CHALMERS | GÖTEBORGS UNIVERSITET

Hoppande gener och antibiotikaresistens

Utveckling av en matematisk modell för att identifiera förekomsten av hoppande gener hos bakterier

Examensarbete för kandidatexamen i matematisk statistik vid Göteborgs universitet

Kandidatarbete inom civilingenjörsutbildningen vid Chalmers tekniska högskola

Emily Curry
Sara Finati
Rikard Isaksson
Anna Källsgård
Anton Martinsson
Deimante Neimantaite

Institutionen för matematiska vetenskaper
Chalmers tekniska högskola
Göteborgs universitet

Göteborg 2016

Hoppande gener och antibiotikaresistens

Utveckling av en matematisk modell för att identifiera förekomsten av hoppande gener hos bakterier

Examensarbete för kandidatexamen i matematisk statistik vid Göteborgs universitet
Rikard Isaksson

Kandidatarbete i bioinformatik inom civilingenjörsprogrammet Bioteknik vid Chalmers tekniska högskola
Anton Martinsson

Kandidatarbete i matematik inom civilingenjörsprogrammet Teknisk matematik vid Chalmers tekniska högskola
Emily Curry
Sara Finati
Anna Källsgård
Deimante Neimantaite

Handledare: Mariana Pereira
Tobias Österlund
Erik Kristiansson
Examinator: Marina Axelson-Fisk
Maria Roginskaya

Institutionen för matematiska vetenskaper
Chalmers tekniska högskola
Göteborgs universitet
Göteborg 2016

Populärvetenskaplig presentation

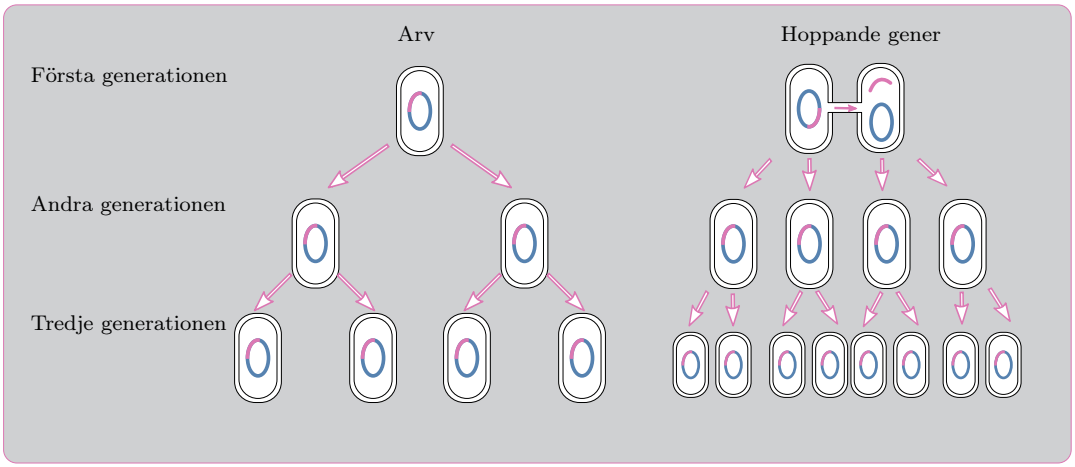
De flesta har någon gång blivit behandlade med antibiotika för att bli friska från en infektion eller för att förhindra att en infektion uppkommer. I framtiden kommer detta kanske inte vara möjligt. Antibiotikaresistens har kallats för den tysta epidemin och anses vara ett lika stort hot mot mänskligheten som klimatförändringar. Enligt Världshälsoorganisationen är antibiotikaresistens inte längre ett hot i framtiden, det är en verklighet. Redan idag rapporteras det om allt fler bakterier som är resistenta mot antibiotika och en fortsatt spridning kommer att påverka människor i alla åldrar jorden runt.

Olika bakterier kan överföra antibiotikaresistens mellan varandra genom att dela DNA. I vårt kandidatarbet har vi utformat en matematisk modell som kan användas för att förstå hur omfattande spridningen av antibiotikaresistens hos bakterier är. Modellen använder sig av olika statistiska metoder för att beräkna sannolikheten för delsekvenser av en bakteries DNA, vilket kan användas för att hitta delar som härstammar från andra bakterier. Detta kan i sin tur användas för att förstå överföringen av antibiotikaresistens mellan bakterier.

I vardagligt tal är antibiotika ett samlingsnamn på en mängd läkemedel som används för att bota bakteriesjukdomar. Det upptäcktes av Alexander Fleming 1928 av en slump och började användas på människor 1941. Antibiotika verkar så att bakteriens celler dör utan att påverka de mänskliga cellerna vilket gör att antibiotika är harmlös för människor. Eftersom många sjukdomar orsakas av bakterier räknas upptäckten av antibiotika till en av medicinhistoriens allra största, och uppskattningsvis har antibiotika räddat livet på mer än 200 miljoner människor. Det är därför uppenbart att antibiotikaresistens är ett stort hot som kräver en snabb lösning.

Antibiotikaresistens uppkommer genom att en bakterie har gener som kodar för detta i sitt DNA. DNA består av de fyra baserna 'A', 'T', 'G' och 'C'. Dessa kallas även för nukleotider.

Kodande delsekvenser av DNA-molekylen kallas för *gener* och kan översättas till olika sorters *protein* som i sin tur styr olika funktioner hos organsimen. Det kan till exempel vara ett protein som ger antibiotikaresistens. Det betyder alltså att en bakterie som är motståndskraftig mot antibiotika har något typ av gen som gör att antibiotikan inte verkar. Bakterien kan ha fått en sådan här gen på olika sätt. Genen kan till exempel ha uppstått till följd av att nukleotider i DNA-sekvensen bytts ut mot andra nukleotider, vilket lett till att en ny typ av gen uppstått. Den här genen kan sedan föras vidare till nya bakterier vid fortplantning. Dessutom kan bakterier, till skillnad från människor som endast kan föra vidare anlag till sina barn, utbyta gener direkt med varandra. Det betyder att genen kan föras vidare under samma generation och mellan olika bakteriearter vilket gör att resistens kan spridas väldigt snabbt, vilket illustreras i Figur 1. Gener som har skickats över på det här sättet kallas för horisontellt överförbara gener och mer informellt *hoppande gener*.



Figur 1: Visar hur gener kan överföras mellan bakterier genom arv under generationer eller via hoppande gener över en generation.

Syftet med kandidatarbetet var alltså att hitta dessa hoppande gener genom att skapa en modell som kan urskilja de hoppande generna från bakteriens ”egna” DNA. Detta gjordes genom att beräkna sannolikheter för DNA-sekvensen med så kallade *Markovkedjor*. En DNA-sekvens kan ses som en Markovkedja av olika ordningar. För en första ordningens Markovkedja beror övergången till en nukleotid endast på den tidigare nukleotiden. Till exempel, låt säga att

'ATATATACG'

är en delsekvens från en bakteries DNA. I sekvensen finns det fyra stycken 'A', tre stycken 'T', ett 'G' och ett 'C'. Den vanligast förekommande nukleotiden i sekvensen är alltså 'A' och det är därför mest sannolikt att en nukleotid i denna sekvensen är 'A'.

Från 'A' sker tre övergångar till 'T' och en övergång till 'C'. Det betyder att tre av fyra övergångar från 'A' är till 'T', vilket motsvarar 75%. Detta innebär att 25% av övergångarna från 'A' är till 'C'. Från 'T' sker tre övergångar till 'A' och från 'C' en övergång till G. Detta ger att 100% av alla övergångar från 'T' är till 'A' och 100% av övergångarna från 'C' är till 'G'. Genom att utöka det här resonemanget till hela bakteriens DNA-sekvens ges en komplett bild över alla övergångar som sker i sekvensen och det är dessa övergångar som vår modell grundar sig på.

För Markovkedjor av ordning två eller högre betraktas istället övergångar från två eller flera nukleotider till en annan, till exempel skulle den första övergången för andra ordningens Markovkedja i exemplet vara från 'AT' till 'A'.

Det går att anta att övergångarna i olika bakteriers DNA skiljer sig åt, alltså att de har olika fördelningar av 'A', 'T', 'G' och 'C'. Det betyder att sannolikheten för de hoppande generna skiljer sig från bakteriens andra gener eftersom de härstammar från en annan bakterie och därmed har en annan fördelning av nukleotider.

Detta antagande ligger till grund för arbetet. Första steget var att beräkna sannolikheten för att de olika övergångarna ska ske baserat på fördelningen av nukelotider för hela DNA-sekvensen. Andra steget var att dela upp hela DNA:t i delsekvenser och jämföra hur väl sannolikheten att övergångarna som sker i varje delsekvens överensstämmer med hela DNA:t. Utifrån detta kan avvikande delsekvenser identifieras eftersom de har betydligt lägre sannolikhetsvärden än de egna generna och kan därför antas vara hoppande gener.

Den färdiga modellen testades först på simulerade DNA-sekvenser för att utvärdera hur väl modellen hittar hoppande gener. Eftersom det också är viktigt att modellen inte pekar ut bakteriens egna DNA som hoppande gener gjordes även tester för detta. För simulerade DNA-sekvenser gav testerna goda resultat och modellen kunde börja användas på riktiga DNA-sekvenser från bakterier för att hitta hoppande gener.

Modellen användes på åtta olika bakterier, bland annat på *E. coli* som finns naturligt i kroppen men som även kan ge upphov till magsjuka samt används flitigt i molekylärbiologi. I slutändan lyckades modellen hitta antibiotikaresistensgener samt en mängd gener som kodar för enzym som hjälper gener att hoppa, vilket tyder på att modellen kan användas för att hitta hoppande gener.

Sammanfattningsvis är antibiotikaresistens ett mycket allvarligt problem som sprids snabbt på grund av bakteriers förmåga att utbyta egenskaper med varandra genom så kallade hoppande gener. Därför krävs nya metoder och strategier för att hitta en lösning. Metoden som vi har arbetat fram för att bestämma omfattningen av hoppande gener har visat sig framgångsrik för de undersökta bakterierna och vi hoppas att resultaten kan användas i fortsatt forskning kring antibiotikaresistens.

Sammanfattning

Hoppande gener mellan bakterier är idag en av de största orsakerna till spridningen av antibiotikaresistens. Då många sjukdomar behandlas med antibiotika är det här ett stort problem. Detta kandidatarbete syftar därför till att utveckla en matematisk modell med uppgift att identifiera främmande gener hos en bakterie.

Genom att betrakta DNA-sekvenser som Markovkedjor utformades en modell som beräknar sannolikhetsvärden för delsekvenser av en bakteries genom. Områden med låga sannolikhetsvärden urskiljdes då motsvarande DNA-sekvenser eventuellt härstammar från en annan bakterie. Detta utfördes för olika ordningar på Markovkedjan samt för olika längder på delsekvenserna för att få en överblick över modellens beteende.

En undersökning av antagandet att DNA-sekvenser kan ses som Markovkedjor resulterade i att antagandet inte gäller. Däremot påvisar både en sensitivites- och specificitetsanalys på simulerade DNA-sekvenser samt tester av verkliga genom goda resultat. Modellen identifierar flertalet hoppande gener varav en del går att härleda till antibiotikaresistenta gener.

Modellen upptäcker också områden som inte är relaterade till hoppande gener. Därför presenteras en redogörelse för hur en eventuell vidareutveckling av modellen kan genomföras.

Abstract

The so called jumping genes are one of the main reasons why antibiotic resistance is becoming more common. Since a lot of diseases are treated with antibiotics, antibiotic resistant bacteria is a huge problem. Consequently, the ability to decide the rate of the spreading of resistance is of importance in order to be able to understand and limit it. The aim of this project is therefore to construct a mathematical model to identify putative antibiotic genes.

In this project, bacterial DNA-sequences are treated as Markov chains and therefore, a model calculating the likelihood of sequences can be constructed. The low-likelihood areas, which could correspond to DNA-sequences originating from different bacteria, are separated and further investigated. The model is evaluated for several parameter values to distinguish the best performing ones.

A verification of the assumption that DNA-sequences from a bacterial genome can be treated as Markov chains concluded in an unaffirmative result. However, the tests done on the model for simulated and real sequences show good results. The model is able to identify jumping genes, some of which were found to be antibiotic resistance genes.

The model also finds regions not related to jumping genes. Therefore, a continuation of this project is presented and discussed.

Förord

Vi vill börja med att tacka Marina Axelson-Fisk, författare till *Comparative Gene Finding, Models, Algorithms, Implementation* som har varit en stor inspirations- och kunskapskälla till det här arbetet samt till Monika Skuriat-Olechnowska som har skrivit *Statistical Inference for Markov Chains with interval censoring* som delar av arbetet baserats på.

Första delen av arbetet utfördes tillsammans. Detta innefattade utformning och implementation av modellen. Inför utvärderingen av modellen delades gruppen in i mindre grupper där Emily och Anna ansvarade för ”Sensitivites- och specificitetsanalys”, Anton och Rikard för ”Utvärdering av träffar” och Sara och Deimante för ”Verifikation av Markovegenskap”. Gruppens individer har alla bidragit till kreativitet, problemlösning och diskussion i arbetet. I Tabell 1 redogörs huvudförfattarna för varje avsnitt. Observera att avsnitten kan ha flera huvudförfattare. Hela texten har dock diskuterats och bearbetats av samtliga gruppens medlemmar. Utöver detta har dagbok och enskilda tidsloggar förts kontinuerligt under arbetet där arbetsprocessen respektive de individuella bidragen beskrivits.

Till sist vill vi tacka våra handledare Tobias Österlund, Mariana Pereira och Erik Kristiansson för stort stöd och engagemang under arbetets gång. Tack för fikat och alla givande diskussioner kring såväl Markovkedjor som portugisiska trolighetsfunktioner. Detta har gjort torsdagar till veckans höjdpunkt.

Tabell 1: Huvudförfattare till avsnitten i rapporten.

Huvudansvarig författare	Avsnitt
E. Curry	1 (ingress), 1.2, 1.3, 1.4, 2 (ingress), 2.2, 2.2.1, 3.3 (ingress), 3.3.1, 4 (ingress), 4.1-4.1.3, 4.2, 5 (ingress), 5.1-5.1.2, 5.2, 5.6.1, 5.6.3-5.6.5, 5.7
S. Finati	2.2.7, 3(ingress) 3.1, 3.5, 4.5, 5.5, 5.6.6, 5.7
R. Isaksson	2.2.3, 2.2.4, 3.4, 3.4.2, 4.4- 4.4.2, 5.4, 5.4.2, 5.6.1
A. Källsgård	1 (ingress), 2 (ingress), 2.1.3, 2.2.2, 3.3 (ingress), 3.3.1, 4 (ingress), 4.1-4.1.3, 4.2, 5 (ingress), 5.1-5.1.2, 5.2, 5.7
A. Martinsson	1.3, 2.1 (ingress), 2.1.2, 3.2, 3.4, 3.4.1, 4.4, 4.4.1, 5.3-5.4.1, 5.6 (ingress), 5.6.2, 5.7
D. Neimantaite	2.1.1, 2.2.5-2.2.7, 4.5, 5.5, 5.6.6, 5.7

Ordlista

alternativ hypotes oftast en motsägelse till nollhypotesen. 10

Athena Matematiska vetenskapers bioinformatik-server. 17

Bootstrap en metod för att mäta tillförlitligheten för en skattare genom slumpmässig sampling. 8

den centrala dogmen informationsflödet i cellen. 3

direkta repeats kopior av en specifik nukleotidsekvens. 4

dubbelhelix dubbelsträngad struktur av DNA dubbelspiral. 3

frihetsgrader ett mått på hur många parametrar i ett system som kan variera. 11

fönster en mindre del av den totala DNA-sekvensen för vilka sannolikhetsvärden beräknas. 12

genom summan av alla gener hos en organism. 3

genomiska öar Ett större transposabelt element, över tusen baspar, som har överförts via horisontell genöverföring. 4

homogen sannolikheten för ett tillstånd/övergång är lika stor oavsett var i processen den sker. 6

hoppande gen en gen som överförts via horisontell genöverföring. 4

horisontell genöverföring överföring av gener mellan organismer (under samma generation). 4

hypoteser antaganden för en mängd data. 10

inklipp en sammanhängande del av DNA-sekvensen som modellen ska identifiera. 15

intitiala sannolikhetsfördelningen sannolikheten för det första tillståndet i en kedja/-proccess. 6

inverterade repeats en sekvens av nukleotider som nedströms följs av sitt komplementerande segment. 4

kedja en stokastisk process med diskret tillståndsrum. 6

klippa- och klistra-mekanism en gen ”klipps” ut ur DNA-strängen och ”klistras” in någon annanstans. 4

kodon tripletter av kvävebaser som avläses till en aminosyra. 3

konfidensgrad den grad av säkerhet som konfidensmängden innehåller en skattad parameter. 8

konfidensintervall ett specialfall av en konfidensmängd där parametern är endimensionell och reellvärd. 8

konfidensmängd en mängd som med given sannolikhet innehåller en skattad parameter. 8

korsklassificering undersökning av en träff genom att beräkna sannolikhetsvärden med avseende på övergångsmatrisen hos en annan bakterie med syfte att undersöka ursprung. 17

korstabell en tabell i matrisformat som visar samband mellan olika kategoriska variabler. 10

log-likelihood logaritmen av sannolikhetsfunktionen. 8

Markovegenskapen föregående och nästkommande tillstånd är oberoende, givet det nuvarande tillståndet. 6

Markovkedja en stokastisk process som är minneslös. 6

multiresistent en bakterie som är resistens mot flera sorters antibiotika. 6

mutation förändring i cellers genetiska material (DNA eller RNA). 4

National Center for Biotechnology Information en hemsida som innehåller en mängd bioinformatiska databaser och annan information inom bioteknik. 12

nollhypotes en hypotes kring en egenskap hos en undersökt datamängd. 10

nukleotider Molekylära byggstenar till DNA och RNA som består av en kvävebas, en sockermolekyl och en eller flera fosfatgrupper. 3

nukleotidnivå sensitivitets- och specificitetsanalys där varje enskild identifierad nukleotid jämförs med nukleotider från inklipp. 16

p-värdet sannolikheten att observera ett mer extremt provutfall än det redan observerade om nollhypotesen är sann. 11

Parametrisk Bootstrap innebär att stickprov simuleras från en modell där parametrar tidigare skattats. 8

parvis klassificering jämför identifierade intervall hos genomet för en bakterie med träffar och hela genom från andra bakterier för att se om träffarna går att hitta i andra bakterier. 17

post-transkriptionell modifiering Förändring av protein som sker efter transkriptionen. 5

provutfall ett värde för prövningen vid hypotesprövning. 10

replikation den process då DNA kopierar sig själv och sker vid celldelning. 3

sensitivitet mått på andelen sanna positiva resultat. 9

signifikansnivån ett mått på sensitivitet vid hypotesprövning som mäter risken att felaktigt förkasta nollhypotesen. 10

specificitet mått på andelen sanna negativa resultat. 9

SSO antalet gånger en specifik STS förekom i DNA-sekvensen. 18

startkodon kodon som initierar proteintillverkning. 3

stokastisk process en process som utvecklas slumpmässigt i tiden. 6

stopkodon kodon som avslutar ett protein. 3

STS tre-tillståndssekvens, innefattande föregående, nuvarande och nästkommande tillstånd. 18

tillståndsrummet alla möjliga värden en slumpvariabel kan anta. 6

transkription den process då DNA översätts till mRNA. 3

translation den process då mRNA översätts till aminosyror, som i sin tur byggs upp till ett protein. 3

transposabla element DNA-sekvenser som kan ”hoppa” inom genomet och mellan olika organismer. 4

transposaser Protein som katalyserar förflyttning av en transposon i genomet. 4

träffar en sammanhängande del av DNA-sekvensen som har ett lägre sannolikhetsvärde än tröskelvärdet. 13

träffnivå sensitivitets- och specificitetsanalys där träffar jämförs med inklipp genom att undersöka om träffen täcker ett helt inklipp. 16

TSO antalet gånger en specifik två-tillstånds sekvens påträffades. Den inkluderades av tillstånden: föregående och nuvarande. 18

vertikal genöverföring överföring av DNA från generation till generation. 4

övergångsmatris en $|S| \times |S|$-matris över alla möjliga övergångar, där S är tillståndsrummet. 6

Innehåll

1 Inledning **1**
- 1.1 Syfte . . . 1
- 1.2 Problem och uppgift . . . 1
- 1.3 Avgränsningar . . . 2
- 1.4 Precisering av frågeställningen . . . 2

2 Teori **3**
- 2.1 Biologisk bakgrund . . . 3
  - 2.1.1 Molekylärbiologi och den centrala dogmen . . . 3
  - 2.1.2 Evolution och horisontell genöverföring . . . 4
  - 2.1.3 Antibiotikaresistens och dess mekanismer . . . 5
- 2.2 Matematisk bakgrund . . . 6
  - 2.2.1 Stokastiska processer och Markovkedjor . . . 6
  - 2.2.2 Markovmodell för DNA-sekvenser . . . 7
  - 2.2.3 Konfidensintervall . . . 8
  - 2.2.4 Parametrisk Bootstrap . . . 8
  - 2.2.5 Sensitivitet och specificitet . . . 9
  - 2.2.6 Statistisk hypotesprövning . . . 10
  - 2.2.7 Analys av korstabell med χ^2-oberoendetest . . . 10

3 Metod **12**
- 3.1 Utformning av modellen och implementation i Python . . . 12
- 3.2 Val av bakterier . . . 14
- 3.3 Utvärdering av modellen . . . 15
  - 3.3.1 Bestämning av parametrar med sensitivitets- och specificitetsanalys . . . 15
- 3.4 Utvärdering av träffar . . . 17
  - 3.4.1 Parvis klassificering . . . 17
  - 3.4.2 Undersökning av ursprung med korsklassifiering . . . 17
- 3.5 Verifikation av Markovegenskap för K12 . . . 18

4 Resultat **20**
- 4.1 Resultaten från sensitivitets- och specificitetsanalysen . . . 20
  - 4.1.1 Hur ordningen påverkar modellens prestanda . . . 20
  - 4.1.2 Hur fönsterstorleken påverkar modellens prestanda . . . 22
  - 4.1.3 Hur olika storlekar på gener påverkar modellens prestanda . . . 25
- 4.2 Antal träffar för *Escherichia coli* stam K12 . . . 26
- 4.3 Antal träffar för olika genom . . . 28
- 4.4 Utvärdering av träffar . . . 29
  - 4.4.1 Parvis klassificering . . . 29
  - 4.4.2 Korsklassificering . . . 30
- 4.5 Resultaten från verifiering av Markovegenskapen för olika ordningar . . . 32

5 Diskussion **34**
- 5.1 Modellens prestanda utifrån sensitivtet och specificitet . . . 34
  - 5.1.1 Val av ordning . . . 35
  - 5.1.2 Val av fönsterstorlek . . . 35
- 5.2 Antalet träffar för *Escherichia coli* stam K12 . . . 38
- 5.3 Antal träffar för olika genom . . . 39
- 5.4 Utvärdering av träffar . . . 39
  - 5.4.1 Parvis klassificering . . . 41
  - 5.4.2 Korsklassificering . . . 41
- 5.5 Verifikation av antagandet om Markovegenskapen . . . 41
- 5.6 Vidareutveckling av modellen . . . 42
  - 5.6.1 Överanpassning till data . . . 42
  - 5.6.2 CpG-separation . . . 42

5.6.3 Undersökning av hur en hoppande gen ska definieras 43
5.6.4 Träffars precision i förhållande till inklipp 43
5.6.5 Definitionen av en sann positiv träff 43
5.6.6 Uppföljning av Markovegenskapen 44
5.7 Slutsatser och återkoppling till frågeställning 44

A Bevis **49**
A.1 Bevis för elementen i övergångsmatrisen 49
A.2 Bevis för högre ordningens Markovkedjor 49

B Implementation **50**
B.1 Python-kod 50
B.2 Beskrivning av funktioner 57

C Resultat **58**
C.1 Resultat för olika inklippsstorlekar 58
C.2 Övriga resultat från utvärdering av träffar 61
C.3 Fullständig frekvenstabell för ordning 1 65

1 Inledning

Ett av de viktigaste läkemedlen inom sjukvården är antibiotika som används för att behandla infektioner orsakade av bakterier. Antibiotika verkar genom att till exempel förstöra eller hindra uppbyggnad av cellväggen hos de skadliga bakterierna [1]. Eftersom antibiotika inte bara används för att direkt behandla sjukdomar, som lunginflammation och kikhosta, utan även i förebyggande syfte för att minska infektionsrisken vid operationer och i samband med cellgiftsbehandlingar [2], är det en nödvändig del i dagens sjukvård. På grund av antibiotikans breda användningsområde i den moderna sjukvården kommer antibiotikaresistens leda till ett stort steg bakåt i den medicinska utvecklingen.

Den omfattande användningen av antibiotika har gjort att allt fler bakterier utvecklar antibiotikaresistens på grund av det ökade selektionstrycket [3], vilket innebär att antibiotikan är verkningslös mot dessa bakterier. När det finns mycket antibiotika i omlopp leder det till fler antibiotikaresistenta bakterier. Till exempel finns det stora mängder antibiotikaresistenta bakterier som orsakar vanliga infektioner som urinvägsinfektion och lunginflammation [3].

Resistensen uppkommer från gener som kodar för antibiotikaresistens och dessa gener kan föras vidare till nya bakterier via celldelning som är bakteriernas förökningsförlopp. Utöver detta har bakterier även förmågan att utbyta arvsmassa direkt med andra bakterier genom så kallade *hoppande gener*. På så sätt kan resistens överföras mellan bakteriestammar och mellan bakterier av olika arter vilket betyder att antibiotikaresistens sprids snabbt [4].

För sjukvården innebär fler antibiotikaresistenta bakterier att tidigare behandlingsbara sjukdomar kan bli livshotande [5]. Till exempel har det rapporterats om flera misslyckade antibiotikabehandlingar mot gonorré vilket innebär att sjukdomen snart kan bli obotlig eftersom det inte finns några andra vaccin eller behandlingsmetoder [4].

Detta är bara ett exempel på en sjukdom som i framtiden kan komma att bli obehandlingsbar till följd av spridningen av antibiotikaresistens. Det är därför ytterst viktigt att kunna bestämma hur ofta bakterier tar upp arvsmassa från andra bakterier för att förstå och försöka begränsa spridningen [6]. Att laborativt undersöka förekomsten av hoppande gener i en bakteries DNA är en relativt kostsam och tidskrävande process. Därför är det av intresse att hitta andra metoder som kan undersöka detta genom att enbart betrakta bakteriers DNA-sekvens.

1.1 Syfte

Det här kandidatarbetet syftar till att konstruera en matematisk modell för att identifiera främmande gener hos bakterier och på så sätt få fram sannolikheten att en bakterie har tagit upp DNA från andra arter under evolutionen.

1.2 Problem och uppgift

Uppgiften är att konstruera en modell för att hitta sekvenser i bakteriers genom, det vill säga en bakteries arvsmassa, som eventuellt härstammar från andra bakterier. De bakterier som undersöks är:

- *Escherichia coli* stammarna K12 och SMS-3-5
- *Bacteroides thetaiotaomicron*
- *Aquifex aeolicus*
- *Chlamydophila pneumoniae*
- *Staphylococcus aureus*
- *Streptococcus pneumoniae*
- *Streptomyces scabiei.*

Genom att anta att genomet hos en bakterie har en unik fördelning och sedan studera sannolikheter för delsekvenser av genomet bör hoppande gener kunna identifieras som sekvenser som avviker från fördelningen.

I modellen betraktas en DNA-sekvens som en Markovkedja och en övergångsmatris beräknas baserat på bakteriens genom. Övergångsmatrisen används sedan för att beräkna sannolikhetsvärden för delsekvenser av genomet, så kallade fönster. Dessa används för att avgöra om det är en hoppande gen, då hoppande gener förväntas ha en annan sannolikhetsfördelning. Modellen analyseras sedan för olika ordningar av Markovkedjor samt för olika storlekar på fönster för att bestämma vilken kombination av parametrar som är mest lämplig.

Till slut undersöks de delsekvenser som modellen identifierar genom att bestämma vilka gener de kodar för. Därifrån dras slutsatser om huruvida det är en hoppande gen eller inte. Dessutom undersöks delsekvensernas ursprung genom att jämföra sekvensernas fördelning med fördelningen hos de övriga bakterierna.

1.3 Avgränsningar

För att konstruera modellen som ska hitta avvikande sekvenser med hjälp av Markovkedjor gjordes en rad avgränsningar. Dessa omfattar såväl utformningen av modellen som undersökningen av modellens resultat.

I modellen betraktas DNA-sekvenserna som homogena Markovkedjor, det vill säga att sannolikheten för en viss övergång från en nukleotid till en annan är lika stor oavsett var i DNA-sekvensen övergången sker, förutsatt att det inte är en hoppande gen. Detta är en förenkling då nukleotidfördelningen varierar beroende på vilken del av genomet som undersöks [7].

Dessutom antas det att Markovegenskapen gäller för DNA-sekvenser. Det betyder att efterföljande nukleotider i sekvensen endast är beroende av de k närmast föregående nukleotiderna, där k är ordningen på Markovkedjan. Den här egenskapen undersöks men oberoende av resultatet skapas modellen utifrån detta antagande.

För att undersöka härkomsten av de sekvenser som modellen identifierar skulle många fler bakterier behöva ingå i undersökningen. Som nämnt i föregående avsnitt undersöks åtta olika bakterier, vilket är en liten del av det totala antalet arter som finns och därför en betydande avgränsning.

1.4 Precisering av frågeställningen

De frågor som behandlas i rapporten är:

1. Kan DNA-sekvenser betraktas som Markovkedjor?
2. Vilken ordning av Markovkedjor ger bäst resultat?
3. Vilken fönsterstorlek ger bäst resultat?
4. Hur bra resultat ger modellen?
5. Vilka typer av gener hittar Markovmodellen?
6. Går det att hitta hoppande gener med hjälp av Markovkedjor?
7. Går det att härleda eventuella hoppande geners ursprung till en bakterie av annan art?
8. Vilka begränsningar har modellen?

2 Teori

Detta kandidatarbete är en kombination av matematisk statistik och bioinformatik med syftet att analysera DNA-sekvenser. I det här avsnittet presenteras den teori som krävs för förståelse av arbetet. Teoridelen omfattar därför molekylärbiologi, där DNA och dess uppbyggnad och funktion beskrivs, evolutionsteori och utvecklingen av genom hos bakterier, där bland annat hoppande gener kommer behandlas samt stokastiska processer och mer specifikt Markovkedjor och dess tillämpning på DNA. Till slut beskrivs teorin bakom olika statistiska metoder och begrepp som används i rapporten, där bland annat Parametrisk Bootstrap, sensitivitets- och specificitetsanalys och statistisk hypotespröving förklaras.

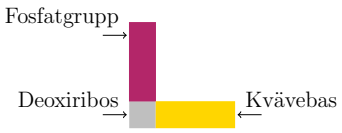
2.1 Biologisk bakgrund

Detta avsnitt behandlar de områden inom biologi som arbetet baseras på. Eftersom det här arbetet fokuserar på bioinformatik och den bakomliggande matematiken ges en översiktlig förklaring av centrala begrepp och mekanismer. Syfte är att ge en grund för att förstå vad DNA är och hur evolution och hoppande gener kan kopplas till antibiotikaresistens.

2.1.1 Molekylärbiologi och den centrala dogmen

DNA (*deoxiribonukleinsyra*) är en molekyl som finns i nästan alla levande organismer. Molekylen kan ses som ett kontrollcenter som sänder ut information om hur en organism ska byggas upp och fungera.

DNA är uppbyggd i form av två spiralvridna kedjor. Varje kedja består av *nukleotider* som i sin tur består av tre komponenter: en kvävebas, en sockermolekyl (deoxiribos) och en fosfatgrupp, se Figur 2. Varje nukleotid i den ena kedjan bildar par med motsvarande komplementära nukleotid i den andra kedjan och på så sätt bildas en stegliknande struktur som kallas *dubbelhelix*. Detta sker genom vätebindningar mellan två nukleotider. Det finns fyra olika kvävebaser, Adenin ('A'), Tymin ('T'), Guanin ('G') och Cytosin ('C'), där 'A' bara kan binda till 'T', och 'G' bara till 'C' [8].

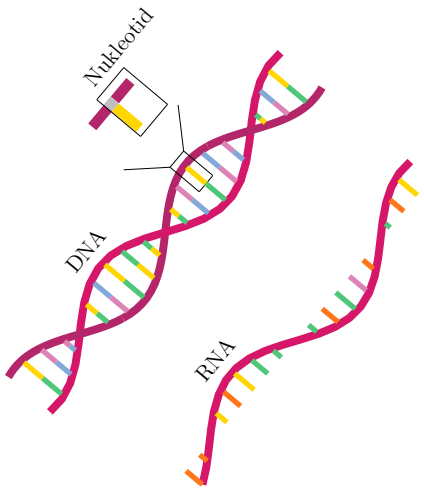


Figur 2: Uppbyggnaden av en nukleotid. Varje nukleotid består av en fosfatgrupp, en sockermolekyl (deoxiribos) och en kvävebas. Bild av: Deimante Neimantaite.

Varje cell i en organism innehåller identiska kopior av DNA-molekylen. Cellen innehåller också en annan DNA-liknande molekyl, RNA (*ribonukleinsyra*). Skillnaden mellan DNA och RNA är att Tymin är ersatt med Uracil, sockret är ribos och att RNA-molekylen är en enkel spiral, se Figur 3 [9]. RNA kan förekomma som olika typer (mRNA, tRNA, miRNA, siRNA, etc) och fungerar som en hjälpreda vid informationsflödet i cellen, även kallad för *den centrala dogmen*.

Den centrala dogmen består av tre processer: *replikation*, *transkription* och *translation*. Replikation innebär att DNA kopierar sig själv och är en process som sker vid celldelning. Transkription är en process där DNA översätts till mRNA där icke-kodande områden är grovt bortsållade. DNA- och RNA-strängar har en 5'-ända och en 3'-ända som bestämmer vilken riktning transkriptionen går åt. mRNA:t avläses tre baspar i taget från 3'- till 5'-ändan av så kallade *kodon* där specifika basparskombinationer motsvarar en viss aminosyra. Aminosyrorna länkas samman och formar en större struktur. Denna struktur bildar i sin tur ett protein. Hela processen kallas translation.

Då ett mRNA translateras startar processen vid *startkodon* och slutar vid *stopkodon* vilket repeteras över hela mRNA-strängen. Vilken typ av protein som bildas vid translation beror alltså på längden och ordningen av baser mellan dessa start- och stoppkodon och de DNA-sekvenser de motsvarar. På så sätt är DNA uppdelad i bitar som kodar för olika proteiner. Dessa bitar kallas för gener, och summan av dem i en organism kallas för *genom* [10].



Figur 3: Bilden visar kedjornas uppbyggnad av nukleotider samt skillnaden mellan DNA och RNA. DNA är dubbelsträngad och har kvävebaserna 'A', 'T', 'G' och 'C' medan RNA är enkelsträngad och har baserna 'A', 'U', 'G' och 'C'. Bild av: Deimante Neimantaite.

Processerna i informationsflödet sker inte alltid felfritt. Ett fel vid till exempel replikation kan vara att en *mutation* har uppstått och att genomet förändras. Sådana fel kan oftast upptäckas och repareras av cellen [8]. Om felet inte repareras leder det ofta till olika typer av skador men en mutation kan också vara positiv för organismen och är nödvändiga för att evolutionen ska ske [11].

2.1.2 Evolution och horisontell genöverföring

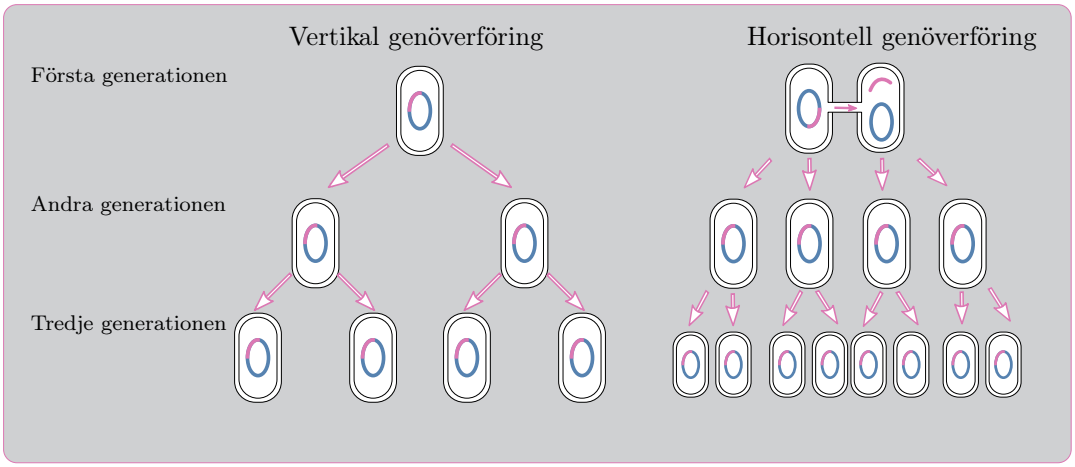
För all evolution finns en drivkraft för vilka förändringar i genomet som stannar kvar över generationerna. Förändringar som bidrar till uttryck för funktionella proteiner har större chans att stanna i genomet. Dessa drivkrafter kan vara ett selektionstryck av ett externt stimuli som gör att endast vissa bättre anpassade individer överlever och därmed för vidare artens gener [12]. Ett exempel på ett sådant selektionstryck kan vara bakterier som utsätts för antibiotika, vilket leder till utveckling av antibiotikaresistens eftersom bara bakterier med egenskapen överlever.

Evolutionen brukar beskrivas som gener som ärvs ner och förändras över generationer, vilket kan beskrivas som *vertikal genöverföring*. Detta beskriver dock inte hela bilden. Vissa DNA-sekvenser, även kallade *transposabla element*, kan "hoppa" inom genomet och mellan olika organismer. Dessa element kan ha olika storlek. Från enskilda gener på cirka 300 baspar till *genomiska öar* som kan vara uppemot tusentals baspar [13]. I detta arbete undersöks spridning av gener mellan organismer. Flyttas en gen mellan olika organismer kallas det *horisontell genöverföring* och genen kallas informellt en *hoppande gen*, se Figur 4.

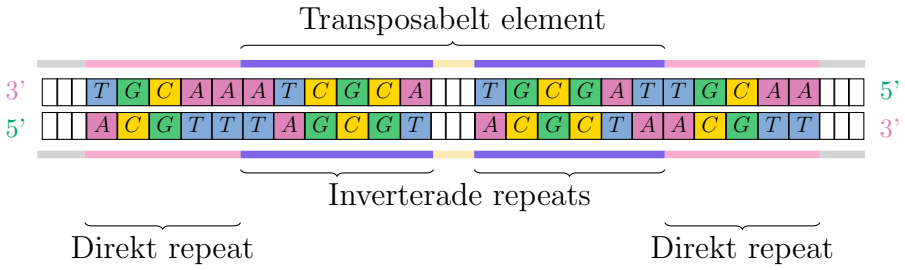
Det allra vanligaste är att en gen hoppar till olika positioner inom det egna genomet. Upp till 50% av det mänskliga genomet kan kopplas till hoppande gener [14]. Ompositionering av en individs gener är dock inte av intresse när spridning mellan arter undersöks. Det är på grund av hoppande gener mellan olika arter som spridningen av antibiotika har skett så snabbt. Det är därför av intresse att undersöka dessa på ett effektivt sätt [14].

Ett transposabelt element kännetecknas av dess uppbyggnad. På ändarna finns så kallade *inverterade repeats*, som är komplementerande segment till varandra. Sekvensen omges av *direkta repeats* som är kopior av varandra. Detta illustreras i Figur 5. Dessa korta segment är alltså inte en del av den hoppande genen, utan stannar kvar i den ursprungliga positionen vid flytt och blir på så sätt ett fotspår av var genen en gång har befunnit sig.

Den vanligaste mekanismen för flytten är så kallad *klippa- och klistra-mekanism* [14]. Specifika enzymer kallade *transposaser*, som många hoppande gener i DNA själva kodar för, gör specifika brott i DNA-strängen som frigör genen. Vid flytt repareras brottet i DNA-strängen och gör den hel igen genom att sätta ihop de två direkta repeats som flankerade



Figur 4: En schematisk bild över vertikal genöverföring (celldelning) och horisontell genöverföring. I vertikal genöverföring överförs gener till nya bakterier över generationer medan i horisontell överföring kan en gen spridas till en annan bakterie under samma generation vilket resulterar i snabbare spridning. Bild av: Emily Curry.



Figur 5: Grafisk överblick som beskriver en hoppande gen. Figuren är utformad med inspiration från [14]. Bild av: Deimante Neimantaite.

genen. Den extraherade genen färdas inom cellen, eller hoppar till en annan cell och hittar en ny position där den gör ett nytt brott i den nya målsekvensen och integreras.

Förutom att hoppande gener kan ta med sig egenskaper till den nya bakterien kan de även ge upphov till andra resultat. I vissa fall händer ingenting alls, eftersom allt DNA inte kodar för några funktioner. Dessutom sker reglering av proteinsyntes och proteiners funktion även efter transkriptionen. Ett exempel på detta är *post-transkriptionell modifiering*. Detta innebär att förändringar i genomet kan modifieras senare i processen och samma protein kan bildas som det ursprungligen hade gjort.En annan effekt som kan uppstå, både inom den egna organismen och i andra, är att ursprungliga gener kan bli tystade då hoppande DNA-segment kan implementeras i en annan gensekvens och därmed förstöra den kodande delen.

2.1.3 Antibiotikaresistens och dess mekanismer

En bakterie som utvecklat antibiotikaresistens har erhållit en gen genom mutationer, arv eller horisontell genöverföring som kodar för protein vilka oskadliggör effekten av antibiotika. Detta kan ske på olika sätt, huvudsakligen genom att

- antibiotikans transport in i bakteriecellen blockeras
- ämnen som bryter ned antibiotikan skapas
- antibiotikan pumpas ut ur bakteriecellen innan det verkar
- ämnen som hindrar att antibiotika fäster på bakteriecellen skapas

- målet för antibiotikan modifieras så att antibiotikan blir verkningslös

Om en bakterie har utvecklat resistens mot flera olika sorters antibiotika kallas den för *multiresistent*. Detta gör bakterien svårbehandlad och utgör ett hot mot sjukvården, som till stora delar beror på antibiotikans funktionalitet [15]. Exempel på antiobiotika som bakterier ofta och lätt kan utveckla resistens mot är fluoroquinoloner och beta-laktamer, där bland annat penicillin ingår. Fluoroquinoloner verkar genom att förhindra att bakteriellt DNA replikeras och beta-laktam förhindrar uppbyggnaden av bakteriers cellväggar [16], [17]. Resistens gör att dessa mekanismer inte fungerar.

2.2 Matematisk bakgrund

I följande avsnitt presenteras den matematiska teori som krävs för arbetet. Här presenteras bland annat Markovkedjor som den framtagna modellen använder. Dessutom beskrivs olika metoder och analyser som kan användas för att utvärdera hur väl modellen presterar.

2.2.1 Stokastiska processer och Markovkedjor

En *stokastisk process* är en matematisk modell av en process som utvecklas slumpmässigt i tiden. Det är alltså en samling av slumpvariabler $\{X(t), t \in T\}$ där tidsmängden T kan vara diskret eller kontinuerlig, vilket innebär att processen kan fortskrida i diskreta tidssteg eller kontinuerligt över ett intervall. Om $X(t_n) = i$ innebär det att processen har tillståndet i vid tiden t_n.

Alla möjliga värden $X(t_n)$ kan anta utgör det så kallade *tillståndsrummet* S. Även S kan vara diskret eller kontinuerligt. Om tillståndsrummet av en stokastisk process är diskret kallas processen ofta för en *kedja*. Till exempel kallas en diskret Markovprocess för en *Markovkedja* [18].

En Markovprocess är en stokastisk process som är minneslös vilket innebär att föregående tillstånd och nästkommande är oberoende, givet det nuvarande tillståndet (kan vara ett eller flera). Detta kallas för *Markovegenskapen*. För en Markovkedja av första ordning, det vill säga då nästa tillstånd endast beror på det tillstånd processen befinner sig i, beskrivs Markovegenskapen matematiskt som

$$Pr\{X_{n+1} = x_{n+1} | X_1 = x_1, \ldots, X_n = x_t\} = Pr\{X_{n+1} = x_{n+1} | X_n = x_n\},$$

där X_n är en annan notation för $X(t_n)$.

Om sannolikheten är oberoende av n, det vill säga om $Pr\{X_{n+1} = j | X_n = i\} = Pr\{X_2 = j | X_1 = i\}$, är kedjan *homogen* [19]. Uttryckligen betyder det att en händelse är lika sannolik att inträffa oavsett var i tiden processen befinner sig.

Utvecklingen av en Markovkedja bestäms av den *intitiala sannolikhetsfördelningen* och av sannolikheterna för alla möjliga övergångar mellan två tillstånd i tillståndsrummet. Det första steget bestäms av den initiala sannolikhetsfördelningen $\pi = [\pi_1, \pi_2, \ldots \pi_{|S|}]$. Det betyder att sannolikheten att första tillståndet i processen är i är $Pr\{X_1 = i\} = \pi_i$, där i är ett tillstånd i S. Sannolikheten att sedan gå från tillstånd i till j på ett steg skrivs som

$$Pr\{X_2 = j | X_1 = i\}.$$

En övergång mellan tillstånd i och j definieras som $p_{ij} = Pr\{X_{n+1} = j | X_n = i\}$ [19]. Alla möjliga övergångar kan sammanfattas i en så kallad *övergångsmatris* P som har formen

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & \cdots & p_{1|S|} \\ p_{21} & p_{22} & p_{23} & \cdots & p_{2|S|} \\ p_{31} & p_{32} & p_{33} & \cdots & p_{3|S|} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{|S|1} & p_{|S|2} & p_{|S|3} & \cdots & p_{|S||S|} \end{bmatrix}.$$

Eftersom P består av sannolikheter har matrisen bara positiva element och varje rad summerar till 1.

För en homogen Markovkedja beräknas den initiala sannolikheten som antalet gynnsamma tillstånd dividerat med antalet möjliga, det vill säga

$$\pi_i = \frac{m_i}{\sum_i m_i}, \tag{1}$$

där m_i är antalet förekomster av tillstånd $i \in S$, och elementen p_{ij} beräknas enligt

$$p_{ij} = \frac{n_{ij}}{\sum_j n_{ij}}, \tag{2}$$

där n_{ij} är antalet övergångar i kedjan från i till j och $i, j \in S$. Det fullständiga beviset för (2) finns i Appendix A.1.

Markovkedjor kan utökas till högre ordningar vilket innebär att processens utveckling är beroende av det nuvarande tillståndet och flera tidigare, det vill säga det nuvarande tillståndet utökas till att gälla flera. Till exempel beror en Markovkedja av ordning två på de två tidigare tillstånden, en kedja av tredje ordningen beror på de tre tidigare tillstånden och så vidare. Övergångsmatrisen för andra ordningens kedja kommer alltså ha element av formen

$$p^2_{ijk} = Pr^2\{X_n = k | X_{n-1} = j, X_{n-2} = i\}$$

och elementen för tredje ordningen kommer vara

$$p^3_{hijk} = Pr^3\{X_n = k | X_{n-1} = j, X_{n-2} = i, X_{n-3} = h\}.$$

[20]

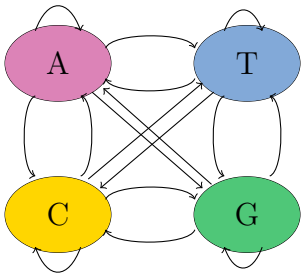
Genom att utöka tillståndsrummet kan en högre ordningens Markovkedja betraktas som en första ordningens kedja där tillståndsrummet S^k ges av alla möjliga kombinationer av k element från S^1. Se Appendix A.1 för bevis.

2.2.2 Markovmodell för DNA-sekvenser

Ett tillämpningsområde för Markovkedjor inom biologi är en DNA-sekvens där de fyra nukleotiderna utgör tillståndsrummet. Genom att anta att DNA-sekvenser uppfyller Markovegenskapen, går det att modellera en DNA-sekvens som en Markovkedja och därefter beräkna sannolikheten för en viss följd av nukleotider.

För en Markovkedja av första ordningen består tillståndsrummet av de olika nukleotiderna som finns i en DNA-sekvens, det vill säga $S^1 = \{\text{'A'}, \text{'T'}, \text{'G'}, \text{'C'}\}$. Varje steg i tiden beskriver då övergången från en bas till en annan, se Figur 6. DNA-sekvensen kan ses som en följd av slumpvariabler $X_1X_2X_3 \dots X_n$ där varje $X_i \in S^1$, $i = 1, 2, \dots, n$. Markovegenskapen ger att denna övergång endast beror på den tidigare basen och inte på hela sekvensen. Exempelvis blir sannolikheten att sekvensen 'TGAC' följs av ett 'A'

$$Pr\{X_5 = \text{'A'} | X_4 = \text{'C'}, X_3 = \text{'A'}, X_2 = \text{'G'}, X_1 = \text{'T'}\} = Pr\{X_5 = \text{'A'} | X_4 = \text{'C'}\}.$$



Figur 6: Tillståndsrummet och de möjliga övergångarna för DNA-modellen (Markovkedjan) av första ordningen. Bild av: Deimante Neimantaite.

Den initiala sannolikhetsfördelningen för en nukleotid, $\pi = \{\pi_{\mathrm{A}}, \pi_{\mathrm{T}}, \pi_{\mathrm{G}}, \pi_{\mathrm{C}}\}$, beräknas som kvoten mellan antalet förekomster av en viss nukleotid och det totala antalet nukleotider i sekvensen. Till exempel beräknas den intiala sannolikheten för 'A' enligt

$$\pi_{\mathrm{A}} = \frac{m_{\mathrm{A}}}{\sum_{i \in S} m_i},$$

där m_i är antalet förekomster av tillstånd $i \in S$.

Övergångssannolikheterna tas fram enligt Ekvation (2) genom att räkna antalet förekomster av en specifik övergång (exempelvis från tillstånd 'C' till 'A') och dividera med totala antalet övergångar från urpsrungstillståndet till alla möjliga tillstånd (från tillstånd 'C' till 'A', 'T', 'G' och 'C'). Övergångsmatrisen för DNA-modellen av första ordningen har då utseendet

$$P = \begin{bmatrix} p_{\mathrm{AA}} & p_{\mathrm{AT}} & p_{\mathrm{AG}} & p_{\mathrm{AC}} \\ p_{\mathrm{TA}} & p_{\mathrm{TT}} & p_{\mathrm{TG}} & p_{\mathrm{TC}} \\ p_{\mathrm{GA}} & p_{\mathrm{GT}} & p_{\mathrm{GG}} & p_{\mathrm{GC}} \\ p_{\mathrm{CA}} & p_{\mathrm{CT}} & p_{\mathrm{CG}} & p_{\mathrm{CC}} \end{bmatrix}.$$

Sannolikheten för en delsekvens beräknas med hjälp av den initiala sannolikhetsvektorn och övergångsmatrisen. För det tidigare givna exemplet 'TGAC' beräknas sannolikheten enligt följande

$$Pr\{\text{'TGAC'}\} = \pi_{\mathrm{T}} p_{\mathrm{TG}} p_{\mathrm{GA}} p_{\mathrm{AC}}.$$

Eftersom sannolikheten för långa sekvenser blir låg ($<< 1$) är det lämpligt att utgå från det logaritmerade värdet av sannolikheten istället (*"log-likelihood"*).

För en Markovkedja av ordning k beror nästkommande tillstånd på det nuvarande och $k-1$ tidigare tillstånden. Sannolikheten i det tidigare exemplet med sekvensen 'TGAC' som följs av ett 'A' blir för en andra ordningens Markovkedja

$$Pr\{X_5 = \text{'A'} | X_4 = \text{'C'}, X_3 = \text{'A'}\}.$$

När modellen utökas till Markovkedjor av högre ordning kan, som tidigare nämnt, ett modifierat tillståndsrum betraktas. Tillståndsrummet S^k blir då mängden av kombinationer av k nukleotider ur S^1, där k är ordningen på Markovkedjan. Exempelvis blir tillståndsrummet för andra ordningens Markovkedja

$$S^2 = \{\text{'AA'}, \text{'AT'}, \text{'AG'}, \text{'AC'}, \ldots, \text{'CA'}, \text{'CT'}, \text{'CG'}, \text{'CC'}\},$$

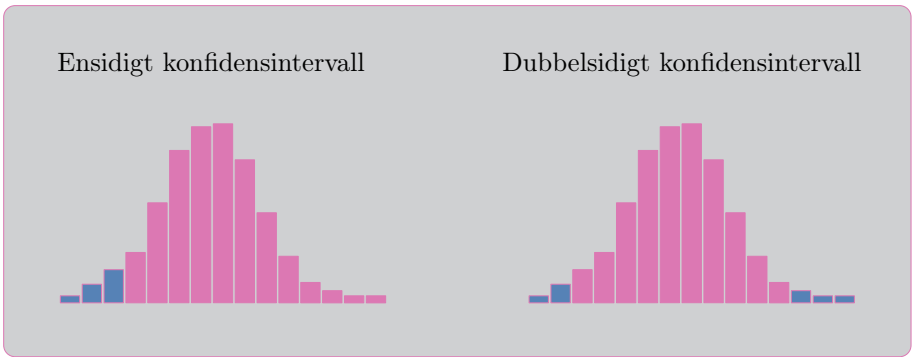
alltså ett rum med $4^2 = 16$ tillstånd som beskriver två nukleotider. Detta medför att de initiala sannolikheterna och övergångsmatrisen kan beräknas på samma sätt som för första ordningens modell.

2.2.3 Konfidensintervall

En *konfidensmängd* är en mängd som med given sannolikhet innehåller en skattad parameter. Om mängden är ett intervall kallas det för ett *konfidensintervall*. Detta kan skrivas som $Pr\{\theta \in B\} = \beta$. Sannolikheten för att parametern θ tillhör mängden B ges alltså av β som även kallas för skattningens *konfidensgrad*. Är sannolikhetsfördelningen för parametern känd kan B beräknas och beror då av β. Om sannolikhetsfördelningen inte är känd kan andra metoder användas, till exempel *Bootstrap* som förklaras nedan. Mängden B kallas för en $100 \cdot \beta-$procentig konfidensmängd. Om B kan skrivas som $[c, \infty)$ alternativt $(-\infty, c]$, för någon konstant c, kallas B för ett ensidigt konfidensintervall. Om B kan skrivas som $[a, b]$, för något a och b, kallas konfidensintervallet tvåsidigt, båda fallen illustreras i Figur 7 [21].

2.2.4 Parametrisk Bootstrap

Bootstrap är en teknik som är användbar för att skatta parametrar med given noggrannhet, bland annat om fördelningen är komplicerad att beräkna. *Parametrisk Bootstrap* innebär att stickprov simuleras från någon modell där parametrar tidigare skattats. Därefter kan vald parameter skattas från dessa stickprov. Genom att simulera ett stort antal stickprov



Figur 7: Ett ensidigt och dubbelsidigt konfidensintervall. Skillnaden är att i ett ensidigt konfidensintervall avskiljs endast de lägsta värdena medan i ett dubbelsidigt avskiljs både de lägsta och högsta värdena. Bild av: Emily Curry.

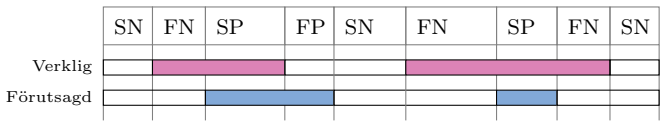
och tillämpa centrala gränsvärdessatsen, som säger att en oändlig summa av likafördelade oberoende stokastiska variabler är approximativt normalfördelade, kan en approximativ konfidensmängd för vald parameter anges. Idén bakom Bootstrap är enkel och en av fördelarna är att den är oberoende av vilken fördelning stickproven har, så länge det finns någon metod för att simulera stickprov [22].

Fördelningen som är av intresse i denna rapport ges av sannolikhetsvärdet för alla sekvenser av längd k som kan uppstå när en Markovkedja simuleras. I en DNA-sekvens finns 4 möjliga nukleotider och därför 4^k olika sekvenser. Antalet möjliga utfall för sannolikhetsvärdet är därför stort och att beräkna den exakta fördelningen är inte ett praktiskt genomförbart alternativ. På grund av att den exakta fördelningen inte kan beräknas kan istället ett värde för vald parameter skattas med parametrisk bootstrap.

2.2.5 Sensitivitet och specificitet

Sensitivitet och specificitet är statistiska mått på hur pålitlig en modell är genom att testa dess klassifikationsförmåga. Klassifikation innebär att datamängden delas in med avseende på vissa egenskaper [23]. Sensitivitet och specificitet antar värden mellan 0 och 1, där en bra modell har värden nära 1.

Sensitivitet mäter andelen sanna positiva resultat, medan *specificitet* andelen sanna negativa resultat. Genom att till exempel modellera gener som inte tillhör det egna genomet (hoppande gener) kommer ett sant positivt resultat innebära att modellen klassificerar en gen som är en hoppande gen som positiv. Ett positivt utslag för en gen som tillhör det egna genomet skulle resultera i ett falskt positivt (FP) resultat. Figur 8 nedan visar ytterligare ett exempel på sanna och falska positiva resultat, och även sanna och falska negativa (SN respektive FN) resultat.



Figur 8: Ett exempel på sanna och falska, positiva eller negativa resultat. Bild av: Deimante Neimantaite.

I detta arbete klassificeras gener som hoppande gener eller nativa gener. Sensitivitet och specificitet beräknas med hjälp av formlerna nedan [24], [25]. Sn står för sensitivitet och $Sp*$ för specificitet.

$$Sn = \frac{SP}{SP + FN} \tag{3}$$

och

$$Sp* = \frac{SN}{SN + FP}. \tag{4}$$

Vid genundersökningar kommer dock värdet av $Sp*$ bli missvisande, då $SN >> FP$ för gener, och därmed blir det en orättvis jämförelse. Därför kan specificitet även beräknas enligt

$$Sp = \frac{SP}{SP + FP} \tag{5}$$

och mäter då andelen förutsagda positiva resultat som faktiskt är positiva. Då däremot sensitivitet mäter andelen positiva resultat förutsagda som positiva.

2.2.6 Statistisk hypotesprövning

Statistiska tester används för att undersöka *hypoteser* (antaganden) för en mängd data. Ett exempel på en sådan hypotes kan vara att anta ett oberoende bland elementen i datamängden.

Det finns två olika typer av hypoteser inom statistik: *nollhypotes* (H_0) och *alternativ hypotes* (H_a) [26] . Den förstnämnda hypotesen behandlar vanligtvis den önskade, men mer osannolika egenskapen hos en datamängd medan den alternativa hypotesen oftast är en motsägelse till nollhypotesen. En hypotesprövning talar om ifall nollhypotesen kan avfärdas eller inte. Alltså ges ingen bekräftelse på nollhypotesen utan endast om den är felaktig.

Vid hypotesprövning används *signifikansnivån* α som ett mått på dess sensitivitet. Mer specifikt mäter signifikansnivån risken att felaktigt förkasta nollhypotesen. Vilken typ av prövning eller test som används beror på nollhypotesen, och utifrån det beräknas ett värde för prövningen som kallas för ett *provutfall*. Provutfallet i sin tur används för beslutstagande angående nollhypotesen [26].

2.2.7 Analys av korstabell med χ^2-oberoendetest

Ett χ^2-test för oberoende testar om elementen i en datamängd är oberoende av varandra. Nollhypotesen och den alternativa hypotesen för testet är

$$H_0 : \text{Elementen är oberoende av varandra}$$
$$H_a : \text{Det finns ett samband, elementen är beroende.}$$

För χ^2-oberoendetestet krävs en kategoriserad datamängd, vilken kan fås med hjälp av en *korstabell* [27]. Korstabeller är ett enkelt och användbart verktyg för att jämföra samband mellan olika kategoriska variabler. Dessa variabler klassificeras som kategoriska då de kan delas in i olika kategorier. Elementen i korstabellen utgörs av antal observationer för varje kategori av variabler. [28] I Tabell 2 illustreras ett exempel på hur en korstabell för två variabler kan se ut

Tabell 2: Exempel på hur en korstabell ser ut.

Variabel B / Variabel A	Kategori 1	Kategori 2	Total
Kategori 1	N_{11}	N_{12}	$N_{1.}$
⋮	⋮	⋮	⋮
Kategori h	N_{h1}	N_{h2}	$N_{h.}$
Total	$N_{.1}$	$N_{.2}$	n

där

$$N_{i.} = \sum_j N_{ij} \quad \text{radsumma,}$$
$$N_{.j} = \sum_i N_{ij} \quad \text{kolonnsumma,}$$
$$n = \sum_i N_{i.} = \sum_j N_{j.} \quad \text{totala antalet observationer.}$$

Testets provutfall Q beräknas på följande sätt

$$Q = \sum_{i=1}^{k} \sum_{j=1}^{l} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}, \tag{6}$$

där O_{ij} och E_{ij} motsvarar de observerade respektive förväntade värdena. I fallet med korstabellen i Tabell 2 beräknas dessa som $O_{ij} = N_{ij}$ respektive $E_{ij} = \frac{N_{i.}N_{.j}}{n}$. Uttrycket för det förväntade värdet kommer ifrån att om båda variablerna antas vara oberoende förväntas det finnas samma andel från kategori j som det finns från det totala, det vill säga

$$\frac{N_{ij}}{N_{.j}} = \frac{N_{i.}}{n}.$$

Provutfallet jämför de observerade värdena med de förväntade för att besluta om de skiljer sig så pass mycket att slutsatsen kan dras att det finns ett samband mellan variablerna. För att fastställa ett beslut används *p-värdet*. [29] P-värdet, $Pr\{'X > Q'\}$, är sannolikheten att observera ett mer extremt provutfall än det redan observerade om nollhypotesen är sann. Vid ett χ^2-oberoendetest betyder det sannolikheten att få den beräknade avvikelsen (Q-värdet) eller högre, givet ett oberoende bland elementen. Denna sannolikhet beräknas med hjälp av antal *frihetsgrader*. Med frihetsgrad menas ett mått på hur många parametrar i ett system som kan variera. Antal frihetsgrader i en korstabell är

$$(r-1)(k-1),$$

där r är antalet rader och k antalet kolumner. Nollhypotesen förkastas om p-värdet är mindre än signifikansnivån, α. Detta innebär att det finns associationer mellan variabel A och B. Variablerna är därmed inte oberoende [30].

3 Metod

Detta avsnitt behandlar projektets arbetsgång där teorin som berördes i Avsnitt 2.2.2 ”Markovmodell för DNA-sekvenser” och Avsnitt 2.2.4 ”Parametrisk Bootstrap”, vidareutvecklas för att tillsammans fastställa den slutgiltiga matematiska modellen. Sedan presenteras en redogörelse för hur modellen presterar med hjälp av sensitivitets- och specificitetsanalys följt av en beskrivning av metoden som användes för att klassificera de sekvenser modellen identifierade och hur dessa sekvensers ursprung undersöktes. Avslutningsvis presenteras verifieringen av Markovegenskapen för bakterien *Escherichia coli* stam K12.

Programmeringsspråket Python är valt som ensamt verktyg för samtliga modelleringar och beräkningar efter rekommendation från handledare då det är ett väletablerat språk inom bioinformatik. För fullständig implementering av metoderna se Appendix B.

3.1 Utformning av modellen och implementation i Python

Modellen som utformades i detta projekt hade som främsta uppgift att hitta intervall i en bakteries DNA-sekvens där sannolikhetsvärdena var ovanligt låga. Detta förverkligades genom följande processer:

1. Läsa in FASTA-fil av en bakteries genom
2. Uppskatta parametrar
3. Beräkna sannolikhetsvärden
4. Fastställa konfidensintervall
5. Hitta intervall

Läsa in FASTA-fil av en bakteries genom En bakteries genom laddades ned från *National Center for Biotechnology Information* (NCBI) i form av en FASTA-fil. FASTA-format är ett textbaserat format som ofta används inom bioinformatik för att representera bland annat nukleotidsekvenser. Filen lästes in till en sträng för vidare arbete.

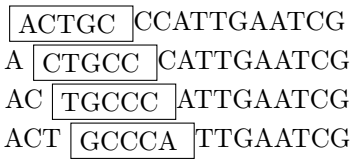
Uppskatta parametrar
DNA-sekvensen från den inlästa datan betraktades, som nämnt tidigare, som en homogen Markovkedja med tillhörande övergångsmatris P och initial sannolikhetsfördelning π. Dessa parametrar skattades genom ekvationerna (1) och (2) genom att räkna antalet förekomster av de fyra nukleotiderna i DNA-sekvensen och antalet övergångar mellan dem. För högre ordningar på Markovkedjan blev risken större att övergångar från och till vissa tillstånd aldrig inträffade. Övergångsmatrisen kunde därmed innehålla nollelement vilket ledde till problem i framtida beräkningar. Genom att addera falska observationer (”pseudocounts”’) löstes problemet. Detta innebar att varje gång det inte förekom en övergång mellan tillstånd i och j lades 1 till. Om till exempel en övergång endast skulle ske en gång i genomet och en falsk observation lades till en övergång som utgår från samma nuvarande tillstånd skulle dessa två övergångar ha lika stora sannolikheter. Därför adderades även 1 till antalet förekomster av alla övergångar från tillstånd i till de övriga för att minska påverkan av de falska observationerna.

Beräkna sannolikhetsvärden
Med den färdiga Markovmodellen kunde nu sannolikhetsvärdena för DNA-sekvenserna i bakterien beräknas. Den totala arvsmassan betraktades i mindre delar av samma längd, så kallade *fönster*. Dessa fönster överlappar varandra på ett sådant sätt att nästkommande fönster börjar en nukleotid efter nuvarande fönstrets start. Denna överlappning illustreras i Figur 9. Sannolikheten att varje fönster inträffar erhölls då genom produkten av den initiala sannolikhetsfördelningen för den första nukleotiden samt parametrarna i övergångsmatrisen för de resterande baserna. Värdet för det första fönstret i Figur 9 beräknades då enligt

$$Pr\{\text{'ACTGC'}\} = \pi_A p_{AC} p_{CT} p_{TG} p_{GC}.$$

Genom att utnyttja det redan uträknade sannolikhetsvärdet för föregående fönster kunde beräkningstiden förkortas avsevärt. De två första faktorerna eliminerades och ersattes med den initiala sannolikheten för fönstrets början samt att sist i uttrycket lades elementet i övergångsmatrisen för fönstrets sista nukleotid till.

Då alla termer i den initiala sannolikhetsfördelningen π och övergångsmatrisen P är mindre än 1 blev sannolikhetsvärdena för fönstrena ohanterligt små. Därför logaritmerades värdena för enklare analys. Det var på grund av detta som ettorna adderades när parametrarna uppskattades då logaritmen av noll är odefinierad.



Figur 9: I figuren illustreras hur fönstren överlappar varandra i en DNA-sekvens. Bild av: Sara Finati.

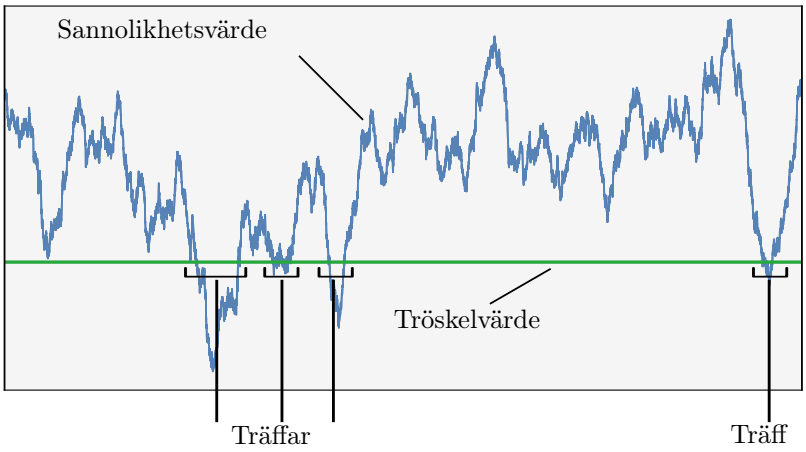
Fastställa ensidigt konfidensintervall

För att sortera ut fönster med låga värden, då låga sannolikhetsvärden antyder på att genen inte tillhör bakteriens egna genom, användes ett ensidigt konfidensintervall. Konfidensgraden som applicerades sattes till 99% då lägre grader gav ohanterligt många träffar. Med Parametrisk Bootstrap som beskrivs i Avsnitt 2.2.4 konstruerades 1 000 nya genom från bakteriens arvsmassa. Genomen skapades genom slumpmässigt valda övergångar från den initiala sannolikhetsfördelningen π och övergångsmatrisen P. För ett genom betraktades 500 påföljande fönster för vilka sannolikheten beräknades. Värdena sorterades i storleksordning varpå storleken motsvarande konfidensgraden särskildes. Detta genomfördes på alla de 1 000 simulerade genomen. Ett genomsnitt av dessa 1 000 särskilda värden beräknades innan det slutgiltiga konfidensintervallet kunde fastställas. Konfidensintervallet kallas i det här arbetet även för tröskelvärde då det utgör gränsen för när DNA-sekvenserna räknas till det ursprungliga genomet eller ska klassas som en hoppande gen. Valet att stanna vid 500 fönster togs i följd av att beräkningstiden för Parametrisk Bootstrap blev oacceptabelt lång för högre antal.

Hitta intervall

Slutligen granskades varje fönsters sannolikhetsvärde i förhållande till det beräknade tröskelvärdet. Värden som befann sig utanför intervallet var de intressanta och klassificerades som *träffar*. En träff definieras som en sammanhängande del av DNA-sekvensen som blivit identifierad av modellen, det vill säga en delsekvens med ett lägre sannolikhetsvärde än tröskelvärdet. Om två träffar låg inom en halv fönsterstorlek mellan varandra slogs de ihop och räknades som en träff. Detta för att minska antalet träffar inom samma område, då sannolikhetsvärdet pendlar mellan att ligga över och under tröskelvärdet. Figur 10 visar ett exempel på en sannolikhetsgraf där den blå grafen motsvarar sannolikhetsvärden för alla fönster av sekvensen. Den horisontella axeln anger positionen i genomet. Varje enskilt sannolikhetsvärde i grafen motsvaras av sannolikheten för det fönster som har sin första nukleotid på den positionen där värdet är placerat. De värden som ligger under tröskelvärdet (den gröna linjen) klassificeras som träffar. Dessa träffar ger då ett intervall i genomet som skulle kunna vara en hoppande gen. Eftersom det sista värdet i intervallet är beräknad för ett fönster framåt, adderades ett fönster till varje träff. De genererade träffarna sparades för vidare utredning.

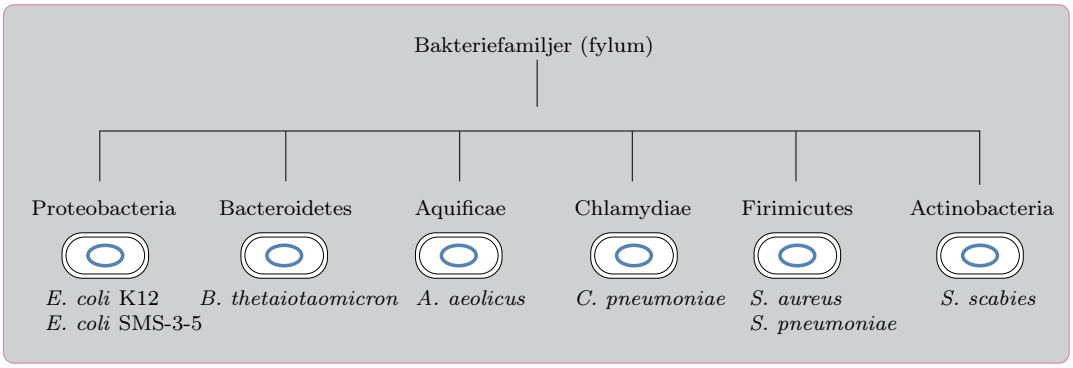
Modellen kördes för flertalet bakterier, olika ordningar på Markovkedjan samt olika fönsterstorlekar för att få en överblick över modellens utveckling. För högre ordningars Markovkedjor utökades tillståndsrummet enligt Avsnitt 2.2.1. Detta gjorde att den beskrivna metoden kunde användas oavsett vald ordning.



Figur 10: Förklaring av hur träffar avläses från sannolikhetsgrafen för en DNA-sekvens. Bild av: Emily Curry.

3.2 Val av bakterier

Eftersom det inte är möjligt att undersöka alla bakteriers genom under den givna tidsramen gjordes ett urval baserat på släkte och egenskaper. Alla olika bakterier härstammar från olika familjer som sedan har differentierat sig vidare, och därför förväntas genomen att skilja sig i basparsfördelning. Följande bakterier valdes att undersökas som även illustreras i Figur 11:



Figur 11: Visar olika grupper av bakteriefamiljer och vilken familj de valda bakterierna tillhör. Bild av: Emily Curry.

- *Escherichia coli*, stammarna K12 och SMS-3-5 - dessa var givna redan från början och var inget aktivt val. *E. coli* används mycket i bioteknik och betraktas som en modellorganism och stammen K12 anses vara normal. Stammen SMS-3-5 är en så kallad superresistent bakterie och är därför av intresse.

- *Bacteroides thetaiotaomicron* - Denna bakterie föreslogs som en första främmande bakterie med avsikt på *E. coli* av handledarna. Bakterien är en av de bakterier som tillhör människomagens naturliga bakterieflora, och familjen *Bacteroides* är en av de vanligaste förekommande bakterierna [31].

- *Aquifex aeolicus* - Detta är en termofil som lever under extrema förhållanden, mellan 85-95 °C i vattenkällor kring vulkaner [32], vilket gör att den är av intresse för industriell bioteknik, då höga temperaturer ofta önskas. Genomet är kort och bakterien härstammar från arkéer (*archea*) vilket gör den extra intressant då den förväntas skilja sig extra mycket.

- *Chlamydophila pneumoniae* - Orsakar lunginflammation och en mängd lungsjukdomar, bland annat TWAR [33]. Det finns inget vaccin för att motverka denna bakterie. Den valdes dels för att den tillhör en intressant sjukdomsframkallande familj och även då den är intressant inom vaccinforskning.
- *Staphylococcus aureus* - Vanligen lever denna bakterie i symbios med människokroppen och orsakar ingen skada. Dock kan den ge upphov till både inflammationer och infektioner. Det finns undergrupper till denna som är multiresistenta mot alla typer av antibiotika [34]. Bakterien valdes då staphylococcer är vanligen förekommande hos människor och är därför intressant ur ett tillämpningsperspektiv.
- *Streptococcus pneumoniae* - Även denna bakterie ger upphov till lunginflammation och en mängd andra lungsjukdomar. Stammen som undersöks är multiresistent, likt SMS-3-5 [35]. Den valdes då den till skillnad från *C. pneumoniae* är en del av streptococcerfamiljen som är en vanligt förekommande sjukdomsframkallande familj.
- *Streptomyces scabiei* - Detta är det hittills längsta genom som har sekvenserats [36]. Bakterien har ingen verkar på människor men har en mängd kända patogena genöar och har en G/C-kvot på 71% [36]. Detta gör att genomet är högst intressant att undersöka.

3.3 Utvärdering av modellen

För att bestämma bästa ordning på Markovkedjan och fönsterstorlek baserat på datan från *E. coli* stam K12 gjordes först en grafisk jämförelse mellan två olika ordningar eller två olika fönsterstorlekar. Dessa ritades i samma figur och på så sätt erhölls en visuell överblick över hur olika värden på parametrarna presterade gentemot varandra. Dessutom utfördes sensitivitets- och specificitetsanalyser (se Avsnitt 2.2.5). Analysens syfte var att dels bestämma ordning och fönsterstorlek, och dels utvärdera modellens tillförlitlighet. Dessutom gjordes en sensitivitets- och specificitetsanalys för att undersöka hur väl modellen identifierar gener av olika längd.

3.3.1 Bestämning av parametrar med sensitivitets- och specificitetsanalys

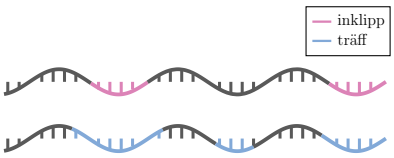
För att kunna utvärdera modellens prestanda för olika ordningar och fönsterstorlekar utfördes en sensitivitets- och specificitetsanalys på simulerade genom med hoppande gener på kända positioner. Analysen utfördes i följande steg:

1. Ett simulerat genom skapades med 100 hoppande gener.
2. Sannolikhetsvärden beräknades för fönster av sekvensen.
3. Tröskelvärde beräknades för den undersökta ordningen och fönsterstorleken.
4. Träffar identifierades genom att jämföra sannolikhetsvärdena med tröskelvärdet.
5. Träffarna klassificerades som sanna positiva eller falska positiva.

I det här sammanhanget definieras ett *inklipp* som en sammanhängande del av sekvensen som modellen ska identifiera. Träffar definieras på samma sätt som tidigare, det vill säga som en delsekvens vars sannolikhetsvärde är lägre än tröskelvärdet. Även här slogs två träffar ihop om de befann sig närmare varandra än ett halvt fönster. I Figur 12 illustreras inklipp respektive träffar i två strängar av samma sekvens av ett simulerat genom. Tabell 3 sammanfattar hur sanna positiva *SP*, falska positiva *FP*, falska negativa *FN* och sanna negativa *SN* definieras utifrån begreppen träff och inklipp. Från tabellen går det till exempel att utläsa att en träff som överensstämmer med ett inklipp är en sann positiv.

Tabell 3: Definition av *SP*, *FP*, *FN* och *SN*.

	Träff	Ej träff
Inklipp	*SP*	*FN*
Ej inklipp	*FP*	*SN*



Figur 12: Ett exempel på simulerat genom, där de två strängarna motsvarar samma sekvens av genomet, som visar inklippens position och vilka delar modellen identifierar som träffar. Den första och tredje träffen är sanna positiva och den andra träffen är en falsk positiv. Bild av: Deimante Neimantaite.

Den simulerade DNA-sekvensen skapades genom att med hjälp av övergångsmatrisen P och den initiala sannolikhetsfördelningen π för en viss ordning av *E. coli* stam K12 simulera en sekvens på 10 000 nukleotider. Detta gjordes på samma sätt som vid Parametrisk Bootstrap då övergångar valdes slumpmässigt utifrån övergångsmatrisen. Simulerade sekvenser sattes sedan omväxlande ihop med 100 delsekvenser från *B. thetaiotaomicron* av en storlek på 1 000 nukleotider. På så sätt skapades en DNA-sekvens där de 100 inklippen från *B. thetaiotaomicron* motsvarade 100 hoppande gener.

Tröskelvärdet beräknades med Parametrisk Bootstrap (se "Fastställa konfidensintervall" i Avsnitt 3.1) för den ordning och den fönsterstorlek som undersöktes. Sedan beräknades sannolikhetsvärden för delsekvenser av fönsterstorlek w på samma sätt som tidigare med P och π. Dessa värden jämfördes sedan med tröskelvärdet för motsvarande ordning och fönsterstorlek. På så sätt erhölls intervall för de sekvenser som modellen identifierade som eventuella hoppande gener.

Sensitivitets- och specificitetsanalysen utfördes på två nivåer: *träffnivå* och *nukleotidnivå*. På träffnivå jämfördes träffar direkt med inklippen genom att undersöka om träffen täckte ett helt inklipp. På nukleotidnivå jämfördes istället varje enskild identifierad nukleotid med nukleotider från inklipp.

Träffnivå

Låt x_1 vara startpostionen och x_2 slutpositionen av ett inklipp och y_1, y_2 vara första respektive sista positionen av en träff, där fönsterstorleken w har lagts till y_2 eftersom ett sannolikhetsvärdet beräknas på w nukleotider framåt. För att träffen ska räknas som en sann positiv och vara en korrekt identifikation måste träffen helt täcka ett inklipp, det vill säga

$$y_1 \leq x_1 \text{ och } y_2 \geq x_2$$

måste vara uppfyllt. Dessutom får träffen inte vara förskjuten mer än högst ett fönster före eller efter inklippet, vilket innebär att

$$x_1 - y_1 \leq w \text{ och } y_2 - x_2 \leq w.$$

Detta innebar att varje träff som inte överlappade ett inklipp, var kraftigt förskjutet eller endast täckte en del av ett inklipp klassificerades som en *FP*.

Nukleotidnivå

Om x_1 istället benämner positionen av en enskild nukleotid som undersöks, så är x_1 en *SP* om den befinner sig inuti ett inklipp och dessutom har blivit identifierad av modellen som en träff. Om x_1 är en träff men inte befinner sig i ett inklipp är det en *FP*.

Sensitiviteten Sn beräknades sedan enligt Ekvation (3) och specificiteterna Sp och $Sp*$ enligt Ekvation (4) och (5). För $Sp*$ skiljer sig analysens metod i steg 1, där det simulerade genomet istället skapades med 100 egna gener, inklippta från K12:s genom. Om modellen identifierar ett sådant inklipp klassificeras det som en *FP*. Ett inklipp som inte identifieras är en *SN*. $Sp*$ beräknas alltså utifrån *FP* och *SN* istället för *SP* och *FP*.

För att bestämma ordning genomfördes analys på träffnivå och nukleotidnivå för ordningar $1-7$ med fix fönsterstorlek på $w = 500$. När ordning valts genomfördes samma analys för fönsterstorlekar $w = 100, 200, \ldots, 2000$.

Eftersom modellen ska hitta resistensgener är det lämpligt att undersöka modellens prestanda utifrån inklippslängder som motsvarar längden på dessa gener. Gener som kodar för resistens mot floroquinoloner är typiskt mellan 214-218 aminosyror långa, det vill säga 642-654 nukleotider långa [37] och de flesta gener som kodar för resistens mot betalaktamaser är ungefär 287-291 aminosyror långa vilket motsvarar 861-873 nukleotider [38]. Därför utfördes analysen för den valda ordningen och de fönsterstorlekar som presterade bäst för olika storlekar på inklipp (storlekar 600, 700, ..., 1 300). Storlekarna 600-900 testades för att undersöka hur väl modellen hittar resistansgener men eftersom modellen ska hitta alla typer av hoppande gener undersöktes även storlekarna $1\,000 - 1\,300$.

3.4 Utvärdering av träffar

För att avgöra om modellens träffar gav faktiska träffar på hoppande gener, och i bästa fall hoppande gener som kodar för antibiotikaresistens, undersöktes vissa av de träffar modellen gav upphov till.

I arbetets inledningsfas valdes sekvenser för vidare analys ut slumpmässigt bland träffarna som modellen indikerade var hoppande gener. Dessa sekvenser jämfördes sedan med BLASTx mot NCBI:s nr-gendatabas för att hitta de proteiner som generna eventuellt kodar för och sedan gensekvensen.

BLAST matchar olika sekvenser med varandra. BLASTx translaterar först input-sekvensen till en aminosyrasekvens och jämför sedan denna mot alla protein i vald databas. Databasen som användes i detta arbete, nr, står för non-redundant database. BLASTn matchar input-sekvensen mot andra gensekvenser från vald databas.

När NCBI gav många blast-träffar valdes den träff som gav bäst resultat, det vill säga den träff som stämde bäst överens med den undersökta sekvensen baserat på hur stor del av sekvensen den täckte upp och hur många av sekvensens nukleotider som stämde överens. Från NCBI:s resultat drogs slutsatser om blast-träffarna, då det går att utläsa mycket information om genen och proteinet den kodar för.

Den slutliga metoden att välja intervall baserades på två olika tillvägagångssätt, *parvis klassificering* och *korsklassificering*. Dessa fungerade som en filtrering av alla de träffar som modellen identifierat utifrån avvikande fördelning från det undersökta genomet, med syfte att välja ut de hoppande generna.

Parvis klassificering grundar sig på att hitta områden som indikerar på en hoppande gen utifrån fler än endast en modell, baserat på att olika gener har olika sannolikheter i olika genom. Dels undersöks vart dessa gener finns och dels vad de kodar för. Korsklassificering undersökte SMS-3-5-träffarnas ursprung baserat på ett hypotestest där nollhypotesen var att träffarna hade sitt ursprung från SMS-3-5 och den alternativa hypotesen är att de inte hade det.

3.4.1 Parvis klassificering

Metoden baserades på att hitta alla intervall som tyder på en hoppande gen i både SMS-3-5 och minst en ytterligare bakterie. För att uppnå detta jämfördes alla träffar från de olika bakterierna med BLASTn genom att använda Matematiska Vetenskapers bioinformatikserver *Athena* och dess inbyggda blast-funktion. Dels jämfördes träffar från SMS-3-5 mot träffar i de andra bakterier, men även träffar från SMS-3-5 mot hela genomen hos de andra arterna. Detta för att se om de förväntade hoppande generna från *E. coli*-modellerna fanns i olika organismer och om de indikerades att vara hoppande gener utifrån flertalet modeller. Skulle det visa sig att en gen finns i två bakterier, men bara anses vara hoppande i SMS-3-5 indikerar det att den har sitt ursprung i den andra då dess sannolikhet i den bakterien hade varit tillräckligt hög för att klassas som nativ.

De områden som hade träffar i fler än en modell undersöktes vidare på NCBI likt processen som skedde inledningsvis.

3.4.2 Undersökning av ursprung med korsklassifiering

Varje träff som modellen gav för SMS-3-5 undersöktes med avseende på övergångsmatrisen som skattats från en annan bakteries genom. Detta för att undersöka huruvida träffen i SMS-

3-5 skulle kunna härstamma från den andra bakterien. Metoden liknar i stora drag den som användes för att sortera ut träffarna i SMS-3-5. Genom att simulera sannolikhetsvärden från övergångsmatrisen kan ett nytt tröskelvärde skattas för fördelningen hos en annan bakterie. Sannolikhetsvärdet för träffen med avseende på detta genomets övergångsmatris beräknades sedan och jämfördes mot tröskelvärdet för att avgöra om träffen rimligtvis härstammar från denna bakterie. Nollhypotesen var att sekvensen i träffen inte härstammar från den stam som testas mot. Den alternativa hypotesen var då att sekvensen härstammar från denna stam. Om sannolikhetsvärdet var tillräckligt högt förkastades nollhypotesen och träffen ansågs ha sitt ursprung i denna stam. Den sekvens som motsvarades av träffen sparades undan för vidare undersökning på NCBI likt utförandet som skedde inledningsvis.

3.5 Verifikation av Markovegenskap för K12

Verifieringen av Markovegenskapen baserades på Monika Skuriat-Olechnowska rapport *Statistical Inference for Markov Chains with interval censoring* [39], där liknande undersökning har genomförts.

För att bekräfta antagandet om att DNA-sekvenser kan ses som Markovkedjor behövde Markovegenskapen, det vill säga att sannolikheten att gå från nuvarande tillstånd till nästa är oberoende av det föregående, bekräftas. Verifieringen utfördes endast på K12:s genom.

Undersökningen gjordes genom att analysera förekomsten av sekvenser bestående av de tre tillstånden: föregående, nuvarande, nästkommande. Sekvenser med samma övergång från nuvarande till nästkommande tillstånd men från olika föregående, räknades för att fastställa om det fanns en skillnad i antal förekomster beroende på föregående tillstånd. Fanns ingen signifikant skillnad i frekvenserna för de räknade sekvenserna, indikerade detta på att Markovegenskapen höll.

Till detta test användes följande tre begrepp:

- STS: Tre-tillståndssekvens, innefattande föregående, nuvarande och nästkommande tillstånd.
- SSO: Antalet gånger en specifik STS förekom i DNA-sekvensen.
- TSO: Antalet gånger en specifik två-tillstånds sekvens påträffades. Den inkluderades av tillstånden: föregående och nuvarande.

För en första ordningens Markovkedja blir STS tillståndsrummet för en Markovkedja av ordning tre. Alltså

$$S^3 = \{'AAA', 'AAT', 'AAG', 'AAC', \ldots, 'CCA', 'CCT', 'CCG', 'CCC'\}.$$

Två-tillståndssekvenserna representeras på liknande sätt av tillståndsrummet för ordning två

$$S^2 = \{'AA', 'AT', 'AG', 'AC', \ldots, 'CA', 'CT', 'CG', 'CC'\}.$$

Den relativa frekvensen att varje STS inträffade beräknades då enligt kvoten

$$\frac{SSO}{TSO}.$$

Detta värde jämfördes sedan med de motsvarande värdena från sekvenserna med likadana nuvarande och nästkommande tillstånd men olika föregående. Om dessa överensstämde med varandra innebar det att sannolikheten att gå från nuvarande tillstånd till nästa är oberoende av föregående och Markovegenskapen kan därför antas hålla för just denna specifika övergång. För att Markovegenskapen ska gälla generellt över hela genomet krävs därför att alla grupper med likadan övergångssekvens från olika föregående tillstånd har likartade frekvensvärden.

För att styrka resultatet utfördes även ett χ^2-test för oberoende. Hypoteserna sattes till

$$H_0 : \text{Markovegenskapen håller}$$
$$H_a : \text{Markovegenskapen håller inte}$$

Tabell 4: En korstabell för en DNA-sekvens av ordning 1.

STS	SSO	TSO-SSO	Total
AAA	N_{11}	N_{12}	$N_{1.}$
⋮	⋮	⋮	⋮
GGG	N_{h1}	N_{h2}	$N_{h.}$
Total	$N_{.1}$	$N_{.2}$	n

Provutfallet Q beräknades enligt Ekvation (6) med de numeriska värdena ifrån Tabell 4. Tabell 4 är en motsvarighet till Tabell 2 med de olika STS som variabel A och SSO och differensen TSO-SSO som variabel B. Skillnaden mellan TSO och SSO står för antalet gånger STS inte förekommer för tillhörande två-tillståndssekvensen, alltså motsatsen till SSO. Till exempel för 'ATG' som STS-variabel är 'AT' motsvarande två-tillståndssekvensen och givet en sekvens med utseendet 'CCATC', fås ett fall då STS inte förekommer. Sannolikheten, $Pr\{'X > Q'\}$, uppskattades med hjälp av Pythons inbyggnda metod `chisqprob` med värdet på Q och antal frihetsgrader som indata. Genom att jämföra ifall det erhållna värdet var mindre eller större än signifikansnivån, $\alpha = 0.05$, kunde nollhypotesen förkastas eller inte förkastas.

Verifieringen av Markovegenskapen genomfördes på Markovkedjor upp till ordning 8. Desto högre ordning, desto större tillståndsrum. Därmed fler möjliga övergångar. Detta medförde en större risk att vissa två- eller tre-tillståndssekvenser aldrig förekom. För att förhindra division med noll i frekvensberäkningarna sattes kvoten SSO/TSO till noll varje gång TSO aldrig förekom. På liknande sätt löstes problemet när Q-värdet beräknades. Division med noll skedde om det förväntande värdet $E_{ij} = \frac{N_{i.}N_{.j}}{n} = 0$. Genom korstabellens konstruktion, se Tabell 4, inträffade detta enbart när radsumman $N_{i.}$, som i det här fallet är ekvivalent med TSO, blev noll. När detta hände sattes termen i Q-summan till noll.

4 Resultat

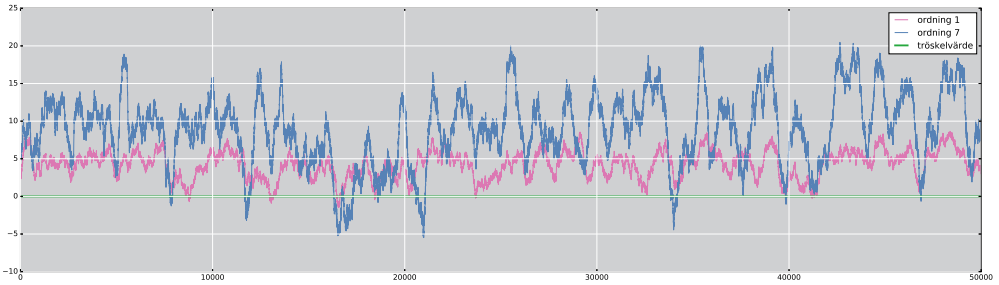
Modellen som beskrivs i föregående avsnitt identifierar sekvenser med avvikande sannolikhetsfördelning i en bakteries genom, för att kunna identifiera hoppande gener. I detta kapitel presenteras resultaten från sensitivitets- och specificitetsanalysen, antalet träffar för de undersökta bakterierna, parvis klassificering och korsklassificering av träffar samt verifieringen av Markovegenskapen.

4.1 Resultaten från sensitivitets- och specificitetsanalysen

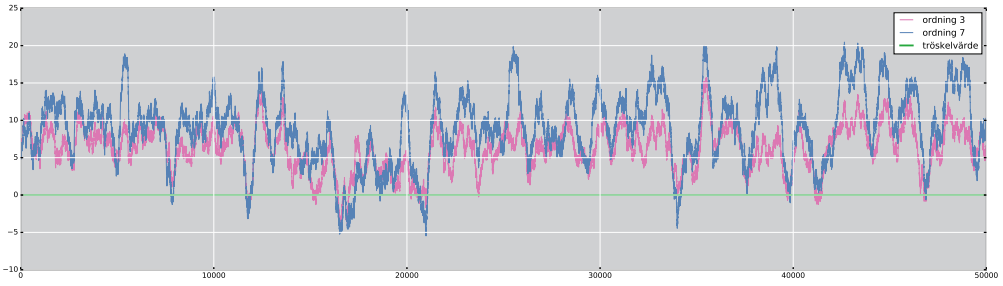
I Avsnitt 4.1.1-4.1.3 presenteras resultaten från sensitivitets- och specificitetsanalysen på träffnivå och nukleotidnivå som gjordes för att utvärdera modellen och för att bestämma bästa ordning och fönsterstorlek. Resultaten omfattar sensitivitets- och specificitetsvärden för ordningar 1-7, fönsterstorlekar 100-2 000 och inklippsstorlekar $600 - 1\,300$. En modell med hög sensitivitet och specificitet har höga värden på Sn, Sp och $Sp*$, det vill säga värden nära 1,000.

4.1.1 Hur ordningen påverkar modellens prestanda

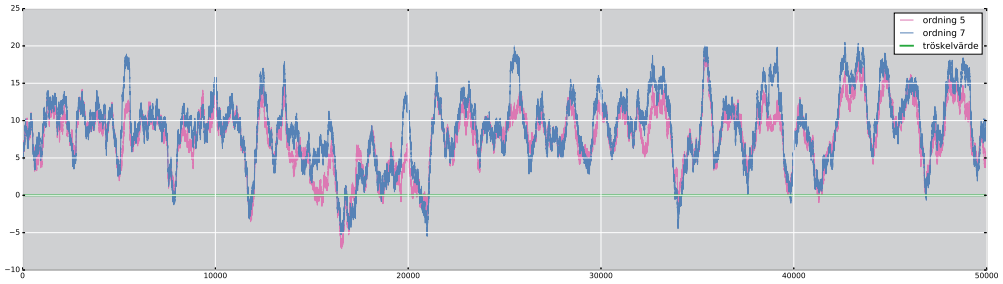
För att jämföra hur ordningen på Markovkedjan påverkar modellens prestanda gjordes grafiska jämförelser mellan två ordningar i figurer där sannolikhetsvärdena för ordningarna ritades mot varandra. Figurerna 13-15 visar jämförelser mellan ordning 7 och ordning 1, 3 och 5. En delsekvens som hamnar under tröskelvärden, den gröna linjen, räknas som en träff. Från figurerna syns det att graferna för sannolikhetsvärdena för ordning 1 och 7 skiljer sig åt mest. Även sannolikhetsvärdena för ordning 3 ser ut att skilja sig från värdena för ordning 7. Träffarna från ordning 7 är djupare än träffarna för ordning 1 och 3. I jämförelsen mellan ordning 5 och 7 är istället graferna betydligt mer lika och ser ut att följa ungefär samma mönster av toppar och dalar.



Figur 13: En jämförelse mellan ordning 1 (rosa) och ordning 7 (blå) av Markovkedjor (fönsterstorlek 500) för en sekvens på 50 000 nukleotider från *E. coli* stam K12 där den gröna linjen motsvarar tröskelvärdet.



Figur 14: En jämförelse mellan ordning 3 (rosa) och ordning 7 (blå) av Markovkedjor (fönsterstorlek 500) för en sekvens på 50 000 nukleotider från *E. coli* stam K12 där den gröna linjen motsvarar tröskelvärdet.



Figur 15: En jämförelse mellan ordning 5 (rosa) och ordning 7 (blå) av Markovkedjor (fönsterstorlek 500) för en sekvens på 50 000 nukleotider från *E. coli* stam K12 där den gröna linjen motsvarar tröskelvärdet.

I Tabell 5 och Tabell 6 finns resultaten från sensitivitets- och specificitetsanalysen för ordningar 1-7 med fönsterstorlek 500. Tabellerna visar hur ordningen påverkar modellens prestanda. Höga värden på sensitivitet och specificitet är önskvärda och från tabellerna framgår det att ordning 7, som är blåmarkerad, presterar bäst. Sn mäter hur många av inklippen som identifieras, Sp mäter andelen korrekta identifikationer och $Sp*$ mäter hur väl modellen identifierar egna gener. Både Sn och Sp ökar med ordningen medan $Sp*$ varierar mellan 0,990-1,000.

Tabell 5: Resultaten från sensitivitets- och specificitetsanalysen på träffnivå för ordningar 1-7, fönsterstorlek 500 (inklipp: 1000 nukleotider).

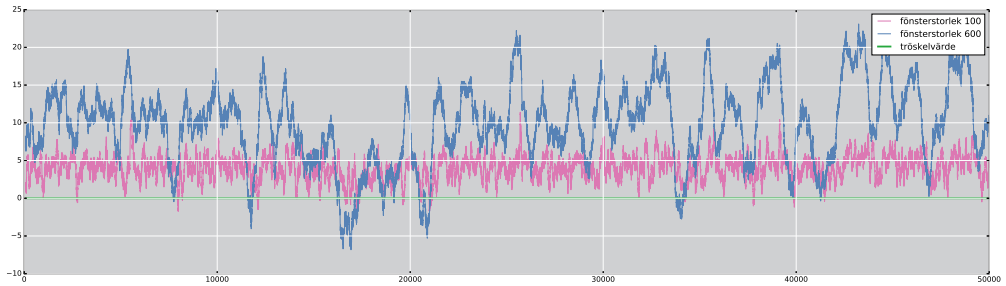
Ordning	Sn	Sp	$Sp*$
1	0,210	0,083	0,990
2	0,490	0,239	1,000
3	0,510	0,285	1,000
4	0,600	0,293	0,990
5	0,690	0,315	1,000
6	0,740	0,307	0,990
7	0,860	0,441	1,000

Tabell 6: Resultaten från Sensitivitets- och specificitetsanalysen på nukleotidnivå för ordningar 1-7, fönsterstorlek 500 (inklipp: 1000 nukleotider).

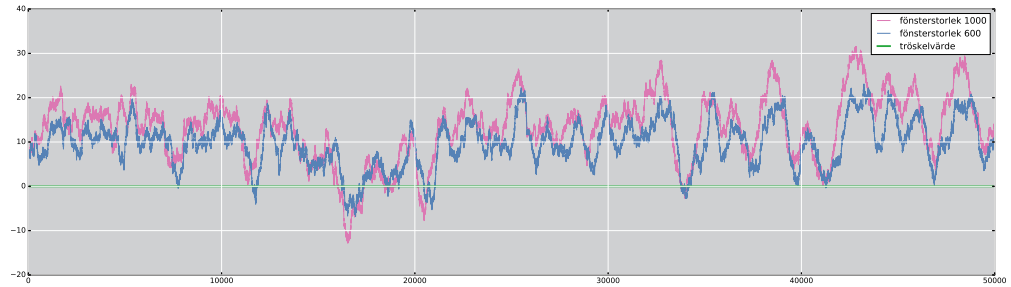
Ordning	Sn	Sp	$Sp*$
1	0,536	0,321	0,900
2	0,745	0,461	0,898
3	0,775	0,514	0,876
4	0,835	0,488	0,848
5	0,886	0,456	0,842
6	0,921	0,390	0,833
7	0,937	0,536	0,884

4.1.2 Hur fönsterstorleken påverkar modellens prestanda

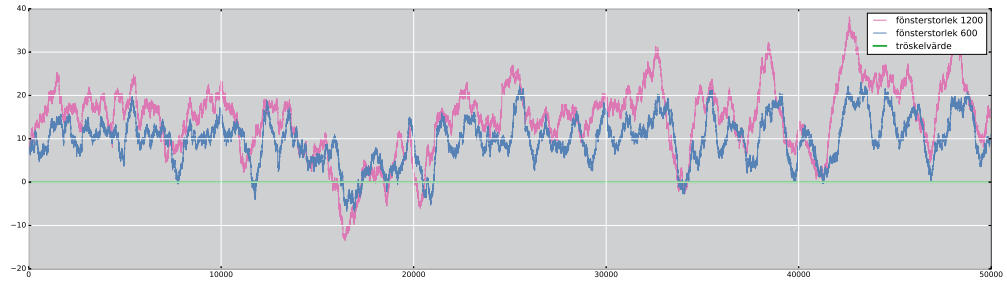
Precis som i undersökningen av bästa ordning på Markovkedjan jämfördes sannolikhetsvärdena för två olika fönsterstorlekar grafiskt, se Figurer 16-18 där fönsterstorlek 600 jämförs med storlekarna 100, 1 000, 1 200 och 2 000 för ordning 7. Som nämnt tidigare är en träff en delsekvens som hamnar under tröskelvärdet (grön linje). I jämförelse med 600 ger fönsterstorlek 100 kortare träffar, en mer kompakt sannolikhetsgraf och det är svårt att urskilja likheter mellan storlekarna. I både Figur 17 och Figur 18, där fönsterstorlek 600 jämförs med 1 000 respektive 1 200, syns tydliga likheter mellan graferna. Skillnaderna blir större då fönsterstorlek 600 och 2 000 jämförs (Figur 19) men graferna ser ut att följa samma trend.



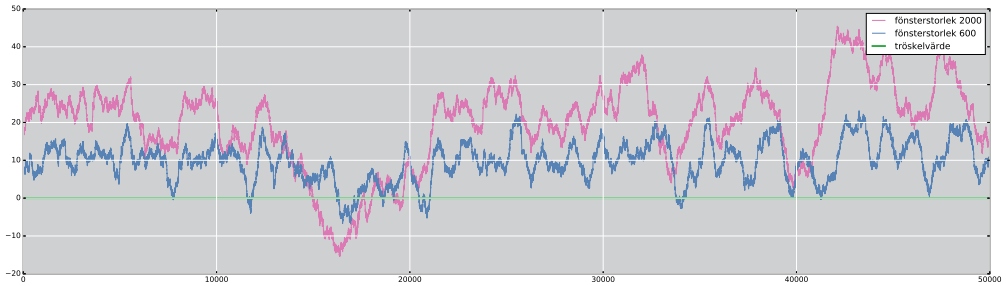
Figur 16: En jämförelse mellan fönsterstorlekar 100 (rosa) och 600 (blå) för en sekvens på 50 000 nukleotider från *E. coli* stam K12, där markovkedjan var av ordning 7. Alla värden under den gröna linjen (tröskelvärdet) räknas som träffar.



Figur 17: En jämförelse mellan fönsterstorlekar 1 000 (rosa) och 600 (blå) för en sekvens på 50 000 nukleotider från *E. coli* stam K12, där markovkedjan var av ordning 7. Alla värden under den gröna linjen (tröskelvärdet) räknas som träffar.



Figur 18: En jämförelse mellan fönsterstorlekar 1 200 (rosa) och 600 (blå) för en sekvens på 50 000 nukleotider från *E. coli* stam K12, där markovkedjan var av ordning 7. Alla värden under den gröna linjen (tröskelvärdet) räknas som träffar.



Figur 19: En jämförelse mellan fönsterstorlekar 2 000 (rosa) och 600 (blå) för en sekvens på 50 000 nukleotider från *E. coli* stam K12, där markovkedjan var av ordning 7. Alla värden under den gröna linjen (tröskelvärdet) räknas som träffar.

I Tabell 7 och Tabell 8 finns resultaten från sensitivitets- och specificitetsanalysen som visar hur väl modellen presterar för fönsterstorlekar 100, 200, . . . , 2 000 för ordning 7. De blå markeringarna visar de intervall av fönsterstorlekar som undersöktes vidare. På träffnivå ökar Sp, som beräknades utifrån den simulerade sekvenser från K12 och inklipp från *B. thetaiotaomicron*, och på nukleotidnivå minskar Sp för längre fönster. Det andra specificitetsvärdet, $Sp*$, som istället beräknades med inklipp från K12, minskar för större fönster på träffnivå och på nukleotidnivå varierar värdena mellan 0,830-0,944 utan att följa någon trend.

Tabell 7: Resultaten från Sensitivitets- och specificitetsanalysen på träffnivå för fönsterstorlekar 100, 200, 300, . . . , 2 000 för ordning 7 (inklipp: 1000 nukleotider). De blå markeringarna visar de fönsterstorlekarna som undersöktes vidare.

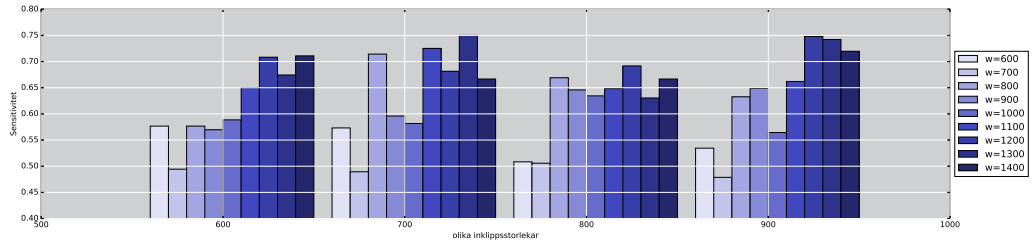
Fönsterstorlek	Sn	Sp	$Sp*$
100	0,000	0,000	1,000
200	0,260	0,074	1,000
300	0,610	0,176	1,000
400	0,720	0,326	1,000
500	0,850	0,450	1,000
600	0,920	0,532	0,980
700	0,870	0,492	0,940
800	0,890	0,586	0,950
900	0,910	0,552	0,920
1000	0,940	0,662	0,920
1100	0,940	0,686	0,910
1200	0,920	0,736	0,890
1300	0,940	0,631	0,890
1400	0,940	0,740	0,880
1500	0,910	0,705	0,930
1600	0,930	0,721	0,910
1700	0,910	0,746	0,920
1800	0,900	0,769	0,900
1900	0,900	0,703	0,910
2000	0,910	0,820	0,940

Tabell 8: Resultaten från Sensitivitets- och specificitetsanalysen på nukleotidnivå för fönsterstorlekar 100, 200, 300, . . . , 2 000 för ordning 7 (inklipp: 1000 nukleotider). De blå markeringarna visar de fönsterstorlekar som undersöktes vidare.

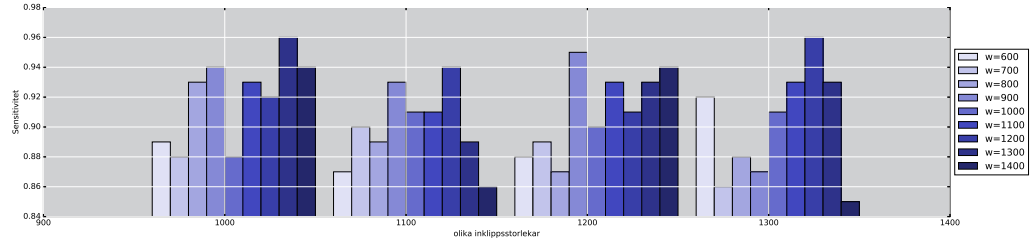
Fönsterstorlek	Sn	Sp	$Sp*$
100	0,524	0,679	0,944
200	0,761	0,637	0,907
300	0,892	0,473	0,846
400	0,917	0,509	0,861
500	0,929	0,517	0,887
600	0,954	0,389	0,839
700	0,966	0,404	0,834
800	0,964	0,417	0,851
900	0,965	0,376	0,871
1000	0,963	0,332	0,855
1100	0,957	0,352	0,876
1200	0,962	0,328	0,854
1300	0,962	0,292	0,883
1400	0,966	0,294	0,830
1500	0,961	0,248	0,844
1600	0,955	0,269	0,882
1700	0,970	0,260	0,867
1800	0,954	0,235	0,866
1900	0,969	0,230	0,870
2000	0,954	0,223	0,874

4.1.3 Hur olika storlekar på gener påverkar modellens prestanda

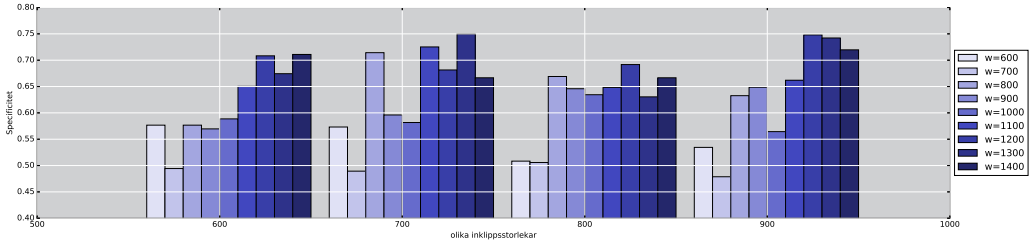
Figur 20 och 21 är två sammanställningar av sensitiviteten på träffnivå för inklipp av längder 600, 700, 800 och 900 respektive 1 000, 1 100, 1 200 och 1 300 nukleotider för fönsterstorlekar 600, 700, ..., 1 400. På samma sätt presenteras specificiteten Sp i Figur 22 och 23. I Appendix C.1 finns likande figurer för $Sp*$ samt alla beräknade värden i tabellform. I Tabell 9 finns de fönsterstorlekar som presterade bäst respektive sämst, med avseende på sensitivitet och de två typerna av specificitet (Sp och $Sp*$), listade efter inklippsstorlek. En noterbar trend är att för sensitiviteten och specificiteten (Sp) presterar fönsterstorlekar på $1\,200-1\,400$ oftast bäst. Den fönsterstorlek som presterar sämst är $w = 700$ som har lägst värden på Sn och Sp för de flesta inklippstorlekarna. För specificiteten ($Sp*$) kan det däremot noteras att $w = 600$ och $w = 1200$ presterar bäst respektive sämst för en majoritet av inklippsstorlekarna. För att få en överblick på hur väl fönsterstorlekarna identifierade inklipp av olika storlekar beräknades medelvärdet av sensitiviteten respektive specificiteten vid de undersökta inklippsstorlekarna för varje fönsterstorlek, se Tabell 10. Från tabellen går det att utläsa att fönsterstorlek 1 100 hade högst specificitet, det vill säga identifierade flest inklipp från *B. thetaiotaomicron*, och fönsterstorlekarna 1 300 och 1 400 presterade bäst med avseende på sensitivitet, vilket innebär att de gjorde minst antal falska identifikationer. Fönsterstorlek 600 hade högst värde på $Sp*$ och var således den storlek som gjorde flest korrekta identifikationer av ”egna” gener.



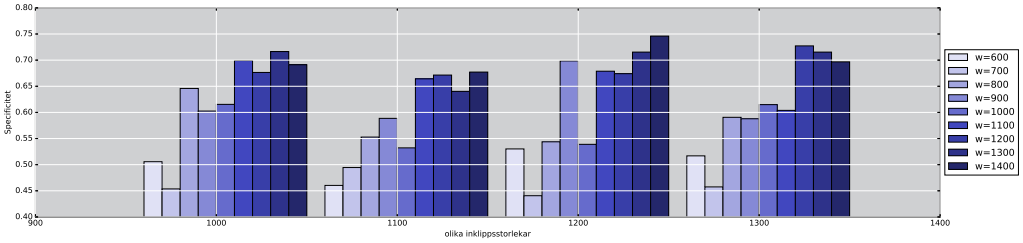
Figur 20: En sammanställning över sensitiviteten på träffnivå för fönsterstorlekar $600, 700, \ldots, 1\,400$, för inklippsstorlekar 600, 700, 800 och 900.



Figur 21: En sammanställning över sensitiviteten på träffnivå för fönsterstorlekar $600, 700, \ldots, 1\,400$, för inklippsstorlekar 1 000, 1 100, 1 200 och 1 300.



Figur 22: En sammanställning över specificiteten på träffnivå för fönsterstorlekar $600, 700, \ldots, 1\,400$, för inklippsstorlekar 600, 700, 800 och 900.



Figur 23: En sammanställning över specificiteten på träffnivå för fönsterstorlekar $600, 700, \ldots, 1\,400$, för inklippsstorlekar 1 000, 1 100, 1 200 och 1 300.

Tabell 9: Sammanställning av de bäst respektive sämst presterande fönsterstorlekarna för olika storlek på inklipp med avseende på Sn, Sp och $Sp*$.

	Sensitivitet Sn		**Specificitet** Sp		**Specificitet** $Sp*$	
Inklipp	Bästa w	Sämsta w	Bästa w	Sämsta w	Bästa w	Sämsta w
600	1400	700	1400	700	1300	700
700	1300	700	1300	700	600	700, 800, 1000
800	1200	700	1200	700	1300	1200
900	1200	700	1200	700	600	1200
1000	1300	700,1000	1300	700	600	1200
1100	1200	1400	1400	600	600	1300
1200	900	800	1400	700	600	1400
1300	1200	1400	1200	700	600	1400

Tabell 10: Medelvärdet av sensitivitet och specificitet (Sp och $Sp*$) vid inklippsstorlekarna 600, 700, ..., 1 300 för fönsterstorlekar 600, 700, ..., 1 400.

Fönsterstorlek	Medelvärde, Sn	Medelvärde, Sp	Medelvärde, $Sp*$
600	0,908	0,526	0,956
700	0,888	0,477	0,930
800	0,915	0,616	0,931
900	0,923	0,617	0,925
1000	0,908	0,584	0,920
1100	0,933	0,667	0,920
1200	0,921	0,697	0,905
1300	0,916	0,698	0,925
1400	0,909	0,697	0,900

4.2 Antal träffar för *Escherichia coli* stam K12

I Tabell 11 finns resultaten för antalet träffar som modellen identifierat hos *Escherichia coli* stam K12 för ordningar 1-7 med fönsterstorlek 500. Av tabellen framgår det att ordning 6 hade flest antal träffar. I Tabell 12 finns resultaten för fönsterstorlekar 100, 200, ..., 2 000 för ordning 7. Tabellen visar att antalet träffar minskar med ökad fönsterstorlek.

Tabell 11: Resultaten över antalet träffar hos *E. coli* för ordning 1-7, fönsterstorlek 500.

Ordning	Antal träffar
1	956
2	815
3	898
4	977
5	1117
6	1235
7	1059

Tabell 12: Resultaten över antalet träffar hos *E. coli* för fönsterstorlekar 100, 200, . . . , 2 000, ordning 7.

Fönsterstorlek	Antal träffar
100	2412
200	2006
300	1949
400	1395
500	1059
600	1021
700	949
800	764
900	719
1000	675
1100	577
1200	553
1300	527
1400	507
1500	467
1600	428
1700	416
1800	406
1900	389
2000	369

Tabell 13 visar hur många av de träffar som fönsterstorlek 600 identifierat för K12 är gemensamma med de träffar som modellen gav för fönsterstorlekar 200, 1 000, 1 500 och 2 000. Genom att jämföra resultatet från tabellen med totala antalet träffar som modellen genererade för olika fönsterstorlekar i Tabell 12 går det att urskilja att för större fönsterstorlekar kommer en högre andel av träffarna att vara gemensamma med träffarna från fönsterstorlek 600.

Tabell 13: Antalet gemensamma träffar mellan fönsterstorlek 600 och fönsterstorlekar 200, 1 000, 1 500 och 2 000.

Fönsterstorlek 1	Fönsterstorlek 2	Antal gemensamma träffar
600	200	842
600	1 000	564
600	1 500	422
600	2 000	341

4.3 Antal träffar för olika genom

Tabell 14 visar antalet identifikationer som modellen gjort för olika bakterier samt antalet träffar per 10^5 baspar (BP). Med en träff menas ett intervall av nukleotider vars följd har låg sannolikhet i jämförelse med resten av genomet. Då längden på bakteriers genom varierar ger den tredje kolumnen ett jämförbart mått på antalet träffar för de undersökta bakterierna. Den fjärde kolumnen anger andelen avvikande baspar hos de olika bakterierna, det vill säga hur stor del av genomet träffarna som modellen identifierar utgör. Till exempel ger modellen att 27,250 % av genomet hos *E. coli* stam SMS-3-5 gav utslag som träffar medan endast 2,110 % av genomet identifierades som träffar hos *A. aeolicus*.

Tabell 14: Resultaten över antalet träffar för olika bakterier för ordning 7 och fönsterstorlek 600.

Bakterie	Antal träffar	Träffar/10^5 baspar	Andel avvikande BP
E. coli stam K12	1021	22,000	26,800 %
E. coli stam SMS-3-5	1088	21,470	27,250 %
B. thetaiotamicron	787	12,570	11,930 %
A. aeolicus	241	15,540	2,110 %
C. pneumoniae	189	15,360	3,220 %
S. aureus	466	15,990	15,950 %
S. pneumoniae	538	25,010	7,170 %
S. scabiei	2457	24,200	4,710 %

4.4 Utvärdering av träffar

Utvärderingen av träffarna från modellen ger dels ett sammanfattat resultat, men även två separata resultat av de två metoderna som går att dra egna slutsatser ifrån. Det sammanfattade resultatet av hela utvärderingen visar att en stor del av träffarna kodar för transposoner eller potentiellt hoppande gener varav flera kan orsaka antibiotikaresistens. Dock så är många av träffarna irreleventa.

4.4.1 Parvis klassificering

Den parvisa klassificeringen grundar sig dels på vilka områden som identifieras och dels vad de eventuellt kodar för baserat på modellen för SMS-3-5 och minst en annan av de andra bakterierna. Det som eftersöks är alltså intervall från SMS-3-5 som återfinns i andra bakterier. I Appendix C.2 (Tabell 32 och Tabell 34) presenteras de områden som ger träffar när alla träffar från SMS-3-5 körs med BLASTn mot träffarna från de andra organismerna samt hela deras genom. I Tabell 15 presenteras de gener som dessa områden kodar för grundat på BLASTx sökningar mot NCBI:s nr-databas.

Det är tydligt att SMS-3-5 ger många träffar med okänd annotering. MFS-superfamiljen som dock har blivit upptäckt av modellen skulle kunna påverka antibiotikaresistens då denna verkar i membranet genom att transportera diverse ämnen i cellen, bland annat läkemedel. De hypotetiska proteinen och eventuellt vissa av icke-träffarna är områden som ännu inte är undersökta av forskningsvärlden.

Övriga bakterier ger en större mängd träffar som kan annoteras till kända gener i studerade bakterier. Vissa av områdena kodar för transposas, som indikerar att det är en hoppande gen. Proteinet UDP-MurNAc-pentapeptid-syntetas bidrar till uppbyggnaden av den del av cellväggen penicilin angriper, vilket gör genen extra intressant.

Tabell 15: Resultat av vad träffarna från Tabell 32 och Tabell 34 ger för träffar mot NCBI:s nr-databas.

Organism	Intervall	Gen-namn
SMS-3-5	233382-235161	Ingen träff
SMS-3-5	234935-236004	Ingen träff
SMS-3-5	236405-237580	Ingen träff
SMS-3-5	2819551-2826799	MFS-Superfamilj
SMS-3-5	3650068-3657193	P-loop_NT-Superfamilj
SMS-3-5	4222099-4223478	Ingen träff
SMS-3-5	4225011-4226177	Ingen träff
SMS-3-5	4331066-4332441	Ingen träff
SMS-3-5	4333993-4335159	Ingen träff
SMS-3-5	4512972-4514358	Ingen träff
SMS-3-5	4515899-4517073	Ingen träff
SMS-3-5	4553143-4554474	Ingen träff
SMS-3-5	4556068-4557243	Ingen träff
S. aureus	521772-527507	Hypotetiskt protein/transposas
S. aureus	1973032-1978859	Hypotetiskt protein/IS1181 transposas
S. pneumoniae	33488-39280	Hypotetiskt protein/transposas
S. scabiei	2854989-2861647	Hypotetiskt protein/aminotransferas
S. scabiei	4398373-4406069	Hypotetiskt protein/Hydrolas
C. pneumoniae	1000327-1005614	Hypotetiskt protein/glycosylas-cellväggs orientat hydrolas
B. thetaiotaomicron	1626414-1629443	Hypotetiskt protein
B. thetaiotaomicron	3034417-3036193	Ingen träff
B. thetaiotaomicron	4983226-4988748	Hypotetiskt protein
A. aeolicus	567182-573353	UDP-MurNAc-pentapeptid-syntetas
A. aeolicus	1191465-1197602	Hypotetiskt protein

4.4.2 Korsklassificering

I Tabell 16 visas resultat från BLASTx av samtliga träffar i SMS-3-5 som också är träff i någon annan av de sju bakterier som jämförts mot. De bakterier som inte finns presenterade i tabellen gav inga träffar alls. De gener som inte påträffades i någon bakterie men som återfinns i tabellen, det vill säga de rader där ingen kolonn är förbockad, gav utslag med metoden men kunde inte kopplas till någon bakterie med hjälp av BLASTx. Flertalet träffar är långa och mer än en gen påträffas då sekvensen körs i BLASTx.

Gener som är extra intressanta är transposas och intregras som indikerar hoppande gener. Korsklassificering ger många blast-träffar på transkriptionsreglerande gener, vilket inte parvis klassificering gör. Däremot kodar korsklassificering även för en mängd hypotetiska proteiner. En annan stor del av blast-träffarna är transferaser som transporterar olika ämnen inom cellen.

Tabell 16: Gener från SMS-3-5 som ger träff i korsklassificeringen, området där genen återfinns samt dess funktion. Bockarna anger om genen påträffas i någon annan bakterie, numrerade enligt 1: *B. thetaiotamicron*, 2: *C. pneumoniae*, 3: *S. aureus*, 4: *S. pneumoniae*

		1	2	3	4
Start - stop	Funktion				
76499-78384	Hypotetiskt protein, membranprotein				
268474-270506	glycosyltransferas, Hypotetiskt protein, glycerol-3-phosphat cytidylyltransferas	✓		✓	✓
306209-315241	putative familj IS3 transposas, integras, Hypotetiskt protein	✓		✓	✓
760436-764606	Hypotetiskt protein EcSMS35_0754, cytochrome bd-II ubiquinol oxidas subunit I, methylaspartat mutas, glutamate mutas			✓	
896078-899757	lipoprotein, Hypotetiskt protein, arginin ABC transport ATP-binding protein ArtP				✓
956513-958729	Integras, Hypotetiskt protein, DNA-binding protein				✓
1043591-1051009	glycosyl transferas familj 1 & 2, Hypotetiskt protein, polysaccharid biosynthesis protein	✓		✓	✓
2544046-2545097	fimbrial yfcV, phosphohistidin phosphatas				
2563631-2567924	helix-turn-helix transcriptional regulator, pilus protein, DNA recombinas, outer membrane autotransport barrel domain-containing protein, DNA-binding transcriptionsregulator DsdC			✓	
2855112-2862743	yttre membran autotransporter barrel domain-containing protein, Hypotetiskt protein,				✓
3228471-3230158	N-acetylmuramoyl-L-alanin amidas AmiC			✓	
3299469-3301157	HTH Superfamilj, IS2 transposas		✓	✓	
3301110-3309249	Transposas, Hypotetiskt protein, N-acylneuraminat cytidylyltransferas,N-acetylneuraminat synthas, UDP-N-acetyl glucosamin 2-epimeras, polysialic acid transport KpsM	✓	✓	✓	✓
3497542-3500800	Hypotetiskt protein, Transcriptionsregulator, glycerat kinas	✓			
3842170-3847982	conserved Hypotetiskt protein, Hypotetiskt protein, putative membranprotein				✓
3862641-3865681	yttre membranlipoprotein, Slp familj, Hypotetiskt protein				✓
3878354-3879847	Transcriptionsregulator, Hypotetiskt protein				
4002520-4006425	Transcriptionsregulator, Hypotetiskt protein, DNA-directed RNA polymeras sigma-70 factor	✓		✓	✓
4036531-4039595	ligas, beta1,3-glucosyltransferas, ADP-heptos–LPS heptosyltransferas, UDP-galactos–(galactosyl) LPS alpha1,2-galactosyltransferas				✓

4080217-4085630	Hypotetiskt protein, integras, sockertransportör			✓
4098383-4102581	Type III effector, type III cell invasion protein, Hypotetiskt protein, invasion protein			
4173975-4176860	Type III effector, Hypotetiskt protein, 6-phosphogluconat phosphatas	✓		
4176561-4177651	Type III effector protein, Hypotetiskt protein, type III effector			
4177371-4179106	Type III effector, Hypotetiskt protein			
4179049-4181446	Type III effector, Hypotetiskt protein			
4283092-4284671	Hypotetiskt protein, membranprotein, protein RarD			
4615658-4617120	aromatic amino acid aminotransferas, Hypotetiskt protein,			
4830518-4832326	Hypotetiskt protein, nitrat/nitrit system sensor histidin kinas			
4870852-4875418	Hypotetiskt protein, transposas, DNA-binding protein, transposas	✓	✓	✓
4994280-4997077	Serin/threonin-protein kinas B, Type III effector pipB2			✓

4.5 Resultaten från verifiering av Markovegenskapen för olika ordningar

Ett utdrag av resultatet från frekvenstesterna av verifieringnen av Markovegenskapen för genomet *Escherichia coli* stam K12 presenteras i Tabell 17-20 för ordningarna 1,4,7,8. Den fullständiga tabellen för ordning 1 finns att studera i Appendix C.3. Tabellerna för de resterande ordningarna utelämnades i denna rapport då de blev för omfattande på grund av att deras storlek ökar exponentiellt med basen 4 (4^{n+2}, n ordning). Till följd av detta dras slutsatsen ifall Markovegenskapen håller eller inte enbart på dessa frekvensutdrag. Genom att granska skillnaderna mellan frekvenserna inom varje tabell ses att frekvensvärdena stämmer bäst överens för ordning 4 följt av ordning 8.

Utöver frekvenstestet genomfördes även ett χ^2-test för oberoende för att kontrollera Markovegenskapen. Tabell 8 presenterar provutfallet, antal frihetsgrader, p-värde samt beslutet för nollhypotes för varje ordning. Som utläses av tabellen avfärdades Markovegenskapen för samtliga ordningar.

Tabell 17: Ett utdrag ur frekvenstabellen för ordning 1, där det föregående tillståndet varierar mellan 'A', 'T', 'G' och 'C'. Nuvarande tillstånd för första delen av tabellen är 'A' och för andra delen 'T' och nästkommande är 'A' respektive 'G'.

STS	SSO	TSO	SSO/TSO
AAA	108964	338006	0,322
TAA	68858	212024	0,325
GAA	83530	267384	0,312
CAA	76654	325327	0,236
ATG	76282	09950	0,246
TTG	76998	339584	0,227
GTG	66142	255699	0,259
CTG	102957	236149	0,436

Tabell 18: Ett utdrag ur frekvenstabellen för ordning 4, där det föregående tillståndet varierar mellan 'A', 'T', 'G' och 'C'. Nuvarande tillstånd för första delen av tabellen är 'AAAA' och för andra delen 'AAAT' och nästkommande är 'A' respektive 'G'.

STS	SSO	TSO	SSO/TSO
AAAAAA	3190	11479	0,278
TAAAAA	2442	6720	0,363
GAAAAA	3343	9074	0,368
CAAAAA	2504	7875	0,318
AAAATG	2297	9040	0,254
TAAATG	1089	5203	0,209
GAAATG	1479	6641	0,223
CAAATG	1077	4865	0,221

Tabell 19: Ett utdrag ur frekvenstabellen för ordning 7, där det föregående tillståndet varierar mellan 'A', 'T', 'G' och 'C'. Nuvarande tillstånd för första delen av tabellen är 'AAAAAAA' och för andra delen 'AAAAAAT' och nästkommande är 'A' respektive 'G'.

STS	SSO	TSO	SSO/TSO
AAAAAAAAA	8	124	0,065
TAAAAAAAA	27	211	0,128
GAAAAAAAA	42	182	0,231
CAAAAAAAA	50	198	0,253
AAAAAAATG	68	232	0,293
TAAAAAATG	61	216	0,282
GAAAAAATG	85	313	0,272
CAAAAAATG	54	181	0,298

Tabell 20: Ett utdrag ur frekvenstabellen för ordning 8, där det föregående tillståndet varierar mellan 'A', 'T', 'G' och 'C'. Nuvarande tillstånd för första delen av tabellen är 'AAAAAAAA' och för andra delen 'AAAAAAAT' och nästkommande är 'A' respektive 'G'.

STS	SSO	TSO	SSO/TSO
AAAAAAAAAA	1	8	0,125
TAAAAAAAAA	3	27	0,111
GAAAAAAAAA	3	42	0,071
CAAAAAAAAA	4	50	0,080
AAAAAAAATG	11	43	0,256
TAAAAAAATG	19	64	0,297
GAAAAAAATG	19	68	0,279
CAAAAAAATG	19	57	0,333

Tabell 21: Resultat från χ^2-oberoendetest för genomet K12.

Ordning	Q-värde	Frihetsgrader	$Pr\{'X > Q'\}$	Beslut för H_0
1	315180,102	63	0	Avfärdas
2	417580,412	255	0	Avfärdas
4	539791,368	4095	0	Avfärdas
7	928739,071	262143	0	Avfärdas
8	1759773,854	1048575	0	Avfärdas

5 Diskussion

Den framtagna modellen i det här kandidatarbetet identifierar sekvenser i bakteriers genom som eventuellt kan vara hoppande gener genom att beräkna sannolikhetsvärden för delsekvenser, fönster, av genomet. Detta grundar sig på antagandet att en bakteries genom har en specifik fördelning av nukleotider som är unik för bakterien i fråga. En hoppande gen har under evolutionens gång överförts från en bakterie till en annan. Alltså härstammar genen från en annan bakterie och kan således antas ha en annan fördelning. Genom att då beräkna sannolikhetsvärden för fönster av genomet bör hoppande gener avvika med ett lägre sannolikhetsvärde eftersom de har en annan fördelning än bakteriens ”egna” gener.

DNA-sekvenser modelleras som Markovkedjor och i modellen skattas övergångsmatrisen och den initiala sannolikhetsvektorn från genomet hos den bakterie som undersöks. Eftersom dessa baseras på hela genomet, inklusive de hoppande generna, antas att sannolikhetsvärdena för fönstrena har en viss fördelning. De hoppande generna antas då tillhöra änden av sannolikhetsfördelningen. Det vill säga, de har låga sannolikhetsvärden. Genom att sedan beräkna ett ensidigt konfidensintervall kan fönster, som antas tillhöra hoppande gener, identifieras.

Resultaten för antalet träffar som modellen ger för de åtta undersökta bakterierna ligger mellan $12,570$ och $25,010$ träffar per 10^5 baspar (Tabell 14). Eftersom detta motsvarar flera tusen träffar är det inte möjligt att undersöka alla under den uppsatta tidsramen för arbetet. Men en första grov undersökning av träffarna från K12 (då ett tiotal träffar undersöktes) gav att många av träffarna inte direkt kan klassificeras som hoppande gener. Det tyder på att modellen även identifierar andra delar av genomet än bara hoppande gener som träffar. Därför krävs en sållning av modellens träffar för att särskilja de intressanta identifikationerna.

De träffar som undersöktes efter sållningen gav betydligt bättre resultat och gener som är kopplade till antibiotikaresistens och andra hoppande gener hittades. Dock gav modellen fortfarande träffar som inte gick att härleda till någon gen eller träffar med okänd annotering, det vill säga gener som ännu inte utforskats. Eftersom modellen hittar alla delsekvenser med en avvikande fördelning betyder det att en sekvens med lågt sannolikhetsvärde inte nödvändigtvis måste vara en hoppande gen. Det innebär att antagandet om en homogen basparsfördelning i bakteriens egna DNA var för grovt och eventuellt måste modifieras för att få mer korrekta resultat. Men eftersom modellen trots allt identifierade hoppande gener uppfyller den sitt syfte.

Efter att modellen skapats och implementerats i Python utfördes de tre undersökningarna som beskrivits i metoden, sensitivitets- och specificitetsanalys, utvärdering av träffar och verifiering av Markovegenskapen parallellt. Detta innebar att en del resultat från de olika delarna inte kunde användas i de andra undersökningarna.

5.1 Modellens prestanda utifrån sensitivtet och specificitet

För att utvärdera hur modellen presterar genomfördes en sensitivitets- och specificitetsanalys. Att analysen utfördes på två sätt var för att få två olika mått som tillsammans ger en mer komplett bild av hur väl modellen presterar. Sensitivitet och specificitet mättes både på träffnivå och nukleotidnivå och för olika ordningar och fönsterstorlekar.

På träffnivå är senstitiviteten ett mått på hur väl modellen identifierar hela gener. Sensitiviteten på nukleotidnivå utvärderar istället hur modellen identifierar varje nukleotid. Det betyder att även träffar som endast täcker en del av ett inklipp ger ökad sensitivitet på nukleotidnivå.

Specificiteten mättes på två sätt. Sp beräknas utifrån antalet sanna positiva och falska positiva träffar. Då blir $1 - Sp$ ett mått på andelen falska positiva, det vill säga hur ofta modellen ger en träff fast den inte borde. Därför säger specificteten hur ofta modellen gör rätt identifikation när det inte är en träff. På träffnivå beskriver Sp hur bra modellen är på att göra korrekta identifikationer av hela gener. Det andra sättet, $Sp*$, beräknas från antalet sanna negativa och antalet falska positiva och är ett mått på hur väl modellen klassificerar det egna genomet. På nukleotidnivå kan Sp istället användas för att indikera hur väl träffarna är positionerade i förhållande till inklippen, det vill säga om träffarna täcker stora eller små områden runt inklippen. $Sp*$ på nukleotidnivå är precis som på träffnivå ett mått på hur väl modellen klassificerar de egna generna.

5.1.1 Val av ordning

Den ordning som presterade bäst valdes utifrån sensitivitet och specificitet på träffnivå och nukleotidnivå. Dessa resultat finns presenterade i Tabell 5 och Tabell 6. Båda analyserna gav att Markovkedjor av ordning 7 hade högst sensitivitet Sn och specificitet Sp, med avseende på sanna positiva och falska positiva, (0,860 respektive 0,441) vilket innebär att det var den ordning som identifierade flest inklipp (syntetiska "hoppande gener") och dessutom gav minst antal falska identifikationer. Även specificiteten $Sp*$ var hög på träffnivå och nukleotidnivå (1,000 respektive 0,884) och därför valdes ordning 7 för vidare analys.

Specificiteten Sp är förhållandevis låg i jämförelse med sensitiviteten. Det beror förmodligen på att de simulerade delarna, baserat på K12, i testsekvensen är mycket längre än de inklippta delarna från *B. thetaiotaomicron* vilket gör att antalet inklipp är mycket lägre än antalet möjliga falska positiva. Därför beräknades även det andra specificitetsmåttet $Sp*$ för en simulerad sekvens med inklipp från K12, där dessa inklipp symboliserar egna gener som modellen alltså inte bör identifiera. Resultaten både på träffnivå och nukleotidnivå var betydligt högre än Sp-resultaten. Dock finns det en osäkerhet i dessa resultat då det är möjligt att vissa av de inklippta sekvenserna från K12 är hoppande gener. Då borde dessa sekvenser avvika med lägre sannolikhetsvärden och därför borde klassificeras som sanna positiva och inte falska positiva. Om det finns sådana sekvenser som modellen identifierar som träffar skulle det resultera i lägre värden på $Sp*$. Å andra sidan, om modellen inte identifierar dessa sekvenser skulle det innebära att modellen inte presterar som den ska. Det är dock viktigt att påpeka att antalet hoppande gener i ett genom är relativt lågt och därför ger resultaten ändå ett relativt pålitligt mått på modellens specificitet.

I tabellerna framgår det att ordning 7 hittar 86 inklipp av 100 och nästan 94 000 inklippta nukleotider av 100 000. Alltså tycks modellen, med avseende på att hitta inklippen, prestera bättre på nukleotidnivå än på träffnivå eftersom 86 träffar av längd 1 000 skulle motsvara 86 000 nukleotider. Eftersom varje inklipp är 1 000 nukleotider långt innebär det att modellen på nukleotidnivå identifierar delar av inklipp som är för korta eller täcker för dåligt för att räknas som sanna positiva på träffnivå. På nukleotidnivå klassificeras 8 000 fler nukleotider som sanna positiva än på träffnivå. På träffnivå missar modellen 14 inklipp och om de 8 000 nukleotididentifikationerna skulle fördelas jämnt över dessa skulle det innebära att modellen fortfarande hittar mer än hälften av inklippen (cirka 570 nukleotider av 1 000). Det är möjligt att dessa träffar är tillräckligt långa för att kunna dra slutsatser om huruvida det är en hoppande gen eller inte. Detta kan betyda att modellen egentligen identifierar fler hoppande gener än vad sensitiviteten på träffnivå visar.

Anledningen till att den högsta ordningen som testades var 7, var att övergångsmatriserna snabbt blir väldigt stora (övergångsmatrisen för ordning 7 är en $4^7 \times 4^7$-matris). En annan anledning är att för högre ordningar blir det vanligare att vissa delsekvenser inte förekommer i genomet, vilket innebär att övergångselementen blir noll. För att lösa det här problemet adderades falska observationer ("pseudocounts") till de övergångar som aldrig inträffade, vilket beskrivs i "Uppskatta parametrar" i Avsnitt 3.1. Detta gjordes bland annat eftersom ett fönster som har en övergång med övergångssannolikheten noll, kommer ha sannolikheten noll vilket inte går att logaritmera. Trots att falska observationer adderades till antalet övergångar för alla som utgick från samma tillstånd kan den här felskattningen ändå få betydande konsekvenser för resultatet om matrisen skulle innehålla många nollelement.

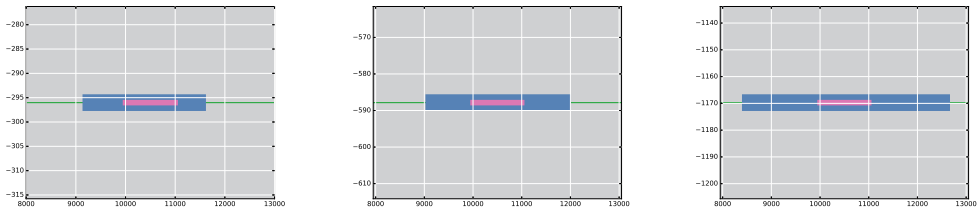
5.1.2 Val av fönsterstorlek

I resultaten av sensitivites- och specificitetsanalysen på träff- och nukleotidnivå för olika fönsterstorlekar (Tabell 7 och 8) framgår det att fönsterstorlekarna 600-1 400 har höga sensitivitesvärden, mellan 0,870-0,940, på träffnivå. Efter fortsatt analys valdes fönsterstorlek 600 att användas vid undersökningar av riktiga genom för att den hade höga värden på Sn, Sp och $Sp*$ på både träffnivå och nukleotidnivå. Även fönsterstorlekarna 1 500-2 000 har sensitivitesvärden över 0,900 men eftersom specificiteten Sp på nukleotidnivå är låg, mellan 0,223-0,269, avfärdades dessa.

I Tabell 7 framgår det att Sp ökar för större fönster på träffnivå. Det kan bero på att det uppkommer en variation i den simulerade datamängden där vissa övergångar är mer sällsynta

än andra och för kortare fönster kommer dessa övergångar att påverka sannolikhetsvärden i en högre grad än för större fönster. Det innebär att få och osannolika övergångar kan resultera i att sannolikhetsvärdet för fönstret blir lägre än konfidensintervallet. Detta kommer ge upphov till korta träffar som klassificeras som falska positiva vilket i sin tur resulterar i låg specificitet på träffnivå. Detta illustreras i Figur 16 som visar sannolikhetsvärden utritade för en sekvens på 50 000 nukleotider från K12. I figuren är dessa värden ritade för fönsterstorlek 100 (rosa) och fönsterstorlek 600 (blått) och det syns tydligt att den kortare fönsterstorleken ger upphov till fler korta träffar. När 600 jämförs med större fönsterstorlekar syns det att graferna har mindre brus och längre träffar (Figur 17- 19).

Utifrån det här resonemanget kan en första slutsats vara att större fönster ger högre specificitet och är att föredra framför kortare. Då skulle specificiteten öka för större fönsterstorlekar, antaget att specificiteten beräknas utifrån antalet sanna positiva och falska negativa. Dock stämmer inte detta eftersom generna som modellen ska identifiera (600-1 300 nukleotider) kommer vara betydligt kortare än längden på fönstret. Det innebär att genens sannolikhetsvärde har en lägre påverkan på det totala värdet för fönstret och därför finns det en risk att genen inte upptäcks. Tabell 7 visar emellertid att fönsterstorlek 2 000 hade högst specificitet på träffnivå, vilket indikerar att skillnaden mellan fönsterstorlek och inklippstorlek inte är för stort. Anledningen till att 2 000 inte valdes, trots att sensitiviteten på träff- och nukleotidnivå var hög, var att fönsterstorleken hade det lägsta värdet på Sp på nukleotidnivå (0,223).



(a) En sann positiv träff för ordning 7 och fönsterstorlek 500.

(b) En sann positiv träff för ordning 7 och fönsterstorlek 1 000.

(c) En sann positiv träff för ordning 7 och fönsterstorlek 2 000.

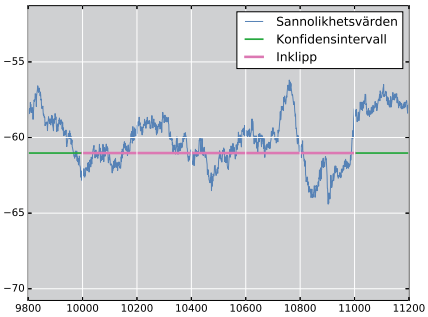
Figur 24: Exempel på hur längden på träffar varierar för fönsterstorlekar 500, 1 000 och 2 000. Den smala gröna linjen motsvarar tröskelvärdet, den rosa markeringen visar inklippets position och den blå visar placeringen av träffen.

På nukleotidnivå minskar Sp för större fönsterstorlekar (fönsterstorlek 100 har specificiteten 0, 679 medan fönsterstorlek 2 000 har specificiteten 0, 223). Detta beror på att större fönster genererar längre träffar. Eftersom inklippstorleken inte ändras kommer varje träff som täcker ett inklipp ha långa ändar som sträcker sig utanför inklippet. Exempel på detta visas i Figur 24 som jämför en sann positiv träff för fönsterstorlek 500, 1 000 och 2 000. Dessa långa ändar kommer från att träffen blir förskjuten åt vänster i förhållande till inklippet eftersom sannolikhetsvärdena beräknas ett fönster framåt. Då kommer fönstersekvensen att täcka inklippet tidigare och därmed blir träffen förskjuten. Anledningen till att träffen täcker längre delar efter inklippen är att ett fönster läggs till varje delsekvens som modellen identifierar. För till exempel fönsterstorlek 2 000 läggs 2 000 nukleotider till, vilket är längre än sekvenserna som klipps in. Därför kommer delar utanför inklippet också räknas som en träff, vilket kan förklara varför specificiteten är låg på nukleotidnivå.

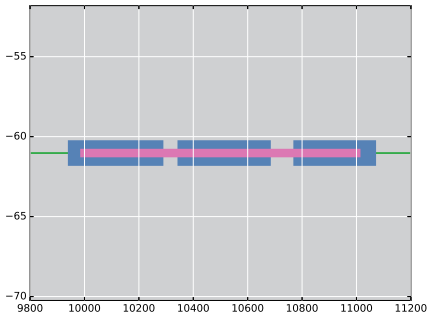
Om modellen skulle användas med en stor fönsterstorlek på ett riktigt genom för att identifiera hoppande gener skulle förmodligen den stora fönsterstorleken påverka resultatet. I Figur 24c, där fönsterstorlek 2 000 används, motsvarar inklippet mindre än en tredjedel av träffen. Detta skulle kunna försvåra identifieringsprocessen, den process där det fastställs om modellen funnit en hoppande gen eller inte.

Sensitiviteten Sn beror på längden på inklipp och vilken fönsterstorlek som används. För korta fönster observerades att det kunde uppstå flera träffar på ett inklipp, se Figur 25, men att dessa var för korta för att räknas som sanna positiva på träffnivå, se Figur 26. Därför kan det vara bra att fundera över huruvida definitionen av en sann positiv träff är

korrekt eller inte. Det är möjligt att den här typen av korta träffar är tillräckliga för att, vid en undersökning, identifieras som hoppande gener och därför borde identifieras som sanna positiva. Men för detta krävs en grundligare undersökning av hur korta träffar ska behandlas.

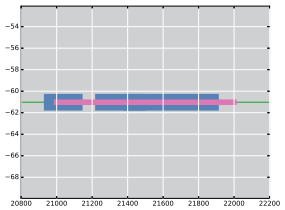


(a) Sannolikhetsvärden i blått för tre träffar. De tre träffarna har värden under den gröna linjen som motsvarar tröskelvärdet och ser ut att täcka delar av det rosa inklippet.

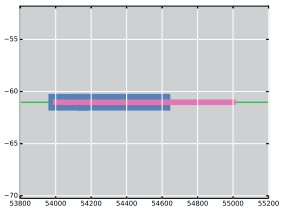


(b) Träffsäkerheten för de tre träffarna ritade med blått. Visar hur träffarna från 25a täcker inklippet i rosa efter att ett fönster lagts till varje träff.

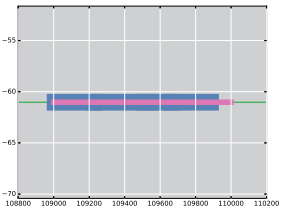
Figur 25: Exempel på tre träffar som överlappar samma inklipp (fönsterstorlek 100, ordning 7) med avseende på dess sannolikhetsvärden och träffsäkerhet.



(a) Två falska positiva träffar som inte täcker inklippet ordentligt (ordning 7 och fönsterstorlek 100).



(b) En falsk positiv träff som inte är tillräckligt lång för att täcka inklippet (ordning 7 och fönsterstorlek 100).



(c) En falsk positiv träff som är för kort för att täcka inklippet (ordning 7 och fönsterstorlek 100).

Figur 26: Exempel på träffar som är för korta eller täcker ett inklipp för dåligt för att klassificeras som sanna positiva. Den smala gröna linjen motsvarar tröskelvärdet, den rosa markeringen visar inklippets position och den blå visar placeringen av träffen.

Ett alternativ är att sådana träffar som Figur 25 visar borde slås samman till en träff vilket skulle resultera i en sann positiv träff istället för tre falska positiva. En sådan här typ av ihopslagning tillämpas redan, se Figur 25a där de tre falska positiva träffarna består av flera mindre träffar. I modellen slås två identifierade intervall ihop till en träff om de befinner sig närmare varandra än ett halvt fönster. Den här storleken valdes eftersom avståndet bör dimensioneras efter fönstret. Exempelvis skulle ett större fixt värde resultera i att för många intervall slås ihop för korta fönster eftersom små fönster gav upphov till många, närliggande träffar vilket diskuterades tidigare.

Den optimala fönsterstorleken beror troligtvis på längden av de gener som modellen ska hitta. Därför utfördes flera analyser på träffnivå där inklippsstorleken varierade mellan 700 och 1 300 för det valda intervallet av fönsterstorlekar (Figur 20-23). Av figurerna går det inte att utläsa något entydigt resultat, det vill säga ingen fönsterstorlek har högst sensitivitet och specificitet (Sp) för alla inklippsstorlekar. I Tabell 9 är den bästa och den sämsta fönsterstorleken listade för varje storlek på inklipp som testades utifrån sensitivitet och specificitet. Den fönsterstorlek som ger sämst resultat för flest inklippsstorlekar är 700. Av figurerna går det att se att 700 alltid presterar bland de tre sämsta fönsterstorlekarna för alla testade inklippsstorlekar (medelvärdet för Sn är $0,888$ och för Sp $0,477$).

De övriga fönsterstorlekarna har ett snitt på över 0,900 i sensitivitet. Det högsta medelvärdet återfinns hos fönsterstorlek 1 100 (medelvärdet 0,933) medan fönsterstorlek 1 200 har det högsta specificitetsvärdet för flest inklippstorlekar. Högst värde på specificiteten Sp påträffas hos storlek 1 300 som har ett snitt på 0,698. Generellt sett verkar Sp öka för större fönsterstorlekar (se Tabell 10), alltså verkar Sp följa samma trend som observerades tidigare. Sp för storlek 1 100 är $0,667$ och är det fjärde högsta värdet.

Med avseende på det andra specificitetsvärdet, $Sp*$, presterar dock fönsterstorlek 600 bäst. Av alla fönsterstorlekar har 600 högst $Sp*$-värde för de flesta inklippsstorlekar (undantag för inklipp av storlekar 600 och 800). Som nämnt tidigare finns det en variation i genomet och därför kommer det att uppstå korta träffar. Eftersom ett fönster läggs till varje träff kommer träffarna av längre fönster totalt sett att bli längre vilket kan leda till att fler inklipp blir identifierade. Detta skulle förklara varför medelvärdet för $Sp*$ minskar för större fönster.

Av undersökningen kan slutsatsen dras att 1 100 presterar bäst med avseende på att hitta de gener som modellen letar efter. Men eftersom sensitivitets- och specificitetsanalysen utfördes parallellt med utvärdering av träffar och korsklassificering valdes fönsterstorlek 600 då det var den storlek som presterade bäst när dessa två undersökningar påbörjades. Dessutom var 600 den fönsterstorlek som gjorde flest sanna negativa identifikationer när olika inklippsstorlekar undersöktes, det vill säga hade högst värde på $Sp*$.

Av fönsterstorlek 100-600 presterar 600 bäst, men när prestandan vid olika inklippsstorlekar studerades upptäcktes det att storlek 1 100 presterade bäst. Speciellt hade fönsterstorlek 600 lägre specificitet på träffnivå än storlek 1 100 (0,526 mot 0,667) vilket innebär att modellen gör fler falska positiva identifikationer i förhållande till sanna positiva än till exempel storlek 1 100. Detta kan dock vara acceptabelt eftersom det är sensitiviteten som är mest betydelsefull, alltså dess förmåga att hitta hoppande gener. Det är viktigare att modellen hittar så många hoppande gener som möjligt än att den gör så få felidentifikationer som möjligt. När modellen används på riktiga genom görs en grov filtrering av träffarna innan de undersöks och därför är antalet felidentifikationer mindre viktigt. Dessutom när sannolikhetsvärdena för olika fönsterstorlekar för *E. coli* stam K12 jämförs grafiskt, se Figur 18 där 600 jämförs med 1 200, syns det att de är förhållandevis lika. Det är också viktigt att komma ihåg att specificiteten minskar avsevärt på nukleotidnivå för större fönster, se Tabell 8 där fönsterstorlek 600 har specificiteten 0,389 och fönsterstorlek 1 100 har 0,352 vilket är en ytterligare anledning att välja en mindre fönsterstorlek. Anledningen till den lägre specificiteten är som diskuterats tidigare att träffarna blir längre för högre fönsterstorlek och därmed täcker inklippen med allt för stora marginaler, se Figur 24. Detta försvårar undersökningen av träffen eftersom en stor del av den identifierade sekvensen inte är del av en hoppande gen. Därför krävs noggrannare analyser av hur välplacerade träffarna är i förhållande till inklippen om en större fönsterstorlek ska användas.

5.2 Antalet träffar för *Escherichia coli* stam K12

Enligt Tabell 11, som visar totala antalet träffar som modellen hittar för *Escherichia coli* stam K12 för olika ordningar, ger ordning 7 på Markovkedjan 1 059 träffar. Det går inte att veta om dessa träffar är sanna eller falska positiva utan att undersöka dem, men med stöd i analysen är det rimligt att anta att ordning 7 hittar flest sanna positiva och minst antal falska positiva av de undersökta ordningarna.

När storleken på fönster varieras för ordning 7 på Markovkedjan så minskar antalet träffar från 2 412 träffar för storlek 100 till 369 träffar för storlek 2 000, se Tabell 12. Detta i samband med resultatet från sensitivtets- och specificitetsanalysen i Tabell 7, där ett värde på Sp på träffnivå mättes till 0,820 för fönsterstorlek 2000, i åtanke tycks antalet hoppande gener hos K12 vara ungefär 300. För den valda fönsterstorleken 600 gör modellen 1 021 träffar. Specificiteten Sp på träffnivå är 0,532. Det skulle kunna vara en indikation på att nästan hälften av alla träffar som modellen ger är falska positiva. Men eftersom Sp beräknas utifrån falska positiva som i förhållande till sanna positiva är mycket större på grund av hur det simulerade genomet skapades (se Avsnitt 5.1.1) är det här måttet inte direkt överförbart. $Sp*$ är 0,980 för fönsterstorlek 600 vilket betyder att 2% av det egna genomet blir felidentifierat. Eftersom antalet egna gener är mycket större än antalet hoppande gener, kan 2 % fortfarande utgöra en betydande del av träffarna.

Fönsterstorlek 2 000 har sensitiviteten 0,910 på träffnivå och fönsterstorlek 600 har 0,920. Detta tyder på att om sensitivitetsanalysen som utförts på simulerad data är överförbar på verklig data så bör de flesta av de träffar som genererades med fönsterstorlek 2 000 även finnas bland träffarna från fönsterstorlek 600. I Tabell 13 framgår det att fönsterstorlekarna 600 och 2 000 har 341 gemensamma träffar (av 369 träffar från storlek 2 000) vilket tyder på att resultaten från analysen stämmer. Eftersom andelen gemensamma träffar ökar för större fönster och specificiteten Sp som mäter andelen falska positiva minskar, tyder det på att modellen sorterar bort sekvenser som inte är hoppande gener för större fönster men som nämnt tidigare kräver större fönsterstorlekar ytterligare undersökningar.

5.3 Antal träffar för olika genom

I och med valet av bakterier från olika familjer är det inte förvånande att de ger olika utslag för antal hoppande gener och framförallt olika andelar avvikande baspar, vilket är ett annat sätt att beskriva hur stor del av genomet som indikeras vara hoppande gener.

Andelen avvikande baspar tillsammans med antal baspar per 10^5 träffar ger en uppfattning om hur stora områdena som betraktas som träffar är. Självklart bör detta skilja sig mellan de olika bakterierna, då deras övergångsmatriser ser olika ut, men att detta skiljer sig avsevärt är underligt. Det kan bero på att vissa genom har träffar vars sannolikhetsvärden ligger närmare tröskelvärdet än andra, vilket gör att fler områden räknas ihop till stora träffar medan de andra får mer distinkta träffar.

Det mest anmärkningsvärda är kanske att K12 ger utslag för fler träffar än SMS-3-5 som är en superresistent bakterie och antogs ha fler hoppande gener i arbetets inledningsfas. Dock har SMS-3-5 en högre andel avvikande baspar än K12, se Tabell 14. Vad detta beror på är oklart, men eftersom K12 och SMS-3-5 är av samma art är det inte konstigt att deras genom är lika varandra på det stora hela.

För båda *E. coli*-stammarna ger modellen ett utslag på över 25% av genomet, vilket kanske snarare indikerar att modellen inte är felfri än att över en fjärdedel av genomet skulle vara främmande (hoppande gener).

Ett tydligt exempel är *A. aeolicus* som är en av de äldsta bakteriearterna som människan har sekvenserat och dessutom lever isolerat i höga temperaturer där inte många andra bakterier lever. På grund av dess isolering från andra typer av bakterier finns det inte heller många andra att byta gener med. Dessutom, skulle det ha skett horisontell genöverföring långt bak i tiden finns möjligheten att, på grund av mutationer över evolutionens gång, gamla hoppande gener numera är nativa.

Likt *A. aeolicus* lever *B. thetaiotamicron* avskilt från många helt främmande bakterier, fast i magsäcken, vilket kan förklara att även denna har ett lågt antal träffar.

S. scabiei som har många kända hoppande genöar borde rimligen ge långa träffar, vilket hade avspeglats i Tabell 14 genom att ha få träffar och en hög andel. Vad antalet träffar och andelen faktiskt visar är att bakterien har många träffar och en låg andel som tyder på är att det är många korta områden, vilket går rakt emot antagandet. Om en bakterie har många och långa främmande segment finns en chans att när modellen räknar ut övergångsmatrisen så betraktas de långa främmande segmenten till slut som nativa, då de har en såpass stor bidragande effekt. För att komma bort från detta problem hade en lösninga kunnat vara att iterera övergångsmatrisberäkningarna där de främmande generna tas bort efter varje beräkning. Detta hade visserligen kunnat ge upphov till andra problem.

S. aureus, C. pneumoniae och *S. pneumoniae* är bakterier som vanligen behandlas med antibiotika och därmed utsätts för ett ökat selektionstryck. Men eftersom dessa är så kallade vildtyper som förekommer naturligt, utan att ha blivit utsatta för antibiotika, borde inte det påverka utslaget. Visserligen kan dessa vildtyper ha påverkats under deras evolution, men inte under så extremt selektionstryck som vid en antibiotikabehandling. Det endast *S. aureus* som ger ett stort utslag med avseende på andel avvikande baspar.

5.4 Utvärdering av träffar

Ingen av tabellerna som redovisar utvärdering av träffar visar ett entydigt resultat. Det finns i huvudsak tre klasser av gener som är av intresse. Träffar som kodar för transposas eller

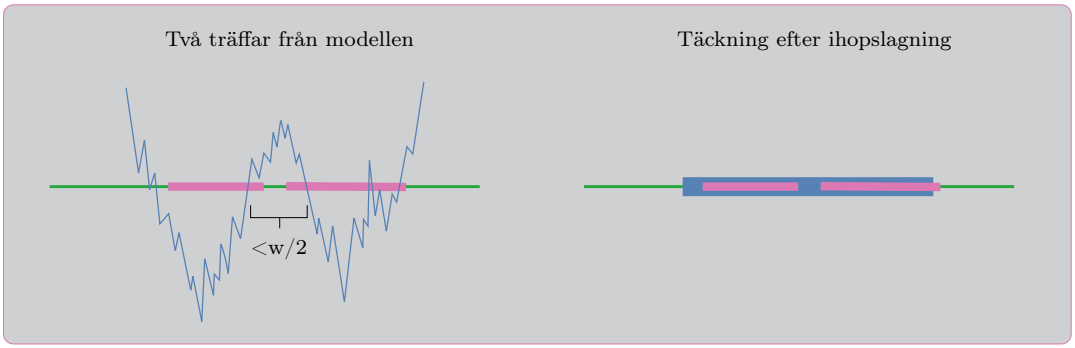
integras kan antas vara hoppande gener. Dessa kan antingen aktivera en flytt av gener eller hjälper till att integrera nytt DNA.

Den andra typen av gener som är intressant är regioner som kodar för transkriptionsreglerande gener. Sådana gener kodas av bland annat *CpG-öar*. CpG-öar är områden som består till största del av basparen 'C'och 'G'. Det medför att dessa områden har en avvikande basparsfördelning som ger utslag för att vara hoppande gener av modellen.

Den sista typen är gener som kan ge upphov till antibiotikaresistens. Att avgöra om de träffar vi får bidrar till antibiotikaresistens är svårare. Många av träffarna säger bara att det är ett ospecifierat protein av en viss typ, exempelvis transposas. Detta beror på att dessa protein inte har undersökts i den utsträckning att det med säkerhet kan sägas något om dem. Även de områden som inte är någon träff kan vara kodande gener i den undersökta bakterien som inte har undersökts ännu. Vissa protein som exempelvis verkar i cellväggar och transportproteiner, transferaser, är intressanta då dessa mycket väl kan ha en bidragande effekt till antibiotikaresistens.

Då många av de BLAST:ade områdena kodar för flera olika gener, där vissa av dem inte kodar för transposoner, är det inte möjligt att med full säkerhet säga att modellen ger en specifik träff på just den hoppande genen. Eftersom dessa träffar är ett ytterligare urval efter Markovmodellens träffar önskas tydligare resultat. Varför detta problem uppstår beror förmodligen på definitionen av en träff där flera närliggande träffar räknas som en träff. Detta kan göra att flertalet gener klassificeras som en träff. En annan bidragande faktor är att antagandet om en homogenfördelning av baspar över hela genomet inte stämmer fullt så bra som hoppats.

Förutom att Markovmodellen inte tar fram tillräckligt specifika områden kan urvalsmodellerna, parvis klassificering och korsklassificering, också vara bristfälliga. Dock verkar det som att det är under det första steget den stora förbättringen kan ske. I Tabell 14 framgår det att i vissa fall anses 25% av genomet vara avvikande, men endast ett fåtal gener identifieras. Återigen beror detta förmodligen på antagandet om vad som är en träff och hur dessa sorteras ut ur genomet. Då två träffar ligger inom en halv fönsterstorlek från varandra slår modellen ihop dessa och betraktar detta som en träff, vilket illustreras i Figur 27. Om strängare antaganden om ihopslagning skulle införas skulle modellen troligtvis ge mer specifika blast-träffar på bekostnad av sämre sensitivitet och specificitet. Ur en tillämpningssynpunkt fungerar trots allt urvalsmetoderna tillräckligt bra, då de ger tillräckligt specifika träffar för vidare undersökningar.



Figur 27: Exempel på problematik som kan uppstå till följd av ihopslagning av träffar. Om två gener (rosa) ligger nära varandra i genomet och ger utslag för två träffar (blått) som är placerade mindre än ett halvt fönster ifrån varandra kommer detta resultera i en träff. Bild av: Emily Curry.

I det stora hela är det svårt att dra generella slutsatser kring resultatet av utvärderingen. Modellen hittar flertalet områden som skulle kunna vara hoppande gener där vissa till och med kan bidra till antibiotikaresistens, men den hittar också många träffar som inte är kopplade till horisontell genöverföring.

Dock är urvalet en grovfiltrering som inte förväntas hitta alla hoppande gener och härleda dess ursprung, utan snarare ta fram en mängd förslag som det går att arbeta vidare med. Utifrån den aspekten fungerade denna metod väl.

5.4.1 Parvis klassificering

Denna metod var tänkt att plocka ut hoppande gener från den stora mängden träffar genom att filtrera ut områden som går att hitta i fler än en bakterie. Den valdes då det är ett relativt enkelt sätt att hantera den stora mängden data.

Det visar sig, och är kanske det mest anmärkningsvärda, att samma områden från alla övriga bakterier ger återkommande träffar i SMS-3-5 samt att alla träffar från SMS-3-5 även är betraktade som träffar i de övriga bakterierna vilket visas i Tabell 15, Tabell 32 samt Tabell 34. Tabell 32 och Tabell 34 går att hitta i Appendix C.2.

Utöver transposasaserna är det UDP-MurNAc-pentapeptid-syntetas och MFS-superfamiljen från Tabell 15 som är värda att fokusera extra på. Det förstnämnda proteinet hjälper till att bygga upp den del av cellväggen pencillin verkar mot och en överproduktion bidrar till en starkare och mer resistent cellvägg. MFS-superfamiljen verkar i cellmembranet med transport av diverse ämnen i cellen och skulle eventuellt kunna ha något med antibiotikatransport att göra.

Alla träffar från Tabell 32 och Tabell 34 ger ungefär samma täckningsgrad av träffen på ungefär 80%, vilket kan bero på förskjutningen som uppstår på grund av fönsterstorleken.

En fördel med denna metod gentemot korsklassificering är att träffarna inte ger utslag på förmodade CpG-öar. Däremot ger metoden dåligt utslag överlag. Förhoppningen var att hitta olika områden i de övriga organismerna som stämmer överens med de olika områdena i SMS-3-5, och utöver det eventuellt kunna härleda ursprung genom att jämföra Tabell 32 och Tabell 34 för att se om vissa områden tyder på träffar i SMS-3-5 och i andra organismer inte, men dessvärre hittades inga sådana områden.

5.4.2 Korsklassificering

Tanken med korsklassificeringen var att försöka avgöra om en gen som anses vara främmande i en stam, möjligtvis är nativ i en annan stam. Från Tabell 16 går det att utläsa att flera av generna ger träff i mer än en annan art i testet, vilket gör att problemet inte kan lösas genom att undersöka fler bakterier. Eftersom metodens syfte är att avgöra vilken art en viss gen härstammar ifrån är detta förvånande och tyder på att metoden är bristfällig.

I många av de övergångsmatriser som skattats för bakterier som jämfördes mot uppstod ett större antal nollelement för ordning 7. Endast *B. thetaiotamicron* innehöll tillräcklig data för att skatta parametrarna med ordning 7. Detta på grund av att flera bakteriers genom är kortare än det hos *E. coli* men även på grund av andelen andelen CpG-öar kan variera för olika bakterier. Av denna anledning har alla jämförelser, förutom *B. thetaiotamicron*, gjorts med ordning 7 för SMS-3-5 mot en lägre ordning.

I Tabell 16 syns även resultat från BLASTx där flera av generna som identifierats inte är hoppande gener, t.ex. de som benämns hypotetiskt protein.

Det är positivt att trots att många områden ger träff på hypotetiska protein ger de även träffar på andra protein. Ett fåtal områden träffar transposaser och flera områden hittar transkriptionsreglerande områden som tyder på CpG-öar. Till skillnad från parvis klassificering ger alla undersökta områden blast-träffar.

5.5 Verifikation av antagandet om Markovegenskapen

För att verifiera antagandet om att DNA-sekvenser kan ses som Markovkedjor utfördes två tester. Ett frekvenstest som baserades på jämförelser mellan antal förekomster av sekvenser med samma övergång från nuvarande till nästkommande tillstånd men från olika föregående. För att styrka resultatet från detta test genomfördes även ett χ^2-test för oberoende.

Från frekvensutdragen i tabell 17-20 bedöms ordning 4 och 8 få bäst resultat då skillnaden mellan värdena inom respektive tabell är marginell. Eftersom frekvensvärdena för alla grupper av likadana övergångssekvenser stämmer överens oberoende föregående tillstånd är det en indikation på att Markovegenskapen håller. Då det inte är möjligt att studera de fullständiga tabellerna på grund av deras snabbt tilltagande storlek blir resultaten från tabellerna lätt missvisande. Värdena som presenteras behöver nödvändigtvis inte spegla de resterande, då utdragen endast utgör en liten del av tabellerna. Då det dessutom är svårt att avgöra vad som anses vara en tillräckligt bra överensstämning mellan frekvenserna läggs en större

vikt på utfallet från χ^2-testet. Från Tabell 21 utläses att Markovegenskapen avfärdades för samtliga ordningar. För de lägre ordningarna var detta förväntat då dessa även valdes bort i sensitivitets- och specificitetsanalysen i Avsnitt 5.1. Då 7 valdes ut som bästa ordning hade det varit fördelaktigt om Markovegenskapen inte avfärdats i detta test eftersom de erhållna resultaten hade då bekräftats i teorin.

Att Markovegenskapen inte håller kan bero på förekomsten av CpG-öar i genomet. CpG-öar medför att vissa av de STS som undersöks har större möjlighet att förekomma än andra. Detta leder till att de motsvarande frekvensvärdena blir märkbart högre än resterande och indikerar då på att Markovegenskapen inte håller. Detsamma gäller för χ^2-testet då det förväntas att kategorierna SSO och TSO-SSO ska vara oberoende av STS-kategorierna, alltså att skillnaden mellan SSO och TSO-SSO, ska vara försumbar för alla STS. Förekomsten av CpG-öar kan medföra att detta inte uppfylls för de STS som innehåller kombinationer av baserna C och G. Det innebär att de motsvarande SSO-värden blir högre medan TSO-SSO blir lägre och en större skillnad mellan dessa erhålls. Existensen av CpG-öar tyder alltså på ett beroende inom genomet, vilket kan vara orsaken till att Markovegenskapen inte håller.

Det ska dock observeras att verifieringen av Markovegenskapen genomfördes på det verkliga genomet K12 medan sensitivitets- och specificitetsanalysen i Avsnitt 5.1 utfördes på ett simulerat genom. Det simulerade genomet baserades på slumpmässiga element från övergångsmatrisen, vilket innebär en lägre sannolikhet för förekomsten av CpG-öar. Därmed är det fullt möjligt att Markovegenskapen håller för det simulerade genomet i sensitivitets- och specificitetsanalysen.

I Avsnitt 4.4 ”Utvärdering av träffar” används modellen under antagandet att Markovegenskapen håller. Från resultaten framgår det att hoppande gener upptäcks även om modellen har en viss svårighet att enbart urskilja gener kodande för antibiotikaresistens. På grund av den faktiska upptäckten av hoppande gener är det därför ett rimligt antagande att DNA-sekvenser kan ses som Markovkedjor i verkligheten.

Verifieringen av Markovegenskapen utfördes enbart på bakterien *E. coli* K12 upp till ordning 8. Valet att stanna vid 8 beror på att beräkningstiden för högre ordningar tog oacceptabelt lång tid då antal STS ökar exponentiellt med basen 4 (4^{n+2}, n ordning). Undersökningar av högre ordningar, andra genom eller delar av genom utan CpG-öar kan eventuellt resultera i att antagandet om Markovegenskapen även stämmer i teorin.

5.6 Vidareutveckling av modellen

Ur en tillämplingssynpunkt fungerar modellen tillräckligt bra för att hitta grova intervall av intresse där specifika gener sedan kan undersökas vidare på exempelvis NCBI. För att utveckla och förfina modellen finns det ett antal områden som kan undersökas vidare. Dessa förbättringsförslag har inte kunnat realiseras under den uppsatta tidsramen utan presenteras som exempel på vidareutvecklingar av projektet.

5.6.1 Överanpassning till data

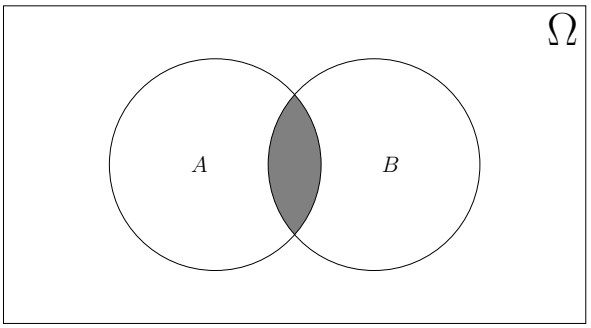
Ordning 7 på Markovkedjan bedömdes som den ordning som presterade bäst, utifrån dess höga sensitivitets- och specificitetsvärden. Dock upptäcktes det att högre ordningar gav upphov till fler nollelement i övergångsmatrisen vilket hanterades genom att lägga till falska observationer i skattningen. Dessa falska observationer kommer ha en viss påverkan på resultatet och därför hade ett exempel på en vidareutveckling av modellen varit att undersöka hur stor påverkan faktiskt blir. Detta hade kunnat användas som ett komplement till sensitivitets- och specificitetsanalysen vid val av ordning på Markovkedjan.

5.6.2 CpG-separation

Som tidigare konstaterats ger modellen även träffar på CpG-öar. Genom att filtrera ut dessa från modellens träffar hade förhoppningsvis ett bättre resultat uppnåtts.

Detta kan uttryckas genom att alla kombinationer av baspar i ett genom ses som utfallsrummet Ω, modellens träffar som delmängden $A \subset \Omega$ och CpG-öarna som delmängden $B \subset \Omega$. Då uppstår problemet med CpG-öar då $A \cap B$ inträffar. Genom att exempelvis sätta

en övre 'G'+'C'-andelsgräns och sålla bort träffar med för hög andel hade detta undvikits. En hög 'G'+'C'-andelsgräns gör att färre eventuella hoppande gener sållas bort, men även att färre CpG-öar filtreras ut.



Figur 28: Ett Venn-diagram som beskriver den eventuella sitationen med CpG-öar där Ω är hela utfallsrummet, $A \subset \Omega$ är modellens utslag på hoppande träffar och $B \subset \Omega$ är genomets CpG-öar. Då dessa sammanfaller vilket motsvarar den skuggade ytan, $A \cap B$, ger modellen utslag för CpG-öar. Bild av: Anton Martinsson.

Denna metod kan användas på två sätt. Det första är att modellen för att hitta hoppande gener utvecklas vidare med att sålla bort CpG-öarna, som är beskrivet tidigare i diskussionen. Efter att CpG-öarna är bortsorterade ur genomet skulle övergångsmatriser kunna beräknas igen, för att få precisare värden. Den andra infallsvinkeln är att ta tillvara på att att CpG-öar har tydligare avvikande fördelning och använda detta för att leta efter CpG-öar. Detta hade dock inte fallit under detta arbete om hoppande gener, men hade kunnat användas i andra sammanhang.

5.6.3 Undersökning av hur en hoppande gen ska definieras

När träffarna undersöktes upptäcktes det att många av de identifierade sekvenserna motsvarade gener som var svåra att klassificera, det vill säga gener som inte direkt kunde härledas till en hoppande gen. Det är dock möjligt att en del av dessa gener någon gång i tiden överförts horisontellt till den undersökta bakterien under ett selektionstryck men att det var så länge sedan att det inte går att fastställa med säkerhet. Då skulle det krävas en tydligare definition av vad som i det här arbetet ska räknas som en hoppande gen och en grundligare undersökning kring vad en hoppande gen är och hur länge en gen kan räknas som en sådan. Detta skulle ge tydligare anvisningar i klassificeringen av modellens träffar som potentiella hoppande gener.

5.6.4 Träffars precision i förhållande till inklipp

I sensitivitets- och specificitetsanalysen upptäcktes det att de träffar som modellen identifierade som sanna positiva var förskjutna både före och efter inklippen, se Figur 24. Detta resulterade som nämnt tidigare i låga sensitivitesvärden för Sp på nukleotidnivå då dessa beräknades efter antalet sanna positivt identifierade nukleotider och antalet falska positiva. Om en undersökning av hur mycket träffen blir förskjuten för olika fönsterstorlekar och olika typer av inklipp genomfördes skulle det innebära att större fönsterstorlekar, $1\,500 - 2\,000$, skulle kunna användas för att identifiera hoppande gener. Detta skulle vara fördelaktigt eftersom de större fönsterstorlekarna hade hög sensitivitet och specificitet (Sp och $Sp*$) på träffnivå vilket innebar att de identifierade många av inklippen och gjorde få falska identifikationer. Detta skulle i sin tur förenkla vidare undersökningar av träffar på riktigt genom.

5.6.5 Definitionen av en sann positiv träff

Från senstivitets- och specificitetsanalysen upptäcktes också att flera träffar klassificerades som falska positiva trots att de täckte en stor del av ett inklipp, se Figur 26. Dessutom gjordes upptäckten att flera korta träffar kunde täcka ett inklipp då korta fönster användes, Figur 25. Som diskuterades ovan borde kanske dessa korta träffar slås ihop till en träff vilket

skulle ge en sann positiv träff. För att kunna slå ihop sådana träffar krävs en grundligare undersökning av hur nära två träffar kan ligga för att räknas som en träff. I det här arbetet användes ett värde på en halv fönsterstorlek. För att kunna hitta ett bättre värde skulle det förmodligen också krävas undersökningar kring hur längden av träffar varierar för olika fönsterstorlekar.

Dessutom borde kanske de träffar som täcker större delen av ett inklipp identifieras som sanna positiva. Det betyder att undersökningar kring hur stor del av en gen som behövs för att säkert kunna identifiera den skulle behöva göras. Detta skulle i sin tur kunna leda till en ändrad definition kring en sann positiv träff och eventuellt högre sensitivitetsvärden.

5.6.6 Uppföljning av Markovegenskapen

Då långa beräkningstider hindrade undersökningen av Markovegenskapen för ordningar högre än 8 för K12 är det av intresse att fortsätta verifieringen eftersom det är möjligt att egenskapen håller för högre ordningar.

Ett annat val av bakterie hade även kunnat ge annorlunda resultat då fördelningen av baspar eventuellt skiljer sig åt. CpG-öar kan därmed vara mer eller mindre framträdande vilket påverkar resultatet. Med exempelvis CpG-separation kan det undersökas hur stor inverkan CpG-öar har på frekvenserna och χ^2-testet. Detta kan göras genom att Markovegenskapen testas på de delar av genomet som inte innehåller CpG-öar.

5.7 Slutsatser och återkoppling till frågeställning

I undersökningen av Markovegenskapen för olika ordningar på hela genomet för *E. coli* stam K12 visade det sig att Markovegenskapen inte håller för de undersökta ordningarna. Det betyder att DNA-sekvensen egentligen inte kan betraktas som en Markovkedja. I metoden antas det dock att egenskapen gäller och modellen påvisar trots detta hoppande gener. Detta innebär att antagandet verkar hålla i praktiken men inte i teorin.

Av de testade ordningarna visade det sig att ordning 7 på Markovkedjan presterade bäst utifrån sensitivitets- och specificitetsanalysen för K12. Ordning 7 visade sig vara den ordning som hade högst värden på Sn, Sp och $Sp*$ av ordningar 1-7.

Vid undersökningen av fönsterstorlekar var resultaten inte lika entydiga. Olika fönsterstorlekar presterade olika bra beroende på storleken på inklippen och dessutom hade ingen fönsterstorlek högst värden på samtliga tre mått. Fönsterstorlek 600 valdes att användas till vidare undersökningar efter en avvägning av dess prestation.

För att besvara huruvida modellen ger bra resultat går det att utvärdera metoden efter hur väl den presterar på simulerad data och hur väl den presterar på verkliga genom.

Utifrån senstivitets- och specificitetanalysen på simulerad data verkade modellen fungera mycket väl. Den gjorde många korrekta identifikationer, både vad gäller att klassificera inklipp från en annan bakterie och inklipp från samma. Denna analys grundar sig på de antaganden som gjorts kring Markovkedjor och basparsfördelning. Slutsatserna gäller då främst under dessa antaganden som har visat sig inte alltid hålla.

När modellen applicerades på faktiska genom visade det sig att de antaganden som modellen grundar sig på förmodligen var för starka och därför inte stämmer helt med verkligheten. Modellen ger inte ett entydigt utslag för en typ av gener. Utifrån urvalet som gjordes är den största gruppen hypotetiska proteiner som ännu inte är undersökta. Detta gör att det är svårt att säga vad de egentligen är. Utöver dessa hittar modellen transposaser och integraser som tyder på hoppande gener, men även transkriptionsreglerande gener som tyder på CpG-öar. Dessa är de tydligaste typerna av gener, men utöver dessa hittar även modellen gener som inte är av intresse i denna undersökning.

Då antalet bakterier som undersöks är mycket litet i förhållande till hur många det finns är det osannolikt att de valda bakterierna skulle ha gett upphov till de eventuella hoppande gener som modellen hittar hos bakterien som undersöks. Vad undersökningen av härkomst, korsklassificeringen, visar, ger generna träffar i mer än en organism vilket är både osannolikt och gör att det inte går att härleda ursprung till en specifik bakterie. Med andra ord går det att hitta de hoppande generna i andra organismer, men inte att härleda dess ursprung.

Eftersom modellen fungerar bäst för Markovkedjor av ordning 7 krävs att genomen som undersöks är tillräckligt stora för att inte ge nollelement i övergångsmatrisen. En annan begränsande faktor som också bidrar till nollelement i övergångsmatrisen är att vissa genom har en övervägande del G och C, vilket minskar sannolikheten att andra övergångar inträffar.

Ytterligare en begränsning för arbetet är att ett urval var tvunget att ske, då mängden träffar blev för stor att hantera. Faktumet att endast åtta bakterier undersöktes är också en begränsning som gjordes på grund av den stora datamängden.

Sammanfattningsvis kan modellen hitta hoppande gener men den gör vissa grova antaganden som gör att resultaten kräver ytterligare undersökningar och bearbetning. Som metoden ser ut just nu hittar den inte enbart gener kopplade till horisontell genöverföring och mer specifikt gener för antibiotikaresistens. Det är dock en bra metod för att göra en första filtrering för att se vilka områden av genomet som är intressanta att undersöka vidare. Vi hoppas att modellen, med modifieringar, kan ligga till grund för framtida modeller med syftet att förstå och begränsa spridning av antibiotikaresistens.

Referenser

[1] Cao M, Wang T, Ye R, Helmann JD. Antibiotics that inhibit cell wall biosynthesis induce expression of the Bacillus subtilis σ and σM regulons. Molecular Microbiology. 2002;45(5):1267–1276. Hämtad 5/5 2016 från http://dx.doi.org/10.1046/j.1365-2958.2002.03050.x.

[2] Gafter-Gvili A, Fraser A, Paul M, Vidal L, Lawrie T, van de Wetering M, et al. Antibiotic prophylaxis for bacterial infections in afebrile neutropenic patients following chemotherapy. Cochrane Database Syst Rev. 2012;18(1):1361–6137. Hämtad 5/5 3016 från http://www.ncbi.nlm.nih.gov/pubmed/22258955.

[3] World Health Organization. Antimicrobial resistance; 2015-04-15. Hämtad 6/5 2016 från http://www.who.int/mediacentre/factsheets/fs194/en/.

[4] Norberg M, Bergström M, Jethava V, Dubhashi D, Hermansson M. The IncP-1 plasmid backbone adapts to different host bacterial species and evolves through homologous recombination. Nature Communications. 2011;2(268):268.

[5] Folkhälsomyndigheten. Frågor och svar;. Hämtad 1/4 2016 från http://www.folkhalsomyndigheten.se/skyddaantibiotikan/\\fragor-och-svar/\#resistens-konsekvenser.

[6] World Helath Organization. Antimicrobial resistance; 2015-04. Hämtad 18/4 2016 från http://www.who.int/mediacentre/factsheets/fs194/en/.

[7] Dillon L. The Genetic Mechanism and the Origin of Life. vol. 1. Springer US; 2012.

[8] Nordén B, Nordlund S. DNA;. Hämtad 18/4 2016 från http://www.ne.se/uppslagsverk/encyklopedi/lång/dna.

[9] Nordén B, Stafshede P. RNA;. Hämtad 18/4 2016 från http://www.ne.se/uppslagsverk/encyklopedi/lång/rna.

[10] Malmquist J, Pettersson U. genom;. Hämtad 24/4 från http://www.ne.se/uppslagsverk/encyklopedi/lång/genom.

[11] Sniegowski PD, Gerrish PJ, Johnson T, Shaver A. The evolution of mutation rates: separating causes from consequences. BioEssays. 2000;22(12):1057–1066.

[12] Alberts B, Johnson A, Lewis J, Raff M, Roberts K, Walter P. Molecular Biology of the Cell. vol. 6. Garland Science; 2015.

[13] Langille MGI, Hsiao WWL, Brinkman FSL. Detecting genomic islands using bioinformatics approaches. Current Drug Targets. 2010 Maj;8(5):373–382.

[14] Pray LA. Transposons: The Jumping Genes. Nature Education. 2008 September;1(1):204.

[15] Antibiotikaresistens. Antibiotikaresistens; 2016-01-15. Hämtad 12/4 2016 från http://www.antibiotikaresistens.se/.

[16] Higgins PG, Fluit AC, Schmitz FJ. Fluoroquinolones: Structure and Target Sites. Current Drug Targets. 2003;4(2):181–190. Hämtad 2/5 2016 från http://www.eurekaselect.com/node/63182/article.

[17] Bush K, Jacoby GA. Updated Functional Classification of β-Lactamases. Antimicrob Agents Chemother. 2010 Mars;54(3):969–976.

[18] Hsu HP. Probability, Random Variables, and Random Processes. McGraw-Hill; 2014.

[19] Grimmett G, Stirzaker D. Probability and Random Processes. Oxford University Press; 2009.

[20] Axelson-Fisk M. Comparative Gene Finding, Models, Algorithms, Implementation. vol. 20. Springer-Verlag London; 2015.

[21] Holm S, Jagers P, Nerman O. Statistisk slutledning. vol. 1. Chalmers tekniska högskolan och Göteborgs universitet; 1989.

[22] Rice JA. Mathematica Statistics and Data Analysis. vol. 3. Brooks/Cole, Cengege Learning; 2007.

[23] Gärdenfors U, Marklund K, Prawitz D. klassifikation;. Hämtad 26/4 2016 från `http://www.ne.se/uppslagsverk/encyklopedi/lång/klassifikation`.

[24] Burset M, Guigo R. Evaluation of Gene Structure Prediction Programs. Genomics. 1996 February;34(0298):353–367.

[25] Zvelebil M, Baum JO. Understanding bioinformatics. Garland Science, Taylor & Francis Group; 2008.

[26] NIST/SEMATECH. What are statistical tests?;. Hämtad 26/4 2016 frpn `http://www.itl.nist.gov/div898/handbook/prc/section1/prc13.htm`.

[27] NIST/SEMATECH. Chi-Square Goodness-of-Fit Test;. Hämtad 1/5 2016 från `http://www.itl.nist.gov/div898/handbook/eda/section3/eda35f.htm`.

[28] Stat Trek. Statistics and Probability Dictionary;. Hämtad 23/4 2016 från `http://stattrek.com/statistics/dictionary.aspx?definition=Contingency\table`.

[29] Chi-Square test of Independence;. Hämtad 2/5 2016 från `https://onlinecourses.science.psu.edu/stat500/node/56`.

[30] Everitt BS. The Analysis of Contingency Tables. London, Chapman and Hall; 1977.

[31] Langille MGI, Hsiao WWL, Brinkman FSL. Junjie Qin et. al. Nature. 2010 Mars;464(7285):59–65.

[32] Deckert G, Warren PV, Gaasterland T, Young WG, Lenox AL, Graham DE, et al. The complete genome of the hyperthermophilic bacterium Aquifex aeolicus. Nature. 1998 Mars;392(0298):353–358.

[33] Campbell LA, Kuo CC, Grayston JT. Chlamydia pneumoniae and cardiovascular disease. Emerging Infectious Diseases. 1998 Oktober;4(4):571–579.

[34] Li Y, Cao B, Zhang Y, Zhou J, Yang B, Wang L. Complete Genome Sequence of Staphylococcus Aureus T0131, an ST239-MRSA-SCCmec Type III Clone Isolated in China. Journal of Bacteriology. 2011 Juli;13(193):3411–3412.

[35] Li G, Hu FZ, Yang X, Cui Y, Yang J, Qu F, et al. Complete Genome Sequence of Streptococcus pneumoniae Strain ST556, a Multidrug-Resistant Isolate from an Otitis Media Patient. Journal of Bacteriology. 2012 Juni;194(12):3294–3295.

[36] Bukhalid RA, Takeuchi T, Labeda D, Loria R. Horizontal Transfer of the Plant Virulence Gene, nec1, and Flanking Sequences among Genetically Distinct Streptomyces Strains in the Diastatochromogenes Cluster. Applied and Environmental Microbiology. 2002 Februari;68(2):571–579.

[37] Jacoby GA, Cattoir V, Hooper D, Martínez-Martínez L, Nordmann P, Pascual A, et al. qnr Gene Nomenclature. Antimicrob Agents Chemother. 2008 July;52(7):2297–2299.

[38] Lahey Clinic. β-Lactamase Classification and Amino Acid Sequences for TEM, SHV and OXA Extended-Spectrum and Inhibitor Resistant Enzymes; 2015-10-15. Hämtad 2/5 2016 från `http://www.lahey.org/studies/`.

[39] Skuriat-Olechnowska M. Statistical Inference for Markov Chains with interval censoring. Delft University of Technology; 2005.

[40] Maximum Likelihood Estimation for Markov Chains; 2009-01-29. Hämtad 6/5 2016 från http://www.who.int/mediacentre/factsheets/fs194/en/.

A Bevis

A.1 Bevis för elementen i övergångsmatrisen

För att skatta parametrarna p_{ij} till en Markovkedja kan Maximum Likelihood-metoden användas. Låt $x^n = \{x_1, x_2, \ldots, x_n\}$ vara en händelse med tillståndsrum S. Sannolikheten för x^n ges av

$$Pr(X^n = x^n) = \pi_{x_1} \prod_{t=1}^{n} Pr\{X_t = i_t | X_{t-1} = x_{t-1}\} = \pi_{x_1} \prod_{t=2}^{n} p_{x_{t-1}, x_t},$$

vilket är ekvivalent med

$$L(p) = \pi_{x_1} \prod_{i=1}^{k} \prod_{j=1}^{k} p_{ij}^{n_{ij}},$$

där n_{ij} är antalet gånger tillstånd i följs av j i x_1^n och $k = |S|$. Maximera $L(p)$ med avseende på p, under förutsättningen att $\sum_j p_{ij} = 1$ för alla $i \in S$. För att få ett uttryck som är enklare att arbeta med betraktar vi logaritmen av $L(p)$, det vill säga

$$\max_p \mathcal{L}(p) = \max_p \log L(p) = \max_p (\log \pi_{x_1} + \sum_{i,j} n_{ij} \log p_{ij}),$$

Genom att Lagrangerelaxera samtliga bivillkor har vi i nästa steg infört Lagrangemultiplikatorer λ_k där $k = 1, \ldots, n$. Funktionen att optimera blir då

$$f(p) = \mathcal{L}(P) - \sum_{i=1}^{n} \lambda_i (\sum_j p_{ij} - 1)$$

som ger att optimum ges då

$$p_{ij} = \frac{n_{ij}}{\lambda_i}.$$

Eftersom $\sum_j p_{ij} = 1$ så är $\sum_j \frac{n_{ij}}{\lambda_i} = 1 \Leftrightarrow \lambda_i = \sum_j n_{ij}$. Det slutgiltiga uttrycket för p_{ij} är därför

$$p_{ij} = \frac{n_{ij}}{\sum_j n_{ij}}, \tag{7}$$

där n_{ij} alltså är antalet övergångar från i till j som sker i händelsen x^n [40].

A.2 Bevis för högre ordningens Markovkedjor

Sannolikheten att nästa värde $X_n = x_n$ för en Markovkedja av ordning k är

$$Pr\{X_n = x_n | X_{n-1} = x_{n-1}, ..., X_{n-k} = x_{n-k}\} = Pr\{X_n = x_n | X_{n-1:n-k} = x_{n-1:n-k}\}.$$

Från definitionen av betingad sannolikhet har vi att

$$Pr\{A \cap B | B \cap C\} = \frac{Pr\{A \cap B \cap B | C\}}{Pr\{B|C\}} = \frac{Pr\{A \cap B | C\}}{Pr\{B|C\}} = Pr\{A | B \cap C\}$$

vilket ger att

$$Pr\{X_n = x_n | X_{n-1:n-k} = x_{n-1:n-k}\} = Pr\{X_{n:n-k+1} = x_{n:n-k+1} | X_{n-1:n-k} = x_{n-1:n-k}\}.$$

Det betyder alltså att nästa steg i en Markov kedja av ordning k kan betraktas som en sekvens av k steg. Där $k-1$ av dessa steg överensstämmer med kedjans föregående tillstånd.

B Implementation

B.1 Python-kod

```
import pickle
import numpy
import math
import random
import matplotlib.pyplot as plt

class complete_model():

  def read_fasta(filename):
    f1 = open(filename,'r')
    sequence = []
    for i,line in enumerate(f1):
      if i > 0:
        for c in line[:-1]: #do not include the last character ('/n')
          sequence.append(c)
    return sequence

  def state_space(order, S, nucleotides, constant = 1):
    if constant != order:
      S_tmp = []
      for j in range(len(S)):
        for k in (range(len(nucleotides))):
          S_tmp.append(S[j]+nucleotides[k])
      S = complete_model.state_space(order, S_tmp, nucleotides, constant+1)
    return S

  def overl_count(string, sub):
    count = start = 0
    while True:
      start = string.find(sub, start) + 1
      if start > 0:
        count+=1
      else:
        return count

  def param_est_pseud(sequence,order):
    # Load sequence as string
    order = int(order)

    alphabet = ['A', 'C', 'T', 'G']

    # Create the state space
    S = complete_model.state_space(order, alphabet, alphabet, 1)
    # Possible Transitions
    TS = complete_model.state_space(order+1, alphabet, alphabet, 1)

    # Storage for occurances
    tuples = {}
    otuples = {}
    init_prob = {}
    # Estimated Transistion probability matrix
    ETPM = {}
    # Initialize all states as 0
    for s in S:
      tuples[s] = 0
      init_prob[s] = 0
      for t in alphabet:
        ETPM[(s,s[1:]+t)] = 0
        otuples[s + t] = 0
    # loop through seq and count
    for k in range(len(sequence)-order-1):
      tuples[sequence[k:k+order]] += 1
      otuples[sequence[k:k+order+1]] += 1

    # Find zero counts and add pseudocounts
    for s in S:
      for t in alphabet:
```

```
68          if(otuples[s+t] == 0):
69            for u in alphabet:
70              otuples[s+u] += 1
71              tuples[s] += 1
72
73      for s in S:
74        # Calculate initial probabilities
75        init_prob[s] = float(tuples[s])/(len(sequence)-order)
76        for t in alphabet:
77          # Generate string to organize Collection ETPM
78          # order number of letters in s (from state)
79          # 1 from t (alphabet)
80          step = s + t
81          # remove first letter to get next state
82          next_state = step[1:]
83          # Calculate p_{s,next_state}
84          # This should never be zero now.
85          if(tuples[s] != 0):
86            ETPM[(s, next_state)] = float(otuples[step])/tuples[s]
87          else:
88            print("zero element")
89            ETPM[(s, next_state)] = 0
90      return {'ETPM':ETPM,'pi':init_prob}
91
92  def param_est(sequence,order):
93      # Load sequence as string
94      order = int(order)
95
96      alphabet = ['A', 'C', 'T', 'G']
97
98      # Create the state space
99      S = complete_model.state_space(order, alphabet, alphabet, 1)
100     # Possible Transitions
101     TS = complete_model.state_space(order+1, alphabet, alphabet, 1)
102
103     # Storage for occurances
104     tuples = {}
105     otuples = {}
106     init_prob = {}
107     # Estimated Transistion probability matrix
108     ETPM = {}
109     # Initialize all states as 0
110     for s in S:
111       tuples[s] = 0
112       init_prob[s] = 0
113       for t in alphabet:
114         ETPM[(s,s[1:]+t)] = 0
115         otuples[s + t] = 0
116     # loop through seq and count
117     for k in range(len(sequence)-order-1):
118       tuples[sequence[k:k+order]] += 1
119       otuples[sequence[k:k+order+1]] += 1
120
121     for s in S:
122       # Calculate initial probabilities
123       init_prob[s] = float(tuples[s])/(len(sequence)-order)
124       for t in alphabet:
125         # Generate string to organize Collection ETPM
126         # order number of letters in s (from state)
127         # 1 from t (alphabet)
128         step = s + t
129         # remove first letter to get next state
130         next_state = step[1:]
131         # Calculate p_{s,next_state}
132         # This should never be zero now.
133         if(tuples[s] != 0):
134           ETPM[(s, next_state)] = float(otuples[step])/tuples[s]
135         else:
136           print("zero element")
137           ETPM[(s, next_state)] = 0
138     return {'ETPM':ETPM,'pi':init_prob}
```

```
139
140 def scores_calc(ETPM, init_prob, sequence, order, w):
141   order = int(order)
142   w = int(w)
143   total = len(sequence)
144   # Log(zero) will never happend if used on
145   # the same seq as is used for estimation
146   lognoll = 20
147
148   score = [0 for x in range(total-w-order)]
149
150   print("Iterating through sequence \n Sequence length:" + str(total) + "\n
    Window size:" + str(w))
151   # Calculate first window
152   subseq = sequence[0:w]
153   #print("subseq", subseq[0:order])
154   prob = math.log10(init_prob[subseq[0:order]])
155   # loop through window
156   for k in range(w-order-1):
157     # Extract steps made
158     this_state = subseq[k:k+order]
159     next_state = subseq[k+1:k+order+1]
160     # calculate probability score for window
161     if( ETPM[(this_state,next_state)] != 0 ):
162       prob += math.log10(ETPM[(this_state,next_state)])
163     else:
164       #print("lognoll")
165       prob -= lognoll
166
167   score[0] = prob
168
169   # loop through whole sequence
170   for i in range(total-w-order-1):
171     if(i%100000==0):
172       print("Iteration number:" + str(i))
173     # Subtract inital prob
174     if( init_prob[sequence[i:i+order]] != 0 ):
175       prob -= math.log10(init_prob[sequence[i:i+order]])
176     else:
177       print("lognoll")
178       prob += lognoll
179     # Subtract first step
180     if( ETPM[(sequence[i+1:i+order+1],sequence[i+2:i+2+order])] != 0):
181       prob -= math.log10(ETPM[(sequence[i+1:i+order+1],sequence[i+2:i+2+
    order])])
182     else:
183       print("lognoll")
184       prob += lognoll
185     # Replace with last probability and
186     # inital probability
187     if(init_prob[sequence[i+1:i+order+1]] != 0):
188       prob += math.log10(init_prob[sequence[i+1:i+order+1]])
189     else:
190       print("lognoll")
191       prob -= lognoll
192     if( ETPM[(sequence[i+w-order-1:i+w-1],sequence[i+w-order:i+w])] != 0):
193       prob += math.log10(ETPM[(sequence[i+w-order-1:i+w-1],sequence[i+w-
    order:i+w])])
194     else:
195       print("lognoll")
196       prob -= lognoll
197     score[i+1] = prob
198
199   return score
200
201 def parametric_bootstrap(ETPM, init_prob, window_size, size, nbrCI):
202   ndrCI = int(nbrCI)
203   ci_array = numpy.zeros(nbrCI)
204   size = int(size)
205   window_size = int(window_size)
206
```

```
    # Sample and window size
    population_size = int(size)+window_size
    resample = population_size - window_size
    alphabet = {"A","C","G","T"}
    states = set([key[0] for key in list(ETPM.keys())])
    scores = [0 for x in range(population_size-window_size)]

    for m in range(nbrCI):
      # Generate "population"
      for k in range(size):
        # take initial step
        rnd = random.random()
        CDF = 0
        for s in states:
          CDF += init_prob[s]
          if(rnd < CDF):
            current_state = s
            score = init_prob[current_state]
            break

        for i in range(window_size):
          CDF = 0
          rnd = random.random()
          # Go through all steps form this state
          # If the random number rnd is less then
          # the sum of probabilities to go from this
          # state, then keep this (break the loop)
          for s in alphabet:
            CDF += ETPM[(current_state,current_state[1:]+s)]
            if(rnd < CDF):
              if( ETPM[(current_state,current_state[1:] + s)] != 0):
                score += math.log10(ETPM[(current_state,current_state[1:] + s)
    ])
                current_state = current_state[1:] + s
                break
              else:
                score -= 10
                current_state = current_state[1:] + s
        scores[k] = score
      print('iteration: ', m) # ta bort sen

      ci_array[m] = numpy.sort(scores)[int(size*0.01)-1]

    CI = math.fsum(ci_array)/nbrCI
    return CI

  def find_all_below(order,window_size,ci, scores):
    order = int(order)
    window_size = int(window_size)
    ci = int(ci)

    logscores = scores
    logscores = numpy.array(logscores)
    indexes = []
    delind = []

    all_below = numpy.where(logscores < ci)
    below_indexes = []
    tmp = []
    for i in range(len(all_below[0])):
      if i < len(all_below[0])-1:
        if all_below[0][i+1]-all_below[0][i] <= int(window_size/2):
          tmp.append(all_below[0][i])
        else:
          tmp.append(all_below[0][i])
          below_indexes.append(tmp)
          tmp = []
      else:
        if all_below[0][i]-all_below[0][i-1] <= int(window_size/2) :
          tmp.append(all_below[0][i])
        else:
```

```
277             below_indexes.append(tmp)
278             tmp = []
279             tmp.append(all_below[0][i])
280       if len(tmp) > 0:
281         below_indexes.append(tmp)
282       return(below_indexes)
283
284     def return_intervals(below_indexes,window_size):
285       nbrDips = len(below_indexes)
286
287       #get the first and last indexes each dip
288       intervals = [[0 for x in range(2)] for x in range(nbrDips)]
289       for i in range(nbrDips):
290         if len(below_indexes[i]) > 1:
291           intervals[i][0] = below_indexes[i][0]
292           intervals[i][1] = below_indexes[i][len(below_indexes[i])-1] +
      window_size
293         else:
294            intervals[i][0] = below_indexes[i][0]
295            intervals[i][1] = below_indexes[i][0] + window_size
296       return(intervals)
297
298     def simulation(ETPM, pi, order, size):
299       order = int(order)
300       size = int(size)
301       alphabet = {"A","C","G","T"}
302       states_tmp = list(ETPM.keys())
303       states = set([key[0] for key in states_tmp])
304       sequence = ""
305
306       # take initial step
307       rnd = random.random()
308       CDF = 0
309       if size != 0:
310         for s in states:
311           CDF += pi[s]
312           if(rnd < CDF):
313             current_state = s
314             sequence += str(s)
315             break
316       for i in range(size-order):
317         CDF = 0
318         rnd = random.random()
319         for s in alphabet:
320           CDF += ETPM[(current_state,current_state[1:]+s)]
321           if(rnd < CDF):
322             if( ETPM[(current_state,current_state[1:] + s)] != 0):
323               current_state = current_state[1:] + s
324               sequence += s
325               break
326       return sequence
327
328     def plot_scores(score, ci, nbr_K12, nbr_bac, nbr_inserts):
329       score_np = numpy.zeros(len(score))
330       for i in range(len(score)):
331         score_np[i] = score[i]
332       fig = plt.figure(1)
333       ax = fig.add_subplot(111)
334       ax.plot(range(1))
335       ax.patch.set_facecolor('#CFD0D2')
336       ax.patch.set_alpha(0.2)
337       plt.grid(b=True, which='both', color='white',linestyle='-')
338       plt.plot(range(len(score)),score, color = '#5682B6' ) #'#1f78b4' '#3c7dbd'
339       plt.hold(True)
340       plt.axhline(y=ci, color = '#26A83E', linewidth=2) #'#ae1c28'
341       x = numpy.zeros(nbr_bac)
342       y = numpy.zeros(nbr_bac)
343       for j in range(nbr_inserts):
344         for i in range(nbr_bac):
345           x[i] = ((j+1)*nbr_K12 + (j*nbr_bac))+i
346           y[i] = ci
```

```
347       plt.plot(x,y,color = '#DC78B3', linewidth=3) #'#e41a1c' '#0d288a'
348     plt.axis([0, (nbr_K12 +nbr_bac)*nbr_inserts+nbr_bac, numpy.min(score_np),
        numpy.max(score_np)])
349     plt.show()
350
351   def plot_hits(score, intervalsOfDips, w, min_hit, add_after, nbr_bac, ci,
        nbr_inserts, nbr_K12):
352     score_np = numpy.zeros(len(score))
353     for i in range(len(score)):
354       score_np[i] = score[i]
355     fig1 = plt.figure(1)
356     plt.plot(range(len(score[0:(nbr_inserts+1)*nbr_K12 + (nbr_inserts*nbr_bac)
        ])),score[0:(nbr_inserts+1)*nbr_K12 + (nbr_inserts*nbr_bac)], color = '
        #5682B6')
357     plt.hold(True)
358     plt.axhline(y=ci, color = '#26A83E', linewidth=2)
359     x = numpy.zeros(nbr_bac)
360     y = numpy.zeros(nbr_bac)
361     for j in range(nbr_inserts):
362       for i in range(nbr_bac):
363         x[i] = ((j+1)*nbr_K12 + (j*nbr_bac))+i
364         y[i] = ci
365       plt.plot(x,y,color = '#DC78B3', linewidth=3)
366     ax = fig1.add_subplot(111)
367     ax.plot(range(1))
368     ax.patch.set_facecolor('#CFD0D2')
369     plt.grid(b=True, which='both', color='white',linestyle='-')
370     plt.axis([0, (nbr_K12 +nbr_bac)*nbr_inserts+nbr_bac, numpy.min(score_np),
        numpy.max(score_np)])
371     fig2 = plt.figure(2)
372     plt.axhline(y=ci, color = '#26A83E', linewidth=2)
373     # plot only the hits > min_hit as blue lines
374     for i in range(len(intervalsOfDips)):
375       y_1 = intervalsOfDips[i][0] # start position of hit i
376       y_2 = intervalsOfDips[i][len(intervalsOfDips[i])-1]
377       if y_1 < (nbr_inserts+1)*nbr_K12 + (nbr_inserts*nbr_bac):
378         if (y_2 - y_1) > min_hit:
379           x_coord = numpy.zeros(len(intervalsOfDips[i])+add_after)
380           y_coord = numpy.zeros(len(intervalsOfDips[i])+add_after)
381           for j in range(len(intervalsOfDips[i])):
382             x_coord[j] = intervalsOfDips[i][j]
383             y_coord[j] = ci
384           for j in range(add_after):
385             x_coord[len(intervalsOfDips[i])+j] = intervalsOfDips[i][len(
        intervalsOfDips[i])-1]+j+1
386             y_coord[len(intervalsOfDips[i])+j] = ci
387           plt.plot(x_coord, y_coord, color = '#5682B6', linewidth=30)
388     #Plot the stretches
389     x = numpy.zeros(nbr_bac)
390     y = numpy.zeros(nbr_bac)
391     for j in range(nbr_inserts):
392       for i in range(nbr_bac):
393         x[i] = ((j+1)*nbr_K12 + (j*nbr_bac))+i
394         y[i] = ci
395       plt.plot(x,y,color = '#DC78B3', linewidth=10)
396       plt.axis([0, (nbr_K12 +nbr_bac)*nbr_inserts+nbr_bac, ci-(numpy.max(
        score_np)-ci), numpy.max(score_np)])
397       plt.hold(True)
398     ax = fig2.add_subplot(111)
399     ax.plot(range(1))
400     ax.patch.set_facecolor('#CFD0D2')
401     plt.grid(b=True, which='both', color='white',linestyle='-')
402     plt.axis([nbr_K12+nbr_bac-200, nbr_K12 +nbr_bac + 200, ci-(numpy.max(
        score_np)-ci), numpy.max(score_np)])
403     plt.show()
404
405   def compare_plot(scores1,scores2,ci_1,ci_2,start,stop):
406     logscores1 = scores1[start:stop]
407     logscores1 = numpy.array(logscores1)
408     logscores1 = logscores1 - ci_1
409     logscores2 = scores2[start:stop]
```

```
410     logscores2 = numpy.array(logscores2)
411     logscores2 = logscores2 - ci_2
412     fig1 = plt.figure(1)
413     plt.plot(range(len(logscores1)),logscores1, color = '#5682B6') #'#DC78B3')
414     plt.hold(True)
415     plt.plot(range(len(logscores2)),logscores2, color = '#5682B6')
416     plt.axhline(y=0, color = '#26A83E', linewidth = 3)
417     ax = fig1.add_subplot(111)
418     ax.plot(range(1))
419     ax.patch.set_facecolor('#CFD0D2')
420     ax.patch.set_alpha(0.2)
421     plt.grid(b=True, which='both', color='white',linestyle='-')
422     plt.show()
423 
424   def compare_window(intervals1, intervals2, w1, w2):
425     nbr_same = 0;
426     for i in range(len(intervals1)):
427       for j in range(len(intervals2)):
428         if intervals1[i][0] <= intervals2[j][0] and intervals1[i][len(
    intervals1[i])-1]+w1 >= intervals2[j][len(intervals2[j])-1]+w2:
429           nbr_same += 1
430           break
431     return nbr_same
```

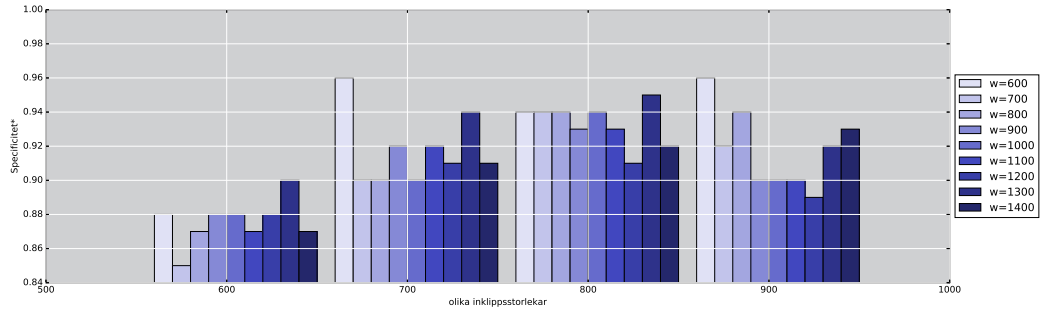
B.2 Beskrivning av funktioner

Tabell 22: Beskrivning av vad funktionerna i `complete_model.py` gör, vilka argument funktionerna tar in och vad respektive funktion returnerar.

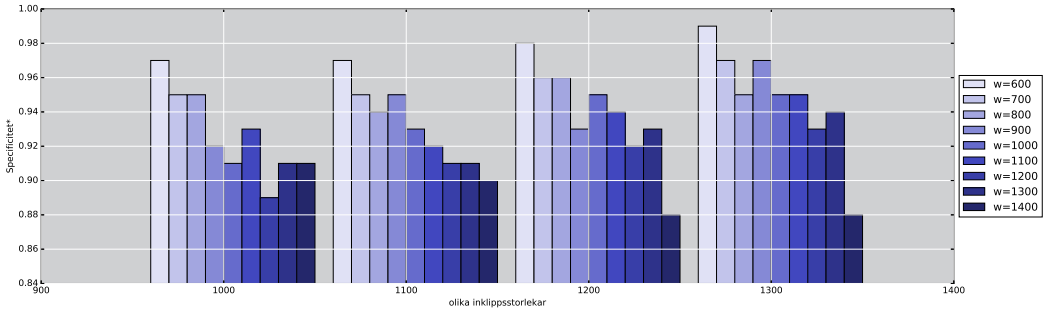
complete_model.py			
Fuktionsnamn	Argument	Returnerar	Beskrivning
`read_fasta`	filename	sequence	Läser in FASTA-fil och returnerar en lista med en strängar som representerar nukleotider
`state_space`	order, S, nucleotides, constant = 1	S	Tar in ordning och utfallsrum och returnerar ett modifierat utfallsrum
`overl_count`	string, sub	count	Undersöker om en viss sekvens finns i en sträng
`param_est_pseudo`	sequence, order	{ETPM, init_prob}	Skatta parametrar och använder sig av pseudosekvens
`param_est`	sequence, order	{ETPM, init_prob}	Skatta parametrar och använder sig inte av pseudosekvens
`scores_calc`	ETPM, init_prob, sequence, order, w	score	Beräknar sannolikhetsvärden för en DNA-sekvens
`parametric_bootstrap`	ETPM, init_prob, window_size, size, nbrCI	CI	Beräknar tröskelvärde med hjälp av parametrisk Bootstrap
`find_all_below`	order, window_size, ci, scores	below_indexes	Hittar alla index i sekvensen för vilka har ett sannolikhetsvärde lägre än tröskelvärdet
`return_intervals`	below_indexes, window_size	intervals	Hittar första och sista index för sekvensintervall med lägre sannolkhetsvärde än tröskelvärdet
`simulation`	ETPM, pi, order, size	sequence	Simulerar en DNA-sekvens utifrån given övergångsmatris
`plot_scores`	score, ci, test, nbr_K12, nbr_bac, nbr_inserts	Inget	Ger graf över sannolikhetsvärden och tillhörande tröskelvärde
`plot_hits`	score, intervalsOfDips, w, min_hit, add_after, nbr_bac, ci, nbr_inserts, nbr_K12	Inget	Ger graf som visar modellens träffar
`compare_plot`	scores1, scores2, ci_1, ci_2, start, stop	Inget	Ritar ut en figur över två olika sannolikhetsvärden
`compare_window`	intervals1, intervals2, w1, w2	nbr_same	Jämför träffar från två olika fönsterstorlekar och returnerar antalet gemensamma träffar

C Resultat

C.1 Resultat för olika inklippsstorlekar



Figur 29: En sammanställning över specificiteten ($SP*$) på träffnivå för fönsterstorlekar $600, 700, \ldots, 1\,400$, för inklippsstorlekar 600, 700, 800 och 900.



Figur 30: En sammanställning över specificiteten ($SP*$) på träffnivå för fönsterstorlekar $600, 700, \ldots, 1\,400$, för inklippsstorlekar $1\,000$, $1\,100$, $1\,200$ och $1\,300$.

Tabell 23: Resultaten från sensitivitets- och specificitetsanalysen på träffnivå för ordning 7, fönsterstorlek 600 och inklipp på 600, 700, ..., 1300 nukleotider.

Ordning 7, fönsterstorlek 600			
Storlek på inklipp	Sensitivitet Sn	Specificitet Sp	Specificitet $Sp*$
600	0,920	0,577	0,880
700	0,940	0,573	0,960
800	0,910	0,508	0,940
900	0,930	0,534	0,960
1000	0,890	0,534	0,970
1100	0,870	0,460	0,970
1200	0,880	0,530	0,980
1300	0,920	0,517	0,990

Tabell 24: Resultaten från sensitivitets- och specificitetsanalysen på träffnivå för ordning 7, fönsterstorlek 700 och inklipp på 600, 700, . . . , 1300 nukleotider.

Ordning 7, fönsterstorlek 700			
Storlek på inklipp	Sensitivitet Sn	Specificitet Sp	Specificitet $Sp*$
600	0,870	0,494	0,850
700	0,920	0,489	0,90
800	0,880	0,506	0,940
900	0,900	0,479	0,920
1000	0,880	0,454	0,950
1100	0,900	0,495	0,950
1200	0,890	0,441	0,960
1300	0,860	0,457	0,970

Tabell 25: Resultaten från sensitivitets- och specificitetsanalysen på träffnivå för ordning 7, fönsterstorlek 800 och inklipp på 600, 700, . . . , 1300 nukleotider.

Ordning 7, fönsterstorlek 800			
Storlek på inklipp	Sensitivitet Sn	Specificitet Sp	Specificitet $Sp*$
600	0,940	0,577	0,870
700	0,950	0,714	0,900
800	0,930	0,669	0,940
900	0,930	0,633	0,940
1000	0,930	0,646	0,950
1100	0,890	0,553	0,940
1200	0,870	0,544	0,960
1300	0,880	0,591	0,950

Tabell 26: Resultaten från sensitivitets- och specificitetsanalysen på träffnivå för ordning 7, fönsterstorlek 900 och inklipp på 600, 700, . . . , 1300 nukleotider.

Ordning 7, fönsterstorlek 900			
Storlek på inklipp	Sensitivitet Sn	Specificitet Sp	Specificitet $Sp*$
600	0,900	0,570	0,880
700	0,900	0,5963	0,920
800	0,930	0,646	0,930
900	0,960	0,649	0,900
1000	0,940	0,603	0,920
1100	0,930	0,589	0,950
1200	0,950	0,699	0,930
1300	0,870	0,588	0,970

Tabell 27: Resultaten från sensitivitets- och specificitetsanalysen på träffnivå för ordning 7, fönsterstorlek 1000 och inklipp på 600, 700, . . . , 1300 nukleotider.

Ordning 7, fönsterstorlek 1000			
Storlek på inklipp	Sensitivitet Sn	Specificitet Sp	Specificitet $Sp*$
600	0,930	0,589	0,880
700	0,890	0,582	0,900
800	0,920	0,634	0,940
900	0,920	0,564	0,900
1000	0,880	0,615	0,910
1100	0,910	0,532	0,930
1200	0,900	0,539	0,950
1300	0,910	0,615	0,950

Tabell 28: Resultaten från sensitivitets- och specificitetsanalysen på träffnivå för ordning 7, fönsterstorlek 1100 och inklipp på 600, 700, . . . , 1300 nukleotider.

Ordning 7, fönsterstorlek 1100			
Storlek på inklipp	Sensitivitet Sn	Specificitet Sp	Specificitet $Sp*$
600	0,930	0,650	0,870
700	0,950	0,725	0,920
800	0,940	0,648	0,930
900	0,940	0,662	0,900
1000	0,930	0,699	0,930
1100	0,910	0,664	0,920
1200	0,930	0,679	0,940
1300	0,930	0,604	0,950

Tabell 29: Resultaten från sensitivitets- och specificitetsanalysen på träffnivå för ordning 7, fönsterstorlek 1200 och inklipp på 600, 700, . . . , 1300 nukleotider.

Ordning 7, fönsterstorlek 1200			
Storlek på inklipp	Sensitivitet Sn	Specificitet Sp	Specificitet $Sp*$
600	0,850	0,708	0,880
700	0,920	0,681	0,910
800	0,920	0,692	0,910
900	0,950	0,748	0,910
1000	0,920	0,676	0,890
1100	0,940	0,671	0,910
1200	0,910	0,674	0,920
1300	0,960	0,727	0,930

Tabell 30: Resultaten från sensitivitets- och specificitetsanalysen på träffnivå för ordning 7, fönsterstorlek 1300 och inklipp på 600, 700, . . . , 1300 nukleotider.

Ordning 7, fönsterstorlek 1300			
Storlek på inklipp	Sensitivitet Sn	Specificitet Sp	Specificitet $Sp*$
600	0,870	0,674	0,900
700	0,930	0,750	0,940
800	0,870	0,630	0,950
900	0,950	0,742	0,920
1000	0,960	0,716	0,910
1100	0,890	0,640	0,910
1200	0,930	0,715	0,930
1300	0,930	0,715	0,940

Tabell 31: Resultaten från sensitivitets- och specificitetsanalysen på träffnivå för ordning 7, fönsterstorlek 1400 och inklipp på 600, 700, . . . , 1300 nukleotider.

Ordning 7, fönsterstorlek 1400			
Storlek på inklipp	Sensitivitet Sn	Specificitet Sp	Specificitet $Sp*$
600	0,910	0,711	0,870
700	0,920	0,667	0,910
800	0,900	0,667	0,920
900	0,950	0,720	0,930
1000	0,940	0,691	0,910
1100	0,860	0,677	0,900
1200	0,930	0,746	0,880
1300	0,850	0,697	0,880

C.2 Övriga resultat från utvärdering av träffar

Tabell 32: Blastresultat från Athena av träffar från SMS-3-5 mot hela genom hos de andra bakterierna.

Intervall	Träff org.	Start	Slut	Täcknings-grad	Träff-längd
233382-235161	*S. aureus*	522565	523605	81.084	1052
233382-235161	*S. pneumoniae*	34426	35462	80.853	1055
233382-235161	*S. scabiei*	4404365	4403293	80.389	1081
233382-235161	*C. pneumoniae*	1001062	1002112	77.976	1067
233382-235161	*B. thetaiotamicron*	3035696	3034780	77.146	932
233382-235161	*A. aelicus*	572280	571330	76.215	967
234935-236004	*S. aureus*	1978277	1977535	79.211	760
234935-236004	*S. pneumoniae*	36112	36858	78.684	760
234935-236004	*B. thetaiotamicron*	4985880	4986629	75.750	767
236405-237580	*S. aureus*	525616	526688	78.623	1104
236405-237580	*S. scabiei*	2857065	2855969	78.018	1110
236405-237580	*S. pneumoniae*	37483	38402	79.656	929
236405-237580	*B. thetaiotamicron*	1627829	1626942	79.847	913
236405-237580	*C. pneumoniae*	1004159	1005083	77.433	935
2819551-2826799	*S. aureus*	1975900	1978277	76.377	2451
2819551-2826799	*S. pneumoniae*	35462	33904	78.607	1594
2819551-2826799	*S. scabiei*	4403293	4404579	80.304	1315
2819551-2826799	*C. pneumoniae*	1002112	1000813	77.441	1321
2819551-2826799	*B. thetaiotamicron*	1626845	1627829	79.010	1010
2819551-2826799	*A. aelicus*	1626845	572512	75.957	1202
3650068-3657193	*S. aureus*	1975900	1978277	76.307	2448
3650068-3657193	*S. pneumoniae*	35462	33904	78.433	1595
3650068-3657193	*S. scabiei*	4403293	4404579	80.304	1315
3650068-3657193	*C. pneumoniae*	1002112	1000813	77.441	1321
3650068-3657193	*B. thetaiotamicron*	1626845	1627829	79.345	1007
3650068-3657193	*A. aelicus*	571330	572512	75.957	1202
4222099-4223478	*S. aureus*	522565	523605	80.722	1053
4222099-4223478	*S. pneumoniae*	34426	35462	80.740	1054
4222099-4223478	*S. scabiei*	4404349	4403293	80.376	1065
4222099-4223478	*C. pneumoniae*	1001063	1002112	77.788	1067
4222099-4223478	*B. thetaiotamicron*	1631156	1630250	76.948	924
4222099-4223478	*A. aelicus*	572256	571330	76.402	924
4225011-4226177	*S. aureus*	525616	526675	78.552	1091
4225011-4226177	*S. scabiei*	2857065	2855989	78.257	1090
4225011-4226177	*S. pneumoniae*	37483	38389	79.694	916
4225011-4226177	*B. thetaiotamicron*	1627829	1626942	79.847	913
4225011-4226177	*C. pneumoniae*	1004159	1005076	77.263	928
4331066-4332441	*S. aureus*	522565	523605	81.007	1053
4331066-4332441	*S. pneumoniae*	34426	35462	80.835	1054
4331066-4332441	*S. scabiei*	4404353	4403293	80.543	1069
4331066-4332441	*C. pneumoniae*	1001062	1002112	77.903	1068
4331066-4332441	*B. thetaiotamicron*	1631166	1630250	77.088	934
4331066-4332441	*A. aelicus*	572256	571330	76.402	945
4333993-4335159	*S. aureus*	525616	526675	78.552	1091
4333993-4335159	*S. scabiei*	2857065	2855989	78.257	1090
4333993-4335159	*S. pneumoniae*	37483	38389	79.694	916
4333993-4335159	*B. thetaiotamicron*	1627829	1626942	79.847	913
4333993-4335159	*C. pneumoniae*	1004159	1005076	77.263	928

Tabell 33: Blastresultat från Athena av träffar från SMS-3-5 mot hela genom hos de andra bakterierna, fortsättning.

Intervall	Träff org.	Start	Slut	Täcknings-grad	Träff-längd
4512972-4514358	*S. aureus*	522565	523605	81.007	1053
4512972-4514358	*S. scabiei*	4404353	4403293	80.543	1069
4512972-4514358	*S. pneumoniae*	34426	35462	80.835	1054
4512972-4514358	*C. pneumoniae*	1001062	1002112	77.903	1068
4512972-4514358	*B. thetaiotamicron*	1631166	1630250	77.088	934
4512972-4514358	*A. aelicus*	572256	571330	76.402	945
4515899-4517073	*S. aureus*	525616	526688	78.533	1104
4515899-4517073	*S. scabiei*	2857065	2855969	78.198	1110
4515899-4517073	*S. pneumoniae*	37483	38402	79.656	929
4515899-4517073	*B. thetaiotamicron*	1627829	1626942	79.847	913
4515899-4517073	*C. pneumoniae*	1004159	1005083	77.433	935
4553143-4554474	*S. aureus*	522565	523605	81.007	1053
4553143-4554474	*S. scabiei*	4404353	4403293	80.543	1069
4553143-4554474	*S. pneumoniae*	34426	35462	80.835	1054
4553143-4554474	*C. pneumoniae*	1001062	1002112	77.903	1068
4553143-4554474	*B. thetaiotamicron*	1631166	1630250	77.088	934
4553143-4554474	*A. aelicus*	572256	571330	76.402	945
4556068-4557243	*S. aureus*	525616	526688	78.623	1104
4556068-4557243	*S. scabiei*	2857065	2855969	78.018	1110
4556068-4557243	*S. pneumoniae*	37483	38402	79.656	929
4556068-4557243	*B. thetaiotamicron*	1627829	1626942	79.847	913
4556068-4557243	*C. pneumoniae*	1004159	1005083	77.433	935

Tabell 34: Blastresultat från Athena av träffar från SMS-3-5 mot träffar hos de andra bakterierna.

Intervall	Träff org.	Start	Slut	Täcknings-grad	Träff-längd
233382-235161	*S. aureus*	521772	527507	81.084	1052
233382-235161	*S. pneumoniae*	33488	39280	80.853	1055
233382-235161	*S. scabiei*	4398373	4406069	80.389	1081
233382-235161	*C. pneumoniae*	1000327	1005614	77.976	1067
233382-235161	*B. thetaiotamicron*	3034417	3036193	77.146	932
233382-235161	*A. aelicus*	567182	573353	76.215	967
234935-236004	*S. aureus*	1973032	1978859	79.211	760
234935-236004	*S. pneumoniae*	33488	39280	78.684	760
234935-236004	*B. thetaiotamicron*	4983226	4988748	75.750	767
236405-237580	*S. aureus*	521772	527507	78.623	1104
236405-237580	*S. scabiei*	2854989	2861647	78.018	1110
236405-237580	*S. pneumoniae*	33488	39280	79.656	929
236405-237580	*B. thetaiotamicron*	1626414	1629443	79.847	913
236405-237580	*C. pneumoniae*	1000327	1005614	77.433	935
2819551-2826799	*S. aureus*	1973032	1978859	76.377	2451
2819551-2826799	*S. pneumoniae*	33488	39280	78.607	1594
2819551-2826799	*S. scabiei*	4398373	4406069	80.304	1315
2819551-2826799	*C. pneumoniae*	1000327	1005614	77.441	1321
2819551-2826799	*B. thetaiotamicron*	1626414	1629443	79.010	1010
2819551-2826799	*A. aelicus*	567182	573353	75.957	1202
3650068-3657193	*S. aureus*	1973032	1978859	76.307	2448
3650068-3657193	*S. pneumoniae*	33488	39280	78.433	1595
3650068-3657193	*S. scabiei*	4398373	4406069	80.304	1315
3650068-3657193	*C. pneumoniae*	1000327	1005614	77.441	1321
3650068-3657193	*B. thetaiotamicron*	1626414	1629443	79.345	1007
3650068-3657193	*A. aelicus*	567182	573353	75.957	1202
4222099-4223478	*S. aureus*	521772	527507	80.722	1053
4222099-4223478	*S. pneumoniae*	33488	39280	80.740	1054
4222099-4223478	*S. scabiei*	4398373	4406069	80.376	1065
4222099-4223478	*C. pneumoniae*	1000327	1005614	77.788	1067
4222099-4223478	*B. thetaiotamicron*	1629887	1631674	76.948	924
4222099-4223478	*A. aelicus*	567182	573353	76.402	924
4225011-4226177	*S. aureus*	521772	527507	78.552	1091
4225011-4226177	*S. scabiei*	2854989	2861647	78.257	1090
4225011-4226177	*S. pneumoniae*	33488	39280	79.694	916
4225011-4226177	*B. thetaiotamicron*	1626414	1629443	79.847	913
4225011-4226177	*C. pneumoniae*	1000327	1005614	77.263	928
4331066-4332441	*S. aureus*	521772	527507	81.007	1053
4331066-4332441	*S. pneumoniae*	33488	39280	80.835	1054
4331066-4332441	*S. scabiei*	4398373	4406069	80.543	1069
4331066-4332441	*C. pneumoniae*	1000327	1005614	77.903	1068
4331066-4332441	*B. thetaiotamicron*	1629887	1631674	77.088	934
4331066-4332441	*A. aelicus*	567182	573353	76.402	945
4333993-4335159	*S. aureus*	521772	527507	78.552	1091
4333993-4335159	*S. scabiei*	2854989	2861647	78.257	1090
4333993-4335159	*S. pneumoniae*	33488	33488	79.694	916
4333993-4335159	*B. thetaiotamicron*	1626414	1629443	79.847	913
4333993-4335159	*C. pneumoniae*	1000327	1005614	77.263	928

Tabell 35: Blastresultat från Athena av träffar från SMS-3-5 mot träffar hos de andra bakterierna, fortsättning.

Intervall	Träff org.	Start	Slut	Täcknings-grad	Träff-längd
4512972-4514358	*S. aureus*	521772	527507	81.007	1053
4512972-4514358	*S. scabiei*	4398373	4406069	80.543	1069
4512972-4514358	*S. pneumoniae*	33488	39280	80.835	1054
4512972-4514358	*C. pneumoniae*	1000327	1005614	77.903	1068
4512972-4514358	*B. thetaiotamicron*	1629887	1631674	77.088	934
4512972-4514358	*A. aelicus*	1191465	1197602	76.402	945
4515899-4517073	*S. aureus*	521772	521772	78.533	1104
4515899-4517073	*S. scabiei*	2854989	2861647	78.198	1110
4515899-4517073	*S. pneumoniae*	33488	39280	79.656	929
4515899-4517073	*B. thetaiotamicron*	1626414	1629443	79.847	913
4515899-4517073	*C. pneumoniae*	1000327	1005614	77.433	935
4553143-4554474	*S. aureus*	521772	527507	81.007	1053
4553143-4554474	*S. scabiei*	4398373	4406069	80.543	1069
4553143-4554474	*S. pneumoniae*	33488	39280	80.835	1054
4553143-4554474	*C. pneumoniae*	1000327	1005614	77.903	1068
4553143-4554474	*B. thetaiotamicron*	1629887	1631674	77.088	934
4553143-4554474	*A. aelicus*	567182	573353	76.402	945
4556068-4557243	*S. aureus*	521772	527507	78.623	1104
4556068-4557243	*S. scabiei*	8425451	8433430	78.018	1110
4556068-4557243	*S. pneumoniae*	33488	39280	79.656	929
4556068-4557243	*B. thetaiotamicron*	1626414	1629443	79.847	913
4556068-4557243	*C. pneumoniae*	1000327	1005614	77.433	935

C.3 Fullständig frekvenstabell för ordning 1

Tabell 36: Fullständig frekvenstabell för ordning 1

STS	SSO	TSO	SSO/TSO	STS	SSO	TSO	SSO/TSO
AAA	108964	338006	0,322	TAA	68858	212024	0,325
GAA	83530	267384	0,312	CAA	76654	325327	0.236
AAT	83021	338006	0,246	TAT	63301	212024	0,299
GAT	86587	267384	0,324	CAT	77041	325327	0,237
AAG	63405	338006	0,188	TAG	27254	212024	0,129
GAG	42503	267384	0,159	CAG	104850	325327	0,322
AAC	82616	338006	0,244	TAC	52611	212024	0,248
GAC	54764	267384	0,205	CAC	66782	325327	0,205
ATA	63721	309950	0,206	TTA	68845	339584	0,203
GTA	52688	255699	0,206	CTA	26770	236149	0,113
ATT	83424	309950	0,269	TTT	109862	339584	0,324
GTT	82622	255699	0,323	CTT	63676	236149	0,270
ATG	76282	309950	0,246	TTG	76998	339584	0,227
GTG	66142	255699	0,259	CTG	102957	236149	0,436
ATC	86523	309950	0,279	TTC	83879	339584	0,247
GTC	54247	255699	0,212	CTC	42746	236149	0,181
AGA	56659	238013	0,238	TGA	83532	322379	0,259
GGA	56222	270252	0,208	CGA	70971	346793	0,205
AGT	49792	238013	0,209	TGT	58397	322379	0,181
GGT	74326	270252	0,275	CGT	73184	346793	0,211
AGG	50653	238013	0,213	TGG	85180	322379	0,264
GGG	47515	270252	0,176	CGG	86904	346793	0,251
AGC	80909	238013	0,340	TGC	95270	322379	0,296
GGC	92189	270252	0,341	CGC	115734	346793	0,334
ACA	58664	256773	0,228	TCA	84101	267395	0,315
GCA	96071	384102	0,250	CCA	86491	271821	0,318
ACT	49886	256773	0,194	TCT	55483	267395	0,207
GCT	80333	384102	0,209	CCT	50447	271821	0,186
ACG	73288	256773	0,285	TCG	71759	267395	0,268
CCG	87076	271821	0,320	GCG	114670	384102	0,299
ACC	74935	256773	0,292	TCC	56051	267395	0,210
GCC	93028	384102	0,242	CCC	47807	271821	0,176