

Max Cheng

Work Sample Portfolio

Bioinformatics and Genome Analysis

Phone: 408-908-0395

Email: mcheng30@jh.edu

LinkedIn: www.linkedin.com/in/maxcheng98

GitHub: <https://github.com/mbcheng88>

Table of Contents

Introduction.....	4
Genomic Records.....	5
ENSEMBL/BioMart/BioMaRt.....	6
Ensembl.....	6
BioMart.....	6
BioMaRt.....	7
Gene Annotation and Prediction.....	9
Prokaryotic Gene Annotation Pipeline.....	9
GLIMMER.....	10
FGENESB.....	10
Eukaryotic Gene Prediction.....	12
Splign.....	12
Genomic BLAST.....	13
BLAT.....	13
Genomic File Types.....	14
BED (browser extension data) files.....	14
WIG (Wiggle Format) files:.....	15
SAM (Sequence Alignment Map) files.....	16
BAM files.....	16
Genome Browsers.....	17
UCSC Genome Browser.....	18
IGV (Integrated Genome Viewer).....	19
NCBI Variation Viewer.....	20
Galaxy.....	21
BEDtools.....	21

NGS/Genome Assembly/Variant Calling Pipeline.....	23
ChIP-seq and RNA-seq Pipeline	25
SNP Research Paper.....	29
Computational Programs and Databases.....	47
SQL.....	47
Python.....	48
Circle "Program"	48
DNA Length and Start Codon Program	48
Motif Counter.....	49
FASTA Sequence Counter.....	49
Parsing Program.....	50
Genome Analysis Program	50
Angle Converter.....	51
DNA Reverse Complement Program.....	52
GFF Feature Exporter	53
Acknowledgements.....	55

Introduction

Hi, my name is Max Cheng, and I am a master's student in Biotechnology with a concentration in Bioinformatics at Johns Hopkins University. My main background is in the biological field, and I recently graduated with a M.S. in NPB (Neurobiology, Physiology, and Behavior) at UC Davis. However, most of my work experience is focused along data analysis and using programs. I've been entering financial data at my family-owned accounting firm since I was young, and during university I was a data analysis research assistant at UC Davis MIND Institute's AIR (Attention, Impulsivity, and Response) ADHD lab. Accounting involved a lot of copying data from one source to another, so I always wanted to automate that process to expedite the work. At the AIR lab, I worked with python code that took fMRI data and prepared it for data analysis and well as designing a VR classroom on the HTC VIVE using Unity. So, to mesh my academic and work experiences, I sought to work towards the goal of becoming a bioinformatics scientist, where I can use my extensive background on biology and translate them into tools and pipelines used by modern biotechnology companies. Thus, I've been slowly transitioning to writing programs and performing data analysis on biological data such as genome sequencing. Through both academics and extracurriculars, I've learned how to write programs with Python, gather and analyze data with R, use the Unix terminal to read and transfer files, and database using SQL.

Now that my university experience is wrapping up, I hope to retain much of what I have learned as a student, so that I can perpetuate these ideas and concepts out into the workforce. Hence, I have created a sample work portfolio that goes through some of my most recent projects at Johns Hopkins, including tools I've used and the programs I've written to analyze genomic data. I hope that this compendium will not only serve as a useful reference to build upon, but an aggregation of what I am capable of as a bioinformatics scientist.

Genomic Records

INSDC (International Nucleotide Sequence Database Collection)

There are 3 main databases with a centralized annotation system:

- Genbank (by NCBI)
- ENA (European Nucleotide Archive)
- DDBJ (DNA Data Bank of Japan)

Example record from NCBI

```
LOCUS      XM_014257108          6526 bp    mRNA    linear    VRT 21-SEP-2015
DEFINITION PREDICTED: Pseudopodoces humilis pleiomorphic adenoma gene 1
            (PLAG1), transcript variant X5, mRNA.
ACCESSION  XM_014257108
VERSION    XM_014257108.1  GI:929410253
DBLINK     BioProject: PRJNA217046
KEYWORDS   RefSeq.
SOURCE     Pseudopodoces humilis (Tibetan ground-tit)
  ORGANISM Pseudopodoces humilis
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Archelosauria; Archosauria; Dinosauria; Saurischia; Theropoda;
            Coelurosauria; Aves; Neognathae; Passeriformes; Paridae;
            Pseudopodoces.
COMMENT    MODEL REFSEQ: This record is predicted by automated computational
            analysis. This record is derived from a genomic sequence
            (NW\_005087555.1) annotated using gene prediction method: Gnomon,
            supported by mRNA and EST evidence.
            Also see:
                Documentation of NCBI's Annotation Process
```

image from http://www.ncbi.nlm.nih.gov/nuccore/XM_014257108.1

These databases will serve as a basis for most of the genomic data being analyzed. Additional databases will be used as a resource for specific organism genomes such as <http://www.wormbase.org> for nematodes and <https://yeastmine.yeastgenome.org> for fission yeast genomes.

ENSEMBL/BioMart/BioMaRt

Ensembl

A genome browser for chordates

- You can search specific genes and species-specific indexes
- Ensembl's data is stored MySQL relational databases allowing for easy access through SQL queries through a terminal or other related program

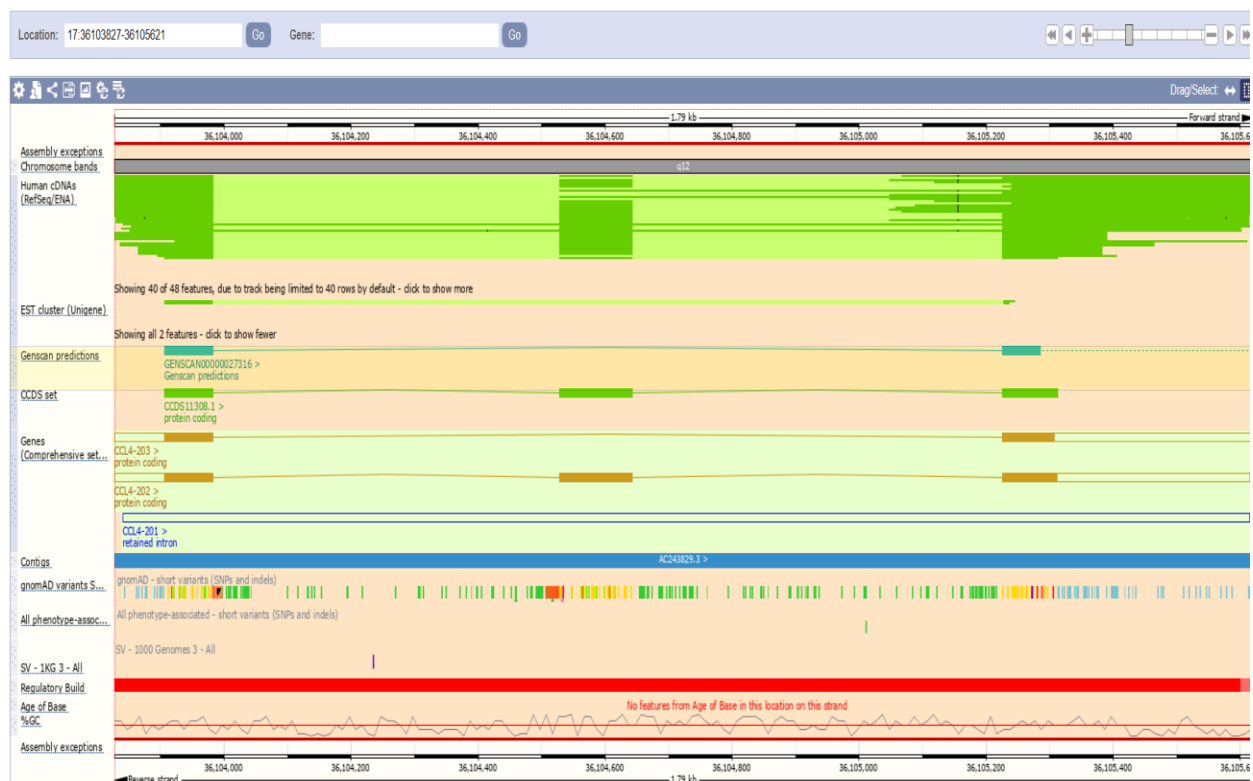


Figure above shows the Gene Summary for the CCL4 gene

BioMart

Allows users to download datasets from ENSEMBL

- Many filters can be applied such as Region, Gene, Phenotype, Gene Ontology, Multi Species Comparisons, Protein Domains and Family, and Variant

- Additionally, you can filter further by Attributes: Gene, External, and Protein Domains and Family

Sample query from the browser version of Ensembl:

<https://m.ensembl.org/info/data/biomart/index.html>

BioMaRt

A Bioconductor package that integrates bioinformatics tools into R

- Does the same thing as BioMart but can be used in a terminal or R to view datasets from ENSEMBL

Code in R I used to pull a specific query from the ENSEMBL database using BioMaRt shown below. The identifiers were pulled from OMIM by searching at OMIM.org for Huntington's disease. Then these values: 603218, 604802, 143100, 606438 are used in the getBM command to find matching ENSEMBL gene ID's:

```
> getBM(attributes=c("entrezgene_id", "hgnc_symbol", "ensembl_gene_id"), filters="mim_morbid_accession", values=c("603218", "604802", "143100", "606438"), mart=
  entrezgene_id hgnc_symbol ensembl_gene_id
1      3064      HTT  ENSG00000197386
2      5621     PRNP  ENSG00000171867
3     57338     JPH3  ENSG00000154118
>
```

```
pallipr(4)(1).fasta x  Untitled1 x  Untitled2 x  shellprt.fasta x
Source on Save
1 if (!requireNamespace("BiocManager", quietly = TRUE))
2   install.packages("BiocManager")
3 BiocManager::install("biomaRt")
4
5 library(biomaRt)
6 listMarts()
7 ensembl=useMart("ensembl")
8 listDatasets(ensembl)
9 ensembl = useMart("ensembl", dataset="hsapiens_gene_ensembl")
10 getBM(attributes=c("ensembl_transcript_id", "hgnc_symbol", "ensembl_gene_id"), filters="mim_morbid_accession", values=c("603218", "604802", "143100", "6064
11 getBM(attributes=c("entrezgene_id", "hgnc_symbol", "ensembl_gene_id"), filters="mim_morbid_accession", values=c("603218", "604802", "143100", "606438"), ma
```

The results of the R query:

```
Console Terminal Jobs
R 4.1.1 ~ /
getBM(attributes=c("ensembl_transcript_id", "hgnc_symbol", "ensembl_gene_id"), filters="mim_morbid_accession", values=c("603218", "604802", "143100", "606438"), mart=ensembl)
> ", "604802", "143100", "606438"), mart=ensembl)
Error: unexpected numeric constant in "", "604802"
> getBM(attributes=c("ensembl_transcript_id", "hgnc_symbol", "ensembl_gene_id"), filters="mim_morbid_accession", values=c("603218", "604802", "143100", "606438"), mart=ensembl)
  ensembl_transcript_id hgnc_symbol ensembl_gene_id
1      ENST00000680239      HTT  ENSG00000197386
2      ENST00000680956      HTT  ENSG00000197386
3      ENST00000680360      HTT  ENSG00000197386
4      ENST00000681528      HTT  ENSG00000197386
5      ENST00000647962      HTT  ENSG00000197386
6      ENST00000649900      HTT  ENSG00000197386
7      ENST00000680291      HTT  ENSG00000197386
8      ENST00000355072      HTT  ENSG00000197386
9      ENST00000648150      HTT  ENSG00000197386
10     ENST00000506137      HTT  ENSG00000197386
11     ENST00000512909      HTT  ENSG00000197386
12     ENST00000510626      HTT  ENSG00000197386
13     ENST00000649131      HTT  ENSG00000197386
14     ENST00000509618      HTT  ENSG00000197386
15     ENST00000650588      HTT  ENSG00000197386
16     ENST00000650595      HTT  ENSG00000197386
17     ENST00000513639      HTT  ENSG00000197386
18     ENST00000513326      HTT  ENSG00000197386
19     ENST00000509043      HTT  ENSG00000197386
20     ENST00000509751      HTT  ENSG00000197386
21     ENST00000512068      HTT  ENSG00000197386
22     ENST00000513806      HTT  ENSG00000197386
23     ENST00000508321      HTT  ENSG00000197386
24     ENST00000430350     PRNP  ENSG00000171867
25     ENST00000379440     PRNP  ENSG00000171867
26     ENST00000424424     PRNP  ENSG00000171867
27     ENST00000457586     PRNP  ENSG00000171867
28     ENST00000537256     JPH3  ENSG00000154118
29     ENST00000301008     JPH3  ENSG00000154118
30     ENST00000284262     JPH3  ENSG00000154118
31     ENST00000563609     JPH3  ENSG00000154118
>
```

```
pallipr(4)(1).fasta x  Untitled1 x  Untitled2 x  shellprt.fasta x
Source on Save Run Source
1 if (!requireNamespace("BiocManager", quietly = TRUE))
2   install.packages("BiocManager")
3 BiocManager::install("biomaRt")
4
5 library(biomaRt)
6 listMarts()
7 ensembl=useMart("ensembl")
8 listDatasets(ensembl)
9 ensembl = useMart("ensembl", dataset="hsapiens_gene_ensembl")
10 getBM(attributes=c("ensembl_transcript_id", "hgnc_symbol", "ensembl_gene_id"), filters="mim_morbid_accession", values=c("603218", "604802", "143100", "606438"), mart=ensembl)
11
12
13
```


GLIMMER

Unix-based program for finding genes in microbial DNA. Has ability to find long ORFs and can be used to build a training set for genome analysis.

Unix commands used to generate an ORF .predict file using a full genome (sheli.fasta) to train the dataset, and predicts the ORF of the partial genome (sheliprt.fasta):

```
[mcheng30@bf3 FASTAS]$ long-orfs -n -t 1.15 sheli.fasta sheli.longorfs
```

```
[mcheng30@bf3 FASTAS]$ extract -t sheli.fasta sheli.longorfs > sheli.train
```

```
[mcheng30@bf3 FASTAS]$ build-icm -r sheli.icm < sheli.train
```

```
[mcheng30@bf3 FASTAS]$ glimmer3 -o50 -g110 -t30 sheliprt.fasta sheli.icm sheliprt
```

```
[mcheng30@bf3 FASTAS]$ extract -t sheliprt.fasta sheliprt.predict > sheliprt.glimmer
```

Predict file with the ORFs:

> *Spiroplasma helicoides* strain TABS-2, partial sequence

orf00001	635	991	+2	2.77
orf00003	1154	1312	+2	0.42
orf00004	1334	1978	+2	7.53
orf00005	2242	2463	+1	5.70
orf00007	2585	4003	+2	11.22
orf00008	4034	4678	+2	7.77
orf00009	4880	5143	+2	7.72

FGENESB

Annotates bacterial genomes using Genbank based on Markov Chain Models that is very useful in finding operon and predicting genes in bacteria.

Same dataset (sheliprt.fasta) as the GLIMMER example but run through FGENESB/softberry instead. *Results of the query are shown below.*

```

Prediction of potential genes in microbial genomes
Time: Tue Jan 1 00:00:00 2005
Seq name: Spiroplasma helicoides strain TABS-2, partial sequence
Length of sequence - 5500 bp
Number of predicted genes - 9
Number of transcription units - 6, operons - 2

```

N	Tu/Op	Conserved pairs (N/Pv)	S		Start	End	Score
1	1 Op	1	+	CDS	635 -	991	117
2	1 Op	2	+	CDS	998 -	1141	144
3	2 Tu	1	-	CDS	1126 -	1365	73
4	3 Tu	1	+	CDS	1334 -	1978	381
5	4 Tu	1	+	CDS	2242 -	2463	231
6	5 Op	1	+	CDS	2585 -	4003	998
7	5 Op	2	+	CDS	4010 -	4678	423
8	5 Op	3	+	CDS	4703 -	4768	72
9	6 Tu	1	+	CDS	4880 -	5143	169

```

Predicted protein(s):
>GENE 1 635 - 991 117 118 aa, chain +
MTYSFSFIIIEGVQYDTSKFLISSIASCAFIIAHLLFEYFSQLILNQSIKLIKLRVIT
AKNFFTENYKVSOLDTGEFININSTKINQLADNYFTSIFDISRCIIAIIISYGFLLYIS
>GENE 2 998 - 1141 144 47 aa, chain +
MLAVMILSLVLVLIPLMSKIGQKRINRVANEENDKFLQTTKDTYNSY
>GENE 3 1126 - 1365 73 79 aa, chain -
MFSVNIKPIPIIYPAQYIQQKNIKICTPRKTTISSKNLVVDITFFIIFWFLTSNFFDPST
IWLISLFVWFMLQYTQYEL
>GENE 4 1334 - 1978 381 214 aa, chain +
MNIGLIFTLNLSSVYCFSSSSAKALMNIINHRKVYLSNYQDNKINNTVIGEDLKITI
EFKNVDFKIKNSSNLIIEKFNLIKINKGDKVLKIGKSGIGKTTLLKTLFNPSPFRSNGQVYV
NEQVEAYDIRSLCSYISQDIVFSKGLIDMLKIANESAEEKQVLSLFELLGLNQLLEKL
PEGLNTKIDDNSNFSGGEKQRFSSIIIRGLLENKS
>GENE 5 2242 - 2463 231 73 aa, chain +
MFVDLLASTSEKLTGNRIVFAFEIIALVVSILMITVGMIONKTSQTGLSALNGGNDLFS
NSKERGMDRMTSI
>GENE 6 2585 - 4003 998 472 aa, chain +
MEENILSLIKQKQKLHLNELLKTFKDEELLMSCLKELQDQYKISWSKENVVYFIGEKYKV
GSIKINEKGFVVKDLNDVEQDYFVPPDSLNSITTDVVFTVYKESSEERYRANVEDISL
RVKSLFIDGEIQPSRDGRFLDFIPSEPGFKNYRIVMINSKDKFLKDDLKVKKILNVKEKK
LFTKIQKIIGDSNKAVDRIISIAEFNINPDPNRTLENADQVAIPINYEDEQVKRRLKN
SLVDKNLVTIDGSDSKDLDDAIYVEKTKDGYKLFVAIADVSYYVLPFSPLDNTALYRGNS
TYLANKVIPMLPEKLSNGVCSLNPNEKLCMVSEMDFDNNGVMKNKKVYESIMNSKARLT
YKEVNDLFERNVSNRDKIIVDMLLVSKELHELIDKERVSRGSIDFDVPEPKIVLDKESNV
VDIVPRDRGVSERLIENFMVSANESVAQIIIEKNLPYVYRNHGAPKEENLIE
>GENE 7 4010 - 4678 423 222 aa, chain +
LIRALGINVKLTDLKVNPKTIRMALDQISKQIEDQTERDVINVTLLKFMEKAAAYELENI
GHFGLASECYTHFTSPIRRYSGLMVHRYLKQYLIDKDLRDFKLDLNEKFINKACKIINET
EKNVNAEREVNVKVCMAEFMTKHIEKEYEGVVAAVLKFGFLVQLSNCVEGLIHISELPEF
TFDPKTNILVNKQNKVFRLGQKVKIKVKNADVKKRIIDFVLV
>GENE 8 4703 - 4768 72 21 aa, chain +
MGEHILLKNKKAYFNYEILD
>GENE 9 4880 - 5143 169 87 aa, chain +
MNIKKYAYANYVKQDPTRTRKLLLNKDEIKKILKRVQLENLTIIPLKLYLKGNKAKLEIG
IGKGGKLIDKRETIKRDIERRLNKIK

```

From:

<http://www.softberry.com/berry.phtml?topic=fgenesb&group=programs&subgroup=gfindb>

Eukaryotic Gene Prediction

Expression-based predictions align cDNA to define CDS' which is very accurate but with limitations such as cDNA obtainability

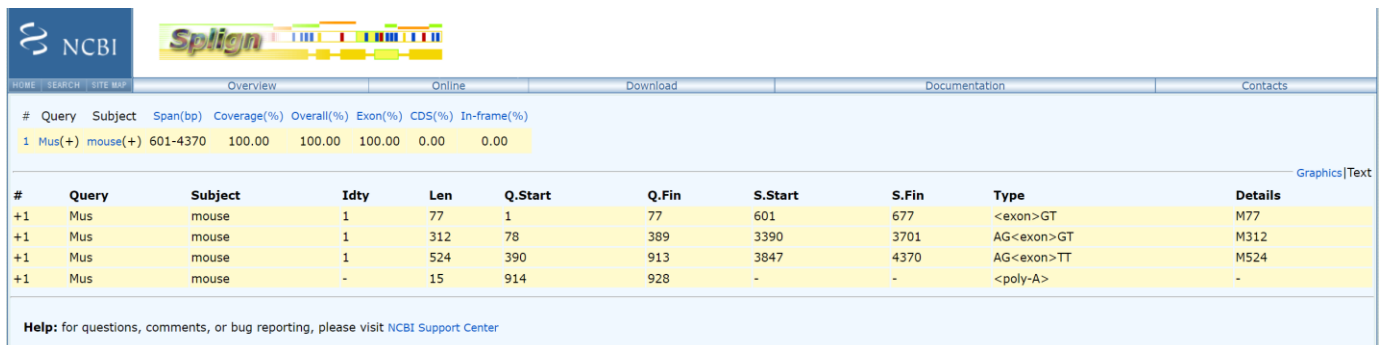
- Content sensors: prediction algorithms that depend on coding sequence features such as nucleotides.
 - Hexamer frequency most used as a content sensor using HMM training sets and probability to predict coding regions in unknown DNA
- Signal Sensors: finds genes based on specific sequence associated with a gene such as promoters, splice sites, stop codons
 - Uses the PSSM (weight matrix) to find conserved positions. Neural networks can also be used through AI training sets.

cDNA's can be aligned to the genome revealing the location of the gene and exon pattern

Programs that use this technique include:

Splign

Hosted by NCBI to align cDNA and genomic DNA and supports cross species alignments with cDNA input. *Results of an alignment where CDS and mRNA locations can be predicted in the figures below:*



#	Query	Subject	Span(bp)	Coverage(%)	Overall(%)	Exon(%)	CDS(%)	In-frame(%)
1	Mus(+)	mouse(+)	601-4370	100.00	100.00	100.00	0.00	0.00

#	Query	Subject	Idty	Len	Q.Start	Q.Fin	S.Start	S.Fin	Type	Details
+1	Mus	mouse	1	77	1	77	601	677	<exon>GT	M77
+1	Mus	mouse	1	312	78	389	3390	3701	AG<exon>GT	M312
+1	Mus	mouse	1	524	390	913	3847	4370	AG<exon>TT	M524
+1	Mus	mouse	-	15	914	928	-	-	<poly-A>	-

Help: for questions, comments, or bug reporting, please visit [NCBI Support Center](#)

Segments Alignment

1 2 3

```

1 AGAGAAGACCATAGGAATCTTGGAGTTACCTCCTCGCCCTACTGCTCTAGGAGACCTGTCACCAGCGTCT
|
601 AGAGAAGACCATAGGAATCTTGGAGTTACCTCCTCGCCCTACTGCTCTAGGAGACCTGTCACCAGCGTCT

71 GTGTCTG.....
|
671 GTGTCTGGTAAG

```

<https://www.ncbi.nlm.nih.gov/sutils/splign/splign.cgi?textpage=online&level=form>

Genomic BLAST

When cDNA is not available so uses EST seq (partial cDNA seq). Main search menu shown below.

Standard Protein BLAST

blastn **blastp** blastx tblastn tblastx

BLASTP programs search protein databases using a protein query. [more...](#)

Enter Query Sequence

Enter accession number(s), gi(s), or FASTA sequence(s) [?](#) [Clear](#)

Query subrange [?](#)

From

To

Or, upload file No file chosen [?](#)

Job Title

Enter a descriptive title for your BLAST search [?](#)

☐ Align two or more sequences [?](#)

Choose Search Set

Database [?](#)

Organism Optional ☐ exclude [Add organism](#)

Enter organism common name, binomial, or tax id. Only 20 top taxa will be shown. [?](#)

Exclude Optional ☐ Models (XM/XP) ☐ Non-redundant RefSeq proteins (WP) ☐ Uncultured/environmental sample sequences

Program Selection

Algorithm

☐ Quick BLASTP (Accelerated protein-protein BLAST)

☒ blastp (protein-protein BLAST)

☐ PSI-BLAST (Position-Specific Iterated BLAST)

☐ PHI-BLAST (Pattern Hit Initiated BLAST)


☐ DELTA-BLAST (Domain Enhanced Lookup Time Accelerated BLAST)

Choose a BLAST algorithm [?](#)

https://blast.ncbi.nlm.nih.gov/Blast.cgi?PROGRAM=blastp&PAGE_TYPE=BlastSearch&LINK_LOC=blasthome

BLAT

Aligns mRNA sequence to a genome without the uses of genomic DNA. *Figure below shows the human TP53 mRNA aligned with UCSC BLAT.*

 [Genomes](#) [Genome Browser](#) [Tools](#) [Mirrors](#) [Downloads](#) [My Data](#) [Help](#) [About Us](#)

Human BLAT Results

BLAT Search Results

Custom track name:

Custom track description:

Build a custom track with these results

ACTIONS	QUERY	SCORE	START	END	QSIZE	IDENTITY	CHRO	STRAND	START	END	SPAN
browser details	tp53_human_cds	1173	1	1182	1182	100.0%	17	-	7669609	7676594	6986
browser details	tp53_human_cds	69	688	813	1182	85.6%	1	+	3727130	3727462	333
browser details	tp53_human_cds	29	214	261	1182	96.8%	2	-	240095525	240095751	227
browser details	tp53_human_cds	27	480	539	1182	89.7%	14	-	72387839	72387897	59
browser details	tp53_human_cds	20	216	235	1182	100.0%	10	-	117946481	117946500	20

<https://genome.ucsc.edu/cgi-bin/hgBlat>

Genomic File Types

BED, WIG, SAM, BAM file formats

BED (browser extension data) files

Stores info on genomic intervals such as exons which have an "interval" from the start to end position

BED3: 3 column table with chromosome number, start, and end positions

Can be zero based position like UCSB genome browser where zero based systems have nucleotides between numbers so "start" is one off "true"

BED 6: chrom, start, end, name, score, strand

```
chr1 23401234 23401496 promoter 0 +
```

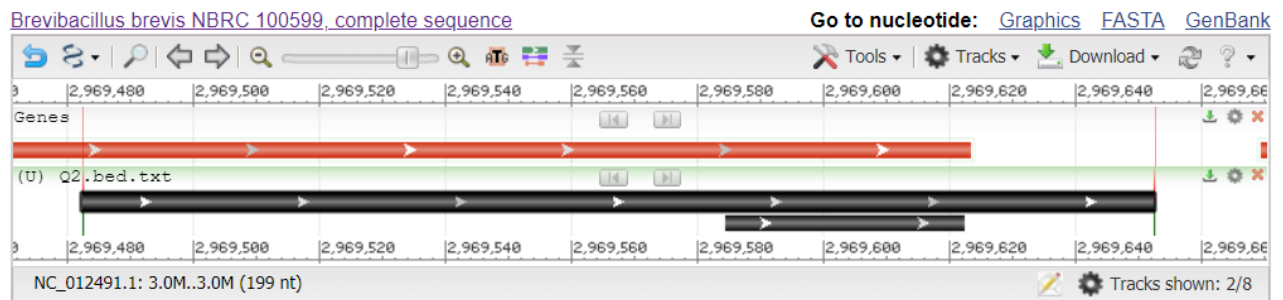
BED 12: chrom, start, end, name, score, strand, thickStart, thickEnd, itemRgb, blockCount, blockSizes, blockStarts

```
chr1 11873 14409 gene_1 0 + 11873 11873 0 3 354,109,1189, 0,739,1347,
```

Below is a BED6 file I created showing the transcription factor DdlR binding to the promoter region of the *ddlR-ddl* operon in *Brevibacillus brevis*.

```
chr1 2969482 2969652 promotor 0 +
chr1 2969584 2969622 5UTR 0 +
```

And this is the same BED file loaded into NCBI Genome Browser as a user custom track showing the promoter, 5' UTR region, and CDS of *ddlR*:



<https://www.ncbi.nlm.nih.gov/variation/view>

WIG (Wiggle Format) files:

Stores continuous data across large genomic locations.

Declaration line followed by chromosome position then data value

```
variableStep chrom=chr2
```

```
300701 12.5
```

```
300702 12.5
```

300703 12.5

300704 12.5

300705 12.5

BED and WIG files can be imbedded to the genome browser of your choosing such as NCBI variation viewer, UCSB genome browser, IGV, etc.

SAM (Sequence Alignment Map) files

Contains info on how sequences align to reference genome mainly results of NGS

There are 11 mandatory fields that represent a single alignment event:

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	[!-?A-~]{1,254}	Query template NAME
2	FLAG	Int	[0,2 ¹⁶ -1]	bitwise FLAG
3	RNAME	String	* [!-()+-<>-~] [!-~]*	Reference sequence NAME
4	POS	Int	[0,2 ³¹ -1]	1-based leftmost mapping POSition
5	MAPQ	Int	[0,2 ⁸ -1]	MAPping Quality
6	CIGAR	String	* ([0-9]+[MIDNSHPX=])+	CIGAR string
7	RNEXT	String	* = [!-()+-<>-~] [!-~]*	Ref. name of the mate/next read
8	PNEXT	Int	[0,2 ³¹ -1]	Position of the mate/next read
9	TLEN	Int	[-2 ³¹ +1,2 ³¹ -1]	observed Template LENgth
10	SEQ	String	* [A-Za-z=.]+	segment SEQUENCE
11	QUAL	String	[!-~]+	ASCII of Phred-scaled base QUALity+33

<https://learn.gencore.bio.nyu.edu/ngs-file-formats/sambam-format/>

BAM files

A compressed binary version of a SAM file that is used to store aligned sequences up to 128MB

BAM files contain a header section and an alignment section:

- **Header**—Contains information about the entire file, such as sample name, sample length, and alignment method. Alignments in the alignments section are associated with specific information in the header section.
- **Alignments**—Contains read name, read sequence, read quality, alignment information, and custom tags. The read name includes the chromosome, start coordinate, alignment quality and the match descriptor string.

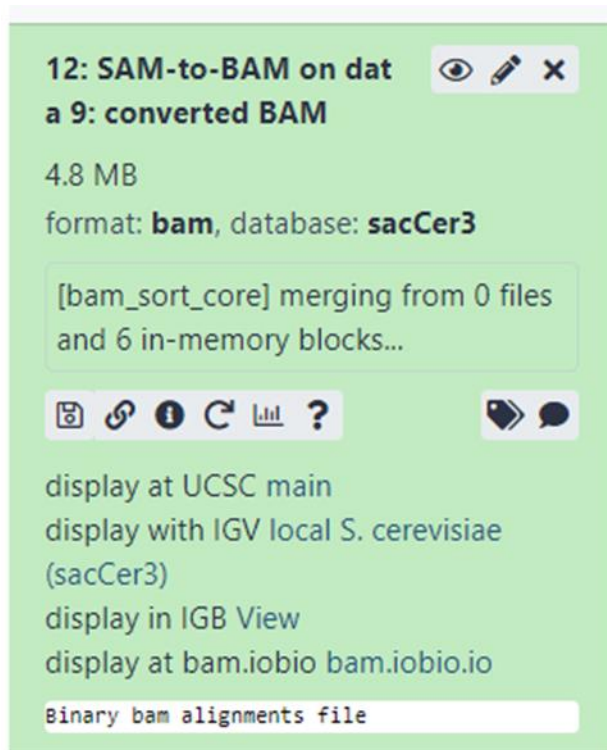
The alignments section includes the following information for each or read pair:

- **RG**: Read group, which indicates the number of reads for a specific sample.
- **BC**: Barcode tag, which indicates the demultiplexed sample ID associated with the read.
- **SM**: Single-end alignment quality.
- **AS**: Paired-end alignment quality.
- **NM**: Edit distance tag, which records the Levenshtein distance between the read and the reference.
- **XN**: Amplicon name tag, which records the amplicon tile ID associated with the read.

BAM index files (*.bam.bai) provide an index of the corresponding BAM file.

https://support.illumina.com/help/BS_App_RNASeq_Alignment_OLH_1000000006112/Content/Source/Informatics/BAM-Format.htm

SAM and WIG files can be converted and processed using SAMtools and BEDtools from Galaxy:



Saccharomyces cerevisiae reference genome aligned with a SAM file and converted from a SAM to a BAM file via the SAM-to-BAM tool in Galaxy.

Genome Browsers

One of the most important tools for visualizing data from both databases and user created datasets.

UCSC Genome Browser

Primarily an online eukaryotic genome browser that includes assembly data, precomputed comparative genomic data, mRNA, EST, and RefSeq gene alignments, and links to NCBI Map Viewer and Ensembl

Tracks

- UCSC genes: Gene prediction set based on RefSeq, Genbank, and UniProt
- Conservation: View conservation levels between 100 species
- PhastCons: Genome wide alignments between human, mouse, dog, zebrafish, fly using blastz

I personally used this browser a lot because it includes a table browser that allows you to directly input datasets and set search parameters at <https://genome.ucsc.edu/cgi-bin/hgTables>:

Table Browser

Use this tool to retrieve and export data from the Genome Browser annotation track database. You can limit retrieval based on data attributes and intersect or merge with data from another track, or retrieve DNA sequence covered by a track. [More...](#)

Select dataset

clade: genome: assembly:
group: track:
table: [describe table schema](#)

Define region of interest

region: ☐ genome ☒ position [lookup](#) [define regions](#)
identifiers (names/accessions): [paste list](#) [upload list](#)

Optional: Subset, combine, compare with another track

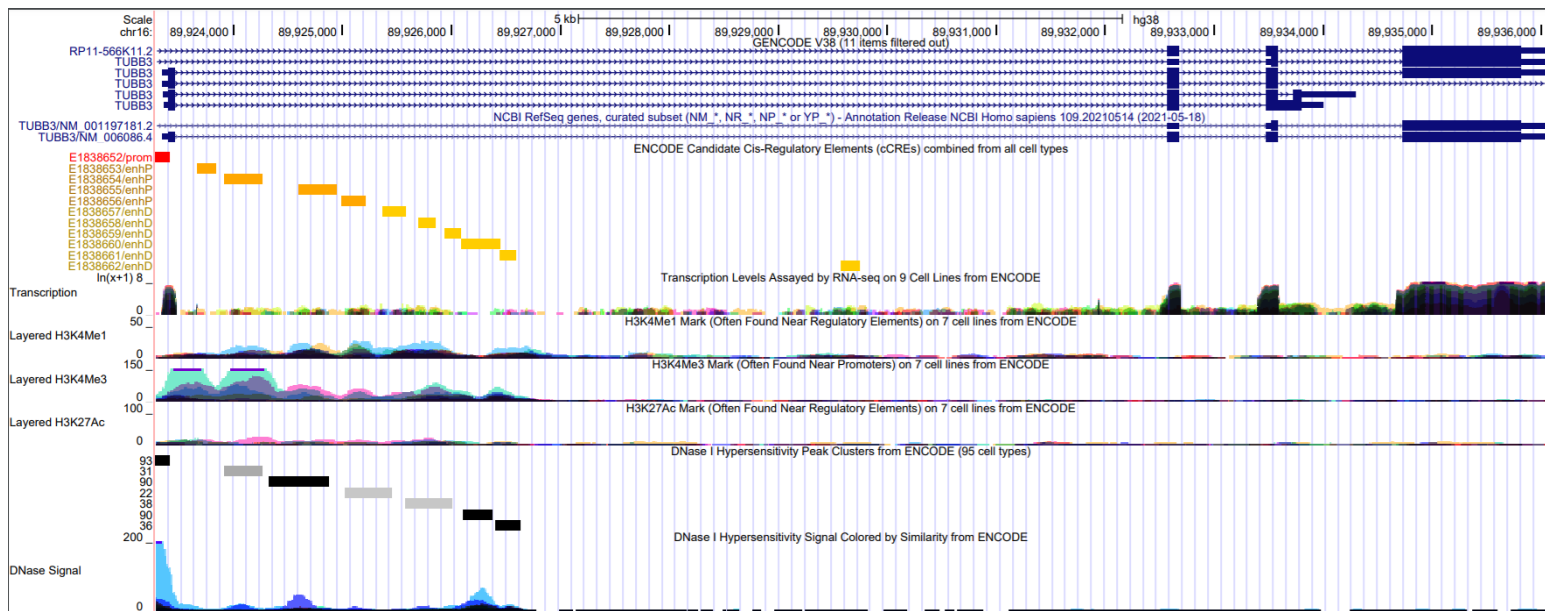
filter: [create](#)
intersection: [create](#)

Retrieve and display data

output format: Send output to ☒ Galaxy ☐ GREAT
output filename: (leave blank to keep output in browser)
file type returned: ☒ plain text ☐ gzip compressed

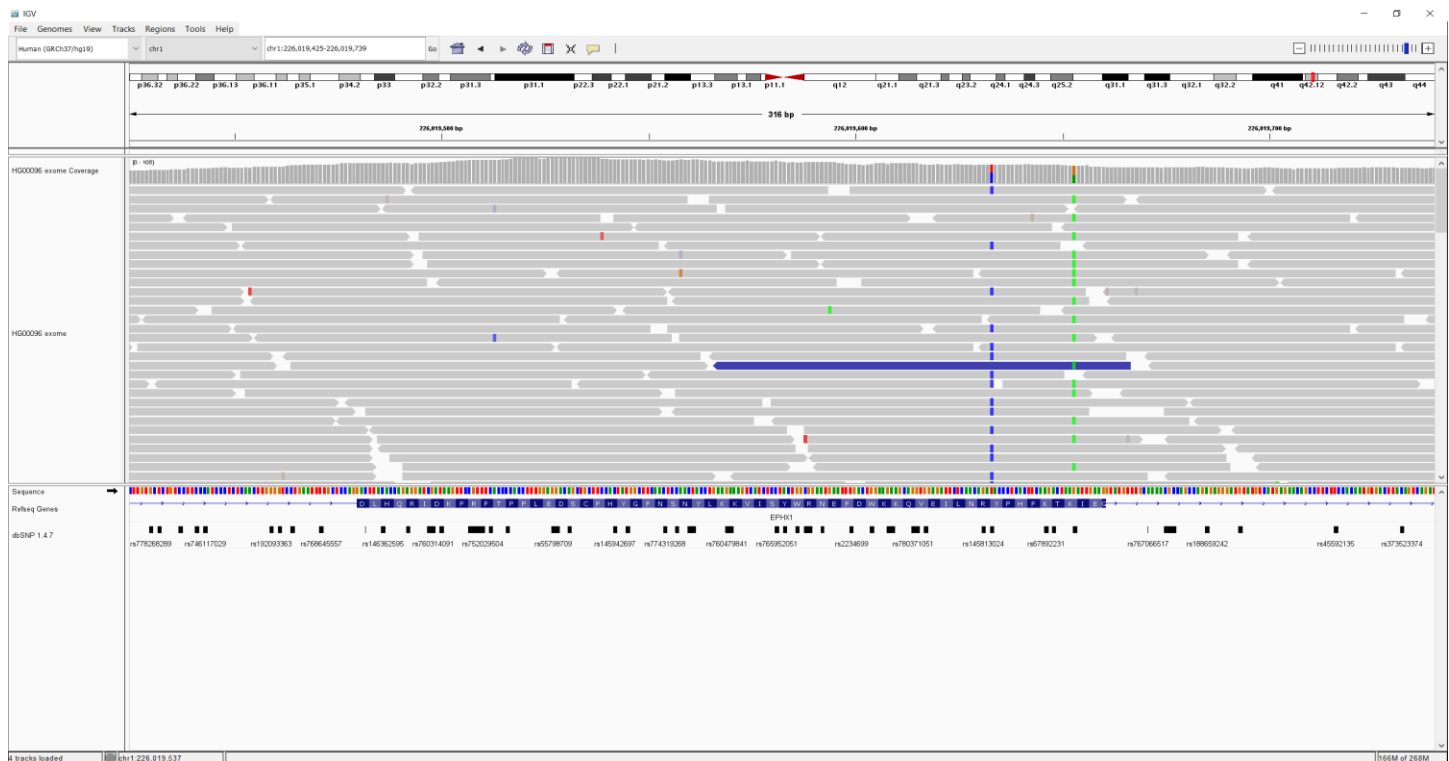
[get output](#) [summary/statistics](#)

When investigating human TUBB3 gene and looking for what histone modifications appear near the transcription start site of the TUBB3 gene, the UCSC Genome Browser was a useful tool to assist with data visualization shown the figure below:



IGV (Integrated Genome Viewer)

Genome browser that is available as local download outside of web browser that allows you to input your own datasets such as BED and BEDGRAPHS as user inputted tracks or downloaded datasets of websites (URL) and servers.

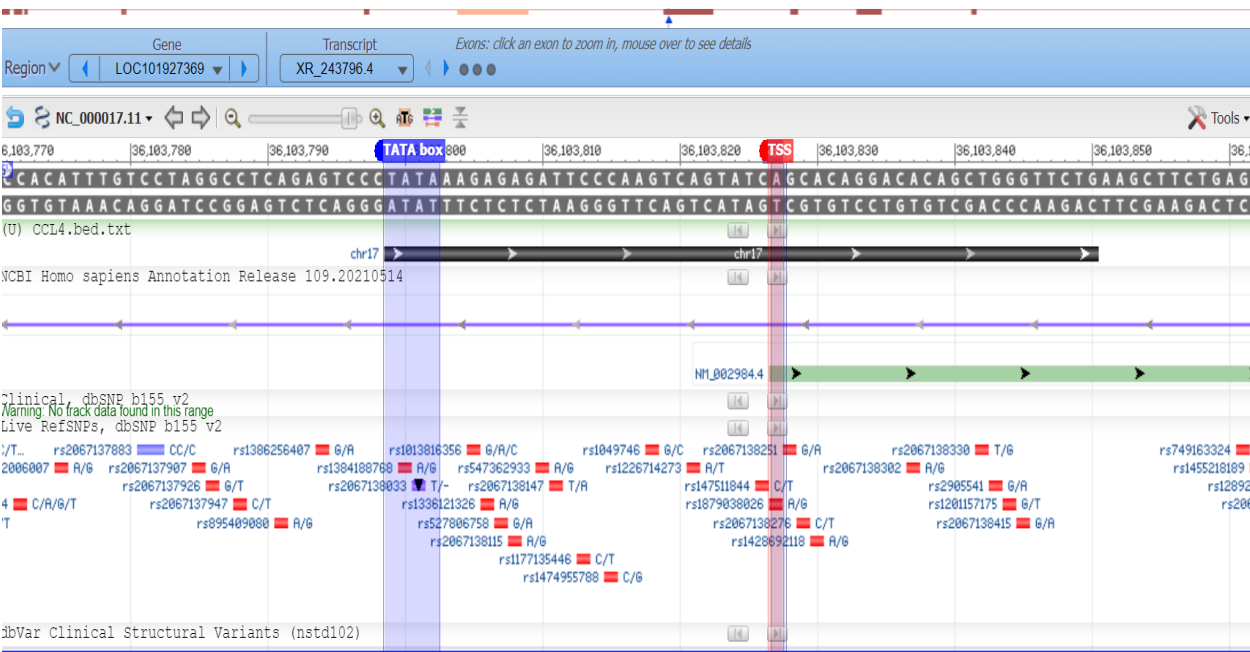


The figure above shows the EPHX1 gene in homo sapiens on the hg19 genome

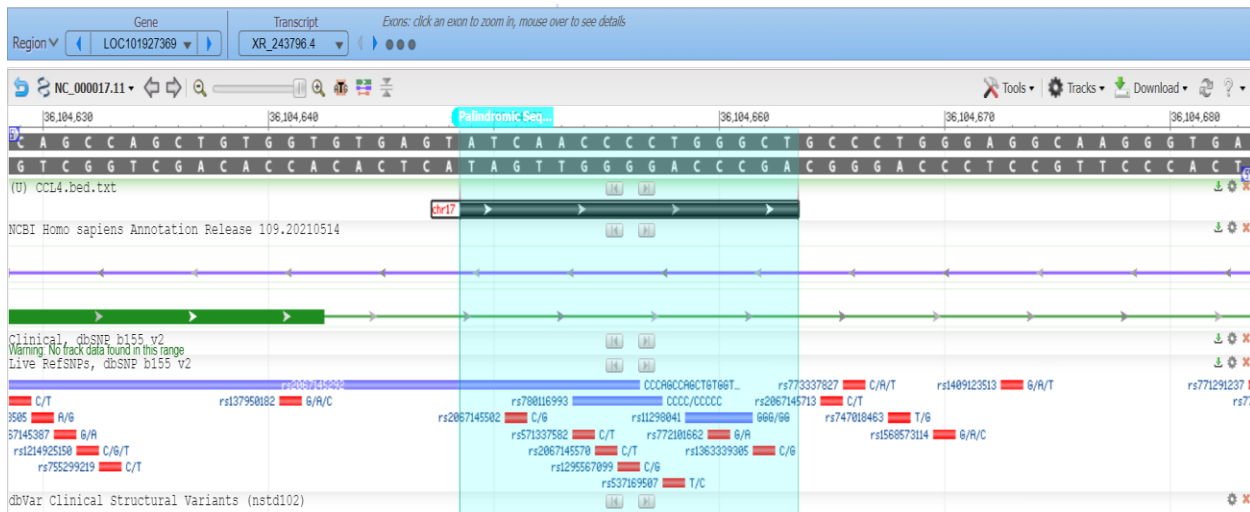
NCBI Variation Viewer

A genome browser that focuses on the human genome that is especially useful to view, search, and navigate variations housed in dbSNP, dbVar, and ClinVar in genomic context.

When researching the CCL4 gene and creating a BED file to find the transcription start site, TATA box, and a DNA location of the palindromic sequence, I used the Variation Viewer to align my used inputted BED files as tracks to compare with known dbSNPs.



Locations of the transcription start site and TATA box in relation to known SNPs



Palindromic sequence in the CCL4 gene and the known SNPs shown on the track below

Galaxy

Galaxy is one of the ultimate genomic tools that allows you to design, execute and visualize an entire genomic pipeline. It is a publicly available web service and package that allows for data analysis with over 100 tools.

- Allows for auto detection of files and has innate conversion capabilities
- Galaxy has data libraries where you can publish and share data/results
- Has a collaborative input (usually user files or from UCSC table browser) and output system (IGV, IGB, UCSC genome browser, ENSEMBL, vcf.io) and much more

BEDtools

One very useful tool that I have used is the BEDtools intersect feature which allows you to take two datasets and find intersects.

For example, when I was researching SNPs in the CFL2 gene (in cows), I used Galaxy's BEDtool to intersect the SNPs (in the form of a user created BED file) with the coding exons after they have matched with a BED file containing all the exons in CFL2

This workflow is shown in the Galaxy history below along with the results of the intersection:

History

search datasets

?
x

Unnamed history
5 shown
780 b

5: bedtools Intersect intervals on data 2 and data 3

3 regions
format: **bed**, database: ?

display in IGB View
display with IGV local

1.Chrom2.Start3.End4
chr214588913045889130C2213G
chr214588861145888611T1694A
chr214588841745888417G1500A

4: bedtools Intersect intervals on data 1 and data 3

2 regions
format: **bed**, database: ?

display in IGB View
display with IGV local

1.Chrom2.Start3.End4
chr214588913045889130C2213G
chr214588861145888611T1694A

3: cow2.bed

2: cow_cfl_all_exons(1).bed

1: cow_cfl_coding_exons(1).bed

It was shown that 3 SNP's intersect with all exons, and 2 SNP's intersect with coding exons

Similarly, to Biomart, BEDtools can also be run from the command line:

Page | 22

Here is example of the code I used to intersect two BED files on from a H3K4me3 cell line versus a reference genome:

```
$bedtools intersect -u -a hs_chr20_H3K4me3_for_linux.bed -b hs_chr20_refseq.bed
```

```
$bedtools intersect -u -a hs_chr20_H3K4me3_for_linux.bed -b hs_chr20_refseq.bed | wc -l
```

```
$bedtools intersect -v -a hs_chr20_H3K4me3_for_linux.bed -b hs_chr20_refseq.bed
```

```
$bedtools intersect -v -a hs_chr20_H3K4me3_for_linux.bed -b hs_chr20_refseq.bed | wc -l
```

NGS/Genome Assembly/Variant Calling Pipeline

NGS Technologies

NGS data can be used to explore metagenomics (the study of genetic materials from environmental samples).

The main technologies that are used are Illumina, ABI Solid, Ion torrent, Pacific, and Oxford

Genome Sequence Assembly

De novo Based assembly

Overlapping fragments assembled from scratch typical of new genomes

- There is usually a read length and error rate tradeoff for example PacBio has a long lead but high error rate. Vice versa for Illumina.
- Some examples of de novo assemblers are graphical approaches and Velvet

Reference Based Assembly

Useful for assembling human/cancer genomes based of the human hg38 genome. Alignment of trimmed data to FASTQ data to a reference genome is required.

- Some examples of aligners include: Bowtie, BWA, and HISAT

Variant Calling

Aligns NGS reads to a reference genome

- Sequencing errors, Heterozygous SNP, Homozygous SNP, and PCR amplification errors can cause mismatches
- Better coverage in sequencing helps variant calling easier to tell if this is a sequencing error
- The output is in the VCF format that shows the chromosomal position, the expected reference nucleotide, and the actual sequenced nucleotide, among many other statistics

Examples include: Freebayes, SAM tools, GATK, Platypus

Full Pipeline

Galaxy can execute this entire workflow with an input FASTQ file and produces a VCF file. The full pipeline is as follows:

FASTQ Groomer: Reformat the FASTQ file(s) to Sanger/Illumina 1.9 encoding.

FASTQ Trimmer or Trimmomatic: Trim low quality regions from FASTQ file(s).

Bowtie2/BWA/HISAT: Align reads in FASTQ file to a reference genome.

SAM-to-BAM: Convert output SAM file(s) to BAM file(s) if necessary.

Filter SAM or BAM (optional): Limit BAM file to a chromosome or a genomic region.

Sort BAM (optional): Sorts BAM file(s) in order based on chromosomal positions; may not be needed as some tools automatically sort BAM.

FreeBayes: Use aligned BAM file and reference genome to call variants; produces VCF file

This pipeline will be featured in my SNP paper later in this portfolio to find a specific SNP from a FASTQ file from the 1000 genomes database.

ChIP-seq and RNA-seq Pipeline

ChIP-seq

Stands for Chromatin immunoprecipitation sequencing

- ChIP experiment is followed by next-gen sequencing, aligned to a reference genome, and regions with many reads are histone associated regions
- Data reported in the FASTQ format and are deposited in the Sequence Read Archive (SRA) and/or the Gene Expression Omnibus (GEO)
- BWA and Bowtie allows for the alignment to the reference genome outputs a SAM file
- MACS2 and Slicer now do peak calling which indicates which regions of DNA are associated with the protein that was immunoprecipitated. Uses bam file so SAM file may need to be converted.

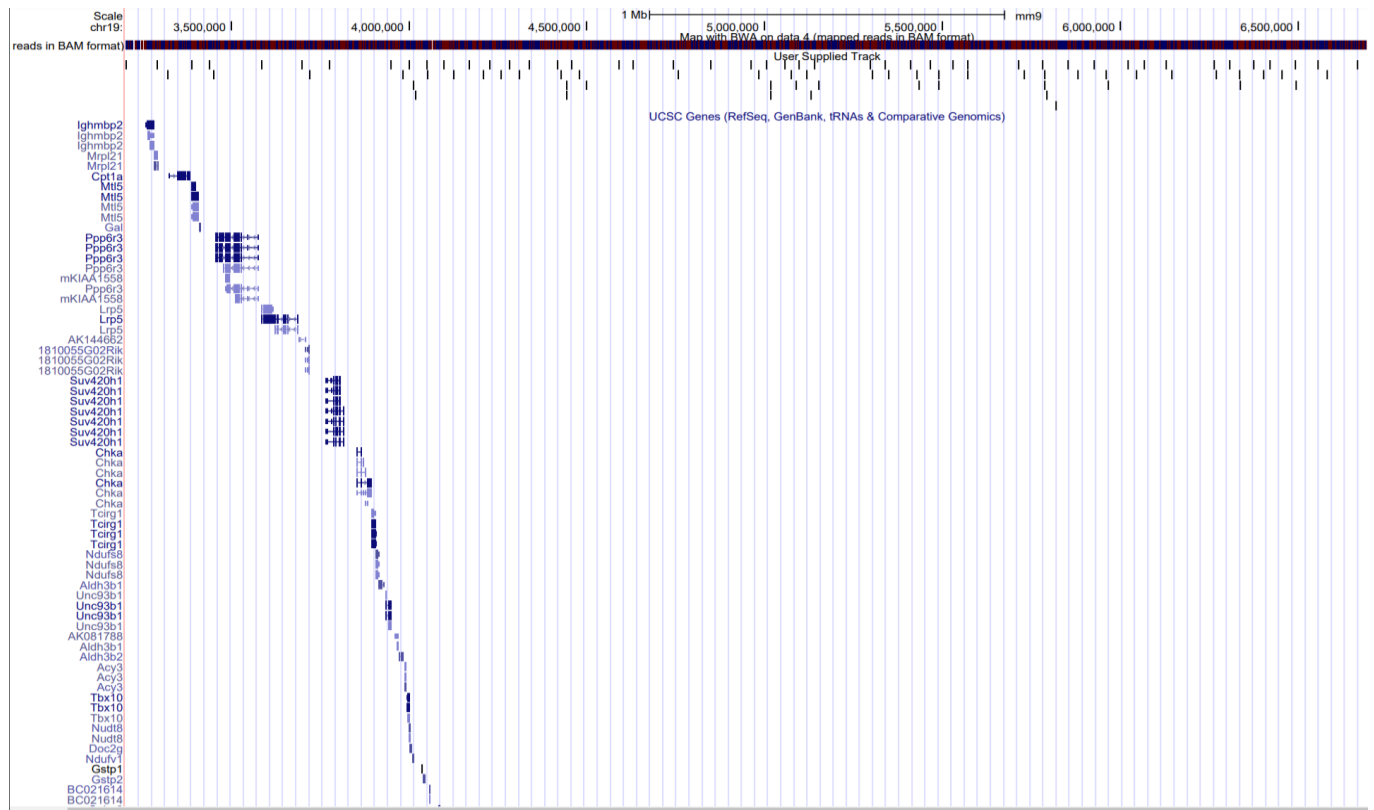
RNA-seq

Also known as massively parallel cDNA sequencing used to define transcription start sites, reveal undiscovered small noncoding RNAs and long noncoding RNAs

- Used to study transcriptomes of organisms with a complete or nearly complete reference genome. One tool is called the TopHat suite which has a bowtie aligner
- Cufflinks: probabilistic model that uses assembled reads and outputs complete transcripts.
- Stringtie: alternative to Cufflinks that performs transcript assembly and expression estimates in one step.

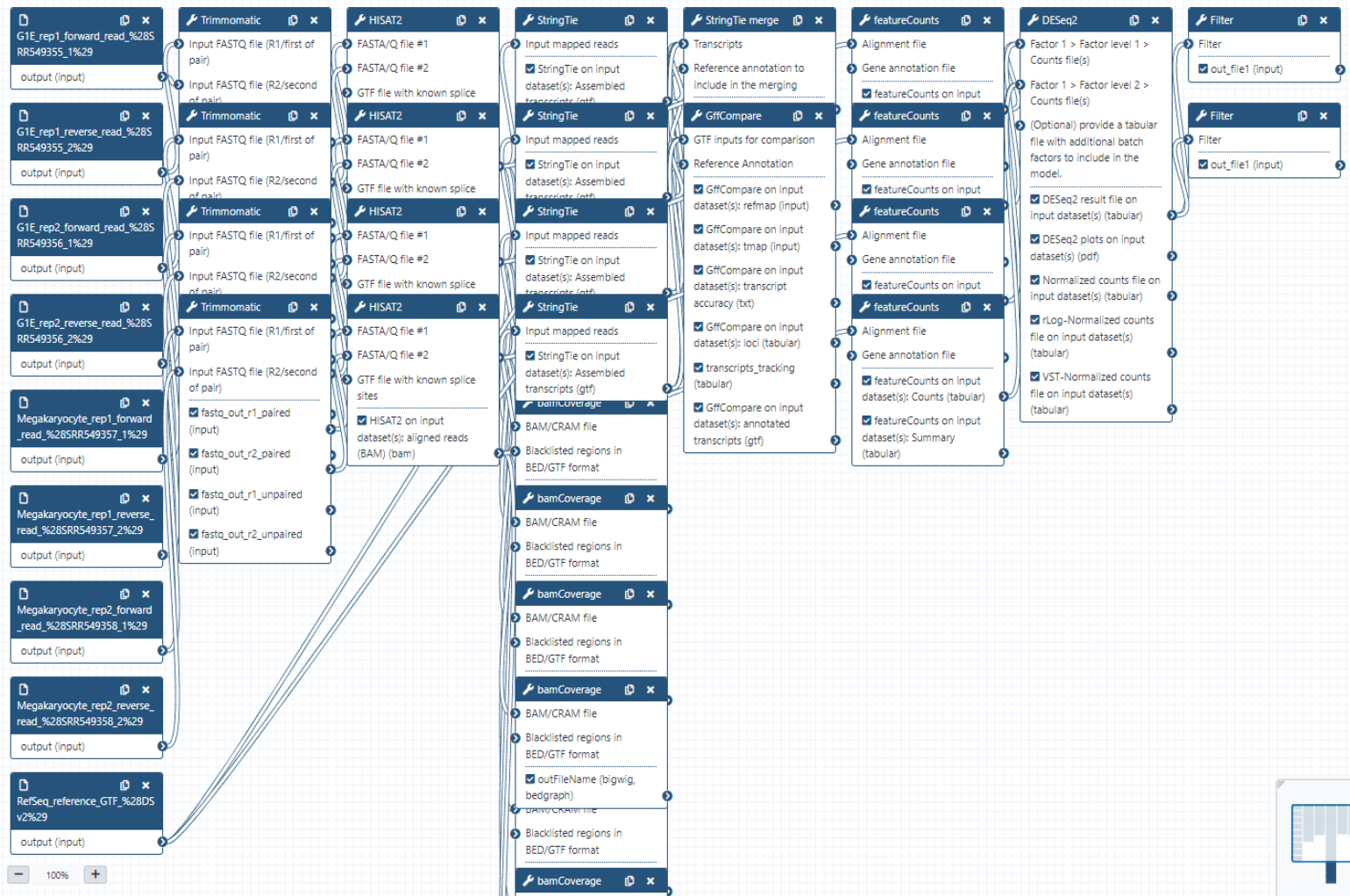
Galaxy can also run ChIP-seq and RNA-seq pipelines which is very similar to the variant calling pipeline. The Ch-IP pipeline is mostly the same except you run a MAC2 callpeak on the aligned BAM file instead of running Freebayes. This results in a MACS2 callpeak graph that can be visualized and compared with reference genes in the genome browser of your choice (IGV, UCSC, etc).

Figure below shows a mouse ChIP-seq reads in the FASTQ format ran through the ChIP pipeline in Galaxy. This produces a MACS2 Bedgraph (on the top track) that can be compared with the mm9 reference genome ChIP-seq genes (on the bottom track)

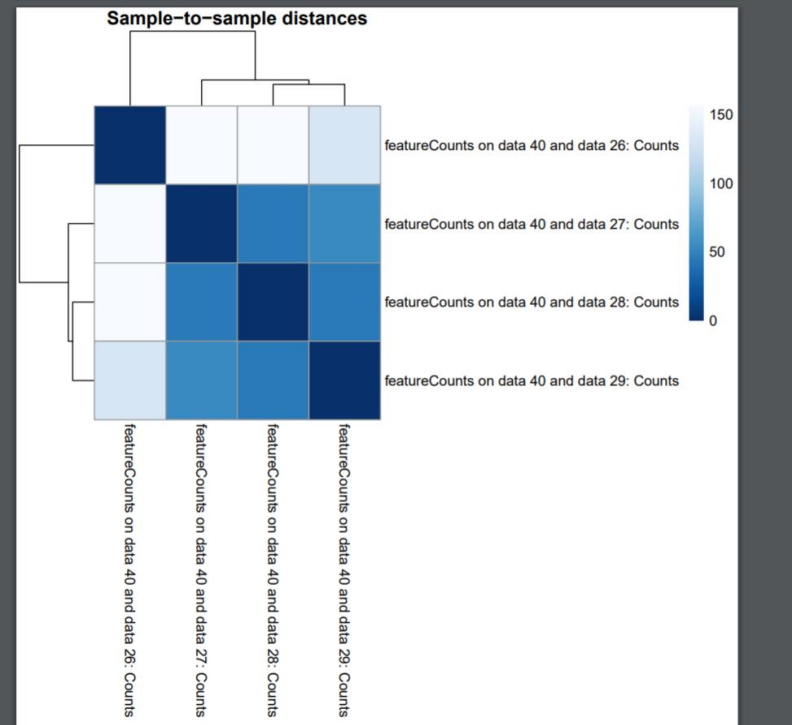
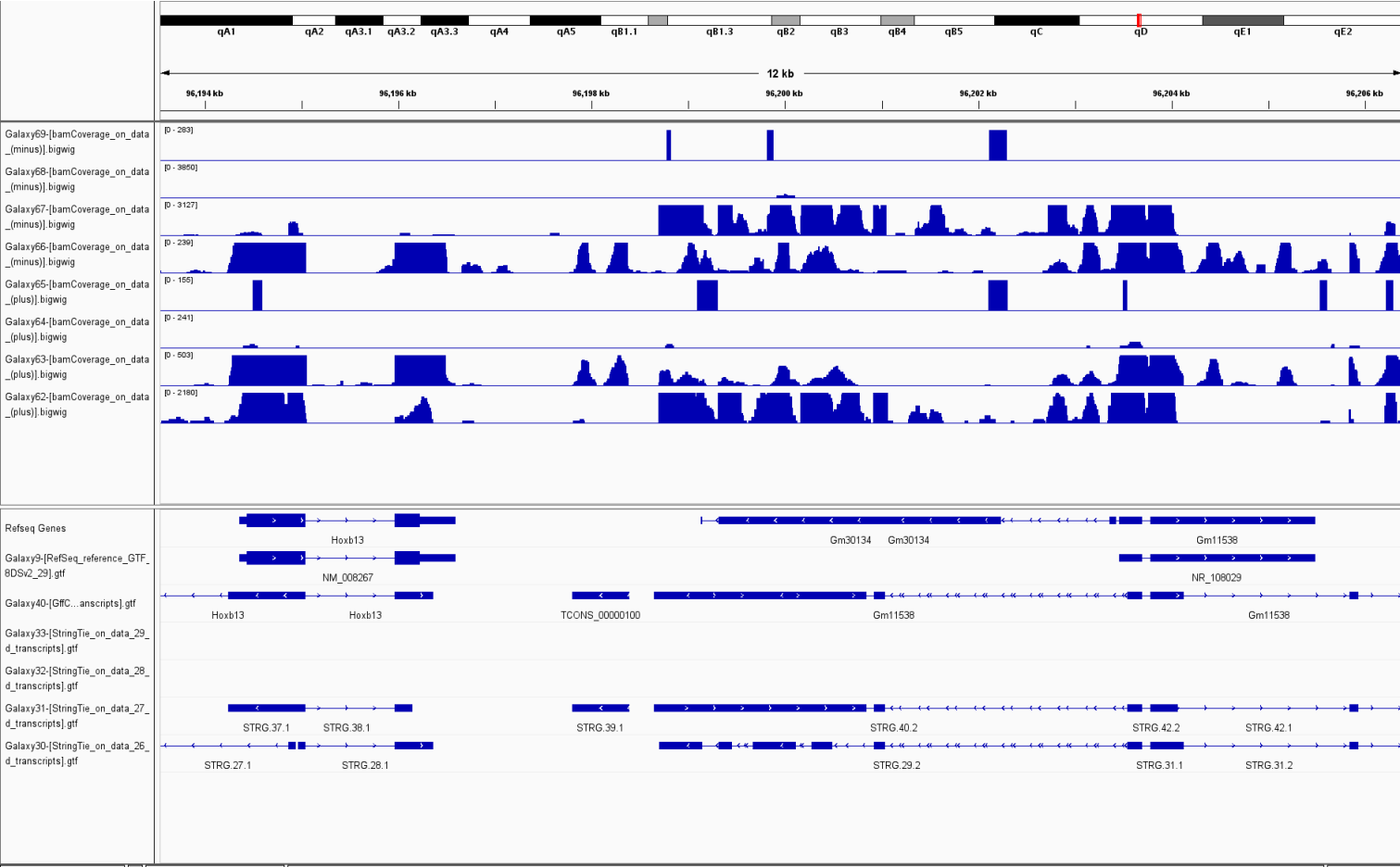


The RNA-seq pipeline is a little more complicated but shares the same steps as the previous pipelines except the reads are aligned and trimmed (ex. Trimmomatic) you run:
HISAT2→Stringtie→Gffcompare→feature counts→bam coverage

Full Galaxy workflow is in the diagram below which was used to run the analysis of a mouse (mm10) genome with the G1E and Megakaryocyte cell lines. There are 4 of each cell line with forward and reverse reads giving a total of 8 input reads:



Loading the 8 BIGWIG (BAM) files generated from the bamcoverage steps of the pipeline into IGV gives the peaks shown in the top tracks. Now we can compare these peaks with the mm10 genome (Refseq genes) and the Refseq reference gens (gtf files on the bottom tracks), to find similar gene expression values between the two cell lines. This gives us a good idea of the upregulation/downregulation of these cell lines. Bottom chart shows the output of the feature counts command comparing the two cell lines in a phylogenetic distance plot



An Investigation of 4 SNPs in the APOE Gene in a Subset of Mexican-American Individuals.

Bioinformatics: Tools for Genome Analysis

Final Paper Group 2

Johns Hopkins University

Christopher Gerth, Max Cheng, Sarah Weisberg

Abstract

Parkinson's Disease is a deadly disease of the brain and nervous system, and through statistical analysis, has been associated with various SNPs. In this investigation, four SNPs found in the promoter and genome region of the APOE protein are examined: rs449647, rs405509 (both found in the promoter region of the APOE gene) and rs7412, rs429358 (both found in exon 4 of the APOE gene). The population for study consisted of three randomly chosen male individuals of Mexican-American ethnicity, geographically located in California, U.S.A.. Male Mexican-American ethnicity in California, has been shown through other studies, to be at a higher risk of Parkinson's disease. Our hypothesis was that the SNPs we were looking for would be in the genotype of one or more of these individuals. Our Results show that even though male, Californian, Mexican-American ethnicity is associated with a higher prevalence of Parkinson's disease, none of these four SNPs were detected in our three individuals.

Introduction

APOE mainly functions in lipid transport between organs via the plasma and interstitial fluids playing a critical role in plasma and tissues lipid homeostasis. APOE is also a core component in the plasma lipoproteins and is involved in their production, conversion and clearance [18]. APOE's lipid transport role in the central nervous system depends on correct structure and concentration if it is to effectively interact with these lipoproteins.

APOE's protein "portion" is mostly transcribed by astrocytes and microglia and is lipidated by the ABCA transporter to form lipoprotein particles [11]. This lipoprotein is a vital part of the (Amyloid beta) A β metabolism where APOE is able to efficiently package soluble A β into a lipid-protein complex through a variety of cell surface receptors including LRP1, LDLR, and HSPG. This process allows for the transport of cholesterol from the astrocytes to the neuron and the breakdown of the amyloid compounds via lysosomal degradation. Failure to clear A β results in the buildup of A β oligomers and amyloid plaques causing neurofibrillary tangles or cerebral amyloid angiopathy (CAA) in nearby blood vessels. These conditions are indicative of neurodegenerative disorders such as Alzheimer's Disease (AD) and Parkinson's Disease (PD). Figure 1 summarizes these processes.

The APOE is also involved in lipoprotein transport and metabolism as such it's no surprise that this gene's transcription gets activated by lipoprotein related receptors. Some of those receptors include liver X receptor, peroxisome proliferator-activated receptor γ and nuclear receptors. The

ligands that bind the liver X receptor include oxysterols which are the oxygenated derivatives of cholesterol [2]. This receptor can also be found in other genes and proteins such as ABC, CETP, FAS, CYP7A1, LPL and ChREBP [2]. This receptor plays a key role in cholesterol metabolism. Peroxisome proliferator-activated receptor γ is a nuclear receptor protein that acts as a transcription factor [6]. This receptor also plays a key role in carbohydrate, lipid and protein metabolism.

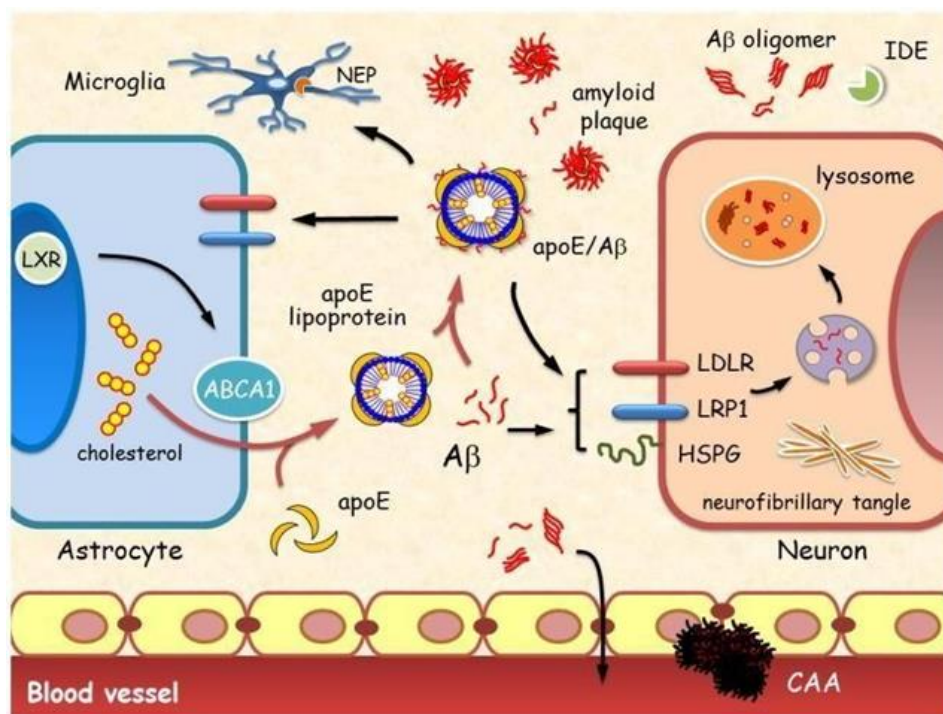


Fig 1. A Diagram showing Apolipoprotein E and amyloid- β metabolism in the brain. [7]

Apolipoprotein (APOE) is a 299 AA long protein that contains multiple amphipathic α -helices bundled in groups of 4 at the non-polar N-terminus [14]. The C-terminus also contains α -helices along with a low-density lipoprotein receptor (LDLR)-binding site. The transcribed APOE protein can be divided by its C and N terminal domains (fig 2) [20]. The C terminal mainly binds lipids while the N terminal mainly contains receptors for its regulation. The main classes of receptors are low-density lipoprotein receptor (LDLR), very-low-density lipoprotein receptor (VLDLR) and apoE receptor 2 (apoER2) [20].

Genetically speaking, the SNPs of APOE pose a big risk factor for the development of AD and PD. For example, APOE can form 4 different isoforms by a combination of allele types from the SNPs rs429358 and rs7412 (found in exon 3 of the coding region and which are designated by episilons, see Table 1). It has been found that: "APOE e2 ...reduces the risk of Alzheimer's; APOE

e4...increases the risk of Alzheimer's and is associated with getting the disease at an earlier age. APOE e3...the most common [allele]... doesn't seem to affect the risk of Alzheimer's." [24]. These isoform types also affect the predisposition to PD in the same way. Further, in a study done with individuals aged 50-59 years old, there was a 40.7 percent chance of developing AD in carriers (epsilon 4) compared to 8.2 percent in non-carriers [10]. In addition, APOE4 also has been found to contribute to multiple brain homeostatic pathways including synaptic integrity and plasticity, glucose metabolism pathways, and cerebrovascular pathways [21]. Further, in a previous study, it was found that the SNPs rs449647-A, rs405509-G were overrepresented in female PD patients from Eastern India. [25]

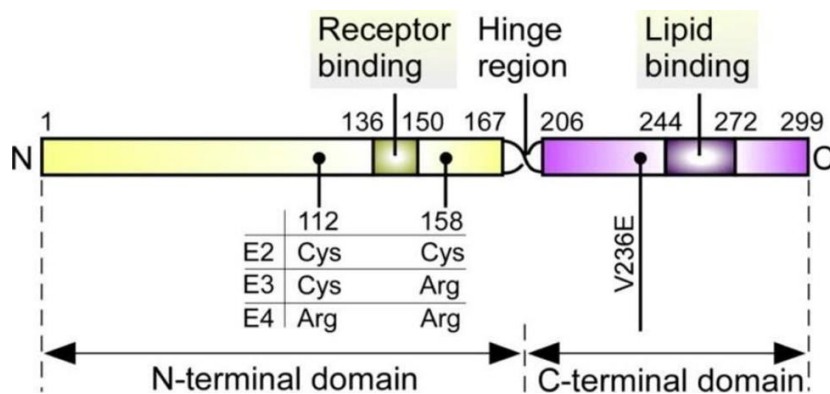


Fig 2. Demonstrates the C and N terminal of the transcribed APOE gene [20].

The APOE gene is located on chr19:44905796-44909393 (hg 38) and has a Coding Region at Chr19:44,906,625-44,909,250 with a Coding Exon Count of 3. A promoter region is found at Chr19:44905553-44905892. [17] (see Fig. 3).

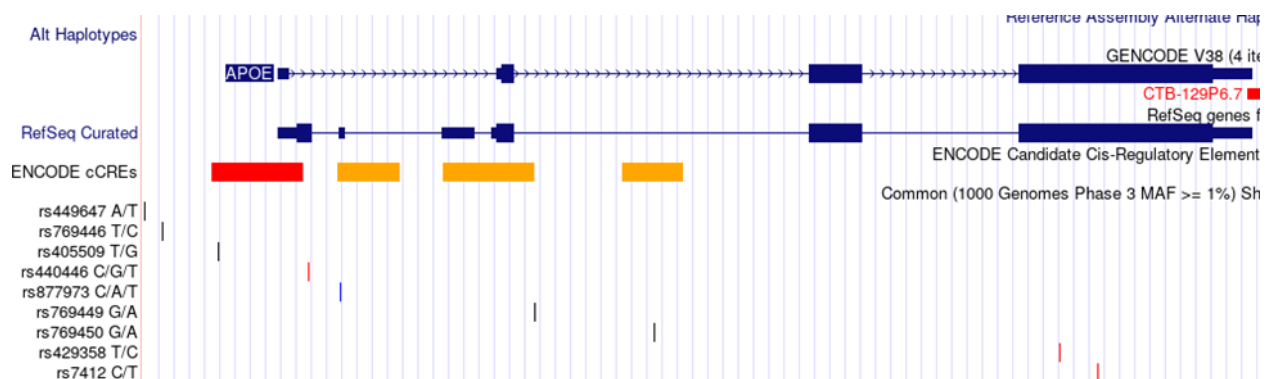


Fig 3. The APOE gene, its associated control regions, SNPS and the exon/intron structure. The promoter region is shown in red, rs405509 can be seen; rs429358 and rs7412 can be seen in the exon region of APOE and lead to the various isoforms. The position of rs449647 can also be seen, this SNP is also associated with an increased risk of PD. [17]

rs429358	rs7412	Epsilon designation
C	T	ε1
T	T	ε2
T	C	ε3
C	C	ε4

Table 1. The different alleles of APOE4, specified as epsilons, [15].

A particular SNP of interest and based on our initial SNP selection, is SNP rs405509 found in the promoter region of the APOE gene. The SNP rs405509 is considered a non-Mendelian factor along with several other polymorphisms related to PD. The rs405509 SNP causes a change from G-T at position -219. People who have the -219T allele have been found to be more susceptible to myocardial infarction. Moreover, people with this SNP have been found to have lower APOE plasma concentration, associated with an increased PD risk in both males and female subjects. Another study found that the rs405509 SNP is associated with increasing numbers of diseased vessels (see Fig. 3). Given the above information about other SNPs found in the APOE gene, this study seeks to probe the expression of not only the main SNP found in the promoter region, but four SNPs in a population of high susceptibility to PD. These SNPs are: rs449647, rs405509, rs7412 and rs429358.

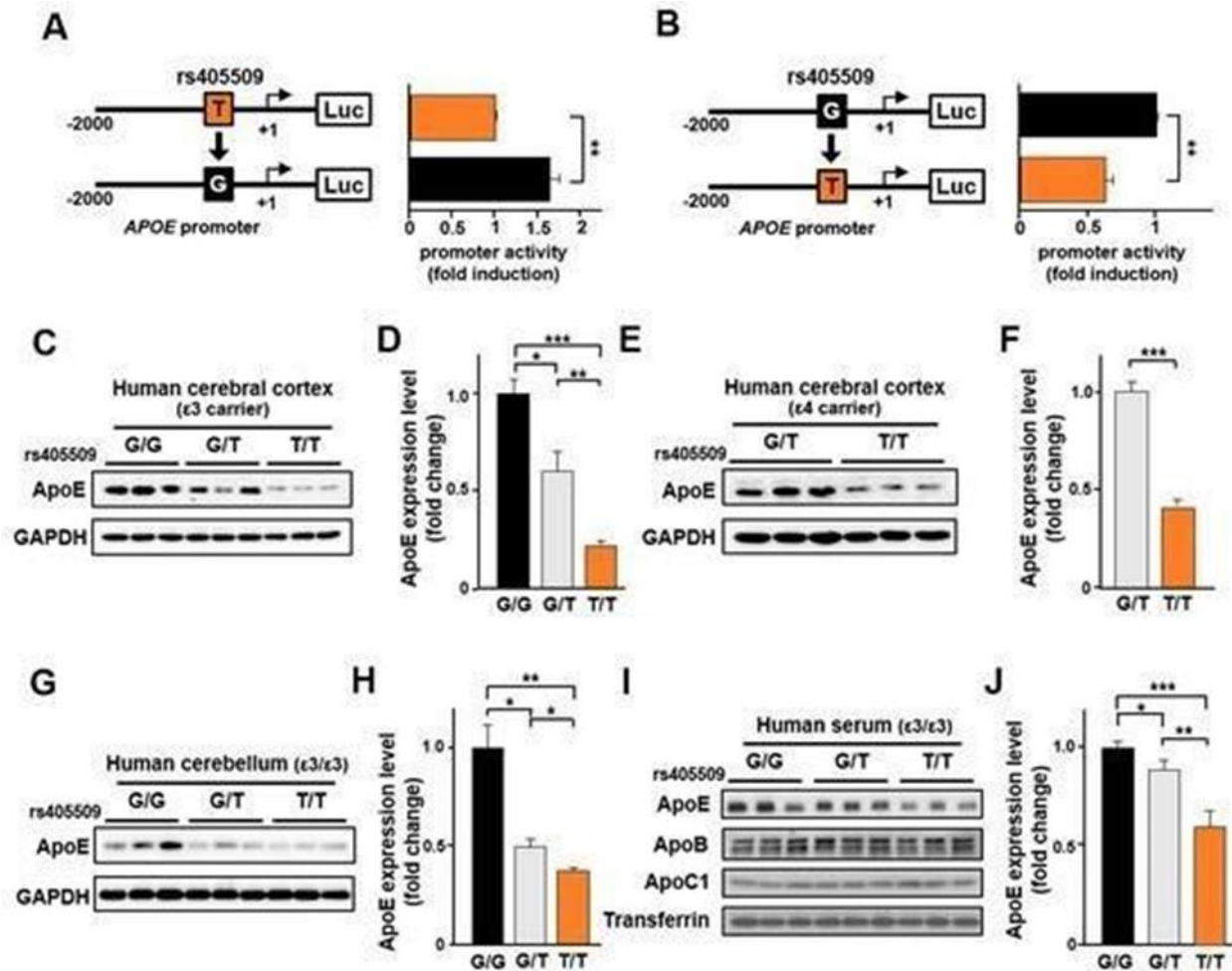


Fig 4. Rs405509 T-allele reduces APOE expression when the T allele is present. Graphs A and B show promoter activity with T to G and G to T substitutions in rs405509 respectively and the resulting promoter activity. Graphs C-J show the expression of APOE of different allelic combinations in different regions of the brain [2].

Rs405509 is part of the promoter region of APOE gene which plays a major role in lipid transport and management especially in the brain. Neurodegenerative diseases such as Parkinson's and Alzheimers can both be attributed to the APOE gene and causes an inefficiency in packaging soluble A β into a lipid-protein complex. This leads to the build up of amyloid plaques which are characteristics for both risk and progression of the aforementioned neurodegenerative disorders. Studies in Alzheimer's patients have found that those with the T-allele tend to have a younger onset and more severe cortical atrophy than those with G-allele [2] for this specific substitution. A reporter gene assay showed that there was a significant difference in APOE expression with the SNP t allele causing reduced activity.

The APOE gene contains many SNPs some of which have been found to be restricted to certain populations. Based on a Medicare study from 2010, it was found that Americans of Mexican heritage residing in and around Los Angeles, CA. area had a higher disposition to PD [19]. The SNP which we are studying (rs405509) has been found to be mostly associated with the elderly population [22], however there are differences between Mexican groups [23]. The closest approximation that could be found is the map represented below (Fig. 5).



Fig. 5. Demonstrates the variability of the rs405509 SNP among the Mexican population. (1) Durango. (2) Mexico City. (3) Guanajuato, (4) Jalisco. (5) Michoacán. (6) Morelos, (7) Nayarit. (8) Oaxaca, (9) Yucatán, (10) Zacatecas [23].

<i>Population</i>	<i>N</i>	<i>APOE Genotype</i>						<i>Hardy-Weinberg Equilibrium</i>	
		<i>*4/*4</i>	<i>*3/*4</i>	<i>*3/*3</i>	<i>*2/*3</i>	<i>*2/*4</i>	<i>*2/*2</i>	χ^2	<i>p</i>
Guadalajara, Jalisco	179	0.5	12.9	72.1	10.6	2.8	1.1	4.919	0.42
Rural newborns	172	0.6	16.9	73.8	8.1	0.6	0	0.667	0.98
Tepic, Nayarit	61	0	22.9	73.7	3.3	0	0	1.389	0.92
Durango, Durango	30	0	23.3	76.7	0	0	0	0.523	0.76
Total	442	0.23	19	74.2	5.4	0.9	0.23	4.220	0.51
Huichol Indians, El Nayar, Nayarit	40	7.5	42.5	50	0	0	0	0.055	0.97

Table 2. APOE Genotype Frequencies in Western and Northwestern Mexican population [23].

Hypothesis

Based on the above data, we believe that individuals of Mexican-American descent, who are prone to a higher incidence of PD should have an associated increase in the alleles associated with PD. Therefore, we are looking at the following SNPs to detect this: rs449647, rs405509, rs7412, rs429358.

SNP and change	Location (hg 38)	Significance
rs449647 G->A	chr19:44905307	SNP in the enhancer region.
rs405509 T->G	chr19:44905579	our main SNP in the promoter region.
rs7412 (see table 1)	chr19: 44908822	One of the SNPs located in exon 4 involved in producing the APOE4 protein
rs429358 (see table 1)	chr19:44908684	The second SNP located in exon 4 involved in producing the APOE4 protein

Table 3: Description of the 3 SNP we were looking for in our 4 individuals and the associated alleles

Methods

We sampled 3 male individuals of Mexican American descent from 1000 genomes as clinical ethnicity studies have shown that Mexican-Americans have a higher susceptibility to this condition (data found above). Since we cannot determine whether these patients have Parkinson's disease due to clinical/HIPAA restrictions on genome analysis, we believe that surveying an ethnicity group with the highest chance for PD was our best option. Thus, the incidence rate of the SNPs that we researched should be higher in this population.

Three randomly chosen whole genome paired sequenced files of males of Mexican-American ethnicity living in California were chosen for investigation from the 1000 genomes web site. The sample numbers and associated FASTQ files were:

1. NA19798

ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR793/SRR793155/SRR793155_1.fastq.gz

ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR793/SRR793155/SRR793155_2.fastq.gz

2. NA19661

ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR034/SRR034567/SRR034567_1.fastq.gz

ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR034/SRR034567/SRR034567_2.fastq.gz

3. NA19685

ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR034/SRR034810/SRR034810_1.fastq.gz

ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR034/SRR034810/SRR034810_2.fastq.gz

The data were chosen from two sections of the 1000 genomes database: 30X GRCH 38 and Phase 3. 30XGRCh38 uses the hg 38 assembly and Phase 3 uses the hg 37 assembly as a reference genome. These two different databases were used because NA19798 could only be found in the 30XGRCh38 and not the higher quality Phase 3 database.

Once the samples were chosen they were imported into Galaxy. Each paired-end file was initially examined using the FASTQC tool and was trimmed according to an average base score of 30 using a sliding window of 4 via the trimmomatic tool. Each sample was then aligned to the appropriate reference genome, hg38 or hg37 using the BowTie program with the paired-end reads option chosen. SNP calling was done by using the FreeBayes tool producing a VCF file. VCF filtering was done to a read depth of 10 or greater, but the data was not filtered for heterozygosity due to the nature of the APOE4 genotype.

To determine if our particular SNPs were present two bed files were imported from the UCSC Genome Table Browser, the hg38 using the All SNPs 147 database and hg37 using the All SNPs

147 database. The SNP coordinates and names were compared against the two bed files to determine if the SNPs of interest were present.

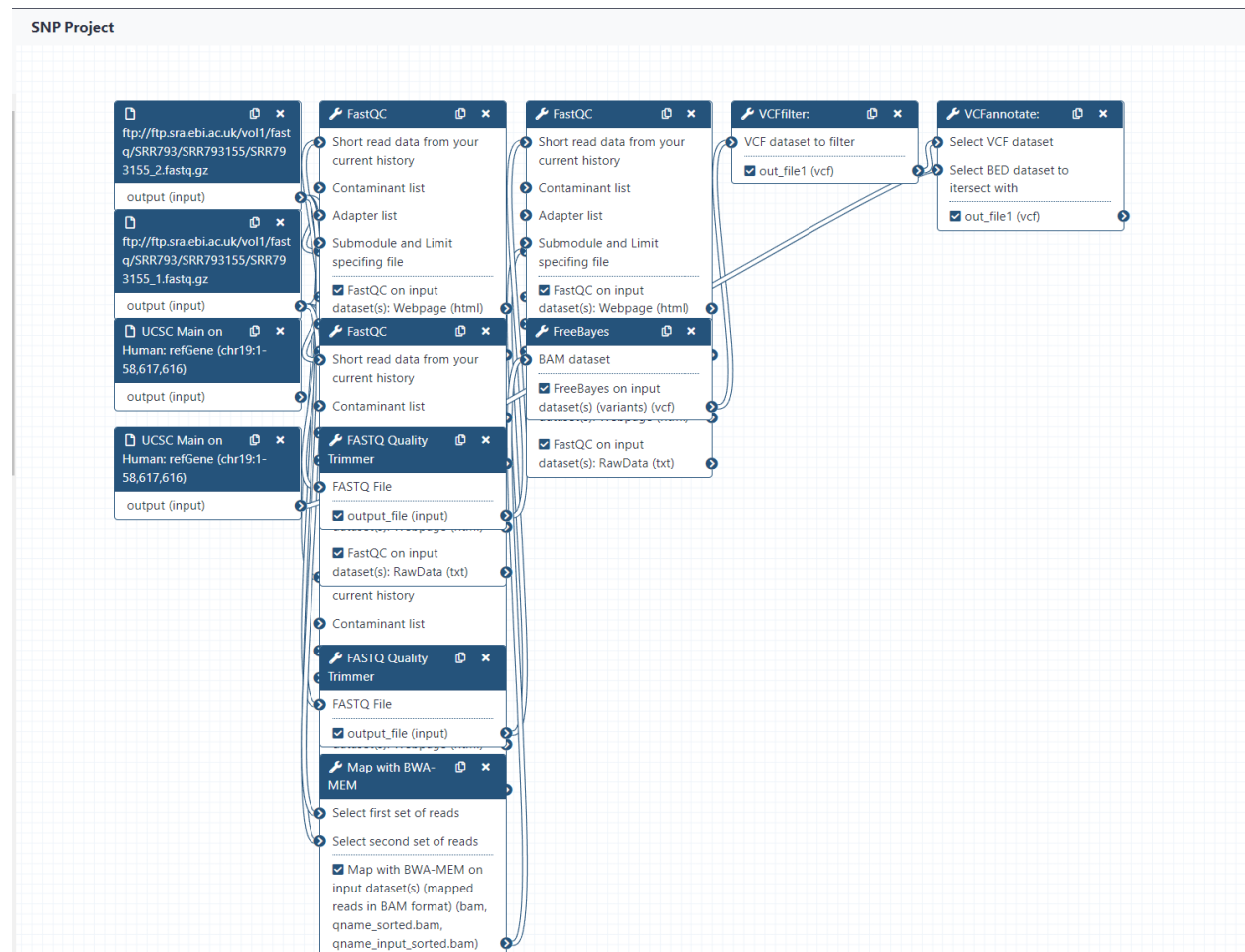


Fig. 6 Galaxy Workflow diagram showing the complete SNP pipeline process including the input files and the VCF output file

Results

<pre>##INFO= (ID=MQMAltNumber=1,Type=Integer,Description="Number of unique non-reference alleles in called genotypes at this position") ##INFO= (ID=MQMAltNumber=1,Type=Float,Description="Base number of unique non-reference allele observations per sample with the corresponding alternate alleles") ##INFO= (ID=MQMNumber=A,Type=Float,Description="Mean mapping quality of observed alternate alleles") ##INFO= (ID=MQMNumber=A,Type=Float,Description="Mean mapping quality of observed reference alleles") ##INFO= (ID=PAIRED,Number=1,Type=Float,Description="Proportion of observed alternate alleles which are supported by properly paired read fragments") ##INFO= (ID=PAIRED,Number=1,Type=Float,Description="Proportion of observed reference alleles which are supported by properly paired read fragments") ##INFO= (ID=MIN_DP,Number=1,Type=Integer,Description="Minimum depth in gVCF output block") ##INFO= (ID=END,Number=1,Type=Integer,Description="Last position (inclusive) in gVCF output record") ##FORMAT= (ID=GT,Number=1,Type=String,Description="Genotype") ##FORMAT= (ID=GQ,Number=1,Type=Float,Description="Genotype Quality: the Phred scaled marginal (or unconditional) probability of the called genotype") ##FORMAT= (ID=GL,Number=0,Type=Float,Description="Genotype Likelihood: log10-scaled likelihoods of the data given the called genotype for each possible genotype generated from the reference and alternate alleles given the sample ploidy") ##FORMAT= (ID=DP,Number=1,Type=Integer,Description="Read Depth") ##FORMAT= (ID=AD,Number=0,Type=Integer,Description="Number of observation for each allele") ##FORMAT= (ID=RD,Number=1,Type=Integer,Description="Reference allele observation count") ##FORMAT= (ID=QD,Number=1,Type=Integer,Description="Sum of quality of the reference observations") ##FORMAT= (ID=AQ,Number=A,Type=Integer,Description="Alternate allele observation count") ##FORMAT= (ID=QA,Number=A,Type=Integer,Description="Sum of quality of the alternate observations") ##FORMAT= (ID=MIN_DP,Number=1,Type=Integer,Description="Minimum depth in gVCF output block") ##INFO= (ID=BED-features,Number=1,Type=String,Description="Annotation from chr4:41941-41941/objects/0/7/0/dataset_3799546-716-422-449-ccdc53b15e.dat delimited by \t")</pre>									
CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO		
chr19	44955312	.	A	G	325.418	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=7.945461PPR=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955313	.	A	G	125.418	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=7.945461PPR=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955314	.	C	A	125.418	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=7.945461PPR=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955316	.	G	T	325.418	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=7.945461PPR=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955317	.	A	C	325.418	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=7.945461PPR=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955319	.	C	G	312.714	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=4.786961PPR=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955322	.	A	C	311.451	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=3.2877;EPPE=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955323	.	C	G	311.451	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=3.2877;EPPE=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955326	.	T	A	311.451	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=3.2877;EPPE=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955327	.	C	G	311.451	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=3.2877;EPPE=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955328	.	C	C	311.451	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=3.2877;EPPE=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955329	.	A	C	311.451	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=3.2877;EPPE=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955338	.	C	T	311.451	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=3.2877;EPPE=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955337	.	T	C	351.251	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=3.2877;EPPE=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955338	.	C	T	349.8	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=4.786961PPR=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955339	.	A	C	349.8	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=4.786961PPR=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955341	.	C	C	349.8	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=4.786961PPR=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955342	.	A	T	349.8	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=4.786961PPR=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955343	.	G	A	349.8	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=4.786961PPR=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955344	.	A	T	349.8	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=4.786961PPR=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955345	.	C	G	349.8	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=4.786961PPR=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955346	.	A	T	349.8	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=4.786961PPR=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955347	.	A	C	349.8	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=4.786961PPR=0GT=0;EN=1;MEANAL=1;MQM=62		
chr19	44955348	.	C	A	349.8	.	AB=2ABP=0AC=2AF=1AN=2AO=11EQCAR=1XCP=11CPB=11CPRA=0EFF=4.786961PPR=0GT=0;EN=1;MEANAL=1;MQM=62		

Fig 7. VCF annotation of Mexican American Male from NA19798 and FASTQ files
ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR793/SRR793155/SRR793155_1.fastq.gz
ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR793/SRR793155/SRR793155_2.fastq.gz Paired-end
files used on hg37 genome from UCSC Table browser in the region 44,900,000 -
45,000,000 (where the APOE gene and our SNP are).

Workflow Visualize Shared Data Help User							
##INFO=<ID=MQM,Number=A,Type=Float,Description="Mean mapping quality of observed alternate alleles">							
##INFO=<ID=MQMR,Number=1,Type=Float,Description="Mean mapping quality of observed reference alleles">							
##INFO=<ID=PAIRED,Number=A,Type=Float,Description="Proportion of observed alternate alleles which are supported by properly paired read fragments">							
##INFO=<ID=PAIREDR,Number=1,Type=Float,Description="Proportion of observed reference alleles which are supported by properly paired read fragments">							
##INFO=<ID=MIN_DP,Number=1,Type=Integer,Description="Minimum depth in gVCF output block">							
##INFO=<ID=END,Number=1,Type=Integer,Description="Last position (inclusive) in gVCF output record">							
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">							
##FORMAT=<ID=GQ,Number=1,Type=Float,Description="Genotype Quality, the Phred-scaled marginal (or unconditional) probability of the called genotype">							
##FORMAT=<ID=GL,Number=G,Type=Float,Description="Genotype Likelihood, log10-scaled likelihoods of the data given the called genotype for each possible genotype generated from the reference and alternate">							
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">							
##FORMAT=<ID=AD,Number=R,Type=Integer,Description="Number of observation for each allele">							
##FORMAT=<ID=RO,Number=1,Type=Integer,Description="Reference allele observation count">							
##FORMAT=<ID=QR,Number=1,Type=Integer,Description="Sum of quality of the reference observations">							
##FORMAT=<ID=AQ,Number=A,Type=Integer,Description="Alternate allele observation count">							
##FORMAT=<ID=QA,Number=A,Type=Integer,Description="Sum of quality of the alternate observations">							
##FORMAT=<ID=MIN_DP,Number=1,Type=Integer,Description="Minimum depth in gVCF output block">							
#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO
chr19	44915300	.	A	T	15.1967	.	AB=0.363636:ABP=4.78696:AC=1:AF=0.5:AN=2:AO=4:C
chr19	44915312	.	C	A	0.000306672	.	AB=0.142857:ABP=18.5208:AC=1:AF=0.5:AN=2:AO=2:C
chr19	44915314	.	TGGAA	CGGAG,TGGAG	141.964	.	AB=0.466667,0.466667:ABP=3.15506,3.15506:AC=1,1:A
chr19	44915354	.	C	T	161.116	.	AB=0.75:ABP=11.6962:AC=1:AF=0.5:AN=2:AO=12:CIGA
chr19	44915646	.	G	T	0.173493	.	AB=0.352941:ABP=6.20364:AC=1:AF=0.5:AN=2:AO=6:C
chr19	44915666	.	C	G	229.015	.	AB=0:ABP=0:AC=2:AF=1:AN=2:AO=21:CIGAR=1X:DP=2
chr19	44915690	.	G	T	4.59962e-05	.	AB=0.238095:ABP=15.5221:AC=1:AF=0.5:AN=2:AO=5:C
chr19	44915697	.	A	C	4.89218e-05	.	AB=0.25:ABP=13.8677:AC=1:AF=0.5:AN=2:AO=5:CIGAR
chr19	44915719	.	G	C	120.184	.	AB=0:ABP=0:AC=2:AF=1:AN=2:AO=11:CIGAR=1X:DP=1
chr19	44961436	.	AC	CT	54.3715	.	AB=0.45:ABP=3.44459:AC=1:AF=0.5:AN=2:AO=9:CIGAR
chr19	44961438	.	T	G	4.27902e-11	.	AB=0:ABP=0:AC=0:AF=0:AN=2:AO=2:CIGAR=1X:DP=22
chr19	44961442	.	G	A	2.92508e-10	.	AB=0:ABP=0:AC=0:AF=0:AN=2:AO=5:CIGAR=1X:DP=24
chr19	44961455	.	C	G,T	6.57561e-05	.	AB=0.407407,0.111111:ABP=5.02092,38.4777:AC=1,0:A
chr19	44961487	.	A	G	12.5913	.	AB=0.571429:ABP=4.25114:AC=1:AF=0.5:AN=2:AO=16:
chr19	44963214	.	G	A	1.81589e-06	.	AB=0:ABP=0:AC=0:AF=0:AN=2:AO=2:CIGAR=1X:DP=11

Fig 8. VCF annotation of Mexican American Male from NA19661 and FASTQ files
ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR034/SRR034567/SRR034567_1.fastq.gz
ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR034/SRR034567/SRR034567_2.fastq.gz Paired-end
files used on hg37 genome from UCSC Table browser in the region 44,900,000 -
45,000,000 (where the APOE gene and our SNP are).

#CHROM	POS	ID	REF	ALT
chr19	44900009	.	T	C
chr19	44900034	.	C	A
chr19	44900093	.	G	C
chr19	44900114	.	G	A
chr19	44901054	.	CGTAAAGT	TGTAAGT
chr19	44901526	.	A	G
chr19	44901557	.	A	G
chr19	44901737	.	C	A
chr19	44901760	.	G	A
chr19	44902820	.	C	T
chr19	44903050	.	T	A
chr19	44903160	.	A	T
chr19	44903428	.	A	G
chr19	44903488	.	A	G
chr19	44903514	.	A	G
chr19	44903571	.	A	T
chr19	44903896	.	T	C
chr19	44903916	.	G	A
chr19	44904897	.	C	T
chr19	44905440	.	A	G
chr19	44905779	.	C	T
chr19	44906051	.	G	A
chr19	44906110	.	T	C
chr19	44906139	.	CCCAT	CACCAT
chr19	44906472	.	A	G
chr19	44906795	.	A	T
chr19	44910204	.	G	A
chr19	44910259	.	C	T
chr19	44910264	.	A	G
chr19	44911414	.	C	T
chr19	44912202	.	T	C
chr19	44912270	.	G	C
chr19	44913231	.	C	G
chr19	44915229	.	T	C

Fig 9. VCF annotation of Mexican American Male from NA19685 and FASTQ files
ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR034/SRR034810/SRR034810_1.fastq.gz

ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR034/SRR034810/SRR034810_2.fastq.gz Paired-end files used on hg37 genome from UCSC Table browser in the region 44,900,000 - 45,000,000 (where the APOE gene and our SNP are).

Sample IDs	rs449647 (promoter)	rs405509 (promoter)	rs7412 (APOE4)	rs429358 (APOE4)
NA19798	No	No	No	No
NA19661	No	No	No	No
NA19685	No	No	No	No

Table 4: Table of samples and SNPs located.

All the genomes we searched did not have any of the SNPs we looked for, nor did they have our SNP, rs405509, which was the main subject of our paper [8]. This is not surprising since these allele frequencies are very small.

Discussion

Out of the three Mexican-American individuals that we studied through the SNP pipeline, none of them had any of the SNPs we were looking for. This result was expected as finding 4 specific SNPs in 3 random individuals (who may or may not have PD), was astronomically low even in an ethnicity that is more prone to PD. In fact, if there was an individual with one of the SNPs we studied, it would either be a false positive and we would still not be able to make any meaningful conclusions. Thus, our hypothesis that Mexican-Americans are more prone to have these SNPs due to their higher incidence of PD still remains unclear. While we did not find these 4 SNPs in these individuals, we cannot prove or disprove our hypothesis due to both sample size and access to genomic data from actual patients who have PD. If we actually had a reasonably

sized genomic sample of PD patients, we may be able to find a correlation that gives us a better idea of the prevalence of the SNPs in the PD patients. If anything, this study was meant to present a technique that allows for the study of specific SNPs in relation to their respective clinical conditions. The SNP pipeline that we used can be applied to any type of medical condition and research the SNP prevalence in accordance. Further analysis of these SNPs in Parkinson's patients is required.

Conclusion

We choose to investigate four SNPs located in the promoter of the APOE protein which were found to be highly correlated with Parkinson's Disease. We hypothesized that our SNPs of interest would be found in a larger percentage in the Mexican-American population as they are at a higher risk for PD. However we did not detect any of the SNPs in any of the individuals we tested. Admittedly our methods contained some limitations; the 1000 genomes database that was used in this experiment was limited in the information it provided; while ethnicity, gender and geographical location were provided, age and phenotype of the individual were not provided. Further, the sample sizes for a particular geographical location and ethnicity were not particularly large, in our case only six samples were available for testing of which we chose three. Obviously a larger database with phenotype data would be more useful in examining this type of statistical association for a Californian, Mexican-American population. Other databases such as Data collection, Public genetic data by Personal Genome Project and TCGA's cancer sequencing data portal provide this type of data and a program could be written to scan them for the alleles of interest and the associated phenotypes, an approach we had considered, but several of these databases need the type of permission usually associated with study conducted under the auspices of a grant: P.I., grant number, associated investigating institute, etc, which we did not have access to.

References

1. 1000 Genomes Project Consortium, Auton, A., Brooks, L. D., Durbin, R. M., Garrison, E. P., Kang, H. M., Korbel, J. O., Marchini, J. L., McCarthy, S., McVean, G. A., & Abecasis, G. R. (2015). A global reference for human genetic variation. *Nature*, 526(7571), 68–74. <https://doi.org/10.1038/nature15393>
2. Choi KY, Lee JJ, Gunasekaran TI, Kang S, Lee W, Jeong J, Lim HJ, Zhang X, Zhu C, Won S-Y, Choi YY, Seo EH, Lee SC, Gim J, Chung JY, Chong A, Byun MS, Seo S, Ko P-W, Han J-W, McLean C, Farrell J, Lunetta KL, Miyashita A, Hara N, Won S, Choi S-M, Ha J-M, Jeong JH, Kuwano R, Song MK, An SSA, Lee YM, Park KW, Lee H-W, Choi SH, Rhee S, Song WK, Lee JS, Mayeux R, Haines JL, Pericak-Vance MA, Choo IH, Nho K, Kim K-W, Lee DY, Kim S, Kim BC, Kim H, Jun GR, Schellenberg GD, Ikeuchi T, Farrer LA, Lee KH, Neuroimaging Initiative AD. APOE Promoter Polymorphism-219T/G is an Effect Modifier of the Influence of APOE ε4 on Alzheimer's Disease Risk in a Multiracial Sample. *Journal of Clinical Medicine*. 2019; 8(8):1236. <https://doi.org/10.3390/jcm8081236>
3. Enis Afgan, Dannon Baker, Bérénice Batut, Marius van den Beek, Dave Bouvier, Martin Čech, John Chilton, Dave Clements, Nate Coraor, Björn Grüning, Aysam Guerler, Jennifer Hillman-Jackson, Vahid Jalili, Helena Rasche, Nicola Soranzo, Jeremy Goecks, James Taylor, Anton Nekrutenko, and Daniel Blankenberg. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update, *Nucleic Acids Research*, Volume 46, Issue W1, 2 July 2018, Pages W537–W544, doi:10.1093/nar/gky379
4. ENSEMBL. Summary - homo_sapiens - ensembl genome browser 104. (n.d.). Retrieved October 4, 2021, from http://uswest.ensembl.org/Homo_sapiens/Transcript/Summary?db=core%3Bg.
5. Galaxy Enis Afgan, Dannon Baker, Bérénice Batut, Marius van den Beek, Dave Bouvier, Martin Čech, John Chilton, Dave Clements, Nate Coraor, Björn Grüning, Aysam Guerler, Jennifer Hillman-Jackson, Vahid Jalili, Helena Rasche, Nicola Soranzo, Jeremy Goecks, James Taylor, Anton Nekrutenko, and Daniel Blankenberg. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update, *Nucleic Acids Research*, Volume 46, Issue W1, 2 July 2018, Pages W537–W544, doi:10.1093/nar/gky379
6. https://en.wikipedia.org/wiki/Apolipoprotein_E
7. <https://www.jillcarnahan.com/2017/03/26/will-get-alzheimers-disease-apoe/> retrieved 10/03/2021
8. Huang, M., Wang, Y., Wang, L., Chen, B., Wang, X., & Hu, Y. (2020). APOE rs405509 polymorphism and Parkinson's disease risk in the Chinese population. *Neuroscience letters*, 736, 135256. <https://doi.org/10.1016/j.neulet.2020.135256>
9. James T. Robinson, Helga Thorvaldsdóttir, Douglass Turner, Jill P. Mesirov. igv.js: an embeddable JavaScript implementation of the Integrative Genomics Viewer (IGV). *bioRxiv* 2020.05.03075499.
10. Kok E, et al. Apolipoprotein E-dependent accumulation of Alzheimer disease-related lesions begins in middle age. *Ann Neurol*. 2009;65:650–657

- 11.** Liu, C. C., Liu, C. C., Kanekiyo, T., Xu, H., & Bu, G. (2013). Apolipoprotein E and Alzheimer disease: risk, mechanisms and therapy. *Nature reviews. Neurology*, 9(2), 106–118.
<https://doi.org/10.1038/nrneurol.2012.263>
- 12.** National Center for Biotechnology Information; [1988] – [cited 2021 Oct 02]. Available from: <https://www.ncbi.nlm.nih.gov/>
- 13.** Papaioannou, Ioannis & Simons, J & Owen, James. (2012). “Targeted In Situ Gene Correction of Dysfunctional APOE Alleles to Produce Atheroprotective Plasma ApoE3 Protein. *Cardiology research and practice*. 2012. 148796. 10.1155/2012/148796”.
- 14.** Phillips MC (September 2014). "Apolipoprotein E isoforms and lipoprotein metabolism". *IUBMB Life*. 66 (9): 616–23. doi:10.1002/iub.1314. PMID 25328986. S2CID 6159310.
- 15.** SNPedia. <https://www.snpedia.com/index.php/APOE#:~:text=Variations%20in%20ApoE%20are%20also%20associated%20with%20altered,these%20genes%20are%20called%20ApoE2%2C%20ApoE3%2C%20and%20ApoE4>. Retrieved 11/1/2021.
- 16.** Solovieff, N., Hartley, S. W., Baldwin, C. T., Perls, T. T., Steinberg, M. H., & Sebastiani, P. (2010). Clustering by genetic ancestry using genome-wide SNP data. *BMC genetics*, 11, 108.
<https://doi.org/10.1186/1471-2156-11-108>
- 17.** UCSC Table Browser: Karolchik D, Hinrichs AS, Furey TS, Roskin KM, Sugnet CW, Haussler D, Kent WJ. The UCSC Table Browser data retrieval tool. *Nucleic Acids Res*. 2004 Jan 1;32(Database issue):D493-6.
- 18.** UniProt ConsortiumEuropean Bioinformatics InstituteProtein Information ResourceSIB Swiss Institute of Bioinformatics. (2021, June 2). Apolipoprotein E. UniProt
- 19.** Wright Willis A, Evanoff BA, Lian M, Criswell SR, Racette BA. Geographic and ethnic variation in Parkinson disease: a population-based study of US Medicare beneficiaries. *Neuroepidemiology*. 2010;34(3):143-151. doi:10.1159/000275491
- 20.** Zhao, N., Liu, C. C., Qiao, W., & Bu, G. (2018). Apolipoprotein E, Receptors, and Modulation of Alzheimer's Disease. *Biological psychiatry*, 83(4), 347–357.
<https://doi.org/10.1016/j.biopsych.2017.03.003>
- 21.** Yamazaki, Y., Zhao, N., Caulfield, T. R., Liu, C. C., & Bu, G. (2019). Apolipoprotein E and Alzheimer disease: pathobiology and targeting strategies. *Nature reviews. Neurology*, 15(9), 501–518.
<https://doi.org/10.1038/s41582-019-0228-7>
- 22.** Abondio, P.; Sazzini, M.; Garagnani, P.; Boattini, A.; Monti, D.; Franceschi, C.; Luiselli, D.; Giuliani, C. The Genetic Variability of APOE in Different Human Populations and Its Implications for Longevity. *Genes* 2019, 10, 222. <https://doi.org/10.3390/genes10030222>
- 23.** Aceves, D., Ruiz, B., Nuño, P., Roman, S., Zepeda, E., & Panduro, A. (2006). Heterogeneity of apolipoprotein E polymorphism in different Mexican populations. *Human biology*, 78(1), 65–75.
<https://doi.org/10.1353/hub.2006.0021>

24. Alzhemiers Genes are You at Risk? <https://www.mayoclinic.org/diseases-conditions/alzheimers-disease/in-depth/alzheimers-genes/art-20046552>. Retrieved 12/20/2021

25. Pal, P., Sadhukhan, T., Chakraborty, S. *et al.* Role of Apolipoprotein E, Cathepsin D, and Brain-Derived Neurotrophic Factor in Parkinson's Disease: A Study from Eastern India. *Neuromol Med* **21**, 287–294 (2019). <https://doi.org/10.1007/s12017-019-08548-4>.

Computational Programs and Databases

SQL

ProteinID	Product	Gene	Start	Stop
ACB01207.1	Fused aspartokinase I and homoserine dehydrogenase I	thrA	337	2799
ACB01208.1	Homoserine kinase	thrB	2801	3733
ACB01209.1	Threonine synthase	thrC	3734	5020
ACB01224.1	Sodium-proton antiporter	nhaA	17489	18655
ACB01283.1	Dephospho-CoA kinase	coaE	86703	87323

These genes were taken from the E. coli K12 annotation on Genbank here:

<http://www.ncbi.nlm.nih.gov/nuccore/CP000948.1>

With the sample database of genes, I can use SQL in the terminal to search for:

Just the thrB gene:

```
SELECT Gene FROM genes WHERE Product=Homoserine Kinase
```

All three genes which starts with 'thr'

```
SELECT Gene FROM genes WHERE Gene='thr%'
```

All genes whose start site falls between base 2500 and 5000.

```
SELECT Gene FROM genes WHERE Start BETWEEN 2500 AND 5000
```

I can create the entire gene table above from scratch with the code:

```
CREATE TABLE genes (Protein_ID VARCHAR(12) NOT NULL,Product VARCHAR(255) NOT NULL,  
Gene VARCHAR(10) NOT NULL, Start Int(8) NOT NULL, Stop Int(8) NOT NULL);
```

I can also add data to the table by using the insert function to add the second row of the table

INSERT INTO genes (ProteinID, Product, Gene, Start, Stop)

VALUES (ACB01208.1, Homoserine kinase, thrB, 2801, 3733)

Python

The scripts I've written are all also on Github but I want to showcase some of my scripts here so that you can get a better understanding of the types of programs I can write, and my progression as a programmer.

Circle "Program"

```
Python01 > ex01_circle.py > ...
1  #!/usr/bin/python3
2  radius=7.5
3  pi=3.14159
4  circumference=2*pi*radius
5  print("Circumference:",circumference)
6  Circumference: 47.12385
7
```

First program (can you even call it that if it is just a math equation) I've ever written!

Everyone must start somewhere I guess, and this was mine. Input your radius and I can get it to output the circumference. But you will have to manually go into my code and change the values to get an output, lame. But with no knowledge of definitions/functions and inputs, there was a lot to improve on.

DNA Length and Start Codon Program

```
Python01 > ex04_dna.py > ...
1  #!/usr/bin/python3
2  dna = 'ATGGAACCAACGTCAGTGACTTCGTCAG'
3  print("The length of your sequence is=",len(dna))
4  print("The first codon is=", dna[0:3])
```

Output: The length of your sequence is= 28 The first codon is= ATG

Learned how to use the len function as well as how to print elements of a string. This allowed me to write a program that can count how long the input DNA sequences are as well as the first codon.

Motif Counter

```
Python01 > 📄 motifinstances.py > ...
1  #!/usr/bin/python3
2  from collections import Counter
3
4  def motifinstances(motif,dna):
5      # motif="AA"
6      #dna = 'ATGGAACCAACGTCAGTGACTTCGTCAG'
7      motif_count=dna.count(motif)
8      print("There are "+ str(motif_count)+" instances of the motif " + "\"" + motif + "\"")
9
10 motifinstances(motif='T', dna='ATGGAACCAACGTCAGTGACTTCGTCAG')
```

Output: There are 6 instances of the motif 'T'

First real function I've written that takes a motif input and a DNA string and counts how many instances of that motif are in the string. Still no input on the command line yet but I can write functions now

FASTA Sequence Counter

```
Python02 > 📄 ex05_seqfunc.py > ...
1  #!/usr/bin/python3
2  import sys
3  def fasta_sequence_count(fasta):
4      count=0
5      for line in open(fasta):
6          if line.startswith(">"):
7              count+=1
8      print("There were "+ str(count) + " sequences within the file." )
9
10 filename=sys.argv[1]
11 fasta_sequence_count(filename)
```

Output: There were 4126 sequences within the file.

My programs are getting more practical now and this one allows you to enter the path to a FASTA file as an argument and return the number of sequence entries within it. Very basic, but very useful program if you have many FASTA files to analyze. This code can also be modified for other filetypes as well.

Parsing Program

```
Python03 > ex04_adjectives.py > words
1  #!/usr/bin/python3
2  import re
3  regex = "(\w*)(,)(\s)(\w*)"
4  input = "The quick, brown fox jumped over some other non-descript animal"
5  search_sentence = re.search(regex, input)
6  words = search_sentence.group(4) + search_sentence.group(2) + search_sentence.group(3) + search_sentence.group(1)
7  output = input.replace(search_sentence.group(0), words)
8  print(output)
9
```

Output: The brown, quick fox jumped over some other non-descript animal

My first introduction into the power of Regex and parsing. This program allows you to input a sentence with two adjectives separated by a comma and reverses them. Parsing with regex allows me to utilize pattern recognition in text ranging from simple sentences to complex genomic data files.

Genome Analysis Program

```
Python04 > ex02_aaseqpercent.py > ...
1  #!/usr/bin/python3
2  f=open("/home/jorvis1/e_coli_k12_dh10b.faa")
3  from collections import defaultdict
4  counts = defaultdict(int)
5  for line in f:
6      if line.startswith(">"):
7          continue
8      for char in line:
9          if char in {"A", "C", "D", "E", "F", "G", "H", "I", "K", "L", "M", "N", "P", "Q", "R", "S", "T", "V", "W", "Y"}:
10             counts[char]+=1
11 total=float(sum(counts.values()))
12 sorter=0
13 for key,val in sorted(counts.items(),key=lambda kv:(kv[1]),reverse=True):
14     sorter=sorter+1
15     if sorter>=6:
16         break
17     print("{}:{}".format(key,str(val)+" ",(str(val/total*100)+"%")))
```

Output: L: 139002 (10.7%)

A: 123885 (9.6%)

G: 95475 (7.4%)

V: 91683 (7.1%)

I: 77836 (6.0%)

A more complex program used to generate an amino acid usage report from one FASTA files with multiple polypeptide sequences. Through this project I learned how to use dictionaries to store and manipulate data.

Angle Converter

```
Exam_02 > convert_angle.py > ...
1  #!/usr/bin/python3
2  def convert_angle(units=None,source_angle=None):
3  # main definition/function
4      if units=='D':
5          #if statement for if starting unit is in degrees
6
7          anglerad=(source_angle/180)
8          #math function of degrees to radians
9
10         print('\n%.1f degrees is %.1f radians' % (source_angle, anglerad))
11         #print function of conversion to radians
12
13     if units=='R':
14         #if statement for if starting unit is in radians
15
16         angleddeg=(180/source_angle)
17         #math function of radians to degrees
18
19         print('\n%.1f radians is %.1f degrees' % (source_angle, angleddeg))
20         #print function of conversion to degrees
21
22     units=input('Enter starting units(D or R):')
23     #allows user input of D or R for degrees or radians
24
25     source_angle=float(input('Enter starting angle: '))
26     #allows user input of starting angle
27
28     convert_angle(units,source_angle)
29     #final function with units and angle
30
```

Output: Enter starting units(D or R):D

Enter starting angle: 120

120.0 degrees is 0.7 radians

Building upon the simple mathematic function between degrees and radians, I wrote a converter that allows users to input their desired units and angle and the program will spit out the corresponding angle. I also began annotating my code to elucidate what I am doing.

DNA Reverse Complement Program

```
Exam_02 > DNA_reversecomp.py > ...
1  #!/usr/bin/python3
2  import collections
3  import re
4  seq="TATGAGCCCGTA"
5  nucleotides={"A","C","G","T"}
6  complement={'A': 'T', 'C': 'G', 'G': 'C', 'T': 'A'}
7  # dictionaries with nucleotides and their complements
8
9  def reverse_complement(seq):
10     complist=[]
11     for base in seq:
12         if base not in complement:
13             print("invalid character found")
14             return False
15         else:
16             complist.append(complement[base])
17 #main definition of the reverse complement, runs for loop to check if sequence is valid
18
19     return''.join(complist[::-1])
20     #reverses string after replacing the nucleotides
21 valid =True
22 while valid==True:
23     seq=input('Enter nucleotide sequence:')
24     if reverse_complement(seq)==False:
25         valid=False
26     else:
27         print(reverse_complement(seq))
28 #Block that runs the program again if the sequence is valid, and terminates if there is an invalid character
29
```

Output: Enter nucleotide sequence:ATGCAGTACGATTTC

GAAATCGTACTGCAT

Enter nucleotide sequence:AATGAGGADTCWD

invalid character found

My next project is a reverse complementary base program that gives you the reverse complement of a user inputted string of DNA. Here I learned that you could add user-friendly, fidelity checks to your code as well as utilizing multiple dictionaries and logic statements.

GFF Feature Exporter

```
Final > export_gff3_feature.py > ...
1  #!/usr/bin/env python3
2  import argparse
3  import re
4
5  # INPUT$ export_gff3_feature.py --source_gff=/path/to/some.gff3 --type=gene --attribute=ID --value=YAR003W
6  # OUTPUT # >gene:ID:YAR003W... sequence here ...
7
8  # file = "C:\Users\LOCALADMIN\AppData\Local\Programs\Python\Python310\Scripts\Final\gff_file.gff"
9
10 parser = argparse.ArgumentParser(description='gff3_feature_exporter')
11 parser.add_argument('-s', '--source_gff', required=True)
12 parser.add_argument('-t', '--type', required=True)
13 parser.add_argument('-a', '--attribute', required=True)
14 parser.add_argument('-v', '--value', required=True)
15 args = vars(parser.parse_args())
16
17 #using argparse to add arguments for user input in the shell
18
19 source = args["source_gff"]
20 att_type = args["type"]
21 att_id = args["attribute"]
22 target_value = args["value"]
23
24 #variables that correspond to the arguments used below
25
26 gene_IDs = []
27 gene_atts = {}
28
29 cur_sequence = ""
30 cur_chromosome = ""
31 full_sequences = {}
32 result_sequence = ""
33
34 #variables used in the main parsing code
35
36 with open(
37     source,
38     'r') as f:
39     lines = f.readlines()
40     for line in lines:
41         line = line.rstrip()
42         if line.startswith("#"):
43             continue
44         if line.startswith(">"):
45             full_sequences[cur_chromosome] = cur_sequence
46             cur_chromosome = line[1:]
47             cur_sequence = ""
48             continue
49         column = line.split("\t")
50
51 #processing file by splitting and telling the code how to read the file,
```

```

52  #stores FASTA sequences as a dictionary called full_sequences
53
54  if len(column) > 1:
55      element = column[2]
56      attribute = column[8]
57      temp = re.match(fr"({att_id}\=[^;]*)", attribute)
58
59      if temp:
60          gene_ID = temp.group(2)
61          if (gene_ID == target_value):
62              gene_IDs.append(temp.group(2))
63
64          gene_atts[gene_ID] = {
65              "chromosome": column[0],
66              "source": column[1],
67              "feature": column[2],
68              "start": int(column[3])-1,
69              "end": int(column[4]),
70              "score": column[5],
71              "strand": column[6],
72              "phase": column[7],
73              "attributes": column[8]
74          }
75
76      else:
77          cur_sequence = cur_sequence + line
78
79  #First checks to see if column is >1, if it is then the code will create a dictionary for the 8 gff3 columns
80
81  if len(gene_IDs) == 0:
82      print("There are no sequences for the given features!")
83  elif len(gene_IDs) > 1:
84      print("There are more than one sequences for the given features!")
85  else:
86      target_gene = gene_atts[gene_IDs[0]]
87
88      if target_gene["strand"] == "+":
89          result_sequence = full_sequences[target_gene["chromosome"]][target_gene["start"]:target_gene["end"]]
90      else:
91          result_sequence = full_sequences[target_gene["chromosome"]][target_gene["end"]:target_gene["start"]:-1]
92      print(">" + att_type + ":" + att_id + ":" + target_value)
93      print(result_sequence)
94
95  #Error messages for if no value (geneID) is found or if there are more than 1
96  #otherwise it searches for FASTA sequences for the chromosome, start, end, and strand and prints the DNA sequence

```

The input and output of the code is a comment at the top of the program

My programming magnum opus (as of now) is a GFF3 feature exporter. This is the longest code I've ever written that ties everything I've learned from the previous programs. From the command line, users can enter a specific gene from the GFF3 file format for any organism and the parser will match the gene with its corresponding nucleotide sequence via the FASTA sequences located at the bottom of the GFF3 file.

Acknowledgements

First, I want to thank my Tools for Genome Analysis Professors Sijung Yun & Sajung Yun for the time and effort spent on making sure that us students are prepared for the bioinformatics field with both their coursework and professional guidance. This showcase would not be possible without them. I also want to thank Professor Jarrett Morrow for the guidance in learning how to use Unix, SQL, and write code in Python.

Next, I appreciate the support of my peers who have answered any questions no matter how simple. Group projects and discussions allowed me to expand my scope of knowledge and provide me with different perspectives.

Furthermore, I am grateful for the resources provided by Johns Hopkins University such as the bioinformatics server when we can practice Unix and SLQ and test programs. The counselor guidance and seminars have been very useful.

Lastly, I want to thank the reader for taking the time out of their day to read this portfolio up to this point. I hope that you now have a better understanding of the content I have been working on and knowledge of the skills that I have developed. If you have any questions or concerns about my resume or my portfolio, please feel free to contact me by phone (408-908-0395) or email (mcheng30@jh.edu) as I would be happy to answer them.