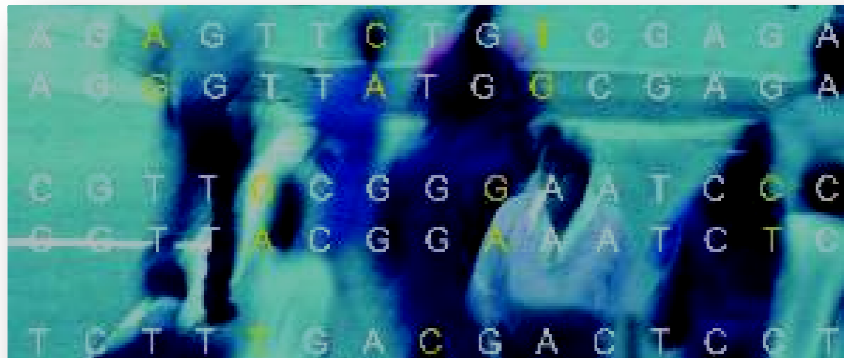
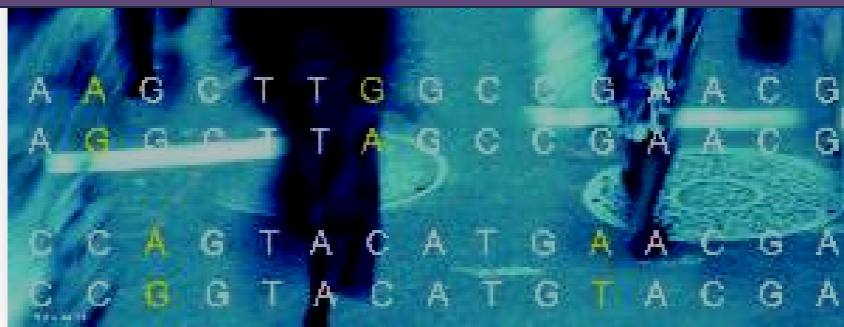




Computational identification of synonymous SNPs in the human genome and their potential role in disease



Lee-Ann Wood
MSc. (Human Genetics)

Wits student number: 330086

Professor Michèle Ramsay
Head of the Molecular Genetics Laboratory and
Professor in the Division of Human Genetics

*COMPUTATIONAL IDENTIFICATION
OF SYNONYMOUS SNPs IN THE
HUMAN GENOME AND THEIR
POTENTIAL ROLE IN DISEASE*

Lee-Ann Wood

Dissertation submitted to the faculty of Health Sciences,
University of the Witwatersrand, Johannesburg, in fulfillment of
the requirements for the degree of Masters of Sciences in
Human Genetics.

Johannesburg, 2012

DECLARATION

I Lee-Ann Wood declare that this dissertation is my own work. It is being submitted for the degree of Masters of Science in Human Genetics in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at this or any other University.

___ day of ___ 20___

PRESENTATIONS ARISING FROM THIS STUDY

Wood, L., Ramsay, M. Computational identification of synonymous SNPs in the human genome and their potential role in disease.

Oral presentations:

- David Landsman Seminar
Johannesburg, South Africa (22 January 2010)
- ISCB Africa ASBCB Conference on Bioinformatics
Cape Town, South Africa (9-11 March 2011)

Poster presentations:

- Second South African Bioinformatics Workshop
Vanderbijlpark, South Africa (11-12 October 2009)
- 1st European Student Council Symposium
Ghent, Belgium (26 September 2010)
- 9th European Conference on Computational Biology
Ghent, Belgium (26-29 September 2010)
- 3rd Postgraduate Cross Faculty Symposium
Johannesburg, South Africa (26-27 October 2010)

Awards:

- Best oral presentation at the ISCB Africa ASBCB Conference on Bioinformatics
Cape Town, South Africa (9-11 March 2011)

ABSTRACT

The potential phenotypic effects of synonymous SNPs (sSNPs) have long been overlooked. Although several sSNPs are no longer thought to be silent, no one has identified which sSNPs may contribute to phenotypic variation on a genome-wide scale. sSNPs that cause a change in codon-usage frequency or mRNA secondary structures may alter translational and protein folding kinetics. In addition, sSNPs that alter splice-site consensus sequences may cause aberrant slicing, which could change the protein product. A sSNP that contributes to any of these molecular mechanisms may thus alter protein structure and function. To computationally identify sSNPs with a potential impact, SynSNP was created. SynSNP is a text-based tool written in Python. All sSNPs published within dbSNP are first identified. SynSNP uses established bioinformatics tools to determine which of the sSNPs may potentially result in a molecular effect. The potentially functional sSNPs are then assessed to determine whether any have previously been associated with a trait or disease in genome-wide association studies (GWAS) and/or occur within genes known to be associated with disease in OMIM (Online Mendelian Inheritance in Man). Of the 90,102 identified sSNPs, 21,086 (23.4%) were predicted to potentially have a functional impact, through one or more of the three molecular mechanisms investigated. Of the sSNPs predicted to potentially have a functional impact, 14 (0.07%) had previously been associated with a trait or disease in GWAS. A subset of 4,057 (19.2%) of the potentially functional sSNPs were within genes known to be associated with disease in OMIM. Only six (0.03%) of the potentially functional sSNPs had previously been associated with a trait or disease in GWAS and occurred within genes known to be associated with disease in OMIM. SynSNP could be developed further to aid the discovery of more sSNPs with a potential functional impact. A significant proportion of sSNPs may have a functional impact and their potential role in disease should therefore not be underestimated or neglected.

ACKNOWLEDGMENTS

It is a pleasure to thank my supervisor, Professor Michèle Ramsay. Without her help and support, this dissertation would not be possible. I would also like to thank Shaun Aron, Ananyo Choudhury and the other members of Wits Bioinformatics, who have made available their support in a number of ways. I am grateful to Professor Michèle Ramsay, the University of the Witwatersrand and the National Research Foundation for funding me during my Masters research.

TABLE OF CONTENTS

DECLARATION	ii
PRESENTATIONS ARISING FROM THIS STUDY	iii
ABSTRACT.....	iv
ACKNOWLEDGMENTS.....	v
TABLE OF CONTENTS.....	vi
LIST OF FIGURES.....	ix
LIST OF TABLES.....	xiv
NOMENCLATURE.....	xvii
GLOSSARY.....	xix
CHAPTER 1: INTRODUCTION.....	1
1.1. Synonymous SNPs: silent or ignored?.....	2
1.1.1. Codon-usage frequency.....	5
1.1.2. mRNA secondary structure and stability	8
1.1.3. Alternative splicing	14
1.2. Computational prediction of the functional impact of mutations	20
1.2.1. Catalogues of biological data – biological databases	20
1.2.2. Viewing data from biological databases – genome browsers.....	24
1.2.3. Retrieval and analysis of biological data.....	26
1.2.4. Computational prediction of biological function.....	27
1.3. Study rationale	37

CHAPTER 2: METHODS.....	39
2.1. SynSNP.....	39
2.2. Identification of sSNPs	41
2.3. sSNS predicted to potentially cause an extreme change in codon-usage frequency....	54
2.4. sSNS predicted to potentially cause an extreme change in the minimum free-energy of local mRNA secondary structures	57
2.5. sSNS predicted to potentially cause an extreme change in the maximum-entropy score of splice-sites	65
2.6. Identification of sSNPs with a potential functional impact	74
2.7. sSNPs predicted to have a potential functional impact, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM	75
CHAPTER 3: RESULTS.....	76
3.1. Identification of sSNPs	77
3.2. sSNS predicted to potentially cause an extreme change in codon-usage frequency....	85
3.3. sSNS predicted to potentially cause an extreme change in the minimum free-energy of local mRNA secondary structures	91
3.4. sSNS predicted to potentially cause an extreme change in the maximum-entropy score of splice-sites	97
3.5. Identification of sSNPs with a potential functional impact	110
3.6. sSNPs predicted to have a potential functional impact, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM	112
CHAPTER 4: DISCUSSION.....	115
4.1. sSNPs predicted to potentially cause an extreme change in codon-usage frequency	117

4.2. sSNPs predicted to potentially cause an extreme change in the minimum free-energy of local mRNA secondary structures.....	119
4.3. sSNPs predicted to potentially cause an extreme change in the maximum-entropy score of splice-sites	123
4.4. sSNPs and disease associations.....	125
4.5. Conclusion	129
APPENDIX A.....	132
APPENDIX B.....	148
APPENDIX C.....	218
APPENDIX D.....	219
REFERENCES	223

LIST OF FIGURES

- Figure 1.1: Three molecular mechanisms by which sSNPs may have a potential functional impact. 3
- Figure 1.2: sSNPs that cause a change in codon-usage frequency may potentially alter translational and protein folding kinetics. 7
- Figure 1.3: Examples of single-stranded and double-stranded regions in RNA secondary structures. 8
- Figure 1.4: sSNPs that alter mRNA secondary structure and/or stability may potentially alter the rate of mRNA degradation or translational and protein folding kinetics. 10
- Figure 1.5: A schematic example of alternative splicing, involving exon skipping. 15
- Figure 1.6: The five basic types of alternative splicing 16
- Figure 1.7: sSNPs within the exonic part of the core splice-sites may alter the final protein structure and function or the abundance of the different protein isoforms. 18

- Figure 2.1: The pipeline used in SynSNP, to identify sSNPs with a potential functional impact, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM. 40
- Figure 2.2: The methods used to identify sSNPs, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM. 43
- Figure 2.3: The screen-shot of the UCSC main table browser in Galaxy, used to retrieve cSNPs from dbSNP131. 44
- Figure 2.4: The screen-shot of the UCSC main table browser in Galaxy, used to retrieve SNPs from dbSNP131. 44
- Figure 2.5: The screen-shot of the tool used in Galaxy to join the cSNP and SNP files. .. 45
- Figure 2.6: Examples of how the reference alleles given by dbSNP131 have to be modified to reflect the reference alleles on the transcript strands. 47

- Figure 2.7: The screen-shot of the UCSC main table browser in Galaxy, used to retrieve the SNPs associated with a trait or disease in GWAS, from the GWAS catalogue. 50
- Figure 2.8: The screen-shot of the UCSC main table browser in Galaxy, used to retrieve the genes known to be associated with disease in OMIM. 50
- Figure 2.9: The screen-shot of the UCSC main table browser in Galaxy, used to retrieve all genes published within OMIM. 51
- Figure 2.10: The screen-shot of the tool used in Galaxy to join the file containing the genes known to be associated with disease in OMIM and the file containing all of the genes published within OMIM. 51
- Figure 2.11: The methods used to identify sSNS that potentially cause an extreme change in codon-usage frequency, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM. 55
- Figure 2.12: The methods used to identify sSNS that potentially cause an extreme change in the minimum free-energy of local mRNA secondary structures, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM. 59
- Figure 2.13: The structure of mature mRNA. 60
- Figure 2.14: The screen-shot of the UCSC main table browser in Galaxy, used to retrieve the coordinates of the coding sequences with introns. 60
- Figure 2.15: The screen-shot of the UCSC main table browser in Galaxy, used to retrieve the sequences of the coding sequences with introns. 61
- Figure 2.16: The screen-shot of the tool used in Galaxy to convert the file containing the sequences of the coding sequences with introns from a FASTA format to a tabular format. 61
- Figure 2.17: The screen-shot of the tool used in Galaxy to join the file containing the coordinates of the coding sequences with introns and the file containing the sequences of the coding sequences with introns. 62

- Figure 2.18: The methods used to identify sSNS that potentially cause an extreme change in the maximum-entropy score of splice-sites, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM. 67
- Figure 2.19: The screen-shot of the UCSC main table browser in Galaxy, used to retrieve the coordinates of the exon with 20 bases at each end sequences..... 68
- Figure 2.20: The screen-shot of the tool used in Galaxy to retrieve the sequences of the exons with 20 bases at each end. 68
- Figure 2.21: Schematic of the exon plus 20 bases at each end sequences retrieved from RefSeq. 70
- Figure 2.22: Examples of the calculations used to determine the positions of the sSNS in the exons. 72
- Figure 2.23: Schematic of the calculations used to identify sSNS within the exonic part of splice-sites. 73
- Figure 2.24: Example of the identification of sSNPs predicted to have a potential functional impact through one ore more of the three molecular mechanisms investigated..... 75

- Figure 3.1: Flow-diagram for the identification of sSNPs, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM. 78
- Figure 3.2: Schematic of the main cause of redundancy in the cSNP file retrieved from dbSNP131, using Galaxy..... 78
- Figure 3.3: Flow-diagram for the identification of sSNS that potentially cause an extreme change in codon-usage frequency, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM. 85
- Figure 3.4: Frequency distribution graph of all the changes in codon-usage frequency caused by sSNS..... 86

- Figure 3.5: The frequency of sSNS predicted to potentially cause an extreme change in codon-usage frequency. 87
- Figure 3.6: Flow-diagram for the identification of sSNS that potentially cause an extreme change in the minimum free-energy of local mRNA secondary structures, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM..... 92
- Figure 3.7: Frequency distribution graph of all the changes in the minimum free-energy of local mRNA secondary structures caused by sSNS..... 95
- Figure 3.8: The frequency of sSNS predicted to potentially cause an extreme change in the minimum free-energy of local mRNA secondary structures..... 96
- Figure 3.9: Flow-diagram for the identification of sSNS that potentially cause an extreme change in the maximum-entropy score of splice-sites, which occur within genes known to be associated with disease in OMIM..... 98
- Figure 3.10: Frequency distribution graph of all the changes in the maximum-entropy score of 3' acceptor splice-sites caused by sSNS. 102
- Figure 3.11: The frequency of sSNS predicted to potentially cause an extreme change in the maximum-entropy score of 3' splice-sites, at the three bases in the exonic part of the 3' splice-sites. 103
- Figure 3.12: The frequency of sSNS predicted to potentially cause an extreme change in the maximum-entropy score of 3' splice-sites. 105
- Figure 3.13: Frequency distribution graph of all the changes in the maximum-entropy score of 5' donor splice-sites caused by sSNS. 106
- Figure 3.14: The frequency of sSNS predicted to potentially cause an extreme change in the maximum-entropy score of 5' splice-sites, at the three bases in the exonic part of the 5' splice-sites. 107
- Figure 3.15: The frequency of sSNS predicted to potentially cause an extreme change in the maximum-entropy score of 5' splice-sites. 109

- Figure 3.16: The proportion of sSNPs predicted to potentially have a functional impact either through an extreme change in codon-usage frequency, the minimum free-energy of local mRNA secondary structures or the maximum-entropy score of splice-sites. ... 111
- Figure 3.17: The number of sSNPs predicted to potentially have a functional impact through one or more of the three molecular mechanisms investigated. 112

- Figure 4.1: The proportion of sSNPs, identified from cSNPs, predicted to have a have a potential functional impact, which had previously been associated with a trait or disease in GWAS and/or occurred within genes known to be associated with disease in OMIM. 126

LIST OF TABLES

• Table 2.1: The table schema of the cSNP data file	45
• Table 2.2: Examples of redundant SNPs within dbSNP131, either one SNP assigned more than one rs number or two different SNPs assigned the same rs number	48
• Table 2.3: Examples of the sSNS identified from the sSNPs.....	53
• Table 2.4: Examples of the calculations used to standardize the codon-usage frequency values in the human codon-usage frequency table	54
• Table 3.1: The 17 sSNPs that have previously been associated with a trait or disease in GWAS and occur within genes known to be associated with disease in OMIM	80
• Table 3.2: The frequency of sSNPs with two or more alleles and the number of sSNS identified	81
• Table 3.3: The frequency of sSNS at the first, second and third positions of codons.....	81
• Table 3.4: The frequency of sSNS changes	82
• Table 3.5: The frequency of amino acids involved in sSNS.....	83
• Table 3.6: The frequency of sSNS on the different chromosomes.....	84
• Table 3.7: The frequency of predicted extreme codon changes from a more frequently used codon to a less frequently used codon (decrease in codon-usage frequency)	88
• Table 3.8: The frequency of predicted extreme codon changes from a less frequently used codon to a more frequently used codon (increase in codon-usage frequency).....	89
• Table 3.9: The three sSNS predicted to potentially cause an extreme decrease in codon-usage frequency, which have previously been associated with a trait or disease in GWAS and occur within genes known to be associated with disease in OMIM	90
• Table 3.10: The three sSNS predicted to potentially cause an extreme increase in codon-usage frequency, which have previously been associated with a trait or disease in GWAS and occur within genes known to be associated with disease in OMIM	90
• Table 3.11: The frequency of coding regions with introns (CWI) on the different chromosomes	93

- Table 3.12: The one sSNS predicted to potentially cause an extreme decrease in the minimum free-energy of a local mRNA secondary structure, which has previously been associated with a trait in GWAS and occurs within a gene known to be associated with disease in OMIM 97
- Table 3.13: The frequency of exons on the different chromosomes 99
- Table 3.14: The frequency of sSNS at the three bases in the exonic part of the 3' acceptor splice-sites..... 100
- Table 3.15: The frequency of sSNS at the three bases in the exonic part of the 5' donor splice-sites..... 101
- Table 3.16: The frequency of sSNPs predicted to have a potential functional impact, which have been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM..... 113
- Table 3.17: The six sSNPs predicted to have a potential functional impact through one or more of the three molecular mechanisms investigated, which have previously been associated with a trait or disease in GWAS and occur within genes known to be associated with disease in OMIM 114

- Table C.1: The human codon-usage frequency table 218

- Table D.1: The frequency of sSNS changes predicted to potentially cause an extreme decrease in codon-usage frequency 219
- Table D.2: The frequency of sSNS changes predicted to potentially cause an extreme increase in codon-usage frequency 219
- Table D.3: The frequency of sSNS changes in the 100 base mRNA sequences 219
- Table D.4: The frequency of sSNS changes predicted to potentially cause an extreme decrease in the minimum free-energy of local mRNA secondary structures 220
- Table D.5: The frequency of sSNS changes predicted to potentially cause an extreme increase in the minimum free-energy of local mRNA secondary structures 220

- Table D.6: The frequency of sSNS changes in the exonic part of the 3' acceptor splice-sites 221
- Table D.7: The frequency of sSNS changes in the exonic part of the 5' donor splice-sites 221
- Table D.8: The frequency of sSNS changes predicted to potentially cause an extreme decrease in the maximum-entropy score of 3' acceptor splice-sites..... 221
- Table D.9: The frequency of sSNS changes predicted to potentially cause an extreme increase in the maximum-entropy score of 3' acceptor splice-sites..... 222
- Table D.10: The frequency of sSNS changes predicted to potentially cause an extreme decrease in the maximum-entropy score of 5' donor splice-sites 222
- Table D.11: The frequency of sSNS changes predicted to potentially cause an extreme increase in the maximum-entropy score of 5' donor splice-sites 222

NOMENCLATURE

ADCY7	Adenylate cyclase 7 gene
APS	Average pain sensitivity
BLAST	Basic local alignment search tool
BLAT	BLAST-like alignment tool
CDS	Coding sequences
CDSN	Corneodesmosin gene
COMT	Catechol- <i>O</i> -methyltransferase gene
cSNP	Coding SNP
CUF	Codon-usage frequency
CWI	Coding regions with introns
ESE	Exonic splicing enhancer
ESS	Exonic splicing silencer
EST	Expressed sequence tag
Exons+20	Exon sequences with an additional 20 bases into the introns at each end of the exons
FAM65C	Family with sequence similarity 65, member C
FTP	File transfer protocol
GP2	Glycoprotein 2 gene
GWAS	Genome-wide association studies
GUI	Graphical user interface
hnRNP	Heterogeneous nuclear ribonucleoprotein
HPS	High pain sensitivity
HSF	Human Splicing Finder
html	Hypertext mark-up language
INDELS	Insertions and deletions
ISE	Intronic splicing enhancer
ISS	Intronic splicing silencer
LPS	Low pain sensitivity
MDR1	Multi-drug resistance 1 gene
ME	The maximum-entropy score of splice-sites
MED	Maximum entropy distribution
MFE	The minimum free-energy of the 100 base local mRNA secondary structures
MIM	Mendelian Inheritance in Man
NCBI	National Centre for Biotechnology Information
NHGRI	National Human Genome Research Institute
NMD	Nonsense mediated mRNA decay
nsSNP	Non-synonymous SNP
OMIM	Online Mendelian Inheritance in Man

PAH	Phenylalanine hydroxylase gene
PC-12 cells	Rat adrenal cells
P-gp	Phospho-glycoprotein
PWM	Position weight matrix
rs	Reference SNP
SCARB1	Scavenger receptor class B type 1 gene
SNP	Single nucleotide polymorphism
snRNA	Small ribonuclear RNA
SR-BI	Scavenger receptor class B type 1
SR-protein	Serine-rich protein
ss	Submitted SNP
sSNP	Synonymous SNP
sSNS	Synonymous single nucleotide substitutions
UCSC	University of California Santa Cruz

GLOSSARY

Coding region	The regions of DNA or RNA that code for proteins
Coding sequence (CDS)	The mRNA sequence, excluding the 5' and 3' untranslated regions
Coding SNP (cSNP)	A SNP within the coding regions that encode proteins
Synonymous SNP (sSNP)	A cSNP that does not change the amino acid encoded
Non-synonymous SNP (nsSNP)	A cSNP that changes the amino acid encoded
Synonymous single nucleotide substitution (sSNS)	One single nucleotide change from a sSNP's reference allele to one of the sSNP's variant alleles
Reference allele	The SNP allele as annotated in the reference genome assembly
Observed alleles	The reference and all of the variant alleles annotated for a SNP
Disease genes	Genes known to be associated with disease in OMIM

CHAPTER 1: INTRODUCTION

Many types of genetic variation contribute to phenotypic variation, such as disease risks and treatment outcome ([Sauna and Kimchi-Sarfaty, 2011](#)). It was assumed that mutations have to change the amino acids of protein sequences to have an impact on phenotypic variation ([Anfinsen, 1973](#)). Due to the degeneracy of the genetic code synonymous mutations do not change the amino acids encoded and were assumed to have little or no phenotypic affect ([Fairbanks and Anderson, 1999](#)). An increase in the knowledge of protein synthesis has brought about the realization that the amino acid sequence alone does not determine the final protein structure and function. The process by which genes make proteins is far more complex. Synonymous mutations that change the DNA, but not the amino acids encoded may influence the final protein structure and function ([Sauna and Kimchi-Sarfaty, 2011](#)).

A growing body of evidence suggests that synonymous mutations may contribute to disease risks. A study of 21,429 single nucleotide polymorphisms (SNPs) associated with disease, curated from 2,113 publications concluded that synonymous SNPs and non-synonymous SNPs had similar likelihood and effect size for disease association ([Chen et al., 2010](#)). Analysis of mutations at the DNA level is not sufficient to correctly identify and characterize synonymous mutations ([Lopez-Bigas et al., 2005](#)). Ignoring these so-called “silent” mutations will hinder the full understanding of the molecular basis of disease.

The objective of this study was to develop a computational approach to identify synonymous SNPs with a potential functional impact. These “silent” mutations, many of which are predicted to have a potential functional impact, may contribute to phenotypic variation, including disease risk.

1.1. Synonymous SNPs: silent or ignored?

SNPs are single nucleotide substitutions with a minor allele frequency of at least one percent in a population ([Risch and Merikangas, 1996](#), [Barnes, 2003](#)). SNPs are only single nucleotide changes, but they may have profound effects when the different alleles of the SNP have functional implications that may influence the phenotype. SNPs are distributed throughout the genome within the intra- and inter-genic regions ([Reumers et al., 2008](#), [Schwender and Ickstadt, 2008](#)). Broadly, SNPs within the protein coding regions of the genome are classified into two main groups: SNPs that change the amino acids encoded, known as non-synonymous SNPs (nsSNPs), and SNPs that do not change the amino acids encoded, known as synonymous SNPs (sSNPs) ([Barnes, 2003](#)).

Much focus has been placed on nsSNPs, as nsSNPs are more likely to have a functional impact on the protein. The potential functional impacts of sSNPs have largely been ignored ([Chamary et al., 2006](#), [Sauna et al., 2007](#)). This is in part due to Anfinsen's principle, which stated that the final protein structure and function is purely determined by the amino acid sequence ([Anfinsen, 1973](#)). Although protein function is highly dependent on protein structure, the structure and function of a protein is not only determined by the amino acid sequence. The way in which a protein folds will determine its structure, which is not only determined by the protein sequence, but also by other factors, such as the cell environment, folding catalysts and other accessory proteins ([Frydman, 2001](#), [Sauna et al., 2007](#)).

Due to Anfinsen's principle, it was assumed that because sSNPs do not change the amino acids encoded, they had little or no phenotypic affect. A better understanding of protein synthesis and folding has lead to the realization that mutations do not need to change the amino acids encoded to have a functional impact ([Sauna and Kimchi-Sarfaty, 2011](#)). sSNPs may affect post transcriptional modifications prior to translation, which may provide the opportunity for sSNPs to alter the final protein structure and function ([Kimchi-Sarfaty et al., 2007](#)).

sSNPs may alter the final protein structure and function through at least three molecular mechanisms (**Fig.1.1**):

1. A change in codon-usage frequency
2. A change in mRNA secondary structure and stability
3. A change in the normal splicing pattern

(Chamary et al., 2006, Parmley and Hurst, 2007, Sauna et al., 2007)

For the purpose of this study, all sSNPs were first identified and grouped for downstream analysis, and nsSNPs were not analyzed and assessed. It is, however, acknowledged that nsSNPs, in addition to causing amino acid change, may also similarly affect downstream processing.

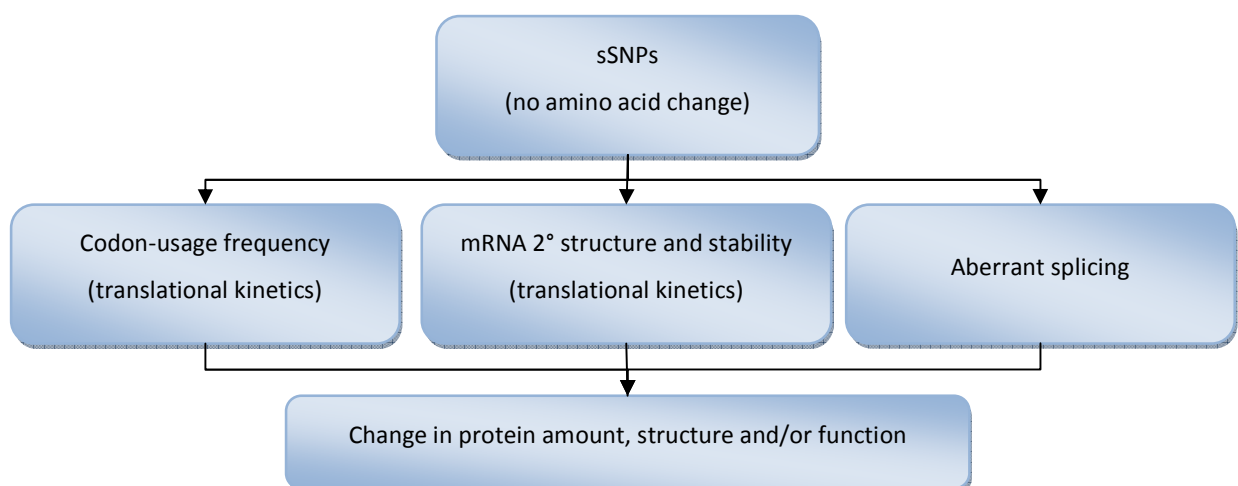


Figure 1.1: Three molecular mechanisms by which sSNPs may have a potential functional impact.

sSNPs that cause a change in any of the three molecular mechanisms may potentially cause a change in the amount of protein product produced or change the structure and/or function of the final protein.

In this study, sSNPs were identified from SNPs published within dbSNP. The sSNPs were grouped and assessed for a potential functional impact, either through a change in codon-usage frequency, a change in mRNA secondary structure and stability or aberrant splicing. To identify sSNPs that are more likely to be of clinical importance, the sSNPs predicted to have a potential functional impact were assessed to determine which occurred within genes known to be associated with disease. The Online Mendelian Inheritance in Man (OMIM) database

was used to retrieve the genes known to be associated with disease (disease genes). OMIM is one of the most established databases for gene and genetic phenotype data. Although the primary focus of OMIM is on Mendelian disorders, OMIM also includes information on select complex traits ([Hamosh et al., 2005](#), [Amberger et al., 2011](#)). The Morbid Map, which is a list of all mapped disorders in OMIM, was used to retrieve the disease genes.

A sSNP that is predicted to have a potential functional impact and is within a disease gene is not necessarily a disease causing sSNP. To get an indication of which sSNPs may be of more functional importance, the sSNPs predicted to have a potential functional impact were assessed to determine whether any had previously been associated with a trait or disease in genome-wide association studies (GWAS). The sSNPs previously associated with a trait or disease in GWAS were retrieved from the GWAS catalog. The GWAS catalog contains the top 50 SNPs associated with a trait or disease from each published GWAS ([Hindorff et al., 2009](#)).

The SNPs that are included in GWAS are chosen based on the SNP's minor allele frequency in the test population, and not based on the SNP's functional impact ([Elbers et al., 2009](#)). For this reason, a SNP that is associated with a trait or disease in GWAS does not mean that the SNP causes the disease or has a functional impact on the disease, but rather that the frequency of the SNP is significantly different in the test population from the control population. When a SNP is associated with a trait or disease in GWAS the causative agent may be:

- The SNP associated with the trait or disease
- An unknown common SNP in linkage-disequilibrium with the associated SNP
- An unknown common SNP or a rare single nucleotide variant, which is tagged to the associated SNP by a haplotype
- A copy number variant that is linked to the associated SNP ([Hindorff et al., 2009](#)).

A SNP that is associated with a trait or disease may not always be the causative agent, but it may give an indication of the region where the disease-causing agent is.

The sSNPs predicted to have a potential functional impact were assessed to determine whether any occurred within disease genes and had previously been associated with a trait or disease in GWAS. These sSNPs are within genes and are significantly associated with the disease; however, these sSNPs may not always be the causative agent. As the sSNPs are in close-proximity to other SNPs, due to linkage-disequilibrium, the potential molecular mechanism of the sSNP may contribute with the effects of the other SNPs to the disease risk. This may especially be true for complex disease, where a number of genetic and environmental factors together contribute to the disease state. It is acknowledged that the sSNPs predicted to have a potential functional impact, which are not within disease genes and/or have not been associated with a trait or disease in GWAS may also be of functional and clinical importance.

1.1.1. Codon-usage frequency

sSNPs may have a functional impact through a change in codon-usage frequency. In theory, it was known that codon bias could affect the final protein structure. It was, however, first demonstrated in 2007, in a study of the multi-drug resistance 1 genes (*MDR1*), that sSNPs could affect the final protein structure and function ([Kimchi-Sarfaty et al., 2007](#)).

The *MDR1* genes encode an ATP-efflux pump that affects multi-drug resistance and the pharmacokinetics of drugs that are phospho-glycoprotein (P-gp) substrates ([Cordon-Cardo et al., 1990](#), [Gottesman et al., 2002](#)). The *MDR1* gene contains at least 50 SNPs, but one of particular interest is a sSNP (C3435T) within exon 26 ([Hoffmeyer et al., 2000](#)). The C3435T sSNP (rs1045642) is within a codon that encodes isoleucine. This sSNP is part of the most common human *MDR1* haplotype, which consists of three SNPs in tight linkage: C1236T - G2677T - C3435T (sSNP-nsSNP-sSNP) ([Horinouchi et al., 2002](#), [Tang et al., 2002](#)). The C3435T sSNP is within the second ATP-binding domain of MDR1. The C allele is observed most frequently across different population groups and is considered the reference allele ([Hoffmeyer et al., 2000](#)).

Although the variant T allele does not influence the relative *MDR1* mRNA or protein levels, compared to the reference allele, the T allele reduces susceptibility to P-gp inhibitors, cyclosporin A and verapamin. Using antibodies that are sensitive to conformational changes and trypsin digestion, it was realized that the reference and variant *MDR1* alleles encode proteins with different tertiary structures and binding abilities (Kimchi-Sarfaty et al., 2007). The differences in the tertiary structures and the binding abilities of the *MDR1* proteins are dependent on the presence of either a cytosine or a thymine at the position of the sSNP in the mRNA. How is it possible that a single sSNP may alter the structure and function of the *MDR1* protein? It was hypothesized that this may be because of the substitution of cytosine by the thymine, that results in a codon that is used less frequently than the codon usually found in that position (Kimchi-Sarfaty et al., 2007).

Due to the degeneracy of the genetic code, a single amino acid can be encoded by more than one codon (Fairbanks and Anderson, 1999). Synonymous codons are not used randomly or in equal frequencies (Sharp and Li, 1987, Karlin and Mrazek, 1996, Duret, 2002). The use of a codon is dependent on the abundance of the codon's corresponding *iso*-acceptor tRNA (Bulmer, 1987). Frequently used codons have a higher abundance of available *iso*-acceptor tRNAs, thus the tRNA can be quickly recruited and rapid translation can occur (Purvis et al., 1987, Karlin and Mrazek, 1996). In the case of *MDR1*, the variant 3435T allele changes the codon from a more frequently used codon to a less frequently used codon (Kimchi-Sarfaty et al., 2007). Less frequently used codons have a lower abundance of available *iso*-acceptor tRNAs, thus the ribosome is stalled while the tRNA is recruited (Purvis et al., 1987, Lavner and Kotlar, 2005). The pause in translation may provide additional waiting time, where the growing polypeptide may fold into a different conformation (Gu et al., 2004, Tsai et al., 2008). As protein folding occurs through distinct folding pathways, if the pause in translation occurs for an extended period, it may result in a distinct energy minima at the bottom of the protein folding funnel. The pause in translation may contribute to the protein being kinetically trapped, resulting in a protein of altered structure and function (**Fig.1.2**) (Tsai et al., 2008).

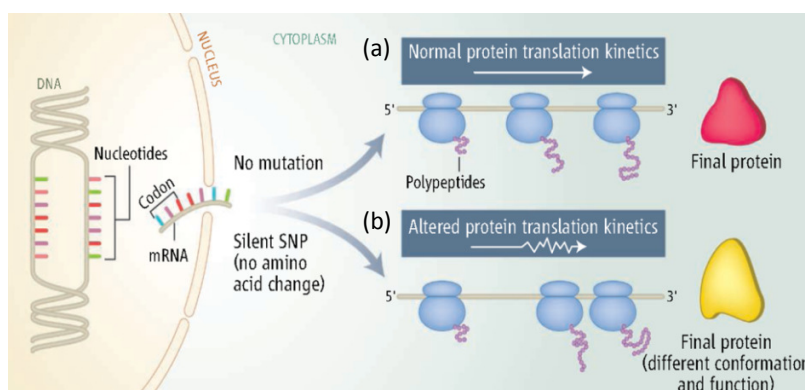


Figure 1.2: sSNPs that cause a change in codon-usage frequency may potentially alter translational and protein folding kinetics.

(a) Normal translational and protein folding kinetics

(b) A change in translational and protein folding kinetics caused by a sSNP, which potentially results in a protein of altered structure and function (Komar et al., 1999).

The effect of C3435T on co-translational folding may influence the formation of the Q-loop. The Q-loop is an essential part of the ATP-binding domain, which governs nucleotide binding (Fung and Gottesman, 2009). The effect of C3435T on co-translational folding may also influence the second walker-A domain, which is responsible for the binding of ATP. The other sSNP (C1236T) in the common *MDR1* haplotype does not affect protein expression or function; however, the variant 1236T allele also results in a codon of decreased usage frequency. The decrease in codon-usage frequency may result in ribosome stalling and influence contranlational folding of TM6. TM6 is required for antibody recognition and substrate binding (Zhou et al., 1999). Although G2677T is a nsSNP, it may also contribute to ribosome stalling through a decrease in codon-usage frequency (Fung and Gottesman, 2009).

Co-translational folding of the ATP domains is essential for the function of the *MDR1* protein (Fung and Gottesman, 2009). Each of the three SNPs within the common *MDR1* haplotype may contribute to a pause in translation, in an additive or synergistic manner. The pause in translation may influence co-translational folding, which may result in a protein of altered structure and function. The effects of each SNP may be independent, but when present together may have a profound effect on the drug affinity of *MDR1* (Fung and Gottesman, 2009).

1.1.2. mRNA secondary structure and stability

Single-stranded mRNA can fold back on itself to form secondary structures. mRNA secondary structures consist of double-stranded regions (helices/stems), separated by single-stranded regions, including loops, bulges and junctions (**Fig.1.3**) (Zuker and Stiegler, 1981, Mount, 2004). In-order for single-stranded mRNA to form secondary structures, a downstream region within the mRNA sequence has to be complementary to a region upstream in the mRNA sequence (Mount, 2004). The double-stranded regions contribute energetic stability to the structure, while the single-stranded regions destabilize the adjacent double-stranded regions (Tinoco et al., 1971). GC base-pairs bind with three hydrogen-bonds and contribute the most energetic stability. AU base-pairs bind with two hydrogen-bonds, which contribute less energetic stability. GU base-pairs bind with one hydrogen bond and contribute the least amount of energetic stability to the mRNA secondary structures (Tinoco et al., 1971, Tinoco and Bustamante, 1999, Mount, 2004).

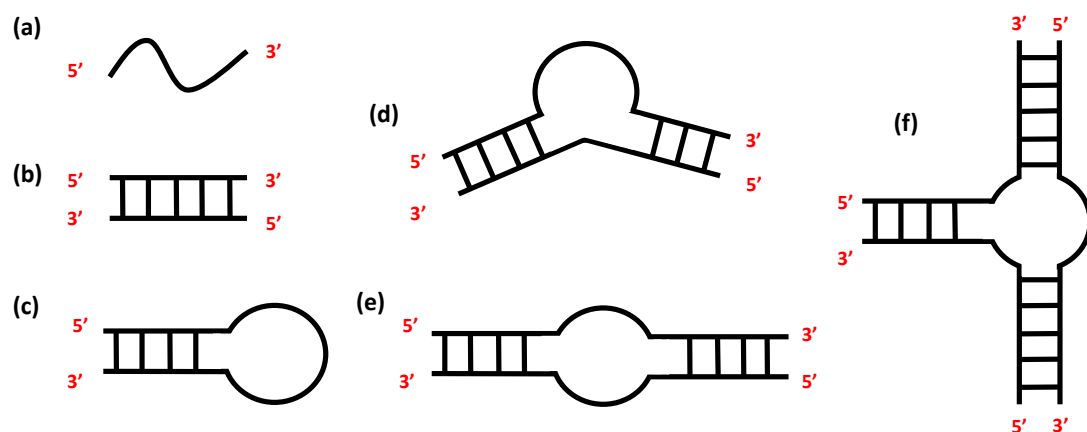


Figure 1.3: Examples of single-stranded and double-stranded regions in RNA secondary structures.

- (a) Single stranded RNA
- (b) Double-stranded RNA (stem/helix)
- (c) Stem-loop/hairpin loop
- (d) Bulge loop
- (e) Interior loop
- (f) Junctions

The stability of the mRNA secondary structures is defined as the amount of energy released or required to form base-pairs, known as free-energy. The free-energy of an mRNA secondary structure is determined by:

- the number of base-pairs in the double-stranded regions, which contribute stabilizing energy to the structure
- the number of bases in the single-stranded regions, which contribute destabilizing energy to the structure
- the proportion of GC, AU and GU base-pairs (Tinoco et al., 1971).

The most likely mRNA secondary structure is often defined as the most stable structure (Tinoco and Bustamante, 1999). The most stable mRNA secondary structure is the structure with the highest proportion of base-pairs and the most energetic stability (minimum free-energy). Various transient mRNA secondary structures may form on the path to the minimum free-energy structure. Some mRNA species may remain kinetically trapped in the transient form, thus the structure with the lowest free-energy is not always the true structure that is formed (Tinoco and Bustamante, 1999).

mRNA secondary structures may reduce the rate of mRNA degradation, through an increase in the stability of the mRNA (Chamary et al., 2006, Hollams et al., 2002). mRNA that has a reduced rate of degradation may be translated repeatedly, increasing the amount of protein expressed. The mRNA secondary structures may also influence the rate of translation, without influencing the amount of protein product produced (Sauna et al., 2007). The mRNA secondary structures need to be unfolded before the ribosome can proceed along the mRNA during translation elongation. The pause in translation, due to ribosome stalling may provide additional waiting time, where the growing polypeptide may fold into a different conformation (Tsai et al., 2008). If the incorrectly folded polypeptide remains kinetically trapped it may result in a protein of altered structure and function (**Fig.1.4**) (Takyar et al., 2005, Tsai et al., 2008).

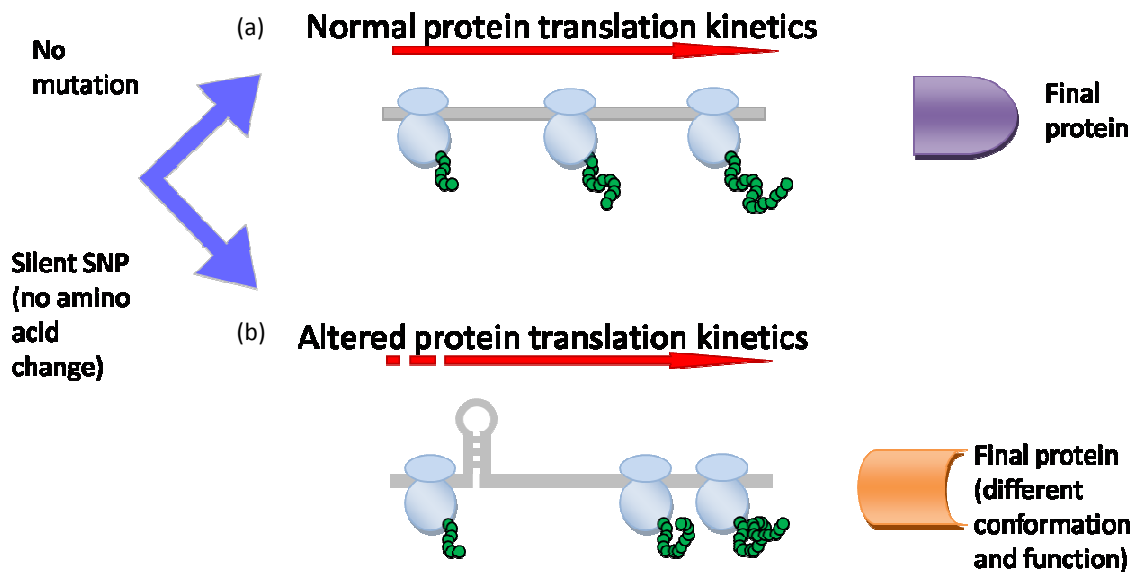


Figure 1.4: sSNPs that alter mRNA secondary structure and/or stability may potentially alter the rate of mRNA degradation or translational and protein folding kinetics.

(a) Normal translational and protein folding kinetics

(b) The presence of a local mRNA secondary structure stalls the ribosome while the secondary structure is being unfolded, which may alter co-translational folding, potentially resulting in a protein of altered structure and function

sSNPs that contribute to the formation or the removal of pairing bases within mRNA secondary structures may alter the structure and stability of the mRNA (Sauna et al., 2007, Biro, 2008). These sSNPs may influence the rate of mRNA degradation or the rate of translation. sSNPs that influence the rate of mRNA degradation could potentially result in a change in the amount of protein product produced (Hollams et al., 2002). sSNPs that influence the rate of translation may influence co-translational folding, which could result in a protein of altered structure and function (Capon et al., 2004, Constantineau et al., 2009). A sSNP that alters the final protein product, through a change in mRNA secondary structure and stability could contribute to a disease state (Hollams et al., 2002).

Examples of the effects of sSNPs on mRNA secondary structure and stability

Example 1

Catechol-*O*-methyltransferase (COMT) is an enzyme responsible for the breakdown of catecholamines, which control pain-sensitivity in humans (Marbach and Levitt, 1976, Diatchenko et al., 2005). The three common human pain sensitivity haplotypes, namely the low pain-sensitivity (LPS), average pain-sensitivity (APS) and high pain-sensitivity (HPS) haplotypes are determined by a combination of four SNPs (Mannisto and Kaakkola, 1999, Diatchenko et al., 2005, Nackley et al., 2006). One of the SNPs is within the *COMT* promoter region and three are within coding region of *COMT*: his62his (C/T, rs45633), leu136leu (C/G, rs4818) and val158met (A/G, rs4680). The LPS haplotype consists of GCGG. The APS haplotype consists of ATCA and the HPS haplotype consists of ACCG (Mannisto and Kaakkola, 1999, Diatchenko et al., 2005, Nackley et al., 2006).

Both the LPS and HPS haplotypes encode val158, thus the nsSNP cannot alone be responsible for the difference between the two most extreme pain-sensitivity phenotypes (Mannisto and Kaakkola, 1999, Diatchenko et al., 2005, Nackley et al., 2006). This suggests that the nsSNP may interact with other SNPs to contribute to pain sensitivity. It was ruled out that the SNP within the promoter region contributes to pain-sensitivity. The sSNPs within the coding region of *COMT* must therefore contribute to pain-sensitivity (Mannisto and Kaakkola, 1999, Diatchenko et al., 2005, Nackley et al., 2006).

sSNPs may contribute to changes in mRNA secondary structure and stability. The mRNA secondary structures of the LPS, APS and HPS mRNA transcripts were computationally predicted using mFold and Afold (Mannisto and Kaakkola, 1999, Diatchenko et al., 2005, Nackley et al., 2006). The LPS haplotype was predicted to code for the shortest and least stable local stem structure in the val158 region. The HPS haplotype was predicted to code for the longest and most stable local stem structure (Mannisto and Kaakkola, 1999, Diatchenko et al., 2005, Nackley et al., 2006).

To confirm the computational predictions, the full-length COMT cDNA clones were constructed in mammalian expression vectors (Mannisto and Kaakkola, 1999, Diatchenko et al., 2005, Nackley et al., 2006). The constructs only differed in the three nucleotides that corresponded to the LPS, APS and HPS haplotypes. The constructs were transiently transfected into rat adrenal (PC-12) cells and the COMT enzymatic activity, mRNA abundance and protein expression measured. The cells expressing the HPS constructs had reduced COMT enzymatic activity and protein expression, in comparison to the cells expressing the LPS constructs. The cells expressing the APS constructs only had a moderate decrease in COMT enzymatic activity and no difference in protein expression. The APS haplotype is the only *COMT* pain-sensitivity haplotype that contains the A allele of the nsSNP (met158 allele). The moderate reduction in pain-sensitivity produced by the APS haplotype may thus be due to the nsSNP, which has previously been reported to decrease protein thermostability (Mannisto and Kaakkola, 1999, Diatchenko et al., 2005, Nackley et al., 2006).

The mRNA abundance and rate of mRNA degradation did not parallel the decrease in the COMT enzymatic activity and protein expression in cells expressing the HPS constructs (Mannisto and Kaakkola, 1999, Diatchenko et al., 2005, Nackley et al., 2006). The difference in the COMT enzymatic activity and protein expression may thus have been due to the differences in the local mRNA secondary structures. Single and double site-directed mutagenesis of the nucleotides that base-pair in the stem-loop structure of the HPS mRNA was performed. The single site-directed mutagenesis destroyed the stable stem-loop structure of the mRNA corresponding to the HPS haplotype. This changed the HPS local mRNA secondary structure to an LPS like structure. When the HPS construct containing the single-site directed mutagenesis was transiently transfected into PC-12 cells, the cells showed an increase in COMT enzymatic activity and protein levels, similar to that of cells expressing the LPS constructs. Double site-directed mutagenesis reconstructed the stable stem-loop structure. When these constructs were transiently transfected into PC-12 cells, the cells showed a decrease in COMT enzymatic activity and protein levels, similar to that of cells expressing the HPS constructs. The sSNPs within the common *COMT* haplotype may

thus change the COMT mRNA secondary structure, which may be responsible for the differences between the two most extreme pain-sensitivity phenotypes ([Mannisto and Kaakkola, 1999](#), [Diatchenko et al., 2005](#), [Nackley et al., 2006](#)).

Example 2

Corneodesmosin (CDSN) is up-regulated in the skin lesions of patients with psoriasis. Psoriasis is an inflammatory skin disorder, characterized by the disruption of the terminal differentiation of keratinocytes ([Haftek et al., 1997](#), [Barker, 1998](#), [Jonca et al., 2002](#)). The *CDSN* gene is highly polymorphic, however a sSNP (*971T) (rs number unknown) within the coding region of *CDSN* has been significantly associated with psoriasis ([Capon et al., 2004](#)). The risk-bearing T allele increases the stability of the CDSN mRNA. The increase in the CDSN mRNA stability may be due to the proximity of the sSNP to upstream mRNA stability/instability motifs. These mRNA stability/instability motifs overlap with cytoplasmic RNA protein sites. The proximity of the sSNP to the mRNA stability/instability motifs may thus cause a reduced affinity for the 39kDa RNA binding protein. The reduced affinity for the RNA binding protein may reduce the stability of the mRNA secondary structure. The possibility that other SNPs may contribute to the up-regulation of *CDSN* has not been ruled-out, however it seems that the sSNP may contribute to an increase in the CDSN mRNA stability. The increase in the mRNA stability may reduce the rate of mRNA degradation, which may cause the increased expression of CDSN in the skin lesions of patients with psoriasis ([Capon et al., 2004](#)).

Example 3

Scavenger-receptor class B type 1 (SR-BI) is expressed from the *SCARB1* gene in the liver and steroidogenic tissues ([Rigotti et al., 1996](#)). SR-BI plays a role in the selective uptake of cholesteryl esters ([Cao et al., 1999](#)). A sSNP (rs5888) within exon 8 of the *SCARB1* gene may be associated with decreased SR-BI protein levels. A decrease SR-BI protein levels may reduce cholesteryl ester uptake, which may alter lipid metabolism ([Acton et al., 1999](#), [Roberts et al., 2007](#)). The sSNP has no effect on the relative mRNA levels; however, *in-silico* and *in-vitro* studies suggest that the variant allele alters the SR-BI mRNA secondary structure

near the sSNP (Constantineau et al., 2009). The decrease in SR-BI protein levels may be due to changes in the SR-BI mRNA secondary structure, induced by the sSNP. The sSNP may reduce the stability of the mRNA secondary structure; this may increase the rate of mRNA degradation. An increase in the rate of mRNA degradation may prevent the mRNA from being translated repeatedly, which may result in a decrease in the amount of SR-BI protein expressed and reduced cholesteryl ester uptake (Constantineau et al., 2009).

1.1.3. Alternative splicing

Splicing is an intricate process in which the intervening introns are removed from the pre-mRNA and the exons are joined to form the mature mRNA (Chabot, 1996, Maniatis and Tasic, 2002). In some cases the intron-exon junctions are not clearly defined, thus a specific sequence may be recognized as an exon in one transcript, but excluded as part of an intron in another transcript, this is known as alternative splicing (**Fig.1.5**). Alternative splicing is a regulatory process that allows a number of different transcripts to be formed by a single gene (Chabot, 1996, Maniatis and Tasic, 2002, Black, 2003). It is estimated that at least 60% and possibly as many as 90% of all genes within the human genome are alternatively spliced (Lander et al., 2001, Johnson et al., 2003, Matlin et al., 2005, Keren et al., 2010).

Each intron-exon junction is defined by 5' donor and 3' acceptor splice-sites (Matlin et al., 2005). These splice-sites may vary slightly in sequence and length, but are generally well conserved. The 5' donor splice-site is generally nine bases in length, with a conserved GU at the start of the downstream intron. The 3' acceptor splice-site is generally 23 bases in length, with an AG at the end of the upstream intron (Matlin et al., 2005).

Recognition of the 5' donor and 3' acceptor splice-sites by the spliceosome is essential for splicing to occur (Fairbanks and Anderson, 1999). The spliceosome is a large ribonucleoprotein complex that performs splicing. The spliceosome consists of more than a hundred core proteins and five small ribonuclear RNAs (snRNAs: U1, U2, U4, U5 and U6)

(Fairbanks and Anderson, 1999, Zhou et al., 2002, Nilsen, 2003). The intron-exon boundaries are first defined by U1 binding to the GU of the 5' donor splice-site and U2 binding to the branch point sequence near the end of the 3' end of the intron. U4, U5 and U6 then bind. U6 replaces U1, catalytically releasing U1 and U4. The 5' end of the intron is then cleaved and joined to the branch-point sequence by a 2',5'-phosphodiester linkage. The 3' end is cleaved and the two exons joined by a phosphodiester bond. The intron is released and degraded (Fairbanks and Anderson, 1999, Black, 2003).

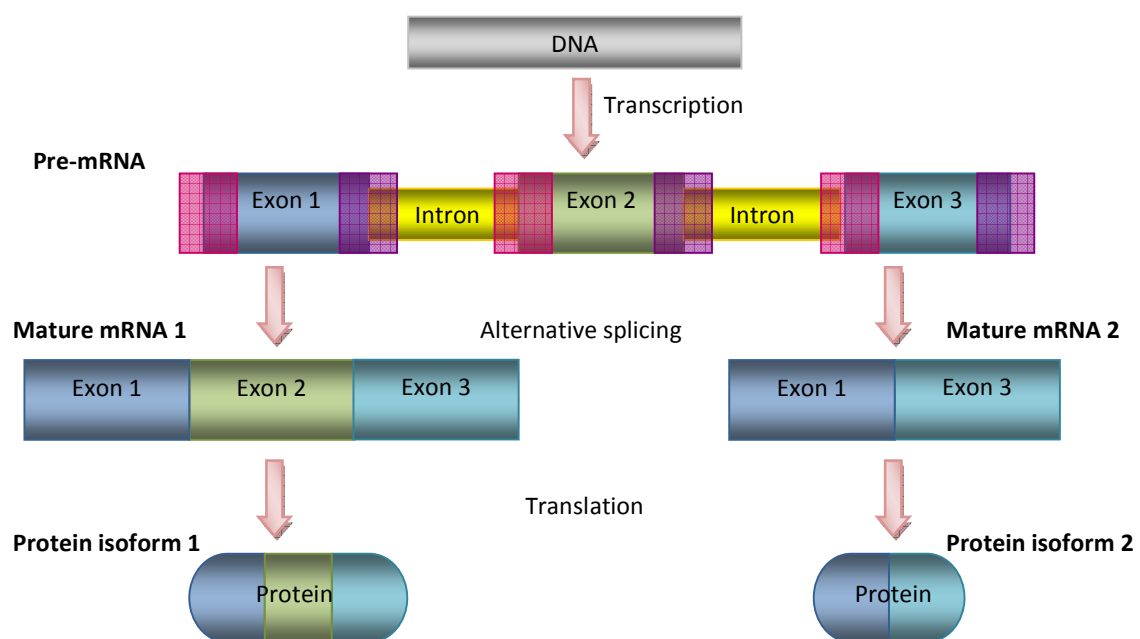


Figure 1.5: A schematic example of alternative splicing, involving exon skipping.

Alternative splicing allows different protein isoforms to be produced from the same mRNA transcript.

 = 3' splice-site

 = 5' splice-site

There are at least five basic types of alternative splicing (**Fig.1.6**):

1. **Exon skipping** is the most common form of alternative splicing in mammals. In exon skipping, a specific exon may be retained or skipped depending on the situation. For instance, a specific exon may be retained in one tissue type, but skipped in another tissue type.

2. **Mutually exclusive exons** is when either one of two exons is retained, but never both.
3. **Alternative donor sites** is when an alternative 5' donor splice-site is used, which alters the boundary of the upstream 3' acceptor splice-site.
4. **Alternative acceptor sites** is when an alternative 3' acceptor splice-site is used, which alters the boundary of the downstream 5' donor splice-site.
5. **Intron retention** is the rarest form of alternative splicing in mammals. In intron retention, an intronic sequence is retained as an exon. The retained sequence needs to be in-frame with the adjacent exons, to prevent a frame-shift mutation that would disrupt the coding sequence.

These five basic types of alternative splicing may be combined to form more complex forms of alternative splicing (Black, 2003, Matlin et al., 2005, Sammeth et al., 2008).

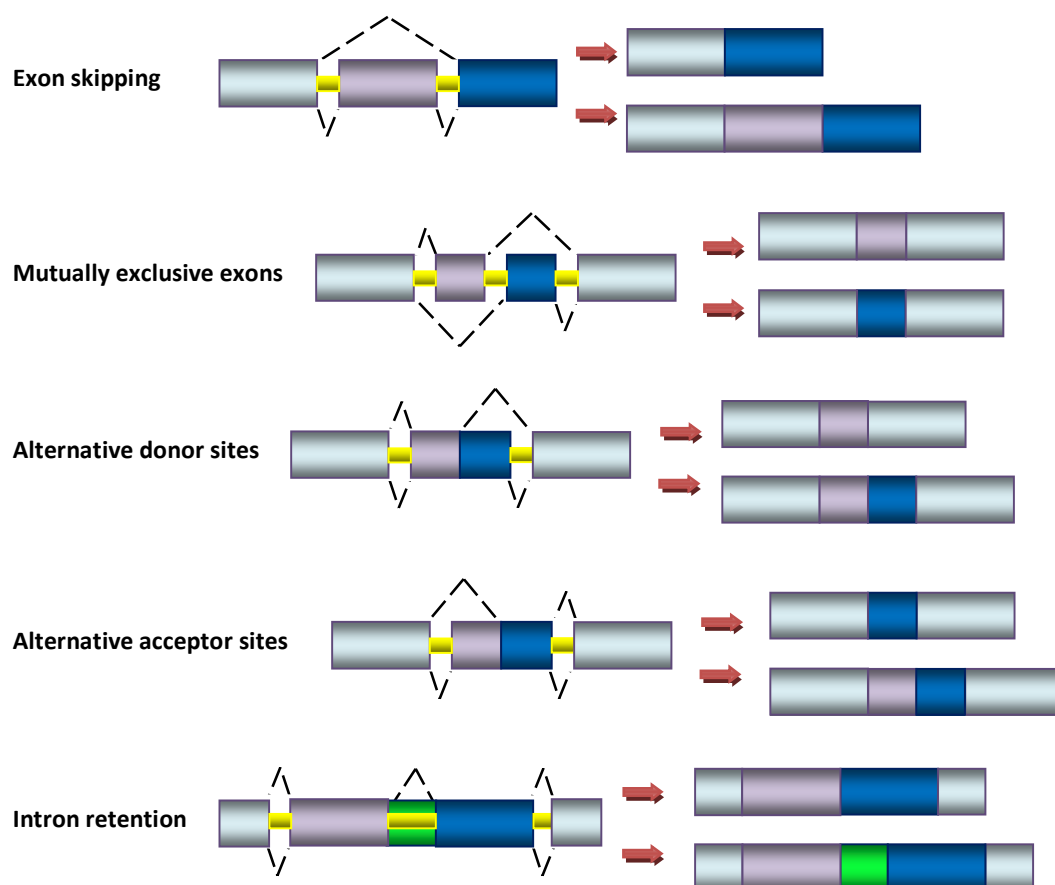


Figure 1.6: The five basic types of alternative splicing

large boxes = exons

small boxes = introns

 = splicing positions

The strengths of authentic splice-sites were computationally compared with the strength of alternative splice-sites (Roca et al., 2003). The splice-sites used were not necessarily the splice-sites with the highest splicing signal strength. Although the alternative splice-sites generally had a weaker signal, they were sometimes more conserved than the authentic splice-sites. This suggests that the core splice-sites are not always sufficient for defining which intron-exons boundaries are spliced. Alternative exons may have a higher requirement for regulation (Roca et al., 2003, Itoh et al., 2004). Splicing can thus only be fully understood through the identification of the core splice-sites as well as the splicing regulatory regions.

Besides the core splice-sites, alternative splicing is further regulated by *trans*-acting factors that bind to *cis*-acting elements (Cartegni et al., 2002, Matlin et al., 2005). Exonic splicing enhancers (ESEs) and intronic splicing enhancers (ISSs) are *cis*-acting regulatory regions that are recognized by *trans*-acting activator proteins. These activator proteins are generally from the serine-rich (SR) protein family, which promote the use of adjacent splice-sites. Exonic splicing silencers (ESSs) and intronic splicing silencers (ISSs) are *cis*-acting regulatory regions that are recognized by *trans*-acting silencer proteins. These silencer proteins are generally from the heterogeneous nuclear ribonucleoprotein (hnRNP) protein family, which inhibit the use of adjacent splice-sites (Matlin et al., 2005, Wang and Burge, 2008). These splicing regulatory regions are five to ten nucleotides in length and have variable consensus sequences depending on the type of protein they bind. Despite their small size, these *cis*-acting regulatory motifs are essential for the regulation of alternative splicing. In some cases, a weak splice-site will be used as opposed to a splice-site with a stronger signal, due to its proximity to a splicing enhancer. Sometimes a splice-site with a stronger signal strength will be ignored due to its proximity to a splicing silencer (Matlin et al., 2005).

Although alternative splicing may result in subtle changes of only a few amino acids, it often inserts or removes an entire protein domain, rather than disrupt a protein domain (Kriventseva et al., 2003). This suggests that alternative splicing is a highly regulated process.

The core splicing signals and regulatory elements act together to prevent detrimental changes, while maintaining protein diversity (Lee and Wang, 2005). Mutations within the core splice-sites or regulatory element may disrupt the normal splicing pattern, potentially resulting in the inclusion or exclusion of large coding regions (Orban and Olah, 2001, Chamary et al., 2006). A disruption of the normal splicing pattern could dramatically alter the final protein structure and function, which could potentially contribute to a disease state. The most dramatic effects result from mutations that introduce a premature termination codon. The premature termination codon activates the nonsense-mediated mRNA decay (NMD) process, causing the degradation of the entire transcript (Nagy and Maquat, 1998, Maquat, 2004). sSNPs within the exonic part of the splice-sites or within ESEs or ESSs could have a dramatic effect on the final protein structure and function through the disruption of the required splicing signals (Fig.1.7) (Zhang and Chasin, 2004, Matlin et al., 2005).

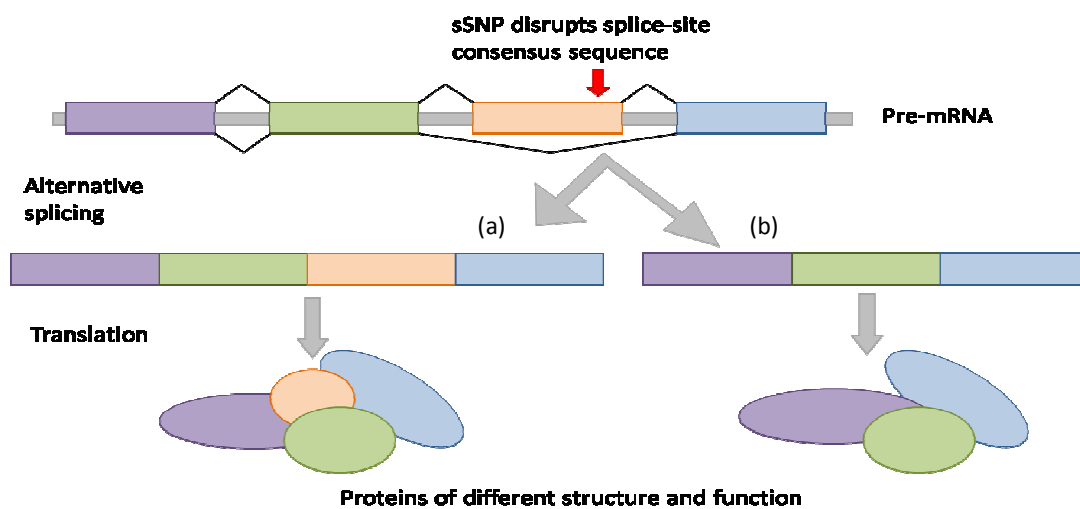


Figure 1.7: sSNPs within the exonic part of the core splice-sites may alter the final protein structure and function or the abundance of the different protein isoforms.

■ (larger boxes) = exons

■ (smaller boxes) = introns

^ = splicing positions

(a) The mRNA produced after normal splicing

(b) The mRNA produced after aberrant splicing, due to a sSNP in a core splice-site

Examples of the effects of sSNPs on alternative splicing

Example 1

α -thalassemia is a rare genetic defect characterized by the decrease or absence of α -globin gene expression (Hartevelde et al., 2004). A silent point mutation (rs number unknown) was identified within codon 22 of the α_2 -globin gene of a 79-year old Surinamese-Hindustani woman. This silent variant activates a putative donor splice-site, causing alternative splicing. This results in a downstream premature termination codon at codons 48 to 49. The premature termination codon activates the NMD process, which is responsible for the α -thalassemia phenotype (Hartevelde et al., 2004).

Example 2

The encoding glycoprotein 2 (*GP2*) gene plays a critical role in intraductal formation during chronic pancreatitis (Freedman et al., 1993). A sSNP (rs number unknown) within exon 9 of the *GP2* gene may prevent idiopathic chronic pancreatitis in French Caucasians. This sSNP prevents the inclusion of exon 9, which results in the removal of 63 of the 537 amino acids encoded by *GP2*, within a critical functional domain and transmembrane helix. This dramatically alters the final protein encoded by *GP2*, resulting in a loss of function. The loss of *GP2* function may provide protection against chronic pancreatitis (Freedman et al., 1993, Boulling et al., 2010).

Example 3

Phenylketonuria is a disease characterized by abnormal phenylalanine metabolism (Dobrowolski et al., 2010). Phenylketonuria is due to mutations in the phenylalanine hydroxylase (*PAH*) gene. A rare synonymous variant within exon 1 of the *PAH* gene (rs number unknown) is in close proximity to an intrinsically weak 5' donor splice-site. This synonymous variant activates a putative ESS motif, which promotes the binding of hnRNP splicing inhibitory proteins. The ESS prevents the identification of the 5' splice-site by the spliceosome, resulting in the exclusion (exon skipping) of exon 1. The exon skipping may be responsible for the phenylketonuria phenotype (Dobrowolski et al., 2010).

The examples above provide evidence for the potential functional role of sSNPs, either through a change in codon-usage frequency, mRNA secondary structure and stability or aberrant splicing. Due to advances in bioinformatics these three molecular mechanisms, by which sSNPs may have a functional impact, can be investigated computationally.

1.2. Computational prediction of the functional impact of mutations

There is now abundant evidence that sSNPs are not silent. Although functional sSNPs have been identified on a case-by-case basis, no one has identified which sSNPs may have a potential functional impact on a genome-wide scale. The completion of the human genome assembly and the development of computational tools which enable the organization and integration of large amounts of data, have made this type of large-scale study possible (Lander et al., 2001, Venter et al., 2001, Kidd et al., 2008). The collaborations of organizations and individuals have made it possible to catalogue a vast amount of genetic data into various databases. These biological databases facilitate large-scale studies, including the identification and annotation of variants within the genome.

1.2.1. Catalogues of biological data – biological databases

Numerous databases have been established for the storage and retrieval of genetic data. Two of the most important biologic databases are RefSeq and dbSNP. RefSeq is the central database for non-redundant nucleotide and protein sequences (Pruitt et al., 2000). dbSNP is the central repository for single nucleotide genetic variation (Sherry et al., 1999).

The RefSeq database is a curated, non-redundant database of nucleotide and protein sequences with annotations, for a variety of species (<http://www.ncbi.nlm.nih.gov/RefSeq/>). RefSeq provides the foundation for sequence and function integration (Pruitt et al., 2000,

Pruitt and Maglott, 2001). The National Centre for Biotechnology Information (NCBI) develops the RefSeq records from the redundant GenBank records, collaborating groups, feedback from the scientific community, computational analysis and manual curation at NCBI. The genome is annotated using transcript alignment, protein support and algorithms based on Hidden Markov Models. The RefSeq records contain GenBank data, annotated genes, transcript and protein data, as well as curation information. RefSeq also provides links to external sources, such as Entrez Gene and variation information from dbSNP. Data can be retrieved from the RefSeq database through Entrez queries, BLAST (basic local alignment search tool), FTP (file transfer protocol) and other provided links. The main aim of RefSeq is to provide accurate, full-length sequence data, thus, much focus is placed on manual curation (Pruitt et al., 2000, Pruitt and Maglott, 2001).

dbSNP was established in 1998 by NCBI, in collaboration with the National Human Genome Research Institute (NHGRI) (<http://www.ncbi.nlm.nih.gov/projects/SNP/>). dbSNP contains data on single nucleotide substitutions, small insertions and deletions (INDELS), microsatellite repeats and small insertions such as transposons (Sherry et al., 1999, Smigielski et al., 2000). The mutation records within dbSNP are for neutral and disease causing polymorphisms. In the context of dbSNP, the term "SNP" reference to any variation involving a small number of nucleotides. Submissions to dbSNP are accepted from any source within the private or public sectors. Each new submission to dbSNP is assigned a unique ss (submitted SNP) number. The ss records are grouped into non-redundant rs (reference SNP) records, so that each SNP is only represented once. The rs records are determined by post-submission computational analysis. Each rs record contains summary information for each SNP, including a list of ss records and summary allele frequencies. Each record also contains the sequence surrounding the polymorphism, the experimental conditions required to repeat the genotyping and descriptions of the populations containing the variation. The population or individual genotypes are also included in the records, so that the sequence information can be used for further research applications. The rs accession numbers serve as an anchor to link the SNP data to other NCBI resources, literature

databases and other external databases, which add further detail to the SNP records. dbSNP can be searched using a variety of accession number, including ss, rs and GenBank accession numbers. dbSNP can also be searched using a variety of search terms. A single query can be used to search dbSNP or a batch search can be performed. The results can be viewed in a variety of forms and downloaded directly from a web-link or via FTP facilities (Sherry et al., 1999, Smigielski et al., 2000).

There are also numerous databases for sequences and genetic variants with specific functional importance. OMIM is one of the most established databases for genes of functional and clinical importance (<http://www.ncbi.nlm.nih.gov/omim>). OMIM includes the descriptions of genes with known function. OMIM also provides information on single gene disorders, as well as select complex diseases, with the molecular basis underlying them (Hamosh et al., 2005, Amberger et al., 2011).

OMIM was first published in print in 1966 as the Mendelian Inheritance in Man (MIM), and has been available online as OMIM since 1987 (Hamosh et al., 2005). Editorial decisions and curation of OMIM is performed at the John Hopkins University School of Medicine. The distribution and software development of OMIM is provided by NCBI. OMIM provides full-text overviews of genes, their variants and associated phenotypes (Amberger et al., 2011). Although the primary focus of OMIM is on single gene Mendelian disorders, OMIM also includes information on complex disease, when the variation in a single gene significantly contributes to the phenotype. Copy number variation disorders have also, recently been included in OMIM (Amberger et al., 2011).

Mutations that contribute to phenotypic variation are cataloged in the “allelic variants” section of the gene entry (Amberger et al., 2011). Only the mutations that meet certain criteria are included in OMIM, based on:

- Which was the first mutation to be discovered
- Whether the mutation causes a distinct phenotype

- If the mutation is unusual for the specific phenotype
- Whether an unusual pathogenetic mechanism is involved
- If the mutation has a high population frequency
- If there is a distinctive inheritance
- If the mutation is historically important ([Amberger et al., 2011](#))

Each entry in OMIM is assigned a unique six digit accession number (MIM number) ([Hamosh et al., 2005](#)). The first digit in the MIM accession number indicates whether the inheritance is autosomal, X-linked, Y-linked or mitochondrial. Entries in OMIM are categorized based on whether the entry contains gene, phenotype or gene and phenotype data. OMIM can be searched using a variety of search terms, including MIM accession numbers, gene names or disorders. OMIM also contains links to other resources for each entry, including clinical synopsis, GenBank and PubMed references ([Hamosh et al., 2005](#)).

The GWAS catalog is one of the most comprehensive databases for SNPs found to be associated with traits or disease, using GWAS (<http://www.genome.gov/gwastudies/>). GWAS have helped identify thousands of SNPs that are significantly associated with traits or disease and have been published for a number of complex diseases ([Elbers et al., 2009](#), [Hindorff et al., 2009](#)). The disease associated SNPs can be mapped back to linkage disequilibrium blocks, containing several genes, thus, the disease associated SNPs can provide information about other variants in close-proximity. In this way GWAS have aided the discovery of susceptibility genes for various complex diseases ([Elbers et al., 2009](#)). The sample sizes and the number of SNPs assayed in GWAS have increased immensely in the last few years ([Hindorff et al., 2009](#)). By cataloging GWAS and identifying the key characteristics of the disease associations, the large number of GWAS conducted allow SNPs to be examined for their potential role in complex disease ([Hindorff et al., 2009](#)).

The GWAS catalog was developed by the NHGRI. The GWAS catalog is a manually curated database of published GWAS ([Hindorff et al., 2009](#)). The database contains SNPs that are

significantly associated with a trait or disease, which have been extracted from published GWAS (Hindorff et al., 2009). The GWAS catalog only includes GWAS where at least 100,000 SNPs have been assayed at the initial stage (<http://www.genome.gov/gwastudies/>). Studies focusing only on candidate genes are not included.

For a SNP to be included in the GWAS catalog the SNP has to be associated with a trait or disease with a p-value less than 1.0×10^{-5} . Only one SNP is cataloged for each gene or region of high linkage-disequilibrium, unless there is strong evidence for independent association. A maximum of 50 SNPs associated with a trait or disease are extracted from each published GWAS. The SNPs with lower p-values, or those reported as novel are given first preference. The published GWAS are identified from various different sources, including PubMed searches and news and media compilations. The gene regions related to the SNPs are determined with the use of the University of California Santa Cruz genome browser. The gene names and risk alleles are extracted from the GWAS publications. The GWAS catalog contains all of the SNPs associated with a trait from published GWAS that meet the setout criteria; however, the number and quality of the SNP associations given in the publications are dependent on the authors and journals (<http://www.genome.gov/gwastudies/>).

1.2.2. Viewing data from biological databases – genome browsers

Genome browsers are created to view the data within biological databases (Karolchik et al., 2007). Genome browsers bring together data from various resources and enable genomic sequences, coordinates and annotations to be viewed, often via a graphical user interface (GUI). In this way, genome browsers may be used to explore the genome for regions of interest. There are many different genome browsers available for the exploration of genomes from a variety of different species (Karolchik et al., 2007).

Ensembl is one example of a public access genome browser for vertebrates and other eukaryotic species (<http://www.ensembl.org/index.html>). Ensembl was created before the

initial human genome sequence was complete (Hubbard et al., 2002). Now with a complete human genome sequence available, Ensembl provides access and visualization of the entire human genome. Ensembl also contains genome assemblies for a variety of other species. The current September 2011 version of Ensembl contains genome assemblies for 57 vertebrate species (<http://www.ensembl.org/info/about/species.html>). Although the primary focus of Ensembl is on vertebrate species, Ensembl also provides links to other genome sequences for a number of invertebrate, plant, fungi and bacteria species. Ensembl provides links to other resources, which provide further detail to the gene annotations (Hubbard et al., 2002).

NCBI also hosts an assembly of the human genome (<http://www.ncbi.nlm.nih.gov>). The genome assembly can be BLAST searched and the results viewed in the Map Viewer. NCBI also provides links to various resources, including RefSeq, OMIM and other external databases through LocusLink (Pruitt et al., 2000).

Another example of a frequently used genome browser is the genome browser hosted by the University of California Santa Cruz (UCSC) (<http://genome.ucsc.edu/cgi-bin/hgGateway>). The UCSC genome browser is based on a MySQL database (Kent et al., 2002, Karolchik et al., 2007). The database contains sequences and annotations for a number of genome assemblies for a variety of species. The assemblies are retrieved from NCBI and stored within positional and non-positional tables. This database contains a vast amount of aligned annotations, including SNP, mRNA and EST alignments. These sequences are extracted from GenBank and aligned to the reference genome assembly, using BLAT (BLAST-like alignment tool). The sequences and annotations can be viewed as tracks, in a text or graphical interface, at various levels of detail. The tables within the database can be queried based on various search terms. Entries within the tracks contain links to other web pages and resources, which provide more detail to the annotations. The query results may be downloaded through html (hypertext mark-up language) links or through FTP facilities (Kent et al., 2002, Karolchik et al., 2007).

There are numerous genome browsers available. Each of the genome browsers provides access to vast amounts of genetic data. The genome browsers will often also provide links to other external resources. With the number of genome browsers available, it is often difficult to decide which one to use and often the choice of which genome browser to use will depend on personal preference or on which one of the browsers the user is most familiar with.

1.2.3. Retrieval and analysis of biological data

With the vast amount of biological data available, it is possible to perform large-scale genomic studies (Goecks et al., 2010). The access and integration of the data is however often a difficult task that requires programming skills and a thorough understanding of the tools required. In addition, many published computational studies are not easily reproducible, as only the person or group who performed the analysis understands the intent and process followed. To overcome these problems of accessibility and reproducibility, researchers from UCSC created Galaxy (Goecks et al., 2010).

Galaxy is a freely available web-based workbench for the computational analysis of genomic data (<http://main.g2.bx.psu.edu/>). Data can be uploaded from a local computer or imported from a variety of different sources, such as dbSNP and RefSeq (Goecks et al., 2010). The imported datasets can be manipulated and analyzed directly from within the Galaxy framework. Galaxy provides access to a variety of analysis tools. The analysis tools can be linked together to form an analysis pipeline, eliminating the requirement for an in-depth understanding of programming or the tools required. For each step in the analysis pipeline, Galaxy records information on the dataset, tools and parameters used. Users may also add additional annotations, explaining their intent and process followed. This information, together with the datasets are stored within Galaxy histories. The Galaxy histories can be accessed at any time and from any computer with internet access. The analysis pipelines may also be converted into workflows. A workflow allows the same pipeline to be used, with

the same tools and the same chosen parameters on a different dataset. The Galaxy histories and workflows can be shared with other Galaxy users, either with all Galaxy users or with only chosen Galaxy users. The sharing of histories and workflows enables the reproducibility of experiments. Galaxy may also be downloaded and altered according to the user's requirements. Galaxy is a flexible workbench, which provides accessibility and reproducibility of genomic data analysis (Goecks et al., 2010).

1.2.4. Computational prediction of biological function

The vast amount of publically available data has dramatically improved knowledge of the fundamental concepts of biology, including genetics. Not only has the integration of biological data enabled the identification of functionally important regions within the human genome, but also the development of algorithms and tools that can predict the potential functional impact of specific sequences. These computational methods have been used in a variety of applications, from the simple prediction of the amino acids encoded by codons, to the predictions of complex interactions within the cells. Algorithms and computational tools have been developed for the prediction of mRNA secondary structures and alternative splicing events.

Computational prediction of mRNA secondary structure and stability

The predictions of RNA secondary structures have come a long way since 1971, when Tinoco *et al.* first studied small RNA molecules to estimate the energies associated with secondary structures (Tinoco et al., 1971). The stabilizing effects of base-paired regions and the destabilizing effects of unpaired regions were taken into account, to determine which structures in larger molecules were more stable (Tinoco et al., 1971, Tinoco and Bustamante, 1999). There are now a number of tools available, based on various algorithms, to computationally predict RNA secondary structures.

RNA secondary structures can be computationally predicted, either through sequence covariant methods or through energy minimization methods ([Zuker and Sankoff, 1984](#), [Mount, 2004](#)). The sequence covariant methods are based on the alignment of a number of RNA sequences. The alignment of the RNA sequences is used to identify interacting base-pairs, based on sequence conservation. Sequence covariant methods are computationally simplistic, while allowing for the identification of secondary and tertiary mRNA structures however, these methods require a large amount of sequence data ([Mount, 2004](#)).

Energy minimization methods aim to identify all possible double-stranded regions within an mRNA sequence, to identify the most thermodynamically favourable RNA secondary structure ([Zuker and Sankoff, 1984](#)). Energy minimization methods are highly reliable, as they are based on experimental evidence and energy calculations. Unlike sequence covariant methods, energy minimization methods cannot predict RNA tertiary structures. Energy minimization methods do not require the large amount of sequence data that are required by sequence covariant methods. Energy minimization methods are based on the identification of all possible double-stranded regions within an RNA sequence, thus as the sequence length increases, so do the number of potential RNA secondary structures that can form. A very long RNA sequence will dramatically increase the requirement for computational power, which may reduce the accuracy of the mRNA secondary structure predictions ([Nussinov and Jacobson, 1980](#), [Zuker and Sankoff, 1984](#), [Mount, 2004](#)).

Energy minimization methods start by identifying all self-complementary regions within the RNA sequence and the minimum free-energy of the complementary regions ([Zuker and Sankoff, 1984](#)). To identify bases that can potentially base-pair, matrices are used to compare each base in the RNA sequence to every other base in the sequence. Once the possible double-stranded regions are identified, the nearest neighbor rule is used to calculate the energy of each of the predicted structures. Base-paired regions contribute a negative stabilizing energy and unpaired regions contribute a positive destabilizing energy. The structure with the lowest free-energy is the structure with the highest proportion of base-

pairs and the most thermodynamically stable structure ([Nussinov and Jacobson, 1980](#), [Zuker and Sankoff, 1984](#), [Mount, 2004](#)).

One of the greatest contributions to the computational predictions of RNA secondary structures came from Zuker and Steigler. In 1981, Zuker and Steigler developed a dynamic programming algorithm that used energy rules to predict the energetically most stable structure ([Zuker and Stiegler, 1981](#)). In this dynamic programming algorithm, the optimal secondary structure is found by first determining the optimal structure of each part of the RNA sequence ([Zuker and Sankoff, 1984](#)). The dynamic programming algorithm uses several types of scoring matrices to evaluate all the possible secondary structures from the RNA sequence. The thermodynamic free-energy of each of the predicted structures is calculated. Once the thermodynamically most stable structure is identified, the data from the matrices are used to predict the minimum free-energy structure. An undesirable consequence of this algorithm is that, as the number of variables increase, so do the number of recursive functions, which requires a great deal of computational resources ([Tsang and Wiese, 2007](#)). Zuker and Steigler developed their algorithm into the FOLD tool. Biochemical and structural information was included into FOLD to aid the identification of base-pairs that are more likely to form, and to enforce topological restraints. FOLD, like previous methods of RNA secondary structure prediction, took on the assumption that the most likely RNA secondary structure is the structure with the lowest free-energy, thus FOLD only predicted one possible mRNA secondary structure for each mRNA sequence ([Zuker and Stiegler, 1981](#), [Zuker and Sankoff, 1984](#), [Jaeger et al., 1989](#)).

One of the biggest problems with minimum free-energy methods is that they only predict one possible mRNA secondary structure ([Zuker and Stiegler, 1981](#), [Zuker and Sankoff, 1984](#), [Jaeger et al., 1989](#)). In reality, RNA can fold into various different secondary structures. The true structure is not always the structure with the lowest free-energy, but rather within a five to ten percent energy range from the minimum free-energy structure ([Tinoco and](#)

Bustamante, 1999, Mohsen et al., 2009). It is generally more accurate to predict all possible structures within a defined range from the predicted minimum free-energy structure.

mFold was developed by Zuker as an improvement on the structures predicted by dynamic programming algorithms (<http://mfold.rna.albany.edu/?q=mfold/download-mfold>). The mFold tool predicts the possible suboptimal RNA secondary structures, within a user-defined percentage of the minimum free-energy structure (Zuker, 1989, Zuker et al., 1991). In-order to determine which of the base-pairs are within the suboptimal structures, which are within the defined range from the minimum free-energy structure, mFold employs different matrices to compare all of the complementary bases within the structure. An energy dot plot is created to depict the positions of the alternative base-pairs in the most stable and suboptimal structures (Zuker, 1989). Predictions made by mFold are generally within good agreement with experimental results. mFold has subsequently become one of the most frequently used tools for the prediction of mRNA secondary structures (Mount, 2004). mFold is demanding on computational resources and is thus, generally used to predict the secondary structures of RNA molecules less than a thousand bases in length. In addition, mFold does not calculate all possible suboptimal structures. For instance, mFold does not produce alternative structures when a base that contributes to a specific base-pair may form a base-pair with another base elsewhere in the sequence. Moreover, if two predicted suboptimal structures are connected by a region of unpaired bases, each structure will be compared individually. This does not take into account the interactions between the structures or the unpaired regions. These limitations reduce the number of suboptimal structures predicted by mFold (Zuker, 1989, Zuker, 2003, Mount, 2004).

A modified trace-back procedure was therefore developed and implemented in the Vienna RNA package (<http://www.tbi.univie.ac.at/RNA/>). The modified trace-back procedure compares the structures that are globally suboptimal for both paired and unpaired regions (Wuchty et al., 1999). This method calculates a large number of suboptimal structures. The Vienna RNA package uses the Boltzmann-partition function to determine the most likely RNA

secondary structure and alternative structures. Due to the large number of structures produced by this method of RNA secondary structure prediction it is extremely computationally taxing ([Wuchty et al., 1999](#)).

Another approach for the prediction of RNA secondary structures is to use a meta-heuristic algorithm ([Mohsen et al., 2009](#)). HSRNAFold is one such tool, which is based on a harmony search meta-heuristic algorithm. HSRNAFold first determines all of the possible base-pairs and then the possible stem structures that can form. The subset of structures that represents the most likely structure with the lowest free-energy is then identified. Once the first structure is predicted, each stem is checked for conflicts with the previous stem. If there are no conflicts, the structure is retained. A new structure is then created, based on the previous structures and other parameters. The minimum free-energy of the new structure is calculated. If the structure is better than the worst structure already in the set, the new structure is included in the set and the worst structure is discarded. The algorithm iterates over the solution vector and creates a new structure for each iteration, until the structure with the lowest free-energy is determined or the maximum number of iterations is reached. HSRNAFold predicts RNA secondary structures with a higher proportion of true positive base-pairs than RNAfold from the ViennaRNA package. Both HSRNAFold and RNAfold are highly reliable for shorter sequences and become less accurate as sequence length increases. HSRNAFold requires less computational power than RNAfold ([Mohsen et al., 2009](#)). HSRNAFold is still reasonably new and not well cited, thus its true potential for predicting RNA secondary structures is still to be established.

Thermodynamic predictions of RNA secondary structures are generally reliable. Although the predictions of RNA secondary structures are always improving, there are still certain assumptions that are often made:

1. The true RNA secondary structure is the structure with the lowest free-energy or energy similar to the lowest free-energy.

2. The energy associated with a base-pair is only influenced by the previous base-pair and no other base-pair in the structure.
3. Single-stranded RNA folds back on itself to form secondary structures, in a way that produces no pseudo-knots (Tinoco and Bustamante, 1999).

These assumptions are not always true, thus any prediction should be viewed with caution. The assumptions the RNA secondary structure prediction tool is based on should always be kept in mind when viewing the predictions of RNA secondary structures (Zuker and Sankoff, 1984, Mount, 2004).

Computational prediction of alternative splicing events

It was long assumed that the proteome diversity of an organism is determined by its number of genes. Bioinformatics has played a critical role in showing that it is not only the number of genes, but also the way in which transcripts are rearranged during alternative splicing that contribute to proteome diversity (Lee and Wang, 2005). Much focus has been placed on the study and computational prediction of alternative splicing.

The alignment of expressed sequence tags (ESTs) has been one of the pioneering computational methods in the identification of alternative splicing events. Although ESTs have shed much light on the process of alternative splicing, the process of EST alignment is fraught with experimental biases and artifacts, which need to be considered during analysis (Modrek and Lee, 2002). Microarray technologies, such as exon arrays have helped further identify alternative splicing events. Microarrays do not have the same limitations of EST studies, such as the biases for the 3' end of genes and the inability to detect splicing in regions of low EST coverage (Johnson et al., 2003). Consequently, microarrays have helped uncover alternative splicing events not identified by EST methods. In addition, the availability of a large number of genomic sequences and a more complete version of the human genome sequence has aided the discovery of alternative splicing events, through comparative genomics, where no EST data is available (Kan et al., 2004, Sorek et al., 2004).

The combination of EST, microarray, genomic and cross-species data has not only confirmed the abundance of alternative splicing events, but also that alternative splicing is often regulated in a tissue-specific manner, further increasing proteome diversity (Kampa et al., 2004).

There are numerous tools available for the computational prediction of alternative splicing events from DNA sequences. The Human Splicing Finder (HSF) tool is one of the most comprehensive tools for the prediction of alternative splicing events (<http://www.umd.be/HSF/>). HSF is a freely available web-service. HSF combines experimentally and computationally derived data, to predict the signal strength of splice-sites, branch-point sites and regulatory motifs. The predictions of these splicing regulatory regions are based on various different algorithms. HSF also contains Ensembl exons and introns, as well as SNP data from dbSNP. Ensembl can thus, be directly queried to identify SNPs within human genes. This enables the comparison of various mutations within the splicing regulatory regions (Desmet et al., 2009).

In-order to predict the signal strength of potential splice-sites, HSF assigns a given weighted score from a position weight matrix (PWM) to each nucleotide in the queried sequence. Each weighted score is summed and only n -mer sequences, 9-mer for the 5' splice-sites and 23-mer for the 3' splice-sites, with a summed score greater or equal to the given threshold are considered potential splice-sites. HSF also contains a maximum-entropy algorithm to analyze whole sequences, where the users may define their own threshold. To identify splicing regulatory regions, matrices for SR and hnRNP proteins, as well as various other splicing motifs have been included in HSF. Although many of the *trans*-acting proteins involved in splicing have been cloned and their *cis*-acting motifs identified, a lot of work still needs to be done before alternative splicing can be fully understood (Desmet et al., 2009). HSF is not designed to handle large-scale studies where many sequences are compared. HSF is still based on version 54 (May 2009) of Ensembl. Users have to be aware that some of the

genomic coordinates may have changed, if using a later version of Ensembl, which may influence the predictions made by HSF.

The maximum-entropy distribution (MED) is one of the best algorithms to model short sequences and is thus ideally suited for the prediction of splice-site sequence motifs. MED takes nearest neighboring contributions into account, has little bias and a very low false positive rate (Yeo and Burge, 2004). Yeo and Burge have used the MED model to develop MaxEntScan (http://genes.mit.edu/burgelab/maxent/Xmaxentscan_scoreseq.html).

MaxEntScan is a freely available web-tool for the prediction of splice-sites (Yeo and Burge, 2004). MaxEntScan is based on the maximum-entropy model, which generalizes previous probabilistic models, such as the weight matrix model and the inhomogeneous markov model. The user may chose to perform the predictions based on a variety of models. The models used to predict potential splice-sites could be run either directly from the web-based GUI or through Perl scripts that can be downloaded from the website. To determine the signal strength of potential 5' splice-sites, MaxEntScan requires a 9-mer sequence, with three bases into the upstream exon and six bases into the downstream intron [-3 to +6]. To determine the signal strength of potential 3' splice-sites, MaxEntScan requires a 23-mer sequence, with 20 bases into the upstream intron and three bases into the downstream exon [-20 to +3]. MaxEntScan scores the input sequences using the models chosen and returns the log-likelihood score for the sequence. A higher log-likelihood score represents a sequence that is more likely to be a true splice-site. MaxEntScan does not provide a given threshold that will determine whether a sequence motif is a true splice-site or not. In-order to determine a threshold, additional analysis of expression data may be required.

MaxEntScan does not include the prediction of the *cis*-acting regulatory motifs, which are essential to understanding alternative splicing. As MaxEntScan is based on an algorithm that includes global as well as local sequence information, it outperforms other algorithms used to predict the signal strength of potential splice-sites. MaxEntScan is able to predict the log-likelihood scores of thousands of sequences in a short period and is ideally suited for large-scale studies (Yeo and Burge, 2004).

There is a difference between being able to identify alternative splicing events and understanding how alternative splicing is regulated, and how it affects gene function. The identification of alternative splicing regulatory mechanisms is a growing field, in which bioinformatics is playing a critical role (Lee and Wang, 2005). Using experimental and computational studies of hexameric sequences enriched within regions of weak splice-sites, in combination with SNP data and comparative genomics, numerous regulatory motifs associated with different alternative splicing events have been identified. Much focus has been placed on ESEs (Fairbrother et al., 2002, Fairbrother et al., 2004, Sakai and Maruyama, 2004).

ESEs not only play an important role in enhancing the use of splice-sites, but also help identify true exons from pseudoexons (Wang and Burge, 2008). Based on the assumption that ESEs are highly conserved and enriched within true-exons, a large number of ESE consensus sequences have been identified. The identified ESE sequence motifs have been collected into databases and tools developed that can identify ESEs, based on sequence similarity (Fairbrother et al., 2002, Zhang and Chasin, 2004). ESEfinder is one such tool.

ESEfinder is a freely available tool for the identification of putative ESEs from RNA sequences (<http://rulai.cshl.edu/cgi-bin/tools/ESE3/ese finder.cgi?process=home>). ESEfinder uses PWM corresponding to the different SR proteins to score the query sequences (Cartegni et al., 2003). High scoring sequence motifs are defined as sequences with a score greater than the chosen threshold value. A sequence with a score greater than the threshold value does not necessarily mean that the sequence motif is a functional ESE. The matrices used in ESEfinder were determined based on human SR proteins, thus ESEfinder may not accurately identify putative ESEs in other species (Cartegni et al., 2003).

As there is a large range of ESEs that are recognized by splicing factors, we still cannot computationally identify all ESEs (Cartegni et al., 2002). A number of naturally occurring ESEs have been identified that do not conform to the computationally predicted motifs. This may

be because experimentally identified ESEs are generally identified in highly-purified *in-vitro* systems that do not represent the true splicing environment ([Cartegni et al., 2002](#)).

The developments of tools that enable the identification of the potential functional impact of mutations have improved the ability to identify disease-causing variants. It should however, be remembered that computational predictions are only as good as the models they are based on. Computational predictions do not necessarily indicate what is truly occurring in the cell. Experimental validation is still the ideal, but is not always feasible or possible. Computational predictions can provide insight where experimental studies are not possible.

1.3. Study rationale

To date little research has been conducted on the potential functional impact of sSNPs, in comparison to the wealth of information available on the potential functional impact of nsSNPs. There is a growing list of sSNPs that have been demonstrated to be associated with disease. It is plausible that sSNPs have a greater effect on protein structure and function than originally proposed. As highlighted in the literature review, sSNPs may have a potential functional impact through at least three molecular mechanisms:

1. A change in codon-usage frequency may alter translational and protein folding kinetics, potentially resulting in a protein of altered structure and function.
2. A change in mRNA secondary structure and stability may either affect the amount of protein product produced or alter translational and protein folding kinetics, which may alter the structure and function of the protein.
3. A change in splice-site consensus sequences may result in aberrant splicing, which may affect the amount of a specific protein isoform or alter the protein structure and function.

Each of these three mechanisms can be assessed computationally. Although several sSNPs are no longer thought to be silent, no one has identified which sSNPs are more likely to contribute to phenotypic variation on a genome-wide scale.

The objective of this study was to computationally determine the proportion of sSNPs within the human genome that may have a potential functional impact, either through a change in translational and protein folding kinetics or aberrant splicing. As a further step, it was assessed whether any of the sSNPs predicted to potentially have a functional impact had previously been associated with a trait or disease in GWAS and/or occurred within genes known to be associated with diseases in OMIM.

STUDY AIMS:

1. Identify all sSNPs published within dbSNP
2. Assess whether any of the identified sSNPs may have a potential functional impact through:
 - i. an extreme change in codon-usage frequency
 - ii. an extreme change in the minimum free-energy of local mRNA secondary structure
 - iii. an extreme change in the maximum-entropy score of splice-sites
3. Determine whether any of the sSNPs predicted to have a potential functional impact have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM.

CHAPTER 2: METHODS

2.1. SynSNP

To computationally identify sSNPs with a potential functional impact, I created SynSNP. SynSNP is a text-based tool, written in Python and executed in command line. The SynSNP pipe-line identifies sSNPs with a potential functional impact either through a change in translation and protein folding kinetics or aberrant splicing. SynSNP then assesses whether the potentially functional sSNPs have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM (**see Appendix A, for a systematic manual on using SynSNP**).

All of the data files required to run SynSNP can be retrieved using the work-flows I have created in Galaxy. SynSNP has a menu where the user can choose to execute specific sections, for instance only identify sSNPs, or run the entire analysis pipeline. The first step when using SynSNP is to download all cSNPs from dbSNP, all SNPs known to be associated with a trait or disease in GWAS, from the GWAS catalogue and all genes known to be associated with a disease in OMIM. SynSNP uses the cSNP file to identify all the sSNPs (**Fig.2.1.a**). The GWAS and OMIM files are then used to determine whether any of the sSNPs have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM.

Once the sSNPs have been identified, the second option in SynSNP may be selected. The second option determines whether any of the sSNPs may potentially cause an extreme change in codon-usage frequency (**Fig.2.b**). The third option of SynSNP determines whether any of the sSNPs may contribute to a change in local mRNA secondary structures. To achieve this, all coding sequences need to be retrieved from RefSeq. SynSNP uses the coding sequence to create the 100 base mRNA sequence surrounding each sSNP. RNAfold is used to

calculate the minimum free-energy of the 100 base local mRNA secondary structures. SynSNP then determines whether any of the sSNPs may potentially result in an extreme change in the minimum free-energy of the local mRNA secondary structures (**Fig.2.1.c**).

The fourth option of SynSNP determines whether any of the sSNPs may contribute to abnormal splicing. To identify sSNPs that may result in aberrant splicing, all exons need to be retrieved from RefSeq. SynSNP uses the exons to determine which of the sSNPs are in the exonic part of the 3' and 5' splice-sites. MaxEntScan is used to calculate the maximum-entropy score of each splice-site that contains a sSNP. SynSNP then determines whether any of the sSNPs may potentially result in an extreme change in the maximum-entropy score of splice-sites (**Fig. 2.1.d**).

Once the sSNPs with a potential functional impact have been identified, option five may be selected. Option five of SynSNP creates a summary report of all the sSNPs predicted to have a potential functional impact. When option six is selected, SynSNP uses the files created in the first step of SynSNP to determine which of the sSNPs predicted to have a potential functional impact have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM (**Fig.2.1.e**).

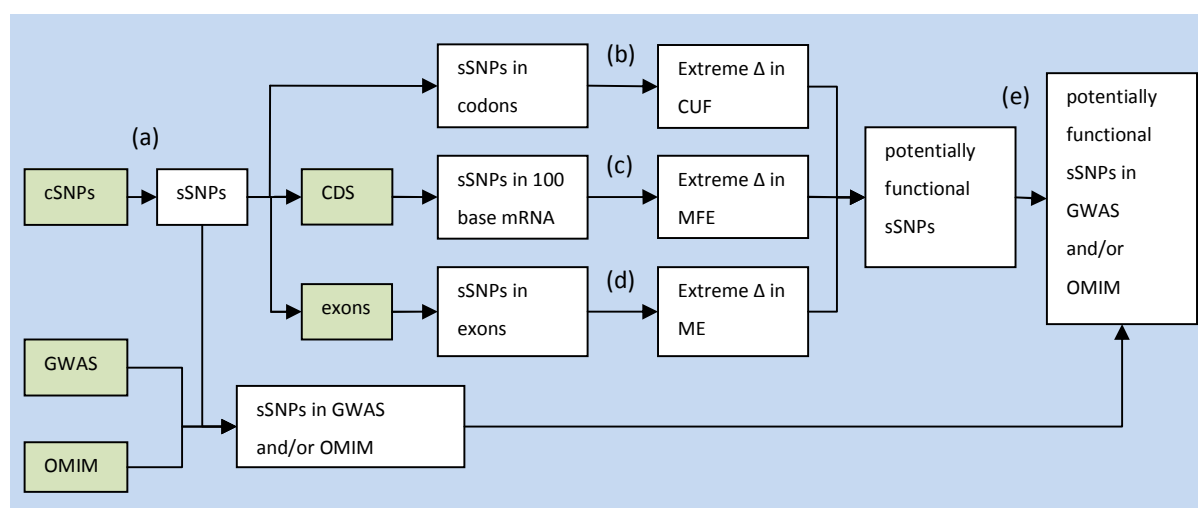


Figure 2.1: The pipeline used in SynSNP, to identify sSNPs with a potential functional impact, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM.

 = files retrieved, using Galaxy

 = files created by SynSNP

- (a) SynSNP identifies sSNPs from the cSNPs and determines which of the sSNPs may have a potential functional impact, either through:
 - (b) an extreme change in codon-usage frequency (CUF)
 - (c) an extreme change in the minimum free-energy of local mRNA secondary structures (MFE)
 - (d) an extreme change in the maximum-entropy score of splice-sites (ME)
- (e) SynSNP then determines whether any of the sSNPs predicted to have a potential functional impact have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM

2.2. Identification of sSNPs

sSNPs are defined as SNPs that do not change the amino acid encoded. For this reason, sSNPs have to be within the coding regions of genes that encode a polypeptide. Only sSNPs within translated coding regions of the human genome were assessed for potential functional impact. The reference SNP alleles were chosen as the alleles given by NCBI's dbSNP131, which are the nucleotides as annotated in the reference genome assembly.

The data files required to identify sSNPs with a potential functional impact were retrieved using the UCSC main table browser in Galaxy. The data files were retrieved for the February 2009 (hg19/GRCh37) version of the human genome assembly on the 30th of May 2011. The first step to identify sSNPs with a potential functional impact was to retrieve all known coding SNPs (cSNPs) and determine which of the cSNPs were synonymous. The methods used to identify the sSNPs are depicted in Figure 2.2.

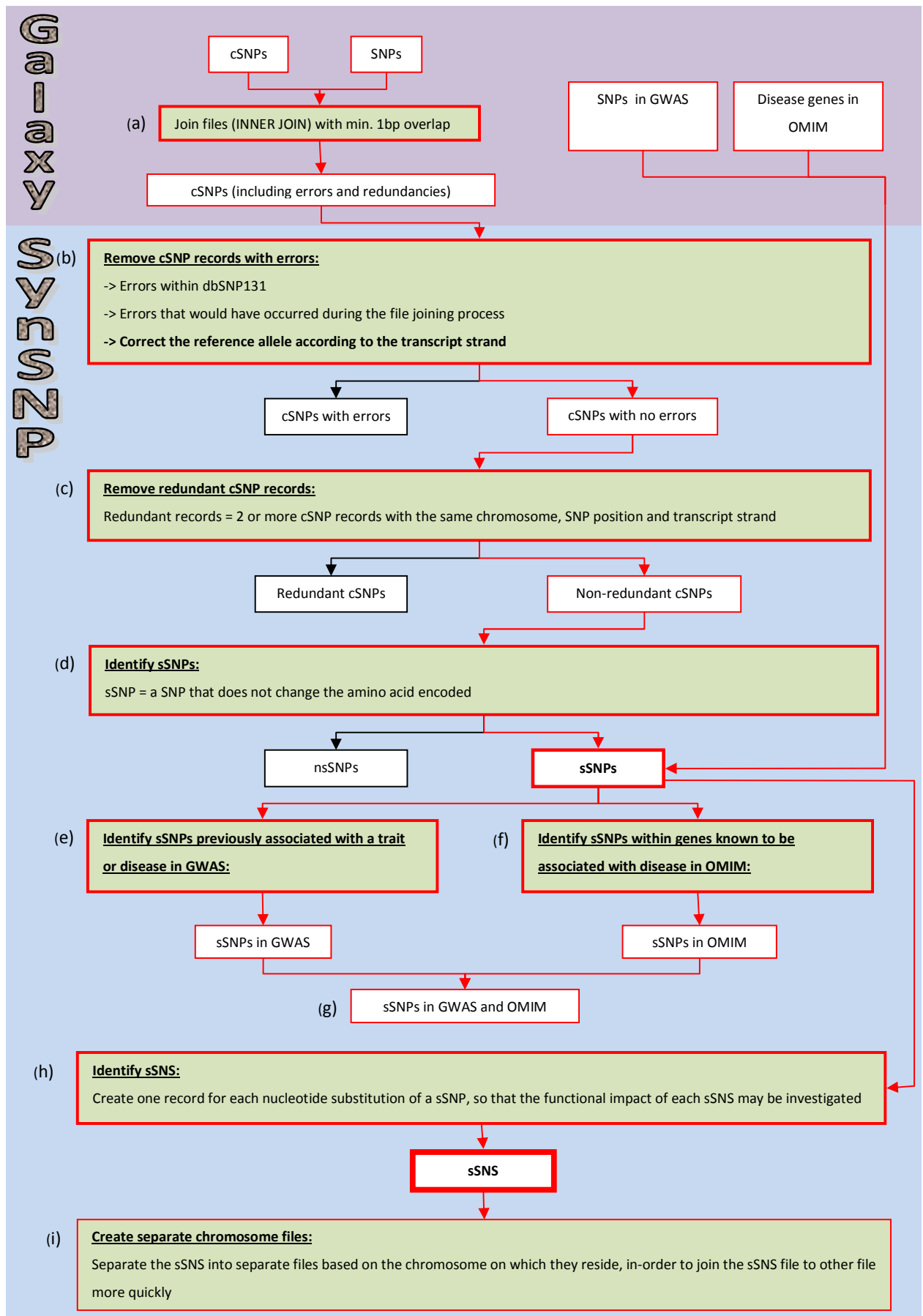
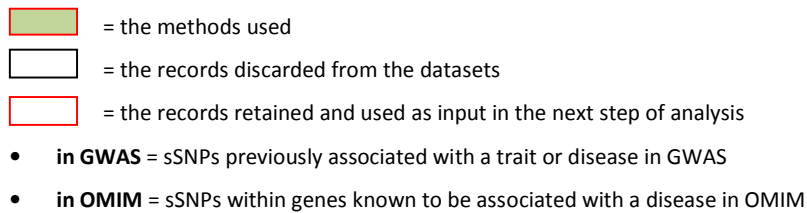


Figure 2.2: The methods used to identify sSNPs, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM.

- 
-  = the methods used
 = the records discarded from the datasets
 = the records retained and used as input in the next step of analysis
- **in GWAS** = sSNPs previously associated with a trait or disease in GWAS
 - **in OMIM** = sSNPs within genes known to be associated with a disease in OMIM

The UCSC main table browser in Galaxy was used to retrieve cSNPs published within dbSNP131 (**Fig.2.3**). dbSNP131 contained SNPs annotated on genomic DNA and coding DNA. To ensure consistency and prevent redundancy, only the cSNPs annotated on genomic DNA were retrieved. The cSNP file included the codons in which the cSNPs resided and the amino acids encoded by the codons. The cSNP file did not include the reference SNP alleles, the observed alleles or the DNA strands on which the SNPs were located. In-order to retrieve the additionally required data it was necessary to retrieve all SNPs published within dbSNP131. The UCSC main table browser in Galaxy was used to retrieve all the SNPs annotated on genomic DNA from dbSNP131 (**Fig.2.4**). The SNP file contained data related to each SNP, including the reference SNP alleles, the observed alleles and the DNA strands on which the SNPs were located. By combining the records in the cSNP file and the SNP file, one file could be created that contained all the data required. Galaxy was used to join the cSNP and SNP files (**Fig.2.5**). The records in the two files were joined based on the SNPs' positions in the chromosomes, thus only SNPs within both the cSNP and SNP files were joined. The resulting file contained the cSNPs, with all the data required to identify sSNPs (**Fig.2.2.a and Table.2.1**).

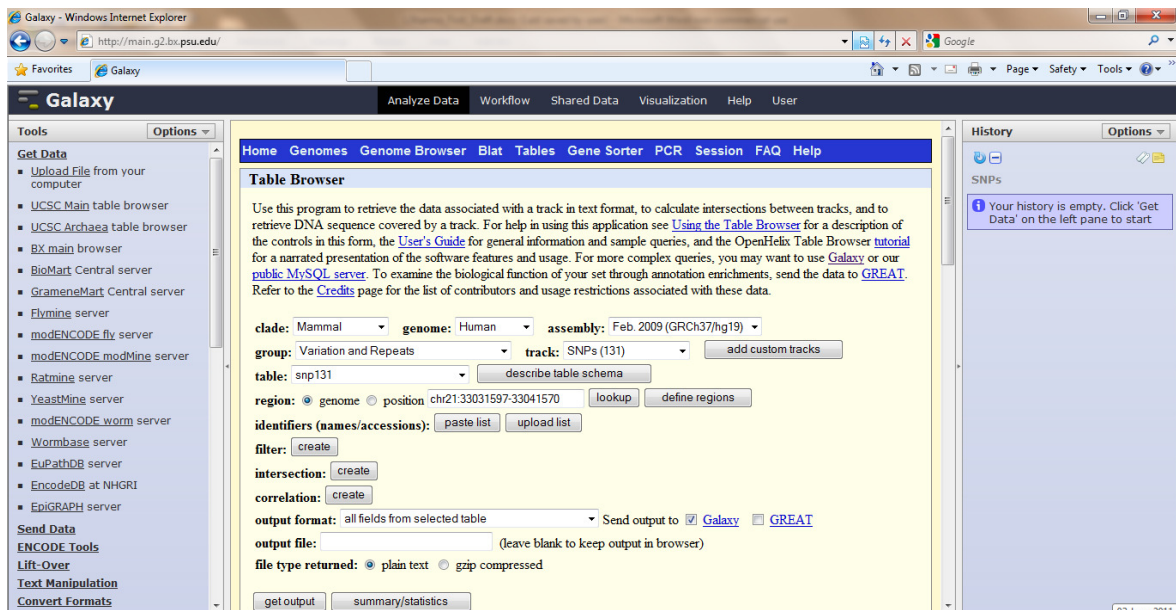


Figure 2.3: The screen-shot of the UCSC main table browser in Galaxy, used to retrieve cSNPs from dbSNP131.

The cSNPs were retrieved from dbSNP131, by selecting the “variation and repeats group”, the “SNPs(131) track”, and the “snp131 table” from the UCSC main table browser in Galaxy. Under the filter settings, the class was chosen to match all variant of type “single” and the molecular type was selected to only retrieve SNPs annotated on the genomic DNA. The function setting was selected to retrieve SNPs within the coding regions of the human genome.

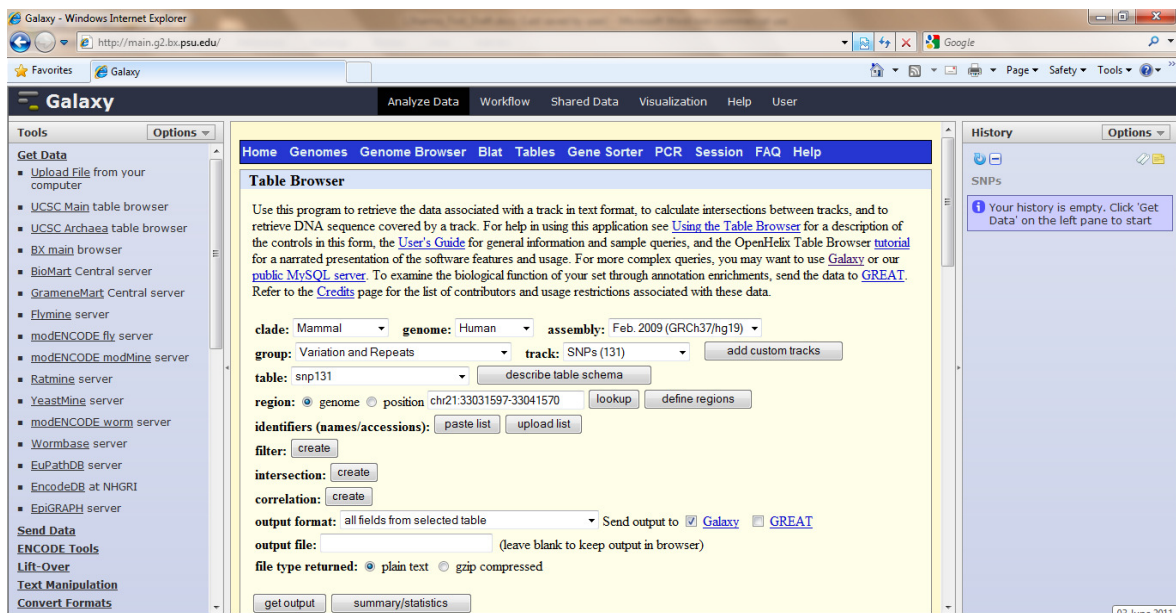


Figure 2.4: The screen-shot of the UCSC main table browser in Galaxy, used to retrieve SNPs from dbSNP131.

The cSNPs were retrieved from dbSNP131, by selecting the “variation and repeats group”, the “SNPs(131) track”, and the “snp131 table” from the UCSC main table browser in Galaxy. Under the filter settings, the class was chosen to match all variant of type “single” and the molecular type was selected to only retrieve SNPs annotated on the genomic DNA.

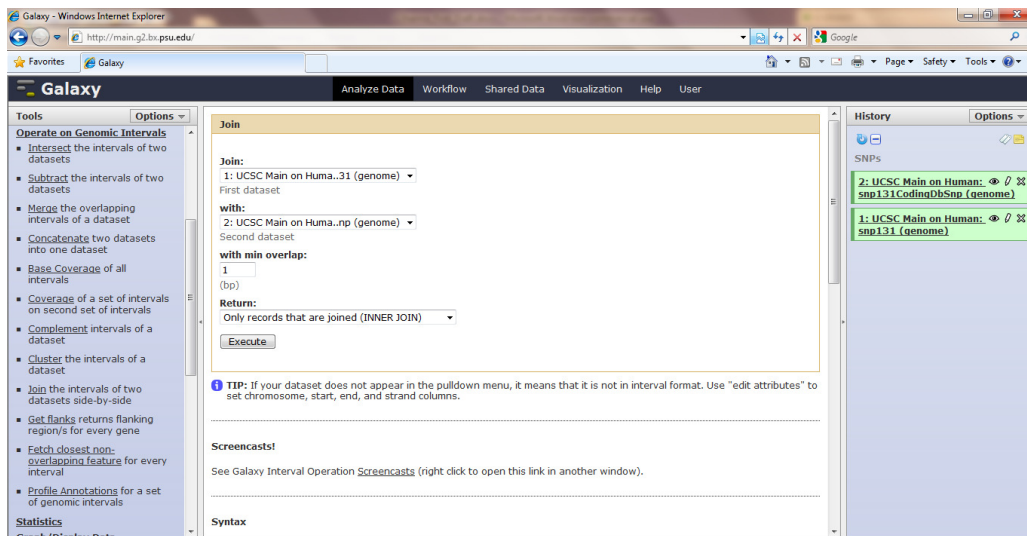


Figure 2.5: The screen-shot of the tool used in Galaxy to join the cSNP and SNP files.

The cSNP and SNP files were joined by selecting the “operate on genomics intervals” and “join the intervals of two datasets side-by-side” options, under the tools section of Galaxy. The files were joined using an inner join, so that only the records of SNPs that occurred in both the cSNP and SNP files were joined.

Table 2.1: The table schema of the cSNP data file

Column	Field	Description
0	Bin number	Indexing field used by Galaxy
1	Chromosome	Reference SNP chromosome
2	SNP start	SNP start position in chromosome
3	SNP end	SNP end position in chromosome
4	SNP name	dbSNP reference SNP identifier (rs number)
5	Score	Not used
6	SNP strand	DNA strand on which the observed alleles are found
7	RefNCBI	Reference allele given by dbSNP, as on the positive DNA strand
8	RefUCSC	Reference allele given by UCSC
9	Observed	All observed alleles, as on the SNP strand
10	Molecular type	Sample type (e.g. genomic)
11	Class	Class of the variant (e.g. single)
12	Valid	Validation status of the SNP
13	AvHet	Average heterozygosity from all observations
14	AvHetSE	Average heterozygosity standard error
15	SNP function	Functional category of the SNP (e.g. synonymous)
16	Locus type	Type of mapping inferred from the size on reference
17	Weight	Code that represents the quality of the alignment
18	Bin number	Indexing field used by Galaxy for indexing
19	cSNP chromosome	Reference coding SNP chromosome
20	cSNP start position	Coding SNP start position in chromosome
21	cSNP end position	Coding SNP end position in chromosome
22	cSNP name	dbSNP reference SNP identifier (rs number)
23	Transcript name	Accession number of the transcript modified by the coding SNP
24	cSNP frame	Frame of transcripts codon modified by the coding SNP
25	Allele count	Number of annotated coding SNP alleles
26	cSNP function code	SNP function code assigned by dbSNP (e.g. 8 = sSNP)
27	Alleles	Observed alleles on transcripts modified by the coding SNP
28	Codons	Codon sequences corresponding to the cSNP function code (5' → 3')
29	Amino acids	Amino acid sequences corresponding to the cSNP function code

The cSNP file was downloaded from Galaxy onto a local desktop computer and checked for errors. The error check removed any errors that may have occurred during the file joining process. This error check included the removal of records that had data missing. Records where the chromosome from the cSNP file was different from the chromosome from the SNP file were also removed. In addition, records where the SNP coordinates from the cSNP file differed from the SNP coordinates from the SNP file were removed. The error check also removed errors that existed in dbSNP131. This error check included the removal of INDELS that were incorrectly annotated as SNPs in dbSNP131. Records where the reference allele given by NCBI/dbSNP131 was different from the reference allele given by UCSC were also removed. In addition, records where the reference allele given was not one nucleotide in length or the codons given were not three nucleotides in length were removed (**B.sSNPs.Remove_Errors()** and **Fig.2.2.b**).

The DNA strands on which the cSNPs were originally identified were given by dbSNP131, however, the DNA strands from which the cSNPs were transcribed were not given. It was, therefore, necessary to determine the transcript strands and the reference cSNP alleles as on the transcript stands (**B.sSNPs.Write_File()**). As the reference alleles were annotated according to the reference genome assembly, the reference alleles given by dbSNP131 were always described on the positive DNA strand. The observed alleles were always described as on the transcript strand. Because the reference alleles were annotated on the positive DNA strand, if the reference allele given was in the observed alleles, it meant that the cSNP was transcribed from the positive DNA strand. The reference allele was therefore the reference allele given by dbSNP131. If the cSNP was published in dbSNP as on the positive DNA strand, the record did not require any corrections (**Fig.2.6.a**). If the cSNP was published in dbSNP as on the negative DNA strand, only the DNA strand had to be changed to the positive strand, in-order to correct the cSNP record (**Fig.2.6.b**). Because the reference alleles were annotated on the positive DNA strand, if the complement of the reference allele given was in the observed alleles, it meant that the cSNP was transcribed from the negative DNA strand. The reference allele was therefore the complement of the reference allele given by dbSNP. If the

cSNP was published in dbSNP as on the negative strand, only the reference allele needed to be complemented, in-order to correct the cSNP record (**Fig.2.6.c**). If the cSNP was published in dbSNP as on the positive strand, the reference allele needed to be complemented and the DNA strand needed to be changed to the negative strand, in-order to correct the cSNP record (**Fig.2.6.d**).

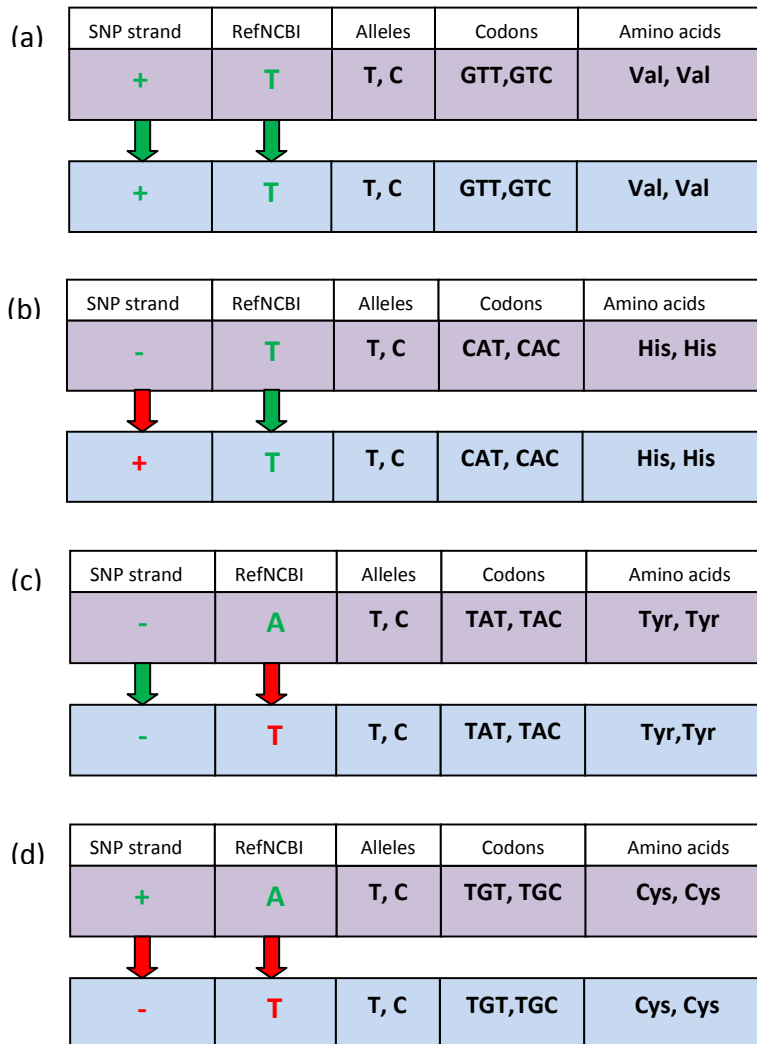


Figure 2.6: Examples of how the reference alleles given by dbSNP131 have to be modified to reflect the reference alleles on the transcript strands.

The SNP strand and RefNCBI are from the SNP file. The alleles, codons and amino acids are from the cSNP file.

- RefNCBI = the reference allele as given by dbSNP131
- Alleles = the observed alleles, which are all of the alleles of a SNP

■ = the records retrieved from dbSNP131

■ = the records after the reference alleles have been corrected according to the transcript strands

- (a) SNP published in dbSNP as on the positive DNA strand and transcribed from the positive DNA strand
- (b) SNP published in dbSNP as on the negative DNA strand, but transcribed from the positive DNA strand
- (c) SNP published in dbSNP as on the negative DNA strand and transcribed from the negative DNA strand
- (d) SNP published in dbSNP as on the positive DNA strand, but transcribed from the negative DNA strand

SNPs published within dbSNP are assigned rs numbers. Although the SNPs that have been assigned rs numbers are meant to represent a non-redundant set of SNPs, there is some redundancy in dbSNP. In some cases, a single SNP may be assigned more than one rs number. In other cases, two different SNPs may be assigned the same rs number (**Table.2.2**). Instead of using the rs numbers, it is better to describe SNPs according to their genomic positions, being mindful of the genome version.

Table 2.2: Examples of redundant SNPs within dbSNP131, either one SNP assigned more than one rs number or two different SNPs assigned the same rs number

Chromosome	Position	rs number	Function	Strand	Reference allele	Alleles	Codons	Amino acids
chr1	12606	rs2974781	coding-synon	+	C	C,T	CCC,CCT	P,P
chr1	12606	rs71277363	coding-synon	+	C	C,T	CCC,CCT	P,P
chr5	69212070	rs10081586	missense	+	A	A,G	AAG,AGG	K,R
chr5	69497705	rs10081586	missense	-	G	G,A	AGG,AAG	R,K

As dbSNP131 contained some redundancy, it was necessary to check the cSNP file for redundant records. The rs numbers of SNPs are sometimes unreliable identifiers, thus the genomic locations of the cSNPs were used to identify redundant cSNP records. Redundant cSNP records were identified as more than one cSNP on the same chromosome, at the same position in the chromosome and on the same DNA strand. When redundant cSNP records were identified, one copy of the cSNP records was retained in the non-redundant cSNP file and the additional redundant cSNP records removed (**B.sSNPs.Remove_Redundancies()** and **Fig.2.2.c**).

After the errors and redundant records had been removed, the sSNPs could be identified from the non-redundant cSNPs. The amino acids given by dbSNP131 in each of the cSNP records were used to identify the sSNPs. As sSNPs do not change the amino acid encoded, if the encoded amino acids given by dbSNP were the same for the different alleles of the cSNP, the cSNP was identified as a sSNP. cSNPs where the encoded amino acids given by dbSNP were different for the different alleles of the cSNP, were identified as nsSNPs and discarded from the dataset(**B.sSNPs.Identify_sSNPs()** and **Fig.2.2.d**).

The identified sSNPs were assessed to determine whether any had previously been associated with a trait or disease in GWAS. The UCSC main table browser in Galaxy was used to retrieve all of the SNPs that had previously been associated with a trait or disease in GWAS, from the GWAS catalogue (**Fig.2.7**). The SNPs within the GWAS catalogue were associated with a trait or disease with a p-value less than 1.0×10^{-5} and were published between 2005/03/10 and 2011/02/10. The GWAS file was downloaded onto a local desktop computer and any redundant records removed. The positions of the SNPs in the chromosomes were used to determine whether any of the sSNPs had previously been associated with a trait or disease in GWAS. The sSNPs that had previously been associated with a trait or disease in GWAS were identified as sSNPs at the same positions in the chromosomes as SNPs in the GWAS catalogue (**B.sSNPs.GWAS()** and **Fig.2.2.e**).

The sSNPs were assessed to determine which occurred within genes known to be associated with disease in OMIM (disease genes). The UCSC main table browser in Galaxy was used to retrieve all genes known to be associated with disease in OMIM (**Fig.2.8**). The disease genes file retrieved from OMIM contained the descriptions of the diseases and the MIM accession numbers for the disease genes. The disease genes file retrieved from OMIM did not include the positions of the genes in the chromosomes. To retrieve the additional data, the UCSC main table browser in Galaxy was used to retrieve all of the genes published with OMIM (**Fig.2.9**). The genes file retrieved from OMIM contained the positions of the genes in the chromosomes, the genes names and the MIM accession numbers of the genes. The genes file retrieved from OMIM did not describe which of the genes were associated with a disease and which were not. By combining the records in the two files retrieved from OMIM, one file could be created that contained all of the data required. Galaxy was used to join the disease genes file and the genes file (**Fig.2.10**). The records in the two files were joined based on the MIM accession numbers. Only genes with the same MIM accession numbers in both the disease genes file and the genes file were thus retained. The resulting file contained all the genes known to be associated with disease in OMIM, with all the data required to identify sSNPs within disease genes.

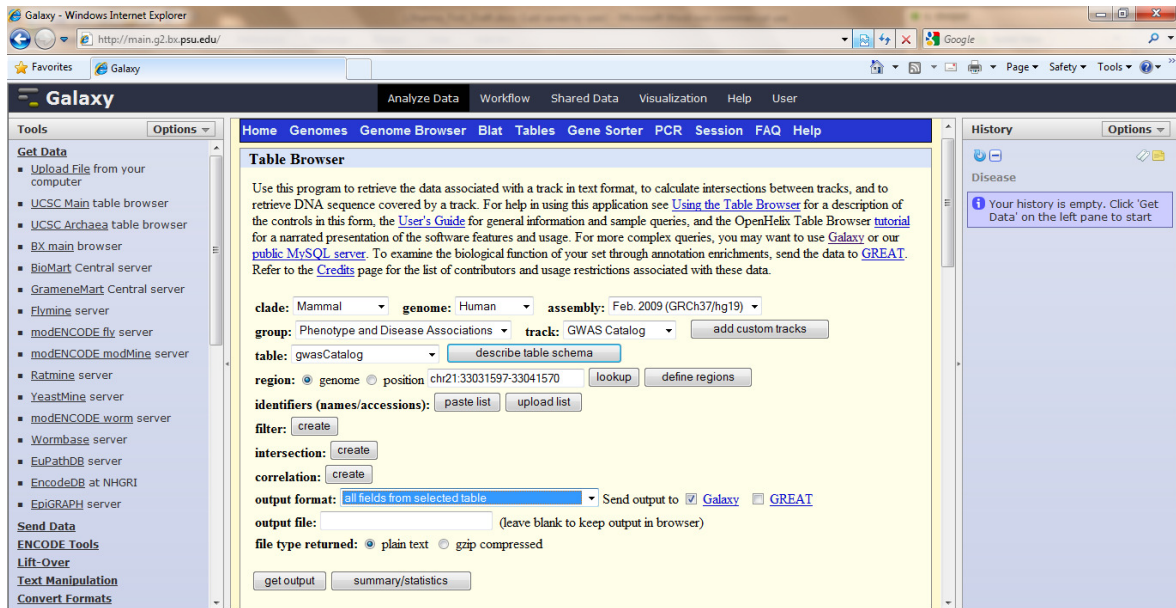


Figure 2.7: The screen-shot of the UCSC main table browser in Galaxy, used to retrieve the SNPs associated with a trait or disease in GWAS, from the GWAS catalogue.

The SNPs previously associated with a trait or disease in GWAS were retrieved by selecting the “phenotype and disease associations group”, the “GWAS catalog track” and the “gwasCatalog table”, from the UCSC main table browser in Galaxy.

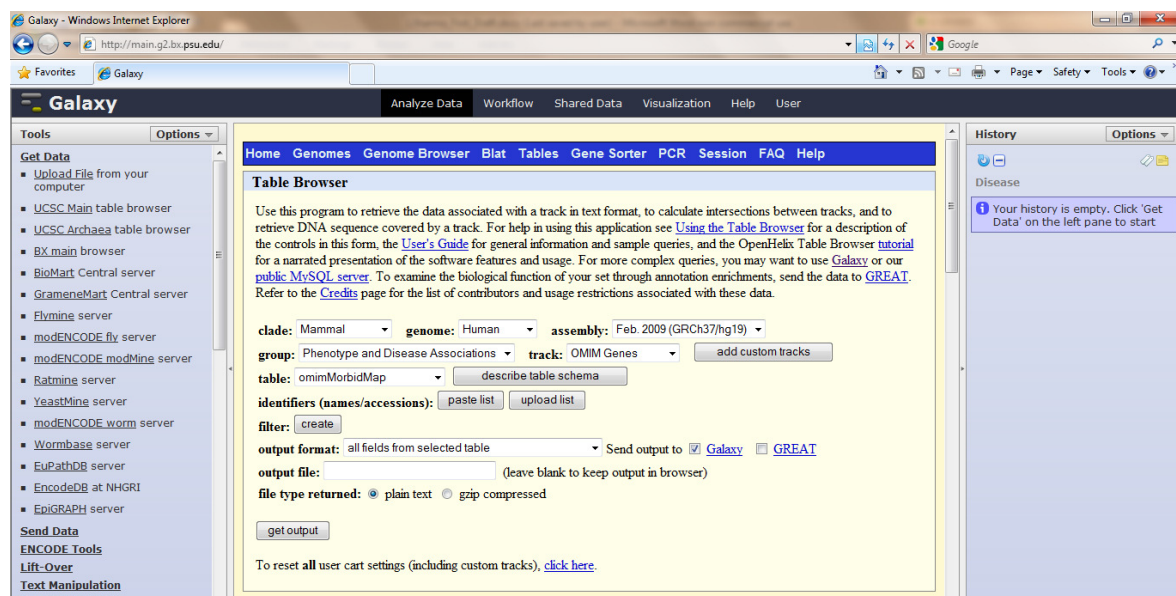


Figure 2.8: The screen-shot of the UCSC main table browser in Galaxy, used to retrieve the genes known to be associated with disease in OMIM.

The genes known to be associated with disease in OMIM were retrieved by selecting the “phenotype and disease associations group”, the “OMIM genes track” and the “omimMorbidityMap table”, from the USCS main table browser in Galaxy.

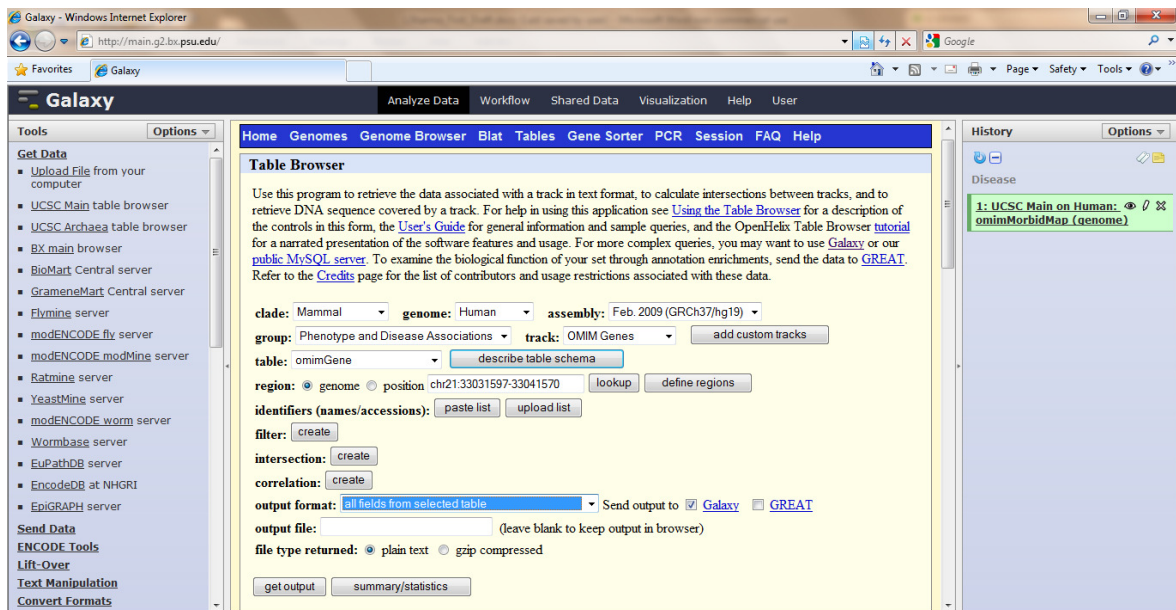


Figure 2.9: The screen-shot of the UCSC main table browser in Galaxy, used to retrieve all genes published within OMIM. All of the genes published within OMIM were retrieved by selecting the “phenotypes and disease associations group”, the “OMIM genes track” and the “omimGene table”, from the UCSC main table browser in Galaxy.

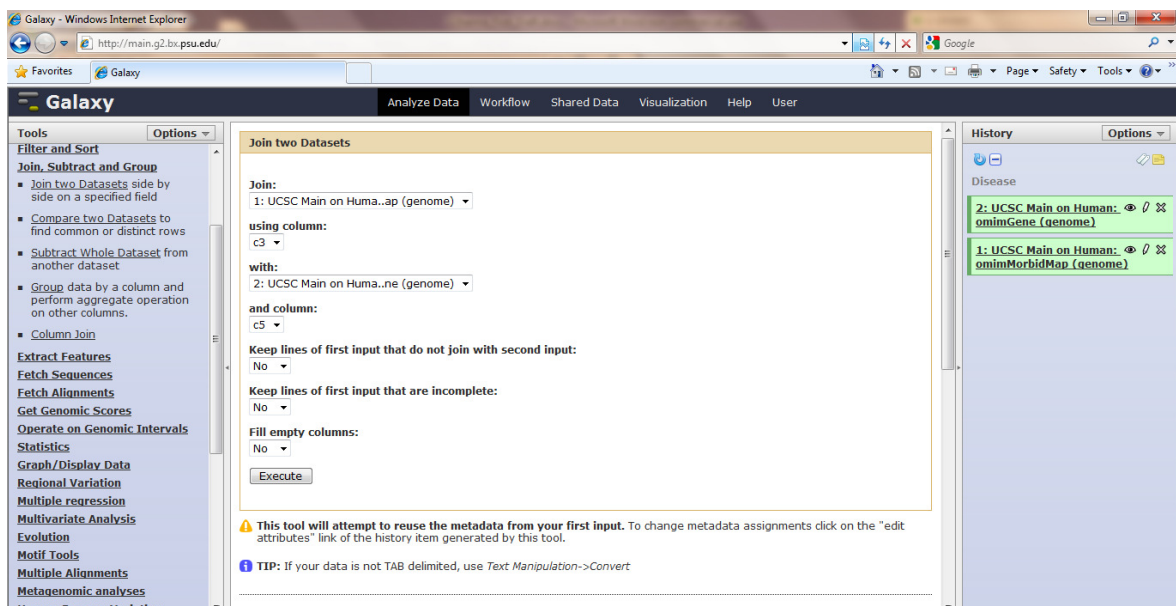


Figure 2.10: The screen-shot of the tool used in Galaxy to join the file containing the genes known to be associated with disease in OMIM and the file containing all of the genes published within OMIM.

The file containing the disease genes and the file containing all of the genes within OMIM were joined by selecting the “join, subtract and group” and “join the two datasets side by side on a specified field” options, under the tools section of Galaxy. The two files were joined based on the OMIM accession numbers (MIM numbers), which were in column three (c3) of the file containing the disease genes and column five (c5) of the file containing all of the genes within OMIM.

The file containing the disease genes was downloaded from Galaxy onto a local desktop computer. Records with data missing were removed. The file was then checked for any redundancy. Redundant records were defined as genes with the same start and end positions in the chromosomes and on the same DNA strand. When redundant records were identified, one copy of the records was retained and the additional redundant records removed. If a redundant records occurred because a single gene was associated with more than one disease, the records were combined to produce one record. The record thus described one gene associated with multiple diseases. After the records with data missing and redundant records were removed, the sSNPs that occurred within disease genes were identified. The sSNPs within genes known to be associated with disease in OMIM were identified as sSNPs whose positions in the chromosomes were within the start and end positions in the chromosomes of the disease genes (**B.sSNPs.OMIM()** and **Fig.2.2.f**).

The sSNPs were assessed to determine whether any had previously been associated with a trait or disease in GWAS and occurred within genes known to be associated with disease OMIM. These sSNPs were identified based on their positions in the chromosomes. The sSNPs that were in both the file containing the sSNPs associated with a trait or disease in GWAS and the file containing the sSNPs within disease genes were identified. The sSNPs in both files were the sSNPs that had previously been associated with a trait or disease in GWAS and occurred within genes known to be associated with disease in OMIM (**B.General.GWAS_And_OMIM()** and **Fig.2.2.g**).

In-order to determine the potential functional impact of each sSNP, it was necessary to compare the allele changes from the reference sSNP allele to each of the variant sSNP alleles (**B.sSNPs.SSNS()** and **Fig.2.2.h**). Each change from the reference sSNP allele to a variant sSNP allele was defined as a synonymous single nucleotide substitution (sSNS). Since a number of the sSNPs had more than one variant allele, a separate sSNS record was created for each change from the reference sSNP allele to each of the variant sSNP alleles.

The sSNPs with two alleles represented one sSNS, from the reference allele to the variant allele. The sSNPs with three alleles represented two sSNS. One of the sSNS was from the reference allele to the first variant allele. The second sSNS was from the reference allele to the second variant allele. The sSNPs with four alleles represented three sSNS, one sSNS from the reference allele to each of the three variant alleles (**Table.2.3**).

Table 2.3: Examples of the sSNS identified from the sSNPs

sSNP records		sSNS records
Reference allele	Observed alleles	
A	A, C	A>C
A	A, C, G	A>C and A>G
A	A, C, G, T	A>C, A>G and A>T

The sSNS records were separated into different files according to the chromosomes on which the sSNS resided (**B.sSNPs.Create_Chrom_Files()** and **Fig.2.2.i**). The separation of the sSNS records according to chromosomes enabled the sSNS records to be joined to other records more quickly. For instance, if the record of an exon on chromosome 1 had to be joined to a sSNS record, it would waste time and computation resources to search the entire sSNS file for the sSNS record. By separating the sSNS according to chromosomes, only the chromosome 1 file needs to be opened and searched to find the sSNS record.

2.3. sSNS predicted to potentially cause an extreme change in codon-usage frequency

The sSNS were assessed to determine whether any could potentially cause an extreme change in codon-usage frequency (CUF). The methods used to identify sSNS that potentially cause an extreme change in CUF are depicted in Figure 2.11. The human CUF table was used to calculate the change in CUF caused by each sSNS (**Table.C.1**). To standardize the CUF values in the table, the CUFs were calculated as a percentage of the total CUF of all the codons that encode a specific amino acid. For instance, alanine is encoded by four amino acids, thus, the CUFs of all four codons were summed ($27.73 + 18.45 + 15.82 + 7.37 = 69.37$). The CUF of each codon that encodes alanine was then calculated as a percentage of the summed CUF (e.g. $(27.73/69.37)*100 = 39.97\%$) (**Table.2.4**).

Table 2.4: Examples of the calculations used to standardize the codon-usage frequency values in the human codon-usage frequency table

Amino acid	Codon	CUF	Calculation	CUF standardized as a percentage (%)
Ala	GCC	27.73	$=(27.73/69.37)*100$	39.97
Ala	GCT	18.45	$=(18.45/69.37)*100$	26.60
Ala	GCA	15.82	$=(15.82/69.37)*100$	22.81
Ala	GCG	7.37	$=(7.37/69.37)*100$	10.62
Total		69.37		100.00

The reference codon was defined as the codon that contained the reference sSNS allele. The variant codon was defined as the codon that encoded the same amino acid as the reference codon, but contained the variant sSNS allele. The standardized human CUF table was used to determine the CUF of each of the reference and variant codons. The change in CUF caused by a sSNS was calculated by subtracting the reference CUF from the variant CUF. A change in CUF less than zero indicated that a sSNS changed a more frequently used codon to a less frequently used codon (a decrease in CUF). A change in CUF greater than zero indicated that a sSNS changed a less frequently used codon to a more frequently used codon (an increase in CUF) (**B.Codons.Codon_Usage_Change()** and **Fig.2.11.a**).

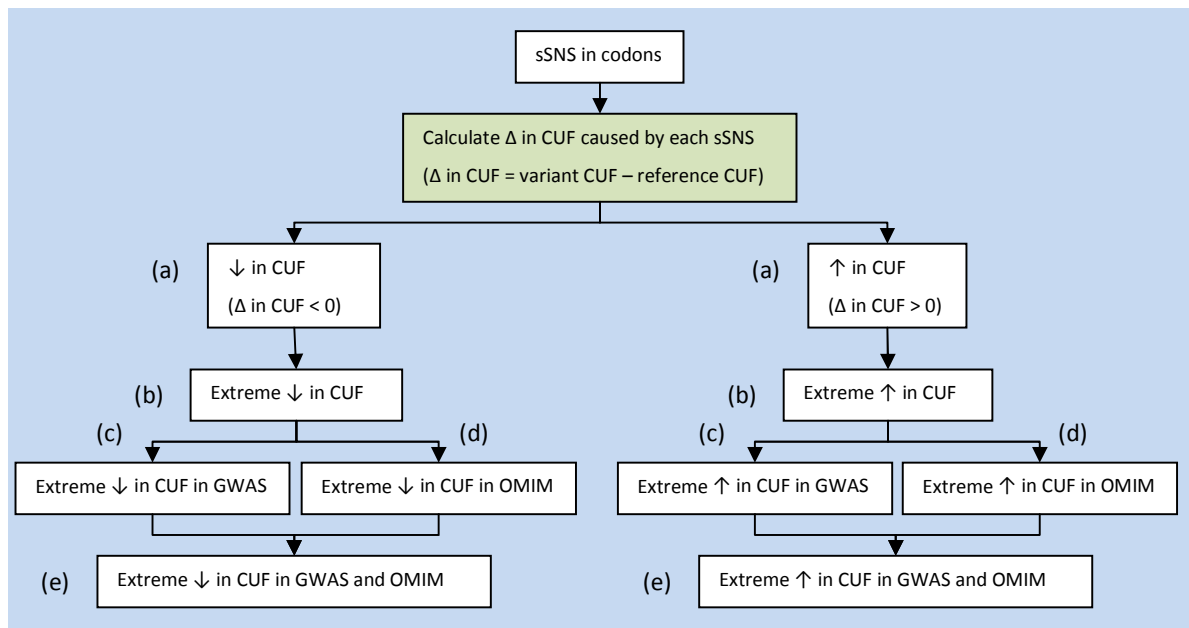


Figure 2.11: The methods used to identify sSNS that potentially cause an extreme change in codon-usage frequency, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM.

- (a)** sSNS predicted cause a change in codon-usage frequency
- (b)** sSNS predicted to cause an extreme change in codon-usage frequency
- (c)** sSNS predicted to cause an extreme change in codon-usage frequency, which have previously been associated with a trait or disease in GWAS
- (d)** sSNS predicted to cause an extreme change in codon-usage frequency, which occur within genes known to be associated with disease in OMIM
- (e)** sSNS predicted to cause an extreme change in codon-usage frequency, which have previously been associated with a trait or disease in GWAS and occur within genes known to be associated with disease in OMIM

The sSNS that were predicted to cause a change in CUF were assessed to determine whether any could potentially cause an extreme change in CUF. All of the observed changes in CUF were ordered in increasing order. The 10% most extreme decrease and 10% most extreme increase in CUF were chosen to represent an extreme change in CUF (**B.Extreme_CUF()** and **Fig.2.11.b**).

The sSNS predicted to potentially cause an extreme change in CUF were assessed to determine whether any had previously been associated with a trait or disease in GWAS. The file containing the sSNS predicted to potentially cause an extreme change in CUF and the file

containing the sSNPs previously associated with a trait or disease in GWAS (created in Methods 2.1) were joined. The two files were joined based on the sSNS positions in the chromosomes. After the joining of the two files, the resulting file (GWAS file) contained the sSNS predicted to potentially cause an extreme change in CUF, which had previously been associated with a trait or disease in GWAS (**B.Codons.Extreme_CUF_In_Disease()** and **Fig.2.11.c**).

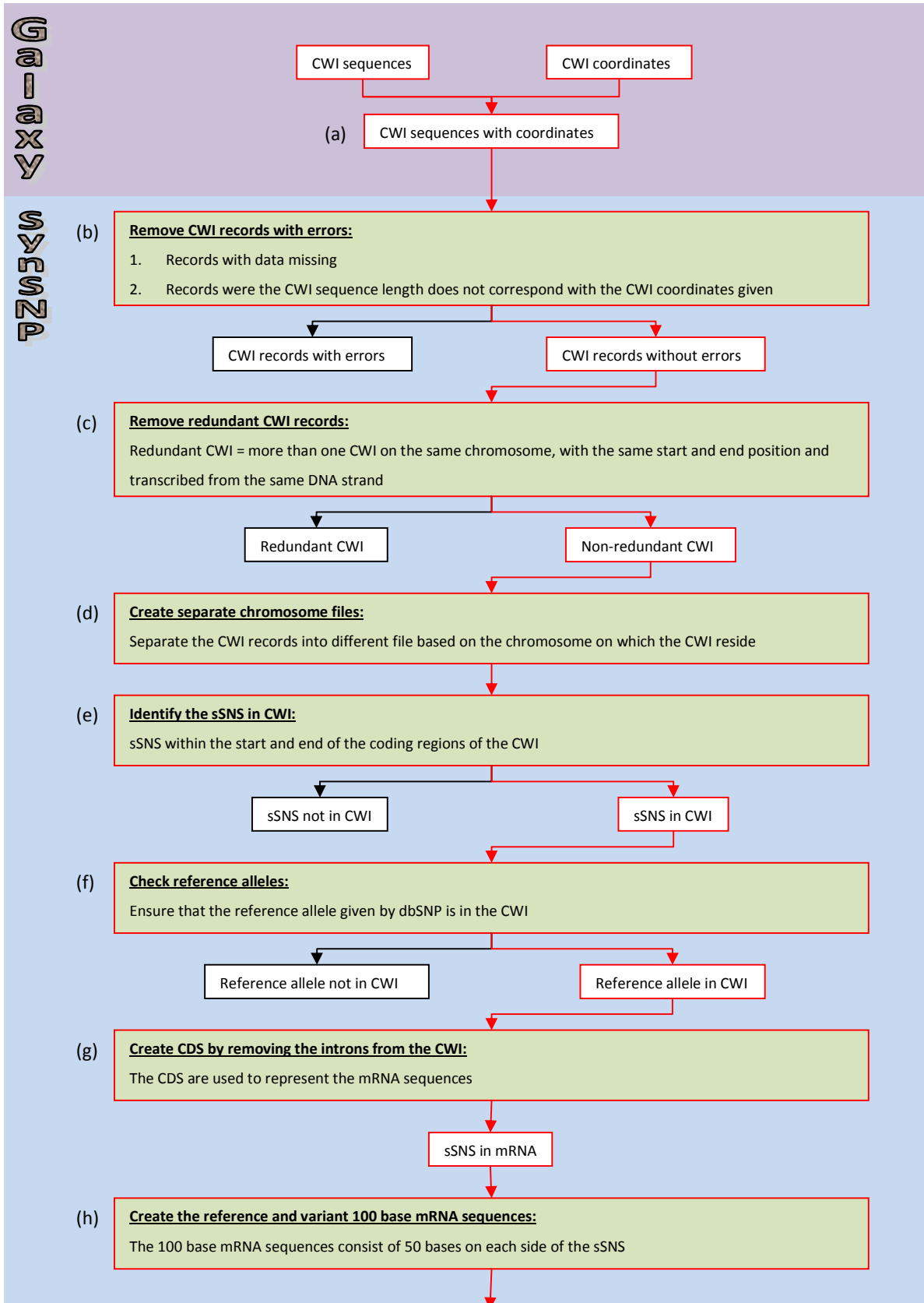
The sSNS predicted to potentially cause an extreme change in CUF were assessed to determine whether any occurred within genes known to be associated with disease in OMIM. The file containing the sSNS predicted to potentially cause an extreme change in CUF and the file containing the sSNPs within disease genes (created in Methods 2.1) were joined. The two files were joined based on the sSNS positions in the chromosomes. The joined file (OMIM file) contained the sSNS predicted to potentially cause an extreme change in CUF, which were within genes known to be associated with disease in OMIM (**B.Codons.Extreme_CUF_In_Disease()** and **Fig.2.11.d**).

The sSNS predicted to potentially cause an extreme change in CUF were assessed to determine whether any had previously been associated with a trait or disease in GWAS and occurred within genes known to be associated with disease in OMIM. The GWAS and OMIM files were joined, based on the sSNS positions in the chromosomes. Only the records of the sSNS in both the GWAS and OMIM file were joined. After the joining of the two files, the resulting file contained the sSNS predicted to potentially cause an extreme change in CUF which had previously been associated with a trait or disease in GWAS and occurred within genes known to be associated with disease in OMIM (**B.Codons.Extreme_CUF_In_Disease()** and **Fig.2.11.e**).

2.4. sSNS predicted to potentially cause an extreme change in the minimum free-energy of local mRNA secondary structures

The sSNS were assessed to determine whether any could potentially cause an extreme change in the minimum free-energy of local mRNA secondary structures (MFE). Minimum free-energy is a measure of the stability of an mRNA secondary structure. A sSNS that causes an extreme change in MFE could potentially have a functional impact, through a change in the rate of translation or mRNA degradation. The methods used to predict sSNS that could potentially cause an extreme change in MFE are depicted in Figure 2.12.

The positions of the sSNS were given by dbSNP131 as genomic coordinates. To determine the positions of the sSNS in mRNA it was, therefore, necessary to have the genomic coordinates of the mRNA sequences. The genomic coordinates are determined according to the reference genome assembly, which includes the coding regions and the introns. As the genomic coordinates of mRNA sequences could not be retrieved, it was necessary to use the genomic coordinates of the coding regions with introns (CWI). By removing the introns, the coding region sequences (CDS) that code for protein could be created, and the genomic coordinates used to identify the positions of the sSNS in the CDS. The CDS were used to represent the mRNA sequences. An unfortunate consequence of using CDS to represent the mRNA sequences is that the CDS does not include the 5' and 3' untranslated regions (UTRs) (Fig.2.13).



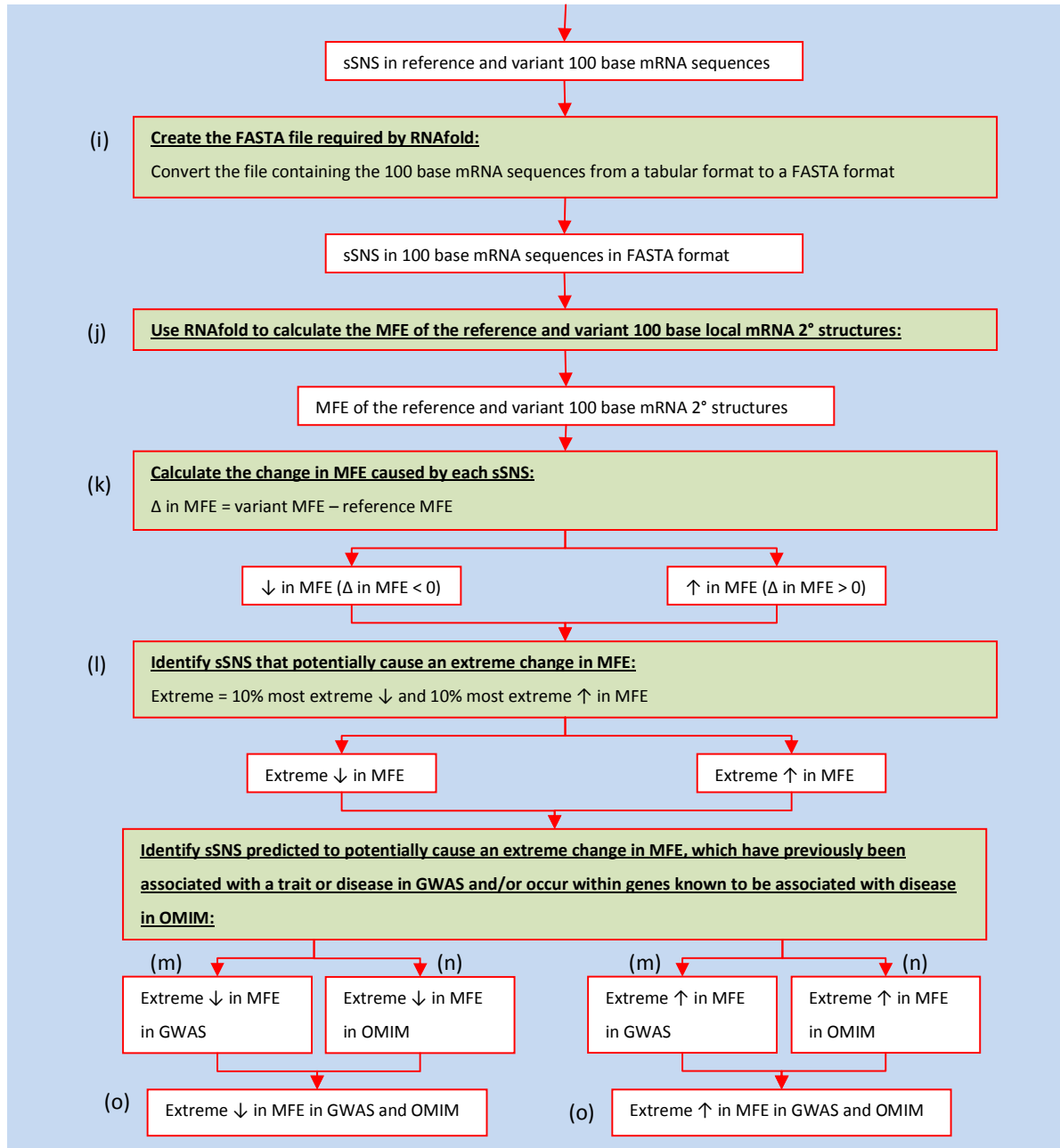


Figure 2.12: The methods used to identify sSNS that potentially cause an extreme change in the minimum free-energy of local mRNA secondary structures, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM.

- Green box = the methods used
- White box = the records discarded from the datasets
- Red box = the records retained and used as input in the next step of analysis

- **CWI** = coding sequences with introns
- **CDS** = coding sequences (used to represent the mRNA sequences, excluding the untranslated regions)
- **MFE** = the minimum free-energy of the 100 base local mRNA secondary structures
- **in GWAS** = sSNS previously associated with a trait or disease in GWAS
- **in OMIM** = sSNS within genes known to be associated with a disease in OMIM



Figure 2.13: The structure of mature mRNA.

 = the region used to represent the mRNA sequences in this project

UTR = untranslated region

The CWI coordinates and the CWI sequences could only be retrieved in two separate files. The two files would then have to be joined, in-order to acquire the CWI coordinates with sequences in one file. The UCSC main table browser in Galaxy was used to retrieve the CWI coordinates from RefSeq (**Fig.2.14**). The CWI sequences were also retrieved from RefSeq using Galaxy (**Fig.2.15**). The file containing the CWI sequences was in FASTA format. In-order to join the two files, the file containing the CWI sequences had to be converted to a tabular format. The file was converted from a FASTA format to a tabular format using Galaxy (**Fig.2.16**). Galaxy was used to join the file containing the CWI coordinates and the file containing the CWI sequences, based on the RefSeq accession numbers (**Fig.2.17**). The file containing the CWI coordinates and sequences was downloaded from Galaxy onto a local desktop computer (**Fig.2.12.a**).

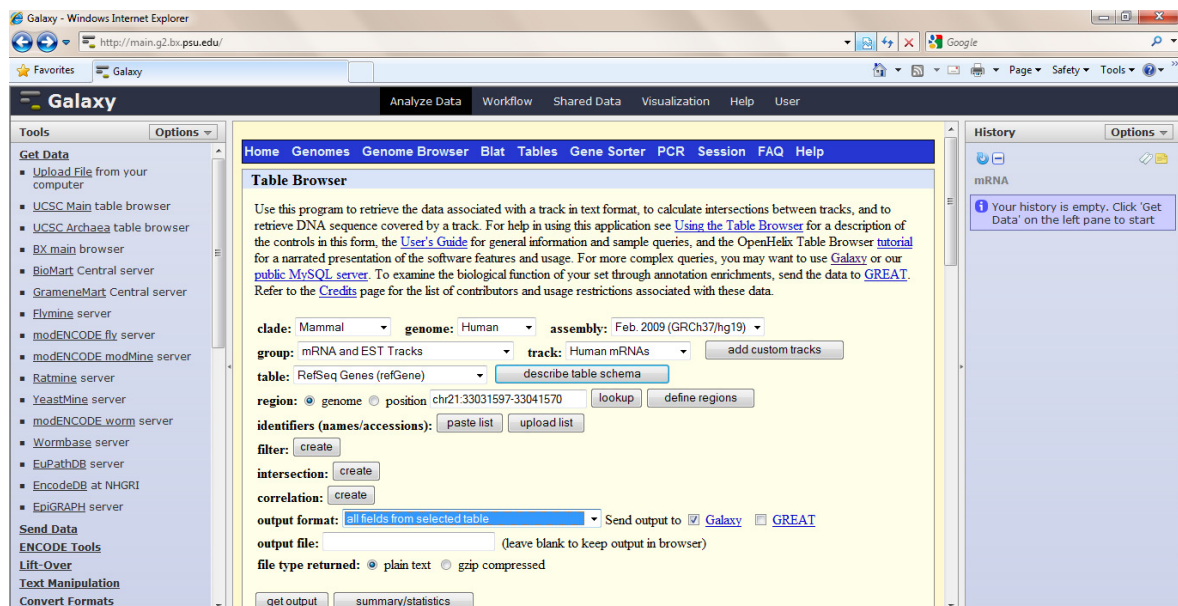


Figure 2.14: The screen-shot of the UCSC main table browser in Galaxy, used to retrieve the coordinates of the coding sequences with introns.

The coordinates of the coding sequences with introns were retrieved from RefSeq by selecting the “mRNA and EST group”, the “human mRNAs track” and the “RefSeq genes (refGene) table”, from the UCSC main table browser in Galaxy. All of the fields from the selected table were chosen.

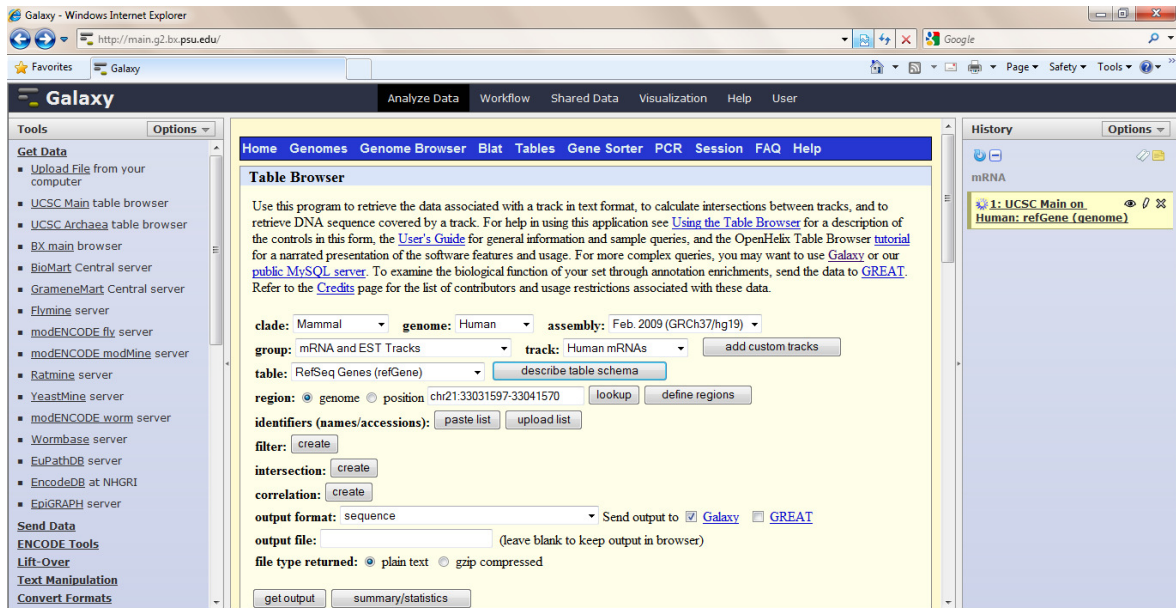


Figure 2.15: The screen-shot of the UCSC main table browser in Galaxy, used to retrieve the sequences of the coding sequences with introns.

The coordinates of the coding sequences with introns were retrieved from RefSeq by selecting the “mRNA and EST group”, the “human mRNA track” and the “RefSeq genes (refGene) table”, from the UCSC main table browser in Galaxy. The output format was selected to output the sequences.

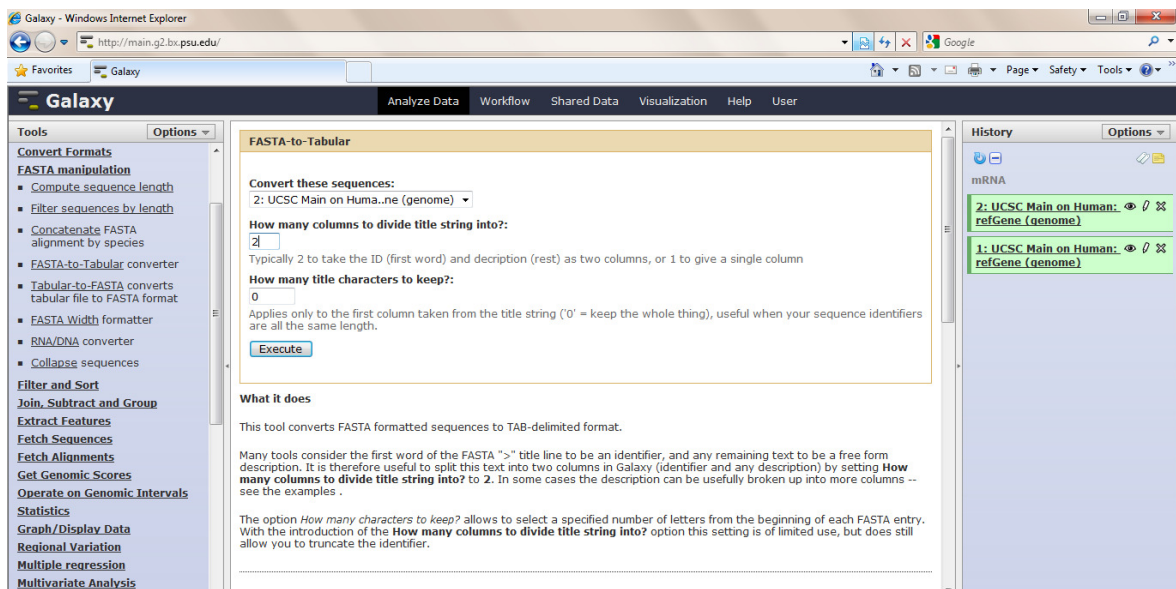


Figure 2.16: The screen-shot of the tool used in Galaxy to convert the file containing the sequences of the coding sequences with introns from a FASTA format to a tabular format.

To convert the coding sequences with introns from a tabular format to a FASTA format the “FASTA manipulation” and “FASTA-to-tabular converter” was selected under the tools section of Galaxy.

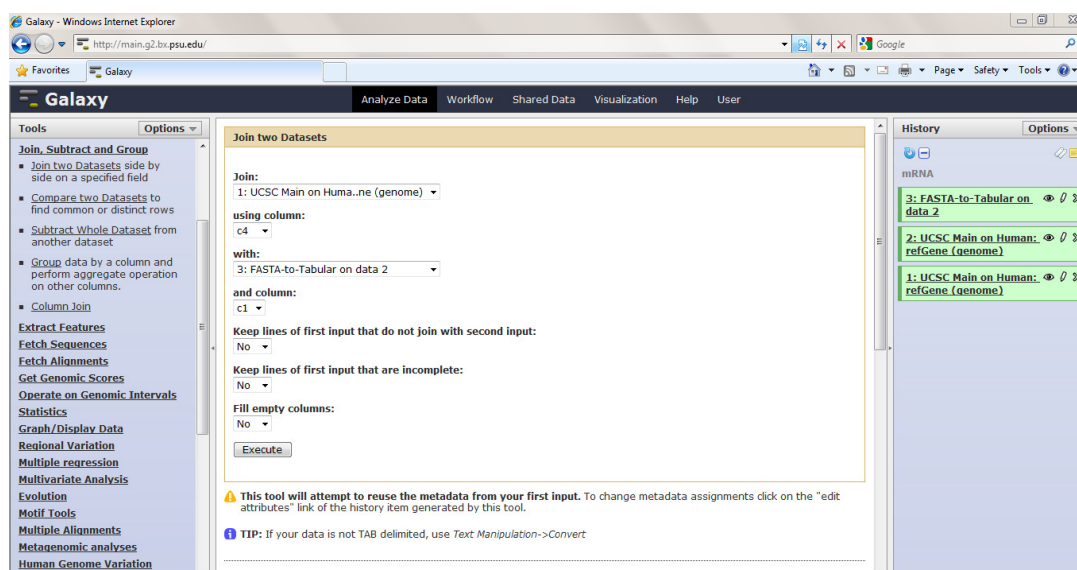


Figure 2.17: The screen-shot of the tool used in Galaxy to join the file containing the coordinates of the coding sequences with introns and the file containing the sequences of the coding sequences with introns.

The two files were joined by selecting the “join, subtract and group” and the “join two datasets side by side on a specified field” options under the tools section of Galaxy. The files were joined based in the RefSeq accession numbers, which were in column four (**c4**) of the file containing the coordinates of the coding sequences with introns, and in column one (**c1**) of the file containing the sequences of the coding sequences with introns.

The CWI file was checked for errors. The error check removed any CWI records that had data missing. The error check also removed any CWI records where the CWI sequence lengths did not correspond to the CWI coordinates given (**B.mRNA.Remove_Errors()** and **Fig.2.12.b**). After the errors were removed, the CWI records were checked for any redundancy. Redundant records were defined as more than one CWI record, where the CWI were on the same chromosome, with the same start and end positions in the chromosome and on the same DNA strand. Only one copy of each redundant set of records was retained. The additional redundant records were removed from the dataset (**B.General.Remove_Redundancies()** and **Fig.2.12.c**). In-order to more quickly identify sSNS in CWI, the CWI records were separated into different files according to the chromosomes on which the CWI resided (**B.General.Create_Chrom_Files()** and **Fig.2.12.d**). The sSNS in CWI were identified as sSNS whose positions in the chromosomes were within the start and end positions of the CWI in the chromosomes (**B.General.Join_Files()** and **Fig.2.12.e**). After the sSNS in CWI had been identified, the records were checked to ensure that the reference

alleles given by NCBI/dbSNP131 were in the CWI sequences. If the reference alleles given were not in the sequences, the record were removed (**B.General.Check_Alleles()** and **Fig.2.12.f**).

The introns of the CWI were given in uppercase and the coding exons of the CWI were given in lowercase. By removing the introns, given in lowercase, the CDS of the mRNA sequences were created. The UTRs were not included in this study. After the introns had been removed, the positions of the sSNS in the mRNA sequences were determined (**B.mRNA.Create_mRNA()** and **Fig.2.12.g**).

It would have required several days to calculate the minimum free-energy of the global mRNA secondary structure for each of the mRNA sequences. For this reason, the minimum free-energies of local mRNA secondary structures were compared. It had previously been determined that a 100 base window allowed for the most accurate predictions of local mRNA secondary structures (Johnson et al., 2011). To create the reference 100 base mRNA sequences, 50 bases on each side of the sSNS were sliced from the mRNA sequences. The variant 100 base mRNA sequences were then created by replacing the reference alleles in the 100 base mRNA sequences, with the variant sSNS alleles (**B.mRNA.Ref_And_Var()** and **Fig.2.12.h**).

RNAfold, which is part of the Vienna RNA package, was used to calculate the MFE (<http://rna.tbi.univie.ac.at/cgi-bin/RNAfold.cgi>). To calculate the MFE, RNAfold required the mRNA sequences in FASTA format. The file containing the reference and variant 100 base mRNA sequences was converted from a tab-delaminated format to a FASTA format (**B.mRNA.Fasta()** and **Fig.2.12.i**). RNAfold was used to calculate the minimum free-energy of each of the reference and variant 100 base local mRNA secondary structures, from the 100 base mRNA sequences (**B.mRNA.RNAfold()** and **Fig.2.12.j**). A lower (more negative) MFE indicates that the mRNA secondary structure contains more base-pairs. An mRNA secondary structure with a lower MFE will be more stable. The change in MFE caused by a sSNS was

calculated by subtracting the reference MFE from the variant MFE

(B.mRNA.Change_In_MFE() and Fig.2.12.k). A change in MFE less than zero indicated that a sSNS potentially caused a decrease in MFE. A change in MFE greater than zero indicated that a sSNS potentially caused an increase in MFE. The observed changes in MFE caused by sSNS were ordered in increasing order. The 10% most extreme decrease and 10% most extreme increase in MFE were chosen to represent an extreme change in MFE

(B.General.Cut_Off_Values() and Fig.2.12.l).

The sSNS predicted to potentially cause an extreme change in MFE were assessed to determine whether any had previously been associated with a trait or disease in GWAS. The file containing the sSNS predicted to potentially cause an extreme change in MFE, and the file containing the sSNPs previously associated with a trait or disease in GWAS (created in Methods 2.1) were joined. The two files were joined based on the sSNS positions in the chromosomes. After the two files were joined, the resulting file (GWAS file) contained the sSNS predicted to potentially cause an extreme change in MFE, which had previously been associated with a trait or disease in GWAS **(B.General.GWAS_Or_OMIM() and Fig.2.12.m).**

The sSNS predicted to potentially cause an extreme change in MFE were assessed to determine which occurred within genes known to be associated with disease in OMIM. The file containing the sSNS predicted to potentially cause an extreme change in MFE, and the file containing the sSNPs within disease genes (created in Methods 2.1), were joined. The two files were joined based on the sSNS positions in the chromosomes. After the two files were joined, the resulting file (OMIM file) contained the sSNS predicted to potentially cause an extreme change in MFE, which occurred within genes known to be associated with disease in OMIM **(B.General.GWAS_Or_OMIM() and Fig.2.12.n).**

The sSNS predicted to potentially cause an extreme change in MFE were assessed to determine whether any had previously been associated with a trait or disease in GWAS and occurred within disease genes. The GWAS and OMIM files were joined, based on the sSNS

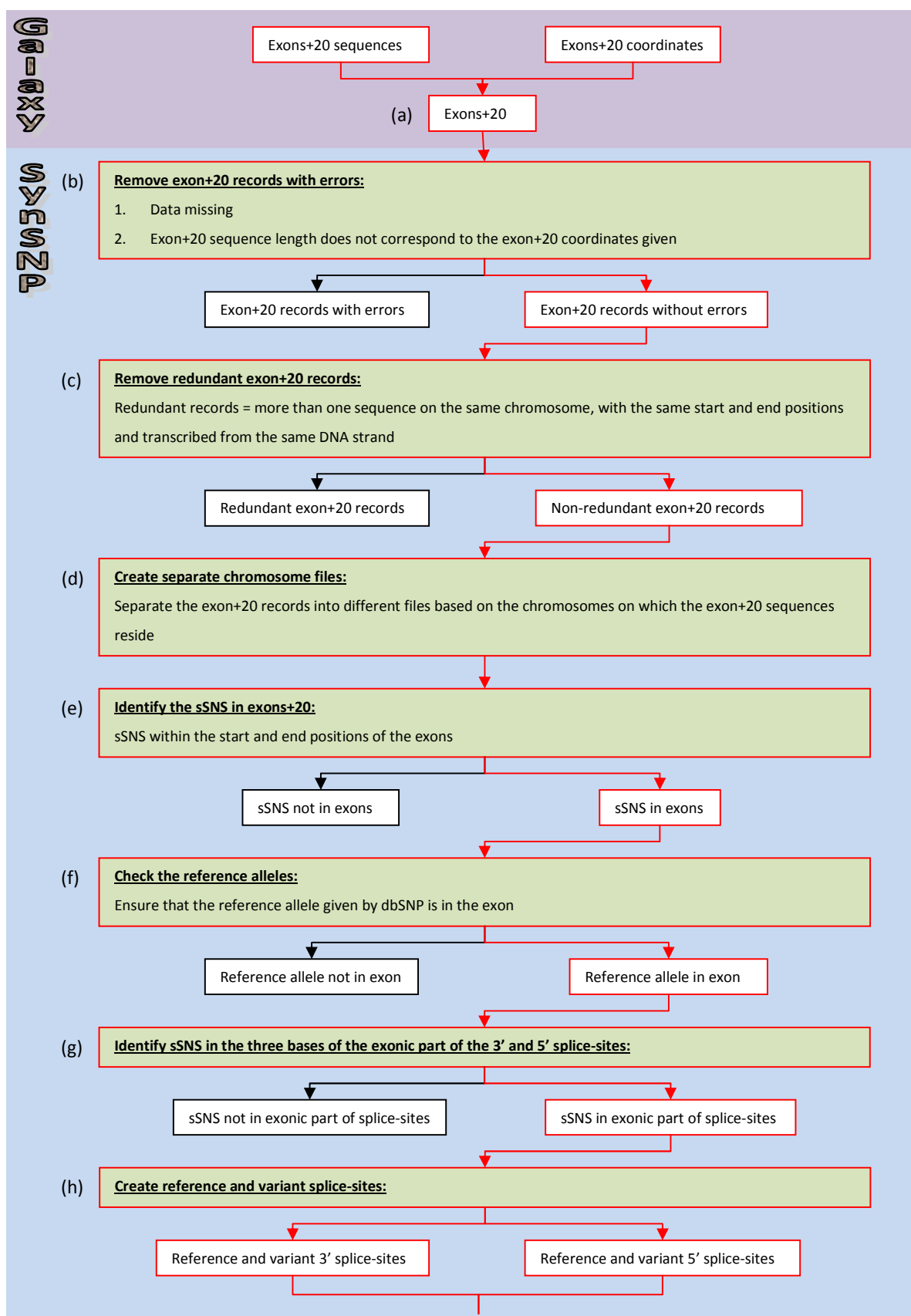
positions in the chromosomes. After the two files were joined, the resulting file contained the sSNS predicted to potentially cause an extreme change in MFE, which had previously been associated with a trait or disease in GWAS and occurred within genes known to be associated with disease in OMIM (**B.General.GWAS_And_OMIM()** and **Fig.2.12.o**).

2.5. sSNS predicted to potentially cause an extreme change in the maximum-entropy score of splice-sites

The sSNS were assessed to determine whether any could potentially cause an extreme change in the maximum-entropy scores of splice-sites (ME). The maximum-entropy score is a measure of the splicing signal strength of a splice-site. The higher the ME the more likely it is that splicing will occur at the splice-site. The methods used to predict sSNS that could potentially cause an extreme change in ME are depicted in Figure 2.18.

The splice-sites are found at the intron-exon junctions. Generally, three bases of the splice-site motifs are within the exons and between six and twenty bases in the introns. By definition sSNPs are not in introns, therefore, only sSNS within the exonic part of the splice-sites were assessed. In-order to calculate ME, the entire splice-site motif was required. Although only sSNS in the exonic part of the splice-sites were assessed, the flanking intron sequences were required to calculate ME.

The splice-sites could not be directly retrieved, thus the exons with flanking introns were retrieved from RefSeq. The splice-sites were identified from the exon-intron junctions. The UCSC main table browser in Galaxy was used to retrieve the coordinates of the exons with an additional 20 bases into the introns at each end of the exons (exons+20) from RefSeq (**Fig.2.19**). The sequences corresponding to the coordinates of the exons+20 were retrieved from RefSeq, using Galaxy (**Fig.2.20**). The file containing the exon+20 sequences and coordinates was downloaded from Galaxy onto a local desktop computer (**Fig.2.18.a**).



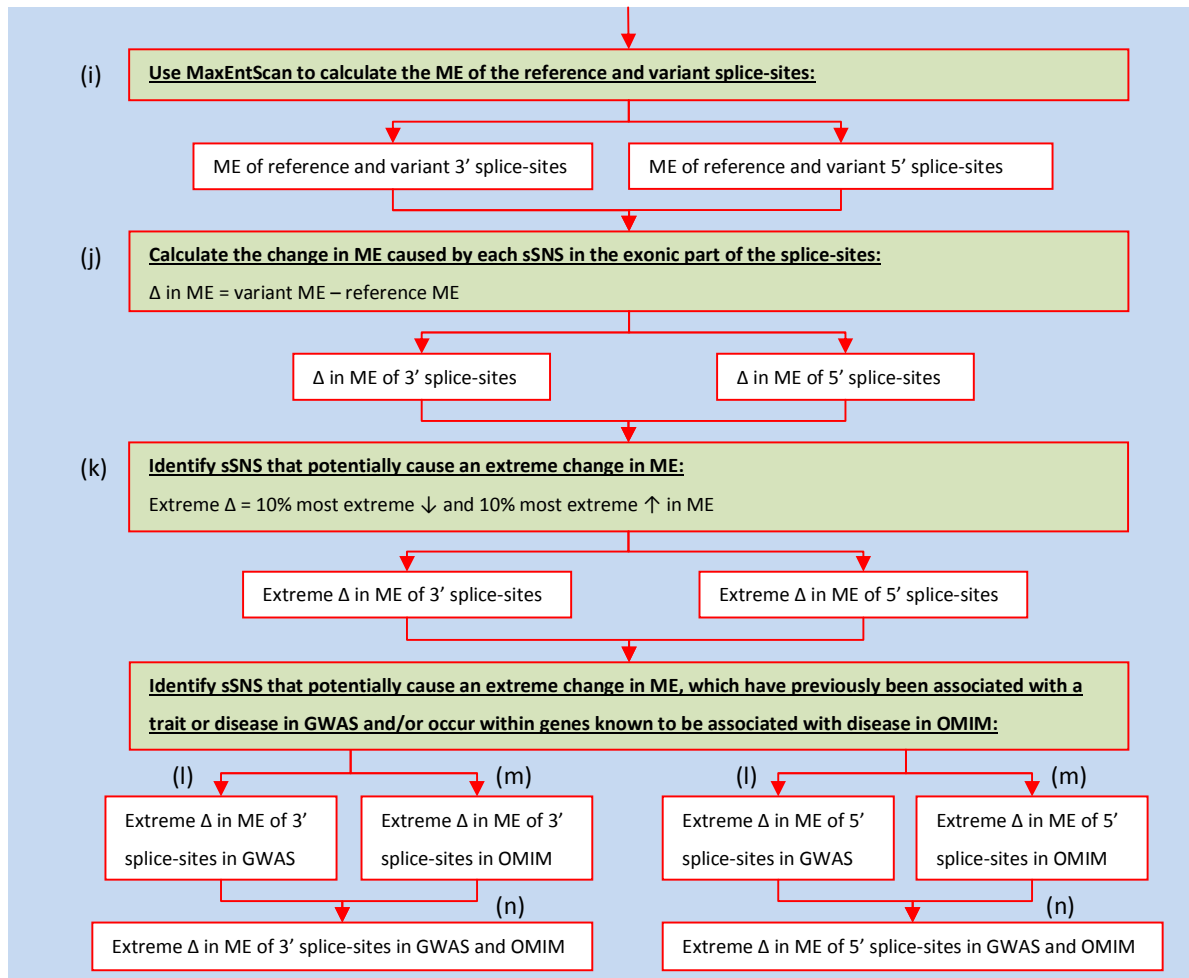


Figure 2.18: The methods used to identify sSNS that potentially cause an extreme change in the maximum-entropy score of splice-sites, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM.

 = the methods used

 = the records discarded from the datasets

 = the records retained and used as input in the next step of analysis

- **Exon+20** = the exon sequence with 20 bases into the flanking introns, on each side of the exon
- **ME** = the maximum-entropy score of splice-sites
- **in GWAS** = sSNS previously associated with a trait or disease in GWAS
- **in OMIM** = sSNS within genes known to be associated with a disease in OMIM

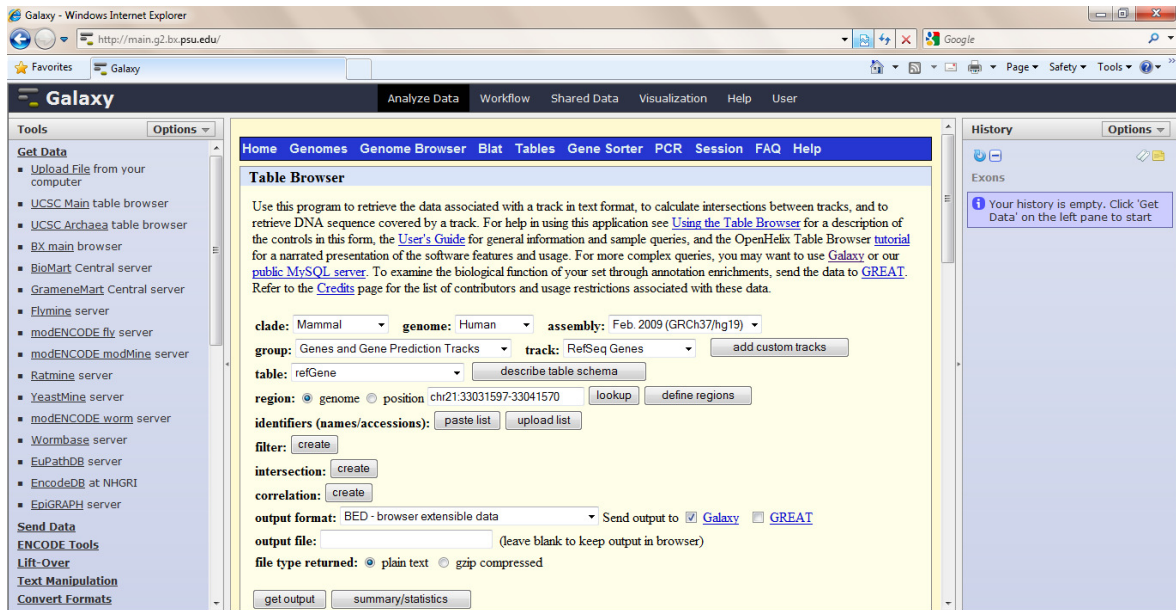


Figure 2.19: The screen-shot of the UCSC main table browser in Galaxy, used to retrieve the coordinates of the exon with 20 bases at each end sequences.

The coordinates of the exons with 20 bases at each end were retrieved from RefSeq by selecting the “genes and gene prediction tracks group”, the “RefSeq genes track” and the “refGene table”, from the UCSC main table browser in Galaxy.

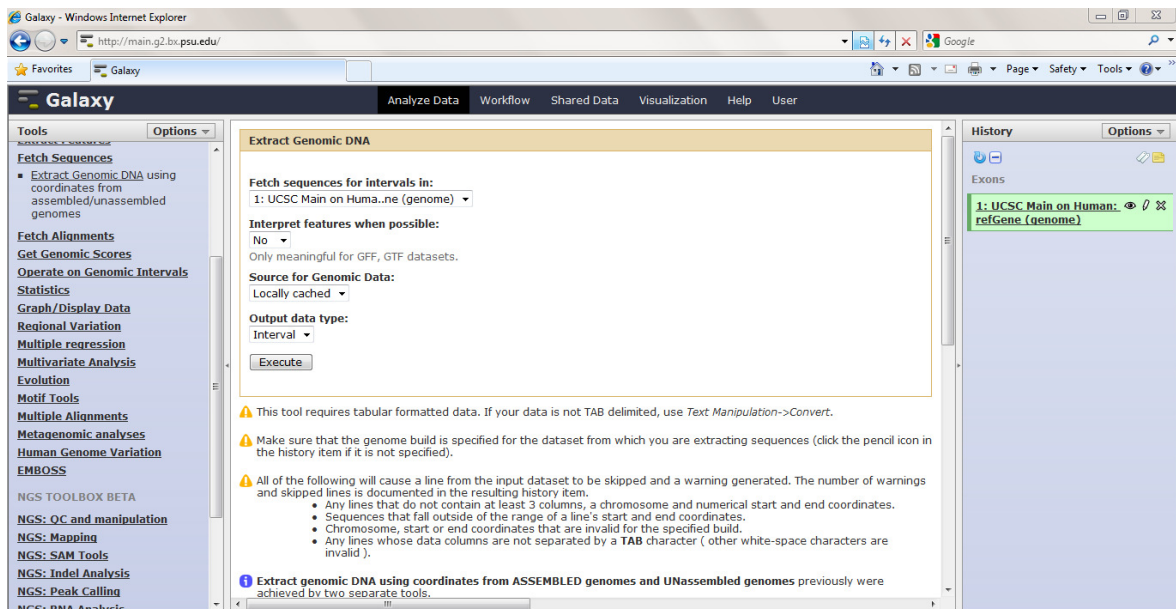


Figure 2.20: The screen-shot of the tool used in Galaxy to retrieve the sequences of the exons with 20 bases at each end.

The sequences of the exons with 20 bases at each end were retrieved from RefSeq by selecting the “fetch sequences” and the “extract genomic DNA using coordinates from assembled/unassembled genomes” options, under the tools section of Galaxy.

The exons+20 file was checked for errors. The error check removed any records with data missing. The error check also removed any records where the exons+20 sequence lengths did not correspond to the sequence coordinates given (**B.SpliceSites.Remove_Errors()** and **Fig.2.18.b**). After the errors were removed, the exons+20 records were checked for redundancy. Redundant records were defined as records where the sequences were on the same chromosome, with the same start and end positions in the chromosome and from the same DNA strand. Only one record for each set of redundant records was retained. The additional redundant records were removed from the dataset (**B.General.Remove_Redundancies()** and **Fig.2.18.c**).

In-order to more quickly identify sSNS in exons, the exons+20 records were separated into different files, according to the chromosomes on which the exon+20 sequences resided (**B.General.Create_Chrom_Files()** and **Fig.2.18.d**). The sSNS in exons were identified as the sSNS whose positions in the chromosomes were within the start and end positions of the exons in the chromosomes, excluding the flanking intron sequences (**B.General.Join_Files()** and **Fig.2.18.e**). After the sSNS in exons were identified, the records were checked to ensure that the reference alleles given by NCBI/dbSNP131 were in the exons. If the reference alleles given were not in the exons, the records were removed from the dataset (**B.General.Check_Alleles()** and **Fig.2.18.f**).

The 5' donor and 3' acceptor splice-sites are named according to the introns. The 5' donor splice-sites are at the 5' start of the introns and the 3' acceptor splice-sites are at the 3' end of the introns. When dealing with exons, it is important to remember that 5' donor splice-sites are at the 3' end of the exons and the 3' acceptor splice-sites are at the 5' start of the exons (**Fig.2.21**).

MaxEntScan was used to calculate ME. According to the requirements of MaxEntScan (http://genes.mit.edu/burgelab/maxent/Xmaxentscan_scoreseq.html), the 3' acceptor splice-sites were defined as 23-mer sequences. Each 23-mer sequence consisted of twenty

bases into upstream intron and three bases into the downstream exon. The 5' donor splice-sites were defined as 9-mer sequences. Each 9-mer sequence consisted of three bases into the upstream exon and six bases into the downstream intron. By definition, sSNPs can only be within the translated regions of the coding sequences. For this reason, only the sSNS at any of the three bases in the exonic part of the 3' splice-sites or at any of the three bases in the exonic part of the 5' splice-sites were assessed for a potential functional impact on splicing (**Fig.2.21**).

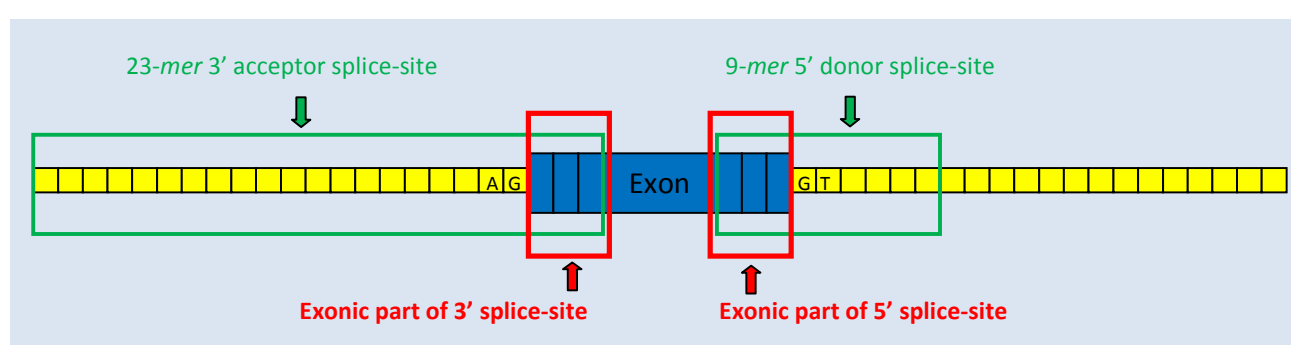


Figure 2.21: Schematic of the exon plus 20 bases at each end sequences retrieved from RefSeq.

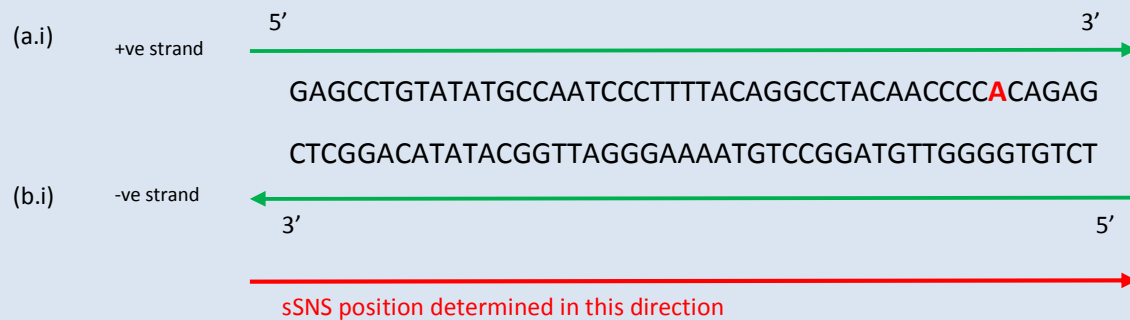
The green boxes show the splice-site sequences sliced from the sequences of the exons plus 20 bases on each end. The red boxes show the exonic part of the splice-sites. Only sSNS within the exonic part of the splice-sites were assessed for a potential functional impact on alternative splicing.

The sSNS within the exonic part of the splice-sites were identified from the sSNS in exon records. The sequences in the sSNS in exon records still contained the exons plus 20 bases into the flanking introns, at each end of the exons. The exon+20 sequences were written in the 5' to 3' direction, regardless of whether the exon sequences were transcribed from the positive or negative DNA strands.

The genomic positions of the sSNPs were determined by NCBI/dbSNP131 according to the reference genome assembly, which is annotated on the positive DNA strand, in the 5' to 3' direction. The genomic positions of the sSNS transcribed from the positive DNA strand were also determined in the 5' to 3' direction, by NCBI/dbSNP131. The positions of the sSNS within the exons were thus determined by subtracting the start position of the exons+20

from the position of the sSNS (**Fig.2.22.a**). As the genomic positions of the sSNS were determined on the positive DNA strand, in the 5' to 3' direction, the genomic positions of sSNS transcribed from the negative DNA strand were actually determined in the 3' to 5' direction by NCBI/dbSNP131. Because the exon sequence were written in the 5' to 3' direction, the positions of the sSNS within exons transcribed from the negative DNA strands had to be determined from the end of the exon sequences. The positions of the sSNS within exons transcribed from the negative DNA strands were calculated by subtracting the position of the sSNS from the end position of the exons+20, minus one. As the first base in a sequence was always counted as zero, when counting backwards from the end of the sequence an additional one always had to be subtracted (**Fig.2.22.b**).

The equations used to determine the positions of the sSNS in the exons were used to identify which of the sSNS were in the exonic part of the splice-sites (**B.SpliceSites.In_Splice_Sites()** and **Fig.2.18.g**). sSNS transcribed from the positive DNA strand and in the exonic part of the 3' splice-sites, were identified by subtracting the start position of the exon+20 from the position of the sSNS (**Fig.2.23.a**). sSNS transcribed from the negative DNA strand and in the exonic part of the 3' splice-sites, were identified by subtracting the position of the sSNS from the end position of the exon+20, minus one (**Fig.2.23.b**). The sSNS in the exonic part of the 3' splice-sites were identified as sSNS at the 20th, 21st or 22nd base, from the start of the exon+20 sequences. These sSNS were at the first, second or third base of the exon. To create the 23-mer reference 3' splice-site sequences, 20 bases from the upstream intron together with three bases from the downstream exon were sliced from the exon+20 sequences (**Fig.2.23.c**). The variant 3' splice-site sequences were created by replacing the reference sSNS allele in the 3' splice-sites, with the variant sSNS alleles (**B.SpliceSites.Acceptor_Splice_Site()** and **Fig.2.18.h**).

(a) Exons as in genome:**(b) Exons as in the data file:**

	SSNS position	Strand	Ref allele	Var allele	Exon start	Exon end	Exon sequence (5' → 3' direction)
(a.ii)	100154631	+	A	T	100154590	100154641	GAGCCTGTATATGCCAATCCCTTTACAGGCCTACAACCCC A CAGAGAGAA
(b.ii)	100154631	-	T	A	100154590	100154641	TTCTCTCT G TGGGGTTGTAGGCCTGTAAAAGGGATTGGCATATACAGGCTC

(a.iii) sSNS on +ve strand: **sSNS position – exon start = sSNS position in exon**

$$100154631 - 100154590 = 41$$

Allele in exon at position 41 = A

(b.iii) sSNS on -ve strand: **exon end – sSNS position - 1 = sSNS position in exon**

$$100154641 - 100154631 - 1 = 9$$

Allele in exon at position 9 = T

Figure 2.22: Examples of the calculations used to determine the positions of the sSNS in the exons.

(a) An exon sequence transcribed from the positive DNA strand

(b) An exon sequence transcribed from the negative DNA strand

(i) The exon sequences as in the human genome

(ii) The exon sequences as in the records retrieved from RefSeq

(iii) The equations used to calculate the positions of the sSNS in the exons

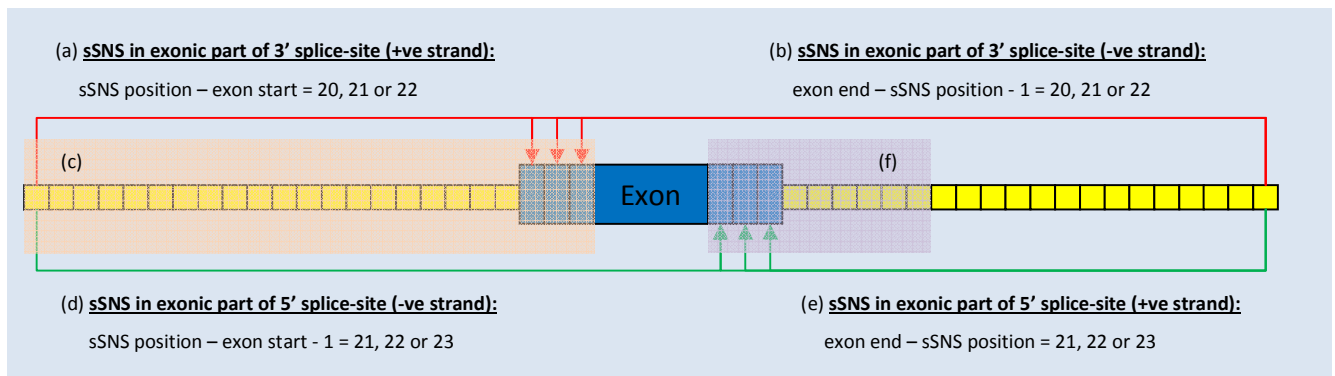


Figure 2.23: Schematic of the calculations used to identify sSNS within the exonic part of splice-sites.

- (a) The equation used to identify sSNS in the 3' splice-sites, transcribed from the positive DNA strand
- (b) The equation used to identify sSNS in the 3' splice-sites, transcribed from the negative DNA strand
- (c) The 3' splice-site motif, sliced from the exon plus 20 bases at each end sequence
- (d) The equation used to identify sSNS in the 5' splice-sites, transcribed from the negative DNA strand
- (e) The equation used to identify sSNS in the 5' splice-sites, transcribed from the positive DNA strand
- (f) The 5' splice-site motif, sliced from the exon plus 20 bases at each end sequence

MaxEntScan was used to calculate the ME of the reference and variant splice-site sequences (**B.SpliceSites.MaxEntScan()** and **Fig.2.18.i**). The score3.pl Perl script of MaxEntScan was used to calculate the ME of the 3' splice-sites. The score5.pl Perl script of MaxEntScan was used to calculate the ME of the 5' splice-sites. The change in the ME caused by a sSNS was calculated by subtracting the reference ME from the variant ME (**B.SpliceSites.Change_In_ME()** and **Fig.2.18.j**). A change in ME less than zero indicated that a sSNS potentially caused a decrease in ME. A change in ME greater than zero indicated that a sSNS potentially caused an increase in ME.

The observed changes in ME caused by sSNS were ordered in increasing order. The 10% most extreme decrease and the 10% most extreme increase in ME were chosen to represent an extreme change in ME (**B.General.Cut_Off_Values()** and **Fig.2.18.k**). This was done separately for the sSNS in the 3' splice-sites and the sSNS in the 5' splice-sites.

The sSNS predicted to potentially cause an extreme change in ME were assessed to determine whether any had previously been associated with a trait or disease in GWAS. The

file containing the sSNS predicted to potentially cause an extreme change in ME and the file containing the sSNPs previously associated with a trait or disease in GWAS (created in Method 2.1) were joined. The two files were joined based on the sSNS positions in the chromosomes. After the two files were joined, the resulting file (GWAS file) contained the sSNS predicted to potentially cause an extreme change in ME, which had previously been associated with a trait or disease in GWAS (**B.General.GWAS_Or_OMIM()** and **Fig.2.18.l**).

The sSNS predicted to potentially cause an extreme change in ME were assessed to determine which occurred within genes known to be associated with disease in OMIM. The file containing the sSNS predicted to potentially cause an extreme change in ME, and the file containing the sSNPs in disease genes (created in Methods 2.1) were joined. The two files were joined based on the sSNS positions in the chromosomes. After the two files were joined, the resulting file (OMIM file) contained the sSNS predicted to potentially cause an extreme change in ME, which occurred within genes known to be associated with disease in OMIM (**B.General.GWAS_Or_OMIM()** and **Fig.2.18.m**).

The sSNS predicted to potentially cause an extreme change in ME were assessed to determine whether any had previously been associated with a trait or disease in GWAS and occurred within disease genes. The GWAS and OMIM files were joined based on the sSNS positions in the chromosomes. After the two files were joined, the resulting file contained the sSNS predicted to potentially cause an extreme change in ME, which had previously been associated with a trait or disease in GWAS and occurred within genes known to be associated with disease in OMIM (**B.General.GWAS_And_OMIM()** and **Fig.2.18.n**).

2.6. Identification of sSNPs with a potential functional impact

The proportion of sSNPs with a potential functional impact was identified from the sSNS predicted to potentially have a functional impact. The files containing the sSNS predicted to have a potential functional impact through an extreme change in CUF, MFE or ME were merged. Each sSNS record contained only one synonymous substitution, from the reference sSNP allele to one of the sSNP's variant alleles.

To identify the sSNPs, each sSNS record of a sSNP was combined. Each record in the merged file, thus, contained a sSNP with all of the alleles of the sSNP. Some of the sSNPs were within more than one of the files that were merged. The sSNPs within more than one file were predicted to potentially have a functional impact through more than one of the three molecular mechanisms investigated. The records of these sSNPs, within more than one file were combined. Each record, thus, contained a sSNP with all of the alleles of the sSNP and all of the molecular mechanisms by which the sSNP was predicted to potentially have a functional impact (**B.Functional.sSNPs()** and Fig.2.24).

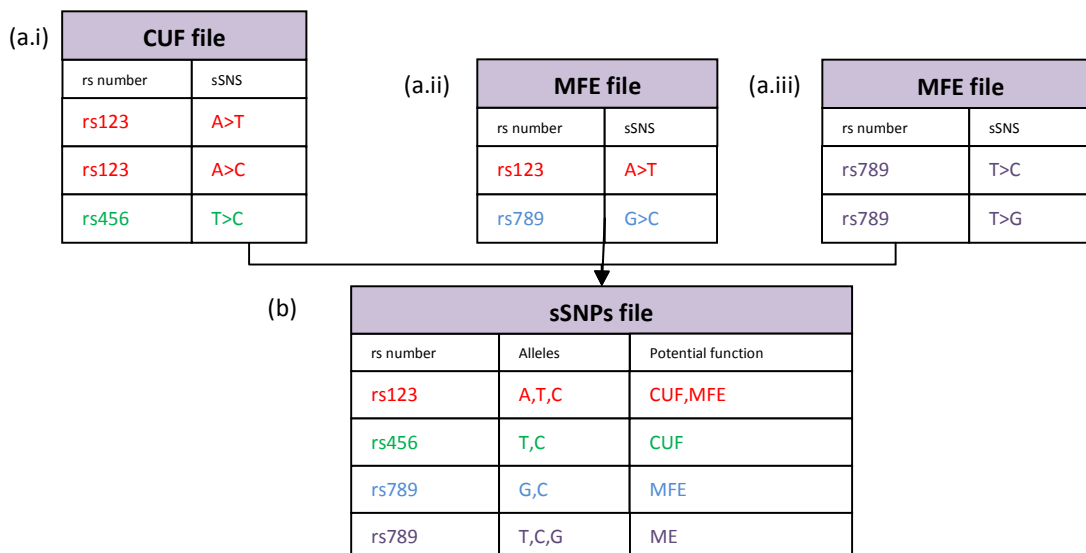


Figure 2.24: Example of the identification of sSNPs predicted to have a potential functional impact through one or more of the three molecular mechanisms investigated.

- (a) Examples of the three files merged, containing sSNS predicted to potentially cause an extreme change in: (i) codon-usage frequency, (ii) the minimum free-energy of local mRNA secondary structures, (iii) the maximum-entropy score of splice-sites
- (b) The merged file, containing the sSNPs predicted to have a potential functional impact, through one or more of the three molecular mechanisms investigated

2.7. sSNPs predicted to have a potential functional impact, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM

The sSNPs predicted to have a potential functional impact were assessed to determine whether any had previously been associated with a trait or disease in GWAS. The file containing the sSNPs predicted to have a potential functional impact (created in Methods 2.7) and the file containing the sSNPs previously associated with a trait or disease in GWAS (created in Method 2.1) were joined. The two files were joined based on the sSNPs positions in the chromosomes. After the two files were joined, the resulting file (GWAS file) contained the sSNPs predicted to have a potential functional impact, which had previously been associated with a trait or disease in GWAS (**B.Disease.Disease()**).

The sSNPs predicted to have a potential functional impact were assessed to determine which occurred within genes known to be associated with disease in OMIM. The file containing the sSNPs predicted to have a potential functional impact (created in Methods 2.7) and the file containing the sSNPs in disease genes (created in Methods 2.1) were joined. The two files were joined based on the sSNPs positions in the chromosomes. After the two files were joined, the resulting file (OMIM file) contained the sSNPs predicted to have a potential functional impact, which occurred within genes known to be associated with disease in OMIM (**B.Disease.Disease()**).

The sSNPs predicted to have a potential functional impact were assessed to determine whether any had previously been associated with a trait or disease in GWAS and occurred within disease genes. These sSNPs were identified by joining the GWAS and OMIM files, based on the sSNPs positions in the chromosomes (**B.Disease.Disease()**).

CHAPTER 3: RESULTS

3.1. Identification of sSNPs

sSNPs were identified from the cSNPs retrieved from dbSNP131 using Galaxy. Approximately 26 million SNPs and 400,000 cSNPs were retrieved. The SNP and cSNP files were joined to produce a dataset consisting of 392,046 cSNPs (**Fig.3.1.a**). This joined cSNP file contained records with errors and redundant records.

The joined cSNP file was checked for errors and 35,123 (9.0%) records with errors removed (**Fig.3.1.b**). The removed records included 1,402 INDELS incorrectly annotated as SNPs in dbSNP131. In addition, 1,134 records where the reference allele given was more than one nucleotide in length or the codons given were not three nucleotides in length. The removed records also included 19 records where the reference allele was not in the reference codon. The error check also removed errors that occurred during the file joining process, including 32,568 records where the SNP coordinates from the SNP file did not correspond with the SNP coordinates from the cSNP file. Once the records with errors were removed, the dataset consisted of 356,923 true cSNPs (**Fig.3.1.c**).

The true cSNP dataset was checked for redundancies. Of the true cSNP records, 132,032 (37.0%) redundant records were removed (**Fig.3.1.d**) to produce a dataset of 224,891 non-redundant true cSNPs (**Fig.3.1.e**). The main cause of redundancy was because dbSNP sometimes assigned more than one rs number to a single SNP. A SNP that has been assigned more than one rs number will have multiple records in dbSNP, each with a different rs number, but with the same SNP coordinate. The redundant SNP records may be in both the SNP file and cSNP file retrieved from dbSNP. The SNP file is joined to the cSNP file based on the coordinates of the SNPs, rather than the rs numbers. Each of the redundant records for a single SNP in the SNP file will, therefore, be joined to each redundant record for the same

SNP in the cSNP file to produce even more redundant records. There is redundancy in dbSNP; however, it is the joining of the two SNP files retrieved from dbSNP that further escalates the problem of redundancy (**Fig.3.2**).

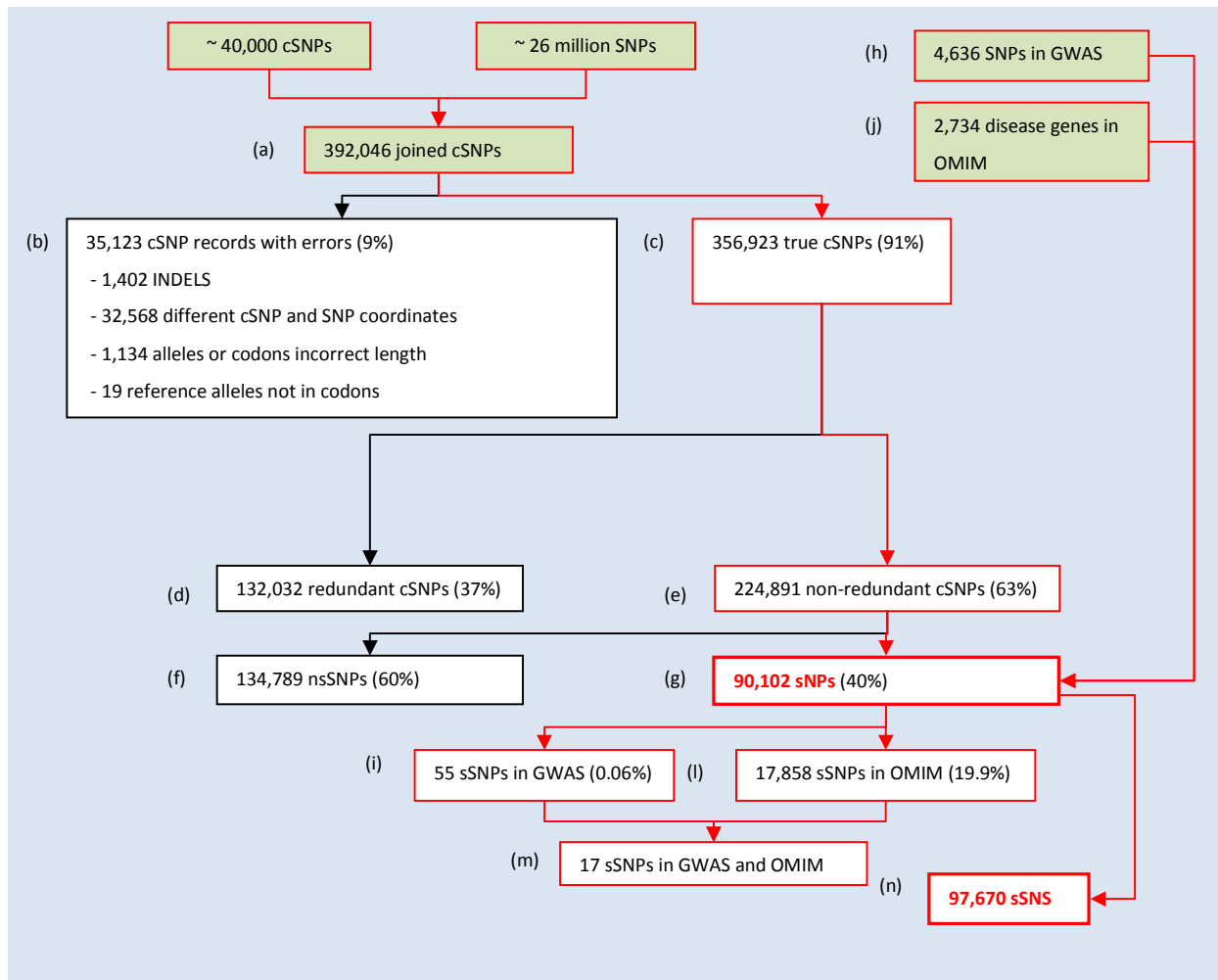


Figure 3.1: Flow-diagram for the identification of sSNPs, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM.

□ = records that were discarded from the datasets.

□ = records that were retained in the datasets and used as input for the next analysis step.

cSNP = coding SNP

nsSNP = non-synonymous SNP

sSNP = synonymous SNP

sSNS = synonymous single nucleotide substitution

in GWAS = sSNS previously associated with a trait or disease in GWAS

in OMIM = sSNS within genes known to be associated with disease in OMIM

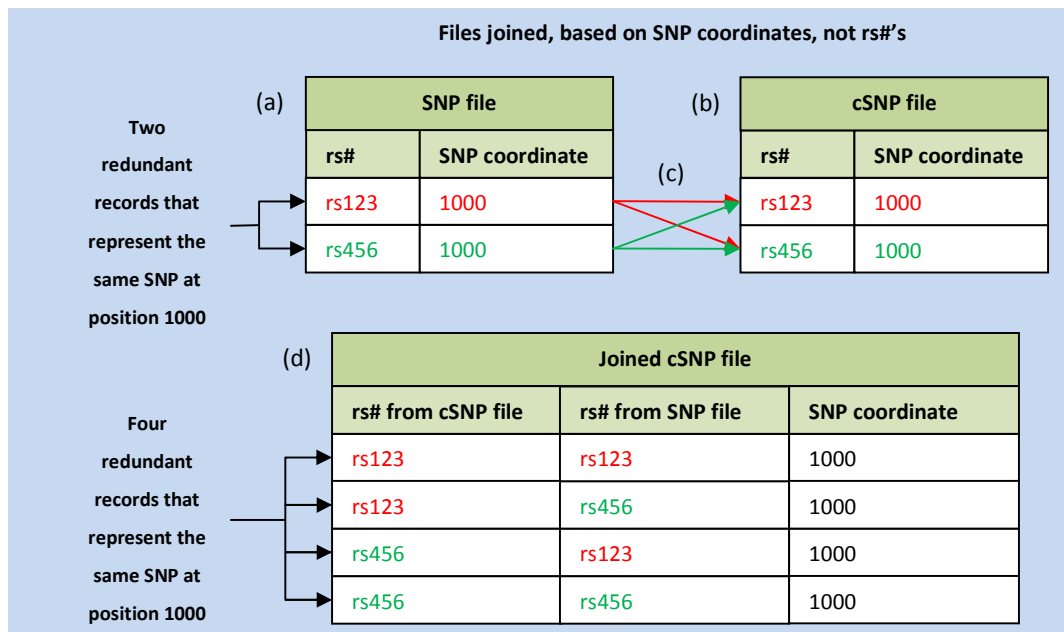


Figure 3.2: Schematic of the main cause of redundancy in the cSNP file retrieved from dbSNP131, using Galaxy.

- (a) The SNP file contains a SNP that has been assigned more than one rs number
- (b) The same redundant SNP, as in the SNP file, occurs in the cSNP file
- (c) Each of the redundant records are joined, during the file joining process
- (d) The redundant records in the final joined file

sSNPs were identified from the non-redundant true cSNPs as those that do not change the amino acid encoded. Of the non-redundant true cSNPs, 134,789 (60.0%) were predicted to be nsSNPs (**Fig.3.1.f**). A subset of 90,102 (40.0%) of the cSNPs were predicted to be sSNPs (**Fig.3.1.g**). The identified sSNPs were assessed to determine whether any had previously been associated with a trait or disease in GWAS and/or occurred within genes known to be associated with disease in OMIM. The GWAS catalogue contained 4,636 SNPs associated with a trait or disease (**Fig.3.1.h**). Of the identified sSNPs, 55 (0.06%) had previously been associated with a trait or disease in GWAS (**Fig.3.1.i**). At the time of this study, OMIM contained 2,734 genes known to be associated with disease (**Fig.3.1.j**). A subset of 17,858 (19.9%) of the identified sSNPs were within genes known to be associated with disease (**Fig.3.1.k**). Only 17 (0.02%) of the sSNPs had previously been associated with a trait or disease in GWAS and occurred within genes known to be associated with disease in OMIM (**Fig.3.1.l**). The majority of these sSNPs were associated with complex multifactorial traits (**Table.3.1**).

Table 3.1: The 17 sSNPs that have previously been associated with a trait or disease in GWAS and occur within genes known to be associated with disease in OMIM

Chromosome	sSNS position	Trait or disease in GWAS → gene: trait: risk-allele, p-value	Gene associated with disease in OMIM → gene: disease (OMIM accession number)
chr1	17674536	<i>PADI4</i> : Rheumatoid arthritis: rs2240335-?, 2.00E-08	<i>PADI4</i> , <i>PADI5</i> , <i>PAD</i> : Rheumatoid arthritis, susceptibility to (605347)
chr2	21232194	<i>APOB</i> : LDL cholesterol, Triglycerides: rs693-G, 4.00E-17	<i>APOB</i> , <i>FLDB</i> , <i>LDLCQ4</i> : Hypercholesterolemia, due to ligand-defective apo B; Hypobetalipoproteinemia; Hypobetalipoproteinemia, normotriglyceridemic (107730)
chr3	133475721	<i>TF</i> : Serum markers of iron status: rs1799852-?, 5.00E-06	<i>TF</i> : Atransferrinemia; Iron deficiency anemia, susceptibility to (190000)
chr3	172165726	<i>GHSR</i> : Height: rs572169-T, 3.00E-18	<i>GHSR</i> : Short stature (601898)
chr4	6303021	<i>WFS1</i> : Type 2 diabetes: rs1801214-T, 3.00E-08	<i>WFS</i> , <i>WFRS</i> , <i>WFS1</i> , <i>DFNA6</i> : Hearing loss, low-frequency sensorineural; Wolfram syndrome; Wolfram-like syndrome, autosomal dominant; Diabetes mellitus, noninsulin-dependent, association with (606201)
chr4	15482359	<i>KIAA1345</i> : Conduct disorder (case status): rs1861050-?, 8.00E-06	<i>MKS6</i> , <i>CC2D2A</i> , <i>KIAA1345</i> : Joubert syndrome 9; Meckel syndrome, type 6 (612013)
chr4	155488820	<i>FGB</i> , <i>FGA</i> , <i>FGG</i> : Fibrinogen: rs6056-A, 8.00E-39	<i>FGB</i> : Afibrinogenemia, congenital; Dysfibrinogenemia, beta type; Thrombophilia, dysfibrinogenemic (134830)
chr6	32363843	<i>BTNL2</i> : Ulcerative colitis: rs9268480-G, 3.00E-06	<i>BTNL2</i> : Sarcoidosis, susceptibility to (606000)
chr6	32363954	<i>BTNL2</i> : Waist-hip ratio: rs2076529-C, 4.00E-07	<i>SS2</i> : Sarcoidosis, susceptibility to (606000)
chr10	90771828	<i>ACTA2</i> , <i>FAS</i> : Immunoglobulin A: rs2234978-A, 6.00E-06	<i>TNFRSF6</i> , <i>APT</i> , <i>FAS</i> , <i>CD95</i> , <i>ALP</i> , <i>S1A</i> : Autoimmune lymphoproliferative syndrome, type IA; Squamous cell carcinoma, burn scar-related, somatic; Autoimmune lymphoproliferative syndrome (134637)
chr13	50080846	<i>PHF11</i> : Cardiac hypertrophy: rs2031532-?, 5.00E-06	<i>PHF11</i> , <i>NYREN34</i> : IgE levels QTL; Asthma (607796)
chr15	78894338	<i>CHRNA3</i> , <i>CHRNA5</i> , NR: Smoking behavior, Lung adenocarcinoma, Nicotine dependence: rs1051730-G, 3.00E-73	<i>CHRNA3</i> , <i>LNCR2</i> , <i>PAOD2</i> : Lung cancer susceptibility (118503)
chr17	61566030	<i>ACE</i> : Angiotensin-converting enzyme activity: rs4343-G, 3.00E-25	<i>ACE</i> , <i>DCP1</i> , <i>ACE1</i> , <i>MVCD3</i> : Renal tubular dysgenesis; Angiotensin I-converting enzyme, benign serum increase; Alzheimer disease, susceptibility to; Microvascular complications of diabetes; Myocardial infarction, susceptibility to; SARS, progression of (106180)
chr19	11210911	<i>LDLR</i> : LDL cholesterol: rs2228671-G, 4.00E-14	<i>LDLR</i> , <i>FHC</i> , <i>FH</i> , <i>LDLCQ2</i> : Hypercholesterolemia, familial (606945)
chr19	49206416	<i>FUT2</i> : Plasma level of vitamin B12: rs492602-G, 5.00E-17	<i>FUT2</i> , <i>SE</i> , <i>B12QTL1</i> : Bombay phenotype; Norwalk virus infection, resistance to; Vitamin B12 plasma level QTL1 (182100)
chr21	37833306	<i>CLDN14</i> : Kidney stones: rs219780-C, 4.00E-12	<i>CLDN14</i> , <i>DFNB29</i> : Deafness, autosomal recessive (605608)
chr22	37469590	<i>TMPRSS6</i> : Serum iron concentration, Red blood cell traits, Hematology traits, Serum marker iron status: rs4820268-G, 4.00E-11	<i>TMPRSS6</i> , <i>IRIDA</i> : Iron-refractory iron deficiency anemia (609862)

To determine the potential functional impact of each sSNP, the sSNS from the reference allele to each of the variant alleles needed to be compared. A number of the sSNPs had more than two alleles. It was, therefore, necessary for a separate record to be created for each of the alleles at the sSNPs (**Table.2.3**). Each of the sSNPs with two alleles represented one sSNS from the reference allele to the variant allele. Of the 90,102 identified sSNPs, 84,751 (94.0%) had two alleles. The sSNPs with three alleles each represented two sSNS, one sSNS from the reference allele to the first variant allele and another sSNS from the reference

allele to the second variant allele. The 3,134 (3.5%) sSNPs with two alleles, therefore, produced 6,268 sSNS. The sSNPs with four alleles each represented three sSNS, one for each of the sSNS from the reference allele to each of the three variant alleles. The 2,217 (2.5%) sSNPs each with four alleles, therefore, produced 6,651 sSNS. From the 90,102 sSNP records, 97,670 sSNS records were created (**Fig.3.1.m** and **Table.3.2**).

Table 3.2: The frequency of sSNPs with two or more alleles and the number of sSNS identified

	Number of sSNPs	Number of sSNS
sSNPs	90,102	
sSNPs with 2 alleles	84,751	84,751
sSNPs with 3 alleles	3,134	6,268
sSNPs with 4 alleles	2,217	6,651
sSNS		97,670

Due to the degeneracy of the genetic code, a single amino acid can be encoded by more than one codon. According to the wobble hypothesis, the base pairing rule is not followed as closely at the third position of the codon. A majority of SNPs that alter the nucleotide at the third position of the codon will therefore still encode the same amino acid. As a result, most of the sSNPs will be at the third position of the codon and a few at the first position of the codon. Any SNP that alters the second nucleotide of the codon will always result in a non-synonymous change. As observed in the sSNS dataset, 95% of the sSNS were at the third position of the codon and 5% were at the first position of the codon. No sSNS resulted from changes in the second position of the codon (**Table.3.3**).

Table 3.3: The frequency of sSNS at the first, second and third positions of codons

Positions of sSNS in codons	Number of sSNS	Percentage of sSNS (%)
1	4,841	4.96
2	0	0.00
3	92,829	95.04
Total	97,670	100.00

The two most common sSNS were cytosine to thymine (C>T) transitions (31.7%) and guanine to adenine (G>A) transitions (21.5%), most likely due to cytosine deamination. The next two most common sSNS were thymine to cytosine (T>C) transition (13.0%) and adenine to guanine (A>G) transitions (10.5%), most likely due to cytosine deamination that has occurred on the opposite strand of the sSNS. Of the 97,670 sSNS, 74,932 (76.7%) were transition substitutions and 22,738 (23.3%) were transversion substitutions. The two least frequent sSNS were thymine to adenine (T>A) transversions (91.6%) and adenine to thymine (A>T) transversions (1.5%) (**Table.3.4**).

Table 3.4: The frequency of sSNS changes

sSNS	Number of sSNS	Percentage of sSNS (%)
C>T	30,959	31.70
G>A	20,986	21.49
T>C	12,736	13.04
A>G	10,251	10.50
C>A	4,527	4.63
C>G	4,000	4.10
G>C	3,988	4.08
G>T	2,887	2.96
A>C	2,477	2.54
T>G	1,848	1.89
T>A	1,524	1.56
A>T	1,487	1.51
Total	97,670	100.00

The most common codons affected by a sSNS, encoded the leucine amino acid (12.1%). This is because leucine is encoded by six different codons (TTA, TTG, CTT, CTC, CTA and CTG). A SNP that alters the first or third nucleotide of the leucine codon is, therefore, likely to result in a synonymous change. Cysteine was the least frequently observed synonymous amino acid. Since cysteine is only encoded by two codons (TGT and TGC), the third nucleotide of the cysteine codon can only either be a thymine or cytosine. SNPs in codons that encode methionine (ATG) or tryptophan (TGG) were not included, as methionine and tryptophan are each encoded by one codon. Any nucleotide substitution within a methionine or tryptophan

codon would result in a non-synonymous or missense change. The three stop codons (TAA, TAG and TGA) were not included, as they do not encode an amino acid (**Table.3.5**).

Table 3.5: The frequency of amino acids involved in sSNS

Synonymous amino acids	Number of codons	Number of sSNS	Percentage of sSNS (%)
Leucine	6	11,797	12.08
Proline	4	10,448	10.7
Alanine	4	10,418	10.67
Threonine	4	9,211	9.43
Serine	6	9,112	9.33
Glycine	4	6,968	7.13
Arginine	6	5,209	5.33
Valine	4	4,810	4.92
Aspartic acid	2	4,253	4.35
Glutamic acid	2	3,572	3.66
Asparagine	2	3,297	3.38
Isoleucine	3	3,169	3.24
Glutamine	2	3,141	3.22
Histidine	2	2,824	2.89
Lysine	2	2,783	2.85
Tyrosine	2	2,760	2.83
Phenylalanine	2	2,311	2.37
Cysteine	2	1,587	1.62
Methionine	1	0	0.00
Tryptophan	1	0	0.00
Stop codons	3	0	0.00
Total	64	97,670	100.00

The sSNS were separated according to the chromosomes on which they resided, in-order to join the sSNS records to other records more efficiently. Chromosome 1 contained the most sSNS (10%), as would be expected since it is the largest chromosome. The Y chromosome, which is the second smallest chromosome, contained the least sSNS (0.2%). Only 0.18% of the sSNS were from unknown chromosomes (**Table.3.6**).

Table 3.6: The frequency of sSNS on the different chromosomes¹

Chromosome	Approximate number of mapped genes () = not placed	Approximate number of mapped CDS () = not placed	Number of sSNS	Percentage of sSNS (%)
chr1	3,511 (6)	4,058 (17)	9,760	9.99
chr19	2,066 (10)	2,547 (15)	7,244	7.42
chr6	2,057	2,056	7,137	7.31
chr2	2,368	2,502	6,451	6.60
chr11	2,168	2,351	5,588	5.72
chr17	1,773 (8)	2,241 (8)	5,502	5.63
chr3	1,926	2,221	4,712	4.82
chr9	1,534 (6)	1,545 (2)	4,680	4.79
chr12	1,714	2,016	4,545	4.65
chr7	1,882 (5)	1,977 (5)	4,488	4.60
chr16	1,326	1,589	4,451	4.56
chr5	1,633	1,728	4,190	4.29
chr10	1,391	1,646	4,081	4.18
chr4	1,444 (3)	1,450 (2)	3,605	3.69
chr15	1,249	1,337	3,236	3.31
chr8	1,315 (2)	1,437 (2)	3,230	3.31
chr20	891	1,111	2,868	2.94
chr14	1,532	1,314	2,829	2.90
chr22	855	931	2,454	2.51
chrX	1,672	1,809	2,087	2.14
chr13	720	656	1,681	1.72
chr18	557	598	1,409	1.44
chr21	450	524	1,068	1.09
chrY	429	191	198	0.20
chrUn	(87)	(46)	176	0.18
Total	36,463 (127)	39,835 (97)	97,670	100.00

¹ The approximate number of genes and coding sequences (CDS) mapped to the human genome were retrieved from the “object count” of the NCBI statistics for the *Homo sapiens* genome build 37 version 3 on the 3rd of November 2011 (<http://www.ncbi.nlm.nih.gov/mapview/stats/BuildStats.cgi?taxid=9606&build=37&ver=3>)

3.2. sSNS predicted to potentially cause an extreme change in codon-usage frequency

The sSNS were assessed to determine whether any could potentially cause an extreme change in codon-usage frequency (CUF). The change in CUF caused by a sSNS was calculated by subtracting the reference CUF from the variant CUF. A change in CUF less than zero indicated that a sSNS changed a more frequently used codon to a less frequently used codon (decrease in CUF). A change in CUF greater than zero indicated that a sSNS changed less frequently used codon to a more frequently used codon (increase in CUF). Of the 97,670 identified sSNS, 58,015 (60.0%) caused a codon change that resulted in a codon with reduced usage frequency. A subset of 39,655 (40.0%) of the sSNS caused a codon change that resulted in a codon with increased usage frequency (**Fig.3.3.a**).

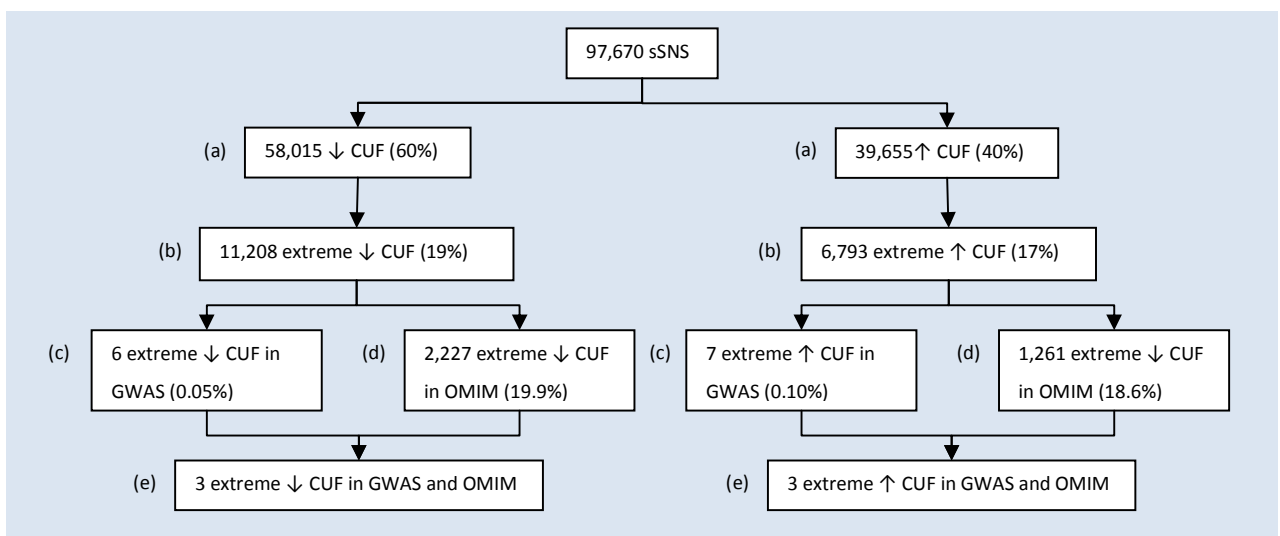


Figure 3.3: Flow-diagram for the identification of sSNS that potentially cause an extreme change in codon-usage frequency, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM.

- (a) sSNS predicted to cause a change in codon-usage frequency
- (b) sSNS that potentially cause an extreme change in codon-usage frequency
- (c) sSNS that potentially cause an extreme change in codon-usage frequency, which have previously been associated with a trait or disease in GWAS
- (d) sSNS that potentially cause an extreme change in codon-usage frequency, which occur within genes known to be associated with disease in OMIM
- (e) sSNS that potentially cause an extreme change in codon-usage frequency, which have previously been associated with a trait or disease in GWAS and occur within genes known to be associated with disease in OMIM

All of the observed changes in CUF were ordered in increasing order. The 10% most extreme decrease and the 10% most extreme increase in CUF were chosen to represent an extreme change in CUF. The changes in CUF caused by sSNS ranged from -47.0 to 47.0. A change in CUF less than or equal to -19.2056 was chosen to represent an extreme decrease in CUF. A change in CUF greater or equal to 19.2056 was chosen to represent an extreme increase in CUF (**Fig.3.4**). Based on the chosen cut-off values, 18,001 (18.4%) of the sSNS were predicted to potentially cause an extreme change in CUF. Of these sSNS, 11,208 were predicted to potentially cause an extreme decrease in CUF. A subset of 6,793 of the sSNS was predicted to potentially cause an extreme increase in CUF (**Fig.3.3b**).

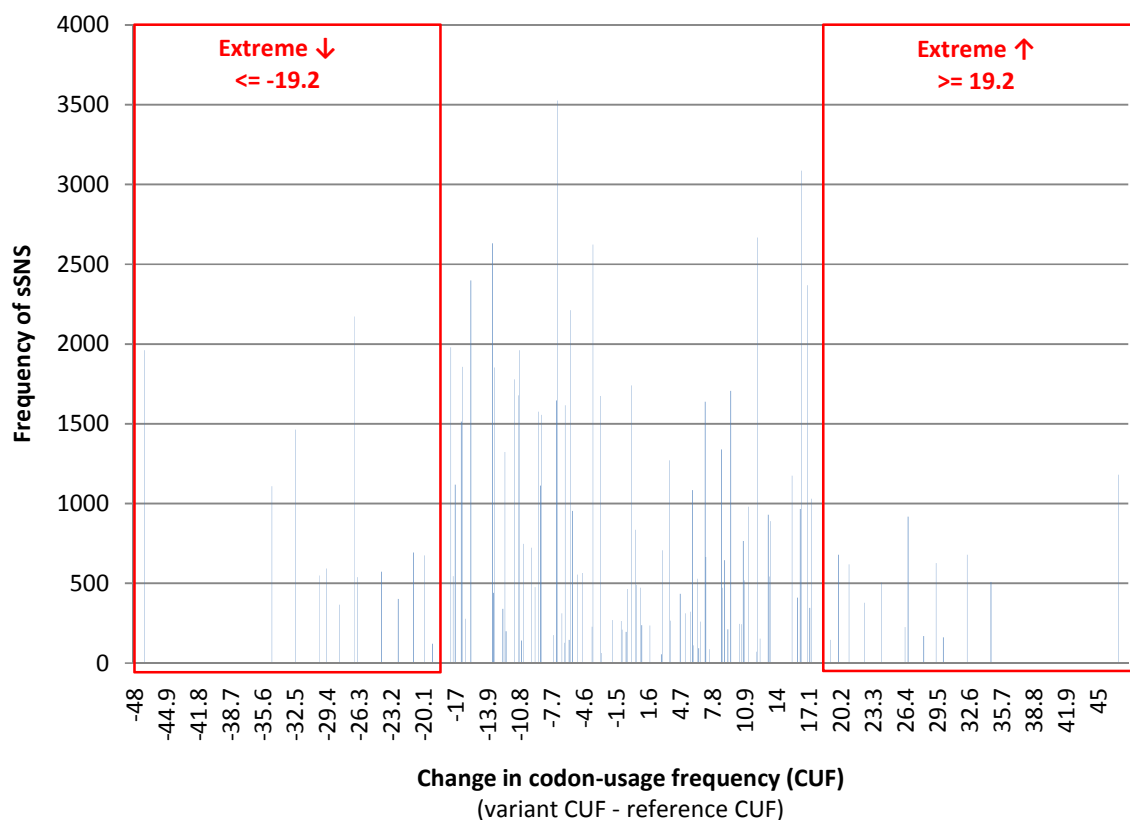


Figure 3.4: Frequency distribution graph of all the changes in codon-usage frequency caused by sSNS.

The changes in CUF to the left of zero (negative values) on the graph, represent synonymous changes from a more frequently used codon to a less frequently used codon (decrease in CUF). The changes in CUF to the right of zero (positive values) on the graph, represent synonymous changes from a less frequently used codon to a more frequently used codon (increase in CUF). The two red squares show the 10% most extreme decrease and 10% most extreme increase in CUF caused by sSNS.

The second most common sSNS were guanine to adenine (G>A) transitions (**Table.3.4**). Guanine to adenine transitions were, however, the most common sSNS predicted to potentially cause an extreme decrease in CUF (40.4%) (**Table.D.1**). No thymine to cytosine (T>C) or adenine to guanine (A>G) transitions were predicted to potentially result in an extreme decrease in CUF, despite that they were the third (13.0%) and fourth (10.5%) most frequent sSNS. No adenine to cytosine (A>C), thymine to guanine (T>G) or adenine to thymine (A>T) transversions were predicted to potentially cause an extreme decrease in CUF (**Fig.3.5**). This is unsurprising as transversion substitution occur less frequently than transition substitutions.

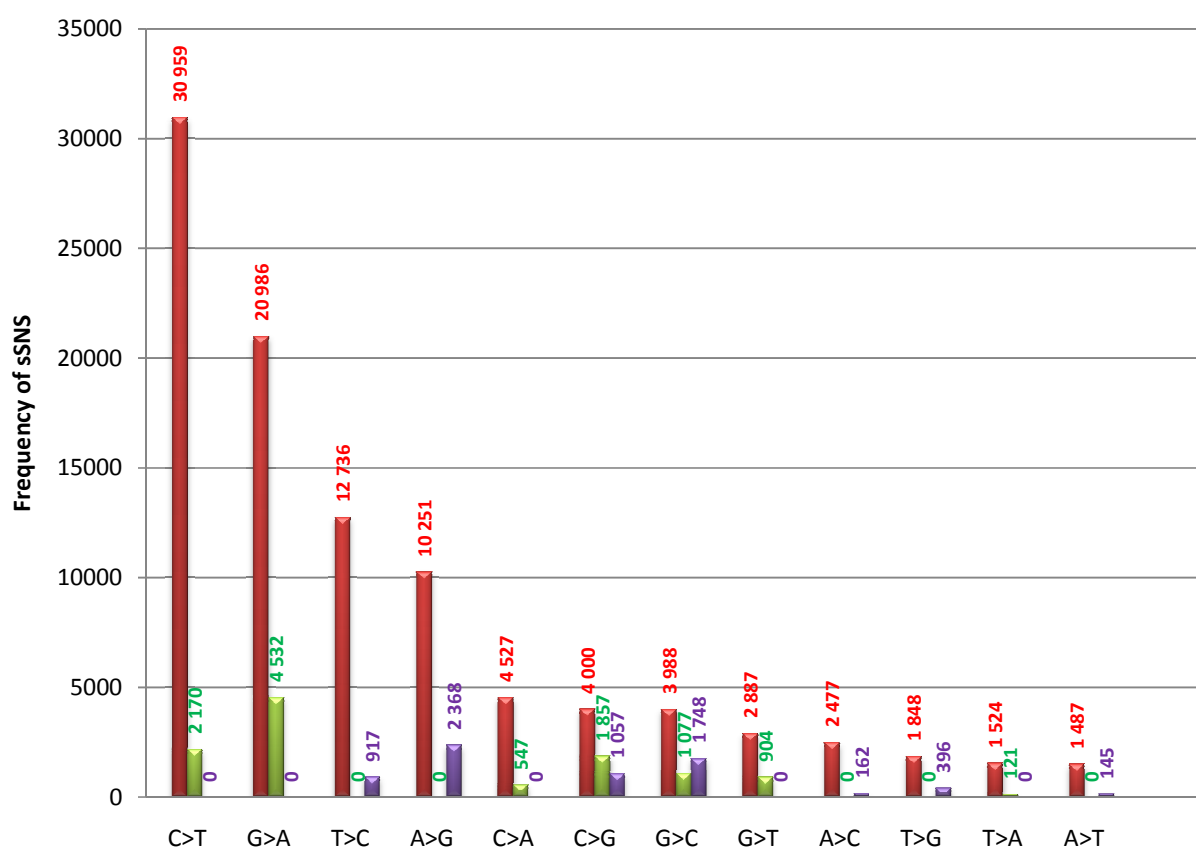


Figure 3.5: The frequency of sSNS predicted to potentially cause an extreme change in codon-usage frequency.

- = sSNS
- = sSNS predicted to potentially cause an extreme decrease in codon-usage frequency
- = sSNS predicted to potentially cause an extreme increase in codon-usage frequency

Adenine to guanine (A>G) transition were the most common sSNS predicted to potentially cause an extreme increase in CUF (34.9%) (**Table.D.2**). None of the most commonly occurring cytosine to thymine (C>T) or guanine to adenine (G>A) transitions were predicted to potentially cause an extreme increase in CUF. No cytosine to adenine (C>A), guanine to thymine (G>T) or thymine to adenine (T>A) transversions were predicted to potentially result in an extreme increase in CUF (**Fig.3.5**). The reason for this may be that these transversion substitutions are only a minor proportion of the sSNS identified.

The most common extreme changes from a frequently used codon to a less frequently used codon were CTG to TTG synonymous leucine changes (19.4%). The second most common extreme decrease in CUF was for CAG to CAA synonymous glutamine changes (17.5%). This synonymous glutamine codon change results in a change in CUF of -47, which is the most extreme decrease in CUF observed. ATT to ATA isoleucine changes were the least frequent synonymous codon changes predicted to potentially result in an extreme decrease in CUF (1.1%). This synonymous isoleucine change results in a change in CUF of -19.2, which is the cut-off value chosen to represent an extreme decrease in CUF (**Table.3.7**).

Table 3.7: The frequency of predicted extreme codon changes from a more frequently used codon to a less frequently used codon (decrease in codon-usage frequency)

Codon change	Codon-usage frequency change	Synonymous amino acid	Number of sSNS	Percentage of sSNS (%)
CTG->TTG	39.57 -> 12.91 = -26.66	Leucine	2,170	19.36
CAG->CAA	73.50 -> 26.50 = -47.00	Glutamine	1,960	17.49
CTG->CTA	39.57 -> 7.14 = -32.43	Leucine	1,462	13.04
GTG->GTA	46.33 -> 11.67 = -34.67	Valine	1,110	9.9
CCC->CCG	32.35 -> 11.31 = -21.04	Proline	694	6.19
CTG->CTC	39.57 -> 19.56 = -20.02	Leucine	675	6.02
GCC->GCG	39.97 -> 10.62 = -29.35	Alanine	591	5.27
ACC->ACG	35.53 -> 11.38 = -24.15	Serine	572	5.1
ATC->ATA	46.99 -> 16.9 = -30.08	Isoleucine	547	4.88
CTG->CTT	39.57 -> 13.17 = -26.41	Leucine	539	4.81
GTG->GTC	46.33 -> 23.83 = -22.51	Valine	402	3.59
GTG->GTT	46.33 -> 18.17 = -28.16	Valine	365	3.27
ATT->ATA	36.11 -> 16.90 = -19.21	Isoleucine	121	1.08
Total			11,208	100.00

The most common extreme changes from a less frequently used codon to a more frequently used codon were CAA to CAG synonymous glutamine changes (17.4%). These synonymous glutamine codon changes result in a change in CUF of 47, which is the most extreme increase in CUF caused by sSNS. ATT to ATT isoleucine changes were the least frequent synonymous codon changes predicted to potentially cause an extreme increase in CUF (1.1%). This synonymous isoleucine change results in a change in CUF of 19.2, which is the cut-off value chosen to represent an extreme increase in CUF (**Table.3.8**).

Table 3.8: The frequency of predicted extreme codon changes from a less frequently used codon to a more frequently used codon (increase in codon-usage frequency)

Codon change	Codon-usage frequency change	Synonymous amino acid	Number of sSNS	Percentage of sSNS (%)
CAA->CAG	26.50 -> 73.50 = 47.00	Glutamine	1,181	17.39
TTG->CTG	12.91 -> 39.57 = 26.66	Leucine	917	13.6
CTA->CTG	7.14 -> 39.57 = 32.43	Leucine	680	10.01
CTC->CTG	19.56 -> 39.57 = 20.02	Leucine	679	10
GCG->GCC	10.62 -> 39.97 = 29.35	Alanine	626	9.22
CCG->CCC	11.31 -> 32.35 = 21.04	Proline	619	9.11
GTA->GTG	11.67 -> 46.33 = 34.67	Valine	507	7.46
ACG->ACC	11.38 -> 35.53 = 24.15	Threonine	503	7.4
GTC->GTG	23.83 -> 46.33 = 22.51	Valine	378	5.56
CTT->CTG	13.17 -> 39.57 = 26.41	Leucine	227	3.34
GTT->GTG	18.17 -> 46.33 = 28.16	Valine	169	2.49
ATA->ATC	16.90 -> 46.99 = 30.08	Isoleucine	162	2.38
ATA->ATT	16.90 -> 36.11 = 19.21	Isoleucine	145	2.13
Total			6,793	100.00

Of the 11,208 sSNS predicted to potentially cause an extreme decrease in CUF, six (0.05%) had previously been associated with a trait or disease in GWAS (**Fig.3.3.c**). A subset of 2,227 (19.9%) of the sSNS predicted to potentially cause an extreme decrease in CUF were within genes known to be associated with disease in OMIM (**Fig.3.3.d**). Only three (0.03%) of the sSNS predicted to potentially cause an extreme decrease in CUF had previously been associated with a trait or disease in GWAS and occurred within genes known to be associated with disease in OMIM (**Fig.3.3.e and Table.3.9**).

Table 3.9: The three sSNS predicted to potentially cause an extreme decrease in codon-usage frequency, which have previously been associated with a trait or disease in GWAS and occur within genes known to be associated with disease in OMIM

Chromosome	sSNS position	Trait or disease in GWAS → Gene: trait: risk-allele, p-value	Gene associated with disease in OMIM → Gene : disease (OMIM accession number)
chr3	133475721	TF: Serum markers of iron status: rs1799852-?, 5.00E-06	TF: Atransferrinemia; Iron deficiency anemia, susceptibility to (190000)
chr6	32363843	BTNL2: Ulcerative colitis: rs9268480-G, 3.00E-06	BTNL2: Sarcoidosis, susceptibility to (606000)
chr21	37833306	CLDN14: Kidney stones: rs219780-C, 4.00E-12	CLDN14, DFNB29: Deafness, autosomal recessive (605608)

Of the 6,793 sSNS predicted to potentially cause an extreme increase in CUF, seven (0.1%) had previously been associated with a trait or disease in GWAS (**Fig.3.3.c**). A subset of 1,261 (18.6%) of the sSNS predicted to potentially cause an extreme increase in CUF were within genes known to be associated with disease (**Fig.3.3.d**). Only three (0.04%) of the sSNS predicted to potentially cause an extreme increase in CUF had previously been associated with a trait or disease in GWAS and occurred within genes known to be associated with disease in OMIM (**Fig.3.3.e and Table.3.10**). The sSNS at position 32363843 on chromosome six, which was predicted to potentially cause an extreme decrease in CUF and associated with ulcerative colitis, is 111 bases away from the sSNS at position 32363954, which was predicted to potentially cause an extreme increase in CUF and associated with waist-hip ratio.

Table 3.10: The three sSNS predicted to potentially cause an extreme increase in codon-usage frequency, which have previously been associated with a trait or disease in GWAS and occur within genes known to be associated with disease in OMIM

Chromosome	sSNS position	Trait or disease in GWAS → Gene: trait: risk-allele, p-value	Gene associated with disease in OMIM → Gene : disease (OMIM accession number)
chr2	21232194	APOB: LDL cholesterol, Triglycerides: rs693-G, 4.00E-17	APOB, FLDB, LDLQC4: Hypercholesterolemia, due to ligand-defective apo B; Hypobetalipoproteinemia; Hypobetalipoproteinemia, normotriglyceridemic (107730)
chr6	32363954	BTNL2: Waist-hip ratio: rs2076529-C, 4.00E-07	BTNL2, SS2: Sarcoidosis, susceptibility to (606000)
chr13	50080846	PHF11: Cardiac hypertrophy: rs2031532-?, 5.00E-06	PHF11, NYREN34: IgE levels QTL; Asthma (607796)

3.3. sSNS predicted to potentially cause an extreme change in the minimum free-energy of local mRNA secondary structures

The sSNS were assessed to determine whether any could potentially cause an extreme change in the minimum free-energy of local mRNA secondary structures (MFE). An extreme change in MFE could indicate that a sSNS changes the stability of the local mRNA secondary structure. Galaxy was used to retrieve 49,083 CWI from RefSeq (**Fig. 3.6.a**). The CWI contained the coding exons as well as the introns.

The CWI were checked for errors and redundancies. A subset of 8,223 (16.8%) of the CWI records with errors were removed. The removed records included one record with data missing and 8,222 records where the length of the CWI sequence did not correspond to the CWI coordinates given (**Fig.3.6.b**). Once the records with errors were removed, the dataset consisted of 40,865 CWI records (**Fig.3.6.c**). The CWI dataset was checked for redundancies. Of the CWI records, 15,997 (39.1%) redundant records were removed (**Fig.3.6.d**) to produce a dataset of 24,868 non-redundant records (**Fig.3.6.e**).

The CWI records were separated according to the chromosomes on which the CWI resided. The separation of the CWI records according to chromosome allowed the sSNS and CWI records to be joined more quickly. Chromosome 1, which is the largest chromosome, contained the most CWI (10.1%). The Y chromosome, which is the second smallest chromosome, contained the least CWI (0.6%). Only 17 (0.07%) of the CWI were from unknown chromosomes (**Table.3.11**).

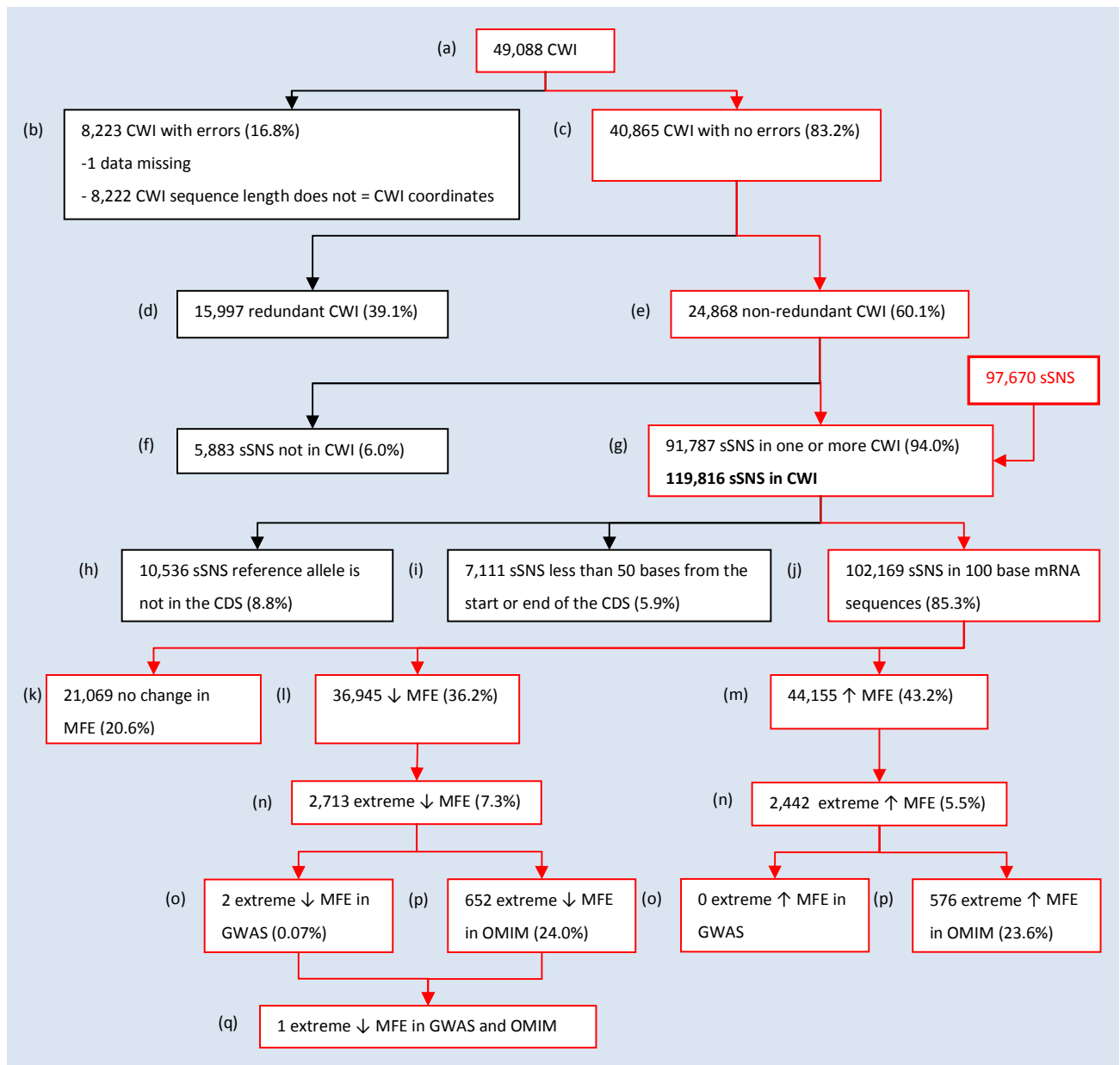


Figure 3.6: Flow-diagram for the identification of sSNS that potentially cause an extreme change in the minimum free-energy of local mRNA secondary structures, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM.

□ = records that were discarded from the datasets.

▬ = records that were retained in the datasets and used as input for the next analysis step.

CWI = coding regions with introns

CDS = coding sequences

MFE = the minimum free-energy of the 100 base local mRNA secondary structures

in GWAS = sSNS previously associated with a trait or disease in GWAS

in OMIM = sSNS within genes known to be associated with disease in OMIM

Table 3.11: The frequency of coding regions with introns (CWI) on the different chromosomes

Chromosome	Number of CWI	Percentage of CWI (%)
chr1	2,508	10.09
chr6	2,399	9.65
chr19	1,600	6.43
chr2	1,487	5.98
chr11	1,486	5.98
chr17	1,356	5.45
chr3	1,312	5.28
chr12	1,218	4.90
chr5	1,084	4.36
chr7	1,083	4.35
chrX	1,051	4.23
chr16	960	3.86
chr10	950	3.82
chr9	936	3.76
chr4	911	3.66
chr8	844	3.39
chr14	728	2.93
chr15	690	2.77
chr20	639	2.57
chr22	517	2.08
chr13	390	1.57
chr18	338	1.36
chr21	277	1.11
chrY	87	0.35
chrUn	17	0.07
Total	24,868	100.00

The sSNS and CWI records were joined to identify sSNS in CWI. Of the 97,670 sSNS, 5,833 (6.0%) were not within any of the CWI retrieved (**Fig.3.6.f**). A subset of 91,787 (94.0%) were within one or more of the CWI, thus, 119,816 sSNS in CWI records were created (**Fig.3.6.g**).

The introns were removed from the CWI sequences, to create the sSNS in mRNA (excluding the 5' and 3' UTRs) records. The 119,816 sSNS in mRNA records were checked for errors. A subset of 10,536 (8.8%) records was removed, as the reference alleles given by dbSNP131/NCBI were not in the mRNA sequences (**Fig.3.6.h**).

To accurately determine the MFE, 100 bases around each sSNS (50 bases on each end) were required. As the assessment of the local mRNA secondary structures did not include the 5'

and 3' UTRs, sSNS that were less than 50 bases from the start or end of the translated region of the mRNA sequences were excluded. Of the sSNS in mRNA sequence records, 7,111 (5.9%) records were removed as the sSNS were less than 50 bases from the start or end of the translated regions of the mRNA sequences (**Fig.3.6.i**). The 100 base reference and variant mRNA sequences were created for the remaining 102,126 sSNS in mRNA records (**Fig.3.6.j**). The frequency of the nucleotide changes in sSNS in the 100 base mRNA sequences followed the frequency distribution of the nucleotide change of the sSNS. As would be expected the most common nucleotide changes of the sSNS in the 100 base mRNA sequence were cytosine to thymine (C>T) transitions (32.0%) and guanine to adenine (G>A) transitions (21.7%) (**Table.D.3**).

RNAfold was used to calculate the MFE of the reference and variant 100 base local mRNA secondary structures. The change in MFE caused by sSNS in the 100 base local mRNA secondary structures was calculated by subtracting the reference MFE from the variant MFE. A change in MFE less than zero indicated that a sSNS potentially resulted in a decrease in MFE. A decrease in MFE could indicate that a sSNS increased the proportion of pairing bases in the mRNA secondary structure. A change in the MFE greater than zero indicated that a sSNS potentially resulted in an increase in MFE. An increase in MFE could indicate that a sSNS decreased the proportion of pairing bases in the mRNA secondary structure.

Of the 102,169 sSNS in the 100 base mRNA sequences, 21,069 (20.6%) were predicted to have no effect on MFE (**Fig.3.6.k**). A subset of 36,945 (36.2%) of the sSNS in the 100 base mRNA sequences were predicted to potentially cause a decrease in MFE (**Fig.3.6.l**). In addition, 44,155 (43.2%) of the sSNS in the 100 base mRNA sequences were predicted to potentially cause an increase in MFE (**Fig.3.6.m**).

All of the observed changes in the MFE caused by sSNS in the 100 base mRNA secondary were ordered in increasing order. The 10% most extreme decrease and 10% most extreme increase in MFE were chosen to represent an extreme change in MFE. The changes in MFE

ranged from -7.1 to 7.1. A change in MFE less than or equal to -3.7 was chosen to represent an extreme decrease in MFE. A change in MFE greater or equal to 4.0 was chosen to represent an extreme increase in MFE (**Fig.3.7**). Based on the chosen cut-off values, 2,713 sSNS were predicted to potentially cause an extreme decrease in MFE. A subset of 2,442 of the sSNS was predicted to potentially cause an extreme increase in MFE (**Fig.3.6.n**). Of the 102,169 sSNS in the 100 base mRNA sequences, 5,155 (5.0%) were predicted to potentially cause and extreme change in MFE.

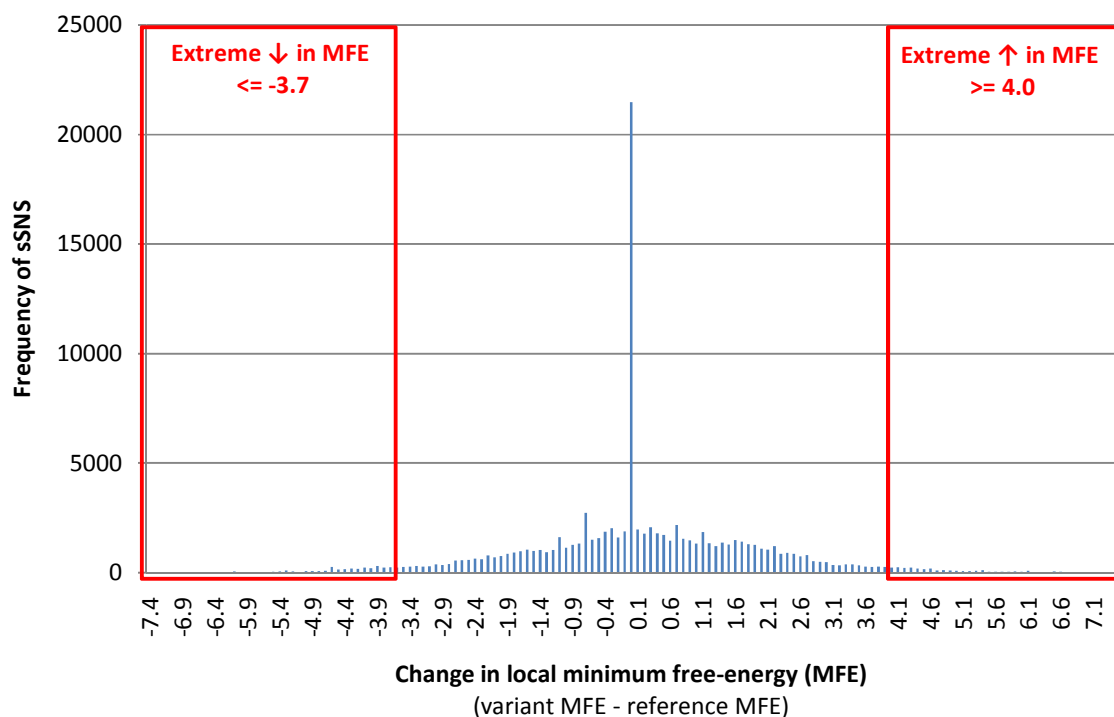


Figure 3.7: Frequency distribution graph of all the changes in the minimum free-energy of local mRNA secondary structures caused by sSNS.

The changes in MFE to the left of zero (negative values) on the graph, represent a decrease in MFE. The changes in MFE to the right of zero (positive values) on the graph, represent an increase in MFE. The two red squares show the 10% most extreme decrease and 10% most extreme increase in MFE caused by sSNS.

Adenine to guanine (A>G) transitions were the most common (30.4%) sSNS changes predicted to potentially result in an extreme decrease in MFE (**Table.D.4**). No adenine to guanine transitions were, however, predicted to potentially result in an extreme increase in MFE (**Table.D.4**). Guanine to adenine (G>A) transitions were by far the most common (55.2%) sSNS changes predicted to potentially result in an extreme increase in MFE. No guanine to adenine transitions were, however, predicted to potentially result in an extreme decrease in MFE. Thymine to cytosine (T>C) transitions were the least frequent (1.1%) sSNS predicted to potentially result in an extreme decrease in MFE, despite that thymine to cytosine transitions are the most common sSNS (**Fig.3.8**).

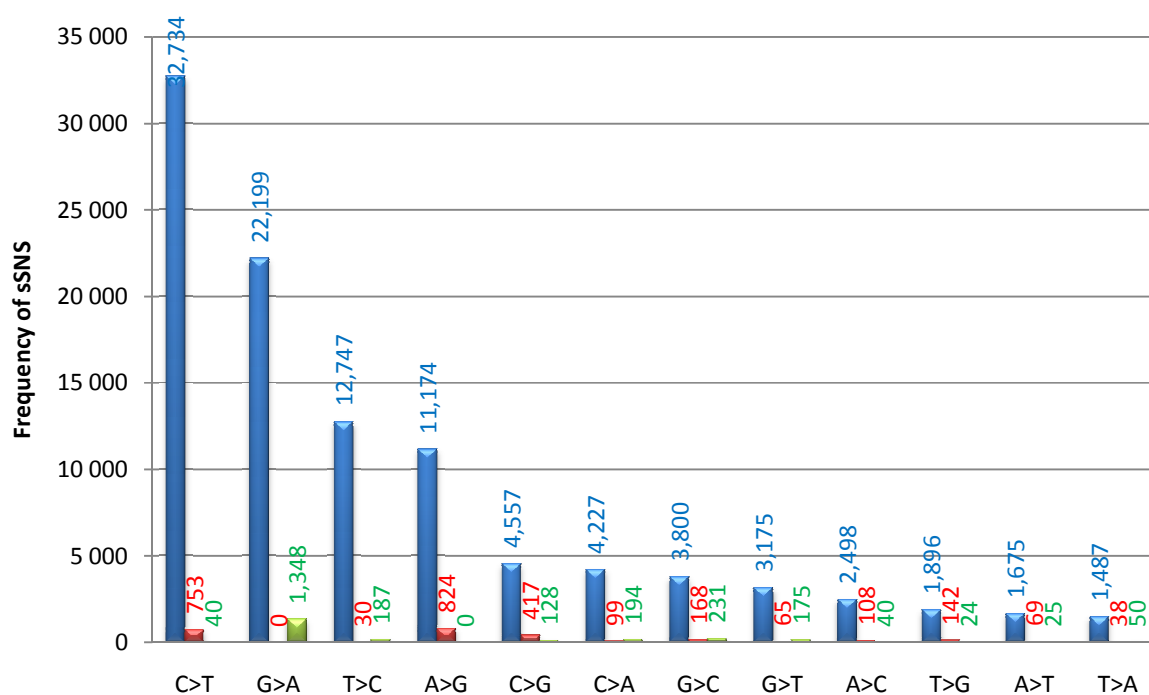


Figure 3.8: The frequency of sSNS predicted to potentially cause an extreme change in the minimum free-energy of local mRNA secondary structures.

- = sSNS in the 100 base mRNA sequences
- = sSNS predicted to potentially cause an extreme decrease in the minimum free-energy of the 100 base local mRNA secondary structures
- = sSNS predicted to potentially cause an extreme increase in the minimum free-energy of the 100 base local mRNA secondary structures

The sSNS predicted to potentially cause an extreme change in MFE were assessed to determine whether any had previously been associated with a trait or disease in GWAS and/or occurred within genes known to be associated with disease in OMIM. Only two (0.07%) of the sSNS predicted to potentially cause an extreme decrease in MFE had previously been associated with a trait or disease in GWAS. None of the sSNS predicted to potentially cause an extreme increase in MFE had previously been associated with a trait or disease in GWAS (**Fig.3.6.o**). Of the sSNS predicted to potentially cause an extreme decrease in MFE, 652 (24.0%) were within genes known to be associated with disease in OMIM. A subset of 576 (23.6%) of the sSNS predicted to potentially cause an extreme increase in MFE occurred within genes known to be associated with disease in OMIM (**Fig.3.6.p**). Only one sSNS predicted to potentially cause an extreme decrease in MFE had previously been associated with a trait in GWAS and occurred within a gene known to be associated with disease in OMIM (**Fig.3.6.q and Table.3.12**).

Table 3.12: The one sSNS predicted to potentially cause an extreme decrease in the minimum free-energy of a local mRNA secondary structure, which has previously been associated with a trait in GWAS and occurs within a gene known to be associated with disease in OMIM

Chromosome	sSNS position	Trait or disease in GWAS	Gene associated with disease in OMIM
		→ Gene: trait: risk-allele, p-value	→ Gene: disease (OMIM accession number)
chr6	32363954	<i>BTNL2</i> : Waist-hip ratio: rs2076529-C, 4.00E-07	<i>BTNL2</i> : Sarcoidosis, susceptibility to (606000)

3.4. sSNS predicted to potentially cause an extreme change in the maximum-entropy score of splice-sites

The sSNS were assessed to determine whether any could potentially cause aberrant splicing through an extreme change in the maximum-entropy score of splice-sites (ME). To identify which of the sSNS were within splice-sites, all known exons were first retrieved. Using Galaxy, 732,668 exons were retrieved from RefSeq (**Fig.3.9.a**). The retrieved exons were checked for errors. One of the records was removed, as the exon sequence length did not correspond to the exon coordinates given (**Fig.3.9.b**). The remaining, 732,667 exon records

with no errors were checked for redundancies (**Fig.3.9.c**). A subset of 501,569 (68.5%) of the exons were removed as redundant exons (**Fig.3.9.d**). The remaining dataset consisted of 231,098 non-redundant exons (**Fig.3.9.e**).

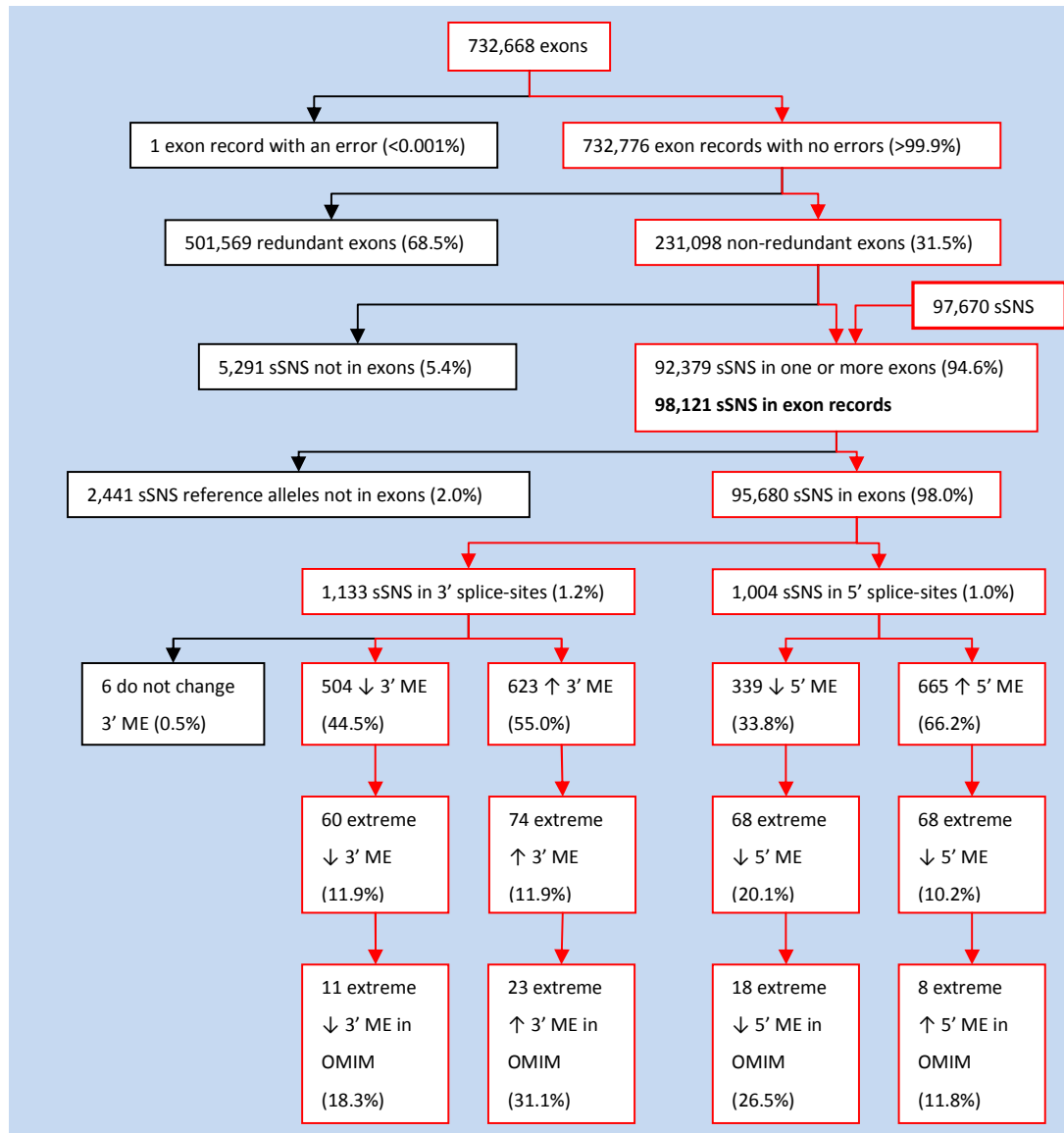


Figure 3.9: Flow-diagram for the identification of sSNS that potentially cause an extreme change in the maximum-entropy score of splice-sites, which occur within genes known to be associated with disease in OMIM.

□ = records that were discarded from the datasets.

□ = records that were retained in the datasets and used as input for the next analysis step.

3' ME = the maximum-entropy score of 3' acceptor splice-sites

5' ME = the maximum-entropy score of 5' donor splice-sites

in OMIM = sSNS within genes known to be associated with disease in OMIM

The exon records were separated according to the chromosomes on which the exons resided, to more quickly join the sSNS and exon records. As would be expected, chromosome 1 contained the most exons (9.8%). The smallest chromosome, chromosome 21 contained the second least amount of exons (1.0 %). The Y chromosome, which is the second smallest and least gene rich chromosome, contained the least amount of exons (0.4%). Only 85 (0.04%) of the exons were from unknown chromosomes (**Table.3.13**).

Table 3.13: The frequency of exons on the different chromosomes

Chromosome	Number of exons	Percentage of exons (%)
chr1	22,567	9.77
chr6	20,401	8.83
chr2	16,412	7.10
chr17	13,206	5.71
chr3	13,104	5.67
chr19	12,975	5.61
chr11	12,258	5.30
chr12	12,192	5.28
chr7	10,753	4.65
chr5	9,660	4.18
chr16	9,495	4.11
chr10	9,138	3.95
chr9	8,865	3.84
chr4	8,551	3.70
chrX	8,278	3.58
chr15	7,975	3.45
chr8	7,599	3.29
chr14	6,622	2.87
chr20	5,569	2.41
chr22	4,796	2.08
chr13	3,938	1.70
chr18	3,344	1.45
chr21	2,367	1.02
chrY	948	0.41
chrUn	85	0.04
Total	231,098	100.00

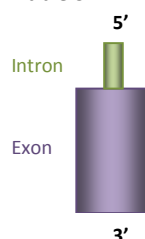
The sSNS in exon records were joined to identify sSNS in exons. Of the 97,670 sSNS, 5,291 (5.4%) were not within any of the exons retrieved (**Fig.3.9.f**). These sSNS may reside in introns or the exons in which these sSNS reside may have not been retrieved from RefSeq. The remaining, 92,379 sSNS were within one or more of the exons, thus, 98,121 sSNS in

individual exons were identified (**Fig.3.9.g**). A subset of 8,441 (2.5%) of the sSNS in exons were excluded as the reference alleles given by dbSNP 131/NCBI were not in the exon sequences (**Fig.3.9.h**).

The 95,680 sSNS in exons were assessed to determine which of the sSNS were in the exonic part of the splice-sites (**Fig.3.9.i**). sSNS in the exonic part of the 3' acceptor splice-sites were identified as sSNS in the first, second or third position of the exon (last three bases of the 3' splice-site). sSNS in the exonic part of the 5' donor splice-site were identified as sSNS in any of the last three bases of the exon (first three bases of the 5' splice-site) (**Fig.2.21**). Of the sSNS in exons, 1,133 (1.2%) were within the exonic part of the 3' splice-sites and 1,004 (1.0%) were within the exonic part of the 5' splice-sites (**Fig.3.9.j**).

Of the 1,133 sSNS in the exonic part of the 3' splice-sites, 193 (17.0%) were at the first position of the exon (third last base of the 3' splice-site). A subset of 348 (30.7%) of the sSNS in the exonic part of the 3' splice-site were at the second position of the exon (second last base of the 3' splice-site). The majority (592, 52.3%) of sSNS in the exonic part of the 3' splice-sites were at the third position of the exon (last base of the 3' splice-sites). Based on these observations it seems that the frequency of sSNS at each position in the exonic part of the 3' splice-sites increases the further away the base position is from the intron-exon junction (**Table.3.14**). As would be expected the most common sSNS changes in the exonic part of the 3' splice-sites were cytosine to thymine (C>T) transitions (24.6%) and guanine to adenine (G>A) transitions (22.7%) (**Table.D.6**).

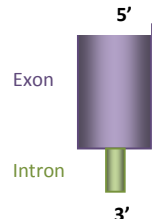
Table 3.14: The frequency of sSNS at the three bases in the exonic part of the 3' acceptor splice-sites



sSNS position in the exonic part of the 3' splice-site	Number of sSNS	Percentage of sSNS (%)
1	193	17.0
2	348	30.7
3	592	52.3
Total	1,133	100.0

Of the 1,004 sSNS in the exonic part of the 5' splice-site, 245 (24.4%) were at the third last position of the exon (first base of the 5' splice-site). A subset of 395 (39.3%) of the sSNS in the exonic part of the 5' splice-site were at the second last position of the exon (second base of the 5' splice-site) and 364 (36.3%) were at the last position of the exon (third base of the 5' splice-site). Based on these observations it seems that there is not a considerable difference between the proportion of sSNS at each of the three base positions in the exonic part of the 5' splice-sites (**Table.3.15**). As opposed to cytosine to thymine (C>T) transitions, guanine to adenine (G>A) transitions was the most common sSNS changes (31.2%) within the exonic part of the 5' splice-sites. Cytosine to thymine transitions was the second most frequent (26.8%) sSNS changes in the exonic part of the 5' splice-sites (**Table.D.7**).

Table 3.15: The frequency of sSNS at the three bases in the exonic part of the 5' donor splice-sites



sSNS position in the exonic part of the 5' splice-site	Number of sSNS	Percentage of sSNS (%)
1	245	24.4
2	395	39.3
3	364	36.3
Total	1,004	100.0

MaxEntScan was used to calculate the ME of the reference and variant splice-sites. The change in ME caused by a sSNS was calculated by subtracting the reference ME from the variant ME. A change in ME less than zero indicated that a sSNS potentially resulted in a decrease in ME. A decrease in ME could indicate that the sSNS decreased the signal strength of the splice-site. A change in the ME greater than zero indicated that a sSNS potentially resulted in an increase in ME. An increase in ME could indicate that the sSNS increased the signal strength of the splice-site.

Of the 1,133 sSNS in the exonic part of the 3' splice-sites, six (0.5%) were predicted to have no affect on the ME of the 3' splice-sites (**Fig.3.9.k**). A subset of 504 (44.5%) of the sSNS in the exonic part of the 3' splice-sites were predicted to potentially cause a decrease in ME

(Fig.3.9.l). In additions, 623 (55.0%) of the sSNS in the exonic part of the 3' splice-sites were predicted to potentially cause an increase in ME (Fig.3.9.m).

All of the observed changes in ME, caused by sSNS, were ordered in increasing order. The 10% most extreme decrease and 10% most extreme increase in ME were chosen to represent an extreme change in ME. The changes in the ME of the 3' splice-sites ranged from -6.30 to 2.48. A change in ME less than or equal to -1.27 was chosen to represent an extreme decrease in the ME of 3' splice-sites. A change in ME greater or equal to 1.11 was chosen to represent an extreme increase in the ME of 3' splice-sites (Fig.3.10). Based on the chosen cut-off values, 60 sSNS were predicted to potentially cause an extreme decrease in the ME of the 3' splice-sites. In addition, 74 sSNS were predicted to potentially cause an extreme increase in the ME of the 3' splice-sites (Fig.3.9.n). Of the sSNS in the exonic part of the 3' splice-sites, 134 (11.8%) were predicted to potentially cause an extreme change in ME.

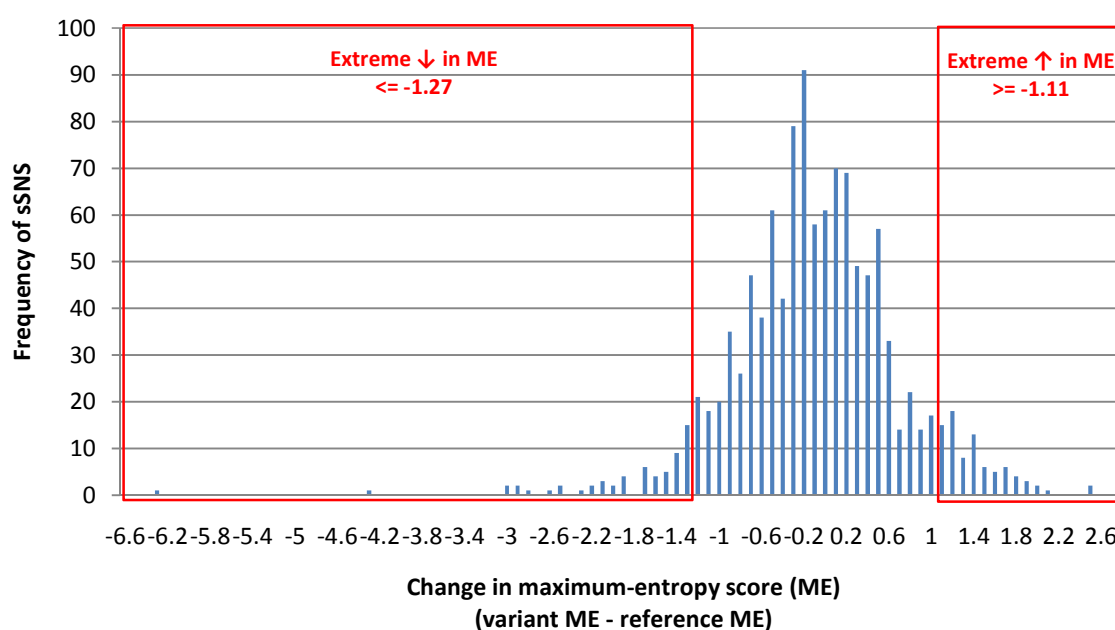


Figure 3.10: Frequency distribution graph of all the changes in the maximum-entropy score of 3' acceptor splice-sites caused by sSNS.

The changes in ME to the left of zero (negative values) on the graph, represent a decrease in ME. The changes in ME to the right of zero (positive values) on the graph, represent an increase in ME. The two red squares show the 10% most extreme decrease and 10% most extreme increase in ME caused by sSNS.

Of the 60 sSNS predicted to potentially cause an extreme decrease in the ME of 3' splice-sites, 36 (60.0%) were at the first position of the exons (third last base of the 3' splice-site). A subset of 24 (40.0%) of the sSNS predicted to potentially cause an extreme decrease in the ME of the 3' splice-sites were at the second position of the exon (second last base of the 3' splice-site). None of the sSNS predicted to potentially cause an extreme decrease in the ME of 3' splice-site were at the third position of the exon (last base on the splice-site) (**Fig.3.11**).

Of the 74 sSNS predicted to potentially cause an extreme increase in the ME of 3' splice-sites, 19 (25.7%) were at the first position of the exon (third base of the 3' splice-sites). The majority (54, 73.0%) of the sSNS predicted to potentially cause an extreme increase in the ME of 3' splice-sites were at the second position of the exon (second last base of the 3' splice-site). Only one (1.3%) of the sSNS predicted to potentially cause an extreme increase in the ME of the 3' splice-sites was at the third position of the exon (last base of the 3' splice-sites) (**Fig 3.11**).

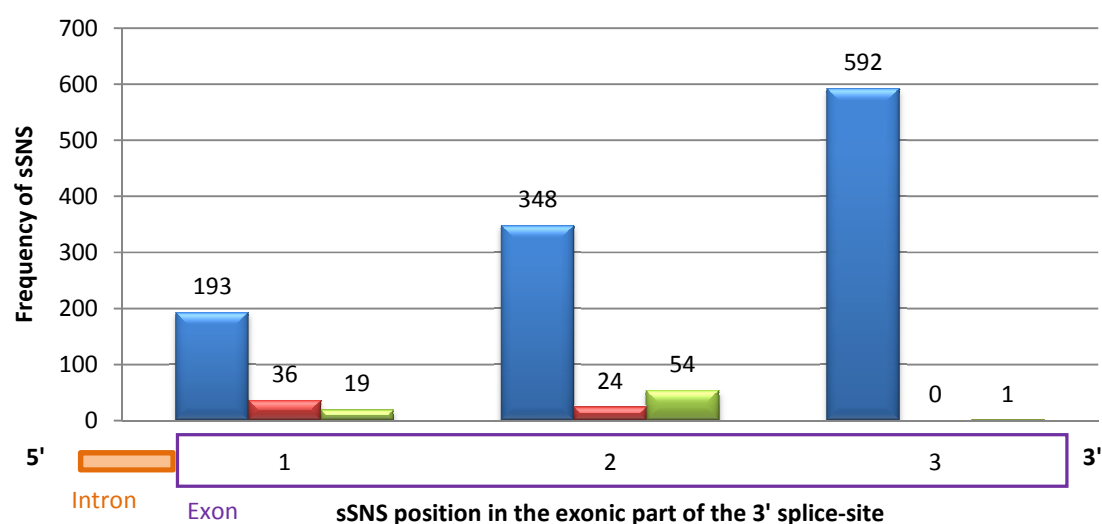


Figure 3.11: The frequency of sSNS predicted to potentially cause an extreme change in the maximum-entropy score of 3' splice-sites, at the three bases in the exonic part of the 3' splice-sites.

- = sSNS in the exonic part of the 3' splice-sites
- = sSNS predicted to potentially cause an extreme decrease in the maximum-entropy score of 3' splice-sites
- = sSNS predicted to potentially cause an extreme increase in the maximum-entropy score of 3' splice-sites

From the predictions, it seems that the third base in the exonic part of the 3' splice-site is more tolerant to change. Although the majority of sSNS in the 3' splice-site were at the third position of the exonic part of the splice-site, only one sSNS was predicted to potentially cause an extreme change in the ME of the 3' splice-site. The third base in the exonic part of the 3' splice-site may be under less selective pressure, allowing for single nucleotide substitutions that will not dramatically alter the signal strength of the 3' splice-sites. Of the 193 sSNS at the first position of the exonic part of the 3' splice-site, 55 (28.5%) were predicted to potentially cause an extreme change in ME. Of the 348 sSNS at the second position of the exonic part of the 3' splice-site, 78 (22.4%) were predicted to potentially cause an extreme change in ME. A large proportion of sSNS at the first two bases in the exonic part of the 3' splice-sites may potentially cause an extreme change in ME. The first two bases in the exonic part of the 3' splice-sites may be under more selective pressure to remain constant. Any changes in the first two bases in the exonic part of the 3' splice-site are thus more likely to alter the signal strength of the 3' splice-sites.

The two most common sSNS changes predicted to potentially cause an extreme decrease in the ME of 3' splice-sites were guanine to adenine (G>A) transitions (28.3%) and thymine to cytosine (T>C) transitions (28.3%). The least frequent sSNS changes predicted to potentially cause an extreme decrease in the ME of 3' splice-sites was cytosine to thymine (C>T) transitions (1.7%), despite that cytosine to thymine transitions was the most common sSNS (**Table.D.8**). Although adenine to guanine (A>G) transitions was the fourth most common sSNS, none were predicted to potentially cause an extreme decrease in the ME of 3' splice-sites. No cytosine to guanine (C>G), cytosine to adenine (C>A) or adenine to cytosine (A>C) transversion were predicted to potentially cause an extreme decrease in the ME of 3' splice-sites. The reason for this may be that these transversion substitutions are only a minor proportion of sSNS (**Fig.3.12**).

The two most common sSNS changes predicted to potentially cause an extreme increase in the ME of 3' splice-sites were cytosine to thymine (C>T) transitions (59.5%) and guanine to

adenine (G>A) transitions (19.0%). Thymine to cytosine (T>C) transition are the third most common sSNS. Guanine to adenine (G>A) transitions are the second most common sSNS. Thymine to cytosine transitions (2.7%) and guanine to adenine transition (1.4%) were, however, the least frequent sSNS predicted to potentially cause an extreme increase in the ME of 3' splice-sites (**Table.D.9**). No cytosine to adenine (C>A), guanine to cytosine (G>C), adenine to cytosine (A>C) or thymine to adenine (T>C) transversions were predicted to potentially cause an extreme increase in the ME of 3' splice-sites. The reason for this may be that these transversion substitutions are only a small proportion of sSNS (**Fig.3.12**).

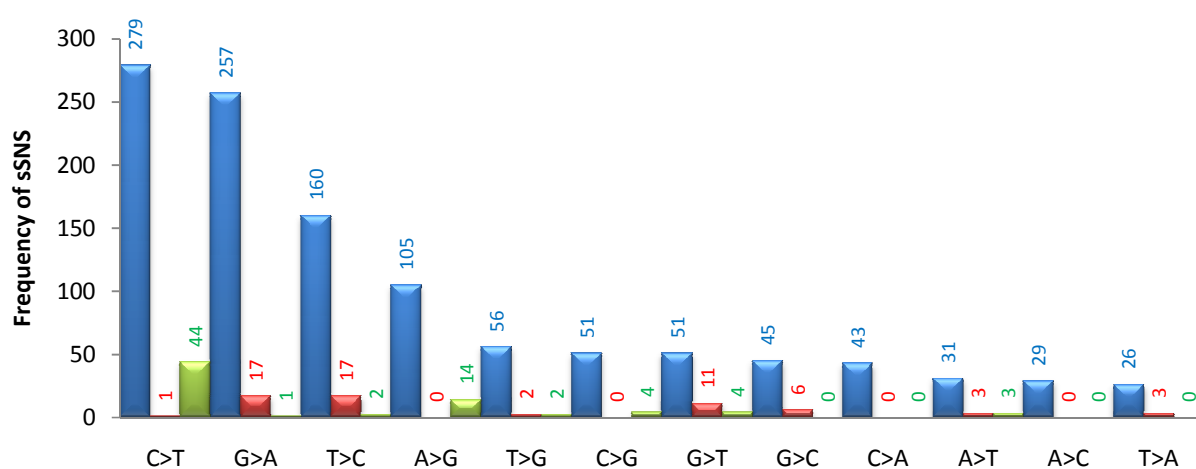


Figure 3.12: The frequency of sSNS predicted to potentially cause an extreme change in the maximum-entropy score of 3' splice-sites.

- = sSNS in the exonic part of 3' splice-sites
- = sSNS predicted to potentially cause an extreme decrease in the maximum-entropy score of 3' splice-sites
- = sSNS predicted to potentially cause an extreme increase in the maximum-entropy score of 3' splice-sites

The changes in the ME of the 5' splice-sites ranged from -12.3 to 7.83. A change in ME less than or equal to -5.28 was chosen to represent an extreme decrease in the ME of 5' splice-sites. A change in ME greater or equal to 1.87 was chosen to represent an extreme increase in the ME of 5' splice-sites (**Fig.3.13**). Based on the chosen cut-off values, 68 sSNS were predicted to potentially cause an extreme decrease in the ME of 5' splice-sites. In addition, 68 sSNS were predicted to potentially cause an extreme increase in the ME of 5' splice-sites (**Fig.3.9.n**). Of the 1,004 sSNS in the exonic part of the 5' splice-sites, 136 (13.5%) were predicted to potentially cause an extreme change in ME.

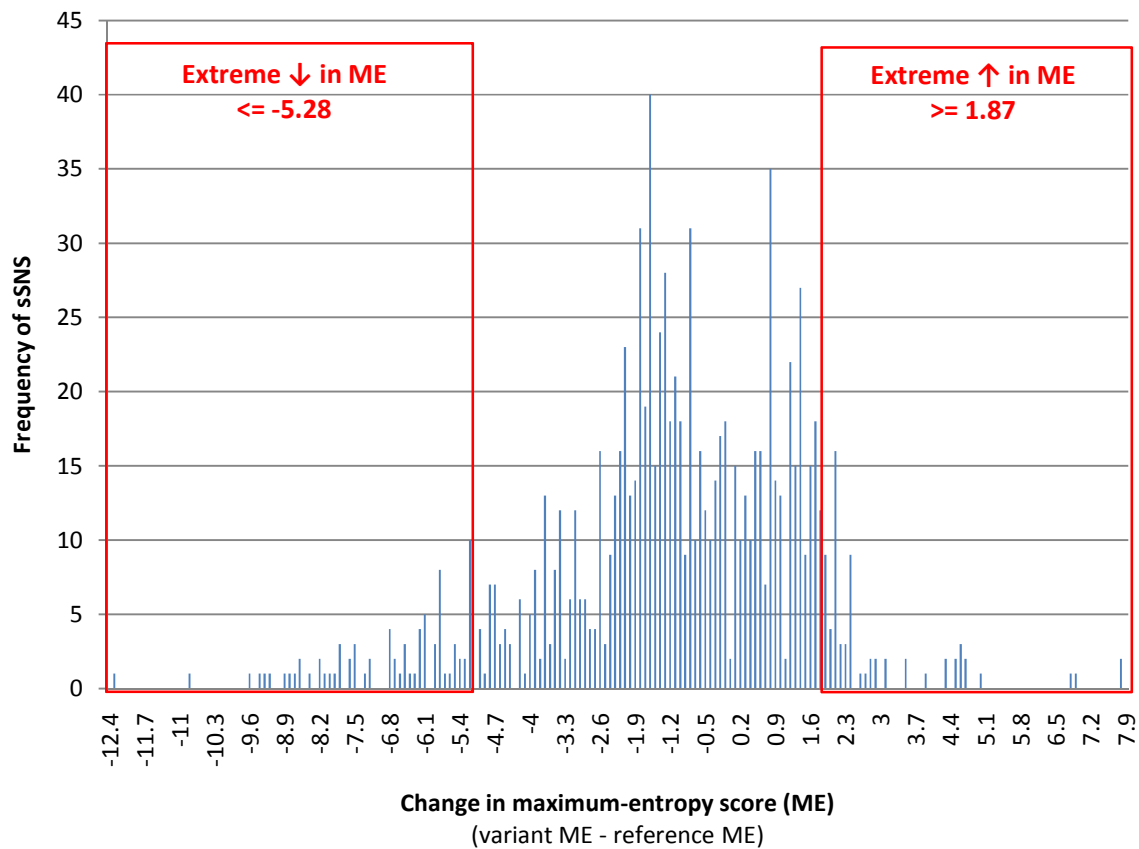


Figure 3.13: Frequency distribution graph of all the changes in the maximum-entropy score of 5' donor splice-sites caused by sSNS.

The changes in ME to the left of zero (negative values) on the graph, represent a decrease in ME. The changes in ME to the right of zero (positive values) on the graph, represent an increase in ME. The two red squares show the 10% most extreme decrease and 10% most extreme increase in ME caused by sSNS.

Of the 68 sSNS predicted to potentially cause an extreme decrease in the ME of the 5' splice-sites, only one (1.5%) was at the third last position of the exon (first base of the 5' splice-sites). Only one (1.5%) of the sSNS predicted to potentially cause an extreme decrease in the ME of the 5' splice-sites was at the second base of the 5' splice-site. The majority (66, 97.0%) of sSNS predicted to potentially cause an extreme decrease in ME of the 5' splice-sites were at the last portions of the exon (third base of the 5' splice-site) (**Fig.3.14**).

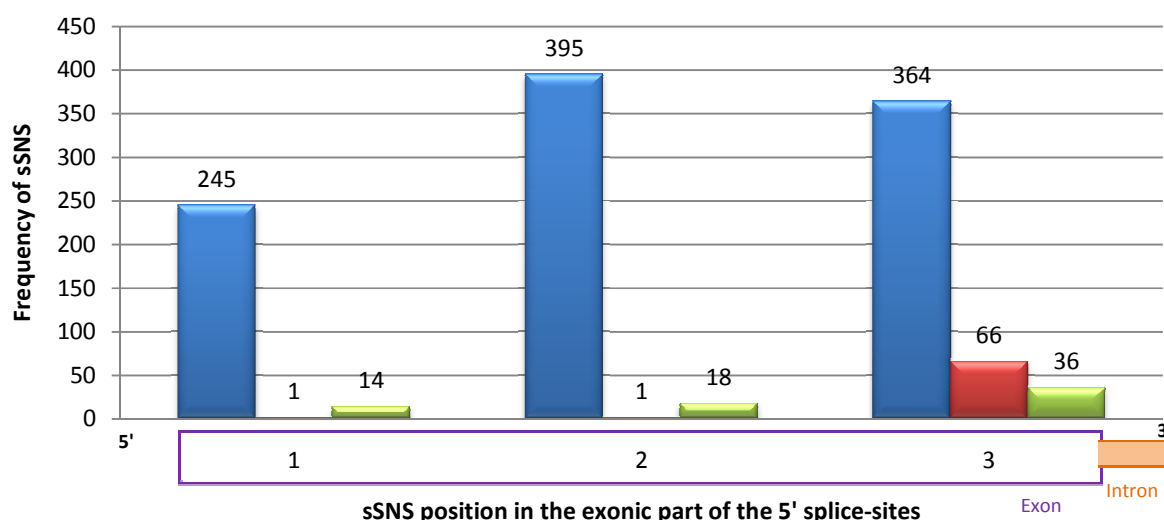


Figure 3.14: The frequency of sSNS predicted to potentially cause an extreme change in the maximum-entropy score of 5' splice-sites, at the three bases in the exonic part of the 5' splice-sites.

- = sSNS in the exonic part of the 5' splice-sites
- = sSNS predicted to potentially cause an extreme decrease in the maximum-entropy score of 5' splice-sites
- = sSNS predicted to potentially cause an extreme increase in the maximum-entropy score of 5' splice-sites

Of the 68 sSNS predicted to potentially cause an extreme increase in the ME of the 5' splice-sites, 14 (20.6%) were at the third last position of the exon (first base of the 5' splice-site). A subset of 18 (26.5%) of the sSNS predicted to potentially cause an extreme increase in the ME of 5' splice-sites were at the second last base of the exon (second base of the 5' splice-sites). The majority (36, 52.9%) of sSNS predicted to potentially cause an extreme increase in the ME of 5' splice-sites were at the last base position of the exon (third base of the splice-site) (**Fig.3.14**).

Based on these observations it seems that the third base in the exonic part of the 5' splice-sites is less tolerant to change. Of the 364 sSNS at the third base position of the exonic part of the 5' splice-sites, 102 (28.0%) were predicted to potentially cause an extreme change in ME. The third base of the exonic part of the 5' splice-sites may be under more selective pressure, to prevent single nucleotide substitutions that will dramatically alter the signal strength of the 5' splice-sites. Of the 245 sSNS at the first base of the exonic part of the 5' splice-site, 15 (6.1%) were predicted to potentially cause an extreme change in ME. Of the

395 sSNS at the second base of the exonic part of the 5' splice-sites, 19 (4.8%) were predicted to potentially cause an extreme change in ME. Only a small portion of sSNS at the first two bases of the exonic part on the 5' splice-sites will potentially cause an extreme change in ME. The first two bases in the exonic part of the 5' splice-sites may be under less selective pressure, allowing for single nucleotide substitutions that will not dramatically alter the signal strength of the 5' splice-sites. Comparing the predictions of the sSNS predicted to potentially cause an extreme change in the ME of the 3' splice-sites (**Fig.3.11**) and the 5' splice-sites (**Fig.3.14**), it seems that the closer the base is to the intron-exon junction the less tolerant it is to change.

By far the most common sSNS changes predicted to potentially cause an extreme decrease in the ME of the 5' splice-sites was guanine to adenine (G>A) transitions (86.8%). Cytosine to thymine (C>T) transition were the most common sSNS. Adenine to guanine (A>G) transitions were the fourth most common sSNS. Cytosine to thymine (C>T) transitions (1.5%) and adenine to guanine (A>G) transitions (1.5%) were, however, the least frequent sSNS changes predicted to potentially cause an extreme decrease in the ME of 5' splice-sites. (**Table.D.10**). No thymine to cytosine (T>C) transitions were predicted to potentially cause an extreme decrease in the ME of 5' splice-sites, despite that thymine to cytosine transitions was the third most common sSNS changes. Of the all the possible transversion substitutions, only guanine to thymine (G>T) transversions (5.8%) and guanine to cytosine (G>C) transversions (4.4%) were predicted to potentially cause an extreme decrease in the ME of 5' splice-sites (**Fig.3.15**).

The most frequent sSNS change predicted to potentially cause an extreme increase in the ME of 5' splice-sites was thymine to cytosine (T>C) transitions (45.6%) (**Table.D.11**). No cytosine to thymine (C>T) transitions were predicted to potentially cause an extreme increase in the ME score of 5' splice-sites, despite that cytosine to thymine transitions was the most common sSNS change. Of all the possible transversion substitutions, only guanine to cytosine (G>C) transversions (10.3%), cytosine to guanine (C>G) transversions (1.5%) and

thymine to guanine (T>G) transversions (1.5%) were predicted to potentially cause an extreme increase in the ME of 5' splice-sites (**Fig.3.15**).

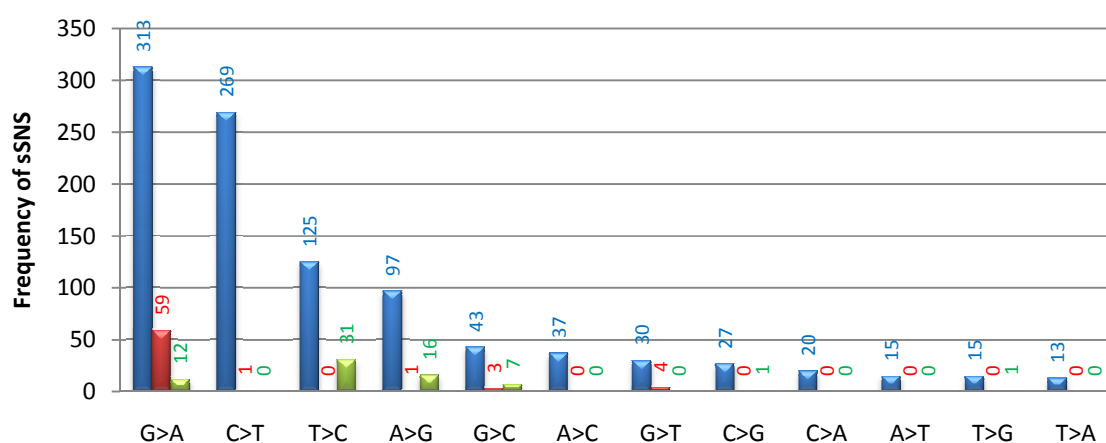


Figure 3.15: The frequency of sSNS predicted to potentially cause an extreme change in the maximum-entropy score of 5' splice-sites.

- = sSNS in the exonic part of 5' splice-sites
- = sSNS predicted to potentially cause an extreme decrease in the maximum-entropy score of 5' splice-sites
- = sSNS predicted to potentially cause an extreme increase in the maximum-entropy score of 5' splice-sites

The sSNS predicted to potentially cause an extreme change in ME were assessed to determine whether any had previously been associated with a trait or disease in GWAS and/or occurred within genes known to be associated with disease in OMIM. None of the sSNS predicted to potentially cause an extreme change in the ME of the 3' or 5' splice-sites had previously been associated with a trait or disease in GWAS. Of the 60 sSNS predicted to potentially cause an extreme decrease in the ME of 3' splice-sites, 11 (18.3%) were within disease genes. Of the 74 sSNS predicted to potentially cause an extreme increase in the ME of 3' splice-sites, 23 (31.1%) were within disease genes. Of the 68 sSNS predicted to potentially cause an extreme decrease in the ME of the 5' splice-sites, 18 (26.5%) were within disease genes. Of the 68 sSNS predicted to potentially cause an extreme increase in the ME of the 5' splice-sites, eight (11.8%) were within disease genes. A subset of 60 (22.2%), of the 270 sSNS predicted to potentially cause an extreme change in ME were within genes known to be associated with disease in OMIM (**Fig.3.9.o**).

3.5. Identification of sSNPs with a potential functional impact

The proportion of sSNPs with a potential functional impact was determined from the sSNPs predicted to have a potential functional impact. Of the 90,102 sSNPs identified, 17,603 (19.5%) were predicted to potentially cause one extreme change in CUF. A subset of 4,029 (4.5%) of the sSNPs were predicted to potentially cause an extreme change in MFE. In addition, 256 (0.3%) of the sSNPs were predicted to potentially cause an extreme change in ME. Of the sSNPs, 21,888 (24.3%) were predicted to have a potential functional impact either through a change in CUF, MFE or ME (**Fig.3.16**).

A proportion of the sSNPs were predicted to have a potential functional impact through more than one of the three molecular mechanisms investigated. Of the sSNPs identified, 21,086 (23.4%) were predicted to potentially have a functional impact through one or more of the three molecular mechanisms investigated (**Fig.3.17**). Of the sSNPs, 16,808 (18.7%) were predicted to have a potential functional impact only through an extreme change in CUF. A subset of 3,263 (3.6%) of the sSNPs were predicted to have a potential functional impact only through an extreme change in MFE. In addition, 215 (0.2%) of the sSNPs were predicted to have a potential functional impact only through an extreme change in ME. Of the sSNPs, 759 (0.8%) were predicted to have a potential functional impact through an extreme change in CUF and MFE. A subset of 34 (0.04%) of the sSNPs were predicted to have a potential functional impact through an extreme change in CUF and ME. In addition, five (0.01%) of the sSNPs were predicted to have a potential functional impact through an extreme change in MFE and ME. Only two (0.002%) of the sSNPs were predicted to have a potential functional impact through an extreme change in CUF, MFE and ME. One of these sSNPs is at position 49212676 of chromosome 20 (rs6126047). This sNP is within the family with sequence similarity 65, member c (*FAM65C*) gene. The function of the protein encoded by *FAM65C* is unknown (<http://www.ncbi.nlm.nih.gov/gene/140876>). The other sNP predicted to potentially cause an extreme change in CUF, MFE and ME is within the adenylate cyclase 7

(*ADCY7*) gene. *ADCY7* encodes adenylate cyclase, which catalysis the production of cyclic AMP from ATP (<http://www.genecards.org/cgi-bin/carddisp.pl?gene=ADCY7&search=ADCY7>).

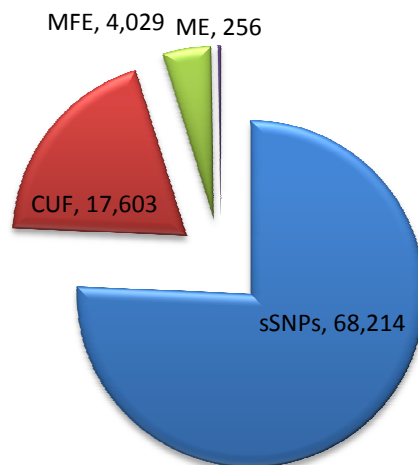


Figure 3.16: The proportion of sSNPs predicted to potentially have a functional impact either through an extreme change in codon-usage frequency, the minimum free-energy of local mRNA secondary structures or the maximum-entropy score of splice-sites.

There is some redundancy between the segments of the graph.

- = sSNPs not predicted to have a potential functional impact
- = sSNPs predicted to potentially cause an extreme change in codon-usage frequency
- = sSNPs predicted to potentially cause an extreme change in the minimum free-energy of local mRNA secondary structures
- = sSNPs predicted to potentially cause an extreme change in the maximum-entropy score of splice-sites

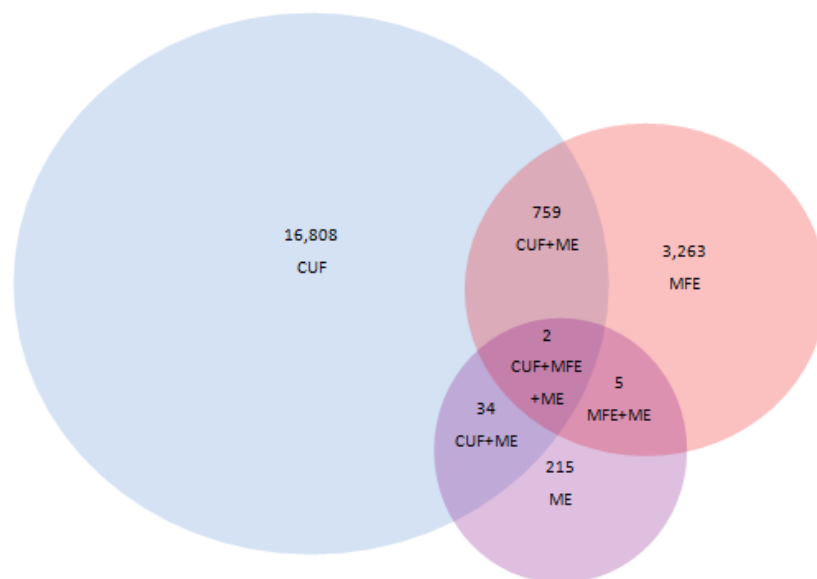


Figure 3.17: The number of sSNPs predicted to potentially have a functional impact through one or more of the three molecular mechanisms investigated.

- **CUF** = sSNPs predicted to potentially cause an extreme change in codon-usage frequency
- **MFE** = sSNPs predicted to potentially cause an extreme change in the minimum free-energy of local mRNA secondary structures
- **ME** = sSNPs predicted to potentially cause an extreme change in the maximum-entropy score of splice-sites

3.6. sSNPs predicted to have a potential functional impact, which have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM

The sSNPs predicted have a potential functional impact were assessed to determine how many had previously been associated with a trait or disease in GWAS and/or occurred within genes known to be associated with disease in OMIM. None of the sSNPs predicted potentially cause an extreme change in ME had previously been associated with a trait or disease in GWAS. A subset of 12 (0.07%) of the sSNPs predicted to potentially cause an

extreme change in CUF had previously been associated with a trait or disease in GWAS. Only one (0.03%) of the 3,263 sSNPs predicted to potentially cause an extreme change in MFE had previously been associated with a trait in GWAS. Of the 759 sSNPs predicted to potentially cause an extreme change in CUF and MFE, only one (0.1%) had previously been associated with a trait in GWAS. A subset of 14 (0.07%) of the 21,086 sSNPs predicted to potentially have a functional impact through one or more of the three molecular mechanisms investigated had previously been associated with a trait or disease in GWAS (**Table.3.16**). This small number is not unexpected as SNPs on GWAS platforms are generally chosen for allele frequency (greater than 0.05) and ability to tag linkage-disequilibrium blocks. They are not expected to be enriched for causal or functional alleles.

Table 3.16: The frequency of sSNPs predicted to have a potential functional impact, which have been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM

	Potentially functional sSNPs	GWAS	OMIM	GWAS and OMIM
CUF	16,808	12	3,151	5
MFE	3,263	1	627	0
ME	215	0	28	0
CUF+MFE	759	1	247	1
CUF+ME	34	0	3	0
MFE+ME	5	0	1	0
CUF+MFE+ME	2	0	0	0
Total	21,086	14	4,057	6

As would be expected, a much larger proportion of sSNPs predicted to have a potential functional impact were within genes known to be associated with disease in OMIM, than associated with a trait or disease in GWAS. Of the 16,808 sSNPs predicted to potentially cause an extreme change in CUF, 3,151 (18.7%) occurred within disease genes. A subset of 627 (19.2%) of the 3,263 sSNPs predicted to potentially cause an extreme change in MFE were within disease genes. In addition, 28 (13.0%) of the 215 sSNPs predicted to potentially cause an extreme change in ME were within disease genes. Of the 759 sSNPs predicted to potentially cause an extreme change in CUF and MFE, 247 (32.5%) were within disease genes. A subset of three (8.8%) of the 34 sSNPs predicted to potentially cause an extreme

change in CUF and ME were within disease genes. Only one (20.0%) of the five sSNPs predicted to potentially cause an extreme change in MFE and ME were within disease genes. None of the two sSNPs predicted to potentially cause an extreme change in CUF, MFE and ME were within disease genes. Of the 21,086 sSNPs predicted to have a potential functional impact through one or more of the three molecular mechanisms investigated, 4,057 (19.2%) were within genes known to be associated with disease in OMIM (**Table.3.16**).

A minor proportion of the sSNPs predicted to have a potential functional impact had previously been associated with a trait of disease in GWAS and occurred within genes known to be associated with disease in OMIM. Of the 16,808 sSNPs predicted to potentially cause an extreme change in CUF, five (0.03%) had previously been associated with a trait or disease in GWAS and were within disease genes. Only one (0.1%) of the 759 sSNPs predicted to potentially cause an extreme change in CUF and MFE had previously been associated with a trait in GWAS and was in a disease gene (**Table.3.17**). None of the other sSNPs predicted to have a potential functional impact had previously been associated with a trait or disease in GWAS and occurred within disease genes. Of the 21,086 sSNPs predicted to have a potential functional impact through one or more of the three molecular mechanisms investigated, only six (0.03%) had previously been associated with a trait or disease in GWAS and occurred within genes known to be associated with disease in OMIM (**Table.3.16**).

Table 3.17: The six sSNPs predicted to have a potential functional impact through one or more of the three molecular mechanisms investigated, which have previously been associated with a trait or disease in GWAS and occur within genes known to be associated with disease in OMIM

Chrom	SSNS position	Molecular mechanism	Trait or disease in GWAS → trait: risk-allele, p-value	Gene associated with disease in OMIM → gene start – gene end: disease (OMIM accession number)
chr2	21232194	CUF	<i>APOB</i> : LDL cholesterol: rs693-G, 4.00E-17	<i>APOB</i> , <i>FLDB</i> , <i>LDLCQ4</i> : Hypercholesterolemia, due to ligand-defective apo B (107730); Hypobetalipoproteinemia (107730); Hypobetalipoproteinemia, normotriglyceridemic (107730)
chr3	133475721	CUF	<i>TF</i> : Serum markers of iron status: rs1799852-?, 5.00E-06	<i>TF</i> : Atransferrinemia, (190000); Iron deficiency anemia, susceptibility to (190000)
chr6	32363954	CUF + MFE	<i>BTNL2</i> : Waist-hip ratio: rs2076529-C, 4.00E-07	<i>BTNL2</i> , <i>SS2</i> : Sarcoidosis, susceptibility to (606000)
chr6	32363843	CUF	<i>BTNL2</i> : Ulcerative colitis: rs9268480-G, 3.00E-06	<i>BTNL2</i> , <i>SS2</i> : Sarcoidosis, susceptibility to (606000)
chr13	50080846	CUF	<i>PHF11</i> : Cardiac hypertrophy: rs2031532-?, 5.00E-06	<i>PHF11</i> , <i>NYREN34</i> : IgE levels QTL (607796); Asthma (607796)
chr21	37833306	CUF	<i>CLDN14</i> : Kidney stones: rs219780-C, 4.00E-12	<i>TMPRSS6</i> , <i>IRIDA</i> : Deafness, autosomal recessive (605608)

CHAPTER 4: DISCUSSION

sSNPs were thought to be silent, as they do not change the amino acid encoded. Although a number of sSNPs are not silent and several have been associated with disease mechanisms, the potential functional impact of sSNPs on a genome-wide scale has not been determined. In-order to identify sSNPs on a genome-wide scale that may have a potential functional impact, I created SynSNP.

SynSNP is the first computational tool that predicts the potential functional impact of sSNPs either through a change in translational and protein folding kinetics or aberrant splicing. SynSNP determines which of the sSNPs predicted to have a potential functional impact are within genes known to be associated with disease in OMIM. SynSNP also determines whether any of the sSNPs predicted to have a potential functional impact have previously been associated with a trait or disease in GWAS. The sSNPs previously associated with a trait or disease may contribute to the trait or disease risk, or may be in linkage disequilibrium with the causative agent.

The purpose of SynSNP is to add to our understanding of the contribution of synonymous mutations to human disease. SynSNP may be used to identify the potential functional impact of any sSNP, given that the input file contains all of the columns required by SynSNP. For this reason, SynSNP may be used to identify the potential functional impact of novel synonymous variants as they are discovered. Although nsSNPs change the amino acid encoded, nsSNPs may also alter the final protein structure and/or function through a change in translational kinetics or alternative splicing. It may be interesting to determine the proportion of nsSNPs that may potentially alter translational kinetics or alternative splicing, with the use of SynSNP.

One potential use of SynSNP may be to assess chromosomal regions identified by associated SNPs in GWAS data. Once the disease associated tagged SNPs have been identified, SynSNP could be used to determine the potential functional impact of all sSNPs lying in the same haploblock as the associated SNPs from the GWAS data. As SynSNP was created to use dbSNP data, some parts of SynSNP may need to be modified to cater for the different data structure. The principles of how SynSNP identifies potentially functional sSNPs will however remain the same. In this way, SynSNP could be used by researchers to identify which of the sSNPs within the chromosomal regions identified by GWAS may potentially contribute to human disease and may be worth investigating further.

SynSNP was used to identify sSNPs with a potential functional impact, from cSNPs retrieved from dbSNP131. Of the cSNPs retrieved from dbSNP 131, 90,102 (40%) were predicted to be sSNPs. It may seem surprising that there are more nsSNPs (60%) than sSNPs (40%) in dbSNP131, as sSNPs are believed to be more likely to be tolerated than nsSNPs and are generally found in much higher abundance. In general, sSNPs have little to no selective advantage or disadvantage. Due to the lack of selective pressure, sSNPs occur more frequently than nsSNPs within the human genome ([Fairbanks and Anderson, 1999](#)). An allele that provides a selective advantage may become fixed in a population due to positive selection. As many sSNPs have little or no selective advantage, there is little reason for sSNPs to become fixed within a population. For these reasons, sSNPs may occur more frequently than nsSNPs within individual genomes, however, more nsSNPs than sSNPs will become fixed in a population, due to local adaption.

In the 1000 Genomes Project, each of the population comparisons had on average a low level of population differentiation; however, hundreds of thousands of SNPs had large allele frequency differences. These SNPs, with large allele frequency differences were highly enriched with nsSNPs, indicative of local adaption ([Consortium, 2010](#)). When comparing different populations more nsSNPs with large allele frequency differences than sSNPs will be identified, due to local adaption. As dbSNP contains SNPs from various different population

groups, the database may contain more nsSNPs than sSNPs due to the impact of local adaption in the different population groups included in dbSNP. Local adaption often causes large allele frequency changes, predominantly on nsSNPs. As more targeted sequencing is performed, it is possible that additional rare variants will be identified and included in dbSNP. As additional rare variants are included, it is possible that the ratio of synonymous variants within dbSNP will increase in relation to the number of non-synonymous variants.

4.1. sSNPs predicted to potentially cause an extreme change in codon-usage frequency

A sSNP that results in a change in CUF may potentially alter translational and protein folding kinetics, which may result in a protein of altered structure and function (Kimchi-Sarfaty et al., 2007). Of the 90,102 sSNP identified, 17,603 (19.5%) were predicted to potentially cause an extreme change in CUF. A sSNP that changes a more frequently used codon to a less frequently used codon may cause the ribosome to be stalled while the corresponding *iso*-acceptor tRNA is being recruited. A pause in translation may provide additional waiting time where the growing polypeptide may fold into a different conformation. This change in the folding of the polypeptide may result in a protein of altered structure and function (Kimchi-Sarfaty et al., 2007, Tsai et al., 2008).

sSNPs that cause a codon change that results in a codon with reduced usage frequency may alter translational and protein folding kinetics. However, it is still unknown whether sSNPs that result in a codon with increased codon-usage frequency may have a functional impact. sSNPs that change a less frequently used codon to a more frequently used codon may increase the rate of translation, as the codon's corresponding *iso*-acceptor tRNA may be recruited more quickly. In the majority of cases the growing polypeptide will likely still fold at the preferred regions, despite the increase in translational kinetics. The increase in translational kinetics may make more of the polypeptide sequences available more quickly.

As more of the polypeptide sequence is available, the polypeptide may fold in a different way. If the folded polypeptide remains kinetically trapped it may result in a protein of altered structure and function. It is, thus, plausible that sSNPs that cause a codon change that results in a codon with a decreased or increase usage frequency may alter translational and protein folding kinetics, but experimental evidence is required to confirm this possibility.

The change in CUF caused by each sSNP was calculated, using the human CUF table. It is likely that the abundance of *iso*-acceptor tRNAs differ in different tissues, due to the requirement of varied gene expression patterns in different tissues. Consequently, because CUF is dependent on the abundance of *iso*-acceptor tRNAs it is possible that CUF differs between different tissues ([Plotkin et al., 2004](#), [Dittmar et al., 2006](#), [Kotlar and Lavner, 2006](#)). It may have been interesting to compare the changes in CUF in different tissues; however, no tissue specific CUF tables are yet available. sSNPs that alter translational kinetics may alter the protein folding pattern of upstream protein domains ([Fung and Gottesman, 2009](#)). Thus, it may have been interesting to determine whether any of the sSNPs that were predicted to potentially cause an extreme change in CUF were downstream of any functionally important protein domains.

Using the chosen cut-off values to represent an extreme change in CUF, the C3435T sSNP that contributes to a change in the MDR1 protein structure and function was not identified. In this study, an extreme decrease in CUF was defined as a change in CUF of -19.2 or a more extreme change. The C3435T sSNP results in a change in CUF of -10.9, thus, the sSNP was not predicted to result in an extreme change in CUF. SynSNP did not identify the C3435T sSNP as a sSNP with a potential functional impact, which could indicate that the chosen cut-off values were too stringent. On the other hand, it is unlikely that a single sSNP will alone have a dramatic effect on the rate of translation. The *MDR1*, C3435T sSNP does reduce the rate of translation; however, it is the three SNPs in the *MDR1* common haplotype that together contribute to the dramatic affect on translational and protein folding kinetics. It is, therefore, more likely that a number of sSNPs in *cis* and on the same allele, rather than a

single sSNP, will together contribute to a change in translational and protein folding kinetics (Kimchi-Sarfaty et al., 2007, Tsai et al., 2008). The next step would be to assess the changes in CUF caused by sSNPs in close-proximity to other SNPs. This may be achieved through the identification of sSNPs in common haplotype blocks. The results obtained in this study give an indication of the proportion of individual sSNPs that may potentially result in an extreme change in CUF. These sSNPs may alter translational and protein folding kinetics when in close-proximity to other SNPs that also result in an extreme change in CUF.

4.2. sSNPs predicted to potentially cause an extreme change in the minimum free-energy of local mRNA secondary structures

MFE gives an indication of the proportion of base-pairs and consequently the stabilities of mRNA secondary structures (Tinoco et al., 1971, Zuker and Stiegler, 1981). sSNPs that increase the proportion of pairing bases in an mRNA secondary structure would decrease the MFE and increase the stability of the mRNA secondary structure. sSNPs that decrease the proportion of pairing bases in an mRNA secondary structure would increase the MFE and decrease the stability of the mRNA secondary structure (Tinoco et al., 1971, Zuker and Stiegler, 1981).

Much is known about the base-pairing rules that govern mRNA secondary structures. It is, however, unclear how changes in mRNA secondary structures may alter translational kinetics. An increase in mRNA stability may decrease the rate of mRNA degradation (Hollams et al., 2002). The reduced rate of mRNA degradation may allow the transcript to be translated more times, increasing the amount of protein product (Duan et al., 2003, Duan and Antezana, 2003). In addition, mRNA secondary structures at the translation initiation factor binding sites may hinder the binding of the ribosome to the mRNA, reducing the rate of translation and amount of protein product (Nackley et al., 2006). The mRNA secondary

structures need to be unfolded before the ribosome can proceed during translation elongation. The pause in translation may reduce the rate of translation and the amount of protein product produced (Sosnick, 2008). Interestingly, the ribosome stalling may also provide additional waiting time where the growing polypeptide may fold into a different conformation (Tsai et al., 2008). The change in the polypeptide folding pattern may result in a protein of altered structure and function (Hollams et al., 2002). My view is that none of these mechanisms can yet be ruled out. Under different scenarios, the mRNA secondary structures may affect translation kinetics and the protein product in different ways. The effect of a sSNP on mRNA secondary structure and the downstream consequences on translation will not only depend on the nature of the SNP, but also on the location of the SNP (Sauna and Kimchi-Sarfaty, 2011). This field requires more exploration before the effects of sSNPs on mRNA secondary structures and the downstream consequences on translation can be fully understood and predicted.

To investigate the effects of sSNPs on mRNA secondary structures, 100 bases around each sSNP was assessed. Evaluation of the minimum free-energy of the full length mRNA may have given a better indication of the stability of the global mRNA secondary structure. It is however, unlikely that a single sSNP will have a major impact on the overall stability of the mRNA secondary structure. By assessing the minimum free-energy of the entire mRNA secondary structure, many of the sSNPs with a potential functional impact may have been missed.

sSNPs may alter mRNA secondary structures through the removal or inclusion of pairing bases. The mRNA secondary structures need to be unfolded before the ribosome can proceed during translation. Changes in local mRNA secondary structures may therefore, alter the rate of translation (Johnson et al., 2011). In this study, only the local mRNA secondary structures were assessed to obtain a better indication of which sSNPs may contribute to a change in translational and protein folding kinetics.

Of the 90,102 sSNPs identified, 4,029 (4.5%) were predicted to potentially cause an extreme change in MFE. It is still unclear how or if these sSNPs may alter translation kinetics. It is, however, possible that these sSNPs may alter the amount or structure and function of the protein product, through changes in local mRNA secondary structures and may be worth investigating further.

The UTRs of the mRNA sequences were not included in this study. For this reason, sSNPs that were less than 50 bases from the start or end of the translated regions of the mRNA sequences were excluded. The exclusion of these sSNPs excluded those where the 100 base mRNA sequence would have been within the UTRs of the mRNA. The UTRs play an important role in determining mRNA secondary structures ([Fairbanks and Anderson, 1999](#)). By including the UTRs, more sSNPs may be identified that contribute to changes in mRNA secondary structures.

SynSNP was designed to run on a personal computer and not on a computer cluster. Since computational resources were limited, only the MFE structures were predicted. The true mRNA secondary structure is not always the structure with the lowest free-energy, but rather within a 10% range of the MFE ([Zuker and Sankoff, 1984](#)). mRNA that is folded or partly unfolded often unfolds and refolds on the path to the thermodynamically most stable structure. Occasionally, the mRNA is kinetically trapped before reaching the thermodynamically most stable structure. For this reason, mRNA often occurs in a heterogeneous group, consisting of several different mRNA secondary structure species. It is, therefore, more accurate to include all possible suboptimal structures during the prediction of mRNA secondary structures ([Zuker et al., 1991](#)). There is, however, merit to predicting MFE structures, as it saves on computational resources and provides a rough estimate of the most likely mRNA secondary structure. The prediction of all possible suboptimal mRNA secondary structures provides a better estimate of the true mRNA secondary structures. Thus, an extension of this project could be to determine which sSNPs may potentially alter mRNA secondary structures by comparing possible suboptimal structures. The suboptimal

structures can be predicted using RNAfold, as used in SynSNP, but with different parameters. The prediction of suboptimal structures can be run on a computer cluster that can manage the computational power required.

Of the sSNPs identified, 21,632 (24%) were predicted to potentially have an effect on translational kinetics through an extreme change in CUF and/or MFE. It is, however, still unclear how big a change in translational kinetics is required to influence the final protein abundance or structure and function. There have been a number of studies that have investigated the effects of sSNPs on mRNA secondary structures. The problem is that few of these studies use experimental evidence to explore the mechanisms by which sSNPs may alter translational kinetics, through a change in mRNA secondary structures.

It has been four years since Kimchi-Sarfaty *et al.* first showed direct experimental evidence that sSNPs could potentially alter translational and protein folding kinetics ([Kimchi-Sarfaty et al., 2007](#)). Since then most of the studies on the functional role of sSNPs have been either reviews or the further investigation of Kimchi-Sarfaty's findings through energy kinetics ([Chamary et al., 2006](#), [Sauna et al., 2007](#), [Tsai et al., 2008](#), [Fung and Gottesman, 2009](#), [Sauna and Kimchi-Sarfaty, 2011](#)). These resources aid in our understanding of the potential functional impact of sSNPs, however, far too many studies on sSNPs just mention that sSNPs may have a potential functional impact, instead of investigating the molecular effects of sSNPs through experimental evidence.

Besides the C3435T sSNP in the *MDR1* gene, no other sSNP has been found to alter translational and protein folding kinetics through a change in CUF. I do not think that this is because no other sSNP alters protein structure and function in a similar manner. The reason is more likely to be related to the time and financial constraints of doing the laboratory-based experiments and the biases towards the idea that most sSNPs are silent and do not warrant further attention.

4.3. sSNPs predicted to potentially cause an extreme change in the maximum-entropy score of splice-sites

ME provides an indication of the signal strength of a splice-site and the likelihood that splicing will occur at the splice-site. sSNPs that result in an increase in ME may increase the likelihood of splicing at the splice-site. sSNPs that result in a decrease in ME may reduce the likelihood of splicing at the splice-site. Of the 90,102 identified sSNPs, 256 (0.3%) were predicted to potentially result in an extreme change in ME. These sSNPs may potentially alter the normal splicing pattern.

The splice-site with the highest splicing signal strength is not always the splice-site that is used. Sometimes a splice-site with weaker signal strength will be used, as opposed to a splice-site with stronger signal strength, due to its proximity to a splicing enhancer. sSNPs that increase the signal strength of splice-sites will not necessarily determine that the splice-site will be used. The splice-site might only be used if it is in close-proximity to a splicing enhancer. Similarly, sSNPs that reduce the signal strength of a splice-site will not necessarily prevent the splice-site from being used, unless the splice-site is in close-proximity to a splicing silencer. To fully understand splicing it is important to assess the core splice-sites as well as the regulatory elements.

To identify sSNPs that may contribute to aberrant splicing, only the sSNPs within the core splice-sites were assessed. As only splice-sites of known exons were included, it was known that splicing occurred at these splice-sites. It could be assumed that these splice-sites were likely to be in close-proximity to splicing enhancers. It is possible that a sSNP that enhances the signal strength of a splice-site will still determine that the splice-site is used. The sSNPs predicted to potentially cause an extreme increase in ME may have no molecular effect. sSNPs that increase ME may, however, have an effect on splicing when there are two or more alternatively used splice-sites in close-proximity to a splicing enhancer. A sSNP that results in an increase in the signal strength of one of the alternatively used splice-sites may

determine that the splice-site is used more frequently. The increased usage of this splice-site may influence the abundance of the protein isoforms. sSNPs that result in a decrease in ME may reduce the likelihood of splicing, regardless of the splice-site's proximity to a splicing enhancer. sSNPs may reduce the signal strength of a splice-site to a point where the splice-site is no longer recognized by the splicing machinery. A sSNP may also alter a splice-site so that it is recognized by a splicing silencer rather than a splicing enhancer. The assessment of splicing regulatory regions were not included in this study, however, the sSNPs predicted to potentially cause an extreme change in the ME of the core splice-sites may contribute to aberrant splicing.

Alternative splicing is a highly complex process. As alternative splicing is dependent on the interaction of many different components, there are several ways in which sSNPs may contribute to aberrant splicing. sSNPs may alter the core splice-sites, splicing enhancers or splicing silencers ([Chamary et al., 2006](#), [Parmley and Hurst, 2007](#)). Splicing regulatory regions may be computationally predicted, for instance with the use of ESEfinder. One of the next steps in this project could be to determine which of the sSNPs predicted to potentially alter the splicing signal strength of splice-sites are within close-proximity to splicing regulatory regions. It may also be useful to determine whether any of the identified sSNPs may potentially alter any of the splicing regulatory regions.

4.4. sSNPs and disease associations

Of the non-redundant 90,102 sSNPs identified, 21,086 (23.4%) were predicted to have a potential functional impact through one or more of the three molecular mechanisms investigated. Of the 21,086 sSNPs predicted to have a potential functional impact, 4,057 (19.2%) were within genes known to be associated with disease in OMIM (**Fig.4.1**). Most of the disease genes were related to complex diseases, such as various forms of mental retardation and cancers. OMIM is one of the most well established databases for gene and genetic phenotype data, however, the number of complex disorders that are cataloged in OMIM is still limited ([Amberger et al., 2011](#)). The effect of sSNPs may be more subtle and therefore they seldom, if ever, cause monogenic traits and are more likely to contribute to complex diseases. More sSNPs may be within disease genes that are not cataloged in OMIM, especially genes that contribute to complex diseases.

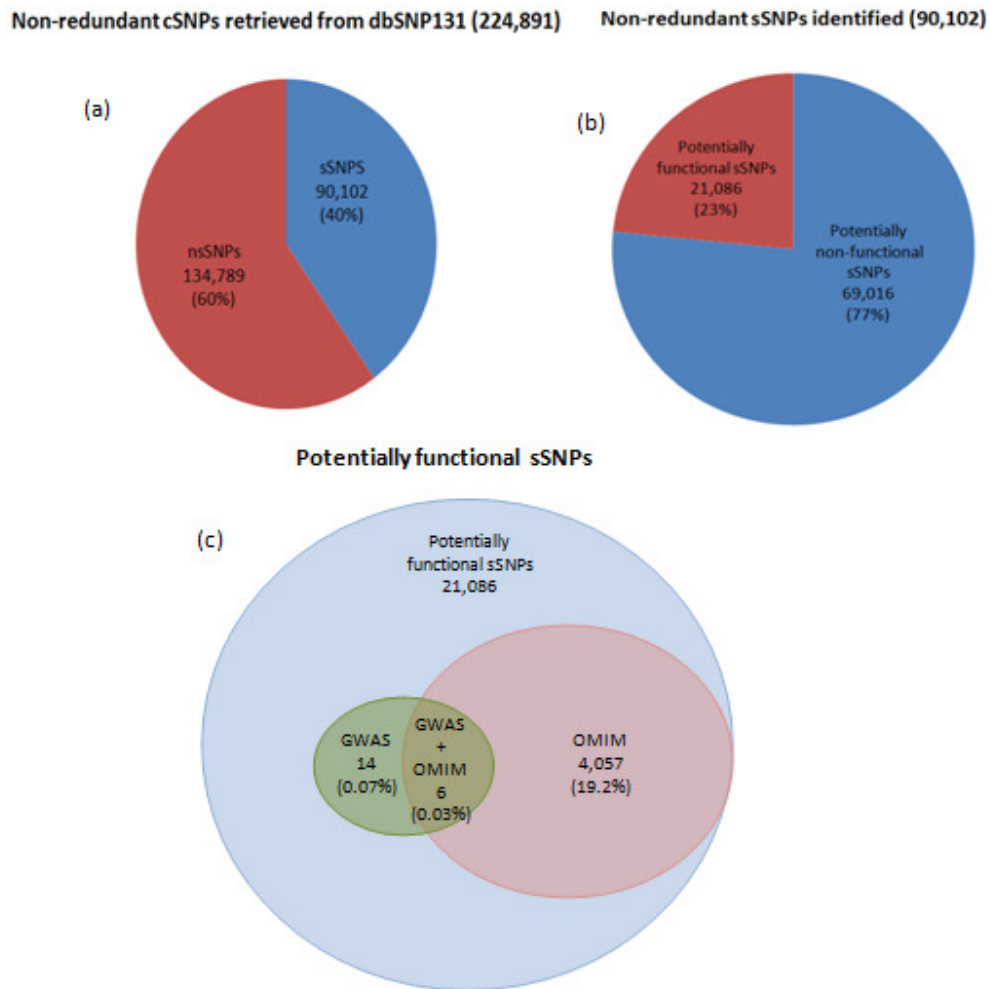


Figure 4.1: The proportion of sSNPs, identified from cSNPs, predicted to have a potential functional impact, which had previously been associated with a trait or disease in GWAS and/or occurred within genes known to be associated with disease in OMIM.

- **cSNPs** = the coding SNPs retrieved from dbSNP131
- **sSNPs** = the synonymous SNPs, identified from the cSNPs
- **GWAS** = the sSNPs predicted to have a potential functional impact, which had previously been associated with a trait or disease in GWAS
- **OMIM** = the sSNPs predicted to have a potential functional impact, which occurred within genes known to be associated with disease in OMIM

Only 14 (0.07%) of the sSNPs predicted to have a potential functional impact had previously been associated with a trait or disease in GWAS (**Fig.4.1**). SNPs for GWAS are not generally chosen for functional impact, but rather on the SNPs' frequency in the population. In general, SNPs whose minor allele frequency is more than 5% in a control population will be included in GWAS ([Hindorff et al., 2009](#)). A SNP associated with a trait or disease in GWAS may, therefore, not be the causal SNP, but rather in linkage disequilibrium with the causal SNP. The 14 sSNPs previously associated with a trait or disease in GWAS were predicted to have a potential functional impact through an extreme change in CUF and/or MFE. These sSNPs may therefore have an impact on translational and protein folding kinetics. These potentially functional sSNPs may not be the causal SNPs; however, these sSNPs are likely in close-proximity to other SNPs, due to linkage disequilibrium. If the SNPs that are in close-proximity to the sSNPs also result in a change in translation kinetics, the effect of the different SNPs may together contribute to a dramatic change in translational kinetics and protein folding. This could potentially result in a protein of altered structure and function.

Of the 4,636 SNPs associated with a trait or disease in the GWAS catalog, only 57 (1.2%) were sSNPs. As sSNPs accounted for 40% of the cSNPs retrieved from dbSNP131, it seems that sSNPs are far underrepresented in GWAS. This could imply that sSNPs are being included in GWAS, but the majority of investigated sSNPs are not associated with the assessed traits. It is, however, more likely that sSNPs are not included in GWAS nearly as frequently as nsSNPs. Genetic studies have shown that many sSNPs are associated with disease. Although sSNPs do not change the amino acids encoded, sSNPs may contribute to phenotypic variation, such as disease risk, through various molecular mechanisms. For these reasons, it is important to include all cSNPs in GWAS, including nsSNPs and sSNPs.

The ability to identify disease-associated SNPs using GWAS depends on the strengths and the limitations of the methods used ([Elbers et al., 2009](#)). Many of the genes within the human genome are of unknown biological function, or have not yet been identified. GWAS can therefore be biased towards SNPs in well-studied genes. The platforms used in GWAS are

designed to cover the entire genome, but not all of the regions within the genome are equally covered. SNPs within poorly covered regions may be missed (Elbers et al., 2009). In addition, the commonly used GWAS platforms are often biased towards populations of European descent (Hindorff et al., 2009).

At present GWAS are designed to identify common variants with a risk allele frequency above 5% and that are associated with modest effect sizes (Elbers et al., 2009). Although GWAS have helped identify many disease associations, GWAS have not been able to explain the missing heritability of complex disease. This may be because most of the disease associated SNPs identified by GWAS only confer small effect sizes. Some of the missing heritability may be due to rare variants (Baker, 2010). As technology advances more rare variants will be identified, which will improve our knowledge of the contributions of genetic variants to complex traits (Elbers et al., 2009). GWAS are using an ever increasing number of SNPs, such as the Illumina Omni5 array, which includes more than 4.3 million SNPs, with minor allele frequencies as low as 1% (<http://www.illumina.com/applications/gwas.ilmn>). It is therefore probable that more SNPs that are functional would be included and may be the causal variants associated with disease traits. Many sSNPs are associated with disease; however, the underlying molecular mechanisms involved are often not fully investigated. sSNPs identified in GWAS should be included in follow-up studies of function and mechanism.

Of the 21,086 sSNPs predicted to have a potential functional impact, only six (0.03%) were within disease genes and had previously been associated with a trait or disease in GWAS (**Fig.4.1**). Five of these sSNPs were predicted to have a potential functional impact through an extreme change in CUF and one was predicted to have a potential functional impact through an extreme change in CUF and MFE. These six sSNPs are within genes known to be associated with disease and are significantly associated with the disease. For this reason, the potential molecular mechanisms of these sSNPs in combination with the effects of other SNPs in close-proximity, due to linkage disequilibrium, may contribute to disease risk. These sSNPs may therefore be worth investigating experimentally.

4.5. Conclusion

It is possible that the majority of sSNPs would not alone have a dramatic phenotypic effect. Mutations often have an interactive or additive effect on downstream processes. It is, therefore, plausible that sSNPs may act in combination with other polymorphisms to affect downstream processes, which may ultimately result in phenotypic variation. If sSNPs could alone have a dramatic phenotypic effect more monogenetic traits would have been influenced by sSNPs and the potential phenotypic effect of sSNPs would have easily been identified. For these reasons, it is likely that more sSNPs could contribute to variations in complex multifactoral traits than monogenetic traits. This may especially be the case in complex diseases, where instead of a single causative agent a number of factors, each with a minor phenotypic effect, contribute to the disease mechanism. Some of the disease mechanisms responsible for complex disease phenotypes may be influenced at least in part by sSNPs.

The conclusion from this study emphasizes the need to examine disease processes at multiple levels. It is important not only to evaluate the potential consequences of variants based on DNA sequence information, but also to study the transcriptome and proteome in different tissues at different times in the development of the disease progression. sSNPs may have been assumed to be silent as more focus has been placed on DNA sequence information rather than transcriptome and proteome information. By combining genomic, transcriptome and proteome information sSNPs with a potential functional impact can be better identified. For instance, a sSNP that changes the rate of transcription will change the amount of mRNA transcribed as well as an equivalent change in the amount of protein translated. A sSNP that changes the rate of translation will have no affect on the amount of mRNA transcribed but could influence the amount of protein translated. A sSNP that has no effect on the amount of mRNA transcribed or the amount of protein translated, but still results in a phenotypic change, could change translational and protein folding kinetics or

alternative splicing. The change in translation and protein folding kinetics does not change the amount of protein product, but rather the structure and function of the protein.

This is the first study to assess which sSNPs may have a potential functional impact either through a change in translational kinetics or alternative splicing on a genome-wide scale. This is only a single step towards the complete understanding of the potential functional impact of sSNPs. More sSNPs may have a potential functional impact through other molecular mechanisms. sSNPs that alter transcription factor binding sites, translation initiation factors or micro-RNA binding sites may also contribute to phenotypic variation (Sauna and Kimchi-Sarfaty, 2011). It may be possible that sSNPs could alter any regulatory sequences within the genome. The change in the regulatory sequences could have downstream consequences, which could result in phenotypic variation.

The sSNPs were identified from the SNPs published within dbSNP131. In April 2010, dbSNP131 contained 23,653,737 rs SNPs. The August 2011 version of dbSNP134 contained 41,365,915 rs SNPs (http://www.ncbi.nlm.nih.gov/projects/SNP/snp_summary.cgi). In just over a year, the number of rs SNPs in dbSNP has almost doubled. The 1000 genomes project contains sequence data for 1,094 individuals (<http://www.1000genomes.org/>). HapMap phase 3 has sampled 1.6 million SNPs in 1,184 individuals from 11 populations (<http://hapmap.ncbi.nlm.nih.gov/>). The improved sequencing methods, including next generation sequencing, and the increase in the number of whole genome sequences will help identify more genetic variants, including rarer variants that may contribute to phenotypic variation. Many more genome, exome, transcriptome and proteome projects are being conducted. As variants cannot be fully characterized based on DNA sequence alone, the exome, transcriptome and proteome data will help uncover other causal variants associated with disease traits and the molecular mechanisms involved. Undoubtedly, due to the advances in technologies and the explosion of data available, more sSNPs will be identified that may have a potential functional impact.

This study was not limited to SNPs that occur in any specific population group. All of the sSNPs published within dbSNP131, which have been identified in various populations were assessed. Genetic studies are often biased towards populations of European descent. African genomes contain the most genetic variation ([Schuster et al., 2010](#)). As more African genomes become available, more SNPs that are novel will be identified. The comparison of variants in different population groups may further uncover rare synonymous variants that may have a potential functional impact.

Functionally important regions are often conserved ([Fairbanks and Anderson, 1999](#)). Cross-species homology would have been another way of identifying potentially functional sSNPs. It may also have been useful to determine which of the sSNPs predicted to have a potential functional impact were within highly conserved regions. sSNPs that were predicted to have a potential functional impact and were within conserved regions would be well worth investigating experimentally to assess their potential functional impact.

The objective of this project was to do an exploratory study to determine the proportion of sSNPs within the human genome that are more likely to have a functional impact. The sSNPs predicted to have a potential functional impact were assessed to determine whether they had previously been associated with a trait or disease in GWAS and/or occurred within genes known to be associated with disease in OMIM. A number of sSNPs were predicted to have a potential functional impact. A proportion of the sSNPs predicted to have a potential functional impact had previously been associated with a trait or disease in GWAS and occurred within genes known to be associated with disease in OMIM. Although much work still needs to be done before the functional impact of sSNPs can be fully appreciated, it can be assumed that a significant proportion of sSNPs may have a functional impact. The functional impact and the role of sSNPs in disease should therefore not be underestimated or ignored.

APPENDIX A

```
*****
*                                     *
*                               ----- *
*                               SynSNP   *
*                               ----- *
*                                     *
* Created by Lee-Ann Wood as (Bioinformatics/Human Genetics) MSc project *
* at the university of the Witwatersrand (2011)                          *
*                                     *
* Project title:                                                            *
* -----                                                                    *
* Computational identification of synonymous SNPs in the human genome    *
* and their potential role in disease                                      *
*                                     *
* Direct any queries to sharma.leeann@gmail.com                           *
*                                     *
*****
```

SynSNP is designed to identify sSNPs within the human genome that may have a potential functional affects either through:

1. a change in codon-usage frequency
(translational kinetics)
2. a change in local mRNA secondary structure and stability
(rate of mRNA degradation or translation)
3. disruption of splice-site motifs
(abnormal splicing pattern)

and to determine which of the sSNPs predicted to have a potential functional impact have previously been associated with a trait or disease in GWAS and/or are within genes known to be associated with disease in OMIM.

-> All files required are downloaded from Galaxy (<http://main.g2.bx.psu.edu/>)

```
*****
* 1. Identify all sSNPs published within dbSNP                                     *
*****
```

1.1. Retrieve all SNPs published within dbSNP

Tools > Get data > UCSC main table browser

```
- clade:      Mammal
- genome:     Human
- assembly:   Feb.2009(GRCh37/hg19)
- group:      variation and Repeats
- track:      SNP(131)
- table:      SNP(131)
- region:     genome
```

```

- filter:
  - bin is: ignored
  - chrom: does
  - ChromStart: ignored
  - ChromEnd: ignored
  - name: does
  - score is: ignored
  - strand: does
  - refNCBI: does
  - refUCSC: does
  - observed: does
  - molType: does, match: genomic
  - class: does, match: single
  - valid: does, match: *
  - avHet is: ignored
  - avHetSE is: ignored
  - func: does, include: *
  - locType: does, match: *
  - weight is: ignored

- output format: all fields from selected table

> get output

> Once Galaxy has retrieved all the SNPs click on the pencil icon
  (top right of history item) to edit the file attributes
  - save as interval

```

1.2. Retrieve all coding SNPs published within dbSNP

```
=====
```

```

Tools > Get data > UCSC main table browser
-----

- clade:      Mammal
- genome:     Human
- assembly:   Feb.2009(GRCh37/hg19)
- group:      variation and Repeats
- track:      SNP(131)
- table:      snp131CodingDbSNP
- region:     genome
- filter:
  - bin is: ignored
  - chrom: does
  - ChromStart: ignored
  - ChromEnd: ignored
  - name: does
  - transcript: does
  - frame: does
  - alleleCount is: ignored
  - funcCodes: does
  - alleles: does
  - codons: does
  - peptides: does

- output format: all fields from selected table

```

- > get output
- > Once Galaxy has retrieved all the SNPs click on the pencil icon (top right of history item) to edit the file attributes
 - save as interval

1.3. Join SNP file to coding SNP file

Tools > Operate on Genomic Intervals > Join the intervals of two

datasets side-by-side

- Join: "SNP file"
- with: "Coding SNP file"
- with min overlap: 1 (bp)
- return: Only records that are joined (INNER JOIN)

> Execute

```
.....
. SNPs joined to coding SNPs from Galaxy file structure .
. ===== .
. - First two lines = Headings .
. - Rec[0] = bin number .
. - Rec[1] = SNP chromosome .
. - Rec[2] = SNP chromosome start position .
. - Rec[3] = SNP chromosome end position .
. - Rec[4] = SNP name (rs number) .
. - Rec[5] = score .
. - Rec[6] = SNP strand .
. - Rec[7] = refNCBI (reference allele) .
. - Rec[8] = refUCSC (reference allele) .
. - Rec[9] = observed allele (e.g. A/G) .
. - Rec[10] = SNP molecular type (e.g. genomic) .
. - Rec[11] = SNP class (e.g. single) .
. - Rec[12] = valid .
. - Rec[13] = average heterozygosity .
. - Rec[14] = average heterozygosity standard error .
. - Rec[15] = SNP function (e.g. coding-synon) .
. - Rec[16] = locus type .
. - Rec[17] = weight .
. - Rec[18] = bin number .
. - Rec[19] = coding SNP chromosome .
. - Rec[20] = coding SNP chromosome start position .
. - Rec[21] = coding SNP chromosome end position .
. - Rec[22] = coding SNP name (rs number) .
. - Rec[23] = transcript name (accession number) .
. - Rec[24] = coding SNP frame .
. - Rec[25] = number of alleles .
. - Rec[26] = coding SNP function codes .
. - Rec[27] = alleles (as in codons) .
```

```
. - Rec[28] = codons .
. - Rec[29] = amino acids encoded .
.....
```

1.4.1. Retrieve all genes known to be associated with disease in OMIM

```
=====

Tools > Get Data > UCSC Main table browser
-----

- clade:           Mammal
- genome:          Human
- assembly:        Feb. 2009 (GRCh37/hg19)
- group:           Phenotype and Disease Association
- track:           OMIM Genes
- table:           omimMorbidityMap
- region:          genome
- output format:   selected fields from primary and related tables
- Send output to:  Galaxy

> get output

- Select Fields from hg19.knownGene: - description
                                      - omimID
- Linked Tables:                    - hg19.kgXref

> Allow selection from table

- Select Fields from hg19.omimMorbidityMap: - description
                                              - omimID
- hg19.kgXref fields:                    - refseq.RefSeq ID

> done with selections
```

1.4.2. Remove additional commas (,) from file retrieved from OMIM

```
=====

Tools > Text Manipulation > Condense consecutive characters
-----

- Condense all consecutive: Commas
- in this Query: "OMIM file"

> Execute
```

1.4.3. Retrieve all genes within OMIM

```
=====

Tools > Get Data > UCSC Main table browser
-----

- clade:           Mammal
- genome:          Human
- assembly:        Feb. 2009 (GRCh37/hg19)
```

- group: Phenotype and Disease Association
- track: OMIM Genes
- table: omimGene
- region: genome
- output format: selected fields from primary and related tables
- Send output to: Galaxy

> get output

- Select Fields from hg19.knownGene:
 - description
 - omimID
- Linked Tables:
 - hg19.kgXref

> Allow selection from table

- Select Fields from hg19.omimMorbidityMap:
 - description
 - omimId
- hg19.kgXref fields:
 - refseq.RefSeq ID

> done with selections

1.4.4. Join the two OMIM files

Tools > Join, Subtract and Group > Join two Datasets side by side on a specified field

- Join: "Disease gene file"
- using: c3
- with: "Gene file"
- and column: c5
- Keep lines of the first input that do not join with second input: No
- Keep lines of first input that are incomplete: : No
- Fill empty columns: : No

> Execute

```
.....
. OMIM from Galaxy file structure .
. ===== .
. - Rec[0] = disease .
. - Rec[1] = gene name .
. - Rec[2] = MIM accession number .
. - Rec[3] = position of gene in chromosome .
. - Rec[4] = score .
. - Rec[5] = chromosome .
. - Rec[6] = gene start position .
. - Rec[7] = gene end position .
. - Rec[8] = MIM accession number .
.....
```

1.5. Retrieve all SNPs known to be associated with disease from GWAS

=====

Tools > Get Data > UCSC Main table browser

```

- clade:           Mammal
- genome:          Human
- assembly:        Feb. 2009 (GRCh37/hg19)
- group:           Phenotype and Disease Association
- track:           GWAS
- table:           gwasCatalog
- region:          genome
- output format:   all fields from selected table
- Send output to:  Galaxy

```

> get output

```

.....
. GWAS from Galaxy file structure .
. ===== .
. - First line = Headings .
. - Rec[0] = bin number .
. - Rec[1] = SNP chromosome .
. - Rec[2] = SNP chromosome start position .
. - Rec[3] = SNP chromosome end position .
. - Rec[4] = SNP name (rs number) .
. - Rec[5] = PubMed ID .
. - Rec[6] = author .
. - Rec[7] = publication date .
. - Rec[8] = journal .
. - Rec[9] = title .
. - Rec[10] = trait .
. - Rec[11] = initial sample .
. - Rec[12] = replicate sample .
. - Rec[13] = chromosome region .
. - Rec[14] = genes .
. - Rec[15] = risk alleles .
. - Rec[16] = risk allele frequency .
. - Rec[17] = p-value .
. - Rec[18] = p-value desc .
. - Rec[19] = orOrBeta .
. - Rec[20] = 95% confidence-interval .
. - Rec[21] = platform .
. - Rec[22] = cnv .
.....

```

1.6. Identify sSNPs from coding SNPs

=====

```

> Run SynSNP.py
> from the menu chose option 1, which will:

```

1. Remove coding SNP records with errors
 - `SSNPs.Remove_Errors()`
 2. Remove redundant coding SNP records
 - `SSNPs.Remove_Redundancies()`
 3. Identify the synonymous SNPs (sSNPs) from the coding SNPs
 - `SSNPs.Identify_sSNPs()`
 4. Identify sSNPs previously associated with a trait/disease in GWAS
 - `SSNPs.GWAS()`
 5. Identify sSNPs within genes known to be associated with disease in OMIM
 - `SSNPs.OMIM()`
 6. Identify sSNPs previously associated with a trait/disease in GWAS and within genes known to be associated with disease in OMIM.
 - `General.GWAS_And_OMIM()`
 7. Creates a separate record for each synonymous single nucleotide substitution from the sSNP records so that the potential functional impact of each sSNP allele may be investigated
 - `SSNPs.SSNS()`
 8. Separates the synonymous single nucleotide substitutions into separate file for each chromosome so that the synonymous single nucleotide substitution records may be joined to other records later in a more efficient manner
 - `SSNPs.Create_Chrom_Files()`
- The log file is saved in
`"Home"\SynSNP\Results\1_sSNPs\Log_sSNPs.txt`
- The sSNP file is saved in
`"Home"\SynSNP\Results\1_sSNPs\3_2_Synonymous_SNPs.txt`

```

.....
. sSNPs file structure .
. ===== .
. - Rec[0] = sSNP CHROMOSOMES .
. - Rec[1] = sSNP START position on the chromosome .
. - Rec[2] = sSNP END position on the chromosome .
. - Rec[3] = sSNP NAME (rs number) .
. - Rec[4] = sSNP FUNCTION CODE .
. - Rec[5] = sSNP FUNCTION .
. - Rec[6] = transcript STRAND .
. - Rec[7] = sSNP REFERENCE ALLELES as on transcript strand .
. - Rec[8] = sSNP ALLELE as on transcript strand .
. - Rec[9] = CODONS as on transcript strand .
. - Rec[10] = sSNP FRAME (i.e. position in codons) .
. - Rec[11] = synonymous AMINO ACIDS encoded .
.....

.....
. Synonymous single nucleotide substitution file structure .
. ===== .
. - Rec[0] = sSNS CHROMOSOMES .

```

```

. - Rec[1] = sSNS START position on the chromosome .
. - Rec[2] = sSNS END position on the chromosome .
. - Rec[3] = sSNS NAME (rs number) .
. - Rec[4] = sSNS FUNCTION CODE .
. - Rec[5] = sSNP FUNCTION .
. - Rec[6] = transcript STRAND .
. - Rec[7] = REFERENCE ALLELE .
. - Rec[8] = VARIANT ALLELE .
. - Rec[9] = REFERENCE CODON .
. - Rec[10] = VARIANT CODON .
. - Rec[11] = sSNS FRAME in codon .
. - Rec[12] = REFERENCE AMINO ACID .
. - Rec[13] = VARIANT AMINO ACID .
.....

```

```

*****
* 2. Identify sSNPs that may potentially alter codon-usage frequency *
*****

```

```

> Run SynSNP.py
> from the menu chose option 2, which will:

1. Calculate the change in codon-usage frequency caused by each
   synonymous single nucleotide substitution
   - Codons.Codon_Usage_Change()
2. Determine the cut-off values chosen to represent an extreme
   change in codon-usage frequency
   - General.Cut_Off_Value()
3. Determine which synonymous single nucleotide substitutions
   result in an extreme change in codon-usage frequency based
   on the chosen cut-off values
   - Codon.Extreme_CUF()
4. Determine if any of the synonymous single nucleotide
   substitutions predicted here to potentially result in an
   extreme change in codon-usage frequency have previously been
   associated with a trait or disease in GWAS
   - General.GWAS_Or_OMIM()
5. Determine if any of the synonymous single nucleotide
   substitutions predicted here to potentially result in an
   extreme change in codon-usage frequency are within genes
   known to be associated with disease in OMIM
   - General.GWAS_Or_OMIM()
6. Determine whether any of the sSNS predicted too potentially
   result in an extreme change in codon-usage frequency have
   previously been associated with a trait or disease in GWAS
   and are within genes known to be associated with disease
   in OMIM
   - General.GWAS_And_OMIM()

- The log file is saved in
  "Home"\SynSNP\Results\2_Codon_usage\Log_Codons.txt

- The sSNP that potentially result in an extreme change in codon-usage
  file is saved in :

```

```
"Home"\SynSNP\Results\2_Codon_usage\2_1_Extreme_DECREASE_CUF.txt
"Home"\SynSNP\Results\2_Codon_usage\2_2_Extreme_INCREASE_CUF.txt
```

```
.....
. Change in codon-usage frequency file structure .
. ===== .
. - Rec[0] = sSNS CHROMOSOMES .
. - Rec[1] = sSNS START position on the chromosome .
. - Rec[2] = sSNS END position on the chromosome .
. - Rec[3] = sSNS NAME (rs number) .
. - Rec[4] = sSNS FUNCTION CODE .
. - Rec[5] = sSNP FUNCTION .
. - Rec[6] = transcript STRAND .
. - Rec[7] = REFERENCE ALLELE .
. - Rec[8] = VARIANT ALLELE .
. - Rec[9] = REFERENCE CODON .
. - Rec[10] = VARIANT CODON .
. - Rec[11] = sSNS FRAME in codon .
. - Rec[12] = REFERENCE CODON .
. - Rec[13] = VARIANT CODON .
. - Rec[14] = sSNS frame .
. - Rec[15] = REFERENCE AMINO ACID .
. - Rec[16] = VARIANT AMINO ACID .
. - Rec[17] = REFERENCE CODON-USAGE FREQUENCY .
. - Rec[18] = VARIANT CODON-USAGE FREQUENCY .
. - Rec[19] = CHANGE IN CODON-USAGE FREQUENCY .
.....
```

```
*****
* 3. Identify sSNPs that may potentially alter mRNA secondary structure *
* and stability *
*****
```

3.1. Retrieve the coordinates of the coding sequences with introns

```
=====
```

```
Tools > Get Data > UCSC Main table browser
```

```
-----
```

```
- clade:           Mammal
- genome:         Human
- assembly:       Feb. 2009 (GRCh37/hg19)
- group:          mRNA and EST Tracks
- track:          Human mRNAs
- table:          RefSeq Genes (refGene)
- region:         genome
- output format:  all fields from selected file
- Send output to: Galaxy
```

```
> get output
```

```
- Create one BED record per: whole Gene
```

3.2. Retrieve the sequences of the coding sequences with introns

=====

Tools > Get Data > UCSC Main table browser

```
- clade:           Mammal
- genome:         Human
- assembly:       Feb. 2009 (GRCh37/hg19)
- group:          mRNA and EST Tracks
- track:          Human mRNAs
- table:          RefSeq Genes (refGene)
- region:         genome
- output format:  sequences
- Send output to: Galaxy
```

> get output

```
- Select sequence type for RefSeq Genes: genomic
```

```
Sequence Retrieval Region Options:  - CDS Exons
                                     - Introns
                                     - One FAST record per gene
Sequence Formatting Options:        - Exons in upper case,
                                     everything else in lower case
```

3.3.1. Convert the sequences of the coding sequences with introns from FASTA format to tabular format

=====

Tools > FASTA-to-Tabular converter

```
- Convert these sequences: "Sequence file"
- How many columns to divide string into?: 2
- How many title characters to keep?: 0
```

3.3.2. Remove excess columns

=====

Tools > Text Manipulation > Cut columns from table

```
- Cut columns: c1, c3
- Delimited by: Tab
- From: "Tabular sequence file"
```

3.3.3. Trim heading to just NM number

=====

Tools > Text Manipulation > Trim leading or trailing characters

```
- this dataset:      "Tabular sequence file"
```

- Trim this column only: 1
- Trim from the beginning to this position: 14
- Remove everything from this position to the end: 0
- Is input dataset in fastq format?: No

3.3.4. Join co-ordinates and sequences of the coding sequences with introns

```
=====
Tools > Join, Subtract and Group > Join two Datasets side by side on a
-----
specified field
-----

- Join: "Co-ordinates file"
- using column:      c4
- with: "Sequence file"
- and column: c1
- Keep lines of first input that do not join with second input: No
- Keep lines of first input that are incomplete: No
- Keep lines of first input that are incomplete: No
- Fill empty columns: No

> Execute
```

3.3.5 Remove excess columns

```
=====
Tools > Text Manipulation > Cut columns from table
-----

- Cut columns: c1,c2,c3,c4,c6,c7,c8,c14
- Delimintaed by: Tab
- From: "Coding sequences with introns file"

.....
. Coding sequences with introns file structure .
. ===== .
. - Rec[16] = chromosome .
. - Rec[17] = sequence start .
. - Rec[18] = sequences end .
. - Rec[19] = Sequence name (NM number) .
. - Rec[20] = DNA strand .
. - Rec[21] = sequence start position .
. - Rec[22] = sequence end position .
. - Rec[23] = sequence .
.....
```

3.4. Identify sSNPs that potentially result in an extreme change in minimum-free energy of mRNA:

```
=====

> Run SynSNP.py
> from the menu chose option 3, which will:
```

1. Remove records with errors
 - mRNA.Remove_Errors()
 2. Remove redundant records
 - General.Remove_Redundancies()
 3. Separate the records into separate files based on their chromosomes
 - General.Create_Files()
 4. Identify synonymous single nucleotide substitutions in the coding sequences with introns
 - General.Join_Files()
 5. Ensures that the reference allele is in the coding sequences with introns
 - General.Check_Alleles()
 6. Create the reference and variant 100 base mRNA sequences
 - mRNA.Ref_And_Var()
 7. Create the Fasta file required for RNAfold
 - mRNA.Fasta()
 8. Run RNA fold to calculate the minimum-free energy of the reference and variant mRNA secondary structures
 - mRNA.RNAfold()
 9. Calculate the change in the minimum-free energy caused by each synonymous single nucleotide substitution
 - mRNA.Change_In_MFE()
 10. Identify single nucleotide substitutions that result in an extreme change in the minimum-free energy
 - mRNA.Extreme_MFE()
 11. Determine whether the synonymous single nucleotide substitutions predicted to potentially cause an extreme change in the minimum-free energy of the local mRNA secondary structures have previously been associated with a trait or disease in GWAS and/or occur within genes known to be associated with disease in OMIM
 - mRNA.MFE_In_Disease()
- The log file is saved in
 "Home"\SynSNP\Results\3_mRNA\Log_mRNA.txt
- The sSNP that potentially result in an extreme change in the minimum-free energy of local mRNA secondary structures file is saved in :
 "Home"\SynSNP\Results\3_mRNA\8_1_Extreme_DECREASE_MFE.txt
 "Home"\SynSNP\Results\3_mRNA\8_2_Extreme_INCREASE_MFE.txt

```

.....
. sSNS in mRNA with change in MFE file structure .
. ===== .
. - Rec[0] = sSNS chromosome .
. - Rec[1] = sSNS start position (genomic) .
. - Rec[2] = sSNS end position (genomic) .
. - Rec[3] = sSNS name (rs number) .
. - Rec[4] = sSNS function code .
. - Rec[5] = sSNS function .
. - Rec[6] = strand .

```

```
. - Rec[7] = reference allele .
. - Rec[8] = variant allele .
. - Rec[9] = CDS start position .
. - Rec[10] = CDS end position .
. - Rec[11] = CDS name (NM number) .
. - Rec[12] = sSNS position in CDS .
. - Rec[13] = reference 100 base mRNA sequence .
. - Rec[14] = variant 100 base mRNA sequence .
. - Rec[15] = reference 100 base mRNA secondary structure .
. - Rec[16] = variant 100 base mRNA secondary structure .
. - Rec[17] = reference mRNA minimum-free energy .
. - Rec[18] = variant mRNA minimum-free energy .
. - Rec[19] = change in minimum free energy .
.....
```

```
*****
* 4. Identify sSNPs that may potentially disrupt or increase splice-site *
* signal strength *
*****
```

4.1. Retrieve exon (plus 20 bases at each end) coordinates

```
=====
```

UCSC main table browser

```
- clade:      Mammal
- genome:    Human
- assembly:  Feb.2009(GRCh37/hg19)
- group:     Genes and Gene Prediction Tracks
- track:     RefSeq Genes
- table:     refGene
- region:    genome
- output format: BED-browser extensible data
- send output to: Galaxy
```

> get output

```
- Create one BED record per: Exons plus 20 bases at each end
```

4.2. Retrieve sequences for exon (plus 20 bases at each end) coordinates

```
=====
```

Tools > Fetch sequences > Extract Genomic DNA using coordinates from

assembled/unassembled genomes

```
- Fetch sequences for intervals in: "Coding exons (plus 20 bases at each
                                     end) coordinates file"
- Interpret features when possible: No
- Source for Genomic Data: Locally cached
- Output data type: Interval
```

> Execute

4.3. Identify sSNPs that potentially result in an extreme change in maximum-entropy of splice-sites

=====

```
> Run SynSNP.py
> from the menu chose option 4, which will:

1. Remove records with errors
   - SpliceSites.Remove_Errors()
2. Remove redundant exons
   - General.Remove_Redundancies()
3. Create separate chromosome files
   - General.Create_Files()
4. Identify synonymous single nucleotide substitution within
   exons
   - General.Join_Files()
5. Ensure that the reference alleles are in the exon
   - General.Check_Alleles()
6. Identify synonymous single nucleotide substitutions in the
   3' acceptor and 5' donor splice-sites
   - SpliceSites.In_Splice_Sites()
7. Runs MaxEntScan to calculate the maximum entropy score of
   each reference and variant splice-site motif
   -SpliceSites.MaxEntScan()
8. Determines which synonymous single nucleotide substitutions
   result in an extreme change in maximum entropy
   - General.Extreme_ME()
6. Determines which of the synonymous single nucleotide
   substitutions predicted to potentially cause an extreme
   change in the maximum entropy score of splice-sites have
   previously been associated with a trait or disease in GWAS
   and/or occur within genes known to be associated with
   disease in OMIM
   -SpliceSites.Extreme_ME_In_Disease()

- The log file is saved in
  "Home"\SynSNP\Results\4_Splice_sites\Log_splice_sites.txt

- The sSNP that potentially result in an extreme change in the maximum
  entropy score of splice-sites file is saved in :
  "Home"\SynSNP\Results\4_Splice_sites\9_1_Acceptor_EXTREME_DECREAS.txt
  "Home"\SynSNP\Results\4_Splice_sites\9_2_Acceptor_EXTREME_INCREASE.txt
  "Home"\SynSNP\Results\4_Splice_sites\9_3_Donor_EXTREME_DECREAS.txt
  "Home"\SynSNP\Results\4_Splice_sites\9_4_Donor_EXTREME_INCREASE.txt

.....
. ssNS in exons file structure .
. ===== .
. - Rec[0] = CHROMOSOME .
. - Rec[1] = ssNS START position .
. - Rec[2] = ssNS END position .
. - Rec[3] = ssNS NAME (rs number) .
```

```
. - Rec[4] = sSNS FUNCTION CODE .
. - Rec[5] = sSNS FUNCTION .
. - Rec[6] = STRAND .
. - Rec[7] = REFERENCE ALLELE .
. - Rec[8] = VARIANT ALLELE .
. - Rec[9] = EXON START position (minus 20 bases) .
. - Rec[10] = EXON END position (plus 20 bases) .
. - Rec[11] = EXON NAME (NM number +) .
. - Rec[12] = exon SEQUENCE .
.....
```

```
.....
. sSNS in that potentially disrupt splice-sites file structure .
. ===== .
. - Rec[0] = CHROMOSOME .
. - Rec[1] = sSNS START position .
. - Rec[2] = sSNS END position .
. - Rec[3] = sSNS NAME (rs number) .
. - Rec[4] = sSNS FUNCTION CODE .
. - Rec[5] = sSNS FUNCTION .
. - Rec[6] = STRAND .
. - Rec[7] = REFERENCE ALLELE .
. - Rec[8] = VARIANT ALLELE .
. - Rec[9] = EXON START position (minus 20 bases) .
. - Rec[10] = EXON END position (plus 20 bases) .
. - Rec[11] = EXON NAME (NM number +) .
. - Rec[12] = exon SEQUENCE .
. - Rec[13] = REFERENCE SPLICE-SITE SEQUENCE .
. - Rec[14] = VARIANT SPLICE-SITE SEQUENCE .
. - Rec[15] = sSNS position in exonic part of splice-site .
. - Rec[16] = REFERENCE MAXENT SCORE .
. - Rec[17] = VARIANT MAXENT SCORE .
. - Rec[18] = CHANGE IN MAXENT SCORE .
.....
```

```
*****
* 5. Identify the sSNPs predicted to have a potential functional impact from *
* the sSNPs predicted to have a potential functional impact *
*****
```

```
> Run SynSNP.py
> from the menu chose option 5, which will:
    1. Identify all of the sSNPs from the sSNS predicted to have a
       potential functional impact
       - Functional.Count_sSNPs()
    2. Count how many sSNPs may have a potential functional impact
       through an extreme change in CUF, MFE, ME, CUF+MFE, CUF+ME,
       MFE+ME, CUF+MFE+ME
       - Functional.Mechanisms()

- The log file is saved in
  "Home"\SynSNP\Results\5_Functional_sSNPs\Log_Functional_sSNPs.txt

- The sSNP predicted to have a potential functional impact
```

file is saved in :
 "Home"\SynSNP\Results\5_Functional_sSNPs\4_sSNPs_Functional.txt

```
.....
. Potentially functional sSNP file structure .
. ===== .
. - Rec[0] = Chromosome .
. - Rec[1] = sSNP position in chromosome .
. - Rec[2] = DNA strand .
. - Rec[3] = Mechanisms .
.....
```

```
*****
* 6. Determine which sSNPs predicted to have a potentially functional may *
* potentially be involved in disease *
*****
```

```
> Run SynSNP.py
> from the menu chose option 6, which will:
    1. Identify which of the sSNPs predicted to have a potential
       functional impact have previously been associated with a
       trait or disease in GWAS OR occur within genes known to be
       associated with disease in OMIM
       - Disease.GWAS_Or_OMIM()
    2. Identify which of the sSNPs predicted to have a potential
       functional impact have previously been associated with a
       trait or disease in GWAS AND occur within genes known to be
       associated with disease in OMIM
       - Disease.GWAS_And_OMIM()

- The log file is saved in
  "Home"\SynSNP\Results\6_Disease\Log_Disease.txt

- The sSNP predicted to have a potential functional impact, which have
  previously been associated with a trait or disease in GWAS and occur
  within genes known to be associated with disease in OMIM
  file is saved in :
  "Home"\SynSNP\Results\6_Disease\3_Functional_GWAS_and_OMIM.txt
```

```
.....
. Potentially functional sSNP file structure .
. ===== .
. - Rec[0] = Chromosome .
. - Rec[1] = sSNP position in chromosome .
. - Rec[2] = Mechanisms .
. - Rec[3] = trait or disease from GWAS .
. - Rec[4] = gene associated with disease in OMIM .
.....
```

```
*****
                        End of SynSNP manual
*****
```

APPENDIX B

```
#####
#                               SynSNP.py                               #
#####

print("="*79)
print("SynSNP MENU".center(79))
print("="*79)
print("\t1. sSNPs from coding SNPs")
print()
print("\t2. sSNS that potentially alter CODON-USAGE frequency")
print()
print("\t3. sSNS that potentially alter mRNA SECONDARY STRUCTURE")
print()
print("\t4. sSNS that potentially disrupt SPLICE-SITES")
print()
print("\t5. Potentially FUNCTIONAL sSNPs")
print()
print("\t6. sSNPs predicted to have a potentially functional impact")
print("\t\t within genes known to be associated with DISEASE")
print("="*79)
print()

Choice = input("Enter choice as list (e.g. 1234) or 0 to exit: ")
print()

if Choice != "0":

    import os
    import sys
    sys.path.append(os.getcwd() + "\\Functions")
    import General

    HomeDir = os.getcwd()

    if "1" in Choice:
        print("Identifying sSNPs from cSNPs...")
        print("=====")
        print()
        import sSNPs
        sSNPs.Create_Chrom_Files(HomeDir)
    if "2" in Choice:
        print("Identifying sSNS that potentially alter CODON-USAGE frequency...")
        print("=====")
```



```

def Write_File(File,Record,Allele,Codon,AminoAcid):

    Compl = {"A":"T","T":"A","C":"G","G":"C"}
    DNA = ["A","C","G","T"]
    ComplStr = {"-":"+", "+":"-"}

    #-----#
    # Correct reference allele (refNCBI) and strand according to codons      #
    #-----#

    if Record[6] == "+":
        if Record[7] in Record[27]:
            refNCBI = Record[7]
            Strand = "+"
        elif Compl[Record[7]] in Record[27]:
            refNCBI = Compl[Record[7]]
            Strand = "-"
    elif Record[6] == "-":
        if Record[7] in Record[27]:
            refNCBI = Record[7]
            Strand = "+"
        elif Compl[Record[7]] in Record[27]:
            refNCBI = Compl[Record[7]]
            Strand = "-"

    #-----#
    # Write the records without errors to file                                #
    #-----#

    for i in range(1,5):          # SNP chrom, SNP start, SNP end, SNP name
        File.write(Record[i])
        File.write("\t")
    File.write(Record[26])        # Function code
    File.write("\t")
    File.write(Record[15])        # Function
    File.write("\t")
    File.write(Strand)            # Codon/transcript strand
    File.write("\t")
    File.write(refNCBI)           # Reference allele as on transcript strand
    File.write("\t")
    File.write(Allele)            # SNP alleles as on transcript strand
    File.write("\t")
    File.write(Codon)             # Codons
    File.write("\t")
    File.write(Record[24])        # SNP frame
    File.write("\t")
    File.write(AminoAcid)         # Amino acids
    File.write("\n")

```

```

#####
# 1. Function: removes SNP records with errors and identifies true SNPs      #
#   - Called by: Remove_Redundancies()                                     #
#####

def Remove_Errors(LogF):

    import General
    import os
    import winsound

    winsound.Beep(400,1000)
    SNPFileName = input("Enter cSNPs file name: ")
    while not os.path.isfile(SNPFileName):
        print ("File does not exist")
        SNPFileName = input("Enter file name: ")
    Header = int(input("Enter number of comment lines: "))
    print()

    print ("Removing errors...")
    print ()

    NumRec=Err1=Err2=Err3=Err4=Err5=Err6=Err7=NoErr = 0
    DNA = ["A","C","T","G"]

    SNPFile = open(SNPFileName,"r")
    ErrFile = open("1_1_SNPs_with_errors.txt","w")
    NoErrFile = open("1_2_SNPs_with_NO_ERRORS.txt","w")

    for i in range(0,Header):
        SNPFile.readline()

    for Line in SNPFile:
        NumRec = NumRec + 1
        Rec = (Line.rstrip()).split("\t")

        #-----#
        # 1.1. No data must be missing i.e. all records must contain 30 fields #
        #-----#

        if len(Rec) != 30:
            Err1 = Err1 + 1
            ErrFile.write("1. Data missing\t")
            ErrFile.write(Line)

```

```

#-----#
# 1.2. The SNP chromosome must be the same for both files joined      #
#       - Rec[1]  = SNP chromosome                                     #
#       - Rec[19] = cSNP chromosome                                    #
#-----#

elif Rec[1] != Rec[19]:
    Err2 = Err2 + 1
    ErrFile.write("2. cSNP and SNP on diff chrom's\t")
    ErrFile.write(Line)

#-----#
# 1.3. The SNP must be one nucleotide in length                      #
#       - Rec[2]  = SNP start postion                                 #
#       - Rec[3]  = SNP end position                                  #
#-----#

elif int(Rec[3]) - int(Rec[2]) != 1:
    Err3 = Err3 + 1
    ErrFile.write("3. SNP not 1 nt in length\t")
    ErrFile.write(Line)

#-----#
# 1.4. The SNP coordinates must be the same for both files joined    #
#       - Rec[2]  = SNP start position                               #
#       - Rec[3]  = SNP end position                                  #
#       - Rec[20] = cSNP start position                              #
#       - Rec[21] = cSNP end position                                 #
#-----#

elif (Rec[2] != Rec[20]) or (Rec[3] != Rec[21]):
    Err4 = Err4 + 1
    ErrFile.write("4. SNP coordinates do not correspond\t")
    ErrFile.write(Line)

#-----#
# 1.5. The reference allele given by refNCBI must be the same as that #
#       given by refUCSC                                              #
#       - Rec[7]  = refNCBI                                           #
#       - Rec[8]  = refUCSC                                           #
#-----#

elif Rec[7] != Rec[8]:
    Err5 = Err5 + 1
    ErrFile.write("5. refNCBI and refUCSC differ\t")
    ErrFile.write(Line)

```

```

#-----#
# 1.6. Each cSNP allele must be one nucleotide, each codon 3      #
#       nucleotides and there must be one amino acid per SNP/codon #
#       - Rec[27] = alleles                                         #
#       - Rec[28] = codons                                           #
#       - Rec[29] = amino acids                                       #
#-----#

else:
    Alleles=Codons=AminoAcids = ""
    AlleleRec  = ((Rec[27].strip('')).rstrip(",")).split(",")
    CodonRec    = ((Rec[28].strip('')).rstrip(",")).split(",")
    AminoAcidRec = ((Rec[29].strip('')).rstrip(",")).split(",")
    for i in range(0,len(AlleleRec)):
        if (len(AlleleRec[i]) == 1) and (AlleleRec[i] in DNA) and (len(CodonRec[i]) ==
3):

            Alleles = Alleles + AlleleRec[i] + ","
            Codons = Codons + CodonRec[i] + ","
            AminoAcids = AminoAcids + AminoAcidRec[i] + ","
    Alleles = Alleles.rstrip(",")
    Codons = Codons.rstrip(",")
    AminoAcids = AminoAcids.rstrip(",")

    if len(Alleles) <= 1:
        Err6 = Err6 + 1
        ErrFile.write("6. Allele not one nucleotide or codon not 3 nucleotides\t")
        ErrFile.write(Line)

#-----#
# 1.7. The alleles given must be in the codons                    #
#       - Rec[24] = SNP frame                                       #
#-----#

else:
    AlleleRec = Alleles.split(",")
    CodonRec = Codons.split(",")
    Wrong = 0
    for i in range(0,len(AlleleRec)):
        if AlleleRec[i] != CodonRec[i][int(Rec[24])-1]:
            Wrong = Wrong + 1
    if Wrong != 0:
        Err7 = Err7 + 1
        ErrFile.write("7. Alleles not in codons\t")
        ErrFile.write(Line)
    else:
        NoErr = NoErr + 1
        Write_File(NoErrFile,Rec,Alleles,Codons,AminoAcids)

```

```

SNPFile.close()
ErrFile.close()
NoErrFile.close()

General.Log(LogF,"SNP records in "+SNPFileName,NumRec)
General.Log(LogF,"SNP records with errors",NumRec-NoErr)
General.Log(LogF,"1. records with data missing",Err1)
General.Log(LogF,"2. cSNP and SNP on different chromosomes",Err2)
General.Log(LogF,"3. SNP not one nucleotide in length",Err3)
General.Log(LogF,"4. cSNP and SNP coordinates differ",Err4)
General.Log(LogF,"5. refNCBI and refUCSC differ",Err5)
General.Log(LogF,"6. cSNP allele or codon incorrect length",Err6)
General.Log(LogF,"7. alleles not in codons",Err7)
General.Log(LogF,"SNP records WITH NO ERRORS (true coding SNPs)",NoErr)
LogF.write("\n")

FileName = str(NoErrFile).split(" ")
return (FileName[1][6:len(FileName[1])-1])

#####
# 2. Function: removes redundant cSNP records #
#           Redundant records = records with the same chromosome, SNP start #
#           position and on the same strand #
# - Called by: Identify_sSNPs() #
#####

def Remove_Redundancies(LogF):

    import General

    NumRec = 0
    NumRed = 0
    List = []

    SNPFile = open(Remove_Errors(LogF),"r")
    RedFile = open("2_1_Redundant_cSNPs.txt","w")
    NonRedFile = open("2_2_Non_redundant_cSNPs.txt","w")

    print ("Removing redundancies...")
    print ()

    for Line in SNPFile:
        NumRec = NumRec + 1
        Rec = (Line.rstrip()).split("\t")

        Id = Rec[0]+Rec[1]+Rec[6] # Id = Chrom + Start + Strand

```

```

        if Id not in List:
            List = List + [Id]
            NonRedFile.write(Line)
        else:
            NumRed = NumRed + 1
            RedFile.write(Line)

SNPFile.close()
RedFile.close()
NonRedFile.close()

General.Log(LogF, "coding SNPs checked for redundancies", NumRec)
General.Log(LogF, "redundant coding SNPs", NumRed)
General.Log(LogF, "NON-REDUNDANT coding SNPs", len(List))
LogF.write("\n")

FileName = str(NonRedFile).split(" ")
return (FileName[1][6:len(FileName[1])-1])

#####
# 4. Function: identifies sSNPs previously associated with a trait/disease in #
# genome-wide association studies (GWAS) #
# - Called by: Identify_sSNPs() #
#####

def GWAS(sSNPFileName, LogF):

    import General
    import winsound
    import os

    winsound.Beep(400, 1000)
    GWASFileName = input("Enter GWAS file name: ")
    while not os.path.isfile(GWASFileName):
        print ("File does not exist")
        GWASFileName = input("Enter file name: ")
    Header = int(input("Enter number of comment lines: "))
    print()

    print("Identifying sSNPs in GWAS...")
    print()

```

```

#-----#
# Save GWAS data in a dictionary                                     #
#   - Rec[1] = chromosome                                           #
#   - Rec[2] = SNP start position                                   #
#   - Rec[10] = trait                                               #
#   - Rec[15] = risk alleles                                        #
#   - Rec[17] = p-value                                             #
#-----#

GWASDict = {}
NumInGWAS = 0

GWASFile = open(GWASFileName, "r")
for i in range(0, Header):
    GWASFile.readline()

for Line in GWASFile:
    Rec = (Line.rstrip()).split("\t")
    Chrom = Rec[1].split("_")
    Chrom = Chrom[0]
    if Chrom not in GWASDict.keys():
        GWASDict[Chrom] = {Rec[2]:Rec[10]+": "+Rec[15]+", "+Rec[17]}
        NumInGWAS = NumInGWAS + 1
    elif Rec[2] not in GWASDict[Chrom].keys():
        GWASDict[Chrom][Rec[2]] = Rec[10]+": "+Rec[15]+", "+Rec[17]
        NumInGWAS = NumInGWAS + 1
    elif Rec[10] not in GWASDict[Chrom][Rec[2]]:
        GWASDict[Chrom][Rec[2]] = GWASDict[Chrom][Rec[2]] + "; " + Rec[10]+": " + Rec[15]
+ ", " + Rec[17]

GWASFile.close()

#-----#
# Loop over sSNP file and join to GWAS records based on SNP position #
#   - Rec[1] = sSNP start position                                   #
#-----#

NumGWAS = 0

SNPFile = open(sSNPFileName, "r")
OutFile = open("4_1_sSNPs_in_GWAS.txt", "w")

for Line in SNPFile:
    Rec = (Line.rstrip()).split("\t")
    Chrom = Rec[0].split("_")
    Chrom = Chrom[0]
    if Rec[1] in GWASDict[Chrom]:
        NumGWAS = NumGWAS + 1

```

```

        OutFile.write(Chrom)
        OutFile.write("\t")
        OutFile.write(Rec[1])
        OutFile.write("\t")
        OutFile.write(GWASDict[Chrom][Rec[1]])
        OutFile.write("\n")

    SNPFFile.close()
    OutFile.close()

    General.Log(LogF, "SNPs previously associated with a trait in GWAS catalogue", NumInGWAS)
    General.Log(LogF, "sSNPs previously associates with a trait in GWAS", NumGWAS)
    LogF.write("\n")

    FileName = str(OutFile).split(" ")
    return(FileName[1][6:len(FileName[1])-1])

#*****#
# 5. Function: identifies which of the sSNPs are within genes known to be      #
# associated with disease in OMIM                                             #
# - Called by: Identify_sSNPs()                                              #
#*****#

def OMIM(sSNPFileName, LogF):

    import General
    import winsound
    import os

    winsound.Beep(400,1000)
    OMIMFileName = input("Enter OMIM file name: ")
    while not os.path.isfile(OMIMFileName):
        print ("File does not exist")
        OMIMFileName = input("Enter file name: ")
    Header = int(input("Enter number of comment lines: "))
    print()

    print("Identifying sSNPs in OMIM...")
    print()

    OMIMFile = open(OMIMFileName, "r")\

    for i in range(0,Header):
        OMIMFile.readline()

```

```

#-----#
# Save OMIM data in a dictionary #
# - Rec[0] = trait or disease #
# - Rec[2] = OMIM id (MIM#) #
# - Rec[5] = chromosome #
# - Rec[6] = gene start position #
# - Rec[7] = gene end position #
#-----#

OMIMDict = {}
NumGenes = 0

for Line in OMIMFile:
    Rec = (Line.rstrip()).split("\t")
    Chrom = Rec[5].split("_")
    Chrom = Chrom[0]
    Gene = Rec[6]+" - "+Rec[7]
    if Chrom not in OMIMDict.keys():
        OMIMDict[Chrom] = {Gene: Rec[0].strip("{}")} + "(" + Rec[2] + ")"
        NumGenes = NumGenes + 1
    elif Gene not in OMIMDict[Chrom].keys():
        OMIMDict[Chrom][Gene] = Rec[0].strip("{}") + "(" + Rec[2] + ")"
        NumGenes = NumGenes + 1
    elif Rec[0] not in OMIMDict[Chrom][Gene]:
        OMIMDict[Chrom][Gene] = OMIMDict[Chrom][Gene] + "; " + Rec[0].strip("{}") + "("
+ Rec[2] + ")"

OMIMFile.close()

#-----#
# Loop over sSNP file and join to OMIM records based on SNP position #
# - Rec[1] = sSNP start position #
#-----#

SNPFile = open(sSNPFileName,"r")
OutFile = open("5_1_sSNPs_in_OMIM.txt","w")

NumOMIM = 0

for Line in SNPFile:

    Rec = (Line.rstrip()).split("\t")
    Chrom = Rec[0].split("_")
    Chrom = Chrom[0]

    if Chrom in OMIMDict.keys():
        List = ""
        for Genes in OMIMDict[Chrom]:

```

```

        Gene = Genes.split(" - ")
        GeneStart = int(Gene[0])
        GeneEnd = int(Gene[1])
        if (int(Rec[1]) >= GeneStart):
            if (int(Rec[1]) < GeneEnd):
                List = List + Genes + ": " + OMIMDict[Chrom][Genes] + " | "
    if List != "":
        NumOMIM = NumOMIM + 1
        OutFile.write(Chrom)
        OutFile.write("\t")
        OutFile.write(Rec[1])
        OutFile.write("\t")
        OutFile.write(List.rstrip(" | "))
        OutFile.write("\n")

    SNPFile.close()
    OutFile.close()

    General.Log(LogF, "Genes known to be associated with disease in OMIM", NumGenes)
    General.Log(LogF, "sSNPs within one or more genes known to be associated with disease in
OMIM", NumOMIM)
    LogF.write("\n")

    FileName = str(OutFile).split(" ")
    return(FileName[1][6:len(FileName[1])-1])

#####
# 3. Function: identify sSNPs as those SNPs that do not change the amino acid #
#          encoded                                                         #
#          - Rec[9] = alleles                                              #
#          - Rec[10] = codons                                              #
#          - Rec[11] = amino acids                                         #
#          - Called by: SSNS()                                            #
#####

def Identify_sSNPs(LogF):

    import General

    SNPfile = open(Remove_Redundancies(LogF), "r")
    NonSynFile = open("3_1_Non_synonymous_SNPs.txt", "w")
    SynFile = open("3_2_Synonymous_SNPs.txt", "w")

    print ("Identifying sSNPs...")
    print ()

```

```

NumRec = 0
NumSyn = 0
NumNonSyn = 0
Compl = {"A":"T", "T":"A", "C":"G", "G":"C"}

for Line in SNPfile:
    NumRec = NumRec + 1
    Rec = (Line.rstrip()).split("\t")

    #-----#
    # 3.1. Loop over the amino acids and keep record of how many times      #
    #       each amino acid is encoded for                                  #
    #-----#

    Dict = {}
    AminoAcidRec = Rec[11].split(",")
    for i in range(0, len(AminoAcidRec)):
        if AminoAcidRec[i] not in Dict.keys():
            Dict[AminoAcidRec[i]] = [i]
        else:
            Dict[AminoAcidRec[i]] = Dict[AminoAcidRec[i]] + [i]

    #-----#
    # 3.2. If the same amino acid is encoded for more than once then the    #
    #       cSNP is identified as synonymous                                #
    #-----#

    Alleles = ""
    Codons = ""
    AminoAcids = ""
    AlleleRec = Rec[8].split(",")
    CodonRec = Rec[9].split(",")
    for AA in Dict:
        if len(Dict[AA]) > 1:
            for i in Dict[AA]:
                if AminoAcidRec[i] != "X":
                    if CodonRec[i] not in Codons:
                        Alleles = Alleles + AlleleRec[i] + ","
                        Codons = Codons + CodonRec[i] + ","
                        AminoAcids = AminoAcids + AminoAcidRec[i] + ","

    #-----#
    # 3.3. Write the identified synonymous and non-synonymous SNPs to      #
    #       separate file                                                    #
    #-----#

    if len(Alleles) > 2:
        NumSyn = NumSyn + 1

```

```

        if Rec[7] in Alleles:
            RefNCBI = Rec[7]
        elif Compl[Rec[7]] in Alleles:
            RefNCBI = Compl[Rec[7]]
        for i in range(0,7):
            SynFile.write(Rec[i])
            SynFile.write("\t")
        SynFile.write(RefNCBI)
        SynFile.write("\t")
        SynFile.write(Alleles.rstrip(","))
        SynFile.write("\t")
        SynFile.write(Codons.rstrip(","))
        SynFile.write("\t")
        SynFile.write(Rec[10])
        SynFile.write("\t")
        SynFile.write(AminoAcids.rstrip(","))
        SynFile.write("\n")
    else:
        NumNonSyn = NumNonSyn + 1
        NonSynFile.write(Line)

SNPfile.close()
SynFile.close()
NonSynFile.close()

General.Log(LogF, "coding SNPs", NumRec)
General.Log(LogF, "non-synonymous SNPs", NumNonSyn)
General.Log(LogF, "SYNONYMOUS SNPs", NumSyn)
LogF.write("\n")

#-----#
# Identify which of the sSNPs have previously been associated with a      #
# trait or disease in GWAS and/or are within genes known to be associated #
# with disease in OMIM                                                    #
#-----#

FileName = str(SynFile).split(" ")
FileName = FileName[1][6:len(FileName[1])-1]

GWASFile = GWAS(FileName, LogF)
OMIMFile = OMIM(FileName, LogF)

Num = General.GWAS_And_OMIM(GWASFile, OMIMFile, "6_1_sSNPs_GWAS_OMIM.txt")
General.Log(LogF, "sSNPs previously associated with a trait or disease in GWAS and within
genes known to be associated with disease in OMIM", Num)
LogF.write("\n")

return (FileName)

```

```

#####
# Function: writes additional synonymous single nucleotide substitution data #
#           to log file #
# - Called by: SSNS() #
#####

def Stats(FileName,Total,LogF):

    FrameDict={}
    NtDict={}
    AADict = {}

    File = open(FileName,"r")
    for Line in File:
        Rec = (Line.rstrip()).split("\t")
        for i in range(0,3):
            if Rec[9][i] != Rec[10][i]:
                Frame = i
            if Frame not in FrameDict:
                FrameDict[Frame] = 1
            elif Frame in FrameDict:
                FrameDict[Frame] = FrameDict[Frame] + 1

            if Rec[7]+">"+Rec[8] not in NtDict:
                NtDict[Rec[7]+">"+Rec[8]] = 1
            elif Rec[7]+">"+Rec[8] in NtDict:
                NtDict[Rec[7]+">"+Rec[8]] = NtDict[Rec[7]+">"+Rec[8]] + 1

            if Rec[12] not in AADict:
                AADict[Rec[12]] = 1
            elif Rec[12] in AADict:
                AADict[Rec[12]] = AADict[Rec[12]] + 1
    File.close()

    LogF.write("SSNS position in codon\t# SSNS\t%\n")
    for i in FrameDict:
        LogF.write(str(i))
        LogF.write("\t")
        LogF.write(str(FrameDict[i]))
        LogF.write("\t")
        LogF.write("%.2f"%(FrameDict[i]/Total)*100))
        LogF.write("\n")
    LogF.write("\n")

    LogF.write("SSNS change\t# SSNS\t%\n")
    for i in NtDict:
        LogF.write(str(i))
        LogF.write("\t")

```

```

        LogF.write(str(NtDict[i]))
        LogF.write("\t")
        LogF.write("%.2f"%((NtDict[i]/Total)*100))
        LogF.write("\n")
    LogF.write("\n")

    LogF.write("Synonymous amino acids\t# SSNS\t%\n")
    for i in AADict:
        LogF.write(str(i))
        LogF.write("\t")
        LogF.write(str(AADict[i]))
        LogF.write("\t")
        LogF.write("%.2f"%((AADict[i]/Total)*100))
        LogF.write("\n")
    LogF.write("\n")

#####
# Function: writes synonymous single nucleotide substitution records to file  #
# - Called by: SSNS()                                                         #
#####

def SSNS_File(File,NumRec,Rec,Strand,RefAllele,Alleles,Codons,AminoAcids):

    if RefAllele not in Alleles:
        print (Rec)
    else:
        Ref = Alleles.index(RefAllele)
        for i in range(0,Ref):
            NumRec = NumRec + 1
            for j in range(0,6):
                File.write(Rec[j])
                File.write("\t")
            File.write(Strand)
            File.write("\t")
            File.write(RefAllele)
            File.write("\t")
            File.write(Alleles[i])
            File.write("\t")
            File.write(Codons[Ref])
            File.write("\t")
            File.write(Codons[i])
            File.write("\t")
            File.write(Rec[10])
            File.write("\t")
            File.write(AminoAcids[Ref])
            File.write("\t")
            File.write(AminoAcids[i])
            File.write("\n")

```

```

    for i in range(Ref+1,len(Alleles)):
        NumRec = NumRec + 1
        for j in range(0,6):
            File.write(Rec[j])
            File.write("\t")
        File.write(Strand)
        File.write("\t")
        File.write(RefAllele)
        File.write("\t")
        File.write(Alleles[i])
        File.write("\t")
        File.write(Codons[Ref])
        File.write("\t")
        File.write(Codons[i])
        File.write("\t")
        File.write(Rec[10])
        File.write("\t")
        File.write(AminoAcids[Ref])
        File.write("\t")
        File.write(AminoAcids[i])
        File.write("\n")

    return(NumRec)

#*****#
# 6. Function: Create separate records for each synonymous single nucleotide  #
#               substitution (SSNS)                                         #
#   - Rec[8]   = alleles                                                    #
#   - Rec[9]   = codons                                                    #
#   - Rec[10]  = sSNP position in codon                                    #
#   - Rec[11]  = amino acids                                              #
#   - Called by: Create_Chrom_Files()                                     #
#*****#

def SSNS(LogF):

    import General

    sSNPFileName = Identify_sSNPs(LogF)

    print ("Identifying synonymous single nucleotide substitution...")
    print ()

    NumRec = 0
    NumSSNS = 0
    DNA = ["A","C","G","T"]
    ComplStr = {"-":"+","+": "-"}
    Compl = {"A":"T","T":"A","C":"G","G":"C"}

```

```

Dict = {}

SNPFile = open(sSNPFileName,"r")
SubFile = open("7_1_Synonymous_single_nucleotide_substitutions.txt","w")

for Line in SNPFile:
    NumRec = NumRec + 1
    Rec = (Line.rstrip()).split("\t")

    AlleleRec = (Rec[8].strip('')).split(",")
    CodonRec = (Rec[9].strip('')).split(",")
    AminoAcidRec = (Rec[11].strip('')).split(",")

    if len(AlleleRec) > 4:
        Alleles=Codons=AminoAcids = ""
        for Nt in DNA:
            if Nt in AlleleRec:
                Index = AlleleRec.index(Nt)
                Alleles = Alleles + AlleleRec[Index] + ","
                Codons = Codons + CodonRec[Index] + ","
                AminoAcids = AminoAcids + AminoAcidRec[Index] + ","
        Alleles = Alleles.rstrip(",")
        Codons = Codons.rstrip(",")
        AminoAcids = AminoAcids.rstrip(",")

    if len(AminoAcidRec) == 4 and AminoAcidRec[1] != AminoAcidRec[2]:
        if 2 not in Dict:
            Dict[2] = 2
        else:
            Dict[2] = Dict[2] + 2
        Allele1 = [AlleleRec[0],AlleleRec[1]]
        Codon1 = [CodonRec[0],CodonRec[1]]
        AminoAcid1 = [AminoAcidRec[0],AminoAcidRec[1]]
        Allele2 = [AlleleRec[2],AlleleRec[3]]
        Codon2 = [CodonRec[2],CodonRec[3]]
        AminoAcid2 = [AminoAcidRec[2],AminoAcidRec[3]]
        if Rec[7] in AlleleRec[0:2]:
            NumSSNS =
SSNS_File(SubFile,NumSSNS,Rec,Rec[6],Rec[7],Allele1,Codon1,AminoAcid1)
            NumSSNS =
SSNS_File(SubFile,NumSSNS,Rec,ComplStr[Rec[6]],Compl[Rec[7]],Allele2,Codon2,AminoAcid2)
        elif Rec[7] in AlleleRec[2:4]:
            NumSSNS =
SSNS_File(SubFile,NumSSNS,Rec,ComplStr[Rec[6]],Compl[Rec[7]],Allele1,Codon1,AminoAcid1)
            NumSSNS =
SSNS_File(SubFile,NumSSNS,Rec,Rec[6],Rec[7],Allele2,Codon2,AminoAcid2)
        else:

```

```

        if len(AlleleRec) not in Dict:
            Dict[len(AlleleRec)] = 1
        else:
            Dict[len(AlleleRec)] = Dict[len(AlleleRec)] + 1
        NumSSNS =
SSNS_File(SubFile,NumSSNS,Rec,Rec[6],Rec[7],AlleleRec,CodonRec,AminoAcidRec)

    SNPFile.close()
    SubFile.close()

    General.Log(LogF,"synonymous SNPs",NumRec)
    SortedDict = sorted(Dict)
    for i in SortedDict:
        General.Log(LogF,"sSNPs with "+str(i)+" alleles",Dict[i])
    General.Log(LogF,"SYNONYMOUS SINGLE NUCLEOTIDE SUBSTITUTIONS",NumSSNS)
    LogF.write("\n")

    FileName = str(SubFile).split(" ")
    FileName = FileName[1][6:len(FileName[1])-1]

    Stats(FileName,NumSSNS,LogF)

    return(NumSSNS,FileName)

#*****#
# 7. Function: separates the SSNS into separate files for each chromosome      #
# - Called by: SynSNP                                                         #
#*****#

def Create_Chrom_Files(Home):

    import os
    import General
    import winsound
    import time

    print ("START:",time.ctime())
    print ()

    General.Create_Dir(os.getcwd() + "\\Results\\1_sSNPs")

    LogFile = open("Log_sSNPs.txt","w")

    NumRec,SSNSFileName = SSNS(LogFile)

    print ("Separating SSNS according to chromosomes...")
    print ()

```

```

Dict = {}
SSNSFile = open(SSNSFileName, "r")
for i in range(0, NumRec):
    Index = SSNSFile.tell()
    Rec = ((SSNSFile.readline()).rstrip()).split("\t")
    Chromosome = (Rec[0].strip('')).split("_")
    if Chromosome[0] not in Dict.keys():
        Dict[Chromosome[0]] = [Index]
    else:
        Dict[Chromosome[0]] = Dict[Chromosome[0]] + [Index]

General.Create_Dir(os.getcwd() + "\\SSNS_chroms")
SortedDict = sorted(Dict)
for Chrom in SortedDict:
    ChromFile = open(Chrom+"_SSNS.txt", "w")
    for Index in Dict[Chrom]:
        SSNSFile.seek(Index)
        ChromFile.write(SSNSFile.readline())
    ChromFile.close()
    General.Log(LogFile, "synonymous single nucleotide substitutions on
"+Chrom, len(Dict[Chrom]))
    LogFile.write("\n")

SSNSFile.close()
LogFile.close()

winsound.Beep(700, 1000)
print("END:", time.ctime())
print ()
print ("COMPLETE")
print ("-----")
print ("The identified sSNPs and SSNS have been saved in:")
print (os.getcwd())
print ()

os.chdir(Home)

#####
#                                     End of sSNPs.pyw                                #
#####

```

```
#####
#                               Codons.pyw                               #
#####

#####
# This library contains all the functions required to determine which SSNS      #
# could potentially result an extreme change in codon-usage frequency (CUF)      #
#####

#####
# Function: creates the codon-usage frequency dictionary, required to          #
#          calculate the change in codon-usage frequency caused by each SSNS    #
#          - Rec[1] = codon sequence                                           #
#          - Rec[3] = codon-usage frequency (%)                                #
#          - Called by: Codon_Usage_Change()                                   #
#####

def Codon_Dictionary(Home):

    import os

    Dict = {}

    File = open(Home + "\\Functions\\Codon_freq.txt", "r")
    for Line in File:
        Rec = (Line.rstrip()).split("\t")
        Dict[Rec[1]] = float(Rec[3])
    File.close()

    return(Dict)

#####
# 1. Function: calculates the change in codon-usage frequency caused by each    #
#          SSNS                                                                #
#          - Rec[9] = reference codon                                           #
#          - Rec[10] = variant codon                                            #
#          - Called by: Extreme_CUF()                                           #
#####

def Codon_Usage_Change(Home, LogF):

    import os
    import General

    CodonDict = Codon_Dictionary(Home)

    print ("Calculating change codon-usage frequency...")
    print()
```

```

NumRec = 0
NumRareToComm = 0
NumCommToRare = 0
Max = 0
Min = 1
CUFList = []

SSNSFile = open((Home +
"\Results\1_sSNPs\7_1_Synonymous_single_nucleotide_substitutions.txt"), "r")
os.chdir(Home + "\Results\2_Codon_usage")
CUFFile = open("1_1_SSNS_with_CUF.txt", "w")

for Line in SSNSFile:
    NumRec = NumRec + 1

    Rec = (Line.rstrip()).split("\t")

    RefFreq = CodonDict[Rec[9]]
    VarFreq = CodonDict[Rec[10]]
    Change = VarFreq-RefFreq

    CUFFile.write(Line.rstrip())
    CUFFile.write("\t")
    CUFFile.write("%.2f"%RefFreq)
    CUFFile.write("\t")
    CUFFile.write("%.2f"%VarFreq)
    CUFFile.write("\t")
    CUFFile.write("%.2f"%Change)
    CUFFile.write("\n")

    if Change not in CUFList:
        CUFList = CUFList + [Change]

    if Change > 0:
        NumRareToComm = NumRareToComm + 1
        if Change > Max:
            Max = Change
    elif Change < 0:
        NumCommToRare = NumCommToRare + 1
        if Change < Min:
            Min = Change

SSNSFile.close()
CUFFile.close()

CUFFileName = str(CUFFile).split(" ")
CUFFileName = CUFFileName[1][6:len(CUFFileName[1])-1]

```

```

General.Distribution(Min,Max,CUFFFileName,"1_2_CUF_distribution.txt",16)

General.Log(LogF,"SSNS records checked for change in CUF",NumRec)
General.Log(LogF,"SSNS that decrease CUF (change in CUF < 0)",NumCommToRare)
General.Log(LogF,"SSNS that that increase CUF (change in CUF > 0)",NumRareToComm)
LogF.write("\n")
General.Log(LogF,"Minimum change in CUF caused by SSNS",Min)
General.Log(LogF,"Maximum change in CUF caused by SSNS",Max)

return(CUFFFileName,sorted(CUFList))

#####
# Function: writes additional synonymous single nucleotide substitution data  #
#           to log file                                                    #
# - Called by: Extreme_CUF                                                #
#####

def Write_To_LogF(LogF,SSNSDict,CodonDict,NumSSNS):

    LogF.write("SSNS change\t#SSNS\t%SSNS\n")
    for SSNS in SSNSDict:
        LogF.write(SSNS)
        LogF.write("\t")
        LogF.write(str(SSNSDict[SSNS]))
        LogF.write("\t")
        LogF.write("%.2f"%((SSNSDict[SSNS]/NumSSNS)*100))
        LogF.write("\n")
    LogF.write("\n")

    LogF.write("Codon change\t#SSNS\t%SSNS\n")
    for Codon in CodonDict:
        LogF.write(Codon)
        LogF.write("\t")
        LogF.write(str(CodonDict[Codon]))
        LogF.write("\t")
        LogF.write("%.2f"%((CodonDict[Codon]/NumSSNS)*100))
        LogF.write("\n")
    LogF.write("\n")

```

```

#####
# 2. Function: identifies SSNS that cause an extreme change in codon-usage      #
#           frequency                                                            #
#   - Called by: Extreme_CUF_In_Disease()                                       #
#####

def Extreme_CUF (Home, LogF) :

    import General

    CUFFFileName, CUFList = Codon_Usage_Change (Home, LogF)

    ExtrDec, ExtrInc = General.Cut_Off_Value (CUFList)
    General.Log (LogF, "Extreme decrease in CUF cut-off value", ExtrDec)
    General.Log (LogF, "Extreme increase in CUF cut-off value", ExtrInc)
    LogF.write ("\n")

    print ("Identifying SSNS that result in an extreme change in CUF...")
    print ()

    NumDec = 0
    NumInc = 0
    SSNSIncDict = {}
    SSNSDecDict = {}
    CodonDecDict = {}
    CodonIncDict = {}

    CUFFFile = open (CUFFFileName, "r")
    DecFile = open ("2_1_Extreme_DECREASE_CUF.txt", "w")
    IncFile = open ("2_2_Extreme_INCREASE_CUF.txt", "w")

    for Line in CUFFFile:
        Rec = (Line.rstrip()).split ("\t")
        if float (Rec[16]) <= ExtrDec:
            NumDec = NumDec + 1
            DecFile.write (Line)
            if Rec[7]+">"+Rec[8] not in SSNSDecDict:
                SSNSDecDict [Rec[7]+">"+Rec[8]] = 1
            else:
                SSNSDecDict [Rec[7]+">"+Rec[8]] = SSNSDecDict [Rec[7]+">"+Rec[8]] + 1
            if Rec[9]+"->"+Rec[10] not in CodonDecDict:
                CodonDecDict [Rec[9]+"->"+Rec[10]] = 1
            else:
                CodonDecDict [Rec[9]+"->"+Rec[10]] = CodonDecDict [Rec[9]+"->"+Rec[10]] + 1
        elif float (Rec[16]) >= ExtrInc:
            NumInc = NumInc + 1
            IncFile.write (Line)
            if Rec[7]+">"+Rec[8] not in SSNSIncDict:

```

```

        SSNSIncDict[Rec[7]+">"+Rec[8]] = 1
    else:
        SSNSIncDict[Rec[7]+">"+Rec[8]] = SSNSIncDict[Rec[7]+">"+Rec[8]] + 1
    if Rec[9]+"->"+Rec[10] not in CodonIncDict:
        CodonIncDict[Rec[9]+"->"+Rec[10]] = 1
    else:
        CodonIncDict[Rec[9]+"->"+Rec[10]] = CodonIncDict[Rec[9]+"->"+Rec[10]] + 1

CUFFFile.close()
DecFile.close()
IncFile.close()

General.Log(LogF,"SSNS that cause an EXTREME DECREASE in CUF",NumDec)
General.Log(LogF,"SSNS that cause an EXTREME INCREASE in CUF",NumInc)
LogF.write("\n")
if len(SSNSDecDict) > 0:
    LogF.write("SSNS that cause an EXTREME DECREASE in CUF\n")
    Write_To_LogF(LogF,SSNSDecDict,CodonDecDict,NumDec)
    LogF.write("\n")
if len(SSNSIncDict) > 0:
    LogF.write("SSNS that cause an EXTREME INCREASE in CUF\n")
    Write_To_LogF(LogF,SSNSIncDict,CodonIncDict,NumInc)
    LogF.write("\n")

DecFileName = str(DecFile).split(" ")
IncFileName = str(IncFile).split(" ")
return(DecFileName[1][6:len(DecFileName[1])-1],IncFileName[1][6:len(IncFileName[1])-1])

#####
# 3. Function: identifies which of the SSNS predicted to potentially result in #
#           an extreme change in codon-usage frequency have previously been #
#           associated with a trait or disease in GWAS and/or are within #
#           genes known to be associated with disease in OMIM #
# - Called by: SynSNP.py #
#####

def Extreme_CUF_In_Disease(Home):

    import os
    import General
    import winsound
    import time

    print("START:",time.ctime())
    print()
    General.Create_Dir(Home + "\\Results\\2_Codon_usage")
    LogFile = open(os.getcwd() + "\\Log_Codons.txt","w")

```

```

DecFileName,IncFileName = Extreme_CUF(Home,LogFile)

print("Identifying SSNS that result in an extreme change in CUF in GWAS...")
print()
NumGWAS = General.GWAS_Or_OMIM(Home,"GWAS",DecFileName,"3_1_Extreme_DECREASE_GWAS.txt")
General.Log(LogFile,"SSNS that cause an EXTREME DECREASE in CUF associated with trait or
disease in GWAS",NumGWAS)
NumGWAS = General.GWAS_Or_OMIM(Home,"GWAS",IncFileName,"3_2_Extreme_INCREASE_GWAS.txt")
General.Log(LogFile,"SSNS that cause an EXTREME INCREASE in CUF associated with trait or
disease in GWAS",NumGWAS)
LogFile.write("\n")

print("Identifying SSNS that result in an extreme change in CUF in OMIM...")
print()
NumOMIM = General.GWAS_Or_OMIM(Home,"OMIM",DecFileName,"4_1_Extreme_DECREASE_OMIM.txt")
General.Log(LogFile,"SSNS that cause an EXTREME DECREASE in CUF within genes known to be
associated with disease in OMIM",NumOMIM)
NumOMIM = General.GWAS_Or_OMIM(Home,"OMIM",IncFileName,"4_2_Extreme_INCREASE_OMIM.txt")
General.Log(LogFile,"SSNS that cause an EXTREME INCREASE in CUF within genes known to be
associated with disease in OMIM",NumOMIM)
LogFile.write("\n")

Num =
General.GWAS_And_OMIM("3_1_Extreme_DECREASE_GWAS.txt","4_1_Extreme_DECREASE_OMIM.txt","5_1_Ext
reme_DECREASE_GWAS_OMIM.txt")
General.Log(LogFile,"SSNS that cause an EXTREME DECREASE in GWAS and OMIM",Num)
Num =
General.GWAS_And_OMIM("3_2_Extreme_INCREASE_GWAS.txt","4_2_Extreme_INCREASE_OMIM.txt","5_2_Ext
reme_INCREASE_GWAS_OMIM.txt")
General.Log(LogFile,"SSNS that cause an EXTREME INREASE in CUF in GWAS and in OMIM",Num)

LogFile.close()

winsound.Beep(700,1000)
print("END:",time.ctime())
print ()
print ("COMPLETE")
print ("-----")
print ("The SSNS predicted to potentially result an extreme change in CUF")
print ("have been saved in:")
print (os.getcwd())
print ()

os.chdir(Home)

#####
#                               End of Codons.pyw                               #
#####

```

```
#####
#                                     mRNA.pyw                                #
#####

#####
# This library contains all the functions required to determine which SSNS      #
# could potentially result an extreme change in the minimum free-energy (MFE)    #
# of local mRNA secondary structures                                           #
#####

#*****#
# 1. Function: removes CDS records with errors                                #
#   - Called by: Ref_And_Var()                                                #
#*****#

def Remove_Errors(LogF):

    import General
    import os
    import winsound

    winsound.Beep(400,1000)
    CDSFileName = input("Enter CDS file name: ")
    while not os.path.isfile(CDSFileName):
        print("File does not exist")
        CDSFileName = input("Enter file name: ")
    Header = int(input("Enter number of comment lines: "))
    print()

    print("Removing errors...")
    print()

    NumRec=Err1=Err2=NoErr = 0

    CDSFile = open(CDSFileName,"r")
    ErrFile = open("1_1_CDS_with_errors.txt","w")
    NoErrFile = open("1_2_CDS_without_errors.txt","w")

    for i in range(0,Header):
        CDSFile.readline()

    for Line in CDSFile:
        NumRec = NumRec + 1
        Rec = (Line.rstrip()).split("\t")
```

```

#-----#
# 1.1. No data must be missing i.e. all records must contain 8 fields #
#-----#

if len(Rec) != 8:
    Err1 = Err1 + 1
    ErrFile.write("1. Data missing\t")
    ErrFile.write(Line)

#-----#
# 1.2. The CDS sequence length must correspond to the CDS coordinates #
#       - Rec[5] = CDS start position                                #
#       - Rec[6] = CDS end position                                  #
#       - Rec[7] = CDS sequence                                     #
#-----#

elif int(Rec[6]) - int(Rec[5]) != len(Rec[7]):
    Err2 = Err2 + 1
    ErrFile.write("2. CDS incorrect length\t")
    ErrFile.write(Line)
else:
    NoErr = NoErr + 1
    NoErrFile.write(Rec[0]) # Chromosome
    NoErrFile.write("\t")
    NoErrFile.write(Rec[5]) # CDS start position
    NoErrFile.write("\t")
    NoErrFile.write(Rec[6]) # CDS end position
    NoErrFile.write("\t")
    NoErrFile.write(Rec[3]) # CDS name (NM#)
    NoErrFile.write("\t")
    NoErrFile.write(Rec[4]) # CDS strand
    NoErrFile.write("\t")
    NoErrFile.write(Rec[7]) # CDS sequence
    NoErrFile.write("\n")

ErrFile.close()
NoErrFile.close()

General.Log(LogF, "CDS in file "+CDSFileName, NumRec)
General.Log(LogF, "CDS records with errors", NumRec-NoErr)
General.Log(LogF, "1. records with data missing", Err1)
General.Log(LogF, "2. CDS coordinates do not correspond with CDS sequence length", Err2)
General.Log(LogF, "CDS records with NO ERRORS", NoErr)
LogF.write("\n")

FileName = str(NoErrFile).split(" ")
return (FileName[1][6:len(FileName[1])-1], NoErr)

```

```

#####
# Function: removes introns from the CDS to create the mRNA sequence and      #
#           determines the sSNP's position in the mRNA                      #
# - Called by: Ref_And_Var()                                                #
#####

def Create_mRNA(PosiInCDS,CDSseq):

    Mseq      = ""
    PosiInSeq = 0
    PosiInmRNA = 0
    sSNPposi  = 0

    for Nt in CDSseq:
        if Nt.isupper():
            Mseq = Mseq + Nt
            if PosiInSeq == PosiInCDS:
                sSNPposi = PosiInmRNA
                PosiInmRNA = PosiInmRNA + 1
            PosiInSeq = PosiInSeq + 1

    return(sSNPposi,Mseq)

#####
# Function: write the 100 base reference and variant mRNA sequences to file  #
# - Called by: Ref_And_Var()                                                #
#####

def Write_To_File(File,Recs,Posi,Ref,Var):

    for i in range(0,12):
        File.write(Recs[i])
        File.write("\t")
    File.write(str(Posi))
    File.write("\t")
    File.write(Ref)
    File.write("\t")
    File.write(Var)
    File.write("\n")

#####
# Function: writes additional SSNS in 100 base mRNA data to log file        #
# - Called by: Ref_And_Var()                                                #
#####

def Write_To_LogF(LogF,Dict,Num):

    LogF.write("SSNS change\t#SSNS\t%SSNS\n")

```

```

for i in Dict:
    LogF.write(i)
    LogF.write("\t")
    LogF.write(str(Dict[i]))
    LogF.write("\t")
    LogF.write("%.2f"%(Dict[i]/Num)*100))
    LogF.write("\n")
LogF.write("\n")

#####
# 2. Function: creates mRNA from CDS and the creates 100 base mRNA from mRNA #
# - Called by: Fasta() #
#####

def Ref_And_Var(LogF,ResultsHome,Home):

    import General

    NoErrFile,NumRec = Remove_Errors(LogF)

    InFileName = General.Check_Alleles(Home,ResultsHome,NoErrFile,NumRec,"CDS",LogF)

    NumSSNS=NumRec-100 = 0
    Dict = {}

    SSNSFile = open(InFileName,"r")
    ErrFile = open("5_1_Reference_allele_not_in_mRNA.txt","w")
    OutFile = open("5_2_sSNPs_in_100b_mRNA.txt","w")

    print("Creating reference and variant mRNA sequences...")
    print()

    for Line in SSNSFile:
        NumSSNS = NumSSNS + 1
        Rec = (Line.rstrip()).split("\t")

        #-----#
        # Determine sSNP position in CDS #
        #-----#

        if Rec[6] == "+":
            SSNSposi = int(Rec[1]) - int(Rec[9])
        elif Rec[6] == "-":
            SSNSposi = int(Rec[10]) - int(Rec[1]) - 1

```

```

#-----#
# Create mRNA sequence from CDS sequence                                     #
#-----#

PosiInmRNA,mRNAseq = Create_mRNA(SSNSposi,Rec[12])

#-----#
# Slice 100bp reference mRNA sequence from mRNA sequence                   #
#-----#

if (PosiInmRNA >= 50) and (len(mRNAseq)-PosiInmRNA >= 50):
    NumRec = NumRec + 1
    if Rec[7]+">"+Rec[8] not in Dict:
        Dict[Rec[7]+">"+Rec[8]] = 1
    elif Rec[7]+">"+Rec[8] in Dict:
        Dict[Rec[7]+">"+Rec[8]] = Dict[Rec[7]+">"+Rec[8]] + 1
    RefSeq = mRNAseq[PosiInmRNA-50:PosiInmRNA+50]
    VarSeq = RefSeq[0:50] + Rec[8] + RefSeq[51:101]
    Write_To_File(OutFile,Rec,PosiInmRNA,RefSeq,VarSeq)
else:
    Not100 = Not100 + 1

SSNSFile.close()
ErrFile.close()
OutFile.close()

General.Log(LogF,"SSNS in mRNA",NumSSNS)
General.Log(LogF,"SSNS less than 50 bases from start or end of mRNA sequence",Not100)
General.Log(LogF,"SSNS in 100 base mRNA",NumRec)
LogF.write("\n")

if len(Dict) > 0:
    LogF.write("SSNS in 100 base mRNA\n")
    Write_To_LogF(LogF,Dict,NumRec)

FileName = str(OutFile).split(" ")
return (FileName[1][6:len(FileName[1])-1])

```

```

#####
# 3. Function: creates the Fasta file required by RNAfold                                     #
#   - Rec[1]   = sSNP start position                                                         #
#   - Rec[4]   = sSNP strand                                                                 #
#   - Rec[7]   = reference allele                                                            #
#   - Rec[8]   = variant allele                                                             #
#   - Rec[10]  = CDS strand                                                                  #
#   - Rec[11]  = CDS start position                                                         #
#   - Rec[13]  = sSNP position in 100b mRNA                                                 #
#   - Called by: RNAfold()                                                                  #
#####

def Fasta(LogF,ResultsHome,Home):

    SSNSFile = open(Ref_And_Var(LogF,ResultsHome,Home),"r")
    FastaFile = open("6_1_sSNPs_in_mRNA.fasta","w")

    print("Creating Fasta file...")
    print()

    for Line in SSNSFile:
        Rec = (Line.rstrip()).split("\t")

        FastaId = ">" + Rec[1] + "." + Rec[9] + "." + Rec[12] + "." + Rec[6] + "." + Rec[7] + "." + Rec[8] + "."
        # SSNS start.CDS start.SSNS posi in mRNA.RefAllele.VarAllele.Strand
        RefId = FastaId + "REF"
        VarId = FastaId + "VAR"

        FastaFile.write(RefId)
        FastaFile.write("\n")
        FastaFile.write(Rec[13])
        FastaFile.write("\n")
        FastaFile.write(VarId)
        FastaFile.write("\n")
        FastaFile.write(Rec[14])
        FastaFile.write("\n")

    SSNSFile.close()
    FastaFile.close()

    SSNSFileName = str(SSNSFile).split(" ")
    FastaFileName = str(FastaFile).split(" ")
    return (SSNSFileName[1][6:len(SSNSFileName[1])-
1],FastaFileName[1][6:len(FastaFileName[1])-1])

```

```

#####
# 4. Function: uses RNAfold to calculate the minimum-free energy (MFE) of the #
#           reference and variant local mRNA secondary structures from the #
#           100 base mRNA sequences                                     #
#   - Called by: Change_In_MFE()                                       #
#####

def RNAfold(LogF,ResultsHome,Home):

    SSNSFile,FastaFile = Fasta(LogF,ResultsHome,Home)

    print("Running RNAfold...")
    print()

    import os

    Command = Home+"\\Functions\\RNAfold.exe -d0 -noPS <" + FastaFile + "
>6_2_RNAfold_results.txt"
    os.system(Command)

    return (SSNSFile,"6_2_RNAfold_results.txt")

#####
# 5. Function: joins the RNAfold results file and the 100 base mRNA file and #
#           calculates the change in MFE caused by each SSNS               #
#   - Rec[1]  = sSNP start position                                       #
#   - Rec[4]  = sSNP strand                                               #
#   - Rec[7]  = reference allele                                          #
#   - Rec[8]  = variant allele                                            #
#   - Rec[10] = CDS strand                                                #
#   - Rec[11] = CDS start position                                        #
#   - Rec[13] = sSNP position in 100b mRNA                               #
#   - Called by: Extreme_MFE()                                           #
#####

def Change_In_MFE(LogF,ResultsHome,Home):

    SSNSFileName,ResultsFileName = RNAfold(LogF,ResultsHome,Home)

    print("Calculating change in minimum-free energy...")
    print()

    import General

    Dict = {}

    ResultsFile = open(ResultsFileName,"r")
    for Line in ResultsFile:

```

```

    Key = (Line.rstrip()).lstrip(">")
    ResultsFile.readline()
    Result = ResultsFile.readline()
    Dict[Key] = [Result[0:100], float(Result[102:len(Result)-2])]
ResultsFile.close()

Min = 100
Max = 0
NumNoChange = 0
NumDec = 0
NumInc = 0
MFEList = []

SSNSFile = open(SSNSFileName, "r")
MFEFile = open("7_1_SSNS_in_mRNA_MFE.txt", "w")
for Line in SSNSFile:
    Rec = (Line.rstrip()).split("\t")
    FastaId = Rec[1]+"."+Rec[9]+"."+Rec[12]+"."+Rec[6]+"."+Rec[7]+"."+Rec[8]+"."
    # SSNS start.CDS start.SSNS posi in mRNA.RefAllele.VarAllele.Strand
    RefId = FastaId + "REF"
    VarId = FastaId + "VAR"
    Change = Dict[VarId][1]-Dict[RefId][1]

    MFEFile.write(Line.rstrip())
    MFEFile.write("\t")
    MFEFile.write(Dict[RefId][0])
    MFEFile.write("\t")
    MFEFile.write(Dict[VarId][0])
    MFEFile.write("\t")
    MFEFile.write("%.2f"%Dict[RefId][1])
    MFEFile.write("\t")
    MFEFile.write("%.2f"%Dict[VarId][1])
    MFEFile.write("\t")
    MFEFile.write("%.2f"%Change)
    MFEFile.write("\n")

    if Change not in MFEList:
        MFEList = MFEList + [Change]

    if Change == 0:
        NumNoChange = NumNoChange + 1
    elif Change < 0:
        NumDec = NumDec + 1
        if Change < Min:
            Min = Change
    elif Change > 0:
        NumInc = NumInc + 1
        if Change > Max:

```

```

        Max = Change

    SSNSFile.close()
    MFEFile.close()

    FileName = str(MFEFile).split(" ")
    FileName = FileName[1][6:len(FileName[1])-1]

    General.Distribution(Min,Max,FileName,"7_2_MFE_distribution.txt",19)

    General.Log(LogF,"SSNS that do not change MFE (change in MFE = 0)",NumNoChange)
    General.Log(LogF,"SSNS that decrease MFE (change in MFE < 0)",NumDec)
    General.Log(LogF,"SSNS that increase MFE (change in MFE > 0)",NumInc)
    LogF.write("\n")
    General.Log(LogF,"Minimum change in MFE caused by a SSNS",Min)
    General.Log(LogF,"Maximum change in MFE caused by a SSNS",Max)

    return(FileName,sorted(MFEList))

#####
# 6. Function: identifies SSNS that result in an extreme change in MFE as the #
#           SSNS that cause a change in MFE less than or greater than the #
#           chosen cut-off values                                         #
# - Called by: Extreme_MFE_In_Disease()                                  #
#####

def Extreme_MFE(LogF,ResultsHome,Home):

    import General

    MFEFileName,MFEList = Change_In_MFE(LogF,ResultsHome,Home)

    ExtrDec,ExtrInc = General.Cut_Off_Value(MFEList)

    General.Log(LogF,"Extreme decrease in MFE cut-off value",ExtrDec)
    General.Log(LogF,"Extreme increase in MFE cut-off value",ExtrInc)
    LogF.write("\n")

    print("Identifying SSNS that result in an extreme change in MFE...")
    print()

    NumDec = 0
    NumInc = 0
    DecDict = {}
    IncDict = {}

    MFEFile = open(MFEFileName,"r")
    DecFile = open("8_1_Extreme_DECREASE_MFE.txt","w")

```

```

IncFile = open("8_2_Extreme_INCREASE_MFE.txt", "w")

for Line in MFEFile:
    Rec = (Line.rstrip()).split("\t")
    if float(Rec[19]) <= ExtrDec:
        NumDec = NumDec + 1
        DecFile.write(Line)
        if Rec[7]+">"+Rec[8] not in DecDict:
            DecDict[Rec[7]+">"+Rec[8]] = 1
        else:
            DecDict[Rec[7]+">"+Rec[8]] = DecDict[Rec[7]+">"+Rec[8]] + 1
    elif float(Rec[19]) >= ExtrInc:
        NumInc = NumInc + 1
        IncFile.write(Line)
        if Rec[7]+">"+Rec[8] not in IncDict:
            IncDict[Rec[7]+">"+Rec[8]] = 1
        else:
            IncDict[Rec[7]+">"+Rec[8]] = IncDict[Rec[7]+">"+Rec[8]] + 1

MFEFile.close()
DecFile.close()
IncFile.close()

General.Log(LogF, "SSNS that cause an EXTREME DECREASE in MFE", NumDec)
General.Log(LogF, "SSNS that cause an EXTREME INCREASE in MFE", NumInc)
LogF.write("\n")
LogF.write("SSNS that result in an EXTREME DECREASE in MFE\n")
Write_To_LogF(LogF, DecDict, NumDec)
LogF.write("SSNS that result in an EXTREME INCREASE in MFE\n")
Write_To_LogF(LogF, IncDict, NumInc)

DecFileName = str(DecFile).split(" ")
IncFileName = str(IncFile).split(" ")
return(DecFileName[1][6:len(DecFileName[1])-1], IncFileName[1][6:len(IncFileName[1])-1])

#*****#
# 7. Function: determines whether any of the SSNS predicted to potentially #
#             result in an extreme change in MFE have previously been #
#             associated with a trait or disease in GWAS and/or are within #
#             genes known to be associated with disease in OMIM #
# - Called by: SynSNP.py #
#*****#

def Extreme_MFE_In_Disease(Home):

    import os
    import winsound
    import time

```

```

import General

print("START:",time.ctime())
print()

General.Create_Dir(os.getcwd() + "\\Results\\3_mRNA")
ResultsDir = os.getcwd()

LogFile = open("Log_mRNA.txt","w")

DecFileName,IncFileName = Extreme_MFE(LogFile,ResultsDir,Home)

print("Identifying SSNS that result in an extreme change in MFE in GWAS...")
print()

NumGWAS = General.GWAS_Or_OMIM(Home,"GWAS",DecFileName,"9_1_Extreme_DECREASE_GWAS.txt")
General.Log(LogFile,"SSNS that cause an EXTREME DECREASE in MFE associated with trait or
disease in GWAS",NumGWAS)
NumGWAS = General.GWAS_Or_OMIM(Home,"GWAS",IncFileName,"9_2_Extreme_INCREASE_GWAS.txt")
General.Log(LogFile,"SSNS that cause an EXTREME INCREASE in MFE associated with trait or
disease in GWAS",NumGWAS)
LogFile.write("\n")

print("Identifying SSNS that result in an extreme change in MFE in OMIM...")
print()

NumOMIM = General.GWAS_Or_OMIM(Home,"OMIM",DecFileName,"10_1_Extreme_DECREASE_OMIM.txt")
General.Log(LogFile,"SSNS that cause an EXTREME DECREASE in MFE within genes known to be
associated with disease in OMIM",NumOMIM)
NumOMIM = General.GWAS_Or_OMIM(Home,"OMIM",IncFileName,"10_2_Extreme_INCREASE_OMIM.txt")
General.Log(LogFile,"SSNS that cause an EXTREME INCREASE in MFE within genes known to be
associated with disease in OMIM",NumOMIM)
LogFile.write("\n")

Num =
General.GWAS_And_OMIM("9_1_Extreme_DECREASE_GWAS.txt","10_1_Extreme_DECREASE_OMIM.txt","11_1_E
xtreme_DECREASE_GWAS_OMIM.txt")
General.Log(LogFile,"SSNS that cause an EXTREME INCREASE in MFE in GWAS and in OMIM",Num)
Num =
General.GWAS_And_OMIM("9_2_Extreme_INCREASE_GWAS.txt","10_2_Extreme_INCREASE_OMIM.txt","11_2_E
xtreme_INCREASE_GWAS_OMIM.txt")
General.Log(LogFile,"SSNS that cause an EXTREME DECREASE in MFE in GWAS and in OMIM",Num)

LogFile.close()

os.chdir(Home)

winsound.Beep(700,1000)

```

```

print("END:",time.ctime())
print ()
print ("COMPLETE")
print ("-----")
print ("The SSNS predicted to potentially result in an extreme change in the")
print ("MFE of mRNA secondary structures have been saved in:")
print (ResultsDir)

#####
#                               End of mRNA.pyw                               #
#####

#####
#                               SpliceSites.pyw                               #
#####

#####
# This library contains all the functions required to determine which SSNS      #
# could potentially result an extreme change in the maximum-entropy score of    #
# splice-sites                                                                    #
#####

#####
# 1. Function: removes exon records with errors                                #
#   - Called by: In_Splice_Sites()                                              #
#####

def Remove_Errors(LogF):

    import General
    import os
    import winsound

    winsound.Beep(400,1000)
    FileName = input("Enter exons file name: ")
    while not os.path.isfile(FileName):
        print("File does not exist")
        FileName = input("Enter file name: ")
    Header = int(input("Enter number of comment lines: "))
    print ()

    print ("Removing errors...")
    print ()

    NumRec=Err1=Err2=NoErr = 0
    File = open(FileName,"r")
    ErrFile = open("1_1_Exons_with_errors.txt","w")

```

```

NoErrFile = open("1_2_Exons_with_no_errors.txt","w")

for i in range(0,Header):
    File.readline()

for Line in File:
    NumRec = NumRec + 1
    Rec = (Line.rstrip()).split("\t")
    Rec = Rec[0:4]+Rec[5:7]

    #~~~~~#
    # 1.1. No data must be missing i.e. all records must contain 6 fields #
    #~~~~~#

    if len(Rec) != 6:
        Err1 = Err1 + 1
        ErrFile.write("1. Data missing\t")
        ErrFile.write(Line)

        #~~~~~#
        # 1.2. The sequence length and exon coordinates must correspond #
        #      - Rec[1] = exon start #
        #      - Rec[2] = exon end #
        #      - Rec[5] = exon sequence #
        #~~~~~#

        elif int(Rec[2]) - int(Rec[1]) != len(Rec[5]):
            Err2 = Err2 + 1
            ErrFile.write("2. Exon incorrect length\t")
            ErrFile.write(Line)

    else:
        NoErr = NoErr + 1
        for Field in Rec:
            NoErrFile.write(Field)
            NoErrFile.write("\t")
        NoErrFile.write("\n")

File.close()
ErrFile.close()
NoErrFile.close()

General.Log(LogF,"Exons in file "+FileName,NumRec)
General.Log(LogF,"Exon records with errors",NumRec-NoErr)
General.Log(LogF,"1. records with data missing",Err1)
General.Log(LogF,"2. exons of incorrect length",Err2)
General.Log(LogF,"Exons with NO ERRORS",NoErr)
LogF.write("\n")

```

```

    FileName = str(NoErrFile).split(" ")
    return (FileName[1][6:len(FileName[1])-1],NoErr)

#####
# Function: creates the reference and variant 3' splice-site sequences      #
#   - Called by: In_Splice_Sites()                                         #
#####

def Acceptor_Splice_Site(Ref,Posi,Allele):

    if Posi == 20:
        Var = Ref[0:20] + Allele + Ref[21:23]
        P = 0
    elif Posi == 21:
        Var = Ref[0:21] + Allele + Ref[22]
        P = 1
    elif Posi == 22:
        Var = Ref[0:22] + Allele
        P = 2

    return (Var,P)

#####
# Function: creates the reference and variant 5' splice-site sequences      #
#   - Called by: In_splice_Sites()                                         #
#####

def Donor_Splice_Site(Ref,Posi,Allele):

    if Posi == 23:
        Var = Allele + Ref[1:9]
        P = 0
    elif Posi == 22:
        Var = Ref[0] + Allele + Ref[2:9]
        P = 1
    elif Posi == 21:
        Var = Ref[0:2] + Allele + Ref[3:9]
        P = 2

    return (Var,P)

#####
# Function: writes the SSNS found within splice-sites to file              #
#   - Called by: In_Splice_Sites()                                         #
#####

def Write_To_File(File,Line,Ref,Var,P):

```

```

File.write(Line.rstrip())
File.write("\t")
File.write(Ref)
File.write("\t")
File.write(Var)
File.write("\t")
File.write(str(P))
File.write("\n")

#####
# Function: writes additional SSNS in splice-sites data to log file #
# - Called by: In_Splice_Sites() #
#####

def Write_Dict(LogF,PosiDict,SSNSDict,NumSSNS):
    LogF.write("SSNS position\t#SSNS\t%SSNS\n")
    for i in PosiDict:
        LogF.write(str(i+1))
        LogF.write("\t")
        LogF.write(str(PosiDict[i]))
        LogF.write("\t")
        LogF.write("%.2f"%((PosiDict[i]/NumSSNS)*100))
        LogF.write("\n")
    LogF.write("\n")

    LogF.write("SSNS change\t#SSNS\t%SSNS\n")
    for i in SSNSDict:
        LogF.write(i)
        LogF.write("\t")
        LogF.write(str(SSNSDict[i]))
        LogF.write("\t")
        LogF.write("%.2f"%((SSNSDict[i]/NumSSNS)*100))
        LogF.write("\n")
    LogF.write("\n")

#####
# 2. Function: determines which of the sSNPs are within 5' and 3' splice-sites #
# - Rec[1] = sSNP start position #
# - Rec[8] = variant allele #
# - Rec[9] = exon start position #
# - Rec[10] = exon end position #
# - Rec[13] = exon strand #
# - Rec[14] = exon sequence #
# - Called by: MaxEntScan() #
#####

def In_Splice_Sites(LogF,ResultsHome,Home):

```

```

import General

NoErrFile, NumRec = Remove_Errors(LogF)

InFileName = General.Check_Alleles(Home, ResultsHome, NoErrFile, NumRec, "EXONS", LogF)

print ("Identifying SSNS in exonic part of splice-sites...")
print ()

sSNPFile = open(InFileName, "r")
AccFile = open("5_1_SSNS_in_acceptor_splice_sites.txt", "w")
DonFile = open("5_2_SSNS_in_donor_splice_sites.txt", "w")

NumRec = 0
NumAcc = 0
NumDon = 0
DonPosiDict = {0:0, 1:0, 2:0}
AccPosiDict = {0:0, 1:0, 2:0}
DonSSNSDict = {}
AccSSNSDict = {}

for Line in sSNPFile:
    NumRec = NumRec + 1
    Rec = (Line.rstrip()).split("\t")

    if Rec[6] == "+":
        if (int(Rec[1])-int(Rec[9]) >= 20) and (int(Rec[1])-int(Rec[9]) <= 22):
            NumAcc = NumAcc + 1
            RefSeq = (Rec[12][0:20]).lower() + (Rec[12][20:23]).upper()
            VarSeq, Place = Acceptor_Splice_Site(RefSeq, int(Rec[1])-int(Rec[9]), Rec[8])
            Write_To_File(AccFile, Line, RefSeq, VarSeq, Place)
            AccPosiDict[Place] = AccPosiDict[Place] + 1
            if Rec[7]+">"+Rec[8] not in AccSSNSDict.keys():
                AccSSNSDict[Rec[7]+">"+Rec[8]] = 1
            elif Rec[7]+">"+Rec[8] in AccSSNSDict.keys():
                AccSSNSDict[Rec[7]+">"+Rec[8]] = AccSSNSDict[Rec[7]+">"+Rec[8]] + 1

        elif (int(Rec[10])-int(Rec[1]) >= 21) and (int(Rec[10])-int(Rec[1]) <= 23):
            NumDon = NumDon + 1
            RefSeq = (Rec[12][len(Rec[12])-23:len(Rec[12])-20]).upper() +
(Rec[12][len(Rec[12])-20:len(Rec[12])-14]).lower()
            VarSeq, Place = Donor_Splice_Site(RefSeq, int(Rec[10])-int(Rec[1]), Rec[8])
            Write_To_File(DonFile, Line, RefSeq, VarSeq, Place)
            DonPosiDict[Place] = DonPosiDict[Place] + 1
            if Rec[7]+">"+Rec[8] not in DonSSNSDict.keys():
                DonSSNSDict[Rec[7]+">"+Rec[8]] = 1
            elif Rec[7]+">"+Rec[8] in DonSSNSDict.keys():
                DonSSNSDict[Rec[7]+">"+Rec[8]] = DonSSNSDict[Rec[7]+">"+Rec[8]] + 1

```

```

elif Rec[6] == "-":
    if (int(Rec[10])-int(Rec[1])-1 >= 20) and (int(Rec[10])-int(Rec[1])-1 <= 22):
        NumAcc = NumAcc + 1
        RefSeq = (Rec[12][0:20]).lower() + (Rec[12][20:23]).upper()
        VarSeq, Place = Acceptor_Splice_Site(RefSeq, int(Rec[10])-int(Rec[1])-1, Rec[8])
        Write_To_File(AccFile, Line, RefSeq, VarSeq, Place)
        AccPosiDict[Place] = AccPosiDict[Place] + 1
        if Rec[7]+">" + Rec[8] not in AccSSNSDict.keys():
            AccSSNSDict[Rec[7]+">" + Rec[8]] = 1
        elif Rec[7]+">" + Rec[8] in AccSSNSDict.keys():
            AccSSNSDict[Rec[7]+">" + Rec[8]] = AccSSNSDict[Rec[7]+">" + Rec[8]] + 1

    elif (int(Rec[1])-int(Rec[9])+1 >= 21) and (int(Rec[1])-int(Rec[9])+1 <= 23):
        NumDon = NumDon + 1
        RefSeq = (Rec[12][len(Rec[12])-23:len(Rec[12])-20]).upper() +
(Rec[12][len(Rec[12])-20:len(Rec[12])-14]).lower()
        VarSeq, Place = Donor_Splice_Site(RefSeq, int(Rec[1])-int(Rec[9])+1, Rec[8])
        Write_To_File(DonFile, Line, RefSeq, VarSeq, Place)
        DonPosiDict[Place] = DonPosiDict[Place] + 1
        if Rec[7]+">" + Rec[8] not in DonSSNSDict.keys():
            DonSSNSDict[Rec[7]+">" + Rec[8]] = 1
        elif Rec[7]+">" + Rec[8] in DonSSNSDict.keys():
            DonSSNSDict[Rec[7]+">" + Rec[8]] = DonSSNSDict[Rec[7]+">" + Rec[8]] + 1

sSNPFile.close()
AccFile.close()
DonFile.close()

General.Log(LogF, "SSNS checked if in splice-site motifs", NumRec)
General.Log(LogF, "synonymous single nucleotide substitutions in exonic part of 3' ACCEPTOR
splice-sites", NumAcc)
General.Log(LogF, "synonymous single nucleotide substitutions in exonic part of 5' DONOR
splice-site", NumDon)
LogF.write("\n")

if len(AccSSNSDict) > 0:
    LogF.write("SSNS in exonic part of 3' ACCEPTOR splice-sites\n")
    Write_Dict(LogF, AccPosiDict, AccSSNSDict, NumAcc)

if len(DonSSNSDict) > 0:
    LogF.write("SSNS in exonic part of 5' DONOR splice-sites\n")
    Write_Dict(LogF, DonPosiDict, DonSSNSDict, NumDon)

AccName = str(AccFile).split(" ")
DonName = str(DonFile).split(" ")
return (AccName[1][6:len(AccName[1])-1], DonName[1][6:len(DonName[1])-1])

```

```

#####
# Function: creates the FASTA file required by MaxEntScan                                #
#       - Rec[13] = reference splice-site sequence                                    #
#       - Rec[14] = variant splice-site sequence                                    #
#       - Called by: MaxEntScan()                                                    #
#####

def MaxEnt_File(InFileName,OutFileName):

    InFile  = open(InFileName,"r")
    OutFile = open(OutFileName,"w")
    for Line in InFile:
        Rec = (Line.rstrip()).split("\t")
        OutFile.write(Rec[13])
        OutFile.write("\n")
        OutFile.write(Rec[14])
        OutFile.write("\n")
    InFile.close()
    OutFile.close()

    FileName = str(OutFile).split(" ")
    return (FileName[1][6:len(FileName[1])-1])

#####
# 3. Function: uses MaxEntScan to calculates the maximum entropy score for          #
#       each reference and variant splice-site                                    #
#       - Called by: Extreme_ME_In_Disease()                                        #
#####

def MaxEntScan(LogF,ResultsHome,Home):

    AccFileName,DonFileName = In_Splice_Sites(LogF,ResultsHome,Home)

    print ("Running MaxEntScan...")
    print ()

    import os

    os.chdir(Home + "\\Functions\\MaxEntScan")

    AccFile =
MaxEnt_File(ResultsHome+"\\ "+AccFileName,ResultsHome+"\\6_1_Acceptor_MaxEnt.txt")
    AccResults = "6_2_Acceptor_MaxEnt_results.txt"
    Command = "perl score3.pl " + AccFile + " >" + ResultsHome + "\\ " +AccResults
    os.system(Command)

    DonFile = MaxEnt_File(ResultsHome+"\\ "+DonFileName,ResultsHome+"\\6_3_Donor_MaxEnt.txt")
    DonResults = "6_4_Donor_MaxEnt_results.txt"

```

```

Command = "perl score5.pl " + DonFile + " >" + ResultsHome + "\\\" + DonResults
os.system(Command)

os.chdir(ResultsHome)

return(AccFileName,DonFileName,AccResults,DonResults)

#*****#
# 4. Function: joins the MaxEnt results file and the splice-sites file and      #
#               calculates the change in maximum-entropy score caused by each  #
#               SSNS                                                            #
# - Called by: Extreme_ME_In_Disease()                                         #
#*****#

def Change_In_ME(LogF, InFileName, ResultsFileName, OutFileName, Header, DistrFile):

    print("Calculating change in ME caused by SSNS in"+Header+"splice-sites...")
    print()

    import General

    Max = 0
    Min = 100
    NoChange = 0
    NumDec = 0
    NumInc = 0
    MEList = []

    InFile = open(InFileName, "r")
    ResultsFile = open(ResultsFileName, "r")
    OutFile = open(OutFileName, "w")

    for Line in InFile:
        Rec = (Line.rstrip()).split("\t")
        Ref = ((ResultsFile.readline()).rstrip()).split("\t")
        Var = ((ResultsFile.readline()).rstrip()).split("\t")
        Change = float(Var[1]) - float(Ref[1])
        if (Rec[13] == Ref[0]) and (Rec[14] == Var[0]):
            OutFile.write(Line.rstrip())
            OutFile.write("\t")
            OutFile.write(Ref[1])
            OutFile.write("\t")
            OutFile.write(Var[1])
            OutFile.write("\t")
            OutFile.write("%0.2f"%Change)
            OutFile.write("\n")

```

```

        if Change not in MEList:
            MEList = MEList + [Change]

    if Change == 0:
        NoChange = NoChange + 1
    elif Change < 0:
        NumInc = NumInc + 1
        if Change < Min:
            Min = Change
    elif Change > 0:
        NumDec = NumDec + 1
        if Change > Max:
            Max = Change

    InFile.close()
    ResultsFile.close()
    OutFile.close()

    General.Distribution(Min,Max,OutFileName,DistrFile,18)

    General.Log(LogF,"SSNS that DO NOT CHANGE MaxEnt score of"+Header+"splice-sites",NoChange)
    General.Log(LogF,"SSNS that DECREASE MaxEnt score of"+Header+"splice-sites",NumDec)
    General.Log(LogF,"SSNS that INCREASE MaxEnt score of"+Header+"splice-sites",NumInc)
    LogF.write("\n")
    General.Log(LogF,"Minimum change in MaxEnt score of"+Header+"caused by a SSNS",Min)
    General.Log(LogF,"Maximum change in MaxEnt score of"+Header+"caused by a SSNS",Max)

    return(OutFileName,sorted(MEList))

#####
# 5. Function: identifies SSNS that result in an extreme change in the #
#           maximum-entropy score of splice-sites as the SSNS that cause #
#           a change in the maximum-entropy score less than or greater #
#           than the chosen cut-off values #
# - Called by: Extreme_ME_In_Disease #
#####

def Extreme_ME(LogF,List,InFileName,DecFileName,IncFileName,Header):

    import General

    ExtrDec,ExtrInc = General.Cut_Off_Value(List)
    General.Log(LogF,"Extreme decrease in"+Header+"splice-site MaxEnt score cut-off
value",ExtrDec)
    General.Log(LogF,"Extreme increase in"+Header+"splice-site MaxEnt score cut-off
value",ExtrInc)

    print("Identifying SSNS that cause an extreme change in ME of"+Header+"splice-sites...")

```

```

print()

NumDec = 0
NumInc = 0
DecPosiDict = {}
IncPosiDict = {}
DecSSNSDict = {}
IncSSNSDict = {}

MEFile = open(InFileName, "r")
DecFile = open(DecFileName, "w")
IncFile = open(IncFileName, "w")

for Line in MEFile:
    Rec = (Line.rstrip()).split("\t")
    if float(Rec[18]) <= ExtrDec:
        NumDec = NumDec + 1
        DecFile.write(Line)
        if Rec[7]+">"+Rec[8] not in DecSSNSDict.keys():
            DecSSNSDict[Rec[7]+">"+Rec[8]] = 1
        else:
            DecSSNSDict[Rec[7]+">"+Rec[8]] = DecSSNSDict[Rec[7]+">"+Rec[8]] + 1
        if int(Rec[15]) not in DecPosiDict.keys():
            DecPosiDict[int(Rec[15])] = 1
        else:
            DecPosiDict[int(Rec[15])] = DecPosiDict[int(Rec[15])] + 1
    elif float(Rec[18]) >= ExtrInc:
        NumInc = NumInc + 1
        IncFile.write(Line)
        if Rec[7]+">"+Rec[8] not in IncSSNSDict.keys():
            IncSSNSDict[Rec[7]+">"+Rec[8]] = 1
        else:
            IncSSNSDict[Rec[7]+">"+Rec[8]] = IncSSNSDict[Rec[7]+">"+Rec[8]] + 1
        if int(Rec[15]) not in IncPosiDict.keys():
            IncPosiDict[int(Rec[15])] = 1
        else:
            IncPosiDict[int(Rec[15])] = IncPosiDict[int(Rec[15])] + 1

MEFile.close()
DecFile.close()
IncFile.close()

General.Log(LogF, "SSNS that cause an EXTREME DECREASE in MaxEnt of"+Header+"splice-
sites", NumDec)
General.Log(LogF, "SSNS that cause an EXTREME INCREASE in MaxEnt of"+Header+"splice-
sites", NumInc)
LogF.write("\n")

```

```

if len(DecSSNSDict) > 0:
    LogF.write("SSNS that cause an EXTREME DECREASE in MaxEnt of")
    LogF.write(Header)
    LogF.write("splice-sites\n")
    Write_Dict(LogF,DecPosiDict,DecSSNSDict,NumDec)
if len(IncSSNSDict) > 0:
    LogF.write("SSNS that cause an EXTREME INCREASE in MaxEnt of")
    LogF.write(Header)
    LogF.write("splice-sites\n")
    Write_Dict(LogF,IncPosiDict,IncSSNSDict,NumInc)

DecFileName = str(DecFile).split(" ")
IncFileName = str(IncFile).split(" ")
return(DecFileName[1][6:len(DecFileName[1])-1],IncFileName[1][6:len(IncFileName[1])-1])

#*****#
# 6. Function: determines whether any of the SSNS predicted to potentially #
#             result in an extreme change in the maximum-entropy scores have #
#             previously been associated with a trait or disease in GWAS      #
#             and/or are within genes known to be associated with disease in #
#             OMIM                                                            #
# - Called by: SynSNP.py                                                    #
#*****#

def Extreme_ME_In_Disease(Home):

    import os
    import winsound
    import time
    import General

    print("START:",time.ctime())
    print()

    General.Create_Dir(os.getcwd() + "\\Results\\4_Splice_sites")
    ResultsDir = os.getcwd()

    LogFile = open("Log_splice_sites.txt","w")

    AccFile,DonFile,AccResFile, DonResFile = MaxEntScan(LogFile,ResultsDir,Home)

    #~~~~~#
    # SSNS in 3' acceptor splice-sites                                     #
    #~~~~~#

    Heading = " 3' ACCEPTOR "

```

```

ResultFile,List =
Change_In_ME(LogFile,AccFile,AccResFile,"7_1_Acceptor_Change_In_ME.txt",Heading,"8_1_Acceptor_
Distribution.txt")

    DecFile,IncFile =
Extreme_ME(LogFile,List,ResultFile,"9_1_Acceptor_EXTREME_DECREASE.txt","9_2_Acceptor_EXTREME_I
NCREASE.txt",Heading)

    Num = General.GWAS_Or_OMIM(Home,"GWAS",DecFile,"10_1_Acceptor_Extreme_DECREASE_GWAS.txt")
    General.Log(LogFile,"SSNS that cause an EXTREME DECREASE in MaxEnt of 3' ACCEPTOR splice-
sites associated with a trait or disease in GWAS",Num)

    Num = General.GWAS_Or_OMIM(Home,"OMIM",DecFile,"10_2_Acceptor_Extreme_DECREASE_OMIM.txt")
    General.Log(LogFile,"SSNS that cause an EXTREME DECREASE in MaxEnt of 3' ACCEPTOR splice-
sites within genes known to be associated with disease in OMIM",Num)

    Num =
General.GWAS_And_OMIM("10_1_Acceptor_Extreme_DECREASE_GWAS.txt","10_2_Acceptor_Extreme_DECREAS
E_OMIM.txt","10_3_Acceptor_Extreme_DECREASE_OMIM_GWAS.txt")
    General.Log(LogFile,"SSNS that cause an EXTREME DECREASE in MaxEnt of 3' ACCEPTOR splice-
sites associated with a trait or disease in GWAS and within genes known to be associated with
disease in OMIM",Num)
    LogFile.write("\n")

    Num = General.GWAS_Or_OMIM(Home,"GWAS",IncFile,"11_1_Acceptor_Extreme_INCREASE_GWAS.txt")
    General.Log(LogFile,"SSNS that cause an EXTREME INCREASE in MaxEnt of 3' ACCEPTOR splice-
sites associated with a trait or disease in GWAS",Num)

    Num = General.GWAS_Or_OMIM(Home,"OMIM",IncFile,"11_2_Acceptor_Extreme_INCREASE_OMIM.txt")
    General.Log(LogFile,"SSNS that cause an EXTREME INCREASE in MaxEnt of 3' ACCEPTOR splice-
sites within genes known to be associated with disease in OMIM",Num)

    Num =
General.GWAS_And_OMIM("11_1_Acceptor_Extreme_INCREASE_GWAS.txt","11_2_Acceptor_Extreme_INCREAS
E_OMIM.txt","11_3_Acceptor_Extreme_INCREASE_OMIM_GWAS.txt")
    General.Log(LogFile,"SSNS that cause an EXTREME INCREASE in MaxEnt of 3' ACCEPTOR splice-
sites associated with a trait or disease in GWAS and within genes known to be associated with
disease in OMIM",Num)
    LogFile.write("\n")

    #~~~~~#
    # SSNS in 5' donor splice-sites #
    #~~~~~#

    Heading = " 5' DONOR "
    ResultFile,List =
Change_In_ME(LogFile,DonFile,DonResFile,"7_2_Donor_Change_In_ME.txt",Heading,"8_2_Donor_Distri
bution.txt")

```

```

DecFile,IncFile =
Extreme_ME(LogFile,List,ResultFile,"9_3_Donor_EXTREME_DECREASE.txt","9_4_Donor_EXTREME_INCREAS
E.txt",Heading)

Num = General.GWAS_Or_OMIM(Home,"GWAS",DecFile,"10_4_Donor_Extreme_DECREASE_GWAS.txt")
General.Log(LogFile,"SSNS that cause an EXTREME DECREASE in MaxEnt of 5' DONOR splice-
sites associated with a trait or disease in GWAS",Num)

Num = General.GWAS_Or_OMIM(Home,"OMIM",DecFile,"10_5_Donor_Extreme_DECREASE_OMIM.txt")
General.Log(LogFile,"SSNS that cause an EXTREME DECREASE in MaxEnt of 5' DONOR splice-
sites within genes known to be associated with disease in OMIM",Num)

Num =
General.GWAS_And_OMIM("10_4_Donor_Extreme_DECREASE_GWAS.txt","10_5_Donor_Extreme_DECREASE_OMIM
.txt","10_6_Donor_Extreme_DECREASE_OMIM_GWAS.txt")
General.Log(LogFile,"SSNS that cause an EXTREME DECREASE in MaxEnt of 5' DONOR splice-
sites associated with a trait or disease in GWAS and within genes known to be associated with
disease in OMIM",Num)
LogFile.write("\n")

Num = General.GWAS_Or_OMIM(Home,"GWAS",IncFile,"11_4_Donor_Extreme_INCREASE_GWAS.txt")
General.Log(LogFile,"SSNS that cause an EXTREME INCREASE in MaxEnt of 5' DONOR splice-
sites associated with a trait or disease in GWAS",Num)

Num = General.GWAS_Or_OMIM(Home,"OMIM",IncFile,"11_5_Donor_Extreme_INCREASE_OMIM.txt")
General.Log(LogFile,"SSNS that cause an EXTREME INCREASE in MaxEnt of 5' DONOR splice-
sites within genes known to be associated with disease in OMIM",Num)

Num =
General.GWAS_And_OMIM("11_4_Donor_Extreme_INCREASE_GWAS.txt","11_5_Donor_Extreme_INCREASE_OMIM
.txt","11_6_Donor_Extreme_INCREASE_OMIM_GWAS.txt")
General.Log(LogFile,"SSNS that cause an EXTREME INCREASE in MaxEnt of 5' DONOR splice-
sites associated with a trait or disease in GWAS and within genes known to be associated with
disease in OMIM",Num)
LogFile.write("\n")
LogFile.close()

os.chdir(Home)

winsound.Beep(700,1000)
print("END:",time.ctime())
print ()
print ("COMPLETE")
print ("-----")
print ("SSNS predicted to potentially cause an extreme change in MaxEnt ")
print ("have been saved in:")
print (ResultsDir)
print ()

```

```
#####
#                                     End of SpliceSites.pyw                                #
#####

#####
#                                     Functional.pyw                                    #
#####

#####
# This library contains all of the functions required to identify the sSNPs      #
# predicted to have a potential functional impact                                #
#####

#####
# 1. Function: identifies all of the sSNPs from the sSNS predicted to have a    #
#               potential functional impact                                     #
#               sSNPs = all sSNS at the same position in the chromosome and    #
#               transcribed from the same DNA strand                          #
# - Called by: Functional_SNPs()                                              #
#####

def Count_sSNPs(Dict,FileName,Funcnt,List):

    File = open(FileName,"r")

    for Line in File:
        Rec = (Line.rstrip()).split("\t")

        #~~~~~#
        # - Rec[0] = chromosome                                                  #
        # - Rec[1] = sSNP position in the chromosome                          #
        # - Rec[6] = DNA strand                                                  #
        #~~~~~#

        Id = Rec[0]+"."+Rec[1]+"."+Rec[6]

        if Id not in List:
            List = List + [Id]

        if Id not in Dict.keys():
            Dict[Id] = Funcnt
        elif Funcnt not in Dict[Id]:
            Dict[Id] = Dict[Id] + "," + Funcnt

    File.close()

    return(Dict,List)
```

```

#####
# Function: writes the sSNPs predicted to have a potential functional impact
#           to file
# - Called by: Functional_SNPs()
#####

def Write_To_File(FileName,List):

    File = open(FileName,"w")

    for sSNP in List:
        Col = sSNP.split(".")
        File.write(Col[0])
        File.write("\t")
        File.write(Col[1])
        File.write("\t")
        File.write(Col[2])
        File.write("\n")

    File.close()

#####
# 2. Function: Count the number of sSNPs predicted to have a potential
#               functional impact through an extreme change in:
#               > codon-usage frequency (CUF)
#               > minimum-free energy (MFE)
#               > maximum entropy (ME)
#               > codon-usage frequency and minimum-free energy (CUF,MFE)
#               > codon-usage frequency and maximum entropy (CUF,ME)
#               > minimum-free energy and maximum entropy (MFE,ME)
#               > codon-usage frequency, minimum-free energy and
#                   maximum entropy (CUF,MFE,ME)
# - Called by: Functional_SNPs()
#####

def Mechansims(Dict,LogF):

    CUF = 0
    MFE = 0
    ME = 0
    CUF_MFE = 0
    CUF_ME = 0
    MFE_ME = 0
    CUF_MFE_ME = 0

    File = open("4_sSNPs_Functional.txt","w")

```

```

for sSNP in Dict:
    Col = sSNP.split(".")
    File.write(Col[0])
    File.write("\t")
    File.write(Col[1])
    File.write("\t")
    File.write(Col[2])
    File.write("\t")
    File.write(Dict[sSNP])
    File.write("\n")

    FCUF = str(Dict[sSNP].find("CUF"))
    if FCUF != "-1":
        FCUF = "0"
    FMFE = str(Dict[sSNP].find("MFE"))
    if FMFE != "-1":
        FMFE = "0"
    FME = str(Dict[sSNP].find("ME"))
    if FME != "-1":
        FME = "0"
    Found = FCUF+FMFE+FME

    if Found == "0-1-1":
        CUF = CUF + 1
    elif Found == "-10-1":
        MFE = MFE + 1
    elif Found == "-1-10":
        ME = ME + 1
    elif Found == "00-1":
        CUF_MFE = CUF_MFE + 1
    elif Found == "0-10":
        CUF_ME = CUF_ME + 1
    elif Found == "-100":
        MFE_ME = MFE_ME + 1
    elif Found == "000":
        CUF_MFE_ME = CUF_MFE_ME + 1

File.close()

import General

General.Log(LogF,"sSNPs that alter CODON-USAGE FREQUENCY ",CUF)
General.Log(LogF,"sSNPs that alter MINIMUM FREE-ENERGY ",MFE)
General.Log(LogF,"sSNPs that alter MAXIMUM-ENTROPY ",ME)
General.Log(LogF,"sSNPs that alter CODON-USAGE FREQUENCY and MINIMUM FREE-ENERGY
",CUF_MFE)
General.Log(LogF,"sSNPs that alter CODON-USAGE FREQUENCY and MAXIMUM ENTROPY ",CUF_ME)

```

```

        General.Log(LogF,"sSNPs that alter MINIMUM FREE-ENERGY and MAXIMUM ENTROPY ",MFE_ME)
        General.Log(LogF,"sSNPs that alter CODON-USAGE FEEQUENCY, MINIMUM FREE-ENERGY and MAXIMUM-
ENTROPY ",CUF_MFE_ME)
        LogF.write("\n")

#*****#
# 3. Function: identifies sSNPs predicted to have a potential functional      #
#               impact, from the different files containing the sSNS predicted #
#               to have a potential functional impact                        #
# - Called by: SynSNP.py                                                    #
#*****#

def Functional_SNPs(Home):

    print ("*" * 46).center(80))
    print ("* 1. Codon-usage                                     *".center(80))
    print ("* 2. mRNA secondary structure and stability *".center(80))
    print ("* 3. Splice-sites                                     *".center(80))

    print ("*" * 46).center(80))
    print ()
    Choice = input("Enter choice as list (e.g. 123): ")

    import os
    import General
    import winsound

    HomeDir = os.getcwd()
    General.Create_Dir(HomeDir + "\\Results\\5_Functional_sSNPs")
    os.chdir(HomeDir + "\\Results\\5_Functional_sSNPs")

    LogFile = open("Log_Functional_sSNPs.txt", "w")

    FuncDict = {}

    if "1" in Choice:
        MechList = []
        CodonHome = Home + "\\Results\\2_Codon_usage\\"

        Codon = CodonHome + "2_1_Extreme_DECREASE_CUF.txt"
        FuncDict,MechList = Count_sSNPs(FuncDict,Codon,"CUF",MechList)

        Codon = CodonHome + "2_2_Extreme_INCREASE_CUF.txt"
        FuncDict,MechList = Count_sSNPs(FuncDict,Codon,"CUF",MechList)

        Write_To_File("1_sSNPs_CUF.txt",MechList)

```

```

        General.Log(LogFile,"sSNPs that potentially cause an extreme change in
CUF",len(MechList))

    if "2" in Choice:
        MechList = []
        mRNAHome = Home + "\\Results\\3_mRNA\\"

        mRNA = mRNAHome + "8_1_Extreme_DECREASE_MFE.txt"
        FunctDict,MechList = Count_sSNPs(FunctDict,mRNA,"MFE",MechList)

        mRNA = mRNAHome + "8_2_Extreme_INCREASE_MFE.txt"
        FunctDict,MechList = Count_sSNPs(FunctDict,mRNA,"MFE",MechList)

        Write_To_File("2_sSNPs_MFE.txt",MechList)

        General.Log(LogFile,"sSNPs that potentially cause an extreme change in
MFE",len(MechList))

    if "3" in Choice:
        MechList = []
        SpliceHome = Home + "\\Results\\4_Splice_sites\\"

        Splice = SpliceHome + "9_1_Acceptor_EXTREME_DECREASE.txt"
        FunctDict,MechList = Count_sSNPs(FunctDict,Splice,"ME",MechList)

        Splice = SpliceHome + "9_2_Acceptor_EXTREME_INCREASE.txt"
        FunctDict,MechList = Count_sSNPs(FunctDict,Splice,"ME",MechList)

        Splice = SpliceHome + "9_3_Donor_EXTREME_DECREASE.txt"
        FunctDict,MechList = Count_sSNPs(FunctDict,Splice,"ME",MechList)

        Splice = SpliceHome + "9_4_Donor_EXTREME_INCREASE.txt"
        FunctDict,MechList = Count_sSNPs(FunctDict,Splice,"ME",MechList)

        Write_To_File("3_sSNPs_ME.txt",MechList)

        General.Log(LogFile,"sSNPs that potentially cause an extreme change in
ME",len(MechList))

    LogFile.write("\n")
    General.Log(LogFile,"sSNPs predicted to potentially be functional through one or more
mechanisms",len(FunctDict))

    Mechansims(FunctDict,LogFile)

    LogFile.close()

    winsound.Beep(400,1000)

```

```

print ()
print ("COMPLETE")
print ("-----")
print ("The SSNS predicted to have a potential functional impact and associated with a
trait in GWAS and/or in genes associated with disease in OMIM have been saved in:")
print (os.getcwd())

#####
#                               End of Functional.pyw                               #
#####

#####
#                               Disease.pyw                                       #
#####

#####
# This library contains all of the functions required to identify sSNPs          #
# predicted to have a potential functional impact, which have previously been    #
# associated with a trait or disease in GWAS and/or occur within genes known    #
# to be associated with disease in OMIM                                         #
#####

#####
# 1. Function: joins the file containing the sSNPs predicted to have a          #
#               potential functional impact with the file containing the sSNPs    #
#               previously associated with a trait or disease in GWAS or with     #
#               the file containing the sSNPs within genes known to be          #
#               associated with disease, based on the sSNPs' positions in the    #
#               chromosomes                                                       #
# - Called by: Disease()                                                         #
#####

def GWAS_Or_OMIM(Home,Choice,sSNPFileName,OutFileName,LogF):

    import General

    if Choice == "GWAS":
        File = open(Home+"\\Results\\1_sSNPs\\4_1_sSNPs_in_GWAS.txt","r")
    elif Choice == "OMIM":
        File = open(Home+"\\Results\\1_sSNPs\\5_1_sSNPs_in_OMIM.txt","r")

    Num = 0
    Dict = {}
    CountDict = {}

```

```

#~~~~~#
# Save sSNP data in a dictionary #
#   - Rec[0] = chromosome #
#   - Rec[1] = SNP start position #
#   - Rec[3] = potential molecular mechanism #
#~~~~~#

InFile = open(sSNPFileName,"r")
for Line in InFile:
    Rec = (Line.rstrip()).split("\t")
    Chrom = Rec[0].split("_")
    Chrom = Chrom[0]
    if Chrom not in Dict.keys():
        Dict[Chrom] = {Rec[1]:Rec[3]}
    elif Rec[1] not in Dict[Chrom].keys():
        Dict[Chrom][Rec[1]] = Rec[3]
InFile.close()

#~~~~~#
# Loop over GWAS/OMIM file and join to sSNP records based on sSNP position #
#   - Rec[0] = chromosome #
#   - Rec[1] = sSNP start position #
#   - Rec[2] = gene associated with a trait/disease #
#~~~~~#

OutFile = open(OutFileName,"w")
for Line in File:
    Rec = (Line.rstrip()).split("\t")
    Chrom = Rec[0].split("_")
    Chrom = Chrom[0]
    if Chrom in Dict.keys():
        if Rec[1] in Dict[Chrom].keys():
            Num = Num + 1
            OutFile.write(Chrom)
            OutFile.write("\t")
            OutFile.write(Rec[1])
            OutFile.write("\t")
            OutFile.write(Dict[Chrom][Rec[1]])
            OutFile.write("\t")
            OutFile.write(Rec[2])
            OutFile.write("\n")
        if Dict[Chrom][Rec[1]] not in CountDict.keys():
            CountDict[Dict[Chrom][Rec[1]]] = 1
        else:
            CountDict[Dict[Chrom][Rec[1]]] = CountDict[Dict[Chrom][Rec[1]]] + 1

InFile.close()
OutFile.close()

```

```

General.Log(LogF,"Potentially functional sSNPs in "+Choice,Num)
for Mech in CountDict:
    LogF.write("\t")
    LogF.write(Mech)
    LogF.write("\t")
    LogF.write(str(CountDict[Mech]))
    LogF.write("\n")
LogF.write("\n")

#####
# 2. Function: joins the file containing the sSNPs predicted to have a      #
#                potential functional impact that have previously been      #
#                associated with a trait or disease in GWAS and the file     #
#                containing the sSNPs predicted to have a potential functional #
#                impact that are within genes known to be associated with    #
#                disease in OMIM, based on the sSNPs' positions in the      #
#                chromosomes                                                  #
# - Called by: Disease()                                                    #
#####

def GWAS_And_OMIM(GWASFileName,OMIMFileName,OutFileName,LogF):

    import General

    #~~~~~#
    # Save GWAS data in a dictionary                                         #
    # - Rec[0] = chromosome                                                  #
    # - Rec[1] = SNP start position                                         #
    # - Rec[3] = trait/disease                                              #
    #~~~~~#

    NumGWASOMIM = 0
    GWASOMIMDict = {}

    GWASFile = open(GWASFileName,"r")

    for Line in GWASFile:
        Rec = (Line.rstrip()).split("\t")

        Id = Rec[0] + Rec[1]

        if Id not in GWASOMIMDict.keys():
            GWASOMIMDict[Id] = Rec[3]

    GWASFile.close()

```

```

#~~~~~#
# Loop over OMIM file and join to GWAS records based on SNP position #
# - Rec[0] = chromosome #
# - Rec[1] = sSNP start position #
# - Rec[2] = potential molecular mechanism #
# - Rec[3] = genes associated with disease in OMIM #
#~~~~~#

CountDict = {}

OMIMFile = open(OMIMFileName,"r")
OutFile = open(OutFileName,"w")

for Line in OMIMFile:
    Rec = (Line.rstrip()).split("\t")
    Id = Rec[0] + Rec[1]
    if Id in GWASOMIMDict:
        NumGWASOMIM = NumGWASOMIM + 1
        for i in range(0,3):
            OutFile.write(Rec[i])
            OutFile.write("\t")
        OutFile.write(str(GWASOMIMDict[Id]))
        OutFile.write("\t")
        OutFile.write(Rec[3])
        OutFile.write("\n")
        if Rec[2] not in CountDict:
            CountDict[Rec[2]] = 1
        else:
            CountDict[Rec[2]] = CountDict[Rec[2]] + 1

OMIMFile.close()
OutFile.close()

General.Log(LogF,"Potentially functional sSNPs in GWAS and OMIM",NumGWASOMIM)
for Mech in CountDict:
    LogF.write("\t")
    LogF.write(Mech)
    LogF.write("\t")
    LogF.write(str(CountDict[Mech]))
    LogF.write("\n")
LogF.write("\n")

```

```

#####
# 3. Function: identifies which of the sSNPs predicted to potentially have a #
#           functional impact through one or more of the three molecular #
#           mechanisms investigated have previously been associated with a #
#           trait or disease in GWAS and/or are within genes known to be #
#           associated with disease in OMIM #
# - Called by: SynSNP.py #
#####

def Disease(Home):

    import time
    import os
    import General
    import winsound

    print("START:",time.ctime())
    print()
    General.Create_Dir(Home + "\\Results\\6_Disease")
    LogFile = open(os.getcwd() + "\\Log_Disease.txt","w")

    print ("Identifying potentially functional sSNPs in GWAS...")
    print ()

    GWAS_Or_OMIM(Home,"GWAS",Home+"\\Results\\5_Functional_sSNPs\\4_sSNPs_Functional.txt","1_Funct
ional_GWAS.txt",LogFile)

    print ("Identifying potentially functional sSNPs in OMIM...")
    print ()

    GWAS_Or_OMIM(Home,"OMIM",Home+"\\Results\\5_Functional_sSNPs\\4_sSNPs_Functional.txt","2_Funct
ional_OMIM.txt",LogFile)

    print ("Identifying potentially functional sSNPs in GWAS and OMIM...")
    print ()

    GWAS_And_OMIM("1_Functional_GWAS.txt","2_Functional_OMIM.txt","3_Functional_GWAS_and_OMIM.txt"
,LogFile)

    LogFile.close()

    winsound.Beep(700,1000)
    print("END:",time.ctime())
    print ()
    print ("COMPLETE")
    print ("-----")
    print ("The sSNPs in GWAS and OMIM have been saved in:")
    print (os.getcwd())

```

```

print ()

os.chdir(Home)

#####
#                               End of Disease.pyw                               #
#####

#####
#                               General.pyw                                   #
#####

#####
# This library contains the functions that are frequently used in SynSNP      #
#####

#####
# Function: creates the required directory to store the result files          #
#####

def Create_Dir(Dir):

    import os

    if not os.path.exists(Dir):
        os.makedirs(Dir)
    os.chdir(Dir)

#####
# Function: writes the results to a log file                                  #
#####

def Log(File,Msg,Num):

    import time

    File.write(time.ctime())
    File.write("\t")
    File.write(Msg)
    File.write("\t")
    File.write(str(Num))
    File.write("\n")

```

```

#####
# Function: creates the frequency distribution of the sSNS predicted to      #
#           potentially cause an extreme change in codon-usage frequency,    #
#           minimum-free energy of local mRNA secondary structures or        #
#           maximum entropy score of splice-sites                          #
#####

def Distribution(Min,Max,InputFileName,OutputFileName,Col):

    print("Creating frequency distribution...")
    print()

    Dict = {}
    for i in range((int(Min)-1)*10, (int(Max)+1)*10):
        Dict[i/10] = 0

    InputFile = open(InputFileName,"r")
    for Line in InputFile:
        Rec = (Line.rstrip()).split("\t")
        Dict[round(float(Rec[Col]),1)] = Dict[round(float(Rec[Col]),1)] + 1
    InputFile.close()

    SortDict = sorted(Dict.keys())
    OutputFile = open(OutputFileName,"w")

    for i in SortDict:
        OutputFile.write(str(i))
        OutputFile.write("\t")
        OutputFile.write(str(Dict[i]))
        OutputFile.write("\n")
    OutputFile.close()

#####
# Function: uses the frequency distribution to identify the 10% most extreme  #
#           decrease and the 10% most extreme increase in codon-usage       #
#           frequency, minimum-free energy of local mRNA secondary structures #
#           or maximum-entropy score of splice-sites                        #
#####

def Cut_Off_Value(List):

    print("Calculating extreme cut-off values...")
    print()

    CutOff = int((len(List)/100)*10)
    if CutOff < 1:
        CutOff = 1

```

```

MinCutOff = List[CutOff-1]
MaxCutOff = List[len(List)-CutOff]

return(MinCutOff,MaxCutOff)

#####
# Function: identifies the sSNS predicted to have a potential functional      #
#          impact, which have previously been associated with a trait or      #
#          disease in GWAS OR occur within genes known to be associated with #
#          disease in OMIM                                                    #
#####

def GWAS_Or_OMIM(Home,Choice,SSNSFileName,OutFileName):

    if Choice == "GWAS":
        File = open(Home+"\\Results\\1_sSNPs\\4_1_sSNPs_in_GWAS.txt","r")
    elif Choice == "OMIM":
        File = open(Home+"\\Results\\1_sSNPs\\5_1_sSNPs_in_OMIM.txt","r")

    Num = 0
    Dict = {}

    #####
    # Save GWAS/OMIM data in a dictionary                                     #
    # - Rec[0] = chromosome                                                    #
    # - Rec[1] = SNP start position                                           #
    # - Rec[2] = trait/disease                                                #
    #####

    for Line in File:
        Rec = (Line.rstrip()).split("\t")
        if Rec[0] not in Dict.keys():
            Dict[Rec[0]] = {Rec[1]:Rec[2]}
        else:
            Dict[Rec[0]][Rec[1]] = Rec[2]
    File.close()

    #####
    # Loop over sSNS file and join to GWAS/OMIM records based on sSNS position #
    # - Rec[0] = chromosome                                                    #
    # - Rec[1] = sSNP start position                                           #
    #####

    InFile = open(SSNSFileName,"r")
    OutFile = open(OutFileName,"w")
    for Line in InFile:
        Rec = (Line.rstrip()).split("\t")
        Chrom = Rec[0].split("_")

```

```

    Chrom = Chrom[0]
    if Chrom in Dict.keys():
        if Rec[1] in Dict[Chrom].keys():
            Num = Num + 1
            OutFile.write(Chrom)
            OutFile.write("\t")
            OutFile.write(Rec[1])
            OutFile.write("\t")
            OutFile.write(Dict[Chrom][Rec[1]])
            OutFile.write("\n")

    InFile.close()
    OutFile.close()

    return(Num)

#*****#
# Function: identifies the sSNS predicted to have a potential functional      #
#           impact, which have previously been associated with a trait or      #
#           disease in GWAS AND occur within genes known to be associated with #
#           disease in OMIM                                                    #
#*****#

def GWAS_And_OMIM(GWASFileName, OMIMFileName, OutFileName):

    #~~~~~#
    # Save GWAS data in a dictionary                                           #
    #   - Rec[0] = chromosome                                                    #
    #   - Rec[1] = sSNS start position                                          #
    #   - Rec[2] = trait/disease                                                #
    #~~~~~#

    NumGWASOMIM = 0
    GWASOMIMDict = {}

    GWASFile = open(GWASFileName, "r")

    for Line in GWASFile:
        Rec = (Line.rstrip()).split("\t")

        if Rec[0] not in GWASOMIMDict.keys():
            GWASOMIMDict[Rec[0]] = {Rec[1]: Rec[2]}
        else:
            GWASOMIMDict[Rec[0]][Rec[1]] = Rec[2]

    GWASFile.close()

```

```

# ~~~~~#
# Loop over OMIM file and join to GWAS records based on SNP position #
# - Rec[0] = chromosome #
# - Rec[1] = sSNS start position #
# ~~~~~#

OMIMFile = open(OMIMFileName, "r")
OutFile = open(OutFileName, "w")
for Line in OMIMFile:
    Rec = (Line.rstrip()).split("\t")
    if Rec[0] in GWASOMIMDict.keys():
        if Rec[1] in GWASOMIMDict[Rec[0]].keys():
            NumGWASOMIM = NumGWASOMIM + 1
            OutFile.write(Rec[0])
            OutFile.write("\t")
            OutFile.write(Rec[1])
            OutFile.write("\t")
            OutFile.write(GWASOMIMDict[Rec[0]][Rec[1]])
            OutFile.write("\t")
            OutFile.write(Rec[2])
            OutFile.write("\n")

OMIMFile.close()
OutFile.close()

return(NumGWASOMIM)

# ~~~~~#
# Function: removes redundant records #
# Redundant records = same chromosome, start, end and strand #
# - Called by: Create_Chrom_Files() #
# ~~~~~#

def Remove_Redundancies(InFileName, NumRec, Mech, LogF):

    import General

    print ("Removing redundant records...")
    print ()

    InFile = open(InFileName, "r")
    RedFile = open("2_1_Redundant_"+Mech+".txt", "w")

    Dict = {}
    NumNonRed=NumRed = 0
    for i in range(0, NumRec):
        Index = InFile.tell()
        Line = InFile.readline()

```

```

Rec = (Line.rstrip()).split("\t")
Id = Rec[1] + Rec[2] + Rec[4] # Exon start + Exon end + Exon strand
Chrom = (Rec[0].strip('')).split("_")
if Chrom[0] not in Dict.keys():
    NumNonRed = NumNonRed + 1
    Dict[Chrom[0]] = {Id:Index}
else:
    if Id not in Dict[Chrom[0]].keys():
        NumNonRed = NumNonRed + 1
        Dict[Chrom[0]][Id] = Index
    else:
        NumRed = NumRed + 1
        RedFile.write(Line)

General.Log(LogF,Mech+" records checked for redundancies",NumRec)
General.Log(LogF,"Redundant "+Mech+" records removed",NumRed)
General.Log(LogF,"NON-REDUNDANT"+Mech+" records",NumNonRed)
LogF.write("\n")

InFile.close()
RedFile.close()

return(Dict) # Dict[Chromosome] = {Id:position in file}

#####
# Function: separates the non-redundant file into different files based on the #
#           chromosomes                                                    #
# - Called by: Join_Files()                                              #
#####

def Create_Chrom_Files(ResultsHome,InFileName,NumRec,Mech,LogF):

    Dict = Remove_Redundancies(InFileName,NumRec,Mech,LogF)

    import os
    import General

    InFile = open(InFileName,"r")

    print("Creating chromosome files...")
    print()

    # Dict[Chromosome] = {Id:position in file}
    # Id = Exon start + Exon end + Exon strand
    # SortedDict = List of chromosomes

    # Create directory to store chromosome files
    Create_Dir(os.getcwd() + "\\ "+Mech+"_chroms")

```

```

# Get list of chromosomes
SortedDict = sorted(Dict)
for Chrom in SortedDict:
    # Create a separate file for each chromosome
    OutFile = open(Chrom+"_"+Mech+".txt", "w")
    # Write all the SSNS on that chromosome to file
    for Id in Dict[Chrom]:
        InFile.seek(Dict[Chrom][Id])
        OutFile.write(InFile.readline())
    OutFile.close()
    General.Log(LogF, Mech+" on "+Chrom, len(Dict[Chrom]))
LogF.write("\n")

InFile.close()

os.chdir(ResultsHome)

return(SortedDict) # SortedDict = List of chromosomes

#####
# Function: Identify sSNS in CDS/exons #
#           sSNS records are joined to CDS/exon records if the sSNS is within #
#           the start and end of the CDS/exon #
# - Called by: Check_Alleles() #
#####

def Join_Files(Home, ResultsHome, InFileName, NumRec, Mech, LogF):

    ChromList = Create_Chrom_Files(ResultsHome, InFileName, NumRec, Mech, LogF)

    import General
    import os

    print("Identifying SSNS in "+Mech+"...")
    print()

    NumJoined = 0
    NumNotJoined = 0
    TotalSeq = 0
    TotalSSNS = 0
    TotalJoined = 0

    NotJoinedFile = open("3_1_SSNS_not_in_"+Mech+".txt", "w")
    JoinFile = open("3_2_SSNS_in_"+Mech+".txt", "w")

    for Chrom in ChromList:
        if os.path.exists(Home+"\\Results\\1_sSNPs\\SSNS_chroms\\"+Chrom+"_SSNS.txt"):
            SeqFile = open(os.getcwd()+"\\"+Mech+"_chroms\\"+Chrom+"_"+Mech+".txt", "r")

```

```

NumSeq = 0
SeqList = []
for Line in SeqFile:
    NumSeq = NumSeq + 1
    TotalSeq = TotalSeq + 1
SeqFile.seek(0)
for i in range(0,NumSeq):
    Index = SeqFile.tell()
    Rec = ((SeqFile.readline()).rstrip()).split("\t")
    SeqList = SeqList + [[int(Rec[1]),int(Rec[2]),Index]]
SeqList = sorted(SeqList) # List = [Exon start,Exon end,Position in file]

# Open SSNS file for this chromosome

SSNSFile = open(Home+"\\Results\\1_sSNPs\\SSNS_chroms\\"+Chrom+"_SSNS.txt","r")

for Line in SSNSFile:
    TotalSSNS = TotalSSNS + 1
    Rec = (Line.rstrip()).split("\t")

    SSNSposi = int(Rec[1])
    i = 0
    Found = 0
    while i < len(SeqList) and SSNSposi >= SeqList[i][0]:
        if SSNSposi >= SeqList[i][0] and SSNSposi < SeqList[i][1]:
            TotalJoined = TotalJoined + 1
            Found = Found + 1
            for j in range(0,9):
                JoinFile.write(Rec[j])
                JoinFile.write("\t")
            SeqFile.seek(SeqList[i][2])
            JoinFile.write(SeqFile.readline())
            i = i + 1
        if Found != 0:
            NumJoined = NumJoined + 1
        elif Found == 0:
            NumNotJoined = NumNotJoined + 1
            NotJoinedFile.write(Line)

    SSNSFile.close()
    SeqFile.close()

NotJoinedFile.close()
JoinFile.close()

General.Log(LogF,"Synonymous single nucleotide substitutions (SSNS)",TotalSSNS)
General.Log(LogF,Mech,TotalSeq)
General.Log(LogF,"SSNS not in "+Mech,NumNotJoined)

```

```

General.Log(LogF, "SSNS IN "+Mech, NumJoined)
General.Log(LogF, "Total SSNS in "+Mech+" records", TotalJoined)
LogF.write("\n")

FileName = str(JoinFile).split(" ")
return (FileName[1][6:len(FileName[1])-1])

#*****#
# Function: corrects the sSNS reference allele according to the transcript #
# strand #
# - Rec[1] = sSNS start position #
# - Rec[6] = sSNS strand #
# - Rec[7] = sSNS reference allele #
# - Rec[8] = sSNS variant allele #
# - Rec[10] = CDS/exon start position #
# - Rec[13] = CDS/exon strand #
# - Rec[14] = CDS/exon sequence #
#*****#

def Check_Alleles(Home, ResultsHome, InFileName, NumRec, Mech, LogF):

    InFileName = Join_Files(Home, ResultsHome, InFileName, NumRec, "CDS", LogF)

    import General

    NumRec = 0
    NoErr = 0
    Err = 0
    Compl = {"A": "T", "T": "A", "C": "G", "G": "C"}
    ComplStr = {"+": "-", "-": "+"}

    InFile = open(InFileName, "r")
    ErrFile = open("4_1_Reference_allele_not_in_"+Mech+".txt", "w")
    NoErrFile = open("4_2_Correct_SSNS_in_"+Mech+".txt", "w")

    for Line in InFile:
        NumRec = NumRec + 1
        Rec = (Line.rstrip()).split("\t")

        if Rec[6] != Rec[13]:
            Rec[7] = Compl[Rec[7]]
            Rec[8] = Compl[Rec[8]]
            Rec[6] = ComplStr[Rec[6]]

        if Rec[6] == "+" and Rec[7] != (Rec[14][int(Rec[1])-int(Rec[10])]).upper():
            Err = Err + 1
            ErrFile.write("Reference allele not in "+Mech+"\t")
            ErrFile.write(Line)

```

```

elif Rec[6] == "-" and Rec[7] != (Rec[14][int(Rec[11])-int(Rec[1])-1]).upper():
    Err = Err + 1
    ErrFile.write("Reference allele not in "+Mech+"\t")
    ErrFile.write(Line)
else:
    NoErr = NoErr + 1
    for i in range(0,9):
        NoErrFile.write(Rec[i])
        NoErrFile.write("\t")
    for i in range(10,13):
        NoErrFile.write(Rec[i])
        NoErrFile.write("\t")
    NoErrFile.write(Rec[14])
    NoErrFile.write("\n")

InFile.close()
ErrFile.close()
NoErrFile.close()

General.Log(LogF,"SSNS in "+Mech,NumRec)
General.Log(LogF,"reference alleles not in "+Mech,Err)
General.Log(LogF,"SSNS REFERENCE ALLELES IN "+Mech,NoErr)
LogF.write("\n")

FileName = str(NoErrFile).split(" ")
return (FileName[1][6:len(FileName[1])-1])

#####
#                                     End of General.pyw                                #
#####

```

APPENDIX C

Table C.1: The human codon-usage frequency table ²

Amino acid	Codon	Codon-usage frequency per one thousand	Codon-usage frequency as a percentage (%)
Ala	GCA	15.82	22.80
Ala	GCC	27.73	39.97
Ala	GCG	7.37	10.62
Ala	GCT	18.45	26.59
Arg	AGA	12.17	21.47
Arg	AGG	11.96	21.10
Arg	CGA	6.17	10.88
Arg	CGC	10.42	18.38
Arg	CGG	11.42	20.14
Arg	CGT	4.54	8.00
Asn	AAC	19.1	52.96
Asn	AAT	16.96	47.03
Asp	GAC	25.1	53.54
Asp	GAT	21.78	46.45
Cys	TGC	12.62	54.39
Cys	TGT	10.58	45.60
End	TAA	0.99	29.64
End	TAG	0.79	23.65
End	TGA	1.56	46.70
Gln	CAA	12.34	26.49
Gln	CAG	34.23	73.50
Glu	GAA	28.96	42.24
Glu	GAG	39.59	57.75
Gly	GGA	16.471	24.98
Gly	GGC	22.22	33.71
Gly	GGG	16.47	24.98
Gly	GGT	10.75	16.30
His	CAC	15.09	58.15
His	CAT	10.86	41.84
Ile	ATA	7.49	16.90
Ile	ATC	20.82	46.98
Ile	ATT	16	36.10
Leu	CTA	7.15	7.13
Leu	CTC	19.59	19.55
Leu	CTG	39.64	39.57
Leu	CTT	13.19	13.16
Leu	TTA	7.67	7.65
Leu	TTG	12.93	12.90
Lys	AAA	24.44	43.41
Lys	AAG	31.86	56.58
Met	ATG	22.04	100.00
Phe	TTC	20.28	53.57
Phe	TTT	17.57	46.42
Pro	CCA	16.92	27.66
Pro	CCC	19.79	32.35
Pro	CCG	6.92	11.31
Pro	CCT	17.54	28.67
Ser	AGC	19.46	23.99
Ser	AGT	12.13	14.95
Ser	TCA	12.21	15.05
Ser	TCC	17.68	21.79
Ser	TCG	4.41	5.43
Ser	TCT	15.22	18.76
Thr	ACA	15.11	28.41
Thr	ACC	18.89	35.52
Thr	ACG	6.05	11.37
Thr	ACT	13.12	24.67
Trp	TGG	13.17	100.00
Tyr	TAC	15.31	55.67
Tyr	TAT	12.19	44.32
Val	GTA	7.08	11.66
Val	GTC	14.46	23.82
Val	GTG	28.12	46.33
Val	GTT	11.03	18.17

² (<http://www.kazusa.or.jp/codon/cgi-bin/showcodon.cgi?species=9606>)

APPENDIX D

Table D.1: The frequency of sSNS changes predicted to potentially cause an extreme decrease in codon-usage frequency

sSNS change	Number of sSNS	Percentage of s SNS (%)
G>A	4,532	40.44
C>T	2,170	19.36
C>G	1,857	16.57
G>C	1,077	9.61
G>T	904	8.07
C>A	547	4.88
T>A	121	1.07
Total	11,208	100.00

Table D.2: The frequency of sSNS changes predicted to potentially cause an extreme increase in codon-usage frequency

sSNS change	Number of sSNS	Percentage of sSNS (%)
A>G	2,368	34.87
G>C	1,748	25.73
C>G	1,057	15.56
T>C	917	13.5
T>G	396	5.83
A>C	162	2.38
A>T	145	2.13
Total	6,793	100.00

Table D.3: The frequency of sSNS changes in the 100 base mRNA sequences

SSNS change	Number of sSNS	Percentage of sSNS (%)
C>T	32,734	32.04
G>A	22,199	21.73
T>C	12,747	12.48
A>G	11,174	10.94
C>G	4,557	4.46
C>A	4,227	4.14
G>C	3,800	3.72
G>T	3,175	3.11
A>C	2,498	2.44
T>G	1,896	1.86
A>T	1,675	1.63
T>A	1,487	1.45
Total	102,169	100.00

Table D.4: The frequency of sSNS changes predicted to potentially cause an extreme decrease in the minimum free-energy of local mRNA secondary structures

sSNS change	Number of sSNS	Percentage of sSNS (%)
A>G	824	30.37
C>T	753	27.76
C>G	417	15.37
G>C	168	6.19
T>G	142	5.23
A>C	108	3.98
C>A	99	3.65
A>T	69	2.54
G>T	65	2.40
T>A	38	1.40
T>C	30	1.11
Total	2,713	100.00

Table D.5: The frequency of sSNS changes predicted to potentially cause an extreme increase in the minimum free-energy of local mRNA secondary structures

sSNS change	Number of sSNS	Percentage of sSNS (%)
G>A	1,348	55.20
G>C	231	9.46
C>A	194	7.94
T>C	187	7.66
G>T	175	7.17
C>G	128	5.24
T>A	50	2.05
C>T	40	1.64
A>C	40	1.64
A>T	25	1.02
T>G	24	0.98
Total	2,442	100.00

Table D.6: The frequency of sSNS changes in the exonic part of the 3' acceptor splice-sites

sSNS change	Number of sSNS	Percentage of sSNS (%)
C>T	279	24.62
G>A	257	22.68
T>C	160	14.12
A>G	105	9.27
T>G	56	4.94
C>G	51	4.50
G>T	51	4.50
G>C	45	3.97
C>A	43	3.80
A>T	31	2.74
A>C	29	2.56
T>A	26	2.30
Total	1,133	100.00

Table D.7: The frequency of sSNS changes in the exonic part of the 5' donor splice-sites

sSNS change	Number of sSNS	Percentage of sSNS (%)
G>A	313	31.18
C>T	269	26.79
T>C	125	12.45
A>G	97	9.67
G>C	43	4.28
A>C	37	3.69
G>T	30	2.99
C>G	27	2.69
C>A	20	1.99
A>T	15	1.49
T>G	15	1.49
T>A	13	1.29
Total	1,004	100.00

Table D.8: The frequency of sSNS changes predicted to potentially cause an extreme decrease in the maximum-entropy score of 3' acceptor splice-sites

sSNS change	Number of sSNS	Percentage of sSNS (%)
G>A	17	28.33
T>C	17	28.33
G>T	11	18.33
G>C	6	10.00
A>T	3	5.00
T>A	3	5.00
T>G	2	3.33
C>T	1	1.68
Total	60	100.00

Table D.9: The frequency of sSNS changes predicted to potentially cause an extreme increase in the maximum-entropy score of 3' acceptor splice-sites

sSNS change	Number of sSNS	Percentage of sSNS (%)
C>T	44	59.46
A>G	14	18.92
C>G	4	5.41
G>T	4	5.41
A>T	3	4.05
T>G	2	2.70
T>C	2	2.70
G>A	1	1.35
Total	74	100.00

Table D.10: The frequency of sSNS changes predicted to potentially cause an extreme decrease in the maximum-entropy score of 5' donor splice-sites

sSNS change	Number of sSNS	Percentage of sSNS (%)
G>A	59	86.76
G>T	4	5.88
G>C	3	4.42
A>G	1	1.47
C>T	1	1.47
Total	68	100.00

Table D.11: The frequency of sSNS changes predicted to potentially cause an extreme increase in the maximum-entropy score of 5' donor splice-sites

sSNS change	Number of sSNS	Percentage of sSNS (%)
T>C	31	45.59
A>G	16	23.53
G>A	12	17.65
G>C	7	10.29
C>G	1	1.47
T>G	1	1.47
Total	68	100.00

REFERENCES

- ACTON, S., OSGOOD, D., DONOGHUE, M., CORELLA, D., POCOVI, M., CENARRO, A., MOZAS, P., KEILTY, J., SQUAZZO, S., WOOLF, E. A. & ORDOVAS, J. M. (1999) Association of polymorphisms at the SR-BI gene locus with plasma lipid levels and body mass index in a white population. *Arterioscler Thromb Vasc Biol*, 19, 1734-43.
- AMBERGER, J., BOCCHINI, C. & HAMOSH, A. (2011) A new face and new challenges for Online Mendelian Inheritance in Man (OMIM(R)). *Hum Mutat*, 32, 564-7.
- ANFENSEN, C. B. (1973) Principles that govern the folding of protein chains. *Science*, 181, 223-30.
- BAKER, M. (2010) Genomics: The search for association. *Nature*, 467, 1135-8.
- BARKER, J. N. (1998) Pathogenesis of psoriasis. *J Dermatol*, 25, 778-81.
- BARNES, G. (2003) *Bioinformatics for Geneticists*, West Sussex, England, John Wiley and Sons Ltd.
- BIRO, J. C. (2008) Correlation between nucleotide composition and folding energy of coding sequences with special attention to wobble bases. *Theor Biol Med Model*, 5, 14.
- BLACK, D. L. (2003) Mechanisms of alternative pre-messenger RNA splicing. *Annu Rev Biochem*, 72, 291-336.
- BOULLING, A., LE GAC, G., DUJARDIN, G., CHEN, J. M. & FEREC, C. (2010) The c.1275A>G putative chronic pancreatitis-associated synonymous polymorphism in the glycoprotein 2 (GP2) gene decreases exon 9 inclusion. *Mol Genet Metab*, 99, 319-24.
- BULMER, M. (1987) Coevolution of codon usage and transfer RNA abundance. *Nature*, 325, 728-30.
- CAO, G., ZHAO, L., STANGL, H., HASEGAWA, T., RICHARDSON, J. A., PARKER, K. L. & HOBBS, H. H. (1999) Developmental and hormonal regulation of murine scavenger receptor, class B, type 1. *Mol Endocrinol*, 13, 1460-73.
- CAPON, F., ALLEN, M. H., AMEEN, M., BURDEN, A. D., TILLMAN, D., BARKER, J. N. & TREMBATH, R. C. (2004) A synonymous SNP of the corneodesmosin gene leads to increased mRNA stability and demonstrates association with psoriasis across diverse ethnic groups. *Hum Mol Genet*, 13, 2361-8.
- CARTEGNI, L., CHEW, S. L. & KRAINER, A. R. (2002) Listening to silence and understanding nonsense: exonic mutations that affect splicing. *Nat Rev Genet*, 3, 285-98.
- CARTEGNI, L., WANG, J., ZHU, Z., ZHANG, M. Q. & KRAINER, A. R. (2003) ESEfinder: A web resource to identify exonic splicing enhancers. *Nucleic Acids Res*, 31, 3568-71.
- CHABOT, B. (1996) Directing alternative splicing: cast and scenarios. *Trends Genet*, 12, 472-8.
- CHAMARY, J. V., PARMLEY, J. L. & HURST, L. D. (2006) Hearing silence: non-neutral evolution at synonymous sites in mammals. *Nat Rev Genet*, 7, 98-108.
- CHEN, R., DAVYDOV, E. V., SIROTA, M. & BUTTE, A. J. (2010) Non-synonymous and synonymous coding SNPs show similar likelihood and effect size of human disease association. *PLoS One*, 5, e13574.
- CONSORTIUM, T. G. P. (2010) A map of human genome variation from population-scale sequencing. *Nature*, 467, 1061-73.

- CONSTANTINEAU, J., GREASON, E., WEST, M., FILBIN, M., KIEFT, J. S., CARLETTI, M. Z., CHRISTENSON, L. K. & RODRIGUEZ, A. (2009) A synonymous variant in scavenger receptor, class B, type I gene is associated with lower SR-BI protein expression and function. *Atherosclerosis*, 210, 177-82.
- CORDON-CARDO, C., O'BRIEN, J. P., BOCCIA, J., CASALS, D., BERTINO, J. R. & MELAMED, M. R. (1990) Expression of the multidrug resistance gene product (P-glycoprotein) in human normal and tumor tissues. *J Histochem Cytochem*, 38, 1277-87.
- DESMET, F. O., HAMROUN, D., LALANDE, M., COLLOD-BEROUD, G., CLAUSTRES, M. & BEROUD, C. (2009) Human Splicing Finder: an online bioinformatics tool to predict splicing signals. *Nucleic Acids Res*, 37, e67.
- DIATCHENKO, L., SLADE, G. D., NACKLEY, A. G., BHALANG, K., SIGURDSSON, A., BELFER, I., GOLDMAN, D., XU, K., SHABALINA, S. A., SHAGIN, D., MAX, M. B., MAKAROV, S. S. & MAIXNER, W. (2005) Genetic basis for individual variations in pain perception and the development of a chronic pain condition. *Hum Mol Genet*, 14, 135-43.
- DITTMAR, K. A., GOODENBOUR, J. M. & PAN, T. (2006) Tissue-specific differences in human transfer RNA expression. *PLoS Genet*, 2, e221.
- DOBROWOLSKI, S. F., ANDERSEN, H. S., DOKTOR, T. K. & ANDRESEN, B. S. (2010) The phenylalanine hydroxylase c.30C>G synonymous variation (p.G10G) creates a common exonic splicing silencer. *Mol Genet Metab*, 100, 316-23.
- DUAN, J. & ANTEZANA, M. A. (2003) Mammalian mutation pressure, synonymous codon choice, and mRNA degradation. *J Mol Evol*, 57, 694-701.
- DUAN, J., WAINWRIGHT, M. S., COMERON, J. M., SAITOU, N., SANDERS, A. R., GELERNTER, J. & GEJMAN, P. V. (2003) Synonymous mutations in the human dopamine receptor D2 (DRD2) affect mRNA stability and synthesis of the receptor. *Hum Mol Genet*, 12, 205-16.
- DURET, L. (2002) Detecting genomic features under weak selective pressure: the example of codon usage in animals and plants. *Bioinformatics*, 18 Suppl 2, S91.
- ELBERS, C. C., VAN EIJK, K. R., FRANKE, L., MULDER, F., VAN DER SCHOUW, Y. T., WIJMENGA, C. & ONLAND-MORET, N. C. (2009) Using Genome-Wide Pathway Analysis to Unravel the Etiology of Complex Disease. *Genetic Epidemiology*, 33, 419-431.
- FAIRBANKS, D. J. & ANDERSON, W. R. (1999) *Genetics the continuity of life*, United States of America, Brooks/Cole Publishing Company.
- FAIRBROTHER, W. G., HOLSTE, D., BURGE, C. B. & SHARP, P. A. (2004) Single nucleotide polymorphism-based validation of exonic splicing enhancers. *PLoS Biol*, 2, E268.
- FAIRBROTHER, W. G., YEH, R. F., SHARP, P. A. & BURGE, C. B. (2002) Predictive identification of exonic splicing enhancers in human genes. *Science*, 297, 1007-13.
- FREEDMAN, S. D., SAKAMOTO, K. & VENU, R. P. (1993) GP2, the homologue to the renal cast protein uromodulin, is a major component of intraductal plugs in chronic pancreatitis. *J Clin Invest*, 92, 83-90.
- FRYDMAN, J. (2001) Folding of newly translated proteins in vivo: the role of molecular chaperones. *Annu Rev Biochem*, 70, 603-47.
- FUNG, K. L. & GOTTESMAN, M. M. (2009) A synonymous polymorphism in a common MDR1 (ABCB1) haplotype shapes protein function. *Biochim Biophys Acta*, 1794, 860-71.

- GOECKS, J., NEKRUTENKO, A. & TAYLOR, J. (2010) Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol*, 11, R86.
- GOTTESMAN, M. M., FOJO, T. & BATES, S. E. (2002) Multidrug resistance in cancer: role of ATP-dependent transporters. *Nat Rev Cancer*, 2, 48-58.
- GU, W., ZHOU, T., MA, J., SUN, X. & LU, Z. (2004) The relationship between synonymous codon usage and protein structure in *Escherichia coli* and *Homo sapiens*. *Biosystems*, 73, 89-97.
- HAFTEK, M., SIMON, M., KANITAKIS, J., MARECHAL, S., CLAUDY, A., SERRE, G. & SCHMITT, D. (1997) Expression of corneodesmosin in the granular layer and stratum corneum of normal and diseased epidermis. *Br J Dermatol*, 137, 864-73.
- HAMOSH, A., SCOTT, A. F., AMBERGER, J. S., BOCCHINI, C. A. & MCKUSICK, V. A. (2005) Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic Acids Res*, 33, D514-7.
- HARTEVELD, C. L., WIJERMANS, P. W., VAN DELFT, P., RASP, E., HAAK, H. L. & GIORDANO, P. C. (2004) An alpha-thalassemia phenotype in a Dutch Hindustani, caused by a new point mutation that creates an alternative splice donor site in the first exon of the alpha2-globin gene. *Hemoglobin*, 28, 255-9.
- HINDORFF, L. A., SETHUPATHY, P., JUNKINS, H. A., RAMOS, E. M., MEHTA, J. P., COLLINS, F. S. & MANOLIO, T. A. (2009) Potential etiologic and functional implications of genome-wide association loci for human diseases and traits. *Proc Natl Acad Sci U S A*, 106, 9362-7.
- HOFFMEYER, S., BURK, O., VON RICHTER, O., ARNOLD, H. P., BROCKMOLLER, J., JOHNE, A., CASCORBI, I., GERLOFF, T., ROOTS, I., EICHELBAUM, M. & BRINKMANN, U. (2000) Functional polymorphisms of the human multidrug-resistance gene: multiple sequence variations and correlation of one allele with P-glycoprotein expression and activity in vivo. *Proc Natl Acad Sci U S A*, 97, 3473-8.
- HOLLAMS, E. M., GILES, K. M., THOMSON, A. M. & LEEDMAN, P. J. (2002) mRNA stability and the control of gene expression: implications for human disease. *Neurochem Res*, 27, 957-80.
- HORINOUCHI, M., SAKAEDA, T., NAKAMURA, T., MORITA, Y., TAMURA, T., AOYAMA, N., KASUGA, M. & OKUMURA, K. (2002) Significant genetic linkage of MDR1 polymorphisms at positions 3435 and 2677: functional relevance to pharmacokinetics of digoxin. *Pharm Res*, 19, 1581-5.
- HUBBARD, T., BARKER, D., BIRNEY, E., CAMERON, G., CHEN, Y., CLARK, L., COX, T., CUFF, J., CURWEN, V., DOWN, T., DURBIN, R., EYRAS, E., GILBERT, J., HAMMOND, M., HUMINIECKI, L., KASPRZYK, A., LEHVASLAIHO, H., LIJNZAAD, P., MELSOPP, C., MONGIN, E., PETTETT, R., POCOOCK, M., POTTER, S., RUST, A., SCHMIDT, E., SEARLE, S., SLATER, G., SMITH, J., SPOONER, W., STABENAU, A., STALKER, J., STUPKA, E., URETA-VIDAL, A., VASTRIK, I. & CLAMP, M. (2002) The Ensembl genome database project. *Nucleic Acids Res*, 30, 38-41.
- ITOH, H., WASHIO, T. & TOMITA, M. (2004) Computational comparative analyses of alternative splicing regulation using full-length cDNA of various eukaryotes. *Rna*, 10, 1005-18.

- JAEGER, J. A., TURNER, D. H. & ZUKER, M. (1989) Improved predictions of secondary structures for RNA. *Proc Natl Acad Sci U S A*, 86, 7706-10.
- JOHNSON, A. D., TRUMBOWER, H. E. & SADEE, W. O. (2011) RNA structures affected by Single Nucleotide Polymorphisms in transcribed regions of the human genome. *WebmedCentral BIOINFORMATICS*, 2, WMC001600.
- JOHNSON, J. M., CASTLE, J., GARRETT-ENGELE, P., KAN, Z., LOERCH, P. M., ARMOUR, C. D., SANTOS, R., SCHADT, E. E., STOUGHTON, R. & SHOEMAKER, D. D. (2003) Genome-wide survey of human alternative pre-mRNA splicing with exon junction microarrays. *Science*, 302, 2141-4.
- JONCA, N., GUERRIN, M., HADJIOLOVA, K., CAUBET, C., GALLINARO, H., SIMON, M. & SERRE, G. (2002) Corneodesmosin, a component of epidermal corneocyte desmosomes, displays homophilic adhesive properties. *J Biol Chem*, 277, 5024-9.
- KAMPA, D., CHENG, J., KAPRANOV, P., YAMANAKA, M., BRUBAKER, S., CAWLEY, S., DRENKOW, J., PICCOLBONI, A., BEKIRANOV, S., HELT, G., TAMMANA, H. & GINGERAS, T. R. (2004) Novel RNAs identified from an in-depth analysis of the transcriptome of human chromosomes 21 and 22. *Genome Res*, 14, 331-42.
- KAN, Z., CASTLE, J., JOHNSON, J. M. & TSINOREMAS, N. F. (2004) Detection of novel splice forms in human and mouse using cross-species approach. *Pac Symp Biocomput*, 42-53.
- KARLIN, S. & MRAZEK, J. (1996) What drives codon choices in human genes? *J Mol Biol*, 262, 459-72.
- KAROLCHIK, D., HINRICHS, A. S. & KENT, W. J. (2007) The UCSC Genome Browser. *Curr Protoc Bioinformatics*, Chapter 1, Unit 1 4.
- KENT, W. J., SUGNET, C. W., FUREY, T. S., ROSKIN, K. M., PRINGLE, T. H., ZAHLER, A. M. & HAUSSLER, D. (2002) The human genome browser at UCSC. *Genome Res*, 12, 996-1006.
- KEREN, H., LEV-MAOR, G. & AST, G. (2010) Alternative splicing and evolution: diversification, exon definition and function. *Nat Rev Genet*, 11, 345-55.
- KIDD, J. M., COOPER, G. M., DONAHUE, W. F., HAYDEN, H. S., SAMPAS, N., GRAVES, T., HANSEN, N., TEAGUE, B., ALKAN, C., ANTONACCI, F., HAUGEN, E., ZERR, T., YAMADA, N. A., TSANG, P., NEWMAN, T. L., TUZUN, E., CHENG, Z., EBLING, H. M., TUSNEEM, N., DAVID, R., GILLET, W., PHELPS, K. A., WEAVER, M., SARANGA, D., BRAND, A., TAO, W., GUSTAFSON, E., MCKERNAN, K., CHEN, L., MALIG, M., SMITH, J. D., KORN, J. M., MCCARROLL, S. A., ALTSHULER, D. A., PEIFFER, D. A., DORSCHNER, M., STAMATOYANNOPOULOS, J., SCHWARTZ, D., NICKERSON, D. A., MULLIKIN, J. C., WILSON, R. K., BRUHN, L., OLSON, M. V., KAUL, R., SMITH, D. R. & EICHLER, E. E. (2008) Mapping and sequencing of structural variation from eight human genomes. *Nature*, 453, 56-64.
- KIMCHI-SARFATY, C., OH, J. M., KIM, I. W., SAUNA, Z. E., CALCAGNO, A. M., AMBUDKAR, S. V. & GOTTESMAN, M. M. (2007) A "silent" polymorphism in the MDR1 gene changes substrate specificity. *Science*, 315, 525-8.
- KOMAR, A. A., LESNIK, T. & REISS, C. (1999) Synonymous codon substitutions affect ribosome traffic and protein folding during in vitro translation. *FEBS Lett*, 462, 387-91.

- KOTLAR, D. & LAVNER, Y. (2006) The action of selection on codon bias in the human genome is related to frequency, complexity, and chronology of amino acids. *BMC Genomics*, 7, 67.
- KRIVENTSEVA, E. V., KOCH, I., APWEILER, R., VINGRON, M., BORK, P., GELFAND, M. S. & SUNYAEV, S. (2003) Increase of functional diversity by alternative splicing. *Trends Genet*, 19, 124-8.
- LANDER, E. S., LINTON, L. M., BIRREN, B., NUSBAUM, C., ZODY, M. C., BALDWIN, J., DEVON, K., DEWAR, K., DOYLE, M., FITZHUGH, W., FUNKE, R., GAGE, D., HARRIS, K., HEAFORD, A., HOWLAND, J., KANN, L., LEHOCZKY, J., LEVINE, R., MCEWAN, P., MCKERNAN, K., MELDRIM, J., MESIROV, J. P., MIRANDA, C., MORRIS, W., NAYLOR, J., RAYMOND, C., ROSETTI, M., SANTOS, R., SHERIDAN, A., SOUGNEZ, C., STANGE-THOMANN, N., STOJANOVIC, N., SUBRAMANIAN, A., WYMAN, D., ROGERS, J., SULSTON, J., AINSCOUGH, R., BECK, S., BENTLEY, D., BURTON, J., CLEE, C., CARTER, N., COULSON, A., DEADMAN, R., DELOUKAS, P., DUNHAM, A., DUNHAM, I., DURBIN, R., FRENCH, L., GRAHAM, D., GREGORY, S., HUBBARD, T., HUMPHRAY, S., HUNT, A., JONES, M., LLOYD, C., MCMURRAY, A., MATTHEWS, L., MERCER, S., MILNE, S., MULLIKIN, J. C., MUNGALL, A., PLUMB, R., ROSS, M., SHOWNKEEN, R., SIMS, S., WATERSTON, R. H., WILSON, R. K., HILLIER, L. W., MCPHERSON, J. D., MARRA, M. A., MARDIS, E. R., FULTON, L. A., CHINWALLA, A. T., PEPIN, K. H., GISH, W. R., CHISSOE, S. L., WENDL, M. C., DELEHAUNTY, K. D., MINER, T. L., DELEHAUNTY, A., KRAMER, J. B., COOK, L. L., FULTON, R. S., JOHNSON, D. L., MINX, P. J., CLIFTON, S. W., HAWKINS, T., BRANSCOMB, E., PREDKI, P., RICHARDSON, P., WENNING, S., SLEZAK, T., DOGGETT, N., CHENG, J. F., OLSEN, A., LUCAS, S., ELKIN, C., UBERBACHER, E., FRAZIER, M., et al. (2001) Initial sequencing and analysis of the human genome. *Nature*, 409, 860-921.
- LAVNER, Y. & KOTLAR, D. (2005) Codon bias as a factor in regulating expression via translation rate in the human genome. *Gene*, 345, 127-38.
- LEE, C. & WANG, Q. (2005) Bioinformatics analysis of alternative splicing. *Brief Bioinform*, 6, 23-33.
- LOPEZ-BIGAS, N., AUDIT, B., OUZOUNIS, C., PARRA, G. & GUIGO, R. (2005) Are splicing mutations the most frequent cause of hereditary disease? *Elsevier, Letters*, 1900-1903.
- MANIATIS, T. & TASIC, B. (2002) Alternative pre-mRNA splicing and proteome expansion in metazoans. *Nature*, 418, 236-43.
- MANNISTO, P. T. & KAAKKOLA, S. (1999) Catechol-O-methyltransferase (COMT): biochemistry, molecular biology, pharmacology, and clinical efficacy of the new selective COMT inhibitors. *Pharmacol Rev*, 51, 593-628.
- MAQUAT, L. E. (2004) Nonsense-mediated mRNA decay: splicing, translation and mRNP dynamics. *Nat Rev Mol Cell Biol*, 5, 89-99.
- MARBACH, J. J. & LEVITT, M. (1976) Erythrocyte catechol-O-methyltransferase activity in facial pain patients. *J Dent Res*, 55, 711.
- MATLIN, A. J., CLARK, F. & SMITH, C. W. (2005) Understanding alternative splicing: towards a cellular code. *Nat Rev Mol Cell Biol*, 6, 386-98.
- MODREK, B. & LEE, C. (2002) A genomic view of alternative splicing. *Nat Genet*, 30, 13-9.

- MOHSEN, A. M., KHADER, A. T. & RAMACHANDRAM, D. (2009) Comparison of HSRNAFold and RNAFold algorithms for RNA secondary structure prediction. *IEEE/ACM Trans Comput Biol Bioinform.*
- MOUNT, D. (2004) *Bioinformatics: sequence and genome analysis*, John Inglis.
- NACKLEY, A. G., SHABALINA, S. A., TCHIVILEVA, I. E., SATTERFIELD, K., KORCHYNSKYI, O., MAKAROV, S. S., MAIXNER, W. & DIATCHENKO, L. (2006) Human catechol-O-methyltransferase haplotypes modulate protein expression by altering mRNA secondary structure. *Science*, 314, 1930-3.
- NAGY, E. & MAQUAT, L. E. (1998) A rule for termination-codon position within intron-containing genes: when nonsense affects RNA abundance. *Trends Biochem Sci*, 23, 198-9.
- NILSEN, T. W. (2003) The spliceosome: the most complex macromolecular machine in the cell? *Bioessays*, 25, 1147-9.
- NUSSINOV, R. & JACOBSON, A. B. (1980) Fast algorithm for predicting the secondary structure of single-stranded RNA. *Proc Natl Acad Sci U S A*, 77, 6309-13.
- ORBAN, T. I. & OLAH, E. (2001) Purifying selection on silent sites -- a constraint from splicing regulation? *Trends Genet*, 17, 252-3.
- PARMLEY, J. L. & HURST, L. D. (2007) Exonic splicing regulatory elements skew synonymous codon usage near intron-exon boundaries in mammals. *Mol Biol Evol*, 24, 1600-3.
- PLOTKIN, J. B., ROBINS, H. & LEVINE, A. J. (2004) Tissue-specific codon usage and the expression of human genes. *Proc Natl Acad Sci U S A*, 101, 12588-91.
- PRUITT, K. D., KATZ, K. S., SICOTTE, H. & MAGLOTT, D. R. (2000) Introducing RefSeq and LocusLink: curated human genome resources at the NCBI. *Trends Genet*, 16, 44-7.
- PRUITT, K. D. & MAGLOTT, D. R. (2001) RefSeq and LocusLink: NCBI gene-centered resources. *Nucleic Acids Res*, 29, 137-40.
- PURVIS, I. J., BETTANY, A. J., SANTIAGO, T. C., COGGINS, J. R., DUNCAN, K., EASON, R. & BROWN, A. J. (1987) The efficiency of folding of some proteins is increased by controlled rates of translation in vivo. A hypothesis. *J Mol Biol*, 193, 413-7.
- REUMERS, J., CONDE, L., MEDINA, I., MAURER-STROH, S., VAN DURME, J., DOPAZO, J., ROUSSEAU, F. & SCHYMKOWITZ, J. (2008) Joint annotation of coding and non-coding single nucleotide polymorphisms and mutations in the SNPeffect and PupaSuite databases. *Nucleic Acids Res*, 36, D825-9.
- RIGOTTI, A., EDELMAN, E. R., SEIFERT, P., IQBAL, S. N., DEMATTOS, R. B., TEMEL, R. E., KRIEGER, M. & WILLIAMS, D. L. (1996) Regulation by adrenocorticotrophic hormone of the in vivo expression of scavenger receptor class B type I (SR-BI), a high density lipoprotein receptor, in steroidogenic cells of the murine adrenal gland. *J Biol Chem*, 271, 33545-9.
- RISCH, N. & MERIKANGAS, K. (1996) The future of genetic studies of complex human diseases. *Science*, 273, 1516-7.
- ROBERTS, C. G., SHEN, H., MITCHELL, B. D., DAMCOTT, C. M., SHULDINER, A. R. & RODRIGUEZ, A. (2007) Variants in scavenger receptor class B type I gene are associated with HDL cholesterol levels in younger women. *Hum Hered*, 64, 107-13.
- ROCA, X., SACHIDANANDAM, R. & KRAINER, A. R. (2003) Intrinsic differences between authentic and cryptic 5' splice sites. *Nucleic Acids Res*, 31, 6321-33.

- SAKAI, H. & MARUYAMA, O. (2004) Extensive search for discriminative features of alternative splicing. *Pac Symp Biocomput*, 54-65.
- SAMMETH, M., FOISSAC, S. & GUIGO, R. (2008) A general definition and nomenclature for alternative splicing events. *PLoS Comput Biol*, 4, e1000147.
- SAUNA, Z. E. & KIMCHI-SARFATY, C. (2011) Understanding the contribution of synonymous mutations to human disease. *Nat Rev Genet*, 12, 683-91.
- SAUNA, Z. E., KIMCHI-SARFATY, C., AMBUDKAR, S. V. & GOTTESMAN, M. M. (2007) Silent polymorphisms speak: how they affect pharmacogenomics and the treatment of cancer. *Cancer Res*, 67, 9609-12.
- SCHUSTER, S. C., MILLER, W., RATAN, A., TOMSHO, L. P., GIARDINE, B., KASSON, L. R., HARRIS, R. S., PETERSEN, D. C., ZHAO, F., QI, J., ALKAN, C., KIDD, J. M., SUN, Y., DRAUTZ, D. I., BOUFFARD, P., MUZNY, D. M., REID, J. G., NAZARETH, L. V., WANG, Q., BURHANS, R., RIEMER, C., WITTEKINDT, N. E., MOORJANI, P., TINDALL, E. A., DANKO, C. G., TEO, W. S., BUBOLTZ, A. M., ZHANG, Z., MA, Q., OOSTHUYSEN, A., STEENKAMP, A. W., OOSTUISEN, H., VENTER, P., GAJEWSKI, J., ZHANG, Y., PUGH, B. F., MAKOVA, K. D., NEKRUTENKO, A., MARDIS, E. R., PATTERSON, N., PRINGLE, T. H., CHIAROMONTE, F., MULLIKIN, J. C., EICHLER, E. E., HARDISON, R. C., GIBBS, R. A., HARKINS, T. T. & HAYES, V. M. (2010) Complete Khoisan and Bantu genomes from southern Africa. *Nature*, 463, 943-7.
- SCHWENDER, H. & ICKSTADT, K. (2008) Empirical Bayes analysis of single nucleotide polymorphisms. *BMC Bioinformatics*, 9, 144.
- SHARP, P. M. & LI, W. H. (1987) The codon Adaptation Index--a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic Acids Res*, 15, 1281-95.
- SHERRY, S. T., WARD, M. & SIROTKIN, K. (1999) dbSNP-database for single nucleotide polymorphisms and other classes of minor genetic variation. *Genome Res*, 9, 677-9.
- SMIGIELSKI, E. M., SIROTKIN, K., WARD, M. & SHERRY, S. T. (2000) dbSNP: a database of single nucleotide polymorphisms. *Nucleic Acids Res*, 28, 352-355.
- SOREK, R., SHEMES, R., COHEN, Y., BASECHESS, O., AST, G. & SHAMIR, R. (2004) A non-EST-based method for exon-skipping prediction. *Genome Res*, 14, 1617-23.
- SOSNICK, T. R. (2008) Kinetic barriers and the role of topology in protein and RNA folding. *Protein Sci*, 17, 1308-18.
- TAKYAR, S., HICKERSON, R. P. & NOLLER, H. F. (2005) mRNA helicase activity of the ribosome. *Cell*, 120, 49-58.
- TANG, K., NGOI, S. M., GWEE, P. C., CHUA, J. M., LEE, E. J., CHONG, S. S. & LEE, C. G. (2002) Distinct haplotype profiles and strong linkage disequilibrium at the MDR1 multidrug transporter gene locus in three ethnic Asian populations. *Pharmacogenetics*, 12, 437-50.
- TINOCO, I. & BUSTAMANTE, C. (1999) How RNA folds. *J. Mol. Biol.*, 293, 271-281.
- TINOCO, I., JR., UHLENBECK, O. C. & LEVINE, M. D. (1971) Estimation of secondary structure in ribonucleic acids. *Nature*, 230, 362-7.
- TSAI, C. J., SAUNA, Z. E., KIMCHI-SARFATY, C., AMBUDKAR, S. V., GOTTESMAN, M. M. & NUSSINOV, R. (2008) Synonymous mutations and ribosome stalling can lead to altered folding pathways and distinct minima. *J Mol Biol*, 383, 281-91.

- TSANG, H. H. & WIESE, K. C. (2007) SARNAPredict: accuracy improvement of RNA secondary structure prediction using permutation-based simulated annealing. *IEEE/ACM Trans Comput Biol Bioinform*, 7, 727-40.
- VENTER, J. C., ADAMS, M. D., MYERS, E. W., LI, P. W., MURAL, R. J., SUTTON, G. G., SMITH, H. O., YANDELL, M., EVANS, C. A., HOLT, R. A., GOCAYNE, J. D., AMANATIDES, P., BALLEW, R. M., HUSON, D. H., WORTMAN, J. R., ZHANG, Q., KODIRA, C. D., ZHENG, X. H., CHEN, L., SKUPSKI, M., SUBRAMANIAN, G., THOMAS, P. D., ZHANG, J., GABOR MIKLOS, G. L., NELSON, C., BRODER, S., CLARK, A. G., NADEAU, J., MCKUSICK, V. A., ZINDER, N., LEVINE, A. J., ROBERTS, R. J., SIMON, M., SLAYMAN, C., HUNKAPILLER, M., BOLANOS, R., DELCHER, A., DEW, I., FASULO, D., FLANIGAN, M., FLOREA, L., HALPERN, A., HANNENHALLI, S., KRAVITZ, S., LEVY, S., MOBARRY, C., REINERT, K., REMINGTON, K., ABU-THREIDEH, J., BEASLEY, E., BIDDICK, K., BONAZZI, V., BRANDON, R., CARGILL, M., CHANDRAMOULISWARAN, I., CHARLAB, R., CHATURVEDI, K., DENG, Z., DI FRANCESCO, V., DUNN, P., EILBECK, K., EVANGELISTA, C., GABRIELIAN, A. E., GAN, W., GE, W., GONG, F., GU, Z., GUAN, P., HEIMAN, T. J., HIGGINS, M. E., JI, R. R., KE, Z., KETCHUM, K. A., LAI, Z., LEI, Y., LI, Z., LI, J., LIANG, Y., LIN, X., LU, F., MERKULOV, G. V., MILSHINA, N., MOORE, H. M., NAIK, A. K., NARAYAN, V. A., NEELAM, B., NUSSKERN, D., RUSCH, D. B., SALZBERG, S., SHAO, W., SHUE, B., SUN, J., WANG, Z., WANG, A., WANG, X., WANG, J., WEI, M., WIDES, R., XIAO, C., YAN, C., et al. (2001) The sequence of the human genome. *Science*, 291, 1304-51.
- WANG, Z. & BURGE, C. B. (2008) Splicing regulation: from a parts list of regulatory elements to an integrated splicing code. *Rna*, 14, 802-13.
- WUCHTY, S., FONTANA, W., HOFACKER, I. L. & SCHUSTER, P. (1999) Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers*, 49, 145-65.
- YEO, G. & BURGE, C. B. (2004) Maximum entropy modeling of short sequence motifs with applications to RNA splicing signals. *J Comput Biol*, 11, 377-94.
- ZHANG, X. H. & CHASIN, L. A. (2004) Computational definition of sequence motifs governing constitutive exon splicing. *Genes Dev*, 18, 1241-50.
- ZHOU, Y., GOTTESMAN, M. M. & PASTAN, I. (1999) The extracellular loop between TM5 and TM6 of P-glycoprotein is required for reactivity with monoclonal antibody UIC2. *Arch Biochem Biophys*, 367, 74-80.
- ZHOU, Z., LICKLIDER, L. J., GYGI, S. P. & REED, R. (2002) Comprehensive proteomic analysis of the human spliceosome. *Nature*, 419, 182-5.
- ZUKER, M. (1989) On finding all suboptimal foldings of an RNA molecule. *Science*, 244, 48-52.
- ZUKER, M. (2003) Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res*, 31, 3406-15.
- ZUKER, M., JAEGER, J. A. & TURNER, D. H. (1991) A comparison of optimal and suboptimal RNA secondary structures predicted by free energy minimization with structures determined by phylogenetic comparison. *Nucleic Acids Res*, 19, 2707-14.
- ZUKER, M. & SANKOFF, D. (1984) RNA secondary structures and their predictions. *Bulletin of Mathematical Biology*, 46, 591-621.
- ZUKER, M. & STIEGLER, P. (1981) Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res*, 9, 133-48.

