AN ABSTRACT OF THE DISSERTATION OF

John Smith for the degree of Doctor of Philosophy in Computer Science presented on September 23, 2011.

Title: The Meaning of Life

Abstract approved: ______________________________

Joan Smythe

This is an abstract statement.

The Meaning of Life

by

John Smith

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented September 23, 2011
Commencement June 2012

Doctor of Philosophy dissertation of John Smith presented on September 23, 2011.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

John Smith, Author

ACKNOWLEDGEMENTS

I would like to acknowledge the Starting State and the Transition Function.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

Chapter 1: Introduction

I have done some excellent research [**?**].

1.1 Introduction to the Introduction

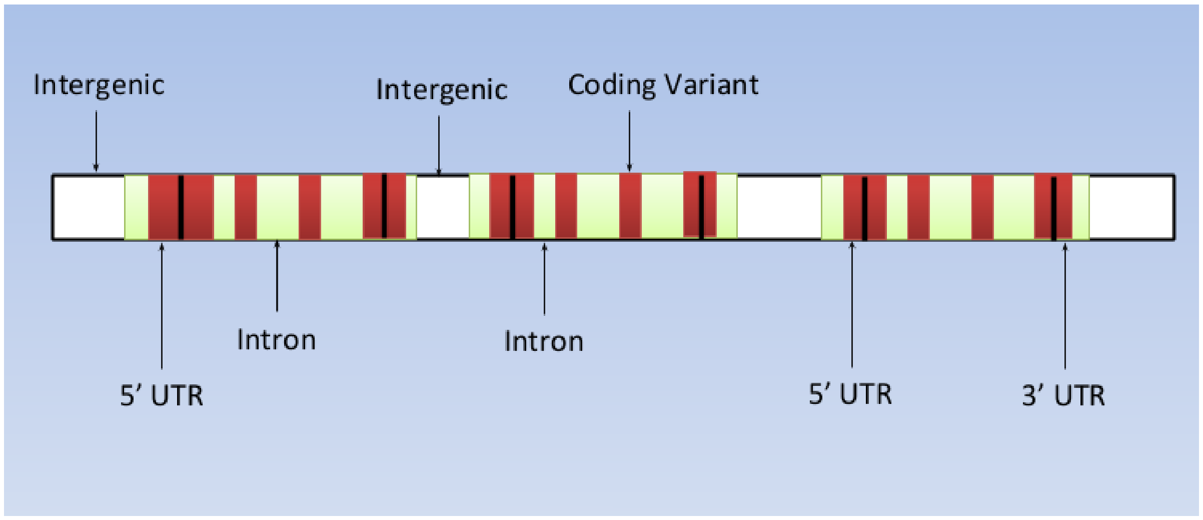


Figure 1.1:

Variant effect	Putative impact
intergenic_region	MODIFIER
intron_variant	MODIFIER
5_prime_UTR_variant	MODIFIER
3_prime_UTR_variant	MODIFIER
non_coding_exon_variant	MODIFIER
synonymous_variant	LOW
non_synonymous_variant	MODERATE

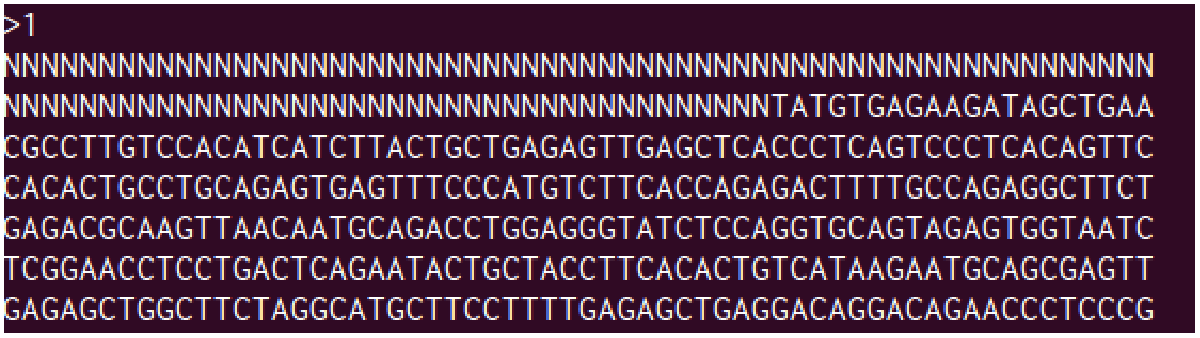
For non-synonymous variant, we annotate the reference amino acid, alternate amino acid and the position of the amino acid instead of the term.

Chapter 2: Methods

2.1 Input

2.1.1 Genome data

The genome data contains the whole nucleotide sequences of each chromosome. It is often stored in a faidx-indexed reference file in the FASTA format. We will use this file to get the amino acid codon of a given variant. Example:



```
>1
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNTATGTGAGAAGATAGCTGAA
CGCCTTGTCCACATCATCTTACTGCTGAGAGTTGAGCTCACCCTCAGTCCCTCACAGTTC
CACACTGCCTGCAGAGTGAGTTTCCCATGTCTTCACCAGAGACTTTTGCCAGAGGCTTCT
GAGACGCAAGTTAACAATGCAGACCTGGAGGGTATCTCCAGGTGCAGTAGAGTGGTAATC
TCGGAACCTCCTGACTCAGAATACTGCTACCTTCACACTGTCATAAGAATGCAGCGAGTT
GAGAGCTGGCTTCTAGGCATGCTTCCTTTTGAGAGCTGAGGACAGGACAGAACCCTCCCG
```

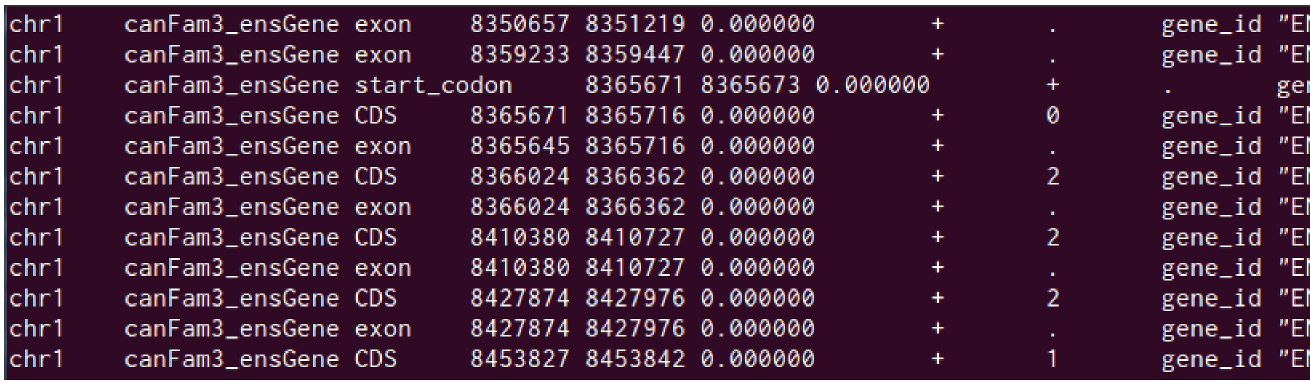
Figure 2.1: Genome data example

2.1.2 Gene structure data

The gene structure data contains information about gene structure, such as start codon, stop codon, exon, strand and gene id. This is the key data to determine the functional area of a given variant. Example:

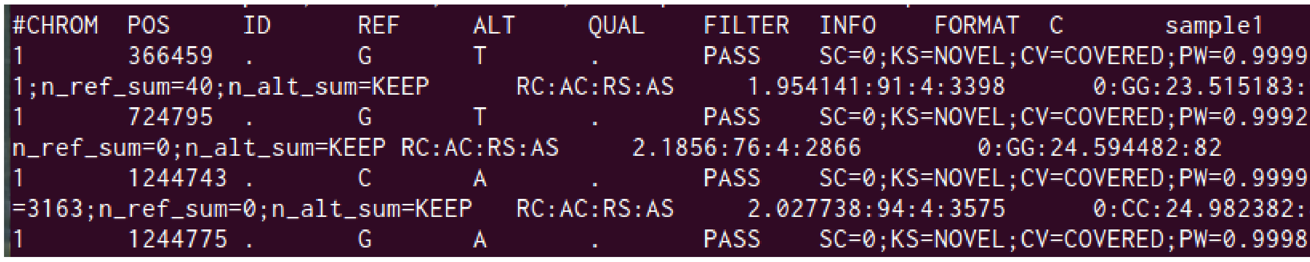
2.1.3 Variant data

This is the data to be annotated. The key information includes chromosome, position, reference base and alternate variant alleles of the variant. The data is typically stored in VCF file. We will annotate the variant in the INFO field. Example:



```
chr1    canFam3_ensGene exon    8350657 8351219 0.000000        +       .       gene_id "E
chr1    canFam3_ensGene exon    8359233 8359447 0.000000        +       .       gene_id "E
chr1    canFam3_ensGene start_codon     8365671 8365673 0.000000        +       .       ge
chr1    canFam3_ensGene CDS     8365671 8365716 0.000000        +       0       gene_id "E
chr1    canFam3_ensGene exon    8365645 8365716 0.000000        +       .       gene_id "E
chr1    canFam3_ensGene CDS     8366024 8366362 0.000000        +       2       gene_id "E
chr1    canFam3_ensGene exon    8366024 8366362 0.000000        +       .       gene_id "E
chr1    canFam3_ensGene CDS     8410380 8410727 0.000000        +       2       gene_id "E
chr1    canFam3_ensGene exon    8410380 8410727 0.000000        +       .       gene_id "E
chr1    canFam3_ensGene CDS     8427874 8427976 0.000000        +       2       gene_id "E
chr1    canFam3_ensGene exon    8427874 8427976 0.000000        +       .       gene_id "E
chr1    canFam3_ensGene CDS     8453827 8453842 0.000000        +       1       gene_id "E
```

Figure 2.2: GTF file example



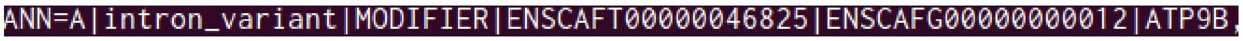
```
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  C       sample1
1       366459  .       G       T       .       PASS    SC=0;KS=NOVEL;CV=COVERED;PW=0.9999
1;n_ref_sum=40;n_alt_sum=KEEP     RC:AC:RS:AS     1.954141:91:4:3398      0:GG:23.515183:
1       724795  .       G       T       .       PASS    SC=0;KS=NOVEL;CV=COVERED;PW=0.9992
n_ref_sum=0;n_alt_sum=KEEP RC:AC:RS:AS     2.1856:76:4:2866        0:GG:24.594482:82
1       1244743 .       C       A       .       PASS    SC=0;KS=NOVEL;CV=COVERED;PW=0.9999
=3163;n_ref_sum=0;n_alt_sum=KEEP   RC:AC:RS:AS     2.027738:94:4:3575      0:CC:24.982382:
1       1244775 .       G       A       .       PASS    SC=0;KS=NOVEL;CV=COVERED;PW=0.9998
```

Figure 2.3: VCF file example

2.2 Output

Genes may overlap. Hence, one variant may have several effects. For each variant, the output is a list of effects. The potential effects are listed in section 1.1. In the project, the output is stored in a VCF file in the INFO field. To be specific, a new subfield, named ANN, will be appended to INFO field. The most important information is the effect of variants and the putative impact of the effect.

Besides the effect of variants, the corresponding Ensemble transcript id and common gene name are also annotated. All the annotation formats follow the standard variant annotation in VCF format. Example:



```
ANN=A|intron_variant|MODIFIER|ENSCAFT00000046825|ENSCAFG00000000012|ATP9B
```

Figure 2.4: VCF file example

2.3 How it works

There are 3 files to be kept track of. If we search the whole files for each variants, the total running time complexity is O(m*n*k), where m, n and k is the size of each file. In addition, the size of the files maybe too large to load them all into RAM. For instance, the size of a typical human genome file is about 3.0 GB. To solve the difficulties in this problem, we develop a fast and memory-efficient algorithm to annotate variants.

The whole problem is split into 2 basic subproblem. The first one is to find the functional area of a variant at given chromosome and position. And the second one is to find out the corresponding DNA sequence around the variant and translate it into amino acids, if the variant is inside an exon.

To speed up the annotation, all the 3 files need to be sorted first by chromosome, then by position, in ascending order. Then, the algorithm starts to annotate each variant in the VCF file. When annotating a variant, the algorithm keeps track of the gene area that the variant is in for the next variant to start annotating.

A special problem when annotating is that genes and exons may overlap. So a variant may have multiple distinct annotations. In this case, we need to continue searching until the current gene area is beyond the current variant position. But we still keep track of the first gene area for next annotation.

To save RAM usage, the algorithm only load one chromosome sequence of the DNA data into memory at the same time. This is because no genes overlap on different chromosomes.

2.4 Other issues

2.4.1 how to get amino acid

2.4.2 strand

Algorithm 1 VARIANT ANNOTATION

```
input  : A GTF file $G$ grouped by gene id. A reference genome file $R$. A VCF file $V$ sorted first
         by chomosome then by position.
output: An annotated VCF File

for each variant in V at chromosome chr and position pos do
    if current chromosome $!=$ chr then
        load chr
    end
    while $pos >$ the end of current gene do
        load next gene
    end
    $A = \varnothing$
    if $pos <$ the start of current gene then
        $A = A$ + "intergenic"
    end
    $g$ = current gene
    while $pos >$ the start of $g$ do
        if $g$ has neither start codon nor stop codon then
            $A = A$ + "non coding"
        end
        else if $pos$ not in exon of $g$ then
            $A = A$ + "intron"
        end
        else if $pos <$ start codon of $g$ then
            $A = A$ + "5' UTR"
        end
        else if $pos >$ stop codon of $g$ then
            $A = A$ + "3' UTR"
        end
        else
            get reference amino acid $ra$ and alternative amino acid $aa$
            if $ra == aa$ then
                $A = A$ + "synonymous"
            end
            else
                $A = A$ + "$ra + pos + aa$"
            end
        end
        $g$ = next gene
    end
    Annotate variant with $A$
end
output:  The annotated VCF file
```

Chapter 3: Result

3.1 Running time

3.2 Memory cost

3.3 success rate

APPENDICES

Appendix A: Redundancy

This appendix is inoperable.