Due 9/4 at 6pm

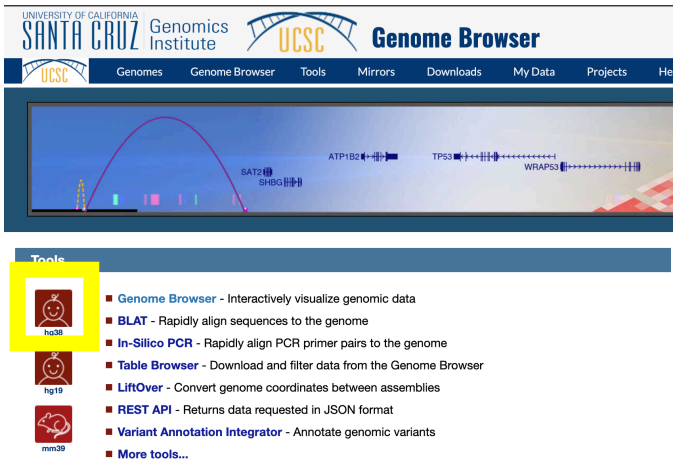
1. You've got some DNA back from the sequencer and you want to see where it maps in the reference genome. Use the code at the bottom to find out where your sequence goes and how long it takes to find it!

   Complete the code by selecting the DNA to search for based on your last name:
   Group 1: A – E: "TATGGAGGTGAAAGACGTGAAGACT"
   Group 2: F – J: "GGGAACCTGTACGAGTCATGCGAGA"
   Group 3: K – O: "CCCGGTGGGGAGGGCCCTCTTGGCC"
   Group 4: P – T: "TTGATCTGTCGTACGTGCTTCGACA"
   Group 5: U – Z: "TTGATCTGTCGTACGTGCTTCGACA"

   What is the coordinate of the best match start?
   What is the mapping %?

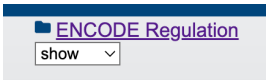
2. It didn't take too long to search! But what if you had 1 billion reads? Calculate based on your prior time how long it would take (in hours!) for that program to run 1 billion times.

3. Head to https://genome.ucsc.edu/ and open up the genome browser for genome build hg38:



   Navigate to genomic position: chr17:74,415,501-74,425,500

   Scroll down to examine the various options and Click 'ENCODE Regulation' and set TF ChIP to 'full'



   What transcription factor ChIP-seq peaks are bound in this region?
   - List in Markdown

- Hint: Looks for the tracks with the boxes!

4. You got the amino acid sequence of a yeast protein from some 'omics experiment. Your PI wants to know what the human equivalent (Assembly GRCh38/hg38) of the gene is. Head over to UCSC's BLAT search (https://genome.ucsc.edu/cgi-bin/hgBlat) and see if you're feeling lucky.
"MSGRGKGGKGLGKGGAKRHRKILRDNIQGITKPAIRRLARRGGVKRISGLIYEEVRAVL
KSFLESVIRDSVTYTEHAKRK TVTSLDVVYALKRQGRTLYGFGG"

   Tell us what the NCBI refSeq gene name is of the homologous gene.
   - List in Markdown

5. You discovered a new protein (maybe)! Use NCBI's BLAST-protein to figure out the most likely identity of the protein and the list of conserved domain hits (Look for the graphical interface):
MSKNQSVSASEDEKEILNNNAEGHKPQRLFDQEPDLTEEALTKFENLDDCIYANKRIGTF
KNNDFMECDCYEEFSDGVNHACDEDSDCINRLTLIECVNDLCSSCGNDCQNQRFQKKQYA
PIAIFKTKHKGYGVRAEQDIEANQFIYEYKGEVIEEMEFRDRLIDYDQRHFKHFYFMMLQ
NGEFIDATIKGSLARFCNHSCSPNAYVNKWVVKDKLRMGIFAQRKILKGEEITFDYNVDR
YGAQAQKCYCEEPNCIGFLGGKTQTDAASLLPQNIADALGVTVSMEKKWLKLKKLSGEPI
IKNENENINIEFLQSLEVQPIDSPVDVTKIMSVLLQQDNKIIASKLLKRLFTIDDDSLRH
QAIKLHGYTCFSKMLKLFITEQPQVDGKGNETEEDDIKFIKGILDFLLELPKTTRNGIES
SQIDNVVKTLPAKFPFLKPNCDELLEKWSKFETYKRITKKDINVAASKMIDLRRVRLPPG
WEIIHENGRPLYYNAEQKTKLHYPPSGSSKVFSSRSNTQVNSPSSSGIPKTPGALDSKKH
KLSDEEYERKKQKRLEYERIALERAKQEELESLKQKLKLENERKSVLEDIIAEANKQKEL
QKEEAKKLVEAKEAKRLKRKTVSQSQRLEHNWNKFFASFVPNLIKKNPQSKQFDHENIKQ
CAKDIVKILTTKELKKDSSRAPPDDLTKGKRHKVKEFINSYMDKIILKKKQKKALALSSA
STRMSSPPPSTSS

   Note: BLAST and BLAT are separate algorithms

```
import time

def simple_dna_alignment(query_sequence, reference_sequence):
    alignment = ""
    match_count = 0

    for query_base, reference_base in zip(query_sequence, reference_sequence):
        if query_base == reference_base:
            alignment += "|"
            match_count += 1
        else:
            alignment += " "

    alignment += "\n" + query_sequence + "\n" + reference_sequence
    return alignment, match_count / len(reference_sequence) * 100

def find_best_match(query, reference):
    best_match_start = 0
    best_match_score = 0

    for i in range(len(reference) - len(query) + 1):
        match_score = sum(1 for q, r in zip(query, reference[i:i+len(query)]) if q == r)

        if match_score > best_match_score:
            best_match_score = match_score
            best_match_start = i

    return best_match_start, best_match_score

# Example DNA sequences
reference_genome =
"ATGCCTTCAGGTCATAACGATAAAAACGCAAATCAAGAGTCTGTGGAAGAGGCTGTTTTGAAATATGTCGGTGTAGGTCTAGATCACCAGAACCAT
GACCCTCAATTGCATACGAAGGATCTGGAAAACAAACACTCGAAAAAGCAGAATATTGTGGAAAGTAGTAGTGATGTTGATGTTAATAACAATGATG
ACAGTAATAGAAACGAGGATAATAATGATGATTCTGAAAATATTAGCGCATTGAATGCGAACGAATCATCTTCGAATGTGGATCATGCCAACTCCAA
TGAACAACATAACGCAGTTATGGATTGGTATTTAAGGCAAACAGCGCATAATCAACAAGACGACGAAGATGACGAAAACAATAATAACACTGACAAC
GGCAATGACAGTAATAATCACTTTTCTCAATCTGACATAGTCGTGGATGACGACGACGACAAGAATAAGAAAGATGCGGGTGTTGGTGTGGACGATG
ATCATCAATCTATGGCGATGGCCGCCGTCGCTGCTGCTTACACTCTATCGAAAAATAACAATAATAATAACAGTATTGCGAACGATAGCAATTCGCG
TAAAAGACAGCATGATAACGGAAATAATCATGAGAATTCACAGAAAAAGCGGAAAAACAATAATGATGACGATGACAGACAAATTGGAAACGTAGAT
CCGGAATTAACCACCTTGGGTGATGCAGATGACAACGATACTAATAATGATGTGATTGATCGTGATCAACTGGTCCATAAGGCCATTATCGACGCTG
ATTCGATTACCCAACATCCTGATTTCCAGCAATACTTGAATACTGCGGCTGATACTGACGATAACGAAAAATTAAAGCATATTAAAGATCATTTGAT
GCGTACACACGGTTTGAATCATCAGAACAAGAATCACAATGATGATACAGATGATTTATCAAATAGTACAAAGCAATACTCTGAACTGCAGAAAGAC
TCTATGCTGGATAGTTCCCTCAACAAATCTAGAAATTATATGGAAGTCTTGCCCAAAGTTATCTCTCAAGATACTCAACCACACCAACAAAAATCTC
CCTCTCATGATAATGAAGCTGGCAGCGTGGACAATTCAGAAATATCTCAACTACTTCAATCAGCCGCCACAAAGGCATCATCTTTGGTATCTTTATC
CTCATCCTCAGCAACGCCATCGACTTCAAGGTCTAACAATAGCAAAGCTTTTGATAAAGCCGAAGACGCCGCTTTAGAAAGATTTATTAACGAGTAT
GAGGCCATTGAACGTTTGACTAGACAACAAGTTTGTGAAAGGATATGGAGTTCCGACAGGCCAAAGGACAACTTTTGGAATAATATTTACAAAGTCT
TACCCTATAGATCTAGCTCCTCTATCTACAAACACATGAGAAGAAAATATCACATTTTTGAACAACGTGGTAAATGGACCGCGGAGGAGGAACAAGA
GCTAGCTAAATTATGTGCAGAAAAAGAAGGTCAATGGGCAGAAATAGGTAAAACTTTAGGCAGAATGCCAGAAGATTGTAGGGATCGTTGGAGAAAC
TACGTAAAATGTGGTACCAATAGAGCATCAAATAGATGGTCCGTTGAAGAAGAAGAGCTTCTGAAAAAGGTTATCAGCGATATGTTGGAAGAAGCCC
AGCAACAACAGTCTCAATTGCATCCAAACTTATTGGAAGAAGAACAGCATTTACTGCAAGATGACCAGAATGATCATCGCAATAACGATGAAGACGA
TGATGATACAGCTTCTGCAGCAGCAGCTGCTGCTGCTGCTATTCAAGAACAACAACAACTTCTTCAACAAAAGCAGCAAGATGATGACGATGCTATT
GCCGCTGCTGCCGCTGCTGCTTCTTCATCCCTGGGAGACAACAAAGACGAAGACAAACCCCCACGATTCATTAGGTATACAGCTCGATGATAATTCCC
AGAACTCAATGGTACCTGCTCCATCAGCAACAAGCACACATTCTAAAAGTTTGTCAAATACAATCAGACGTCACAATAATAAACTGAGGAAATCTTT
GATGGGTAACGGTAAGTTAGATTTCAAAGACATCATTAACTGGACCATTGTCAGTGAGCGTATGGGCGGTACGAGATCACGTATTCAATGTCGTTAT
AAATGGAATAAATTGGTCAAAAGGGAAGCAATTGCCAAAATTCAGACTGTAAAAGATGATGATATGTTATGGATTTTTGAAAAATTAAGAGATTTAG
GTATAACAGAAGATTCTCAAGTAGATTGGGATGAACTTGCAGCTTTGAAGCCTGGCATGAAATTAAATGGGTTAGAGTTGAAATTGTGCTATGAAAG
AATGAAGAAAAAGGTCAAAGGCTATAAGCAAAAATCAATCAATGAAATCAGTAAAGAGCTAGTTGATTATTTTAGCTCCAATATTTCAATGAAAACA
GAAAATTAA"

query_sequence = "TATGGAGGTGAAAGACGTGAAGACT"

start_time = time.time()
best_match_start, best_match_score = find_best_match(query_sequence, reference_genome)
end_time = time.time()

alignment_result, match_percentage = simple_dna_alignment(query_sequence,
reference_genome[best_match_start:best_match_start+len(query_sequence)])

print("\nBest Match Start:", best_match_start)
print("Best Match Score:", best_match_score)
print("\nAlignment:")
print(alignment_result)
```

```
print("Match Percentage:", match_percentage, "%")
print("Time taken:", end_time - start_time, "seconds")
```