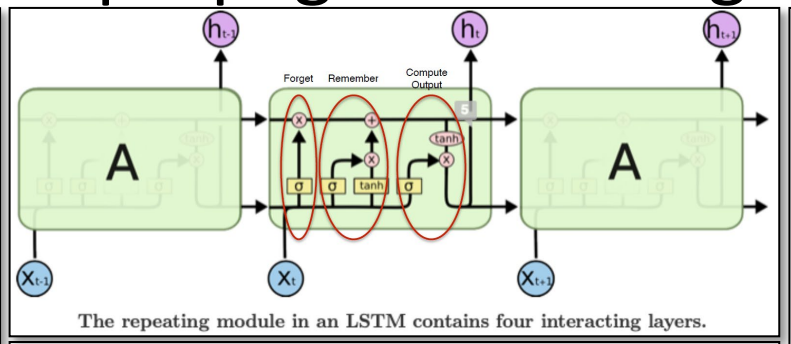
Teacher-forcing: train from **y** and **x** in parallel

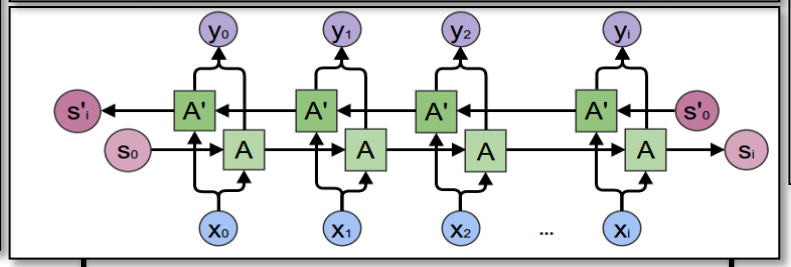
- Recurrent networks ⇔ inter-time dependencies, sharing parameters btw time steps
- Such dependencies include: value of hidden units / outputs of the previous stage
  - Examples: speech understanding, language translation
- We can unfold a simple one layer network to handle sequential data with all parameters shared between time points
- We can train an RNN based upon data likelihood as before
- **Problem 1:** Vanishing and exploding gradients during training. Hard to encode long-range context, need memory modules [**Soln: LSTMs**]
- **Problem 2**: Need to train sequentially (cuz output of previous is input of next) ➔ slow [**Soln: Transformers**]

RNN LSTM Backpropagation through a memory cell

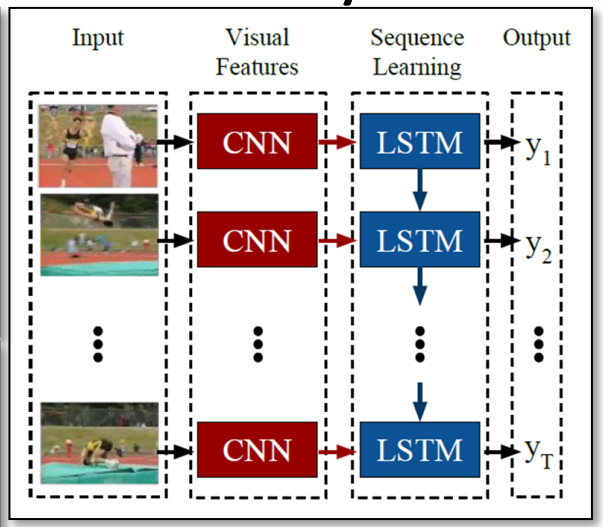


LSTMs: Gated RNNs





Bidirectional LSTMs



CNNs and LSTMs can be combined

keep 0 | keep 1 | keep 1 | keep 0

1.7 | 1.7 | 1.7

write 1 | read 0 | write 0 | read 0 | write 0 | read 1

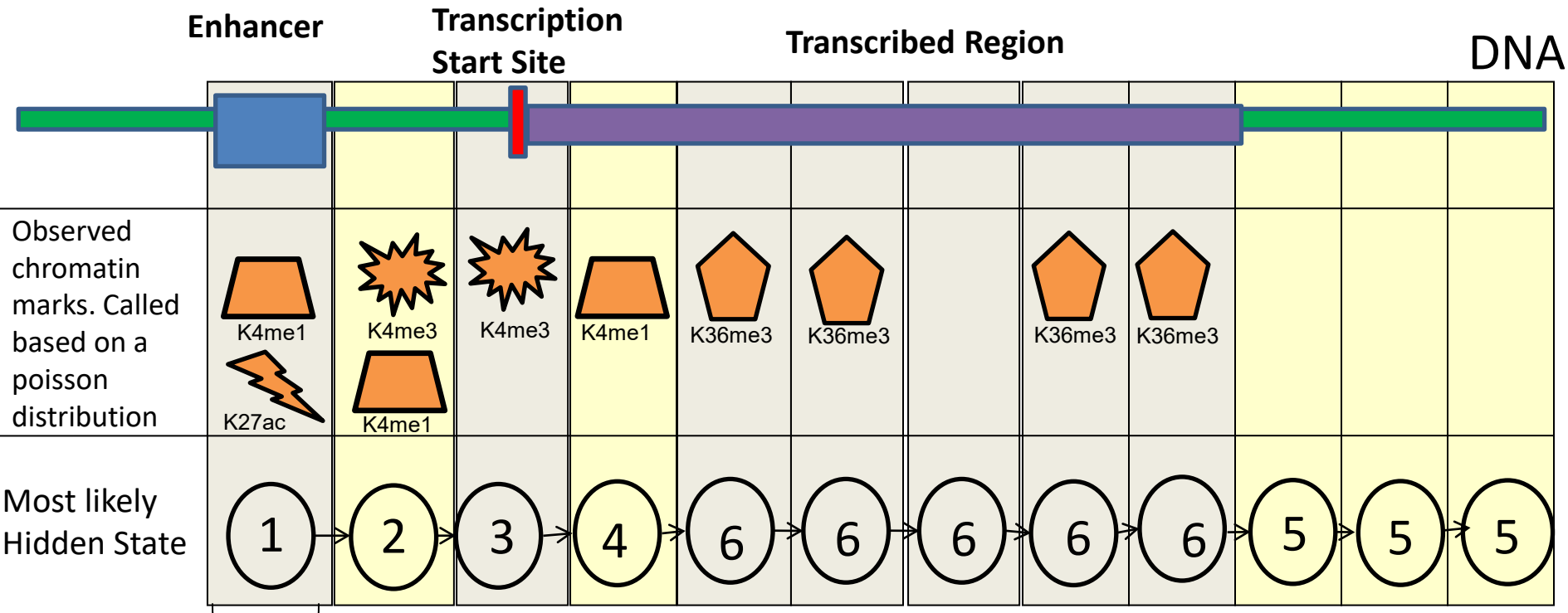
1.7 | 1.7

time →

Examples of HMMs for genome annotation

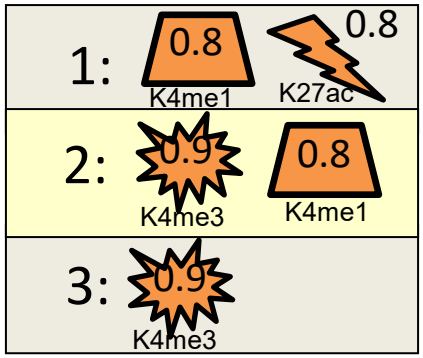
Detection of conserved regions	Detection of GC-rich regions	Detection of CpG-rich regions	Detection of protein-coding exons	Detection of protein-coding gene structures	Detection of chromatin states	Detection of protein-coding conservation
2 states, different conservation levels	2 states, different nucleotide composition	8 states, 4 each +/-, different transition probabilities	2 states, different tri-nucleotide composition	~20 states, different composition/ conservation, specific structure	40 states, different chromatin mark combinations	2 states, different evolutionary signatures
Conserved / non-conserved	GC-rich / AT-rich	CpG-rich / CpG-poor	Coding exon / non-coding (intron or intergenic)	First/last/middle coding exon,UTRs, intron1/2/3, intergenic, *(+/- strand)	Enhancer / promoter / transcribed / repressed / repetitive	Coding exon / non-coding (intron or intergenic)
Level of conservation	Nucleotides	Di-Nucleotides	Triplets of nucleotides	Codons, nucleotides, splice sites, start/stop codons	Vector of chromatin mark frequencies	64x64 matrix of codon substitution frequencies
L2:alignmnt	L4:HMMs1	L5:HMMs2	L5:HMMs2	L5:HMMs2	L8:Epignmcs	L17:CompG

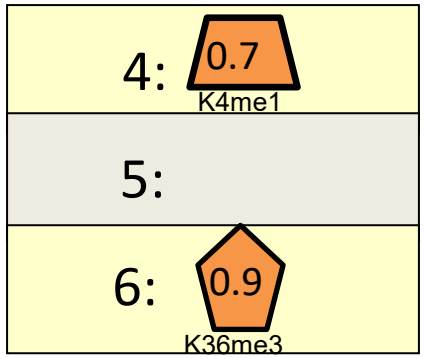
Multivariate HMM for Chromatin States



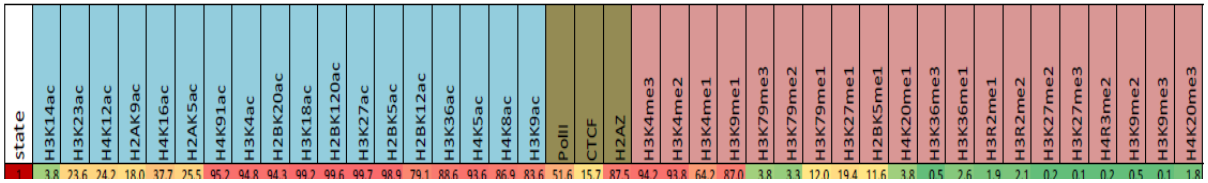
200bp intervals

High Probability Chromatin Marks in State





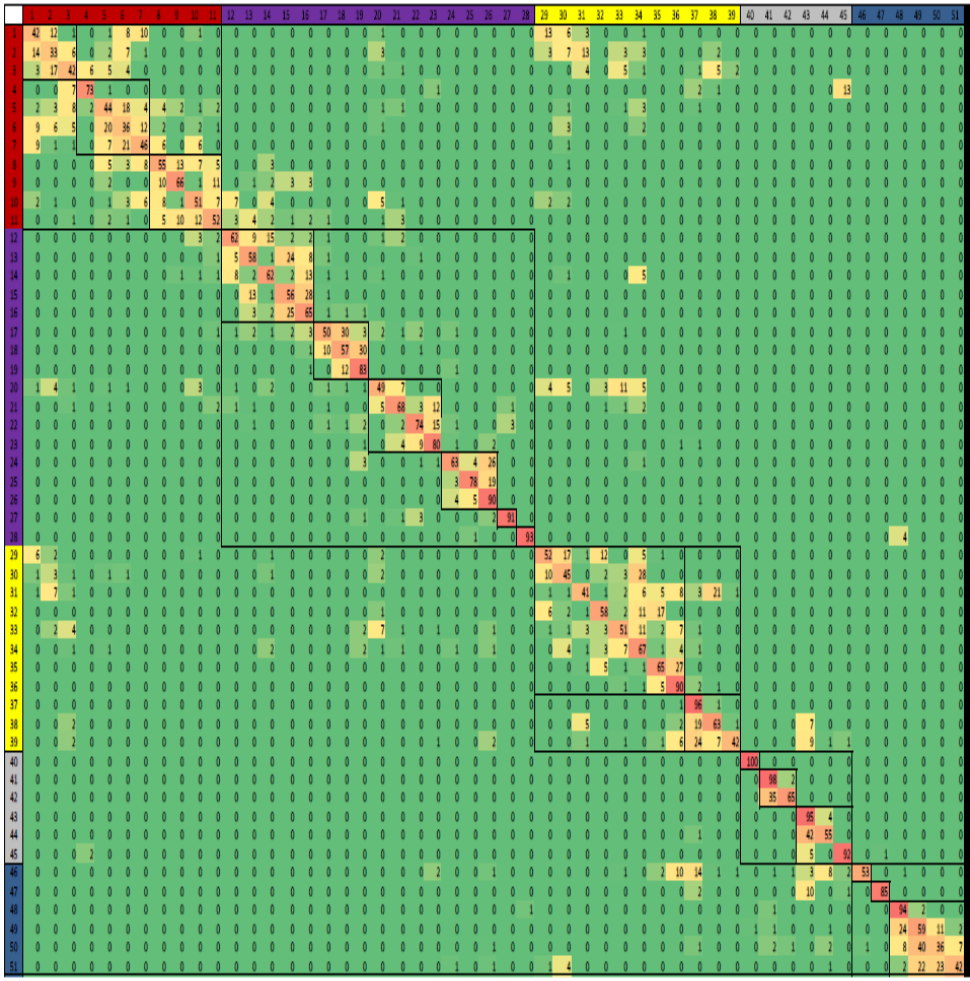
All probabilities are learned from the data



Ernst and Kellis

Nature Biotech 2010

Chromatin State: Emission & Transition Matrices



- **Emission matrix $e_k(x_i)$**
  - Multi-variate HMM
  - Emits vector of values
- **Transition matrix a_{kl}**
  - Learn spatial relationships
  - No a-priori ‘gene’ structure

Ernst and Kellis, Nature Biotech 2010, Nature 2011, Nature Methods 2012

Design Choice

- How to model the emission distribution
  - Model the signal directly
  - Locally binarize the data
- For *M* input marks each state *k* has a vector of $(p_{k1},\ldots,p_{kM})$ of parameters for independent Bernoulli random variables which determine the emission probability for an observed combination of marks

Data Binarization

- Leads to biologically interpretable models that can be robustly learned
- Let c_{ij} be the number of reads for mark *i.* mapping to bin *j.* λ_i be the average number of reads mapping to a bin for modification *i*. The input for feature *i* becomes *'1'* if

$$P(X>c_{ij})<10^{-4}$$

where *X* is a Poisson random variable with mean λ_i

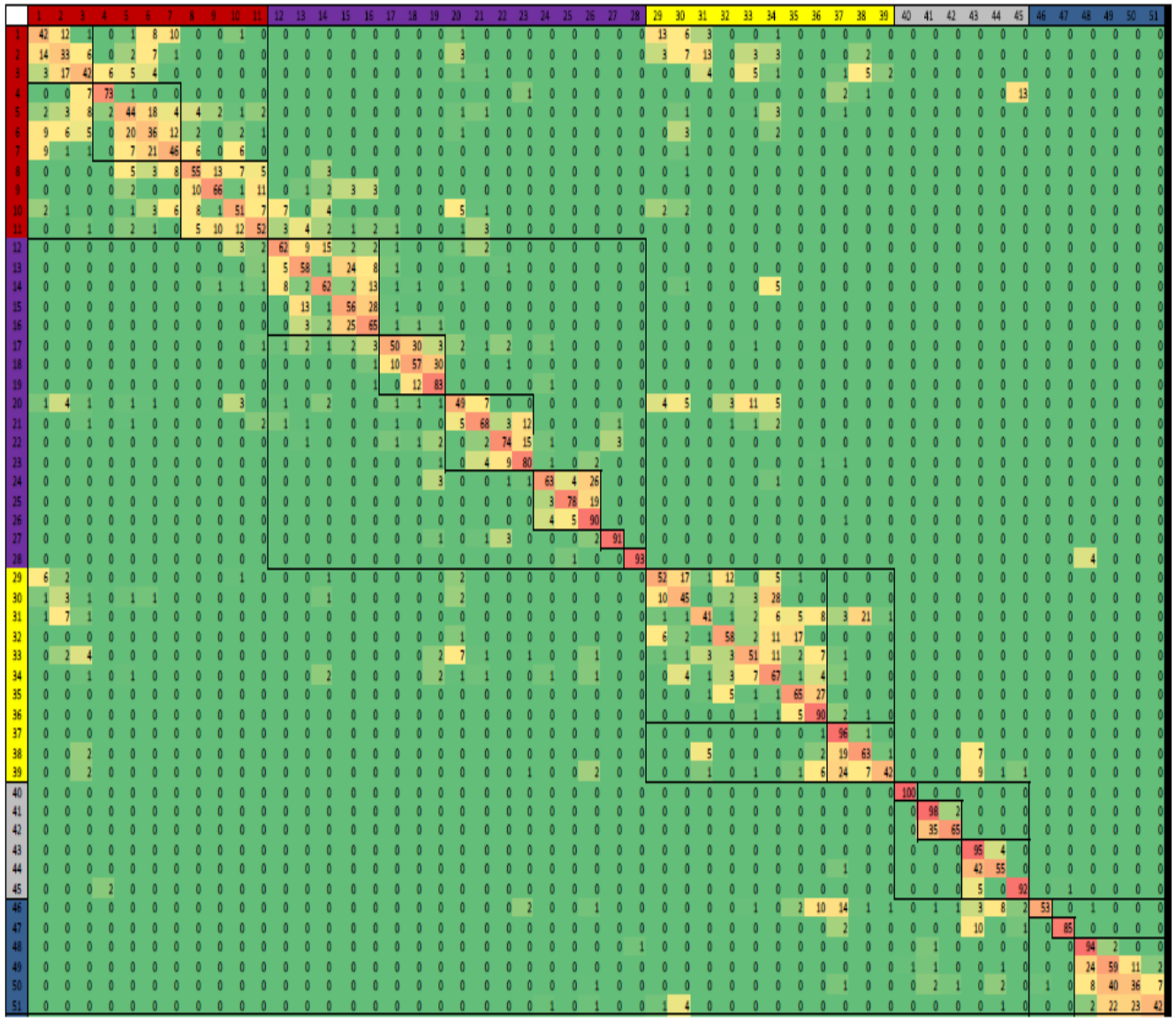
Emission Parameter Matrix $e_k(\vec{x}_i)$

state	H3K14ac	H3K23ac	H4K12ac	H2AK9ac	H4K16ac	H2AK5ac	H4K91ac	H3K4ac	H2BK20ac	H3K18ac	H2BK120ac	H3K27ac	H2BK5ac	H2BK12ac	H3K36ac	H4K5ac	H4K8ac	H3K9ac	PolII	CTCF	H2AZ	H3K4me3	H3K4me2	H3K4me1	H3K9me1	H3K79me3	H3K79me2	H3K79me1	H3K27me1	H2BK5me1	H4K20me1	H3K36me3	H3K36me1	H3R2me1	H3R2me2	H3K27me2	H3K27me3	H4R3me2	H3K9me2	H3K9me3	H4K20me3
1	3.8	23.6	24.2	18.0	37.7	25.5	95.2	94.8	94.3	99.2	99.6	99.7	98.9	79.1	88.6	93.6	86.9	83.6	51.6	15.7	87.5	94.2	93.8	64.2	87.0	3.8	3.3	12.0	19.4	11.6	3.8	0.5	2.6	1.9	2.1	0.2	0.1	0.2	0.5	0.1	1.8
2	2.5	17.5	9.2	3.2	5.9	6.3	44.6	44.4	47.0	73.2	74.1	85.9	71.2	22.1	33.5	61.9	63.3	35.4	18.1	10.9	91.2	86.7	90.4	66.9	78.3	2.4	2.2	7.9	17.6	8.7	2.3	0.6	2.2	1.7	1.5	0.4	0.5	0.2	0.4	0.1	1.4
3	0.5	5.8	1.8	1.0	0.9	1.2	12.3	9.5	8.8	22.6	21.3	22.8	12.1	2.2	4.2	8.4	12.8	7.1	11.2	16.3	77.1	93.9	80.3	45.6	74.2	1.5	1.3	4.6	4.3	7.0	8.8	0.2	1.4	2.1	1.5	0.2	2.1	0.1	0.1	0.1	1.2
4	0.1	0.8	0.1	0.4	0.3	0.2	1.5	0.9	0.7	2.1	2.1	0.5	0.2	0.1	0.1	0.1	0.3	0.3	1.9	6.2	19.0	77.9	20.8	21.1	26.4	0.3	0.0	0.1	0.0	1.3	14.9	0.1	0.2	1.4	0.9	0.0	10.2	0.1	0.1	0.4	1.3
5	0.0	0.2	0.8	1.3	2.0	0.4	26.6	12.6	6.7	15.8	23.1	26.8	24.3	1.5	4.3	0.9	1.4	7.7	53.5	20.6	21.8	87.0	11.2	2.7	15.7	6.3	3.8	2.5	0.0	5.5	14.2	0.0	0.3	0.2	0.6	0.0	0.0	0.0	0.1	0.0	1.5
6	0.1	1.8	3.6	6.9	6.1	1.9	74.5	63.5	53.0	75.7	84.3	89.4	86.7	20.8	41.5	20.5	21.6	62.7	69.2	25.5	61.2	98.3	37.4	7.1	40.3	5.3	2.7	5.6	0.6	6.0	11.6	0.0	0.5	0.4	0.9	0.0	0.0	0.0	0.0	0.1	1.6
7	1.2	8.7	20.6	43.0	53.7	9.8	98.7	98.6	95.7	99.4	99.9	99.9	99.9	76.5	93.3	81.8	76.6	99.2	88.0	26.9	77.1	99.7	38.3	2.2	37.9	32.1	24.9	14.0	2.6	6.5	16.2	0.1	1.0	0.5	1.2	0.0	0.0	0.0	0.0	0.1	1.9
8	1.2	12.7	5.2	11.9	5.6	6.8	56.9	56.1	37.5	52.4	69.8	89.1	85.5	21.3	24.8	16.7	10.3	60.8	62.1	12.0	31.4	96.7	51.7	14.9	45.3	86.3	80.1	23.8	1.7	6.7	42.2	4.5	0.4	1.1	0.9	0.0	0.0	0.0	0.0	0.0	0.5
9	0.5	7.2	3.0	1.1	0.5	2.1	4.0	7.4	2.4	2.5	11.6	35.3	28.0	2.7	2.8	2.0	1.8	8.6	34.7	4.2	4.5	79.2	41.5	23.8	36.1	86.0	82.6	12.0	1.9	6.7	43.5	7.4	0.2	0.6	0.7	0.0	0.0	0.0	0.0	0.2	0.4
10	4.1	24.8	13.1	17.5	24.4	37.0	90.4	88.6	82.0	89.8	95.5	97.0	95.1	54.0	56.4	67.2	45.7	55.7	46.6	10.2	40.8	84.6	92.3	91.4	92.8	67.1	67.2	63.4	29.2	53.8	65.2	4.5	6.8	5.7	3.4	0.1	0.0	0.3	0.1	0.0	1.0
11	1.6	21.0	3.9	3.8	3.2	8.4	28.0	26.1	14.5	22.6	37.6	56.8	47.4	6.0	6.4	13.5	8.8	18.4	30.9	6.9	20.1	92.4	93.5	94.3	94.5	73.8	74.2	55.1	24.0	57.2	79.9	8.9	6.6	4.4	2.5	0.1	0.0	0.2	0.1	0.1	0.7
12	3.6	17.0	8.9	2.2	14.1	34.9	60.3	51.0	38.8	35.6	56.6	53.3	55.3	11.9	11.5	30.0	15.4	1.7	18.0	2.7	0.7	5.4	58.2	96.0	77.8	87.4	87.0	76.3	41.6	79.8	82.2	13.4	3.1	6.4	3.6	0.3	0.0	0.7	0.2	0.1	0.4
13	1.2	10.8	3.6	0.7	2.5	7.4	9.1	6.5	2.6	2.5	6.5	7.7	5.5	0.5	1.0	5.5	3.3	0.3	10.2	1.5	0.0	2.4	56.2	83.7	82.9	92.7	92.6	64.4	38.2	80.0	89.8	12.0	2.4	3.3	2.1	0.3	0.0	0.4	0.2	0.1	0.3
14	0.7	5.3	7.9	1.0	2.4	18.0	19.8	20.7	14.6	7.9	24.5	20.1	21.8	6.7	6.6	11.3	8.6	0.3	6.9	1.7	2.1	1.3	8.0	33.0	16.3	61.8	62.1	37.9	9.7	14.3	18.3	9.7	0.2	1.7	1.2	0.2	0.0	0.1	0.3	0.4	1.0
15	0.2	1.9	2.9	0.3	0.3	1.5	0.8	1.3	0.3	0.2	1.2	1.5	1.2	0.2	0.4	0.7	1.2	0.1	5.0	0.7	0.0	0.2	11.0	17.3	29.3	84.7	82.2	33.1	8.0	26.4	56.2	5.2	0.2	0.7	0.9	0.1	0.0	0.1	0.1	0.6	0.2
16	0.0	0.3	0.4	0.1	0.1	0.5	0.4	0.6	0.2	0.1	0.5	0.4	0.3	0.1	0.2	0.2	0.3	0.0	0.6	0.3	0.1	0.1	1.2	2.8	3.9	29.0	25.2	8.5	0.7	1.3	7.9	1.2	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.4	0.1
17	1.2	9.8	2.8	0.9	2.4	7.8	6.8	6.1	2.3	4.0	3.5	8.4	3.5	0.3	1.0	9.3	6.6	0.5	3.5	1.1	0.4	1.0	52.3	68.9	83.7	22.7	23.5	61.2	48.3	64.7	57.3	21.8	2.8	4.9	2.0	1.0	0.1	0.5	0.5	0.1	0.5
18	0.3	2.6	2.1	0.4	0.5	2.4	1.4	1.6	0.5	0.6	0.9	1.6	0.7	0.2	0.5	1.9	1.9	0.1	1.6	0.7	0.1	0.1	10.4	9.7	29.1	15.0	13.5	34.0	14.9	19.9	21.4	10.6	0.5	1.2	0.9	0.5	0.1	0.1	0.3	0.2	0.2
19	0.1	0.3	0.5	0.2	0.1	0.5	0.1	0.3	0.0	0.0	0.1	0.2	0.1	0.1	0.1	0.1	0.3	0.0	0.4	0.3	0.0	0.0	0.5	0.2	2.0	4.4	2.9	7.3	1.0	1.0	2.3	1.4	0.0	0.2	0.3	0.1	0.0	0.0	0.1	0.4	0.1
20	2.5	10.7	5.4	3.1	9.9	26.2	58.2	48.8	41.7	49.3	54.8	57.1	51.5	13.0	14.1	31.6	21.7	4.0	14.5	6.7	15.6	20.9	56.8	97.1	70.5	5.4	5.6	33.8	31.1	52.6	38.4	7.4	3.4	6.9	3.8	0.5	0.1	0.7	0.3	0.0	1.0
21	0.2	0.8	2.0	1.3	7.2	11.3	32.3	15.4	11.5	5.7	18.6	8.4	12.4	2.1	1.2	3.2	2.3	0.5	15.1	6.5	0.6	4.7	17.0	68.7	33.3	8.7	6.4	37.2	9.6	65.4	87.7	9.2	1.3	7.1	5.3	0.1	0.2	1.4	0.0	0.0	0.8
22	0.1	0.1	1.1	0.6	6.2	2.4	7.8	1.8	0.6	0.1	1.5	0.8	1.5	0.1	0.1	0.7	0.7	0.1	8.5	1.0	0.0	0.0	5.3	8.4	14.6	15.5	9.1	50.0	9.6	77.5	94.1	22.9	0.5	5.6	4.6	0.1	0.0	1.4	0.0	0.1	0.7
23	0.0	0.1	0.1	0.4	2.0	1.6	4.9	1.2	0.5	0.2	0.9	0.2	0.3	0.0	0.1	0.1	0.1	0.0	1.4	1.1	0.0	0.0	0.5	2.6	1.4	0.5	0.1	5.4	1.3	19.4	36.8	2.5	0.1	1.4	1.5	0.1	0.2	0.3	0.0	0.0	0.2
24	0.3	1.8	2.1	0.9	3.2	3.8	4.0	2.3	0.9	0.6	1.1	3.6	1.9	0.1	0.4	3.7	3.9	0.3	2.2	1.0	0.1	0.1	6.0	4.5	17.2	1.3	0.2	15.6	29.8	29.3	7.1	49.5	1.3	4.7	2.2	1.0	0.0	0.6	0.3	0.1	0.3
25	0.1	0.3	0.8	0.5	0.5	0.6	0.3	0.3	0.1	0.0	0.1	0.5	0.3	0.1	0.1	0.4	0.7	0.1	0.8	0.4	0.0	0.1	0.4	0.1	1.4	0.8	0.1	2.0	8.8	2.8	0.4	60.1	0.4	1.1	0.9	0.6	0.1	0.2	0.3	1.7	0.3
26	0.1	0.2	0.6	0.2	0.2	0.2	0.1	0.2	0.0	0.0	0.1	0.3	0.2	0.0	0.1	0.2	0.4	0.0	0.6	0.3	0.0	0.0	0.3	0.1	0.8	0.2	0.0	0.8	2.3	0.9	0.2	4.2	0.1	0.2	0.4	0.1	0.0	0.0	0.1	0.1	0.0
27	0.0	0.5	4.4	0.4	1.3	1.2	1.3	0.7	0.3	0.1	0.7	2.1	2.4	0.1	0.1	1.6	2.7	0.1	21.7	1.4	0.0	0.0	1.1	1.1	3.5	4.6	1.2	9.9	3.0	7.1	31.7	34.0	0.2	0.7	1.1	0.0	0.0	0.1	0.0	0.3	0.1
28	0.0	0.0	0.2	0.3	0.0	0.4	0.1	0.2	0.2	0.0	0.3	0.1	0.1	0.1	0.0	0.1	0.1	0.0	1.3	0.3	0.0	3.3	0.2	0.5	0.5	13.8	3.4	5.7	1.3	3.0	1.5	68.8	0.1	2.7	2.7	0.3	0.2	0.8	0.4	43.0	74.9
29	4.6	8.4	11.1	6.6	20.4	54.7	88.5	88.1	89.6	86.6	95.3	86.3	86.9	68.1	60.2	67.6	42.6	10.4	13.6	3.8	24.2	7.6	24.7	84.4	25.8	4.9	5.6	14.9	17.6	21.7	5.0	2.9	0.9	4.8	3.1	0.4	0.1	0.4	0.8	0.2	4.4
30	1.2	3.6	8.4	2.4	2.6	13.5	24.9	34.5	34.4	24.6	52.1	60.1	64.6	27.4	23.8	20.7	16.7	3.2	9.1	2.9	12.0	6.9	8.8	35.8	6.5	2.8	2.9	4.4	3.7	3.0	1.0	2.8	0.1	0.9	1.0	0.0	0.1	0.1	0.4	0.6	3.4
31	1.7	7.6	4.7	1.7	2.4	6.8	17.7	18.4	21.5	37.9	31.6	35.4	20.1	9.3	13.0	48.3	57.7	3.3	1.2	5.9	69.0	10.5	16.1	41.8	11.0	0.6	0.5	1.2	10.7	2.2	0.0	1.1	1.4	2.0	1.3	1.1	0.5	0.2	0.8	0.2	1.1
32	1.4	1.6	1.0	1.9	8.1	50.1	72.4	57.2	60.9	42.5	57.6	12.1	14.8	23.6	19.5	16.0	5.4	0.3	1.6	1.7	3.6	0.1	2.1	41.4	5.0	1.8	1.7	12.2	10.9	17.1	4.1	4.1	0.9	6.5	3.0	1.3	0.3	0.6	0.5	0.3	3.8
33	1.2	4.6	0.9	0.9	1.2	9.8	12.9	10.3	7.0	12.7	10.5	9.8	5.7	1.3	2.1	6.3	4.1	0.4	2.6	4.3	8.6	1.5	16.3	77.1	23.5	0.7	0.5	5.1	10.3	10.8	3.9	1.8	1.1	2.9	1.6	0.7	0.3	0.2	0.4	0.1	0.4
34	0.2	0.7	2.0	0.5	0.7	5.4	7.2	6.2	4.9	1.2	8.9	5.7	6.2	2.9	2.2	2.3	2.3	0.1	1.5	0.9	1.0	0.2	0.6	6.3	1.0	1.9	1.7	5.2	2.6	2.2	0.6	1.4	0.0	0.5	0.6	0.0	0.0	0.0	0.1	0.2	1.0
35	0.6	1.1	0.1	1.1	1.3	32.1	29.9	20.0	19.6	16.4	11.8	1.1	0.5	3.2	5.4	4.0	1.5	0.2	0.3	1.0	2.0	0.1	0.3	2.6	0.6	0.4	0.2	1.9	2.6	2.2	0.5	1.7	0.9	4.2	2.0	2.6	0.8	0.3	0.5	0.2	0.6
36	0.2	0.3	0.0	0.5	0.2	5.7	2.5	2.1	1.0	1.6	0.8	0.1	0.1	0.3	0.6	0.3	0.1	0.1	0.1	0.3	0.3	0.0	0.1	0.3	0.1	0.2	0.0	0.2	0.4	0.4	0.1	0.4	0.3	1.3	0.8	0.8	0.5	0.1	0.3	0.3	0.1
37	0.1	0.2	0.0	0.2	0.1	0.7	0.1	0.2	0.0	0.1	0.1	0.1	0.0	0.0	0.1	0.1	0.1	0.0	0.1	0.2	0.2	0.0	0.0	0.1	0.1	0.1	0.0	0.0	0.3	0.1	0.5	0.4	0.2	1.2	0.1	0.1	0.2	0.1	0.1	0.2	0.1
38	0.4	2.5	0.6	0.8	0.2	1.9	1.0	1.9	1.4	3.8	2.2	2.4	0.9	0.8	1.4	4.8	9.2	0.4	0.1	0.7	33.0	1.3	1.5	5.9	0.9	0.2	0.1	0.2	1.7	0.3	0.0	0.3	0.8	1.0	0.9	1.0	1.1	0.2	0.6	0.2	0.3
39	0.1	0.4	0.5	0.6	0.1	1.2	0.7	0.8	1.1	0.9	1.0	0.4	0.2	0.3	0.4	1.2	2.0	0.1	4.1	86.2	12.3	1.4	1.6	5.0	1.2	0.2	0.0	0.3	0.6	0.7	0.7	0.2	0.8	2.5	1.2	0.2	0.8	0.1	0.0	0.2	0.6
40	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0
41	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.5	0.3	0.1	0.1	1.0	2.0	0.4
42	0.1	0.1	0.0	0.8	0.1	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.2	0.1	0.6	0.6	0.1	0.0	0.1	0.1	0.1	0.0	0.0	1.8	0.5	0.0	0.5	1.3	2.0	2.3	8.7	1.2	0.8	5.6	2.3	4.4
43	0.0	0.2	0.0	0.3	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.3	0.4	1.0	0.1	0.5	0.5	0.1
44	0.6	1.1	0.0	1.2	0.2	0.7	0.1	0.4	0.0	0.4	0.1	0.1	0.0	0.0	0.2	0.1	0.1	0.3	0.1	0.6	2.0	0.2	0.2	0.3	0.3	0.1	0.0	0.0	0.3	1.1	0.0	1.6	3.3	5.7	3.1	9.0	9.9	0.9	2.5	0.4	1.0
45	0.1	0.7	0.0	0.5	0.0	0.4	0.1	0.1	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.6	0.6	0.4	0.1	0.4	0.1	0.1	0.0	0.0	0.0	0.1	0.2	0.3	0.5	1.0	0.9	0.4	16.4	0.2	0.3	1.1	0.4
46	1.6	0.7	0.4	1.4	4.2	6.1	5.2	3.6	0.8	4.5	2.4	0.7	0.6	0.4	0.5	0.7	0.4	0.5	0.6	2.0	1.1	2.7	2.1	28.4	5.5	0.4	0.1	1.1	8.4	63.8	16.5	5.9	17.8	48.9	25.0	9.8	7.8	11.9	3.7	1.6	17.0
47	0.0	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.3	0.0	0.6	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0	0.0	1.9	0.1	0.4	0.7	0.3	3.1	0.1	0.7	32.3	1.8
48	0.0	0.0	0.0	0.2	0.0	0.1	0.0	0.1	0.1	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.1	0.2	0.2	0.1	0.0	0.1	0.0	0.3	0.0	0.1	0.2	0.1	0.0	1.0	0.1	0.5	0.9	0.6	0.3	0.7	0.7	11.9	37.8
49	2.3	0.5	0.6	4.3	9.5	0.2	0.3	0.3	0.2	0.2	0.3	0.1	0.2	0.2	1.4	0.4	0.7	2.4	0.9	3.0	4.3	3.0	0.3	0.5	0.7	1.2	0.1	0.1	10.4	4.4	0.3	16.1	9.0	12.5	38.0	15.3	5.6	49.1	24.5	49.3	85.1
50	21.0	4.2	7.8	16.1	42.7	1.8	1.1	2.0	0.9	0.8	1.1	0.5	0.7	1.5	6.2	2.4	2.9	12.7	14.1	25.3	16.8	36.0	8.4	17.5	14.1	8.3	1.5	1.3	60.0	40.0	9.2	80.0	68.2	70.4	84.9	64.8	28.1	87.5	76.1	79.8	97.1
51	78.3	38.3	65.7	68.2	92.6	20.6	21.4	30.6	27.7	19.7	20.9	12.5	18.3	25.3	54.7	34.7	43.4	65.5	69.6	79.8	72.7	95.1	60.9	73.4	75.8	48.8	24.9	25.3	95.2	85.3	59.0	96.0	96.5	98.4	98.6	90.5	88.5	97.2	96.4	92.5	99.6

- Multi-variate HMM emits vector of values, not just one value
- Can emit real values (SegWay) or binary presence /absence values (ChromHMM)
- Use to learn mark combinations

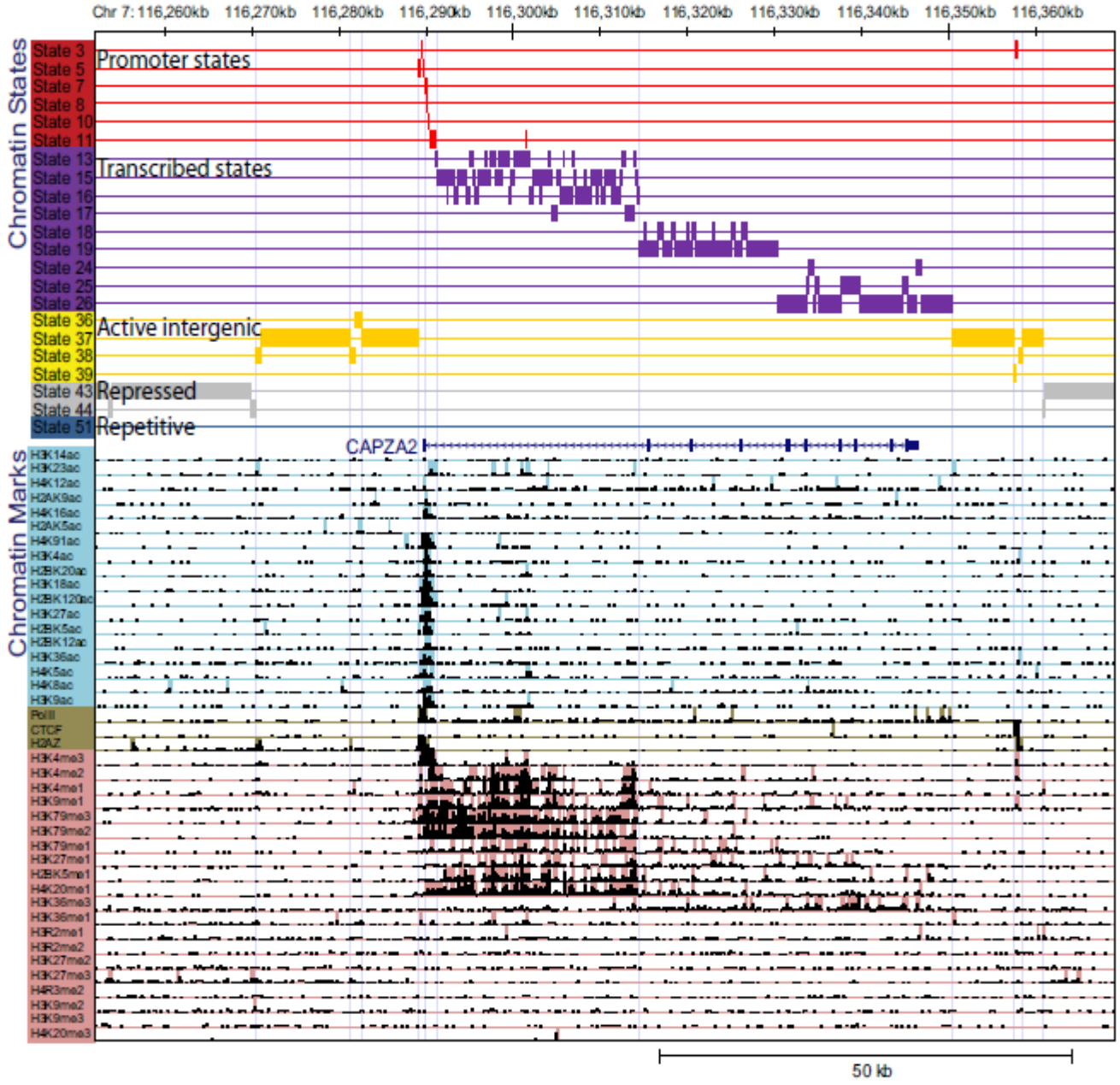
Ernst and Kellis
Nature Biotech 2010

Transition matrix a_{kl}



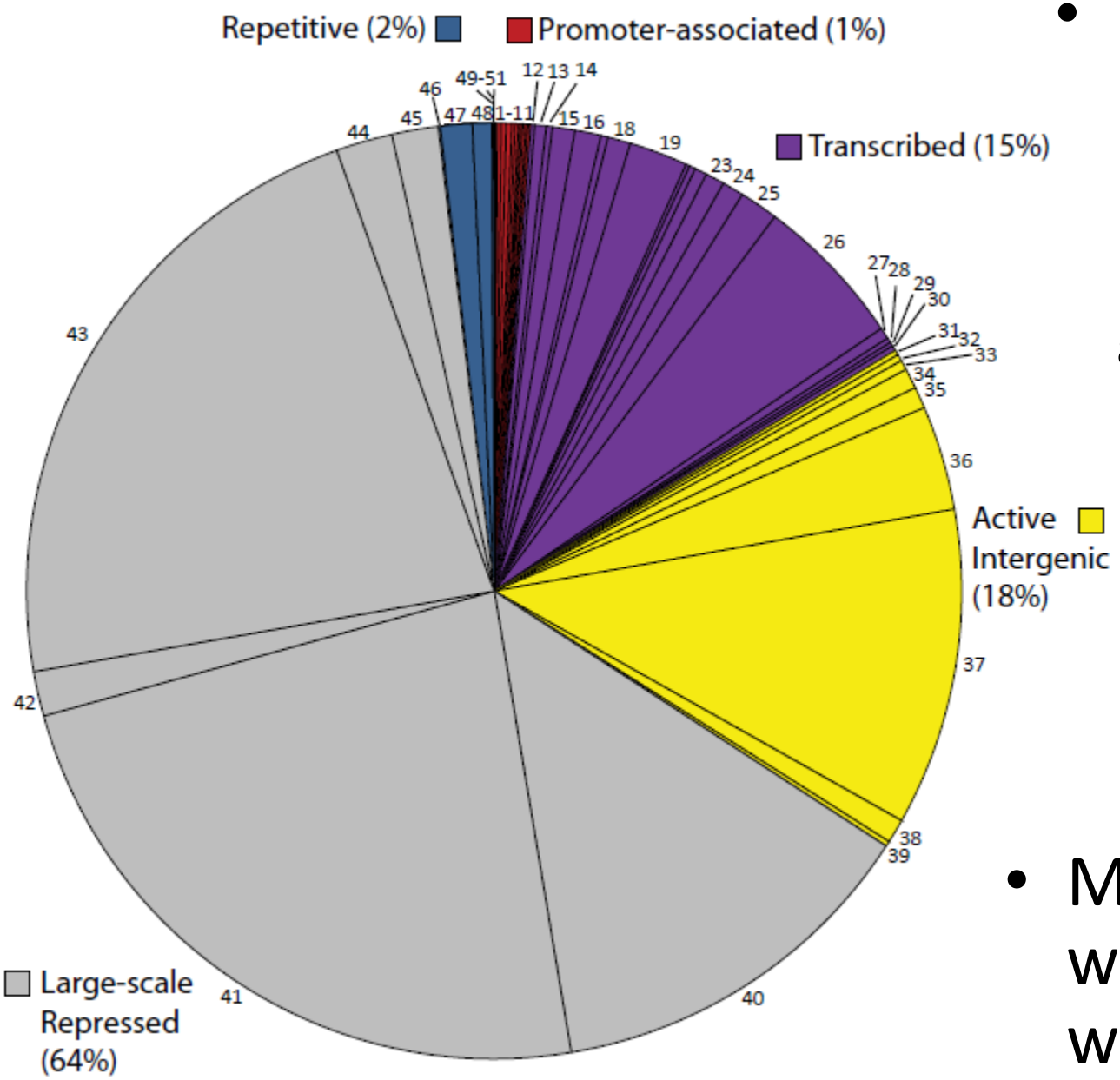
- Learns spatial relationships between neighboring states
- Reveals distinct sub-groups of states
- Reveals transitions between different groups

Example Chromatin State Annotation



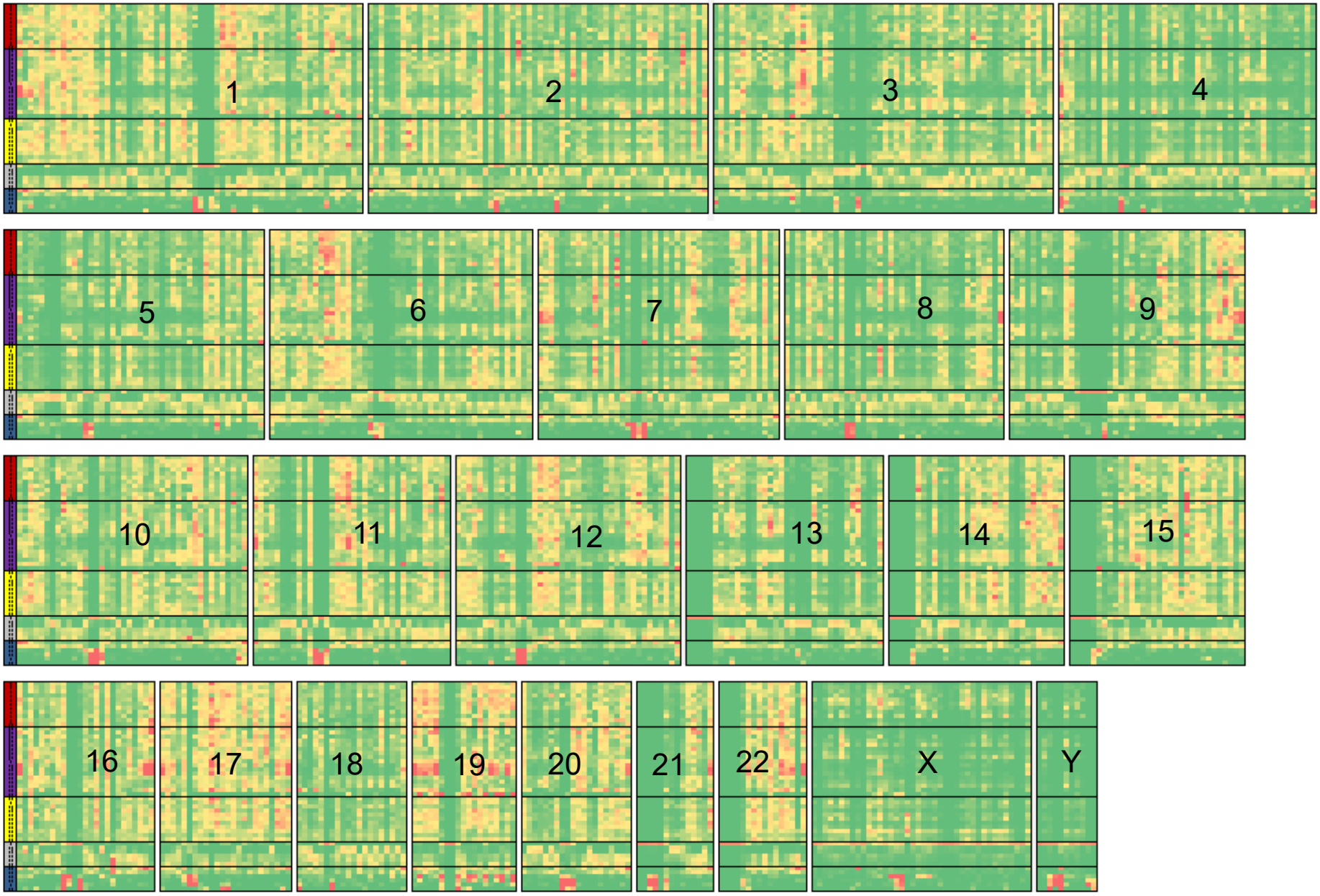
- Use Baum Welch to learn hidden states and their annotations
- Learned states correspond to known functional elements
- *De novo* discovery of major types of chromatin

Model complexity matches that of genome



- Handful of repressed states capture vast majority of genome
  - Only 1% of genome split in 14 promoter states
- Modeling power well distributed where needed

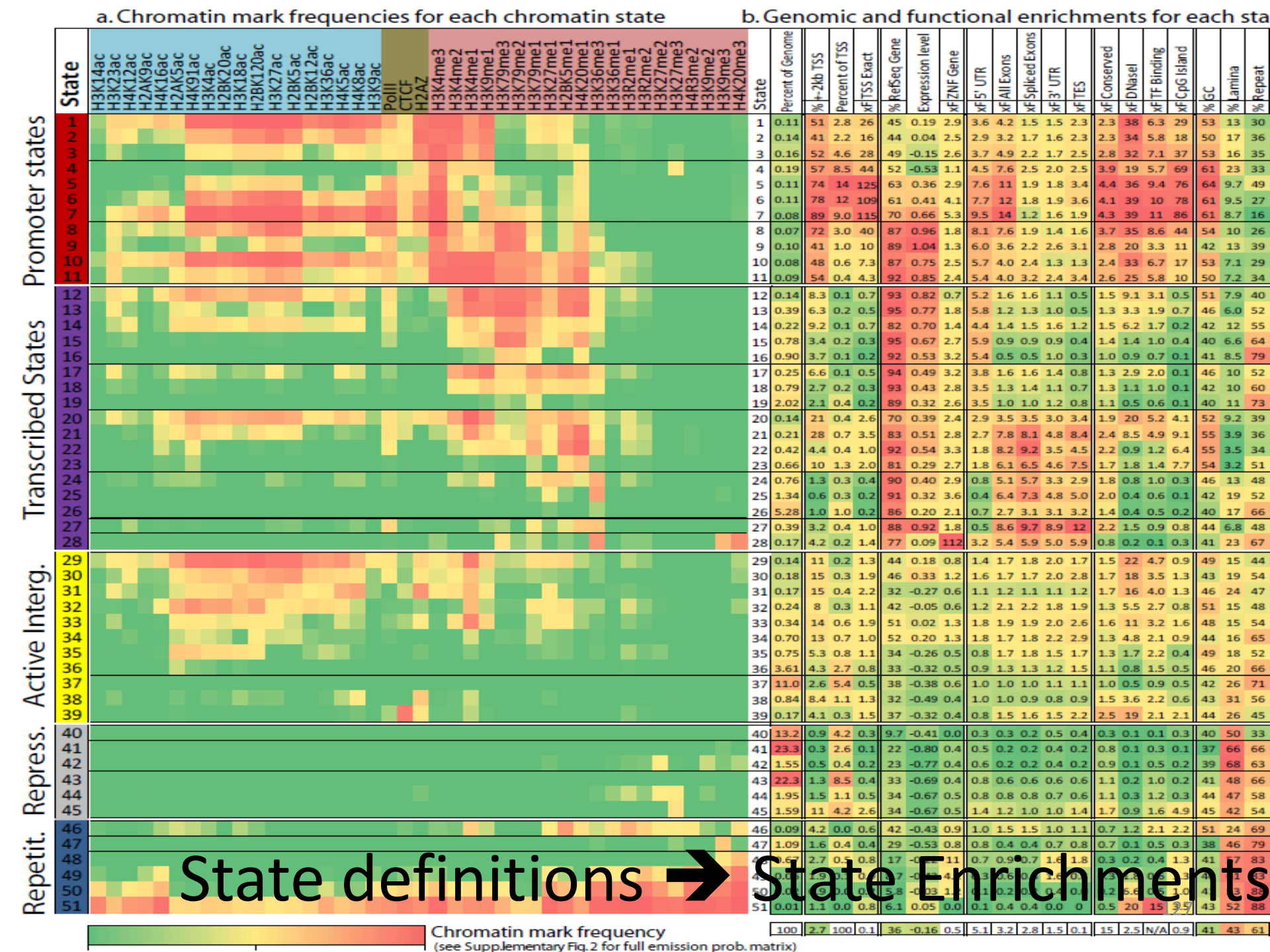
Apply genome wide to classify chromatin states *de novo*



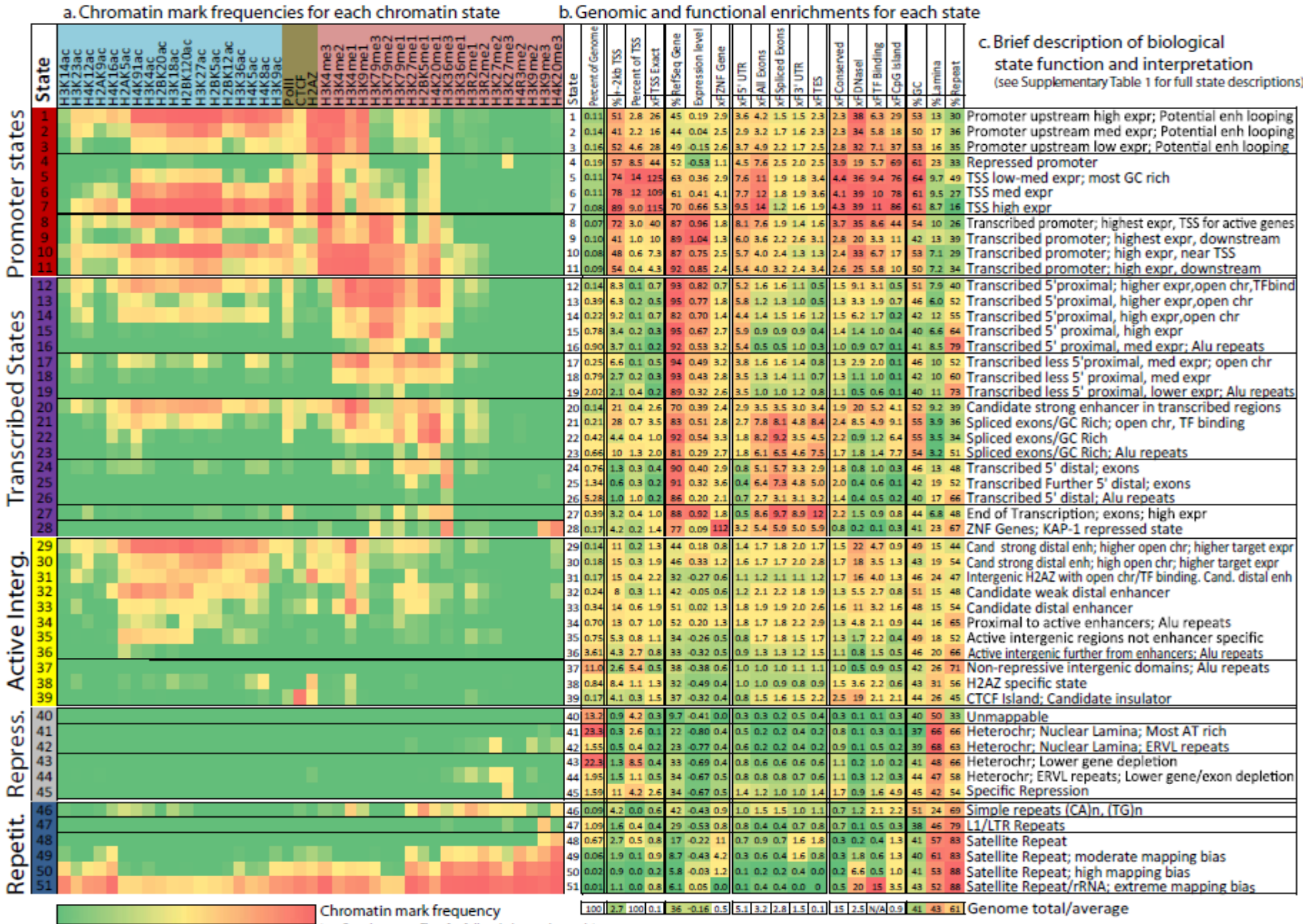
Now what? Interpret these states biologically

Goals for today: Computational Epigenomics

1. Introduction to Epigenomics
   - Overview of epigenomics, Diversity of Chromatin modifications
   - Antibodies, ChIP-Seq, data generation projects, raw data
2. Primary data processing: Read mapping, Peak calling
   - Read mapping: Hashing, Suffix Trees, Burrows-Wheeler Transform
   - Quality Control, Cross-correlation, Peak calling, IDR (similar to FDR)
3. Discovery and characterization of chromatin states
   - HMM Foundations, Generating, Parsing, Decoding, Learning
   - Chromatin state characterization: Functional/positional enrichment
4. Model complexity: selecting the number of states/marks
   - Selecting the number of states, selecting number of marks
   - Capturing dependencies and state-conditional mark independence
5. Learning chromatin states jointly across multiple cell types
   - Stacking vs. concatenation approach for joint multi-cell type learning
   - Defining activity profiles for linking enhancer regulatory networks

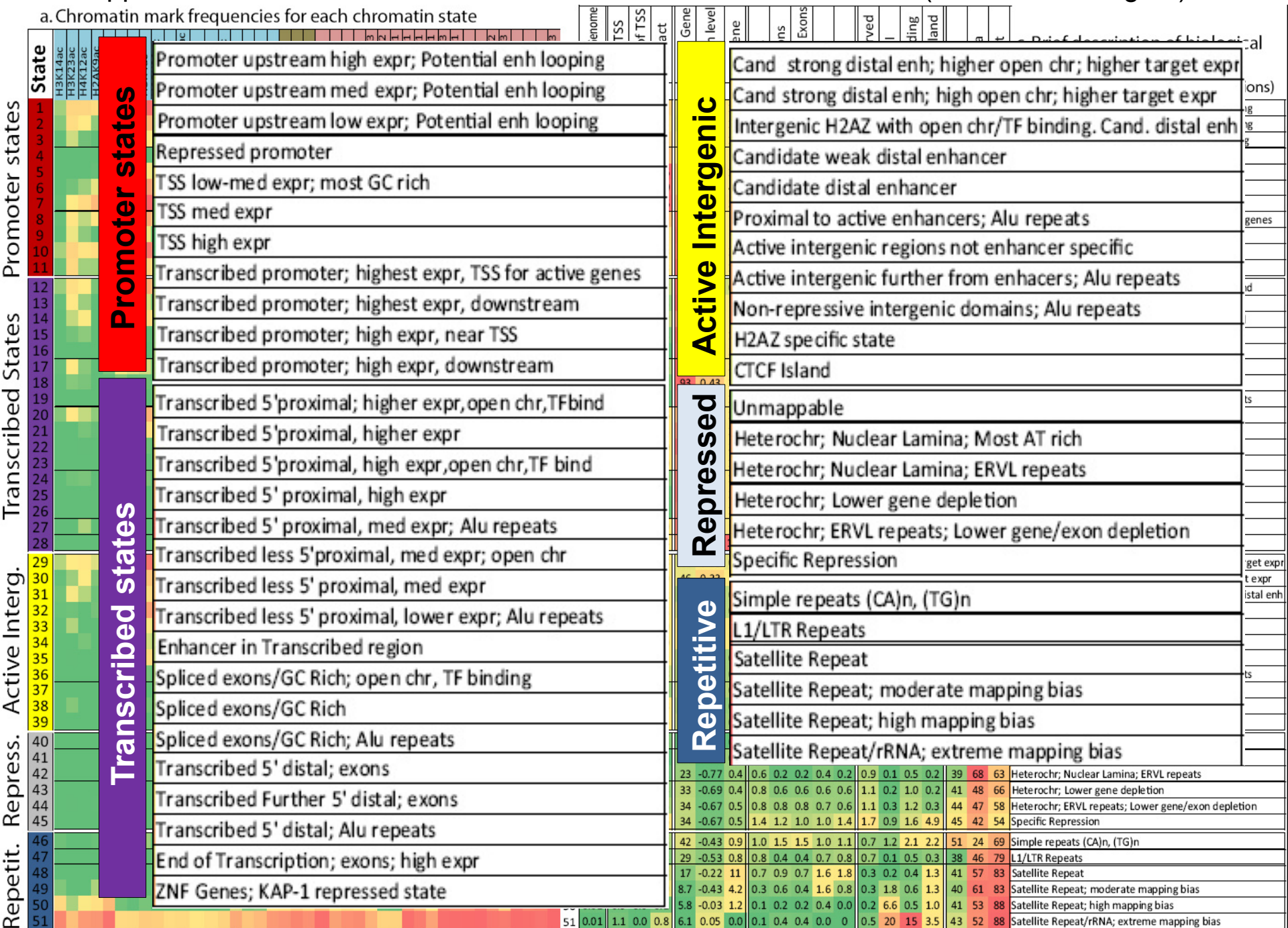

a. Chromatin mark frequencies for each chromatin state
b. Genomic and functional enrichments for each sta
State
H3K14ac
H3K23ac
H4K12ac
H2AK9ac
H4K16ac
H2AK5ac
H4K91ac
H3K4ac
H2BK20ac
H3K18ac
H2BK120ac
H3K27ac
H2BK5ac
H2BK12ac
H3K36ac
H4K5ac
H4K8ac
H3K9ac
PolII
CTCF
H2AZ
H3K4me3
H3K4me2
H3K4me1
H3K9me1
H3K79me3
H3K79me2
H3K79me1
H3K27me1
H2BK5me1
H4K20me1
H3K36me3
H3K36me1
H3R2me1
H3R2me2
H3K27me2
H3K27me3
H4R3me2
H3K9me2
H3K9me3
H4K20me3
State
Percent of Genome
% +-2kb TSS
Percent of TSS
xFTSS Exact
% RefSeq Gene
Expression level
xFZNF Gene
xF5' UTR
xFAll Exons
xFSpliced Exons
xF3' UTR
xFTES
xFConserved
xFDNaseI
xFTF Binding
xFCpG Island
% GC
% Lamina
% Repeat
Promoter states
Transcribed States
Active Interg.
Repress.
Repetit.
State definitions → State Enrichments
Chromatin mark frequency
(see Supplementary Fig. 2 for full emission prob. matrix)

Functional enrichments enable annotation of 51 distinct states



State	Percent of Genome	%/-2kb TSS	Percent of TSS	xFTSS Exact	%RefSeq Gene	Expression level	xFZNF Gene	xF5' UTR	xFAll Exons	xFSpliced Exons	xF3' UTR	xFTES	xFConserved	xFDNaseI	xFTF Binding	xFCpG Island	%GC	%Lamina	%Repeat	c. Brief description of biological state function and interpretation (see Supplementary Table 1 for full state descriptions)
Promoter states																				
1	0.11	51	2.8	26	45	0.19	2.9	3.6	4.2	1.5	1.5	2.3	2.3	38	6.3	29	53	13	30	Promoter upstream high expr; Potential enh looping
2	0.14	41	2.2	16	44	0.04	2.5	2.9	3.2	1.7	1.6	2.3	2.3	34	5.8	18	50	17	36	Promoter upstream med expr; Potential enh looping
3	0.16	52	4.6	28	49	-0.15	2.6	3.7	4.9	2.2	1.7	2.5	2.8	32	7.1	37	53	16	35	Promoter upstream low expr; Potential enh looping
4	0.19	57	8.5	44	52	-0.53	1.1	4.5	7.6	2.5	2.0	2.5	3.9	19	5.7	69	61	23	33	Repressed promoter
5	0.11	74	14	125	63	0.36	2.9	7.6	11	1.9	1.8	3.4	4.4	36	9.4	76	64	9.7	49	TSS low-med expr; most GC rich
6	0.11	78	12	109	61	0.41	4.1	7.7	12	1.8	1.9	3.6	4.1	39	10	78	61	9.5	27	TSS med expr
7	0.08	89	9.0	115	70	0.66	5.3	9.5	14	1.2	1.6	1.9	4.3	39	11	86	61	8.7	16	TSS high expr
8	0.07	72	3.0	40	87	0.96	1.8	8.1	7.6	1.9	1.4	1.6	3.7	35	8.6	44	54	10	26	Transcribed promoter; highest expr, TSS for active genes
9	0.10	41	1.0	10	89	1.04	1.3	6.0	3.6	2.2	2.6	3.1	2.8	20	3.3	11	42	13	39	Transcribed promoter; highest expr, downstream
10	0.08	48	0.6	7.3	87	0.75	2.5	5.7	4.0	2.4	1.3	1.3	2.4	33	6.7	17	53	7.1	29	Transcribed promoter; high expr, near TSS
11	0.09	54	0.4	4.3	92	0.85	2.4	5.4	4.0	3.2	2.4	3.4	2.6	25	5.8	10	50	7.2	34	Transcribed promoter; high expr, downstream
Transcribed States																				
12	0.14	8.3	0.1	0.7	93	0.82	0.7	5.2	1.6	1.6	1.1	0.5	1.5	9.1	3.1	0.5	51	7.9	40	Transcribed 5'proximal; higher expr,open chr,TFbind
13	0.39	6.3	0.2	0.5	95	0.77	1.8	5.8	1.2	1.3	1.0	0.5	1.3	3.3	1.9	0.7	46	6.0	52	Transcribed 5'proximal, higher expr,open chr
14	0.22	9.2	0.1	0.7	82	0.70	1.4	4.4	1.4	1.5	1.6	1.2	1.5	6.2	1.7	0.2	42	12	55	Transcribed 5'proximal, high expr,open chr
15	0.78	3.4	0.2	0.3	95	0.67	2.7	5.9	0.9	0.9	0.9	0.4	1.4	1.4	1.0	0.4	40	6.6	64	Transcribed 5' proximal, high expr
16	0.90	3.7	0.1	0.2	92	0.53	3.2	5.4	0.5	0.5	1.0	0.3	1.0	0.9	0.7	0.1	41	8.5	79	Transcribed 5' proximal, med expr; Alu repeats
17	0.25	6.6	0.1	0.5	94	0.49	3.2	3.8	1.6	1.6	1.4	0.8	1.3	2.9	2.0	0.1	46	10	52	Transcribed less 5'proximal, med expr; open chr
18	0.79	2.7	0.2	0.3	93	0.43	2.8	3.5	1.3	1.4	1.1	0.7	1.3	1.1	1.0	0.1	42	10	60	Transcribed less 5' proximal, med expr
19	2.02	2.1	0.4	0.7	89	0.32	2.6	3.5	1.0	1.0	1.2	0.8	1.1	0.5	0.6	0.1	40	11	73	Transcribed less 5' proximal, lower expr; Alu repeats
20	0.14	21	0.4	2.6	70	0.39	2.4	2.9	3.5	3.5	3.0	3.4	1.9	20	5.2	4.1	52	9.2	39	Candidate strong enhancer in transcribed regions
21	0.21	28	0.7	3.5	83	0.51	2.8	2.7	7.8	8.1	4.8	8.4	2.4	8.5	4.9	9.1	55	3.9	36	Spliced exons/GC Rich; open chr, TF binding
22	0.42	4.4	0.4	1.0	92	0.54	3.3	1.8	8.2	9.2	3.5	4.5	2.2	0.9	1.2	6.4	55	3.5	34	Spliced exons/GC Rich
23	0.66	10	1.3	2.0	81	0.29	2.7	1.8	6.1	6.5	4.6	7.5	1.7	1.8	1.4	7.7	54	3.2	51	Spliced exons/GC Rich; Alu repeats
24	0.76	1.3	0.3	0.4	90	0.40	2.9	0.8	5.1	5.7	3.3	2.9	1.8	0.8	1.0	0.3	46	13	48	Transcribed 5' distal; exons
25	1.34	0.6	0.3	0.2	91	0.32	3.6	0.4	6.4	7.3	4.8	5.0	2.0	0.4	0.6	0.1	42	19	52	Transcribed Further 5' distal; exons
26	5.28	1.0	1.0	0.2	86	0.20	2.1	0.7	2.7	3.1	3.1	3.2	1.4	0.4	0.5	0.2	40	17	66	Transcribed 5' distal; Alu repeats
27	0.39	3.2	0.4	1.0	88	0.92	1.8	0.5	8.6	9.7	8.9	12	2.2	1.5	0.9	0.8	44	6.8	48	End of Transcription; exons; high expr
28	0.17	4.2	0.2	1.4	77	0.09	112	3.2	5.4	5.9	5.0	5.9	0.8	0.2	0.1	0.3	41	23	67	ZNF Genes; KAP-1 repressed state
Active Interg.																				
29	0.14	11	0.2	1.3	44	0.18	0.8	1.4	1.7	1.8	2.0	1.7	1.5	22	4.7	0.9	49	15	44	Cand strong distal enh; higher open chr; higher target expr
30	0.18	15	0.3	1.9	46	0.33	1.2	1.6	1.7	1.7	2.0	2.8	1.7	18	3.5	1.3	43	19	54	Cand strong distal enh; high open chr; higher target expr
31	0.17	15	0.4	2.2	32	-0.27	0.6	1.1	1.2	1.1	1.1	1.2	1.7	16	4.0	1.3	46	24	47	Intergenic H2AZ with open chr/TF binding. Cand. distal enh
32	0.24	8	0.3	1.1	42	-0.05	0.6	1.2	2.1	2.2	1.8	1.9	1.3	5.5	2.7	0.8	51	15	48	Candidate weak distal enhancer
33	0.34	14	0.6	1.9	51	0.02	1.3	1.8	1.9	1.9	2.0	2.6	1.6	11	3.2	1.6	48	15	54	Candidate distal enhancer
34	0.70	13	0.7	1.0	52	0.20	1.3	1.8	1.7	1.8	2.2	2.9	1.3	4.8	2.1	0.9	44	16	65	Proximal to active enhancers; Alu repeats
35	0.75	5.3	0.8	1.1	34	-0.26	0.5	0.8	1.7	1.8	1.5	1.7	1.3	1.7	2.2	0.4	49	18	52	Active intergenic regions not enhancer specific
36	3.61	4.3	2.7	0.8	33	-0.32	0.5	0.9	1.3	1.3	1.2	1.5	1.1	0.8	1.5	0.5	46	20	66	Active intergenic further from enhancers; Alu repeats
37	11.0	2.6	5.4	0.5	38	-0.38	0.6	1.0	1.0	1.0	1.1	1.1	1.0	0.5	0.9	0.5	42	26	71	Non-repressive intergenic domains; Alu repeats
38	0.84	8.4	1.1	1.3	32	-0.49	0.4	1.0	1.0	0.9	0.8	0.9	1.5	3.6	2.2	0.6	43	31	56	H2AZ specific state
39	0.17	4.1	0.3	1.5	37	-0.32	0.4	0.8	1.5	1.6	1.5	2.2	2.5	19	2.1	2.1	44	26	45	CTCF Island; Candidate insulator
Repress.																				
40	13.2	0.9	4.2	0.3	9.7	-0.41	0.0	0.3	0.3	0.2	0.5	0.4	0.3	0.1	0.1	0.3	40	50	33	Unmappable
41	23.3	0.3	2.6	0.1	22	-0.80	0.4	0.5	0.2	0.2	0.4	0.2	0.8	0.1	0.3	0.1	37	66	66	Heterochr; Nuclear Lamina; Most AT rich
42	1.55	0.5	0.4	0.2	23	-0.77	0.4	0.6	0.2	0.2	0.4	0.2	0.9	0.1	0.5	0.2	39	68	63	Heterochr; Nuclear Lamina; ERVL repeats
43	22.3	1.3	8.5	0.4	33	-0.69	0.4	0.8	0.6	0.6	0.6	0.6	1.1	0.2	1.0	0.2	41	48	66	Heterochr; Lower gene depletion
44	1.95	1.5	1.1	0.5	34	-0.67	0.5	0.8	0.8	0.8	0.7	0.6	1.1	0.3	1.2	0.3	44	47	58	Heterochr; ERVL repeats; Lower gene/exon depletion
45	1.59	11	4.2	2.6	34	-0.67	0.5	1.4	1.2	1.0	1.0	1.4	1.7	0.9	1.6	4.9	45	42	54	Specific Repression
Repetit.																				
46	0.09	4.2	0.0	0.6	42	-0.43	0.9	1.0	1.5	1.5	1.0	1.1	0.7	1.2	2.1	2.2	51	24	69	Simple repeats (CA)n, (TG)n
47	1.09	1.6	0.4	0.4	29	-0.53	0.8	0.8	0.4	0.4	0.7	0.8	0.7	0.1	0.5	0.3	38	46	79	L1/LTR Repeats
48	0.67	2.7	0.5	0.8	17	-0.22	11	0.7	0.9	0.7	1.6	1.8	0.3	0.2	0.4	1.3	41	57	83	Satellite Repeat
49	0.06	1.9	0.1	0.9	8.7	-0.43	4.2	0.3	0.6	0.4	1.6	0.8	0.3	1.8	0.6	1.3	40	61	83	Satellite Repeat; moderate mapping bias
50	0.02	0.9	0.0	0.2	5.8	-0.03	1.2	0.1	0.2	0.2	0.4	0.0	0.2	6.6	0.5	1.0	41	53	88	Satellite Repeat; high mapping bias
51	0.01	1.1	0.0	0.8	6.1	0.05	0.0	0.1	0.4	0.4	0.0	0	0.5	20	15	3.5	43	52	88	Satellite Repeat/rRNA; extreme mapping bias
	100	2.7	100	0.1	36	-0.16	0.5	5.1	3.2	2.8	1.5	0.1	15	2.5	N/A	0.9	41	43	61	Genome total/average

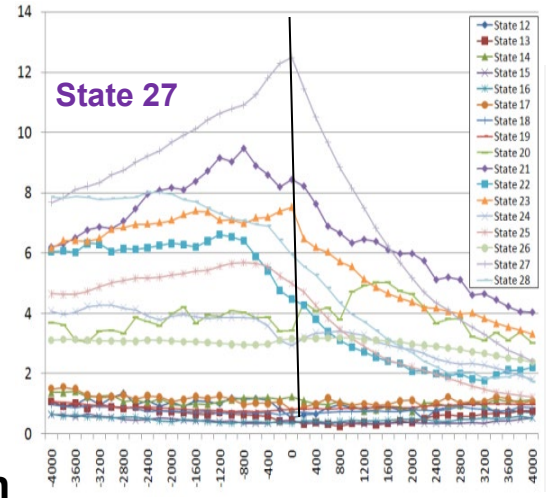
Application of ChromHMM to 41 chromatin marks in CD4+ T-cells (Barski'07, Wang'08)



Functional properties of discovered chromatin states

GO Category	State 3	State 4	State 5	State 6	State 7	State 8
Cell Cycle Phase	2.10 ($2x10^{-7}$)	0.57 (1)	1.61 (0.001)	1.45 (1)	1.15 (1)	1.51 (1)
Embryonic Development	1.24 (1)	2.82 ($9x10^{-23}$)	1.07 (1)	0.85 (1)	0.54 (1)	1.00 (1)
Chromatin	1.20 (1)	0.48 (1)	2.2 ($1.4x10^{-7}$)	1.64 (1)	0.85 (1)	0.85 (1)
Response to DNA Damage Stimulus	1.20 (1)	0.35 (1)	1.55 (0.074)	2.13 ($6.5x10^{-11}$)	1.97 ($1.0x10^{-4}$)	0.84 (1)
RNA Processing	0.49 (1)	0.26 (1)	1.31 (1)	1.91 ($4.2x10^{-11}$)	2.64 ($8.7x10^{-24}$)	2.45 ($3.0x10^{-4}$)
T cell Activation	0.77 (1)	0.88 (1)	1.27 (1)	0.70 (1)	0.79 (1)	4.72 ($2x10^{-7}$)

Promoter state ➔ gene GO function

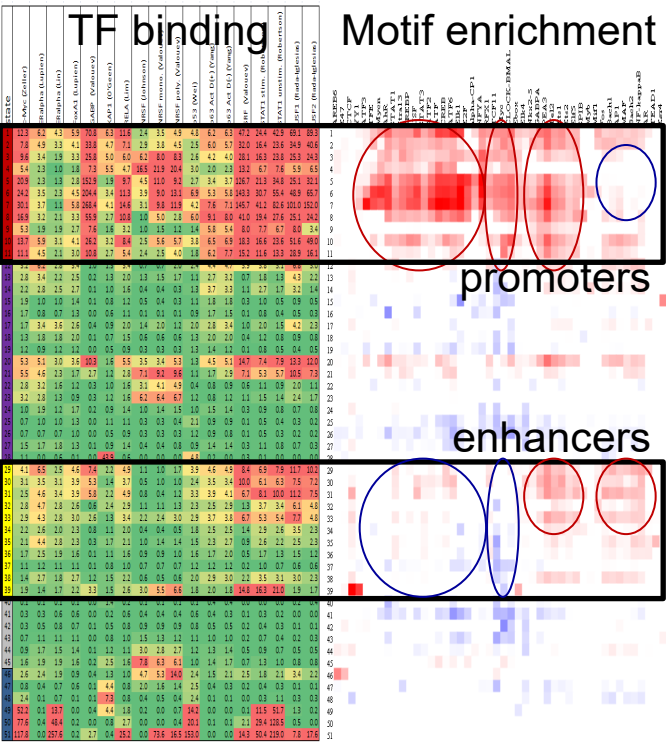


Transcription End State

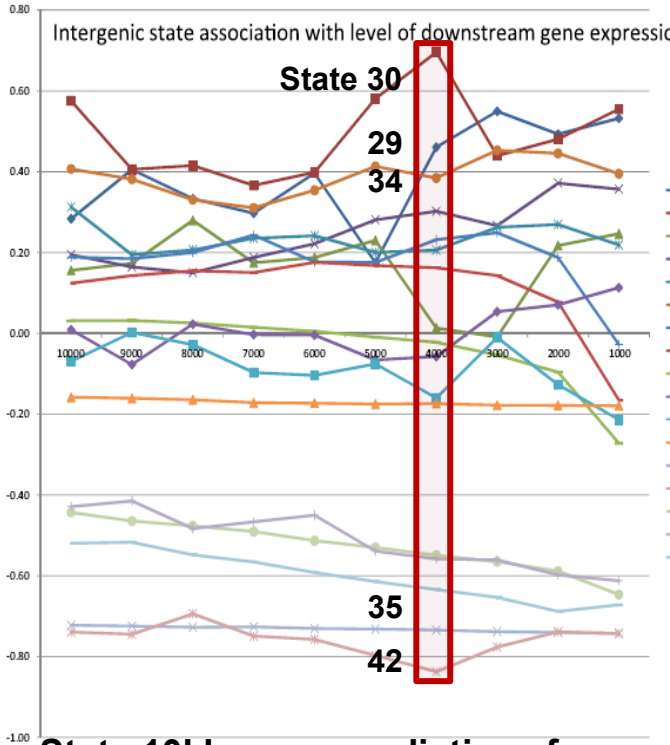
State 28: 112-fold ZNF enrich

"The achievement of the repressed state by wild-type KAP1 involves decreased recruitment of **RNA polymerase II**, reduced levels of histone **H3 K9 acteylation** and **H3K4 methylation**, an increase in **histone occupancy**, enrichment of **trimethyl histone H3K9**, **H3K36**, and **histone H4K20** …" MCB 2006.

ZNF repressed state recovery



Promoter vs. enhancer regulation

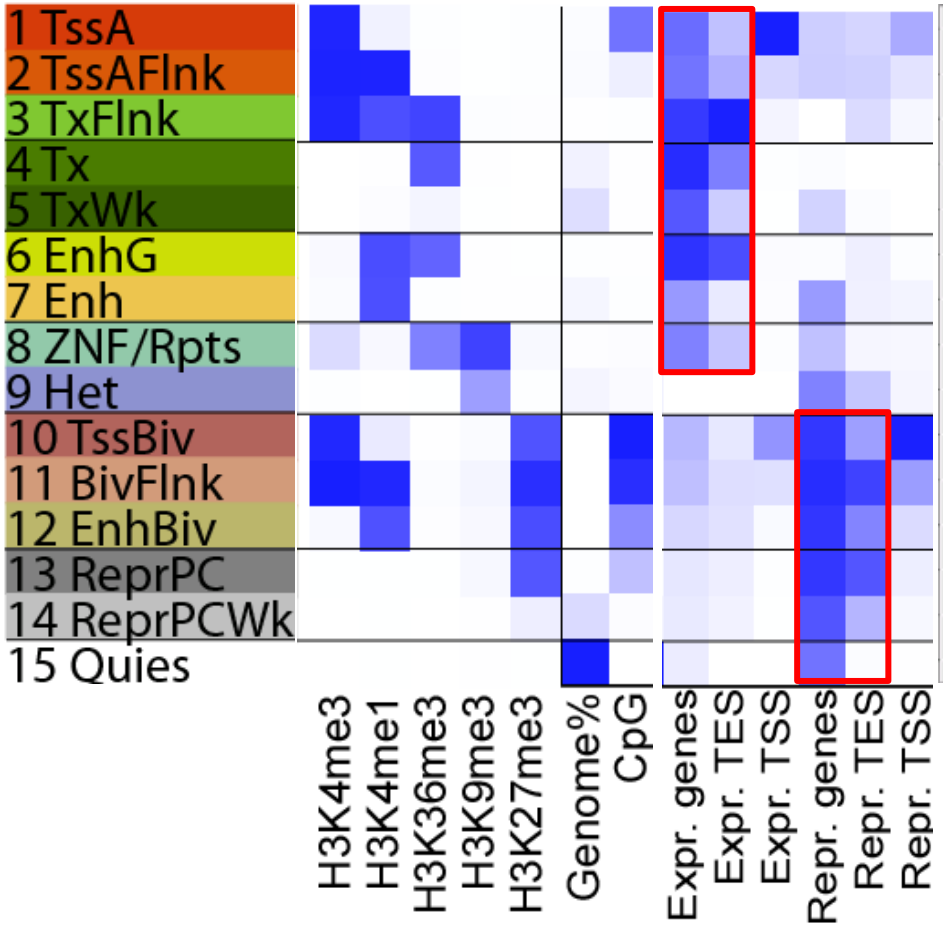


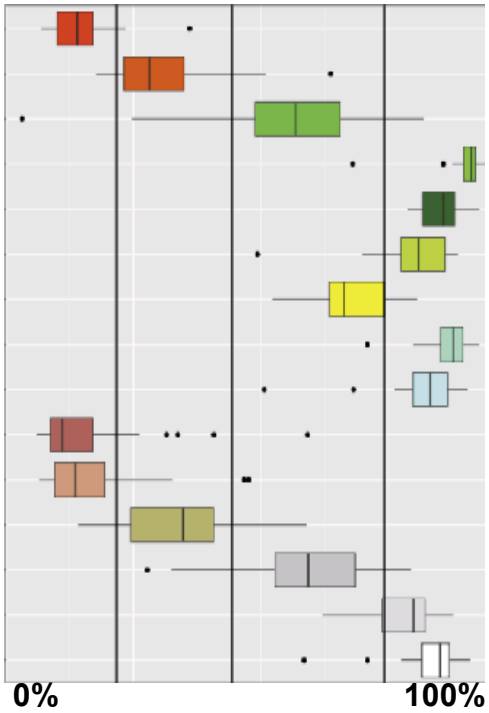
State 10kb away predictive of expr.

state	stalk	variable heterocrhomatic	acrocentric	gneg	gpos25	gpos50	gpos75	gpos100
40	7.6	6.6	6.1	0.6	0.4	0.5	0.3	0.5
41	0.0	0.2	0.4	0.5	0.4	0.9	1.7	2.5
42	0.0	0.2	0.3	0.5	0.5	1.1	1.7	2.2
43	0.0	0.1	0.1	1.1	1.2	1.3	1.1	0.7
44	0.0	0.1	0.1	1.2	1.3	1.3	1.0	0.6
45	0.0	0.1	0.1	1.3	1.6	1.3	0.8	0.4
46	0.0	0.1	0.2	1.7	1.7	0.7	0.4	0.2
47	0.0	0.2	0.1	1.2	1.3	1.3	0.9	0.6
48	0.0	3.2	6.2	0.8	2.2	0.2	0.5	0.4
49	0.0	3.6	11.2	0.5	1.8	0.2	0.4	0.1
50	0.0	4.7	12.0	0.6	0.6	0.2	0.2	0.1
51	0.0	4.4	12.7	0.5	1.4	0.2	0.2	0.1
% Overall	0.6	3.9	3.6	42.1	6.8	13.6	13.1	16.2

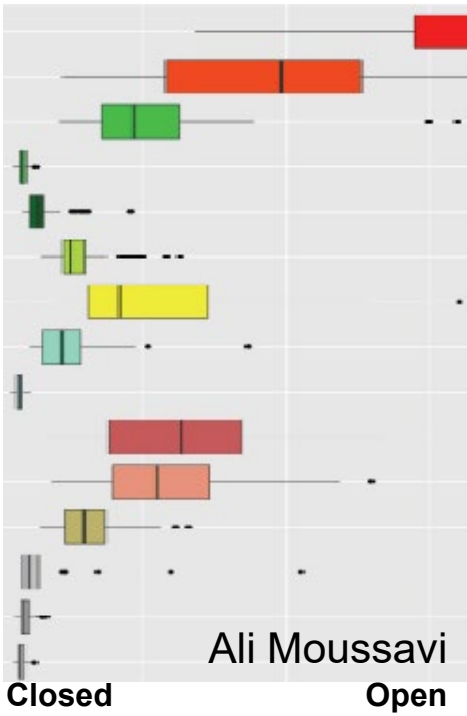
Distinct types of repression
- **Chrom bands / HDAC resp**
- **Repeat family / composition**

States show distinct mCpG, DNase, Tx, Ac profiles





DNA methylation
(WGBS, 37 epigenomes)



DNA accessibility
(DNaseI, 53 epigenomes

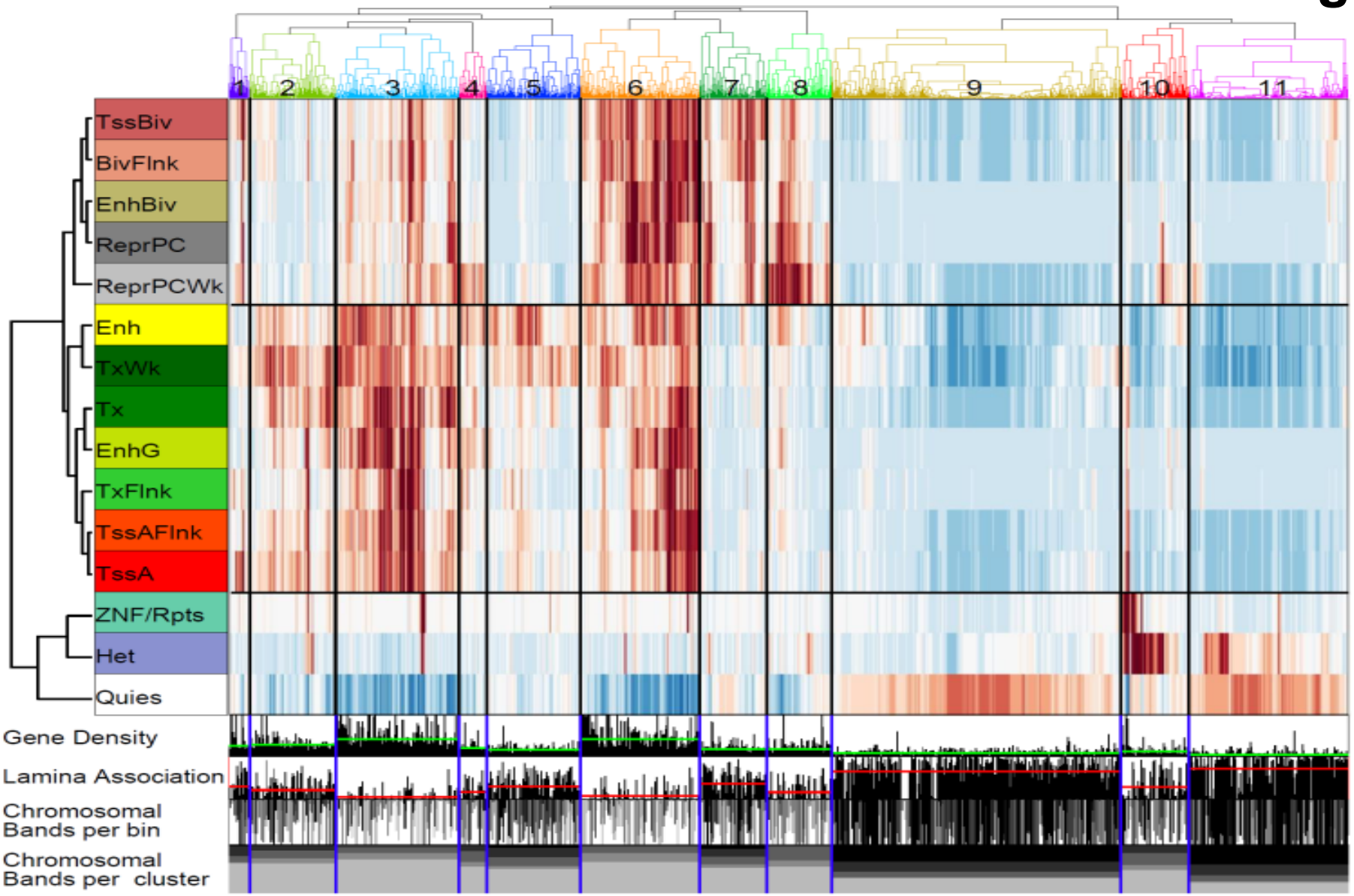
TssA vs. **TssBiv**: diff. activity, both open, both unmethylated!

Enh vs. **ReprPC**: diff. activity, both intermediate DNase/Methyl

Tx: Methylated, closed, actively transcribed

➔ Distinct modes of repression: **H3K27me3** vs. **DNAme** vs. **Het**

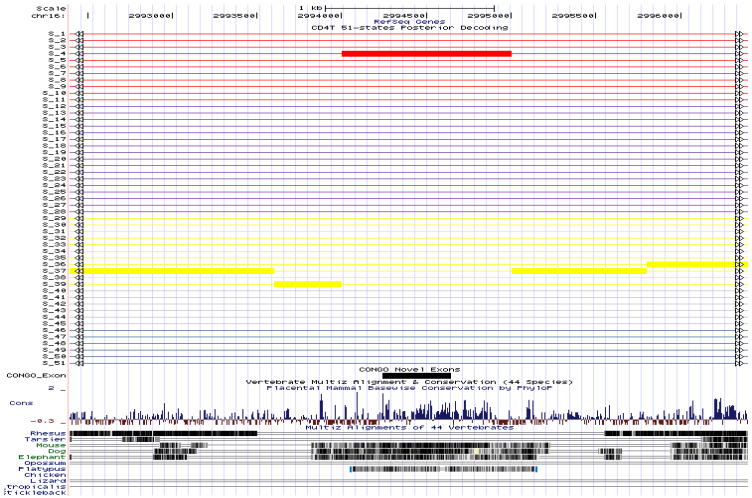
Chromosomal ‘domains’ from chromatin state usage



- State usage ➔ gene density, lamina, cytogenetic bands
- Quies/ZNF/het | gene rich/poor, each active/repressed

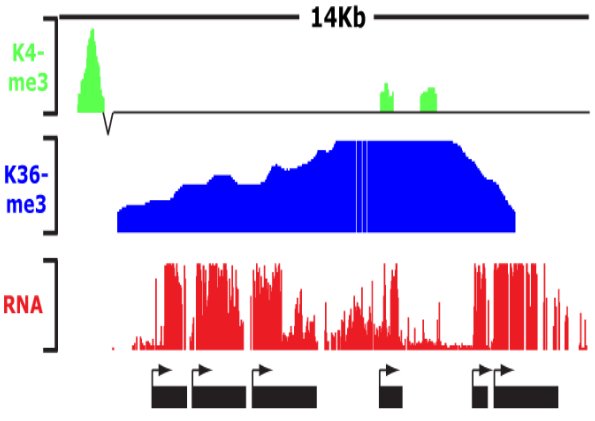
Applications to genome annotation

New protein-coding genes

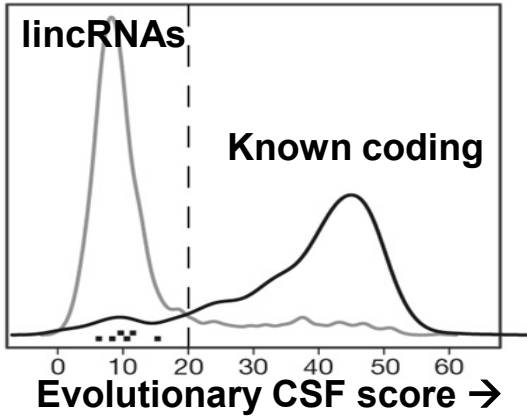


In promoter(short)/low-expr states

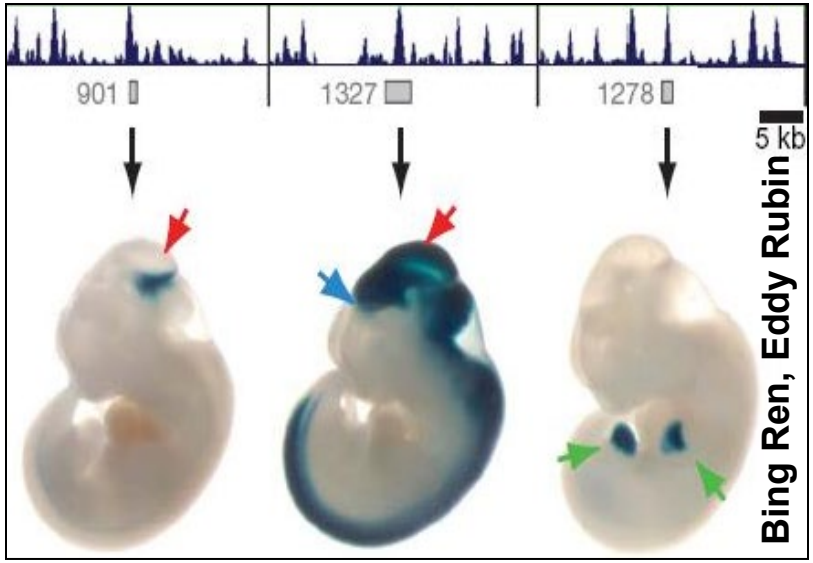
Long intergenic non-coding RNAs/lincRNAs



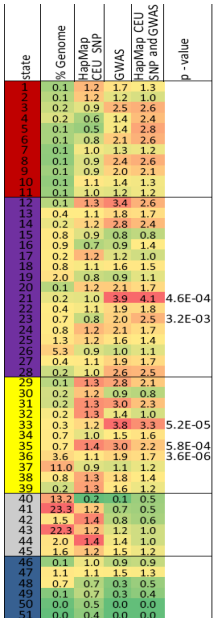
Chromatin signature: promoter / transcribed

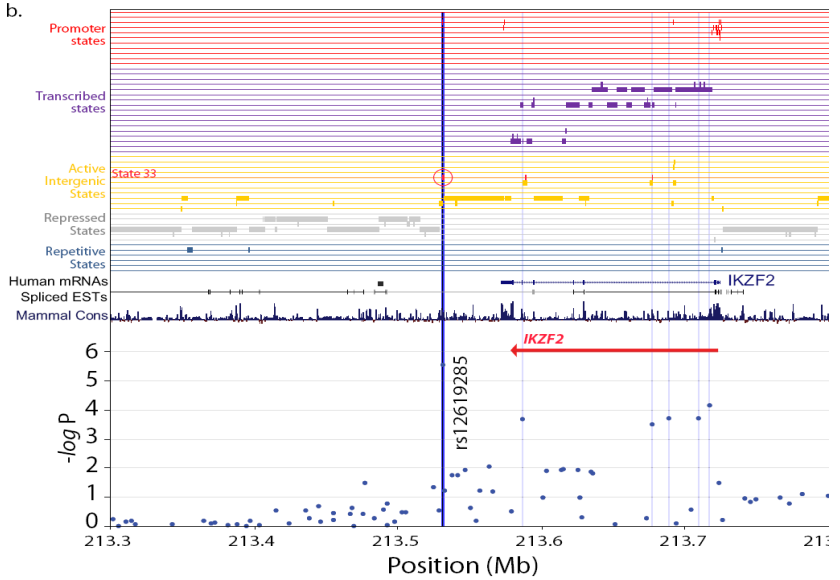


Evolutionary signature: not protein-coding



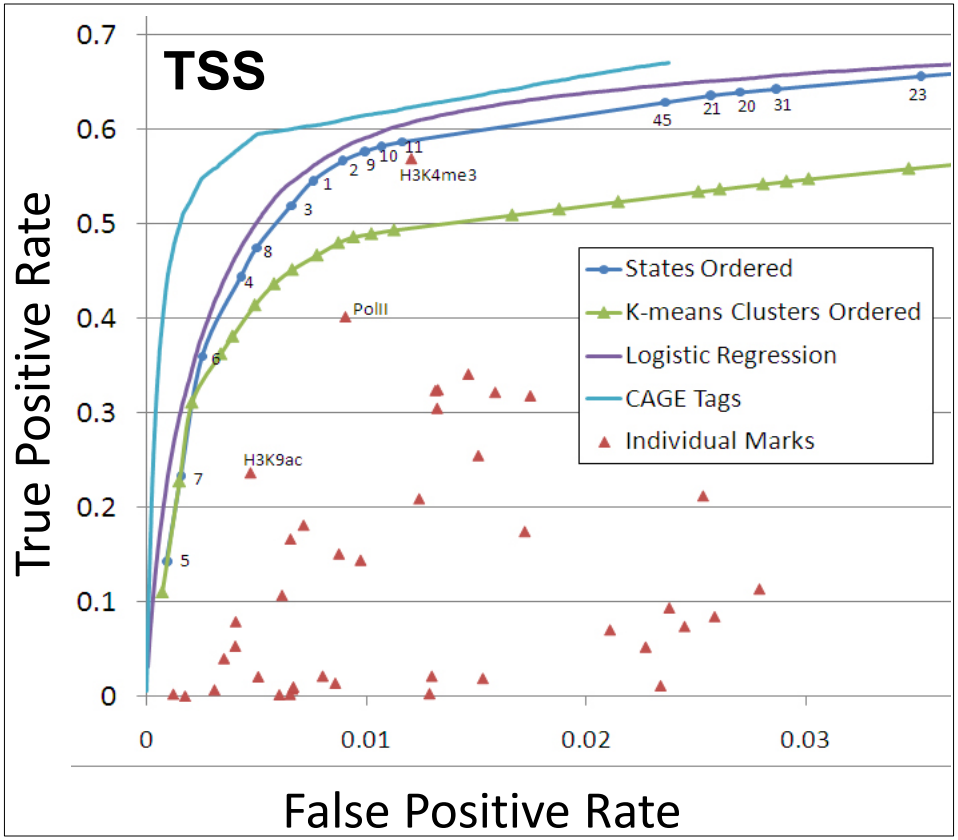
New developmental enhancer regions

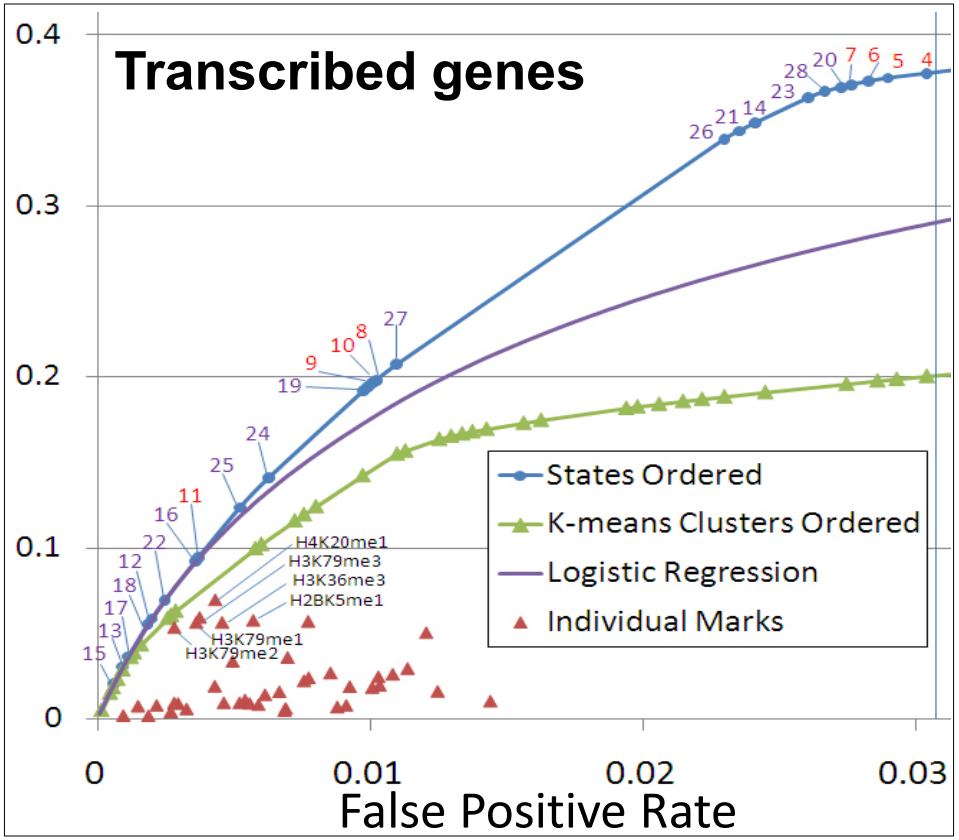




Assign candidate functions to intergenic SNPs from genome-wide association studies

Discovery power for promoters, transcripts





- Significantly outperforms single-marks
- Similar power to supervised learning approach
- CAGE experiments give possible upper bound

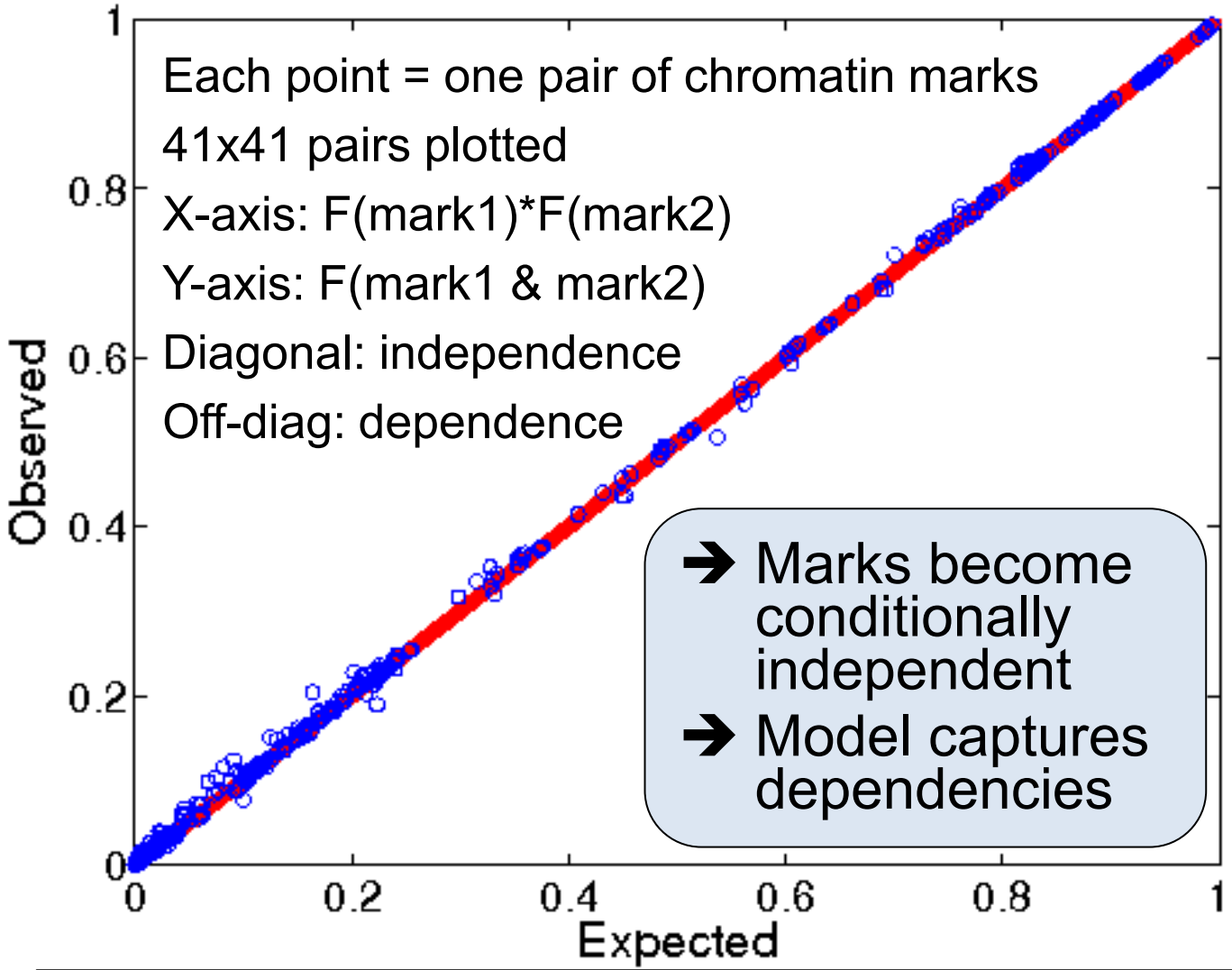
Goals for today: Computational Epigenomics

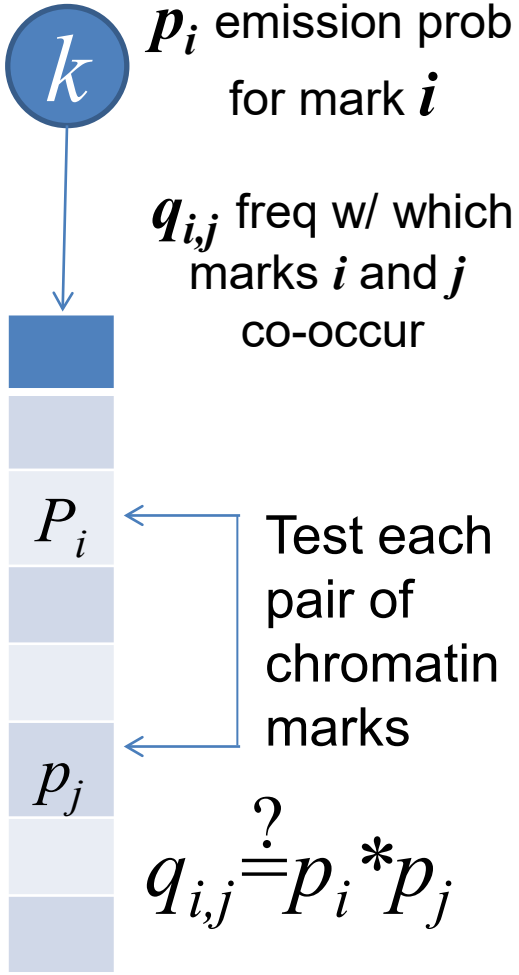
1. Introduction to Epigenomics
   - Overview of epigenomics, Diversity of Chromatin modifications
   - Antibodies, ChIP-Seq, data generation projects, raw data
2. Primary data processing: Read mapping, Peak calling
   - Read mapping: Hashing, Suffix Trees, Burrows-Wheeler Transform
   - Quality Control, Cross-correlation, Peak calling, IDR (similar to FDR)
3. Discovery and characterization of chromatin states
   - HMM Foundations, Generating, Parsing, Decoding, Learning
   - ChromHMM: Multi-variate HMM for chromatin state learning
4. Model complexity: selecting the number of states/marks
   - Capturing dependencies. State-conditional mark independence
   - Selecting the number of states, selecting number of marks
5. Learning chromatin states jointly across multiple cell types
   - Stacking vs. concatenation approach for joint multi-cell type learning
   - Defining activity profiles for linking enhancer regulatory networks

State-conditional mark independence

Do hidden states actually capture dependencies between marks?

Pairwise Expected vs. Observed Mark Co-Occurence

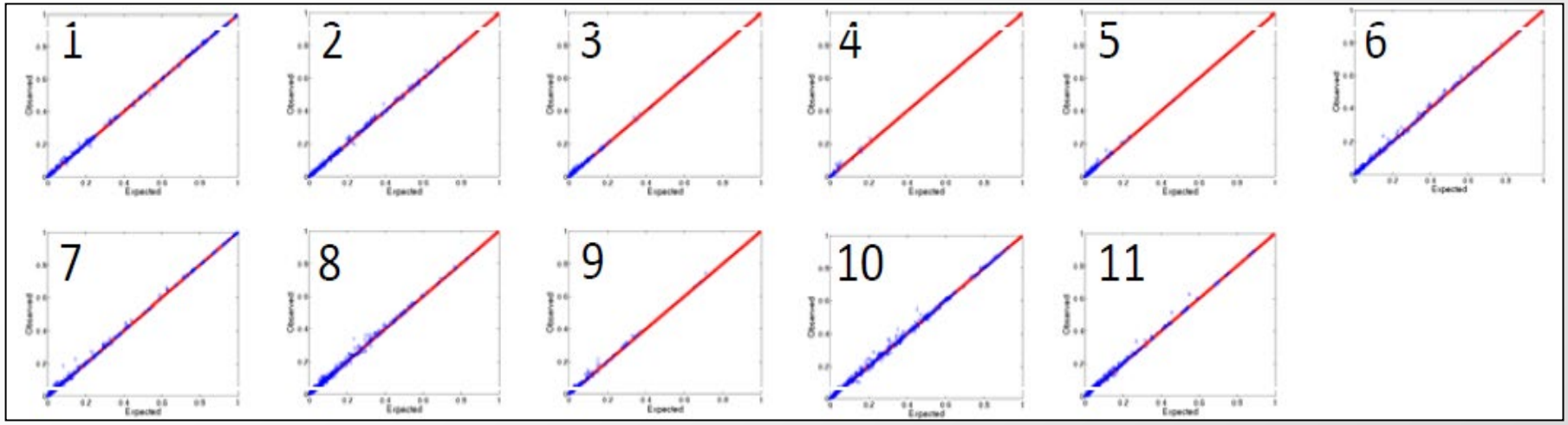




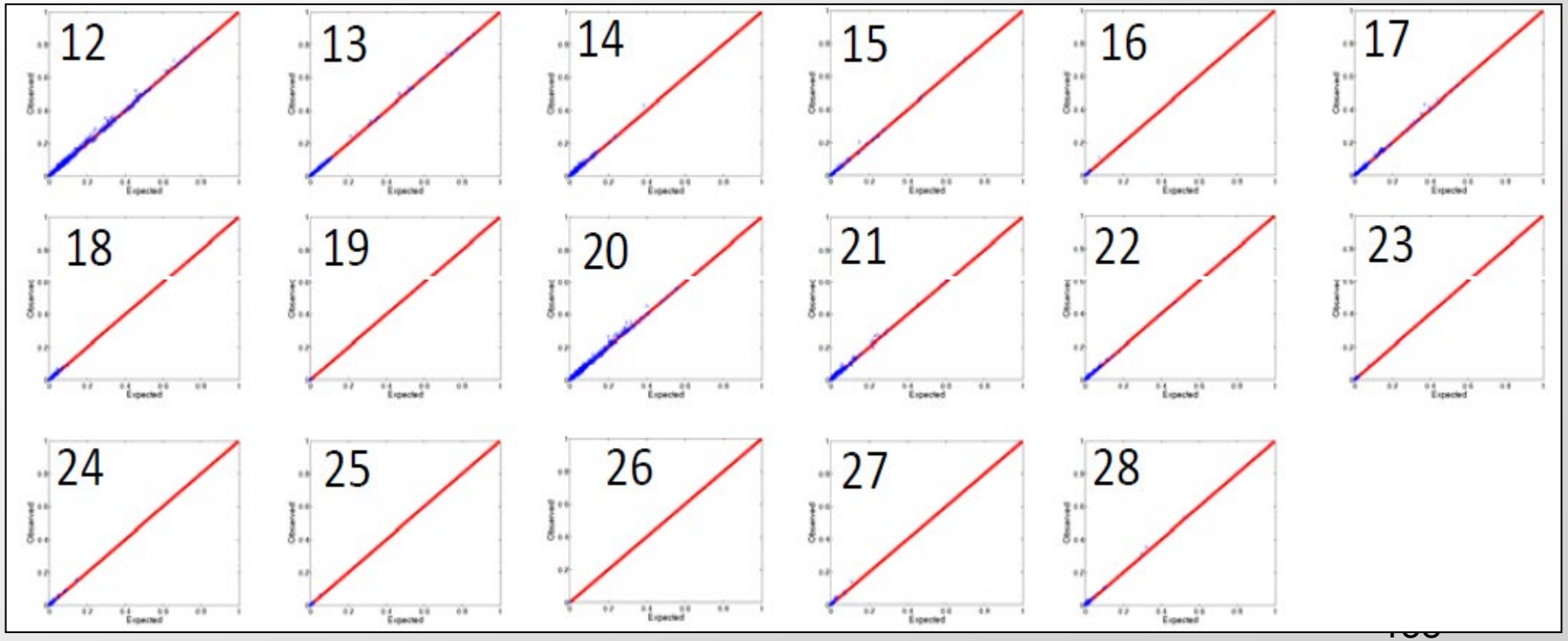
Multi-variate HMM emits entire vector of marks at a time
Model assumes mark independence **conditional** upon state
In fact, it specifically seeks to **capture** these dependencies

Test conditional independence for each state

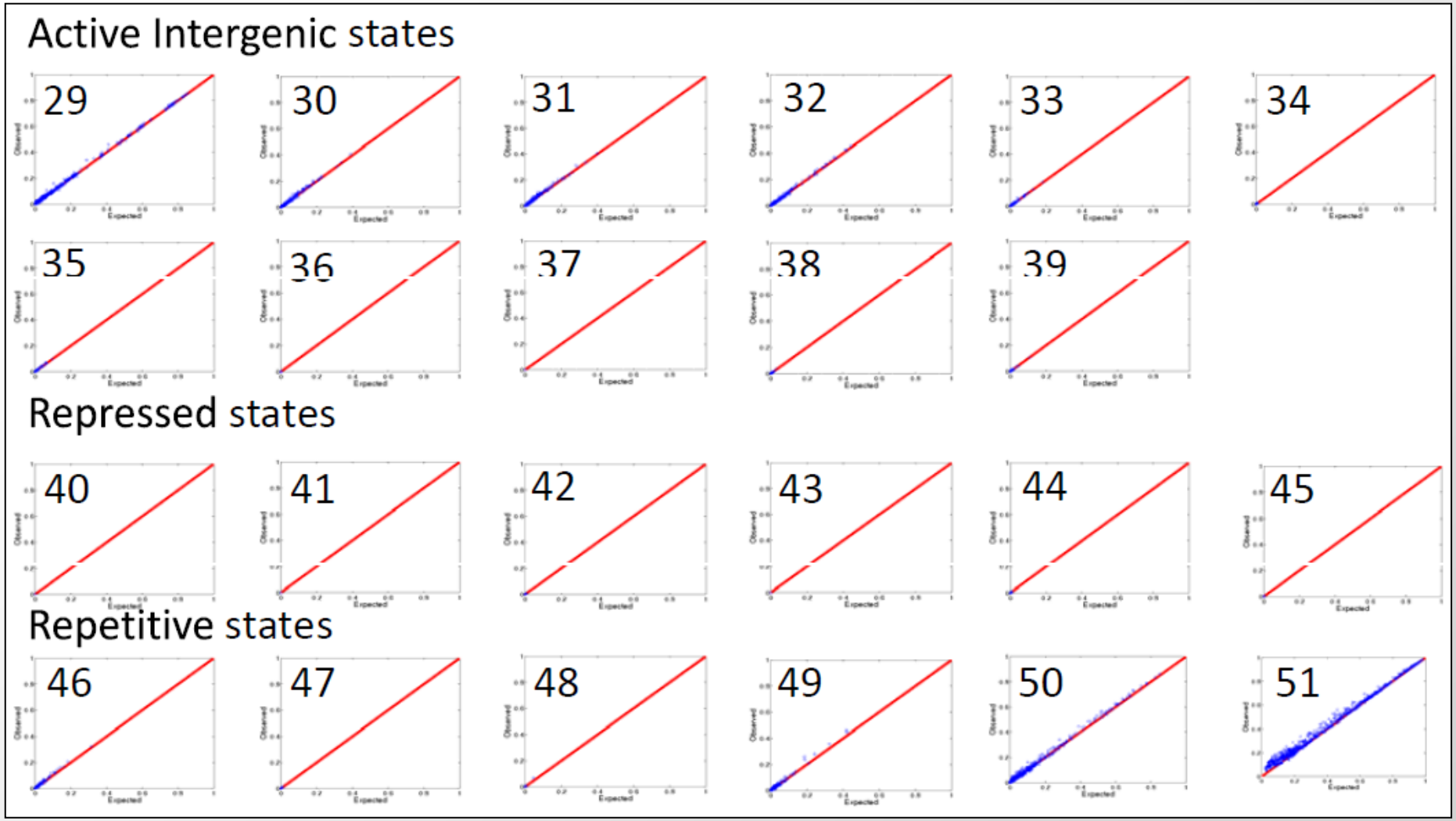
Promoter states



Transcribed states

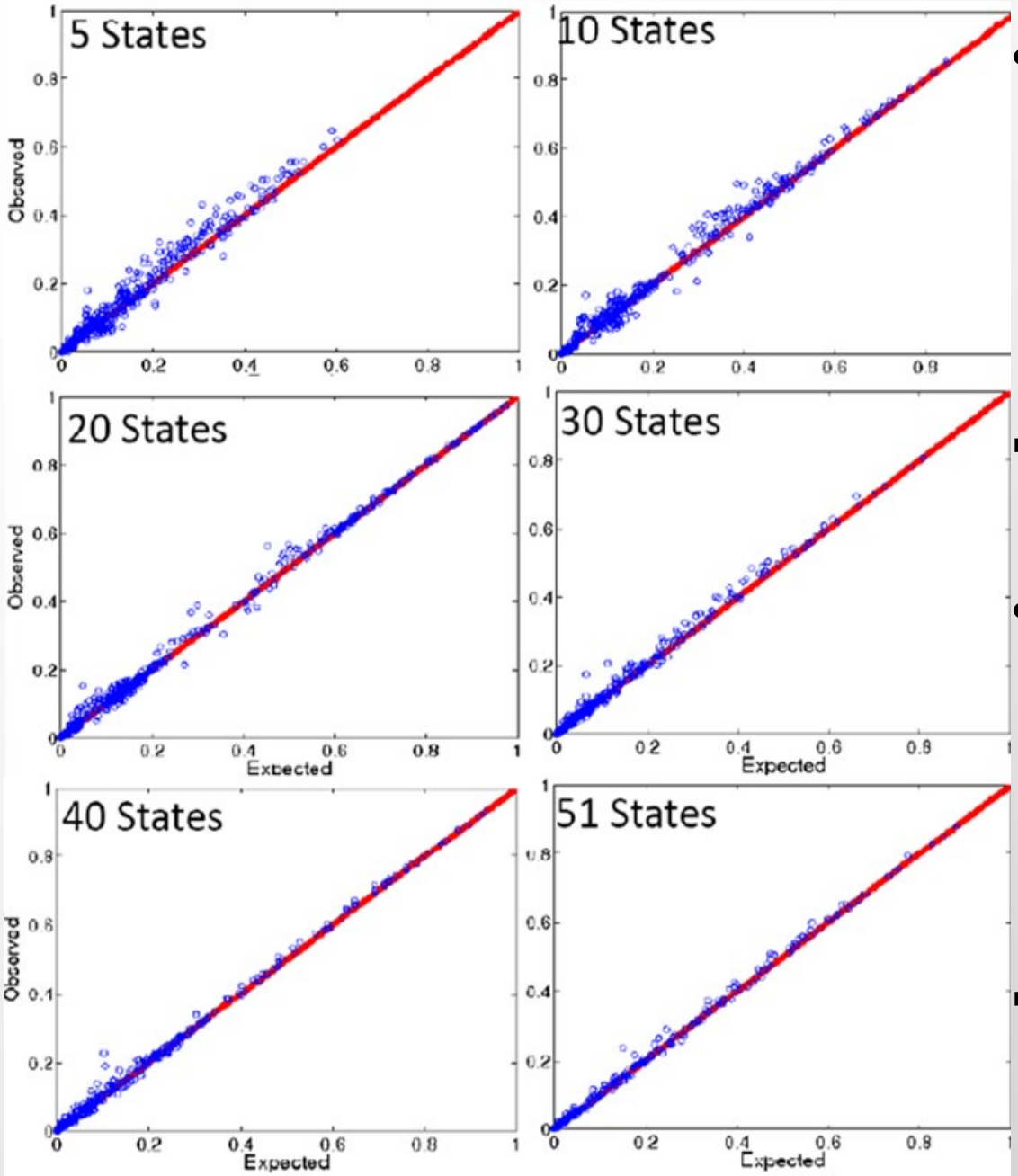


Non-independence reveals cases of model violation



- **Repetitive states show more dependencies**
- **Conditional independence does not hold**

As more states are added, dependencies captured



- With only 5 states in HMM, not enough power to distinguish different properties

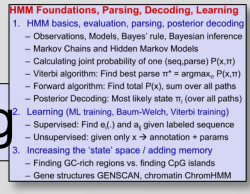
➔ Dependencies remain

- As model complexity increases, states learned become more precise

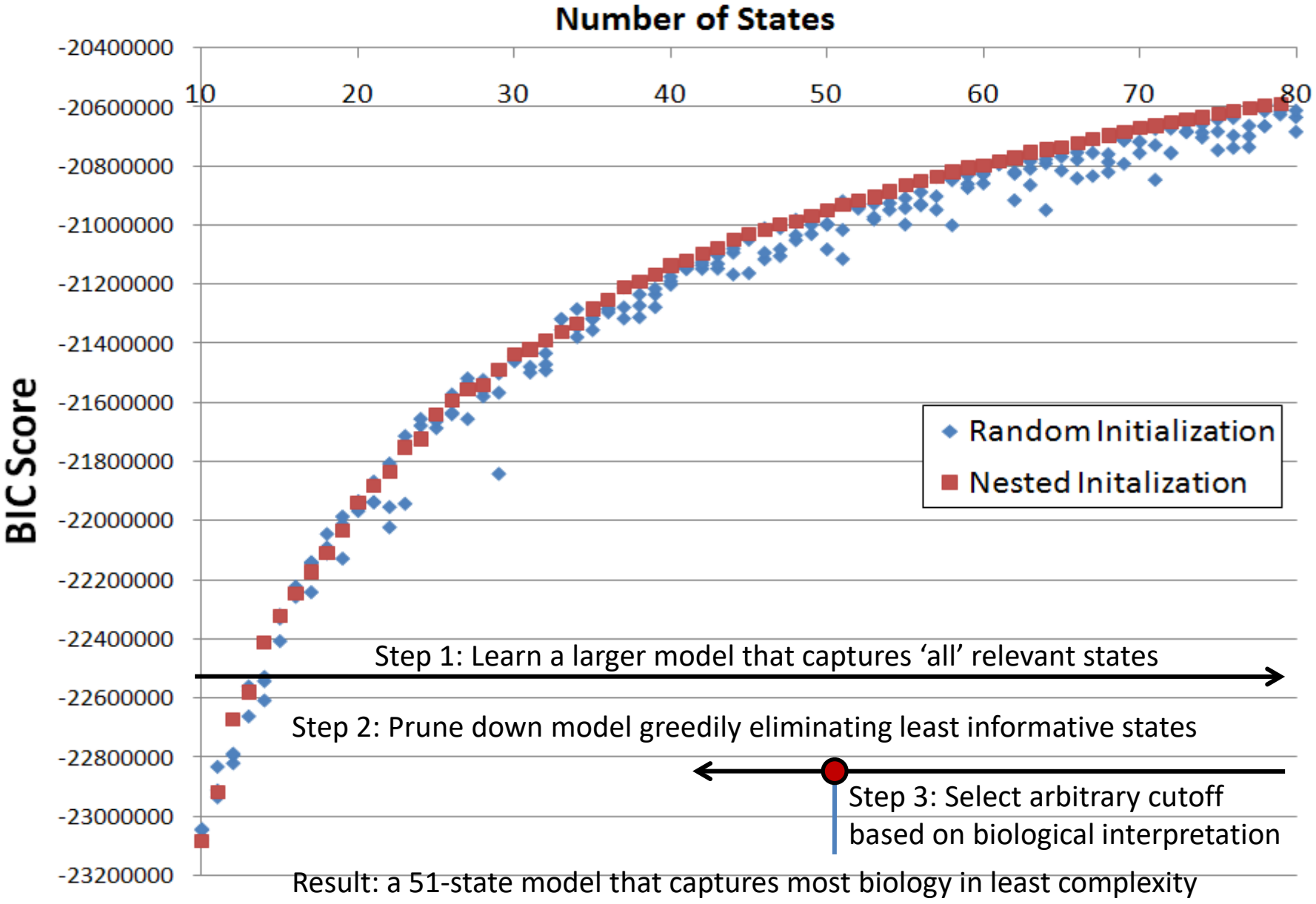
➔ Dependencies captured

Goals for today: Computational Epigenomics

1. Introduction to Epigenomics
   - Overview of epigenomics, Diversity of Chromatin modifications
   - Antibodies, ChIP-Seq, data generation projects, raw data
2. Primary data processing: Read mapping, Peak calling
   - Read mapping: Hashing, Suffix Trees, Burrows-Wheeler Transform
   - Quality Control, Cross-correlation, Peak calling, IDR (similar to FDR)
3. Discovery and characterization of chromatin states
   - HMM Foundations, Generating, Parsing, Decoding, Learning
   - ChromHMM: Multi-variate HMM for chromatin state learning
4. Model complexity: selecting the number of states/marks
   - Capturing dependencies. State-conditional mark independence
   - Selecting the number of states, selecting number of marks
5. Learning chromatin states jointly across multiple cell types
   - Stacking vs. concatenation approach for joint multi-cell type learning
   - Defining activity profiles for linking enhancer regulatory networks

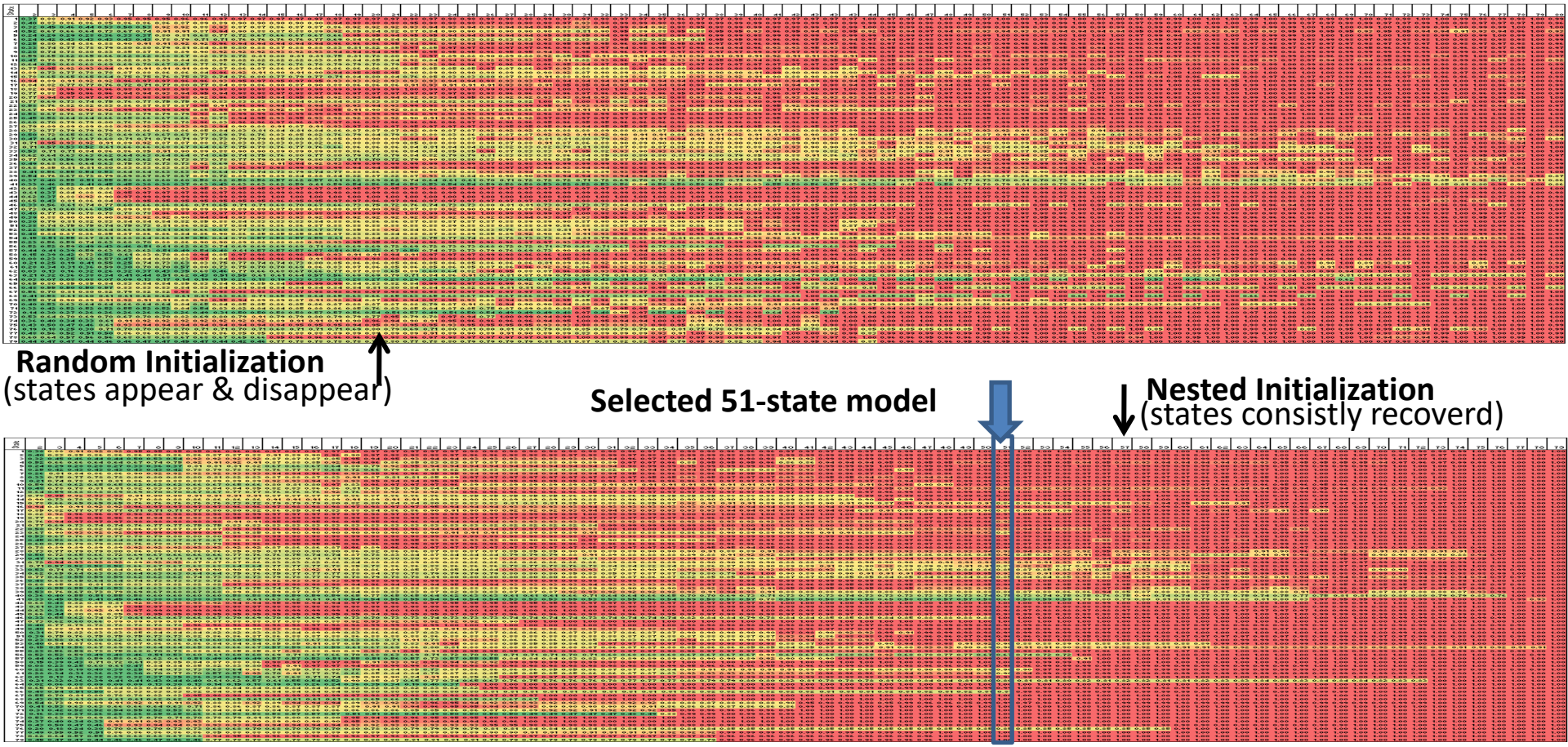


Comparison of BIC Score vs. Number of States for Random and Nested Initialization



- Standard model selection criteria fail due to genome complexity: more states always preferred
- Instead: Start w/complex model, keep informative states, prune redundant states. Pick cutoff

Recovery of 79-state model in random vs. nested initialization



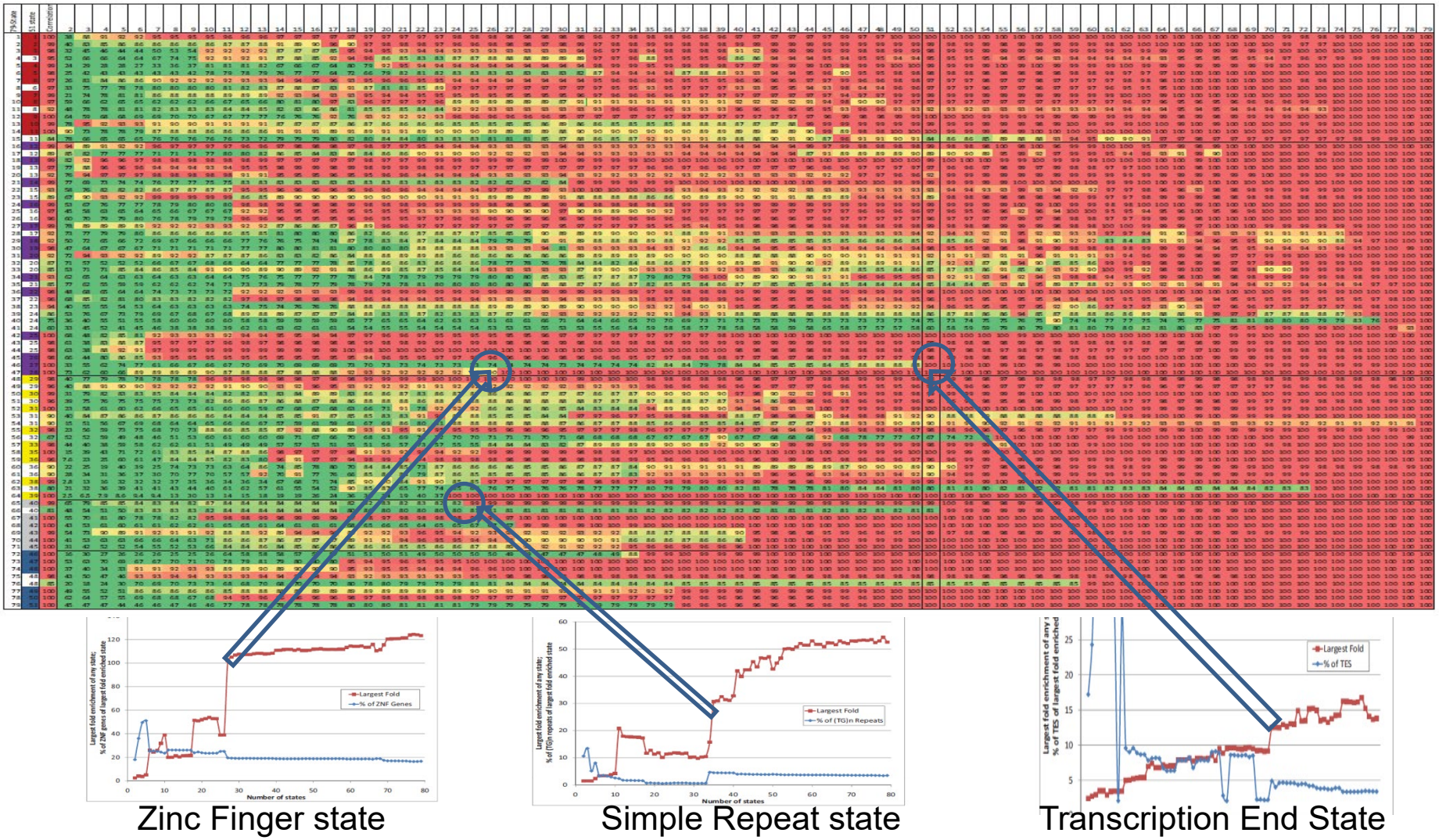
Nested initialization approach:

- **First pass:** learn models of increasing complexity
- **Second pass:** form nested set of emission parameter initializations by greedily removing states from best BIC model found

Nested models criteria:

- Maximize sum of correlation of emission vectors with nested model
- Models learned in parallel

Functional recovery with increasing numbers of states



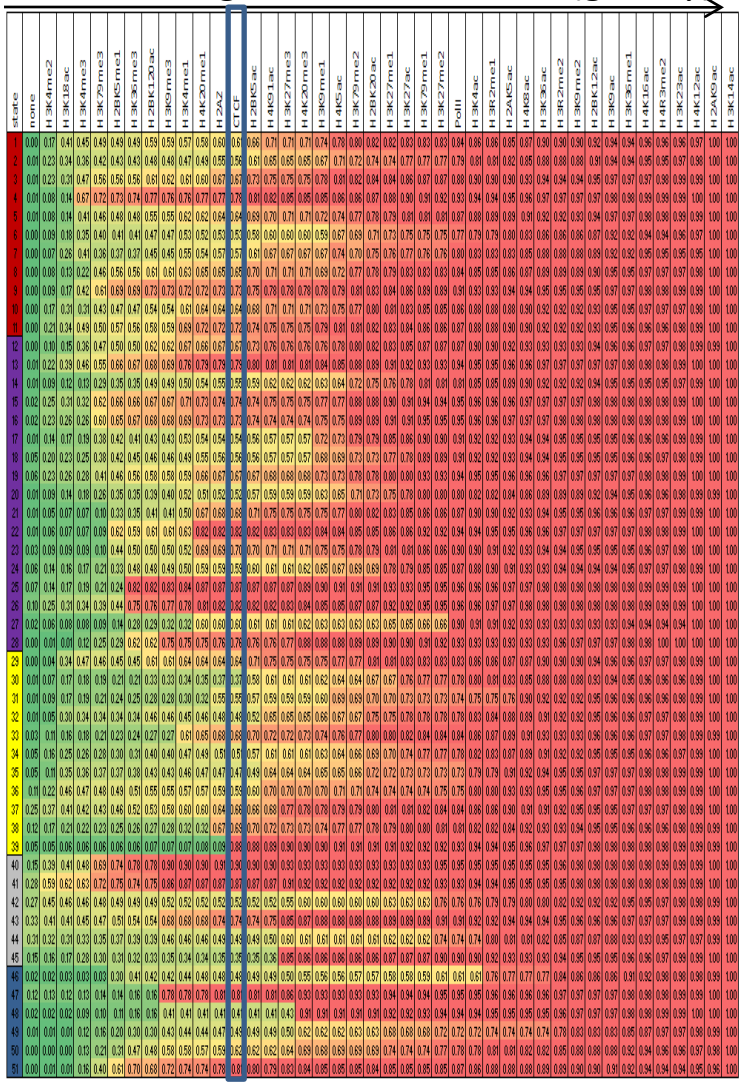
Zinc Finger state | Simple Repeat state | Transcription End State

- Red: Maximum fold functional enrichment for corresponding biological category
- Blue: Percent of that functional category that overlaps regions annotated to this state
- Top plot: Correlation of emission parameter vector for that state to closest state

Chromatin state recovery with increasing numbers of marks

Which states are well-recovered?

Increasing numbers of marks (greedy)



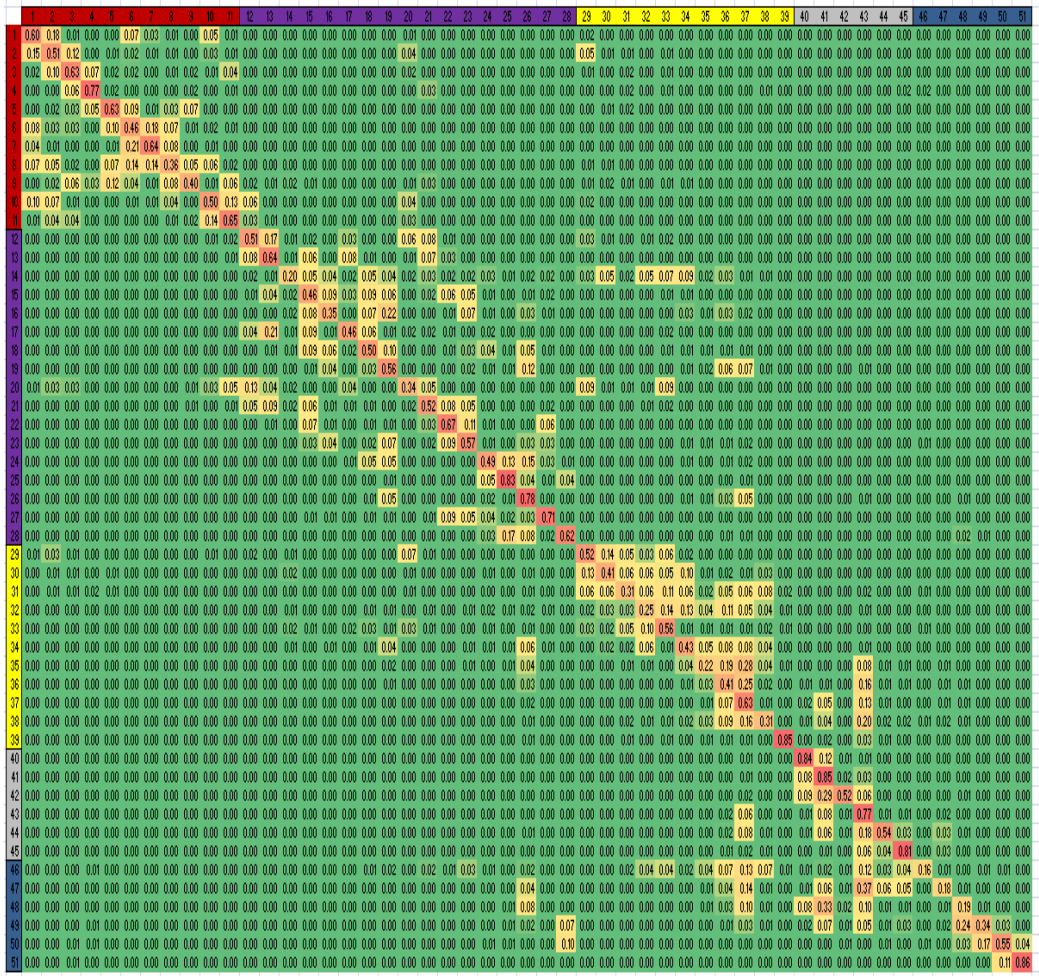
State Inferred with all 41 marks

Recovery of states with increasing number of marks

Precisely what mistakes are made?

(for a given subset of 11 ENCODE marks)

State Inferred with subset of marks

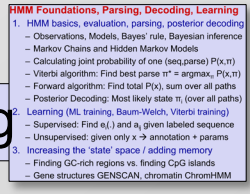


State Inferred with all 41 marks

State confusion matrix with 11 ENCODE marks

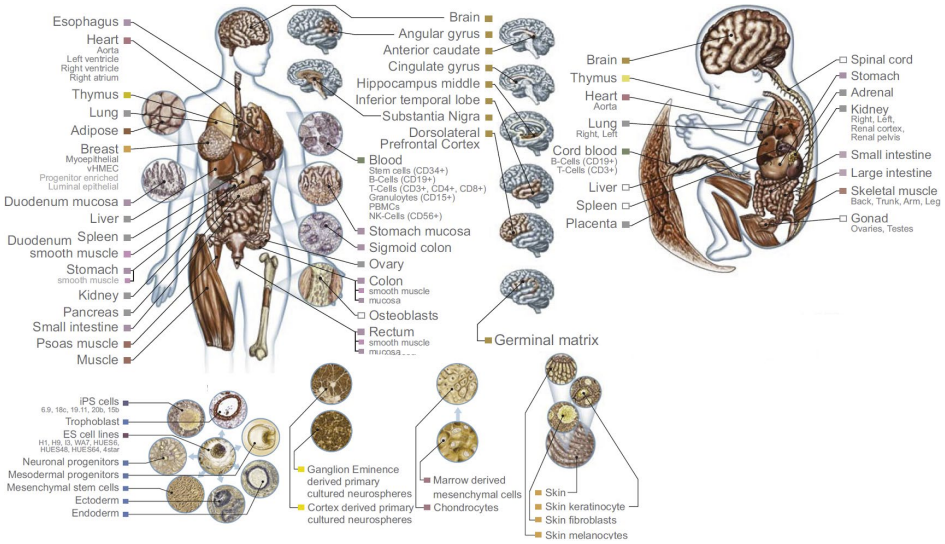
Goals for today: Computational Epigenomics

1. Introduction to Epigenomics
   - Overview of epigenomics, Diversity of Chromatin modifications
   - Antibodies, ChIP-Seq, data generation projects, raw data
2. Primary data processing: Read mapping, Peak calling
   - Read mapping: Hashing, Suffix Trees, Burrows-Wheeler Transform
   - Quality Control, Cross-correlation, Peak calling, IDR (similar to FDR)
3. Discovery and characterization of chromatin states
   - HMM Foundations, Generating, Parsing, Decoding, Learning
   - ChromHMM: Multi-variate HMM for chromatin state learning
4. Model complexity: selecting the number of states/marks
   - Selecting the number of states, selecting number of marks
   - Capturing dependencies and state-conditional mark independence
5. Learning chromatin states jointly across multiple cell types
   - Stacking vs. concatenation approach for joint multi-cell type learning
   - Defining activity profiles for linking enhancer regulatory networks



Epigenomic mapping across 100+ tissues/cell types

Diverse tissues and cells



Adult tissues and cells (brain, muscle, heart, digestive, skin, adipose, lung, blood…)

Fetal tissues (brain, skeletal muscle, heart, digestive, lung, cord blood…)

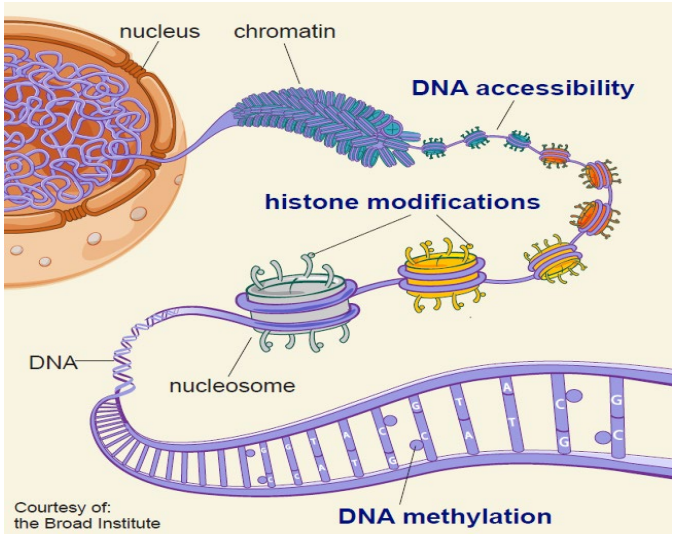
ES cells, iPS, differentiated cells (meso/endo/ectoderm, neural, mesench…)

ROADMAP epigenomics PROJECT

ENCODE

X

Diverse epigenomic assays



Histone modifications
- **H3K4me3, H3K4me1, H3K36me3**
- **H3K27me3, H3K9me3, H3K27/9ac**
- **+20 more**

Open chromatin:
- **DNA accessibility**

DNA methylation:
- **WGBS, RRBS, MRE/MeDIP**

Gene expression
- **RNA-seq, Exon Arrays**

ENCODE: Study nine marks in nine human cell lines

9 marks

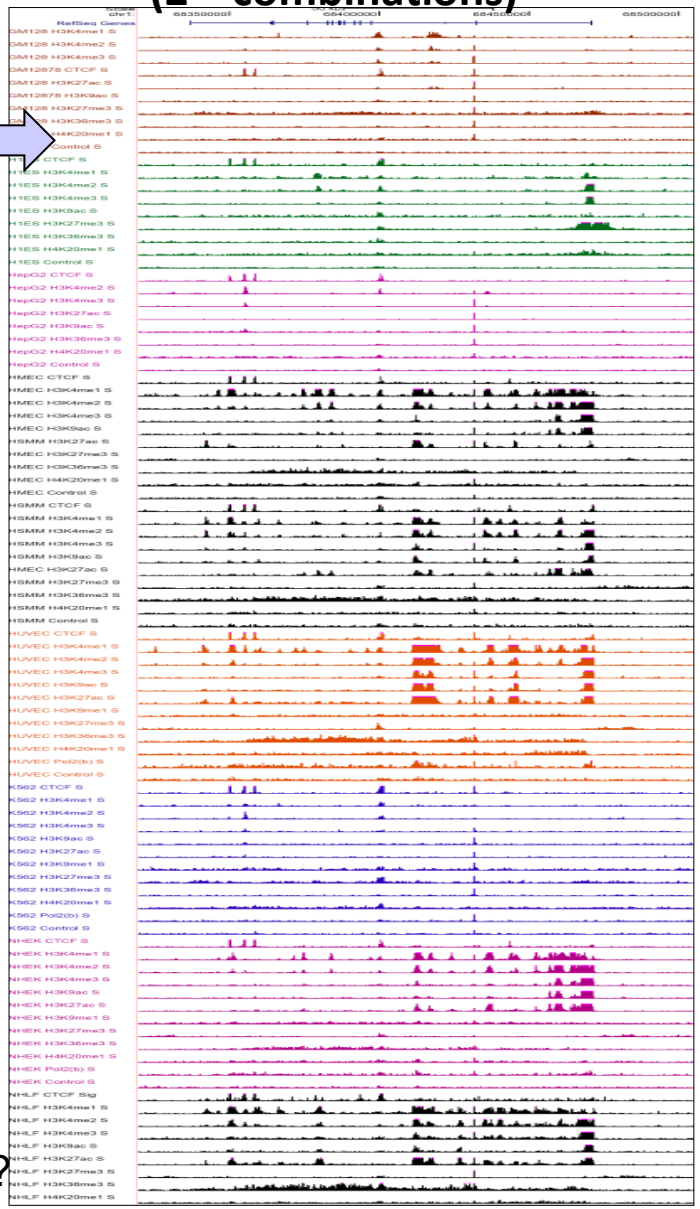
H3K4me1
H3K4me2
H3K4me3
H3K27ac
H3K9ac
H3K27me3
H4K20me1
H3K36me3
CTCF
+WCE
+RNA

X

9 human cell types

HUVEC	Umbilical vein endothelial
NHEK	Keratinocytes
GM12878	Lymphoblastoid
K562	Myelogenous leukemia
HepG2	Liver carcinoma
NHLF	Normal human lung fibroblast
HMEC	Mammary epithelial cell
HSMM	Skeletal muscle myoblasts
H1	Embryonic

81 Chromatin Mark Tracks
(2^{81} combinations)



Brad Bernstein ENCODE Chromatin Group

b.

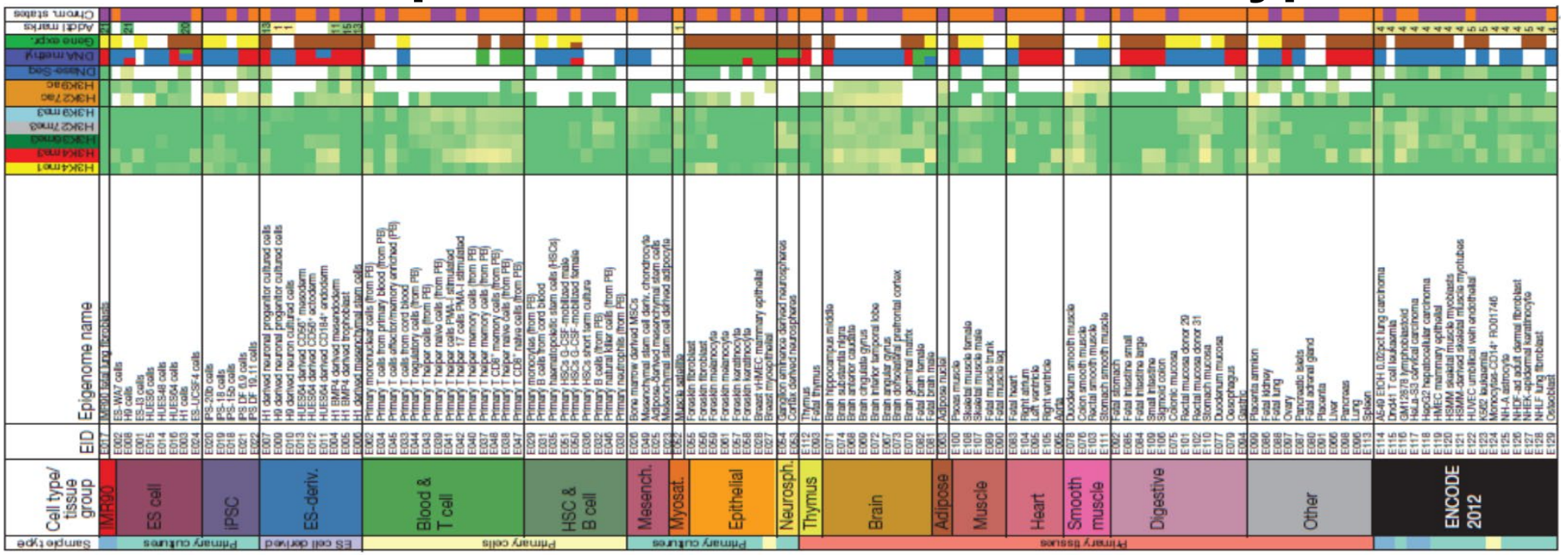
Chromatin States	State	CTCF	H3K27me3	H3K36me3	H4K20me1	H3K4me1	H3K4me2	H3K4me3	H3K27ac	H3K9ac	WCE
	1	16	2	2	6	17	93	99	96	98	2
	2	12	2	6	9	53	94	95	14	44	1
	3	13	72	0	9	48	78	49	1	10	1
	4	11	1	15	11	96	99	75	97	86	4
	5	5	0	10	3	88	57	5	84	25	1
	6	7	1	1	3	58	75	8	6	5	1
	7	2	1	2	1	56	3	0	6	2	1
	8	92	2	1	3	6	3	0	0	1	1
	9	5	0	43	43	37	11	2	9	4	1
	10	1	0	47	3	0	0	0	0	0	1
	11	0	0	3	2	0	0	0	0	0	0
	12	1	27	0	2	0	0	0	0	0	0
	13	0	0	0	0	0	0	0	0	0	0
	14	22	28	19	41	6	5	26	5	13	37
	15	85	85	91	88	76	77	91	73	85	78

Chromatin Mark Observation Frequency (%)

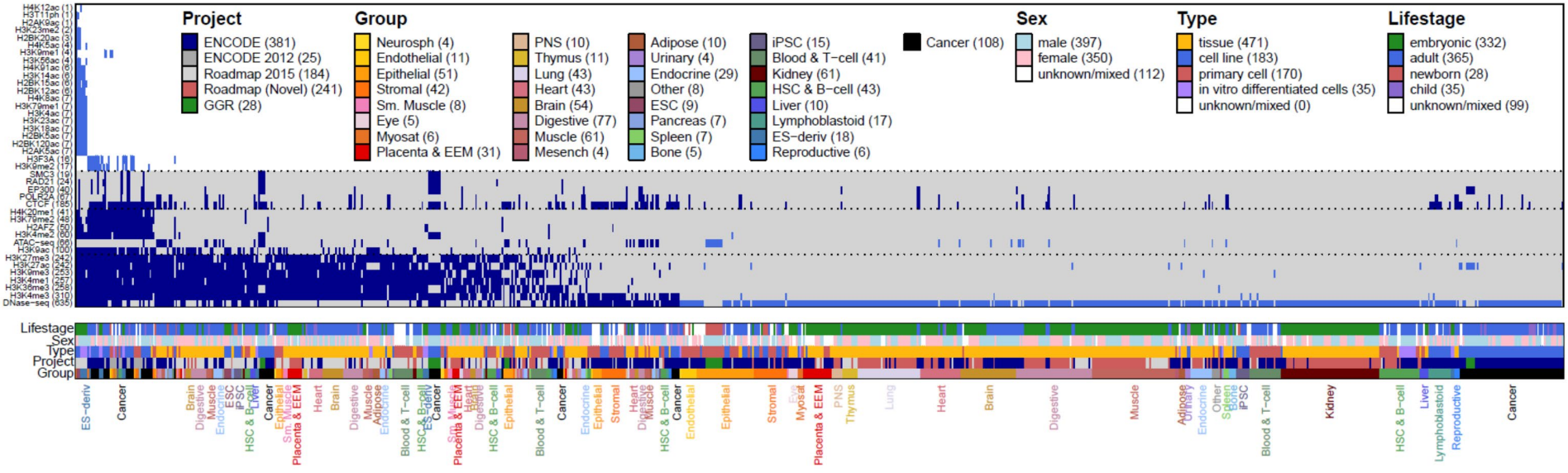
How to learn single set of chromatin states?

Ernst et al, *Nature* 2011

Roadmap 2015: 12 marks x 127 cell types



ENCODE 2019: 40 marks x 834 cell types



Solution 1: Learn independent models and cluster

state	CTCF	H3K27me3	H3K36me3	H4K20me1	H3K4me1	H3K4me2	H3K4me3	H3K27ac	H3K9ac	WCE
1	13.2	72.0	0.2	9.1	47.9	77.8	49.5	1.3	10.2	0.7
2	11.9	1.9	6.1	9.0	52.7	93.7	95.0	14.1	44.1	0.9
3	16.4	1.5	2.4	5.5	17.0	92.6	99.0	95.7	98.1	1.9
4	11.4	0.6	14.5	11.3	96.3	99.3	75.1	97.2	85.7	3.7
5	5.3	0.2	9.5	2.6	88.1	56.8	5.3	84.4	24.9	1.5
6	6.7	0.9	1.0	3.2	58.3	74.7	8.4	5.8	5.4	0.8
7	1.6	0.6	1.6	1.3	56.5	2.7	0.4	5.9	1.6	0.6
8	91.5	1.8	0.9	2.8	6.3	3.3	0.4	0.5	1.0	0.8
9	4.6	0.3	43.2	43.1	36.5	11.5	1.9	9.1	3.9	1.3
10	1.2	0.1	47.2	2.7	0.4	0.0	0.1	0.3	0.3	0.5
11	0.4	0.1	2.7	1.7	0.2	0.1	0.1	0.2	0.3	0.4
12	0.9	26.8	0.0	2.1	0.4	0.1	0.1	0.1	0.1	0.4
13	0.2	0.4	0.0	0.1	0.1	0.0	0.0	0.0	0.1	0.1
14	21.9	27.9	19.1	41.0	5.7	4.8	25.9	5.3	13.1	37.5
15	85.2	85.1	91.5	88.4	76.2	76.9	90.8	72.7	85.0	78.3

Promoter

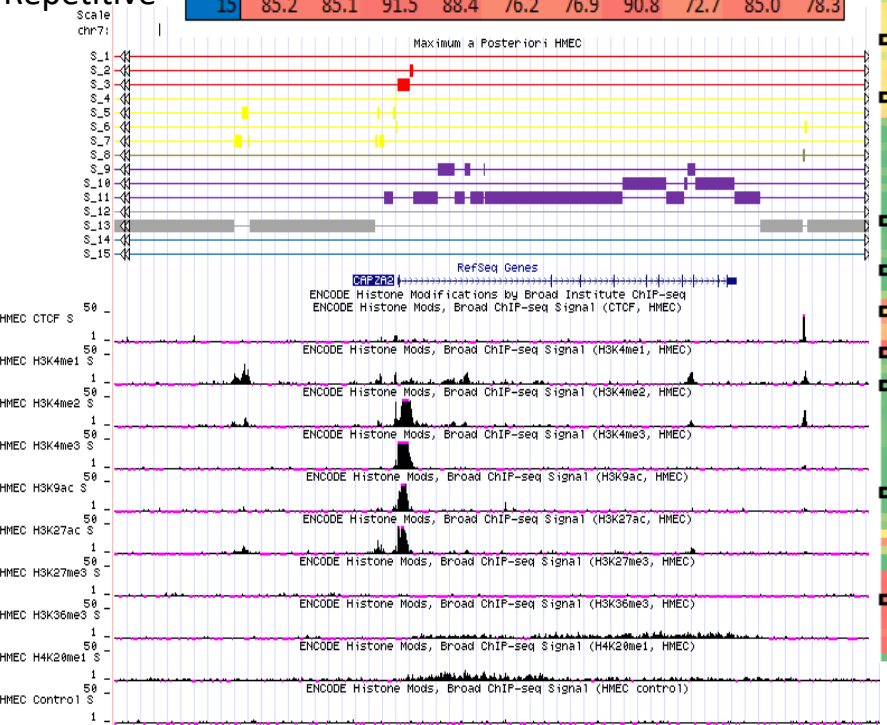
Candidate enhancer

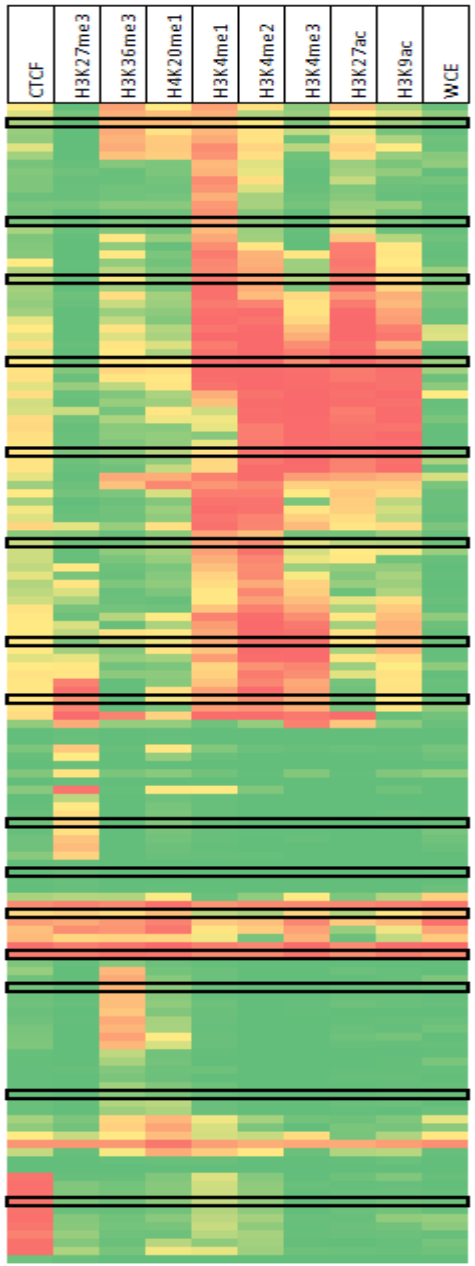
Insulator

Transcribed

Repressive

Repetitive





Basic approach:

a) Train a k-state model in each cell type independently

b) Cluster models learned independently

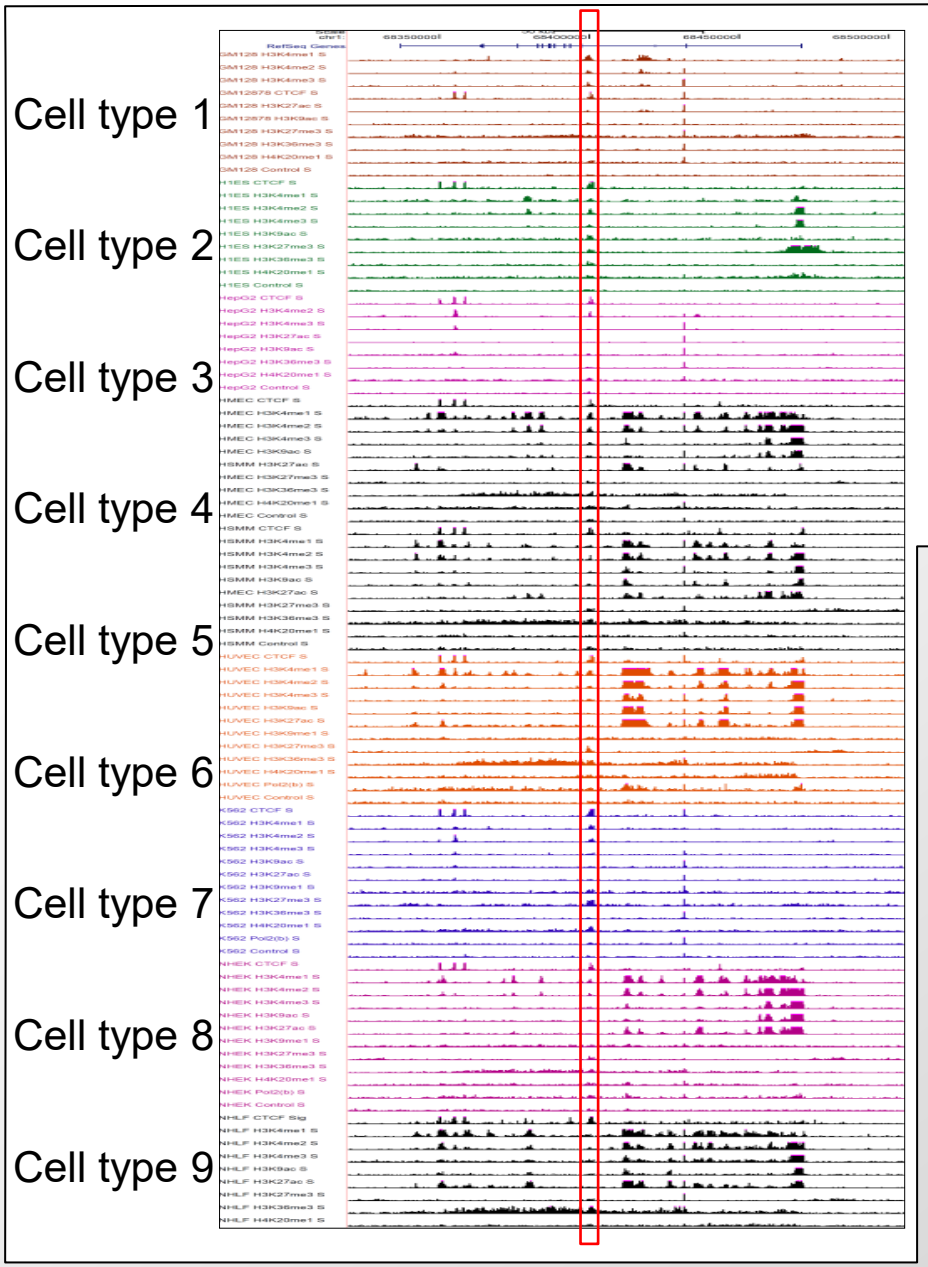
c) Merge clusters and re-apply to each cell type

How to cluster

a) Using emission probability matrix: most similar definitions

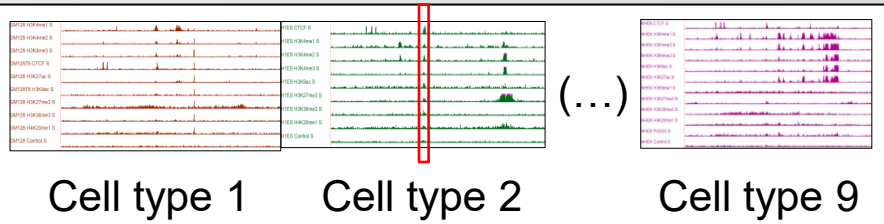
b) Using genome annotation: posterior probability decoding

Joint learning of states across multiple cell types



Solution 2: Stacking

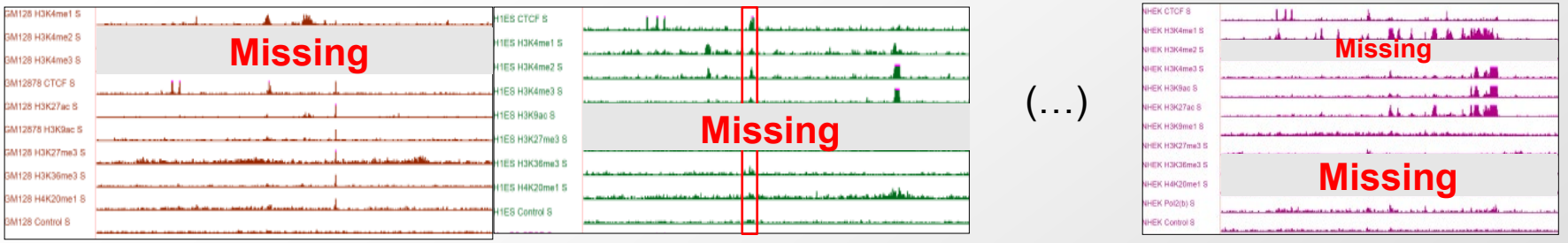
- Learns each combination of activity as a separate state
- Ex: ES-specific enhancers: enhancer marks in ES, no marks in other cell types



Solution 3: Concatenation

- Requires that profiled marks are the same (or treat as missing data)
- Ensures common state definitions across cell types

Joint learning with different subsets of marks (Solution 3)



Cell type 1 Cell type 2 Cell type 9

Option (a) Treat missing tracks as missing data

- EM framework allows for unspecified data points
- As long as pairwise relationship observed in some cell type

Option (b) Chromatin mark imputation

- Explicitly predict max-likelihood chromatin track for missing data
- Less powerful if ultimate goal is chromatin state learning

ENCODE: Study nine marks in nine human cell lines

9 marks

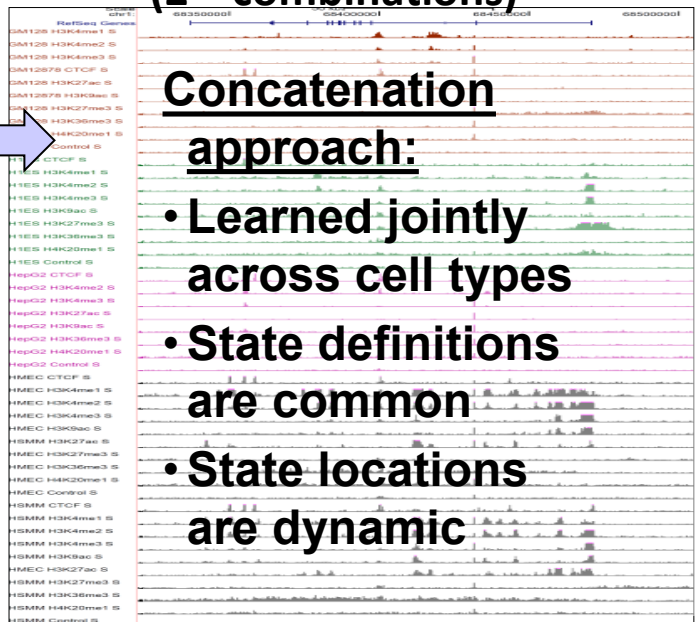
H3K4me1
H3K4me2
H3K4me3
H3K27ac
H3K9ac
H3K27me3
H4K20me1
H3K36me3
CTCF
+WCE
+RNA

X

9 human cell types

HUVEC	Umbilical vein endothelial
NHEK	Keratinocytes
GM12878	Lymphoblastoid
K562	Myelogenous leukemia
HepG2	Liver carcinoma
NHLF	Normal human lung fibroblast
HMEC	Mammary epithelial cell
HSMM	Skeletal muscle myoblasts
H1	Embryonic

81 Chromatin Mark Tracks (2^{81} combinations)



Concatenation approach:

- Learned jointly across cell types
- State definitions are common
- State locations are dynamic

Brad Bernstein ENCODE Chromatin Group

b.

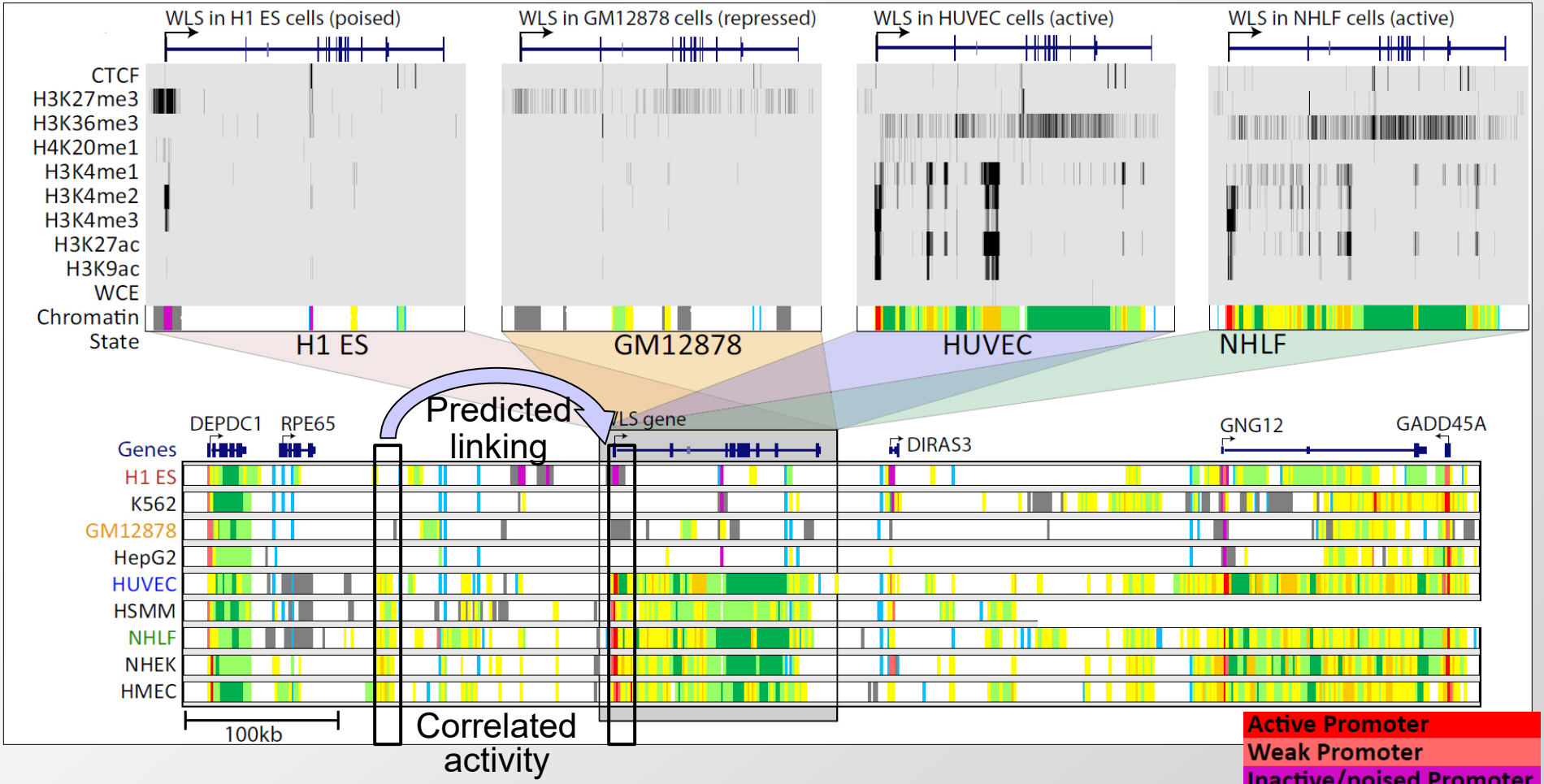
State	CTCF	H3K27me3	H3K36me3	H4K20me1	H3K4me1	H3K4me2	H3K4me3	H3K27ac	H3K9ac	WCE	Coverage: Median	Coverage: H1 ES	Coverage: GM	Median Length	+/-2kb TSS	Conserved non-exon	DNase (K562)	C-Myc (K562)	NF-κB (GM12878)	Transcript	Nuclear Lamina (NHLF)	Candidate state annotation
1	16	2	2	6	17	93	99	96	98	2	0.6	0.5	1.2	1.0	83	3.8	23.3	82.0	40.7	0.2	0.15	Active Promoter
2	12	2	6	9	53	94	95	14	44	1	0.5	1.2	1.3	0.4	58	2.8	15.3	12.6	5.8	0.6	0.30	Weak Promoter
3	13	72	0	9	48	78	49	1	10	1	0.2	4.0	1.0	0.6	49	4.3	10.8	3.1	1.0	0.4	0.68	Inactive/poised Promoter
4	11	1	15	11	96	99	75	97	86	4	0.7	0.1	1.1	0.6	23	2.7	23.1	31.8	49.0	1.3	0.05	Strong enhancer
5	5	0	10	3	88	57	5	84	25	1	1.2	0.2	0.7	0.6	3	1.8	13.6	6.3	15.8	1.4	0.10	Strong enhancer
6	7	1	1	3	58	75	8	6	5	1	0.9	1.3	1.0	0.2	17	2.4	11.9	5.7	7.0	1.1	0.31	Weak/poised enhancer
7	2	1	2	1	56	3	0	6	2	1	1.9	1.2	1.1	0.4	4	1.5	5.1	0.6	2.4	1.3	0.20	Weak/poised enhancer
8	92	2	1	3	6	3	0	0	1	1	0.5	1.4	1.0	0.4	3	1.5	12.8	2.5	1.2	1.1	0.61	Insulator
9	5	0	43	43	37	11	2	9	4	1	0.7	1.3	1.0	0.8	4	1.1	4.5	0.7	0.8	2.4	0.02	Transcriptional transition
10	1	0	47	3	0	0	0	0	0	1	4.3	0.6	1.2	3.0	1	0.9	0.3	0.0	0.0	2.5	0.11	Transcriptional elongation
11	0	0	3	2	0	0	0	0	0	0	12.5	1.3	0.8	2.6	2	0.9	0.3	0.0	0.1	1.9	0.24	Weak transcribed
12	1	27	0	2	0	0	0	0	0	0	4.1	0.3	0.7	2.8	5	1.4	0.3	0.0	0.1	0.8	0.63	Polycomb-repressed
13	0	0	0	0	0	0	0	0	0	0	71.4	1.0	1.0	10.0	1	0.9	0.1	0.0	0.0	0.7	1.30	Heterochrom; low signal
14	22	28	19	41	6	5	26	5	13	37	0.1	0.9	1.2	0.6	3	0.4	1.9	0.3	0.2	0.4	1.44	Repetitive/CNV
15	85	85	91	88	76	77	91	73	85	78	0.1	0.9	1.0	0.2	1	0.2	5.9	9.5	7.4	0.4	1.30	Repetitive/CNV

Chromatin States

Chromatin Mark Observation Frequency (%)

(%) (fold) (kb) (%) Functional enrichments (fold)

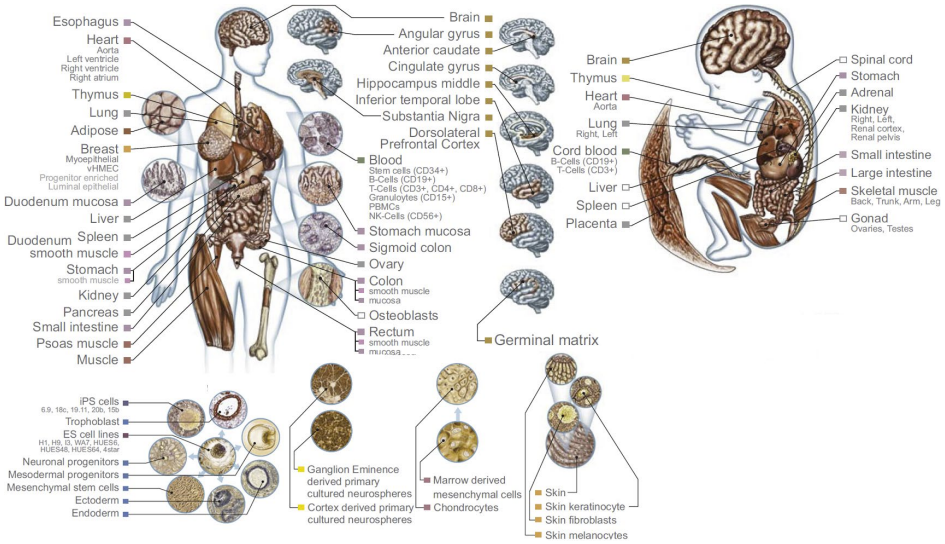
Chromatin states dynamics across nine cell types



- Single annotation track for each cell type
- Summarize cell-type activity at a glance
- Can study 9-cell activity pattern across ↓

Epigenomic mapping across 100+ tissues/cell types

Diverse tissues and cells



Adult tissues and cells (brain, muscle, heart, digestive, skin, adipose, lung, blood…)

Fetal tissues (brain, skeletal muscle, heart, digestive, lung, cord blood…)

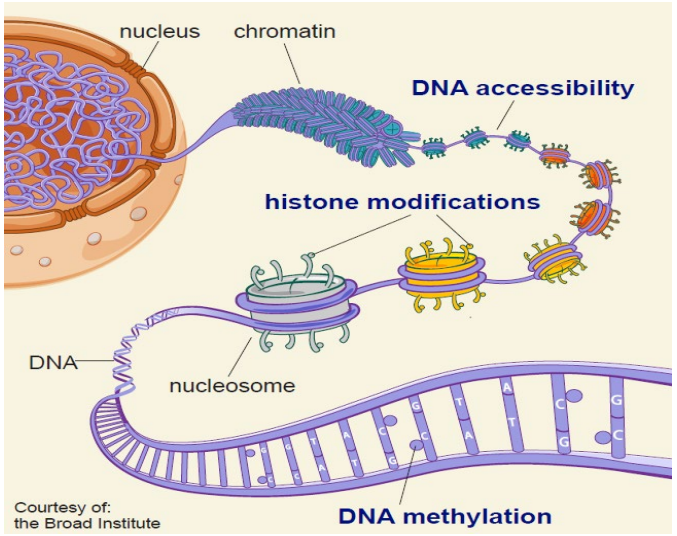
ES cells, iPS, differentiated cells (meso/endo/ectoderm, neural, mesench…)

ROADMAP epigenomics PROJECT

ENCODE

X

Diverse epigenomic assays



Histone modifications
- **H3K4me3, H3K4me1, H3K36me3**
- **H3K27me3, H3K9me3, H3K27/9ac**
- **+20 more**

Open chromatin:
- **DNA accessibility**

DNA methylation:
- **WGBS, RRBS, MRE/MeDIP**

Gene expression
- **RNA-seq, Exon Arrays**

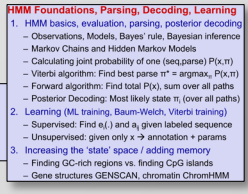
Chromatin state annotations across 127 epigenomes

Reveal epigenomic variability: enh/prom/tx/repr/het

Anshul Kundaje

Goals for today: Computational Epigenomics

1. Introduction to Epigenomics
   - Overview of epigenomics, Diversity of Chromatin modifications
   - Antibodies, ChIP-Seq, data generation projects, raw data
2. Primary data processing: Read mapping, Peak calling
   - Read mapping: Hashing, Suffix Trees, Burrows-Wheeler Transform
   - Quality Control, Cross-correlation, Peak calling, IDR (similar to FDR)
3. Discovery and characterization of chromatin states
   - HMM Foundations, Generating, Parsing, Decoding, Learning
   - ChromHMM: Multi-variate HMM for chromatin state learning
4. Model complexity: selecting the number of states/marks
   - Selecting the number of states, selecting number of marks
   - Capturing dependencies and state-conditional mark independence
5. Learning chromatin states jointly across multiple cell types
   - Stacking vs. concatenation approach for joint multi-cell type learning
   - Defining activity profiles for linking enhancer regulatory networks



HMM Foundations, Parsing, Decoding, Learning

1. HMM basics, evaluation, parsing, posterior decoding
   - Observations, Models, Bayes' rule, Bayesian inference
   - Markov Chains and Hidden Markov Models
   - Calculating joint probability of one (seq,parse) $P(x,\pi)$
   - Viterbi algorithm: Find best parse $\pi^* = \text{argmax}_\pi P(x,\pi)$
   - Forward algorithm: Find total $P(x)$, sum over all paths
   - Posterior Decoding: Most likely state π_i (over all paths)
2. Learning (ML training, Baum-Welch, Viterbi training)
   - Supervised: Find $e_i(.)$ and a_{ij} given labeled sequence
   - Unsupervised: given only x → annotation + params
3. Increasing the 'state' space / adding memory
   - Finding GC-rich regions vs. finding CpG isla
   - Gene structures GENSCAN, chromatin Chr