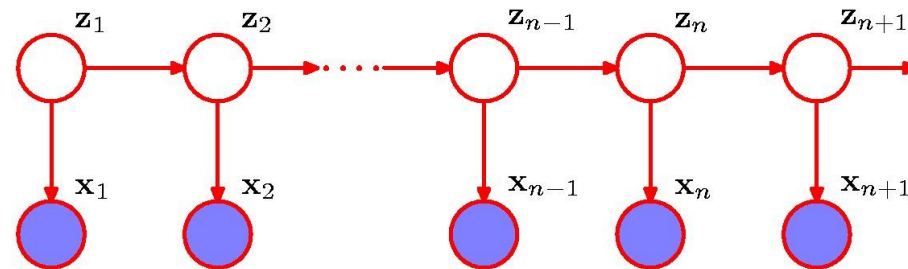


Hidden Markov Models

Hand In 3 - Gene finding



- A gene is a substring of the DNA sequence of A,C,G,T's
- A gene starts with a start-codon **atg**
- A gene ends with a stop-codon **taa**, **tag** or **tga**
- The number of nucleotides in a gene is a multiplum of 3

N: No gene

Even more b

There can be genes in

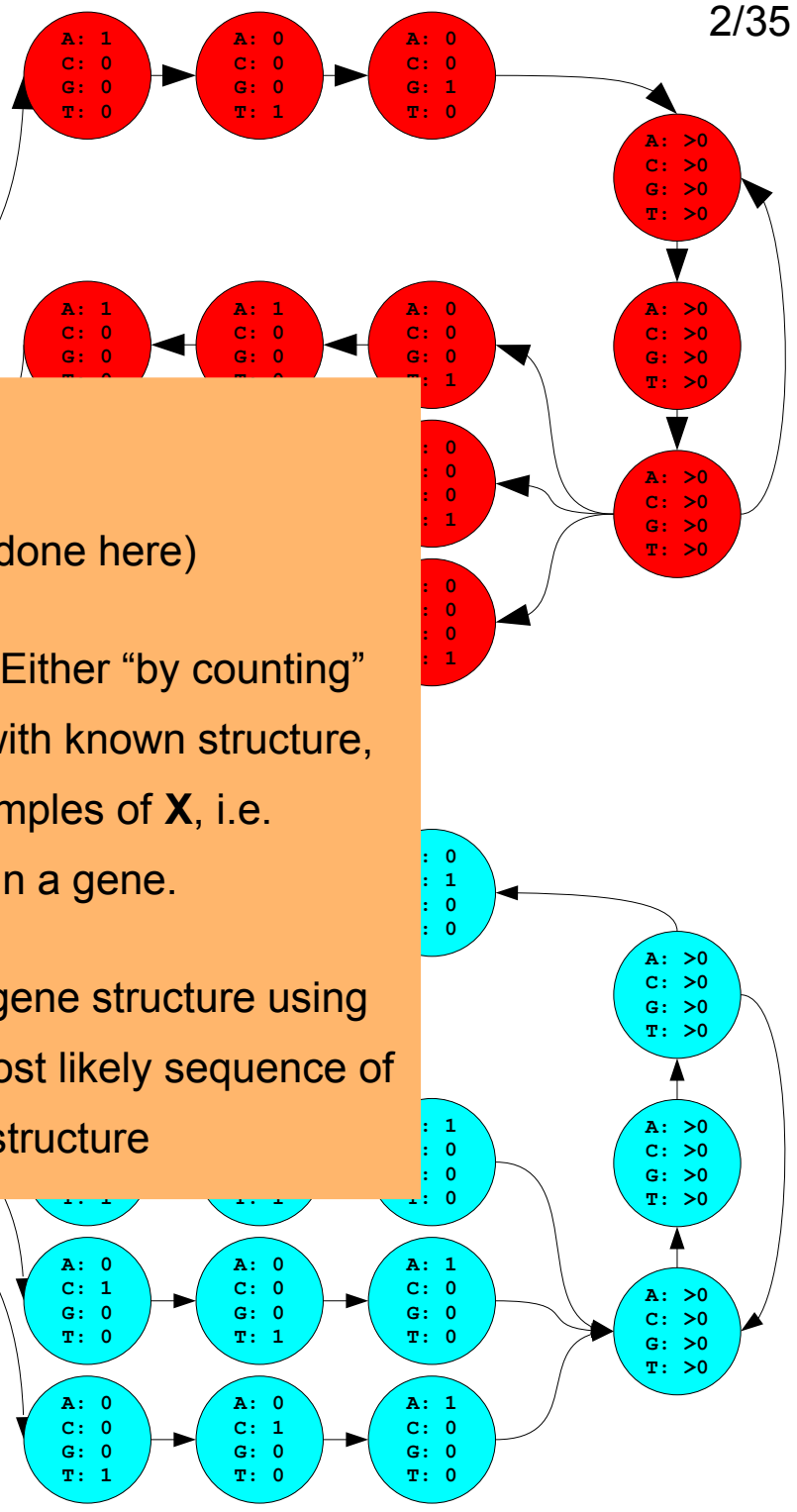
$$\pi_N = 1$$
$$\pi_C = 0$$

C: coding left-to-right

R: coding right-to-left

Gene finding

- Select initial model structure (e.g. as done here)
- Select model parameters by training. Either “by counting” from examples of (X,Z)’s, i.e. genes with known structure, or by EM- or Viterbi-training from examples of X, i.e. sequences which are known to contain a gene.
- Given a new sequence X, predict its gene structure using the Viterbi algorithm for finding the most likely sequence of underlying latent states, i.e. its gene structure





Hand in 3 - Machine Learning, Fall 2021

This project is about using a hidden Markov model for gene finding in prokaryotes as explained in class and explored in the exercises in week 47.

You must do the project in groups similar to the previous hand ins. The project is mandatory and each group must hand in a report describing their work and results as outlined in section Report below. Handin the report in pdf-format via BrightSpace no later than **Wednesday, December 8, 2021, at 23:59**.

Data set

You are give a data set containing 10 [Staphylococcus](#) genomes, each containing several genes (i.e. substring) obeying the "gene syntax" explained in class. The genomes are between 1.8 million and 2.8 million nucleotides long. For 5 of the genomes, you are also given the gene structure, i.e. the location of the genes. For the remaining 5 genomes, you only know that they contain genes according to the "gene syntax". You have already explored two of the genomes in the practical exercises in week 47.

The genomes and their annotations are given in [FASTA format](#). There are several Python libraries for reading sequences in FASTA format (use Google to find one), and you are welcome to use the simple reader function `read_fasta_file` below that you also used in the exercises in week 47.

The data is:

- The files [genome1.fa](#), [genome2.fa](#), [genome3.fa](#), [genome4.fa](#), [genome5.fa](#), [genome6.fa](#), [genome7.fa](#), [genome8.fa](#), [genome9.fa](#), and [genome10.fa](#) contain the 10 genomes.
- The files [true-ann1.fa](#), [true-ann2.fa](#), [true-ann3.fa](#), [true-ann4.fa](#), and [true-ann5.fa](#) contain the annotation of the first 5 genomes. The annotation is given in FASTA format as a sequence over the symbols N, C, and R. The symbol N, C, or R at position i in `true-annk.fa` gives the "state" of the nucleotide at position i in `genomek.fa`. N means that the nucleotide is non-coding. C means that the nucleotide is coding and part of a gene in the direction from left to right. R means that the nucleotide is coding and part of gene in the reverse direction from right to left. This is exactly as in the exercises in week 47.

... All the above files are available in [data-handin3.zip](#).

Data

You are given a data set containing 10 [Staphylococcus](#) genomes, each containing several genes (i.e. substring) obeying the "gene syntax" explained in class. The genomes are between 1.8 million and 2.8 million nucleotides long. For 5 of the genomes, you are also given the gene structure, i.e. the location of the genes. For the remaining 5 genomes, you only know that they contain genes according to the "gene syntax". You have already explored two of the genomes in the practical exercises in week 47.

The genomes and their annotations are given in [FASTA format](#). There are several Python libraries for reading sequences in FASTA format (use Google to find one), and you are welcome to use the simple reader function `read_fasta_file` below that you also used in the exercises in week 47.

The data is:

- The files [genome1.fa](#), [genome2.fa](#), [genome3.fa](#), [genome4.fa](#), [genome5.fa](#), [genome6.fa](#), [genome7.fa](#), [genome8.fa](#), [genome9.fa](#), and [genome10.fa](#) contain the 10 genomes.
- The files [true-ann1.fa](#), [true-ann2.fa](#), [true-ann3.fa](#), [true-ann4.fa](#), and [true-ann5.fa](#) contain the annotation of the first 5 genomes. The annotation is given in FASTA format as a sequence over the symbols `N`, `C`, and `R`. The symbol `N`, `C`, or `R` at position i in `true-ann_k.fa` gives the "state" of the nucleotide at position i in `genome_k.fa`. `N` means that the nucleotide is non-coding. `C` means that the nucleotide is coding and part of a gene in the direction from left to right. `R` means that the nucleotide is coding and part of gene in the reverse direction from right to left. This is exactly as in the exercises in week 47.
- All the above files are available in [data-handin3.zip](#)

A simple analysis of the annotations and the corresponding genomes, it is clear that there are several start-codons possible (and not only `atg` as modelled in class).

Data

```

NC_002737.1 Streptococcus pyogenes M1 GAS, complete genome.
>genome1
TTGTTGATATTCTGTTTTTCTTTTTAGTTTTCCACATGAAAAATAGTTGAAAAAATA
GCGGTGTCCTCTTAAATGGCTTTTCCACAGGTTGTGGAGAACCCAAATTAACAGTGTTA
ATTTATTTTCCACAGGTTGTGGAAAACTAACTATTATCCATCGTCTCTGGAAAACTAG
AATAGTTTTATGGTAGAATGTTCTAGAATATCCACAAGAACGAACTAGTATGACTGAA
AATAAGACAAATTTTGGACAGGCTTTGGAATTAGCTCGAAGTGAATTTAAACACGGCA
ACTTATGAATTTTTTGTTCATGATGCCGCTCTATTAAGGTCGATAAGCATATTGCAACT
ATTTACTTATAGATCAAAATGAAAGAGCTCTTTTGGGAAAAAATCTTAAAGATGTTATTC
CTCTGGTTTTGAAGTTTAAACGCTCAAAATTTCTGTTGACTATGTTTCTGGAAGAAGAC
CTAATGATTTAGACAAATCAGACAAAATCAACCAAACTAAGTTCGAGCAAGCTTAAAT
TCITTTGCTACTGTTACTTCAGATTTAAACTCGAAATATAGTTTGA AAAACTTTATCAA
GGAGATGAAAACTGTTGGGCTGTTGCTGCTTCAATAGCAGTAGTAACTCTTGGAAGT
ACCTATAATCCTTTGTTTTATTGGGGTGGCCCTGGGCTTGAAAAACCCATTATTAATAT
GCTATTGGTAATCTGTACTATTAGAAAACCAAATGCTCGAATTAAATATATCACAGCT
GAAAACTTTATTAATGAGTTTGTATTCATATTCGCTTGATACCATGGATGAATTGAAA
GAAAAATTTCTGAATTTAGATTTACTCTTATTGATGATATCCAATCTTAGCTAAAAAA
ACCGTCTCTGGACACAGAAGAGTCTTTAAATCTTTTAAAGCATTCAATATTAAC
AAACAAATGTCTACAACGACCGTACACAGATCATCTCAATGATTAGAAGATCGA
TTAGTAGTCTGTTTTAAATGGGGATTAAACGTCATATCACACCTCTGATTTTGAAACA
CGAGTGCTATTTGACAAATAAAATCAAGAATAAATCTATTTTCTTCTCAAGTACC
ATTGAGTATTTGGCTGTCATTTGATTTCAATGTCAGAGATTTAGAAGGTGCTTAAAA
GATATTAGTCTGGTGTCTAAATTTCAACAAATGACACGATTACTGTTGACATTGCTGCC
GAAGCTATTCCGCCAGAAAGCAAGTAGGACCTAAAAAGCAGGATTTCCCATCGAAGTA
TTCAACGGCCCAAGTTGAAAAATTTACGGTGTACCGTCAAGAAAAATTAAGTCAATAA
CGAACACAAAAATTTGTTTTAGCAAGACAAGTAGCTATGTTTTAGCACGTGAAATGACA
GATAACAGTCTTCTAAATTTGAAAAAGAAATTTGGTGCGCAGAGACCATTCACAGTACT
CATGCCATAATAAAAAACAAAAATGATCAGCCAGGACGAAGCCCTAGGATCGAAAT
GAAACATAAAAAACAAAAATTAATAACATGCTGGAAAGAAGATCTTTTTATGAATAAGT
ATCCACAAGTTGTGAACATCCATTTAGTCTTGGAATCTCTCGTTTATTTAGAGTTATCCA
CTATTATACACAAGACCTACTACTACTTACTTATTTATTAATCTTTAAATAAAGGAGTTCT
CATGATTCAAATTTCAATTAATGCACATATTTATTCATGCTTTAAATACAGATTAACG
TGCTATTAGCACTAAAAATGCCATTTCTATTTCTCATCAATAAAATTTGAAGTCACCTC
TACAGGAGTAACTTTAACAGGGTCTTAACGGTCAAATATCAATTGAAAACACTATTCCTGT
AAGTAAATGAAAATGCTGGTTTGCTAATTAACGTTCCAGAGCTATTTTATAGAAGCTAG
TTTTTTTATTAATATTTTCAAGTTTGCAGATATTAGTAAATGTTAAAGAAATTTGA
ACAACACCAAGTTGTTTTAACCAGTGGTAAATCAGAGATTACCTTAAAAAGGAAAAGATGT
TGACCAGTATCTCGTCTACAAGAAGTATCAACAGAAATCCTTTGATTTAAAAACCAAA
ATTTATGAAGTCTATTATTGCTGAAACAGCTTTTGACGCCAGTTTACAAGAAAGTCGTG
TATTTTAAACAGGCTCATATGTGATTAAAGTAAATCAAAAGTTTAAAGCAGTAGCGAC
TGACTCTCATCGTATGAGCCAACGTTAATCACTTTGGACAATACTTCAGCAGATTTTGA
TGTAGTTATTCCAAGTAAATCTTTGAGAGAAATTTCCAGCAGATTTACAGATGATATTGA
GACCGTTGAGGTATTTTTCTACCAAGGCAAAATCTGTTGCAAGAGTGAACACATTTCTTT
TTATACACGCCCTTTAGAAGAAAAATTCGCGATACAGACCGTTTATTAATGACAGAAAT
TGAGACGGAGGTTGTTTTCAATACCCAATCCCTTGCCACGCTAGTGAACGTGCGCTCTTT
GATTTCTAATGCTACTCAAAATGGTATGTTAAAGCTTGAGATTAAGTCAAAATCATATTT
AGCTCTGTTAACTCACTGAGGTTGTTGAAGTAAACGAGGATTACTGATGTTTGTATGCA
GTCTGGTAGTGATTTAACTATCAGCTTCAATCCAACCTACCTTATTGAGTCTTTAAAGC
-:--- genome1.fa Top of 1.9M (1,0) (Fundamental)

```

genome1.fa

[illegible]

true-ann1.fa

Analysis of data

```
Length of genome1: 1852441 (1852441)
Length of genome2: 2211485 (2211485)
Length of genome3: 2499279 (2499279)
Length of genome4: 1796846 (1796846)
Length of genome5: 2685015 (2685015)
Length of genome6: 2127839 (2127839)
Length of genome7: 2742531 (2742531)
Length of genome8: 2046115 (2046115)
Length of genome9: 2388435 (2388435)
Length of genome10: 1570485 (1570485)
Length of genome11: 2096309 (2096309)
```

We observe

3 typical start-codons

3 stop-codons

We may ignore rare start and stop codons, i.e. a gene that starts (or ends) with a ignored start (or stop) codon does not correspond to a path in your model.

Start-codon in normal genes:

```
ATG [8423, 'NCCC']
ATC [3, 'NCCC']
ATA [1, 'RCCC']
GTG [713, 'NCCC']
ATT [3, 'NCCC']
CTG [2, 'NCCC']
GTT [1, 'NCCC']
CTC [1, 'NCCC']
TTA [1, 'NCCC']
TTG [1020, 'NCCC']
```

Stop-codon in normal genes:

```
TAG [1949, 'CCCN']
TGA [1531, 'CCCN']
TAA [6686, 'CCCN']
```

Reversed stop-codon in reversed genes:

```
TTA (reverse-complement: TAA) [6596, 'NRRR']
CTA (reverse-complement: TAG) [2014, 'NRRR']
TCA (reverse-complement: TGA) [1148, 'NRRR']
```

Reversed start-codon in reversed genes:

```
TAT (reverse-complement: ATA) [2, 'RRRN']
ATG (reverse-complement: CAT) [1, 'RRRN']
GAT (reverse-complement: ATC) [1, 'RRRN']
CAT (reverse-complement: ATG) [8077, 'RRRN']
AAT (reverse-complement: ATT) [4, 'RRRN']
TAC (reverse-complement: GTA) [1, 'RRRN']
CAC (reverse-complement: GTG) [715, 'RRRN']
CAA (reverse-complement: TTG) [953, 'RRRN']
CAG (reverse-complement: CTG) [4, 'RRRN']
```

Data

You are given a data set containing 10 [Staphylococcus](#) genomes, each containing several genes (i.e. substring) obeying the "gene syntax" explained in class. The genomes are between 1.8 million and 2.8 million nucleotides long. For 5 of the genomes, you are also given the gene structure, i.e. the location of the genes. For the remaining 5 genomes, you only know that they contain genes according to the "gene syntax". You have already explored two of the genomes in the practical exercises in week 47.

The genomes and their annotations are given in [FASTA format](#). There are several Python libraries for reading sequences in FASTA format (use Google to find one), and you are welcome to use the simple reader function `read_fasta_file` below that you also used in the exercises in week 47.

The data is:

- The files [genome1.fa](#), [genome2.fa](#), [genome3.fa](#), [genome4.fa](#), [genome5.fa](#), [genome6.fa](#), [genome7.fa](#), [genome8.fa](#), [genome9.fa](#), and [genome10.fa](#) contain the 10 genomes.
- The files [true-ann1.fa](#), [true-ann2.fa](#), [true-ann3.fa](#), [true-ann4.fa](#), and [true-ann5.fa](#) contain the annotation of the first 5 genomes. The annotation is given in FASTA format as a sequence over the symbols `N`, `C`, and `R`. The symbol `N`, `C`, or `R` at position i in `true-ann_k.fa` gives the "state" of the nucleotide at position i in `genome_k.fa`. `N` means that the nucleotide is non-coding. `C` means that the nucleotide is coding and part of a gene in the direction from left to right. `R` means that the nucleotide is coding and part of gene in the reverse direction from right to left. This is exactly as in the exercises in week 47.
- All the above files are available in [data-handin3.zip](#)

A simple analysis of the annotations and the corresponding genomes, it is clear that `fx` several start-codons are possible (and not only `atg` as modelled in class).

Problem

Your task is to predict the gene structure of the 5 unannotated genomes, i.e. [genome6.fa](#), [genome7.fa](#), [genome8.fa](#), [genome9.fa](#), and [genome10.fa](#), in the best possible manner using an HMM based approach.

As explained in class, predicting gene structure using an HMM involves:

- Deciding on an initial model structure, i.e. the number of hidden states and which transitions and emission should have a fixed probability (e.g. 0 for "not possible", or 1 for "always the case"). You might get inspiration from, or even use, one of the model structures discussed in class. However, be aware that the models presented in class model only one start-codon atg. This is not the case in the presented data, where multiple start-codons are possible (as also mentioned in the lecture/slides about this project).
- Tune model parameters by training, i.e. set the non-fixed emission and transition probabilities. This can be done by several methods as explained in class. You can use (1) Training-by-Counting using the 5 genomes with know gene structure because you know the underlying hidden state for each observation (i.e. symbol) in these sequences, or (2) Viterbi- or EM-training using all the 10 genomes, including the 5 genomes with unknown gene structure. You can of course also combine the two approaches. You can evaluate the performance of your gene finder by computing the approximate correlation (AC) between your predictions and the true structure using this small python program [compare_anns.py](#) (see [this paper](#) for details).

In this project, Training-by-Counting is mandatory, while Viterbi- or EM-training is optional. If you implement EM-training you should of course be aware of the numerical problems of the forward- and backward-algorithms unless you use scaling or similar techniques.

- Evaluate the performance of your gene predictor. In this project, you must do a 5-fold cross validation on the 5 genomes with known gene structure, i.e.:

You consider genome 1 to 5 in five rounds. In a each you train your model using Training-by-Counting on the remaining 4 genomes, predict the gene structure of the genome you consider, and compute the approximate correlation coefficient (AC) between your predicted gene structure and the true gene structure using the python program [compare_anns.py](#).

- Use your best model to predict the gene structure for the 5 unannotated genomes using the Viterbi algorithm with subsequent backtracking. I.e. for each unannotated genome you must find a most likely sequence of states in your model generating it, and convert this sequence of states into a FASTA file giving the annotation of each nucleotide as **N** , **C** , or **R** . You should make files, `pred-ann6.fa` , ..., `pred-ann10.fa` , giving the annotation for the 5 unannotated genomes.

You can use the www-service [GeneFinder Verifier](#) to compare your predictions on Genome 6-10 against their true structures.

Problem - Selecting an initial model

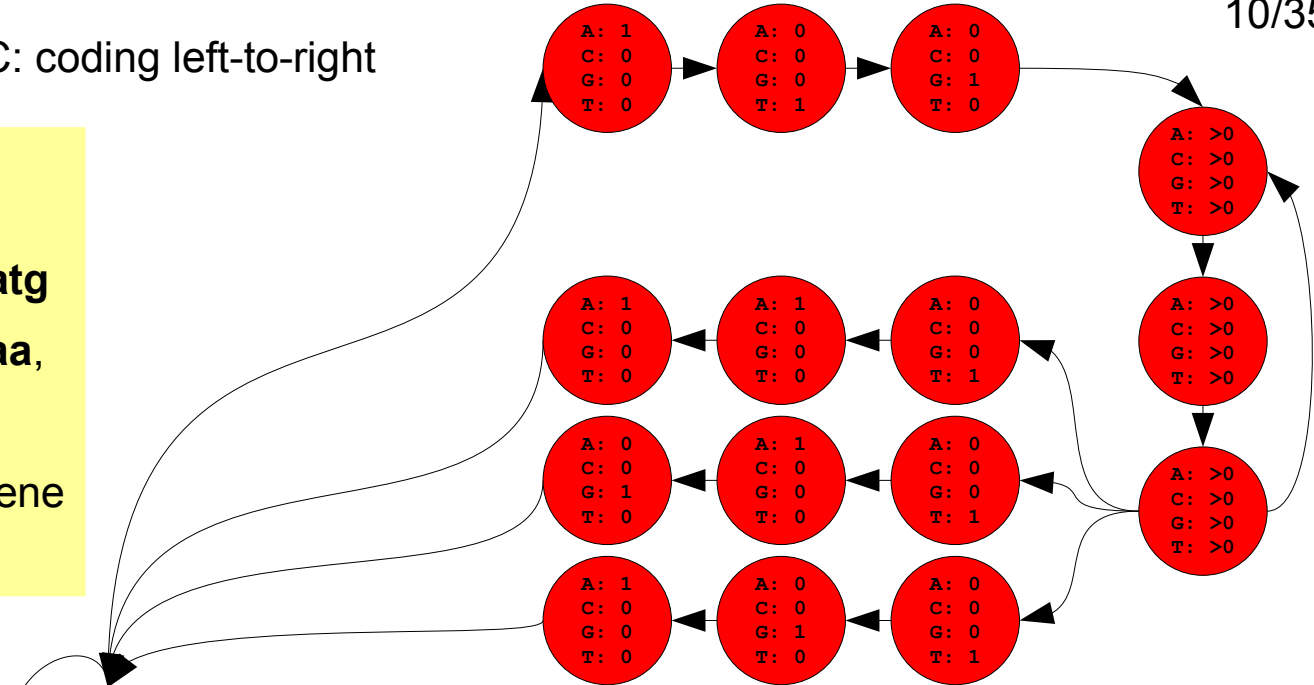
- A gene is a substring of the DNA sequence of A,C,G,T's
- A gene starts with a start-codon **atg**
- A gene ends with a stop-codon **taa**, **tag** or **tga**
- The number of nucleotides in a gene is a multiplum of 3

Even more biology

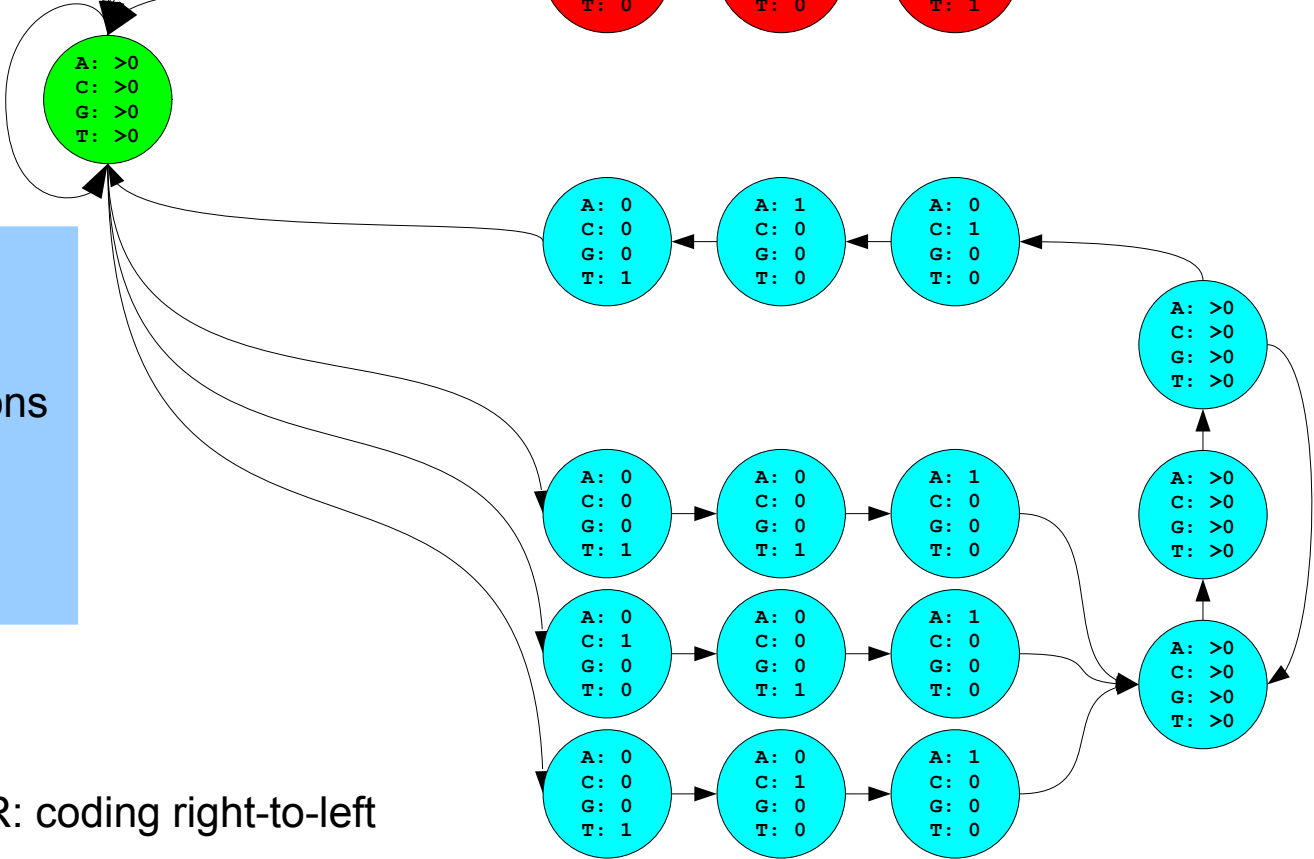
There can be genes in both directions

$$\pi_N = 1$$
$$\pi_C = 0$$

C: coding left-to-right

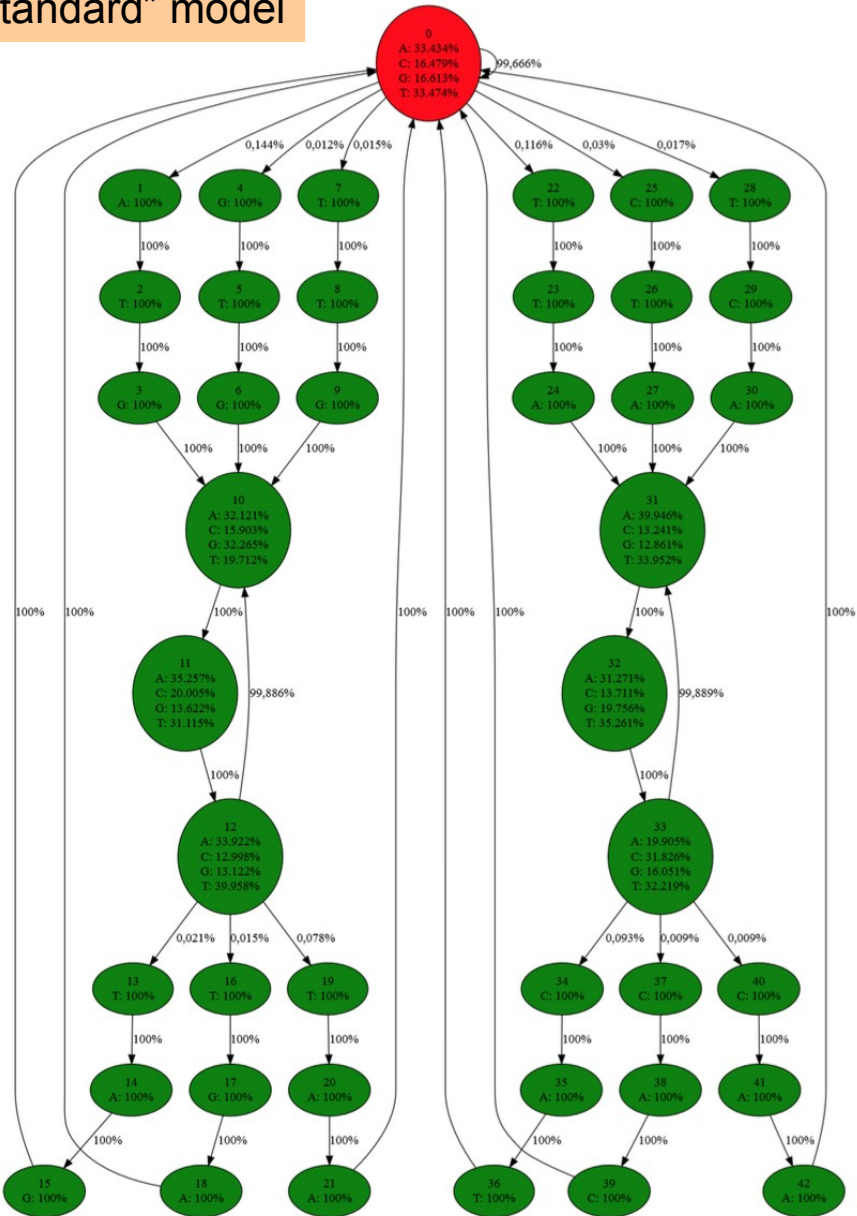


N: Non-coding

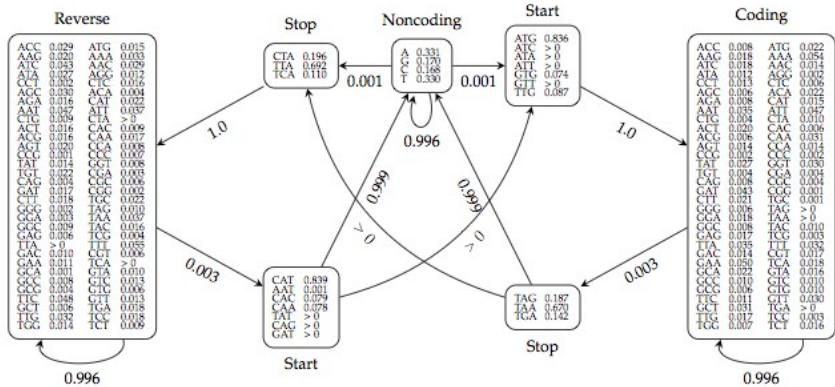


R: coding right-to-left

“Standard” model



“Codon” model



Problem - 5-fold cross validation

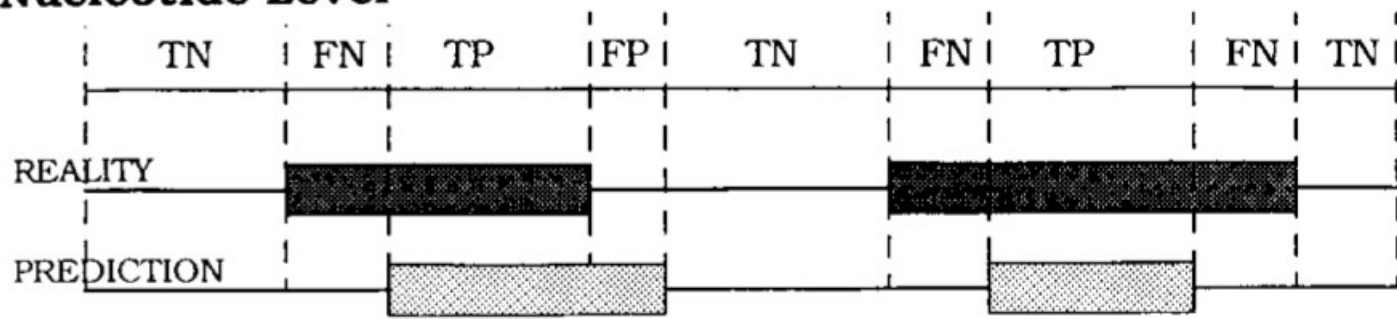
You consider genome 1 to 5 in five rounds. In a each you train your model using Training-by-Counting on the remaining 4 genomes, predict the gene structure of the genome you consider, and compute the approximate correlation coefficient (AC) between your predicted gene structure and the true gene structure using the python program `compare_anns.py`.

	Round 1	Round 2	Round 3	Round 4	Round 5
Genome 1	Validate	Train	Train	Train	Train
Genome 2	Train	Validate	Train	Train	Train
Genome 3	Train	Train	Validate	Train	Train
Genome 4	Train	Train	Train	Validate	Train
Genome 5	Train	Train	Train	Train	Validate

	AC		
	Only Cs	Only Rs	Both
Genome 1	0.5983	0.6470	0.4347
Genome 2	0.6305	0.6529	0.4510
Genome 3	0.6552	0.6510	0.4805
Genome 4	0.6240	0.6002	0.4079
Genome 5	0.6550	0.6027	0.4302

Evaluating performance

Nucleotide Level



		REALITY		
		coding	no coding	
PREDICTION	coding	TP	FP	TP+FP
	no coding	FN	TN	FN+TN
		TP+FN	FP+TN	

$$S_n = \frac{TP}{TP + FN}$$

Sensitivity

$$S_p = \frac{TN}{TN + FP}$$

Specificity

$$CC = \frac{(TP \times TN) - (FN \times FP)}{\sqrt{(TP + FN) \times (TN + FP) \times (TP + FP) \times (TN + FN)}}$$

Correlation Coefficient

$$ACP = \frac{1}{4} \left[\frac{TP}{TP + FN} + \frac{TP}{TP + FP} + \frac{TN}{TN + FP} + \frac{TN}{TN + FN} \right]$$

$$AC = (ACP - 0.5) \times 2$$

Approximate Correlation

compare_anns.py

```
$ python compare_anns.py true-ann6.fa pred-ann6.fa
```

```
Cs    (tp=757332, fp=164766, tn=305197, fn=57217): Sn = 0.9298, Sp = 0.8213, AC = 0.6213
```

```
Rs    (tp=715865, fp=127462, tn=304830, fn=57584): Sn = 0.9255, Sp = 0.8489, AC = 0.6603
```

```
Both  (tp=1473197, fp=292228, tn=247613, fn=114801): Sn = 0.9277, Sp = 0.8345, AC = 0.4520
```

compare_anns.py

```
$ python compare_anns.py true-ann6.fa pred-ann6.fa
```

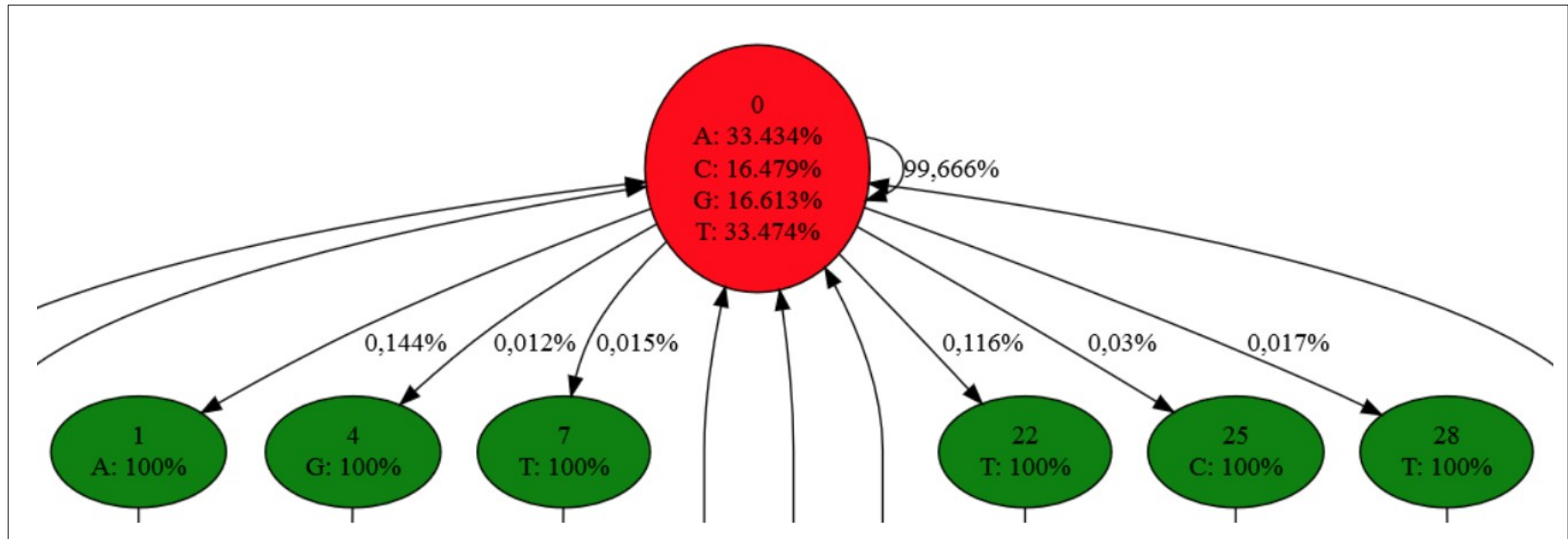
```
Cs    (tp=757332, fp=164766, tn=305197, fn=57217): Sn = 0.9298, Sp = 0.8213, AC = 0.6213
```

```
Rs    (tp=715865, fp=127462, tn=304830, fn=57584): Sn = 0.9255, Sp = 0.8489, AC = 0.6603
```

```
Both (tp=1473197, fp=292228, tn=247613, fn=114801): Sn = 0.9277, Sp = 0.8345, AC = 0.4520
```

Problem – Training the model

Training by counting



In theory, we are given $(\mathbf{X}, \mathbf{Z})$ pairs, but we are given $(\mathbf{X}, \text{"Z"})$ pairs, where the NCR-annotations have to be translated to $\mathbf{Z}$ s.

Also, we may ignore rare start and stop codons, i.e. a gene that starts (or ends) with a ignored start (or stop) codon does not correspond to a path in your model.

Training by counting – Typical solution

Example: How to set the transition probabilities:

$N \rightarrow N$	$N \rightarrow CCC$, where CCC emits ATG	$N \rightarrow RRR$, where RRR emits TTA
	$N \rightarrow CCC$, where CCC emits GTG	$N \rightarrow RRR$, where RRR emits CTA
	$N \rightarrow CCC$, where CCC emits TTG	$N \rightarrow RRR$, where RRR emits TCA

We count:

$\#(N \rightarrow N)$ = no. of occurrences of “NN” our annotations.

$\#(N \rightarrow CCC, \text{ where CCC emits XYZ})$ = no. of occurrence “NCCC” in our annotations, where CCC is an annotation of XYZ (in our training data).

$\#(N \rightarrow RRR, \text{ where RRR emits XYZ})$ = no. of occurrence “NRRR” in our annotations where, RRR is an annotation of XYZ (in our training data).

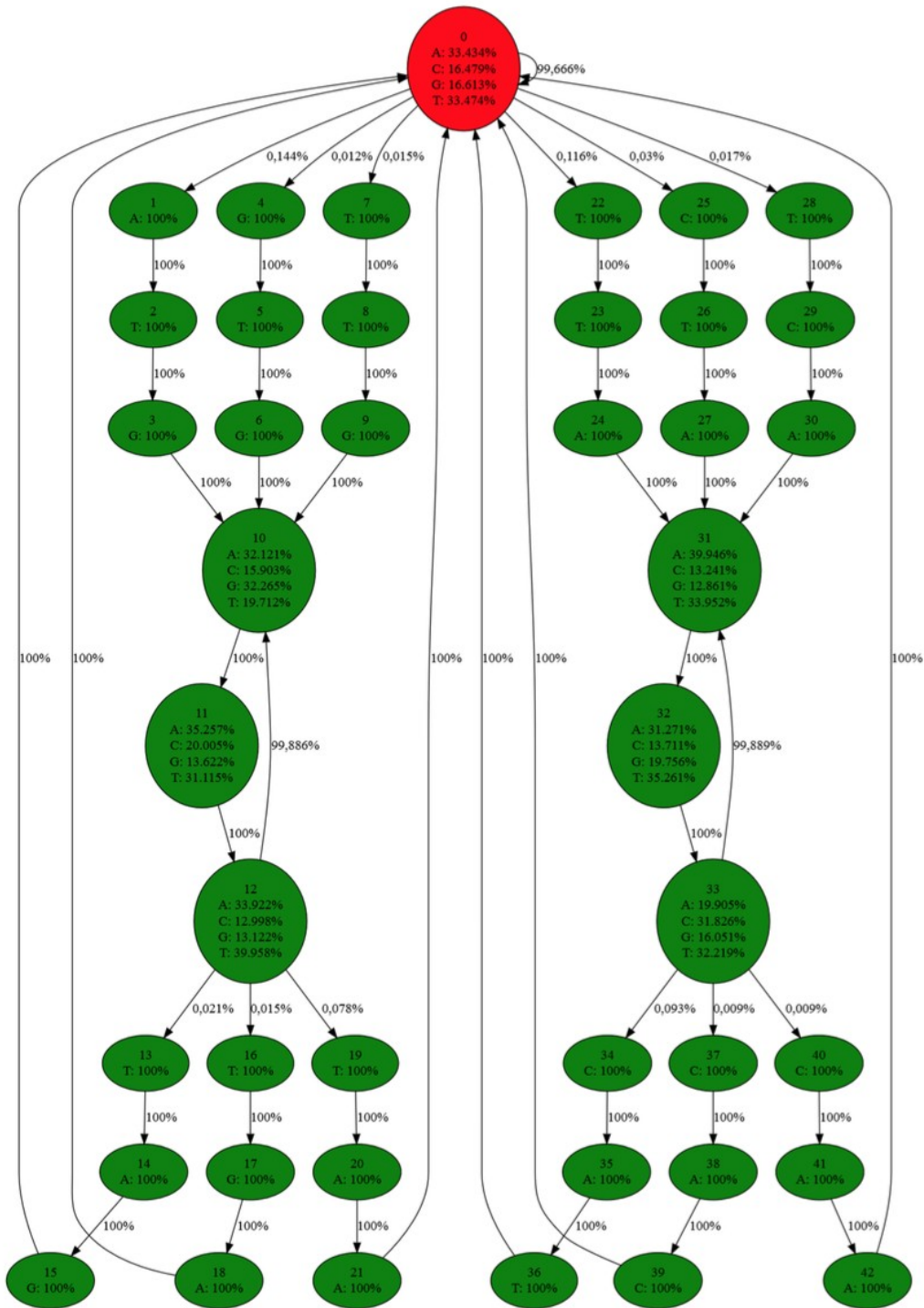
We compute:

$\text{Total} = \#(N \rightarrow N) + \#(N \rightarrow CCC \text{ where CCC emits XYZ}) + \#(N \rightarrow RRR, \text{ where RRR emits XYZ})$

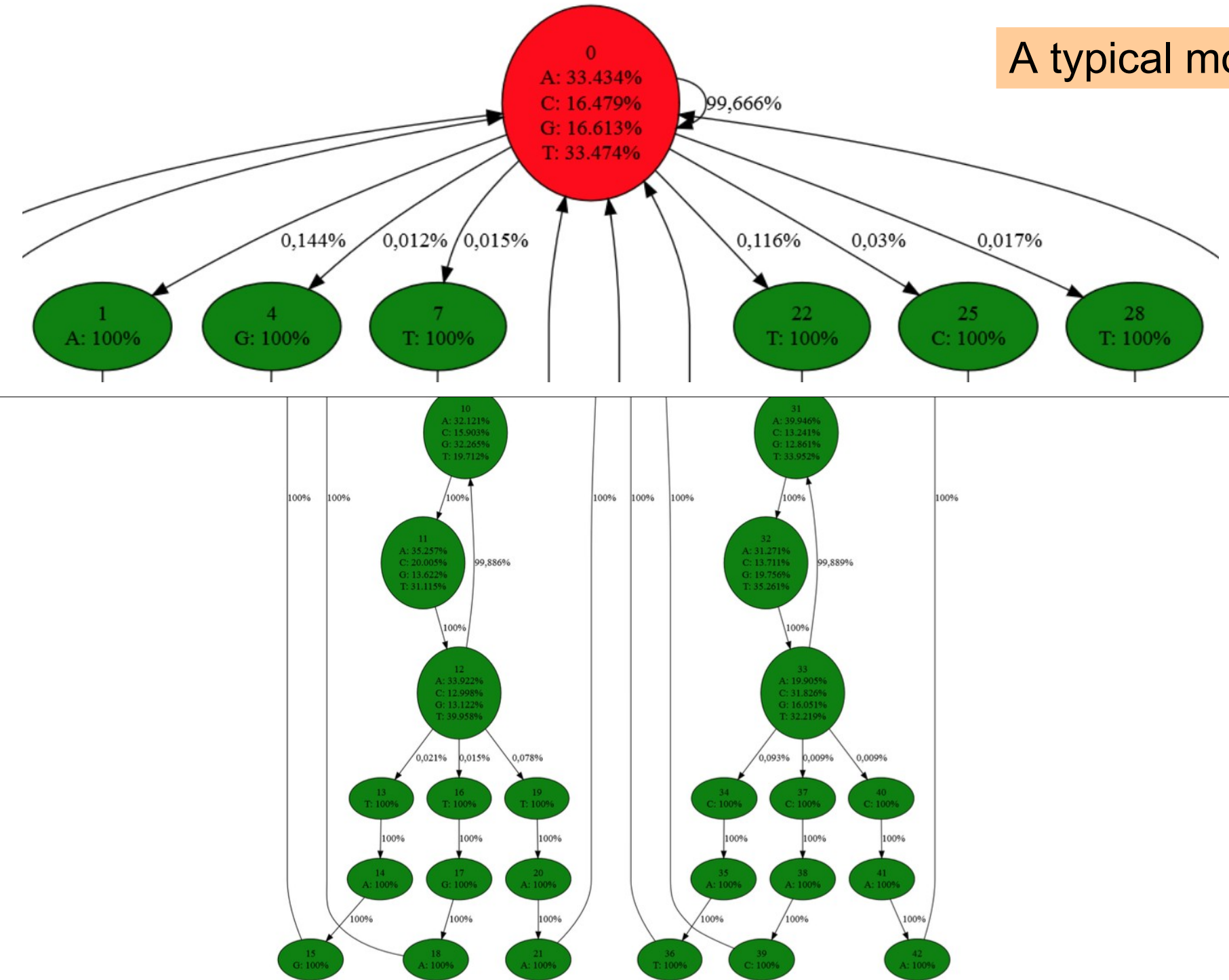
We set:

$P(N \rightarrow X) = \#(N \rightarrow X) / \text{Total}$ for each of the 7 transitions

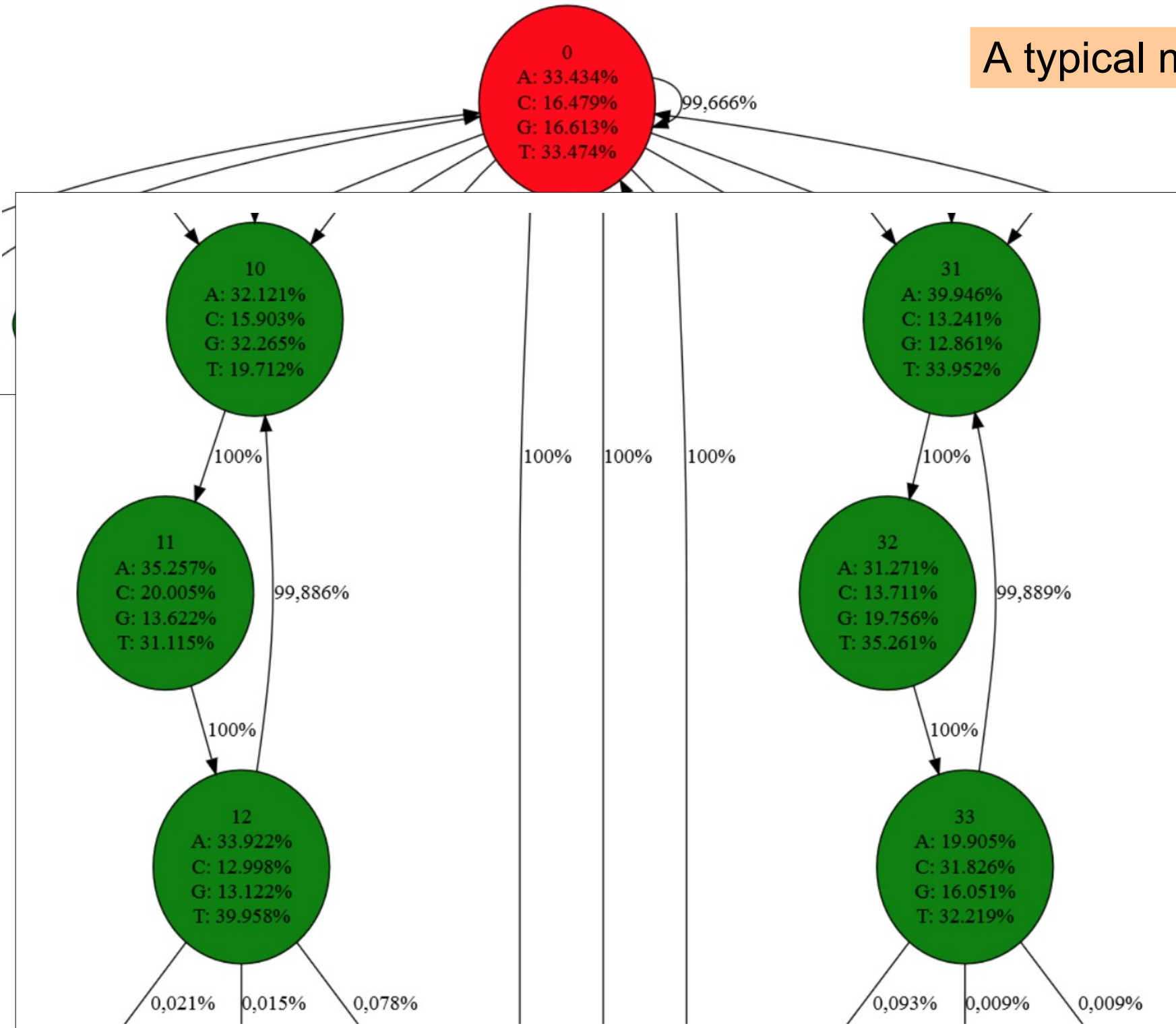
A typical model



A typical model



A typical model



```

*scratch* 1 genome6.fa pred-ann6.fa 3
NC_007432.1 Streptococcus agalactiae A909, complete genome.
>genome6
ATTAACATTTGTTGATTTTATCTTCAACATCTGTGAAAACTTTATTTTTTATGTTACA
ATATAACAATAATTATCCACAAGACAATAAGGAAGAAGCTATGACGGAAAAACGAACACT
ATTTTGGAAATAGAGTACTAGAGCTATCTCGTTCTCAATAGCACAGCAGCTATGAATT
TTTTGTTCTAGAGGCTAGACTCTCAAAATTGAACATCAAACTGCAGTTATTACTTTAGA
TAACATTTGAAATGAAAAAGCTATTCTGGGAACAAAATTTGGGCGCTGTTATCTCAACAGC
TGGTTTTGAAATTTTCAATGCTGAAATTACAGCTAACTATGTCTCAACGATTTACATTT
ACAAGAACTAGTTTTTCTAACTACCAGCAATCTAGCAATGAAGTAAATATCTTACCAAT
TAGAAAAATCGACTCTAACCTTAAAGAGAAATACACTTTTGTCAATTTTGTTCAGGAGA
TAGAAAATAGATGGGCTGTTTCAAGCATCAATTTGCTGTAGCTGATAGTCTGGCAGCACTTA
TAATCTCTCTATTTATCTGGGGGGGACCTGGTCTAGGAAAGACCGCATCTACTAAATGCTAT
TGGAAATCAAGTTTTAAGAGATAATCCAAACGCCAGGGTTTTATACATCACTGCTGAGAA
TTTTTATTAATGAAATTTGTCAGTCAATTTCTGTTAGATTCTGATGGAAGATTAAGAAAGAA
GTTTCGCAACTTTGGACTTACTCCTGATTGATGATATTCAGTCGCTTGCTAAGAAAAACCTT
AGGGGGGACCCCAAGAGGAGTTCTCTCAATCTTTCAATGCTTTACATCAAAACGATAAACA
AATCGCTATTGACCAAGTACCGCAATCAAAATCAATTAAGATCTAGAAGAACGCTGTAGT
CAGCGCTTTAGTTGGGGACTCCCGATTAATATCACCAACCTGATTTTGAACACAGGAT
TGCTATTTTAAACCAATAAAATCAAGAATATCCTTATGATTTTCTCAAGATACCATTTGA
ATACCTTAGCAGGAGAATTTGATTCCAACGTACGTGAATTAGAAGGAGCCTTGAAAAATAT
TAGTCTAGTTTGTGACTTTAAGCATGCTAAAACTATTACAGTAGATATAGCTCGAGAAGT
TCTCAGACGACGTAAAAATGATGGTCTTATTTGCTTACTGTCATCTCTATAGAAGAAATTC
AATACAAGTTGGTAAATTTCTATGGCGTAACGTGTAAGAGAGATAAAAGCAACTAAAGAAC
ACAAGATATTTGCTTTCGAAGCAGGTAGGCACTGTACTAGTCTGATGAGTACAGATAAA
CAGTCTCCCAAAAAATAGTTAAGAAATTTGGGGACGACGATCACTCAACTGTTTCTCCAGC
TTATAATAAAAAATAAAATATGGTTGCTCAAGATGACAACCTTACGAATTGAGATAGAAAC
TATCAAAAAATAAATCAGATAGGCTTTTGGGATAGATACAGAAAAATACGAGACTTTTCT
CAGAGTTGTGAACAAAGTATTTTCTTATGAATACTAAAGTTATAAATAGCTATTCAACA
CTTTACAGCGACTACTATTACTACTAAATTTACTATAATATATAAAGTATATACTATGA
TTCATTTTTCAATAAATAAAAAATTTCTTCTGATGCTCTAACGGTAACCTAACAGAGCTA
TTAGTCAAAAAATGCGAATTTCCAATCTTTCAACTGTTAAATAGAAGTAACTAGAGAGT
CTATCATTTTCAACGGGTCAAGTGCAACAAATTTCAATGAAAACTACTTTCTGCTTTCA
ATGAAAATGCAGGTTTTACTAGTAACCAACTCCAGGCTCTATTTTATTAGAAGCTGGTTTCT
TTATTAATATATTTTCAAGTTTACCAGATGTAACCTTAGAATTTACAGAAATTGAAACAAC
ATCAAAATGTTCTTACTAGTTGGAAAAATCAGAGATTACTTTGAAAGGTAAGGATGTCGATC
AATACCTCGTCTGCAGGAATGACACAGACATCTCAATTAAGATAGAACTAACTAAGCT
TAAATCAATTATTAATGAACTGCTTTTGTGCTAGCCAACAAGAAAGCCGCTCAACTCT
TAACAGGTTGTCATTTGGTTATTCAGTCAAAATATAATCTTTAAGGCTGTTGCGACAGATT
CACCCGATGAGTCAACGCATCTTCCAATAGAGTAAATCGGCTCAATATAATTTGATTTGG
TTGTTCCAAGTAAATCCCTTCGAGAATTTTGGCTGTTTACAGATGATATTGAAACTG
TAGAGGTTTTCTTCTCAGATAGTCAAAATGCTATTTAGAAGTAAAAATATCAGCTTCTATA
CAGGTTTGTGTAAGGAACTACCTCTGATACGTGCGCTCCTAACTAATCTTTGATTTGAAA
CCGAAATTTATCTTTAATACAAAGTTTCTACGCATGCTATGACAGCTGCTATTTAATTT
CGAATGCAACTCAGAACGGTACTGTTGTTTGAAGAAATCAAAATGAAACGCTCAGCTC
ATGTAAACTCTCCAGAAGTTGGTAAAGTTAATGAGGAATGGATCTGTTAGCTTAAAG
GTGATAGTTAAATAGTTTAAATCAACTACTTAACTGATCTTTAAAGACAGTAA
AAAGCGAAGTTTACGATTGCGATTATTTCTCAGTACGCTCCATTCACTTTGACACTGT
GTGAAGATACTGAAGATTTCATACAATTAATAACTCCTGTTCTGACTAACTAAATAAGAA
- - - genome6.fa Top of 2.2M (1,0) (Fundamental)
Find file: ~/
```

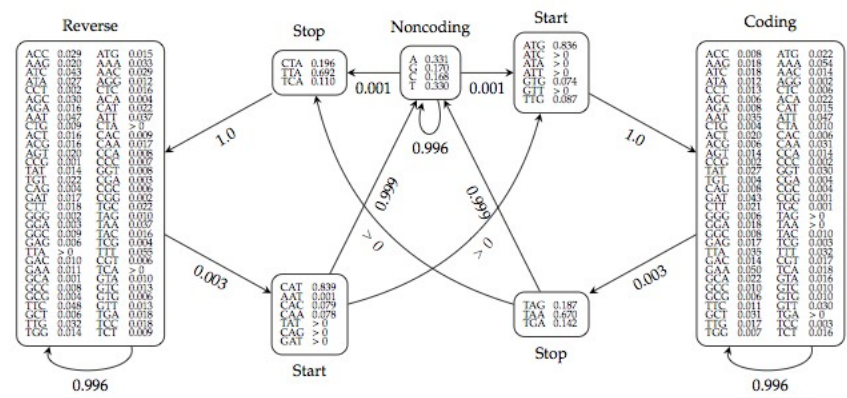
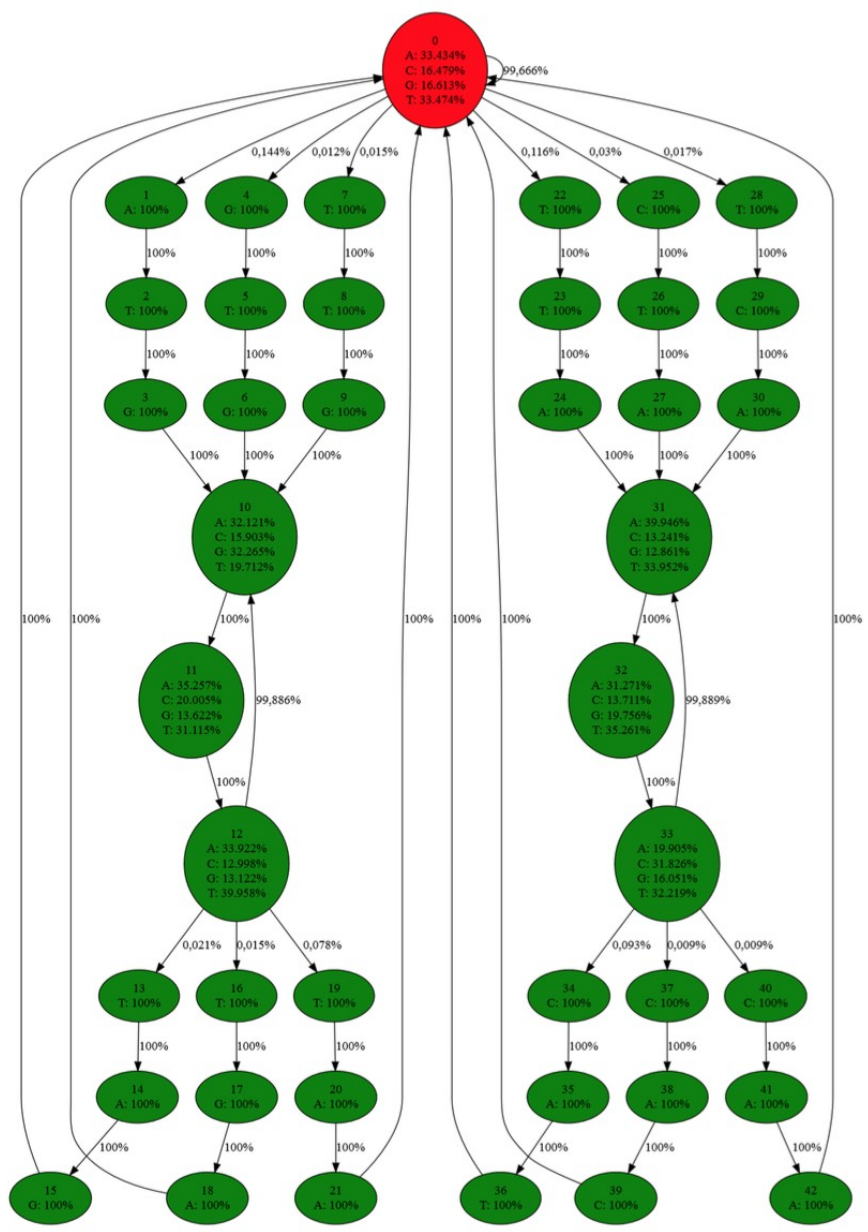
[illegible]

Report

You (i.e. your group) must hand in a short report (in pdf-format) describing your work and results and a zip-archive of the files `pred-ann6.fa` and `pred-ann7.fa` ,..., `pred-ann10.fa` containing your predictions on the 5 genomes with unknown structure. The files *must* be in FASTA format similar to the annotation of the other genomes. Handin the report and zip-archive via Blackboard no later than **Wednesday, December 8, 2021, at 23:59**. Your report should be no more than 3 pages. It must cover:

- The status of the work, i.e. does it work, if not, then why.
- An explanation and illustration of your model structure (i.e. a transition diagram). This should include an explanation of how you model start and stop codons, and an explanation of how you have trained your model. For Training-by-Counting you might comment on how you translate the given annotations of C , N , and R 's into corresponding sequences of hidden states. The transition-diagram must contain all non-0 emission- and transition probabilities for the model that you have used for predicting the gene structure of the 5 genomes with unknown gene structure.
- An explanation of how you have predicted the gene structure for the the 5 genomes with unknown gene structure. You should comment on how you translate a most likely sequence of hidden states as returned by Viterbi decoding into a sequence of C , N , and R 's.
- The result of your 5-fold cross valiation on the 5 genomes with known gene structure. The report must include a table with the AC obtained in each of the five rounds of the 5-fold cross validation.
- The result of comparing your predictions on the 5 genomes with unknown gene structure against their true structures via the www-service [GeneFinder Verifer](#).

Report – Transition diagram



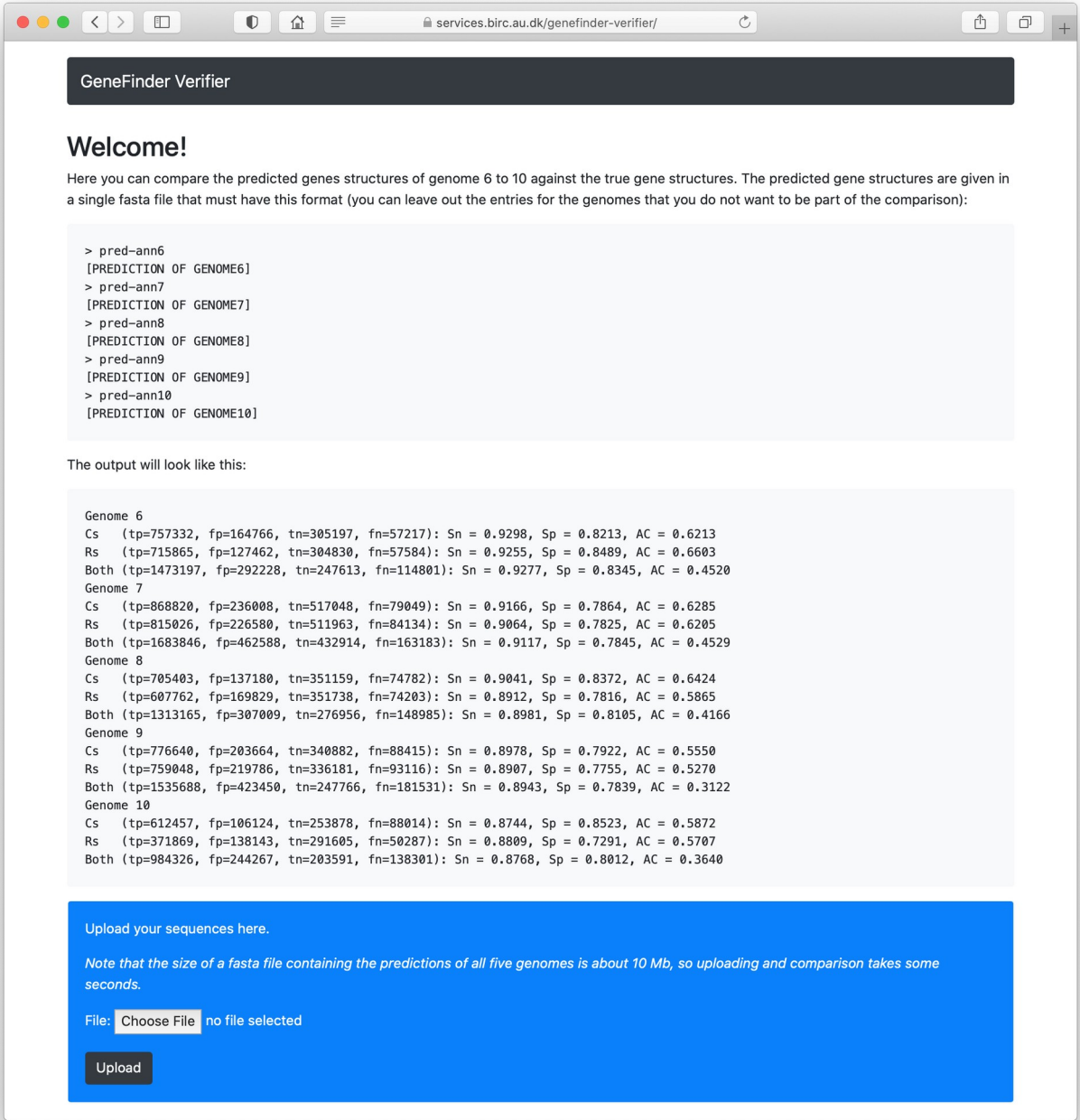
Report - 5-fold cross validation

You consider genome 1 to 5 in five rounds. In a each you train your model using Training-by-Counting on the remaining 4 genomes, predict the gene structure of the genome you consider, and compute the approximate correlation coefficient (AC) between your predicted gene structure and the true gene structure using the python program `compare_anns.py`.

	Round 1	Round 2	Round 3	Round 4	Round 5
Genome 1	Validate	Train	Train	Train	Train
Genome 2	Train	Validate	Train	Train	Train
Genome 3	Train	Train	Validate	Train	Train
Genome 4	Train	Train	Train	Validate	Train
Genome 5	Train	Train	Train	Train	Validate

	AC		
	Only Cs	Only Rs	Both
Genome 1	0.5983	0.6470	0.4347
Genome 2	0.6305	0.6529	0.4510
Genome 3	0.6552	0.6510	0.4805
Genome 4	0.6240	0.6002	0.4079
Genome 5	0.6550	0.6027	0.4302

Report - Performance on genome 6-10



<https://services.birc.au.dk/genefinder-verifier/>

Report - Performance on genome 6-10

Input is a fasta-file containing your predictions. It is about 10 Mb big!

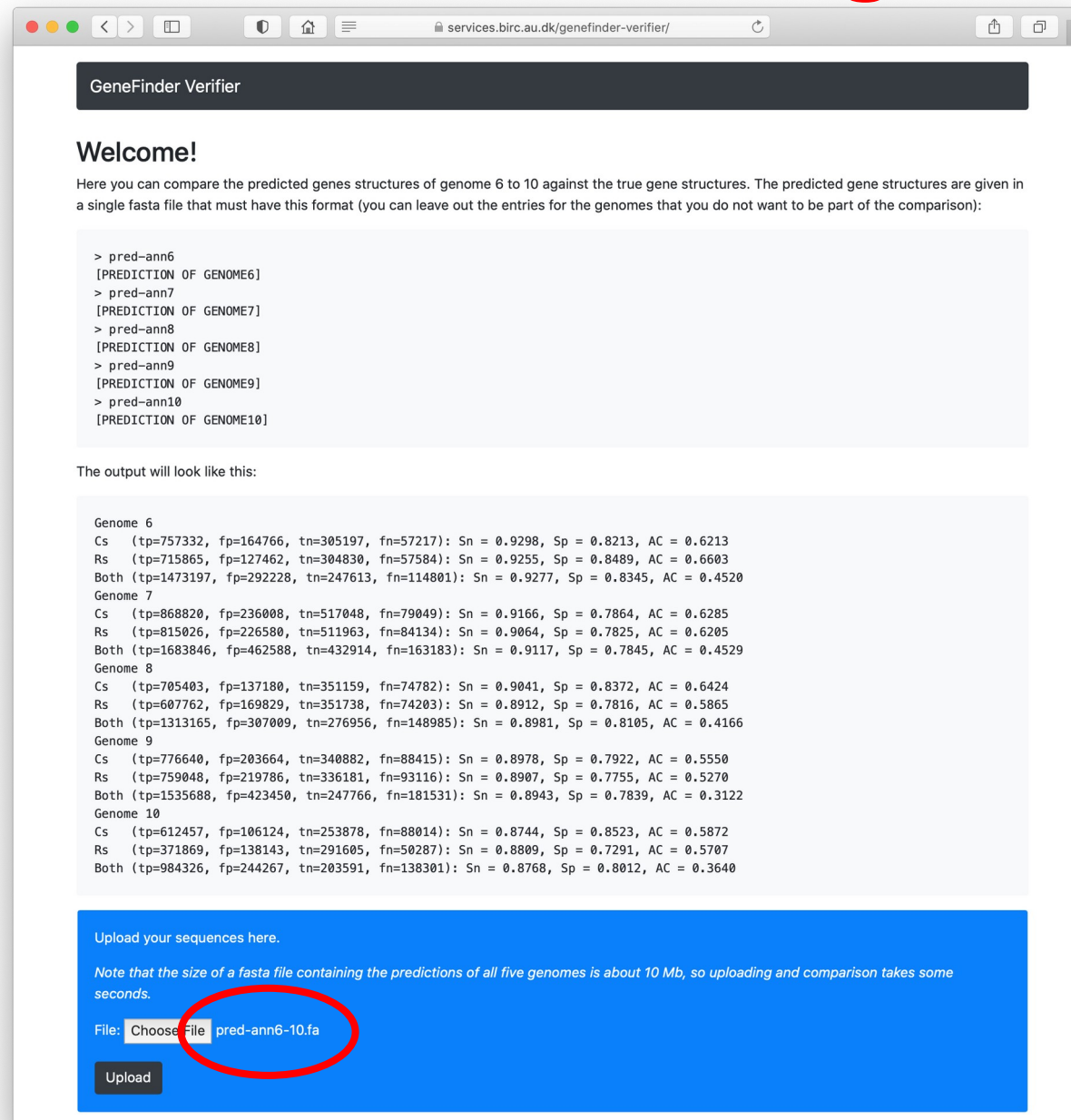
```
> pred-ann6  
[PREDICTION OF GENOME6]  
> pred-ann7  
[PREDICTION OF GENOME7]  
> pred-ann8  
[PREDICTION OF GENOME8]  
> pred-ann9  
[PREDICTION OF GENOME9]  
> pred-ann10  
[PREDICTION OF GENOME10]
```

```
$ cat pred-ann6.fa pred-ann7.fa pred-ann8.fa pred-ann9.fa pred-ann10.fa > pred-ann6-10.fa
```

Beware of the newlines!

pred-annX.fa should end with a newline in order for the above to work

Report - Performance on genome 6-10



GeneFinder Verifier

Welcome!

Here you can compare the predicted genes structures of genome 6 to 10 against the true gene structures. The predicted gene structures are given in a single fasta file that must have this format (you can leave out the entries for the genomes that you do not want to be part of the comparison):

```
> pred-ann6
[PREDICTION OF GENOME6]
> pred-ann7
[PREDICTION OF GENOME7]
> pred-ann8
[PREDICTION OF GENOME8]
> pred-ann9
[PREDICTION OF GENOME9]
> pred-ann10
[PREDICTION OF GENOME10]
```

The output will look like this:

```
Genome 6
Cs (tp=757332, fp=164766, tn=305197, fn=57217): Sn = 0.9298, Sp = 0.8213, AC = 0.6213
Rs (tp=715865, fp=127462, tn=304830, fn=57584): Sn = 0.9255, Sp = 0.8489, AC = 0.6603
Both (tp=1473197, fp=292228, tn=247613, fn=114801): Sn = 0.9277, Sp = 0.8345, AC = 0.4520
Genome 7
Cs (tp=868820, fp=236008, tn=517048, fn=79049): Sn = 0.9166, Sp = 0.7864, AC = 0.6285
Rs (tp=815026, fp=226580, tn=511963, fn=84134): Sn = 0.9064, Sp = 0.7825, AC = 0.6205
Both (tp=1683846, fp=462588, tn=432914, fn=163183): Sn = 0.9117, Sp = 0.7845, AC = 0.4529
Genome 8
Cs (tp=705403, fp=137180, tn=351159, fn=74782): Sn = 0.9041, Sp = 0.8372, AC = 0.6424
Rs (tp=607762, fp=169829, tn=351738, fn=74203): Sn = 0.8912, Sp = 0.7816, AC = 0.5865
Both (tp=1313165, fp=307009, tn=276956, fn=148985): Sn = 0.8981, Sp = 0.8105, AC = 0.4166
Genome 9
Cs (tp=776640, fp=203664, tn=340882, fn=88415): Sn = 0.8978, Sp = 0.7922, AC = 0.5550
Rs (tp=759048, fp=219786, tn=336181, fn=93116): Sn = 0.8907, Sp = 0.7755, AC = 0.5270
Both (tp=1535688, fp=423450, tn=247766, fn=181531): Sn = 0.8943, Sp = 0.7839, AC = 0.3122
Genome 10
Cs (tp=612457, fp=106124, tn=253878, fn=88014): Sn = 0.8744, Sp = 0.8523, AC = 0.5872
Rs (tp=371869, fp=138143, tn=291605, fn=50287): Sn = 0.8809, Sp = 0.7291, AC = 0.5707
Both (tp=984326, fp=244267, tn=203591, fn=138301): Sn = 0.8768, Sp = 0.8012, AC = 0.3640
```

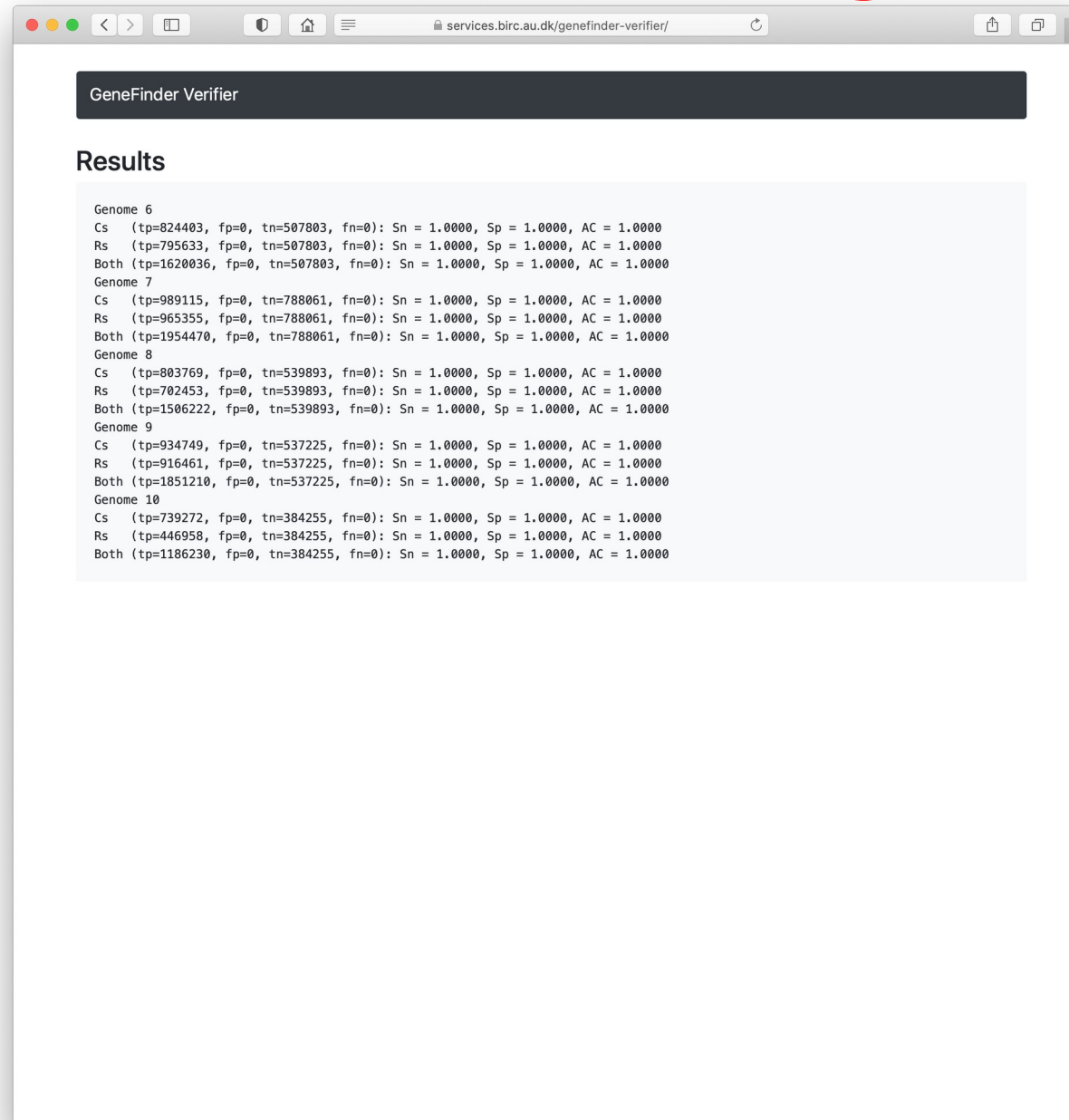
Upload your sequences here.

Note that the size of a fasta file containing the predictions of all five genomes is about 10 Mb, so uploading and comparison takes some seconds.

File: pred-ann6-10.fa

<https://services.birc.au.dk/genefinder-verifier/>

Report - Performance on genome 6-10



<https://services.birc.au.dk/genefinder-verifier/>

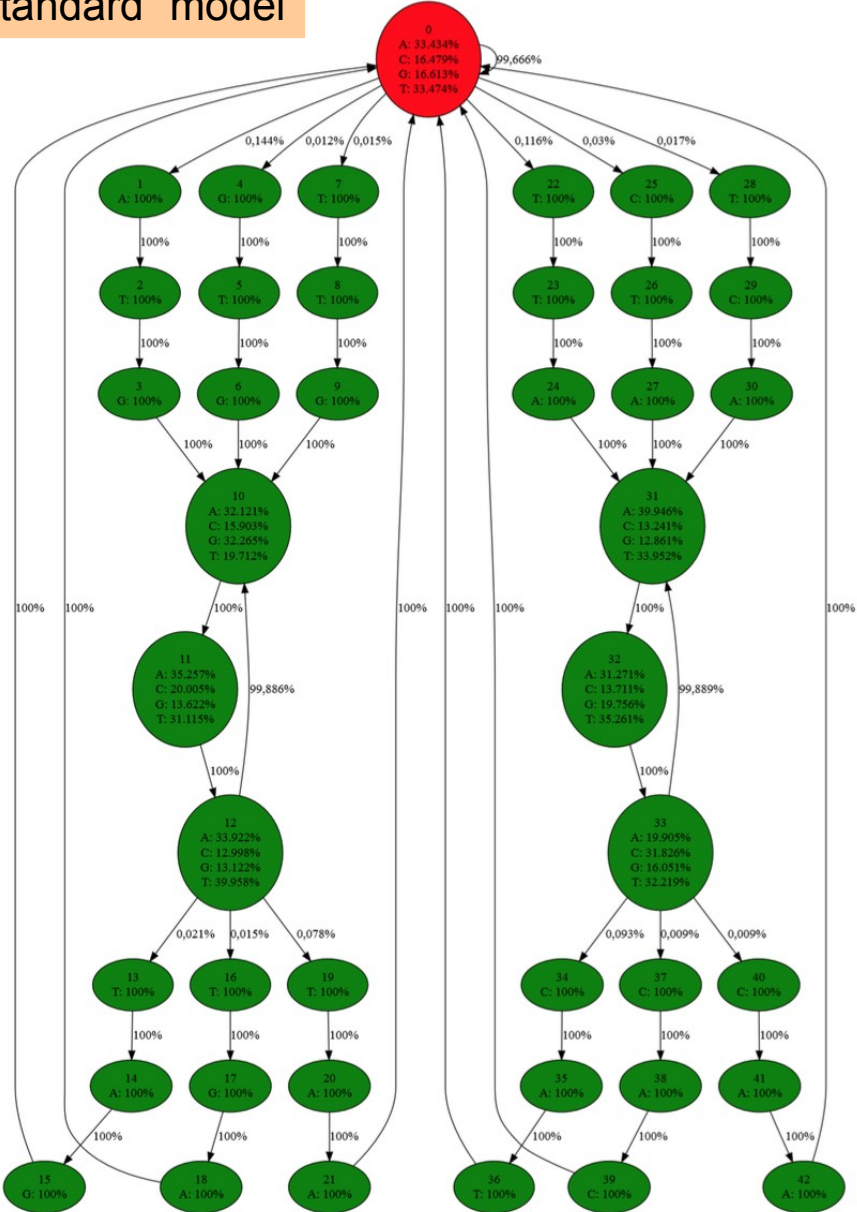
Previous results

ACs between predictions and true annotations sorted by average

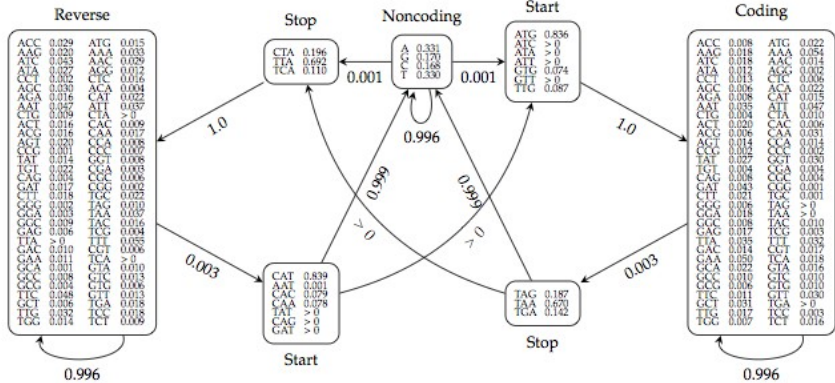
	Genome 6	Genome 7	Genome 8	Genome 9	Genome 10	Average	
3	0,6313	0,6378	0,6007	0,5310	0,4762	0,5754	Codon-models
22	0,6313	0,6378	0,6007	0,5310	0,4762	0,5754	
9	0,6053	0,6147	0,5708	0,4919	0,4197	0,5405	
10	0,4628	0,4587	0,4245	0,3244	0,3611	0,4063	Models inspired by the model presented in class maybe allowing more start/stop codons.
21	0,4619	0,4582	0,4219	0,3239	0,3603	0,4052	
13	0,4539	0,4547	0,4159	0,3121	0,3650	0,4003	
11	0,4522	0,4504	0,4172	0,3139	0,3643	0,3997	
1	0,4520	0,4529	0,4165	0,3122	0,3645	0,3996	
25	0,4543	0,4515	0,4144	0,3133	0,3595	0,3986	
18	0,4521	0,4481	0,4112	0,3095	0,3619	0,3966	
15	0,4501	0,4449	0,4187	0,2974	0,3585	0,3939	
24	0,4237	0,4326	0,4063	0,2769	0,3815	0,3842	
14	0,4359	0,4383	0,3974	0,2642	0,3467	0,3765	
2	0,4286	0,4548	0,3725	0,2473	0,3733	0,3753	Problems with training or prediction
19	0,3856	0,4075	0,3727	0,2926	0,3059	0,3529	
23	0,3854	0,4073	0,3551	0,2336	0,3710	0,3505	
5	0,3850	0,4008	0,3619	0,2559	0,3119	0,3431	
6	0,3821	0,3914	0,3504	0,2365	0,3263	0,3373	
16	0,2917	0,3462	0,2618	0,2240	0,3213	0,2890	
7	0,0247	0,0978	0,0311	0,0642	0,1441	0,0724	
4	0,0254	0,0939	0,0253	0,0631	0,1387	0,0693	
20	0,0775	0,0526	0,1344	-0,0246	0,0765	0,0633	
12	0,0344	0,0820	0,0089	0,0549	0,1144	0,0589	
17	-0,2748	-0,2571	-0,2635	-0,2657	-0,3075	-0,2737	
8							

Examples of models

“Standard” model

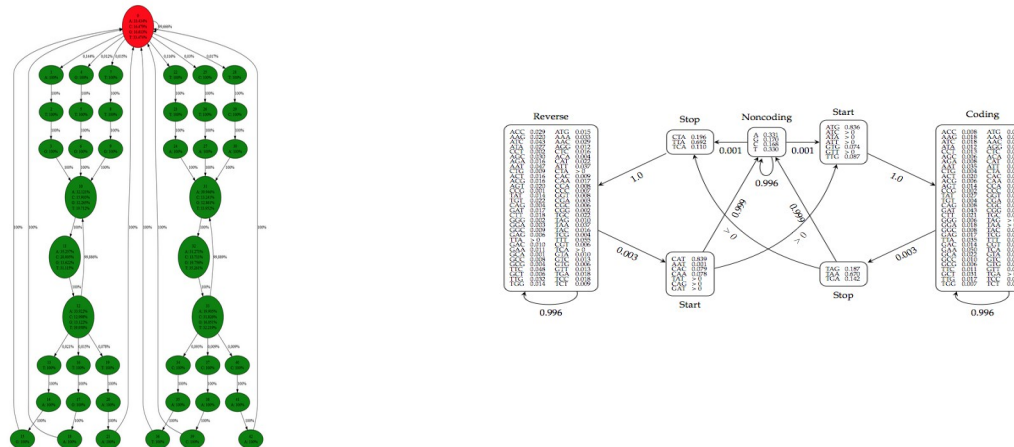


“Codon” model



ACs between predictions and true annotations sorted by average

	Genome 6	Genome 7	Genome 8	Genome 9	Genome 10	Average	
3	0,6313	0,6378	0,6007	0,5310	0,4762	0,5754	Codon-models
22	0,6313	0,6378	0,6007	0,5310	0,4762	0,5754	
9	0,6053	0,6147	0,5708	0,4919	0,4197	0,5405	
10	0,4628	0,4587	0,4245	0,3244	0,3611	0,4063	Models inspired by the model presented in class maybe allowing more start/stop codons.
21	0,4619	0,4582	0,4219	0,3239	0,3603	0,4052	
13	0,4539	0,4547	0,4159	0,3121	0,3650	0,4003	
11	Can you do better than this?					0,3997	
1						0,3996	
25						0,3986	
18						0,3966	
15						0,3939	
24	0,4237	0,4326	0,4063	0,2769	0,3816	0,3842	
14	0,4359	0,4383	0,3974	0,2642	0,3467	0,3765	
2	0,4286	0,4548	0,3725	0,2473	0,3733	0,3753	
19	0,3856	0,4075	0,3727	0,2926	0,3059	0,3529	Problems with training or prediction
23	0,3854	0,4073	0,3551	0,2336	0,3710	0,3505	
5	0,3850	0,4008	0,3619	0,2559	0,3119	0,3431	
6	0,3821	0,3914	0,3504	0,2365	0,3263	0,3373	
16	0,2917	0,3462	0,2618	0,2240	0,3213	0,2890	
7	0,0247	0,0978	0,0311	0,0642	0,1441	0,0724	
4	0,0254	0,0939	0,0253	0,0631	0,1387	0,0693	
20	0,0775	0,0526	0,1344	-0,0246	0,0765	0,0633	
12	0,0344	0,0820	0,0089	0,0549	0,1144	0,0589	
17	-0,2748	-0,2571	-0,2635	-0,2657	-0,3075	-0,2737	
8							



Running time in practice

Using the 7-state “Codon-Model”, my implementation of Viterbi with

```

Chr_CodonHMM — ￼1
[~/Dropbox/Work/Teaching/Classes/Current/ML/Code/2021/Chr_CodonHMM% time python codonhmm.py genome1.fa > pred1.fa
python codonhmm.py genome1.fa > pred1.fa 22.31s user 0.21s system 99% cpu 22.538 total
[~/Dropbox/Work/Teaching/Classes/Current/ML/Code/2021/Chr_CodonHMM%
[~/Dropbox/Work/Teaching/Classes/Current/ML/Code/2021/Chr_CodonHMM% python compare_anns.py true-ann1.fa pred1.fa
Cs (tp=713369, fp=132703, tn=286808, fn=14600): Sn = 0.9799, Sp = 0.8432, AC = 0.7292
Rs (tp=605003, fp=99958, tn=287970, fn=13438): Sn = 0.9783, Sp = 0.8582, AC = 0.7671
Both (tp=1318372, fp=232661, tn=273370, fn=28038): Sn = 0.9792, Sp = 0.8500, AC = 0.6382
[~/Dropbox/Work/Teaching/Classes/Current/ML/Code/2021/Chr_CodonHMM%
[~/Dropbox/Work/Teaching/Classes/Current/ML/Code/2021/Chr_CodonHMM% python
Python 3.9.7 | packaged by conda-forge | (default, Sep 29 2021, 19:24:02)
[Clang 11.1.0 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> 43**2 / 7**2
37.734693877551024
[>>>
[>>> 43**2 / 7**2 * 23
867.8979591836735
[>>>
[>>> 43**2 / 7**2 * 23 / 60
14.464965986394558
[>>>
[>>> █

```

