**# Install packages**

Pip install Bio python

Pip install wget

**# Load the package**

import Bio python

import wget

**# Replace this URL with the actual file URL you want to download**

url = ‘https://ftp.ncbi.nlm.nih.gov/genomes/Viruses/Vieuvirus.fn'

**# Replace this with the desired destination path for the downloaded file.**

destination\_path = 'Vieuvirus.fn'

**# Download the file from the URL to the specified destination path**

wget.download(url, destination\_path)

print("Download completed!")

**# create the bed file with fasta file**

from Bio import SeqIO

fasta\_file='/content/Vieuvirus.fn'

def fasta\_to\_bed(fasta\_file, bed\_file):

with open(bed\_file, 'w') as bed:

for record in SeqIO.parse(fasta\_file, 'fasta'):

chromosome = [record.id](http://record.id/)

sequence\_length = len(record.seq)

start\_position = 0

end\_position = sequence\_length

bed.write(f"{chromosome}\t{start\_position}\t{end\_position}\n")

**# Replace 'input.fasta' with the path to your FASTA file**

**# Replace 'output.bed' with the desired output BED file path**

fasta\_to\_bed('/content/Vieuvirus.fn', 'sample.bed')

**# load packages**

import random

from Bio import SeqIO

from Bio.Seq import Seq

def extract\_subsequences(fasta\_file, bed\_file):

**"""**

**Extract subsequences from a given primary multi-fasta file based on BED coordinates.**

**Parameters:**

**fasta\_file (str): Path to the primary multi-fasta file.**

**bed\_file (str): Path to the BED file containing the coordinates for extraction.**

**Returns:**

**dict: A dictionary containing the extracted subsequences, where keys are scaffold names**

**and values are corresponding sequences.**

**"""**

sequences = SeqIO.to\_dict(SeqIO.parse(fasta\_file, "fasta"))

extracted\_subsequences = {}

with open(bed\_file, "r") as bed:

for line in bed:

scaffold\_name, start\_pos, end\_pos = line.strip().split()

start\_pos, end\_pos = int(start\_pos), int(end\_pos)

extracted\_subsequences[scaffold\_name] = sequences[scaffold\_name][start\_pos-1:end\_pos]

return extracted\_subsequences

def reverse\_complement\_sequences(extracted\_subsequences):

**"""**

**Reverse complement the extracted subsequences.**

**Parameters:**

**extracted\_subsequences (dict): A dictionary of extracted subsequences.**

**Returns:**

**dict: A dictionary containing the reverse complemented subsequences,**

**where keys are scaffold names and values are corresponding reverse complemented sequences.**

**"""**

reversed\_sequences = {}

for scaffold\_name, sequence in extracted\_subsequences.items():

reversed\_sequences[scaffold\_name] = sequence.reverse\_complement()

return reversed\_sequences

def introduce\_random\_mutation(reversed\_sequences):

**"""**

**Introduce random 1 nucleotide change in the reversed sequences.**

**Parameters:**

**reversed\_sequences (dict): A dictionary of reverse complemented sequences.**

**Returns:**

**dict: A dictionary containing the sequences with random one-nucleotide changes,**

**where keys are scaffold names and values are corresponding sequences with mutations.**

**"""**

mutated\_sequences = {}

for scaffold\_name, sequence in reversed\_sequences.items():

random\_pos = random.randint(0, len(sequence) - 1)

original\_nucleotide = sequence[random\_pos]

mutated\_nucleotide = random.choice("ACGT".replace(original\_nucleotide, ""))

mutated\_sequence = sequence[:random\_pos] + Seq(mutated\_nucleotide) + sequence[random\_pos + 1:]

mutated\_sequences[scaffold\_name] = mutated\_sequence

return mutated\_sequences

def insert\_processed\_subsequences(fasta\_file, mutated\_sequences, output\_file):

**"""**

**Insert the processed subsequences back into the primary sequence (multifasta).**

**Parameters:**

**fasta\_file (str): Path to the primary multi-fasta file.**

**mutated\_sequences (dict): A dictionary of sequences with random one-nucleotide changes.**

**output\_file (str): Path to the output processed multi-fasta file.**

**Returns:**

**None: The processed multi-fasta file is written to the output\_file path.**

**"""**

with open(output\_file, 'w') as output\_fasta:

for record in SeqIO.parse(fasta\_file, "fasta"):

scaffold\_name = [record.id](http://record.id/)

if scaffold\_name in mutated\_sequences:

record.seq = mutated\_sequences[scaffold\_name]

SeqIO.write(record.seq, output\_fasta, "fasta")

**# Replace 'Vieuvirus.fn' with the path to your primary multi-fasta file**

fasta\_file = "Vieuvirus.fn"

**# Sample BED file with at least 3 scaffolds and coordinates**

bed\_file = "sample.bed"

**# Output file for the processed multi-fasta**

output\_file = "processed\_" + fasta\_file

**# Step 1: Extract subsequences based on BED coordinates**

extracted\_subsequences = extract\_subsequences(fasta\_file, bed\_file)

**# Step 2: Reverse complement the extracted subsequences**

reversed\_sequences = reverse\_complement\_sequences(extracted\_subsequences)

**# Step 3: Introduce random nucleotide changes in the reversed sequences**

mutated\_sequences = introduce\_random\_mutation(reversed\_sequences)

**# Step 4: Insert the processed subsequences back into the primary sequence (multifasta)**

insert\_processed\_subsequences(fasta\_file, mutated\_sequences, output\_file)

**# scaffolds (specified intervals like a chromosomes number and position of chromosome )**

output\_file = "my\_bed\_file.bed"

with open(output\_file, "w") as bed\_file:

for interval in bed\_data:

bed\_file.write("\t".join(map(str, interval)) + /n)

**# replace /n inputfile of sample.bed file**