3a)

The answer after 1000 iterations is:

1.0 0.0 0.0

0.0 1.0 0.0

0.0 0.0 1.0

stochastic: true doublystochastic: true substochastic true

aperiodicity: false gcd 2

irreducibility: false

b) The answer after 111 iterations is:

0.33333333333333337 0.33333333333333337 0.33333333333333337

0.33333333333333337 0.33333333333333337 0.33333333333333337

0.33333333333333337 0.33333333333333337 0.33333333333333337

stochastic: true doublystochastic: true substochastic true

aperiodicity: true gcd 1

irreducibility: true

c) the answer after 57 iterations is:

0.33333333333333337 0.33333333333333337 0.33333333333333337

0.33333333333333337 0.33333333333333337 0.33333333333333337

0.33333333333333337 0.3333333333333333 0.33333333333333337

stochastic: true doublystochastic: true substochastic true

aperiodicity: true gcd 1

irreducibility: true

d) the answer after 195 iterations is:

0.5138888888888917 0.20833333333333448 0.2777777777777793

0.5138888888888917 0.2083333333333345 0.27777777777777934

0.5138888888888917 0.20833333333333448 0.2777777777777793

stochastic: true doublystochastic: false substochastic false

aperiodicity: true gcd 1

irreducibility: true

e) the answer after 1000 iterations is:

3.842443645764679E-93 2.7212337003377595E-93 2.0282878455395072E-93

3.842443645764679E-93 2.7212337003377595E-93 2.0282878455395072E-93

4.7495215458772654E-93 3.363629836280763E-93 2.5071016550234244E-93

stochastic: false doublystochastic: false substochastic true

aperiodicity: true gcd 0

irreducibility: true

4)

See attached work.

P1=1/10

P2 = 3P1 = 3/10

P3 = P1/2 = 1/20

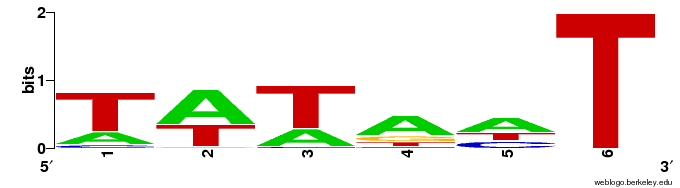
P4 = P1/2 = 1/20

P5 = 12P1/5 = .24

P6 = 1-(P1+P2+P3+P4+P5)= .26

5)

From gibbs\_sampler.java program



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ith position in string | Prob A | Prob C | Prob G | Prob T |
| 0 | 0.229337 | 0.072641 | 0.007114 | 0.690875 |
| 1 | 0.59116 | 0.001416 | 0.029906 | 0.377485 |
| 2 | 0.280619 | 0.035604 | 0.001416 | 0.682328 |
| 3 | 0.613952 | 0.052698 | 0.178055 | 0.155263 |
| 4 | 0.488596 | 0.246431 | 0.004265 | 0.260676 |
| 5 | 0.001416 | 0.001416 | 0.001416 | 0.995718 |

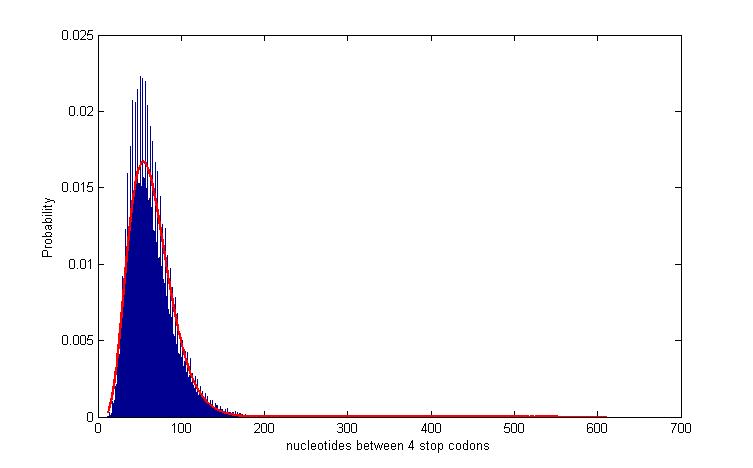
6)

c)

d)

e)

f)

g)

h) From hw3\_6.m

chi square stat = 164.1297. p-value = 0

7)

From nucleotide.java

a)

A=488253

C=315946

G=310016

T=494333

Frequency:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | C | G | T |
| A | 203766 | 63134 | 91594 | 129759 |
| C | 93385 | 71925 | 56618 | 94018 |
| G | 80693 | 96243 | 70077 | 63003 |
| T | 110409 | 84644 | 91727 | 207552 |

Probabilities:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | C | G | T |
| A | 0.12668 | 0.03925 | 0.05694 | 0.08067 |
| C | 0.05806 | 0.04471 | 0.0352 | 0.05845 |
| G | 0.05017 | 0.05983 | 0.04357 | 0.03917 |
| T | 0.06864 | 0.05262 | 0.05702 | 0.12903 |

b)From nucleotide.java

c)

A=466913

C=318896

G=331972

T=490767

Frequency:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | C | G | T |
| A | 171870 | 73290 | 108713 | 113040 |
| C | 89549 | 56049 | 55770 | 117528 |
| G | 104833 | 96831 | 59315 | 70992 |
| T | 100661 | 92726 | 108173 | 189207 |

Probabilities:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | C | G | T |
| A | 0.10685 | 0.04556 | 0.06758 | 0.07027 |
| C | 0.05567 | 0.03484 | 0.03467 | 0.07306 |
| G | 0.06517 | 0.0602 | 0.03687 | 0.04413 |
| T | 0.06258 | 0.05765 | 0.06725 | 0.11763 |

d)

from trinucleotideshuffle in ntshuffle.java

psuedocode based on Altschul-Erickson algorithm from “Significance of Nucleotide Sequence Alignments: A method for Random Sequence Permutation that Preserves Dinucleotide and Codon Usage”

Store first 2 and last 2 nucleotides.

Create hashtable to store dinucleotides where key=substring(i to i+2) and value = substring(i+1 to i+3).

Randomly select a dinucleotide from each key where the key != last dinucleotide.

Create graph where each vertex=key and last nucleotide and each edge=randomly selected dinucleotide from previous step

Traverse graph using depth first search

If(can reach a node == last dinucleotide)

Insert edge back into corresponding key value from which it was taken

Shuffle hash table values from 1:n-1

Start random sequence with first 2 nucleotides

Add nucleotide to list from value according to next item in list that wasn’t used already that has the corresponding key

Test code:

**public** **static** **void** main(String[] args) {

ntshuffle nt = **new** ntshuffle();

String s = "GCTTTTAGGGGTGTTAGGGGTTTATCAAAAATCTAAAAACGCCCTTTCTTCTCAAGCAATTGTCGCTACGAGCATGAGCAATTTAGCCCTTAAAGAATACTTAAAATCCCAAGATTTAGAATTGAAGCATTGCGCGATTGGGGATAAGTTTGTGAGCGAATGCATGCAATTGAATAAAGCCAATTTTGGAGGCGAGCAAAGCGGGCATATCATTTTTAGCGATTACGCTAAAACAGGCGATGGTTTGGTGTGCGCTTTGCAAGTGAGCGCGTTAGTGTTAGAAAGCAAGCAAGTAAGCTCTGTTGCACTAAACCCCTTTGAATTATACCCCCAAAGCCTAATAAATTTGAATATCCAAAAAAAGCCTCCTTTAGAAAGCCTGAAAGGTTATAGCGCTCTTTTAAAGGAATTAGACAAGCTAGAAATCCGCCATTTGATCCGCTATAGCGGCACTGAAAACAAATTACGAATCCTCTTAGAAGCTAAAGATGAAAAACTTTTAGAATCCAAAATGCAAGAATTAAAAGAGTTTTTTGAAGGGCATTTGTGCTAAAAACCACCCAAAAAAGCCTGTTGATTTTTATAGTGGTTTTTTCCCTTATTTTTGGCACGGATCAAGCGATTAAATACGCTATTTTAGAGGGGTTTCGCTATGAAAGTTTGATTATAGATATTGTTTTAGTGTTCAATAAAGGCGTGGCGTTTTCCTTGCTCAGTTTTTTAGAGGGGGGTTTGAAATACTTGCAAATCCTTTTGATTTTAGGGCTTTTTATCTTTTTAATGTGCCAAAAGGAGCTT";

String sss = nt.trinucleotideshuffle(s);

System.*out*.println("triple " + sss);

}

Input:

GCTTTTAGGGGTGTTAGGGGTTTATCAAAAATCTAAAAACGCCCTTTCTTCTCAAGCAATTGTCGCTACGAGCATGAGCAATTTAGCCCTTAAAGAATACTTAAAATCCCAAGATTTAGAATTGAAGCATTGCGCGATTGGGGATAAGTTTGTGAGCGAATGCATGCAATTGAATAAAGCCAATTTTGGAGGCGAGCAAAGCGGGCATATCATTTTTAGCGATTACGCTAAAACAGGCGATGGTTTGGTGTGCGCTTTGCAAGTGAGCGCGTTAGTGTTAGAAAGCAAGCAAGTAAGCTCTGTTGCACTAAACCCCTTTGAATTATACCCCCAAAGCCTAATAAATTTGAATATCCAAAAAAAGCCTCCTTTAGAAAGCCTGAAAGGTTATAGCGCTCTTTTAAAGGAATTAGACAAGCTAGAAATCCGCCATTTGATCCGCTATAGCGGCACTGAAAACAAATTACGAATCCTCTTAGAAGCTAAAGATGAAAAACTTTTAGAATCCAAAATGCAAGAATTAAAAGAGTTTTTTGAAGGGCATTTGTGCTAAAAACCACCCAAAAAAGCCTGTTGATTTTTATAGTGGTTTTTTCCCTTATTTTTGGCACGGATCAAGCGATTAAATACGCTATTTTAGAGGGGTTTCGCTATGAAAGTTTGATTATAGATATTGTTTTAGTGTTCAATAAAGGCGTGGCGTTTTCCTTGCTCAGTTTTTTAGAGGGGGGTTTGAAATACTTGCAAATCCTTTTGATTTTAGGGCTTTTTATCTTTTTAATGTGCCAAAAGGAGCTT

Output:

GCTAGCACTTTTTGAGCAAGAGCTAGTTTTTCTTTAATCCAAAGGCATCAAAACCCCCTAGTTTTGTGGGCGCTCCAGAATTGTTAACGCTGAAGCTTAGATAAAACATAAAATTTTTTTACGATCCTGCATAGGGAGAATAGAAAAACGTTTCAACGCGCTTAGGATAAGGGTTTAATCTTATCTCATAAGAAGCGCGCTTTTATTAATAAAGTGTAGGCATTTTAGCCTTACTAAAGAGGTTTGCGCTTGTTTTCGTTTGAGCCTATCTGATTTTGAAAGAGATGCAAAAGTGTGCAAATTTACTTGAAATACAAATAAAATTTGATTTATAGGTTATTAATCCAATCTCTTAATGGGTGTTTAGCAAAGGGGTGGGGTTGGGGGGTTGAATTTGTCCAGGCAAGGGGAGGAGCGAAGTGTGAAGCAAATCCAAAGCGCCAATTTACTTTTTAGATTGAGCCCGGTTTTTGCTTTTTGGAAAGACCTTCATAGTTTTACGCGCCTCGGCAAAGCCTTCCTCCTAGCGATTGCGATTATGAAAGATGCTAGAAAGCAAATCAAATATTTTGATCATAAAAAAGCGTTAGCCCACAATTTTCCACTTGAGCTATTAAGCAATTTTTGAAAGAAAAAGCTACCAAAAATTGCTTTAGAAGCACGATGAAATTATCCCCCCCTAGTGCAAAATATTTGGCCGATTGTGTTTTTTGAAGCGAAAAATGAAGGCGAATGCGGGCCTCTGCTAGCAACCTTATAAAGTGTTATTAAAGCAAAAATGGCGCTTTTATTTTAGTT

g)

from ptata.pl