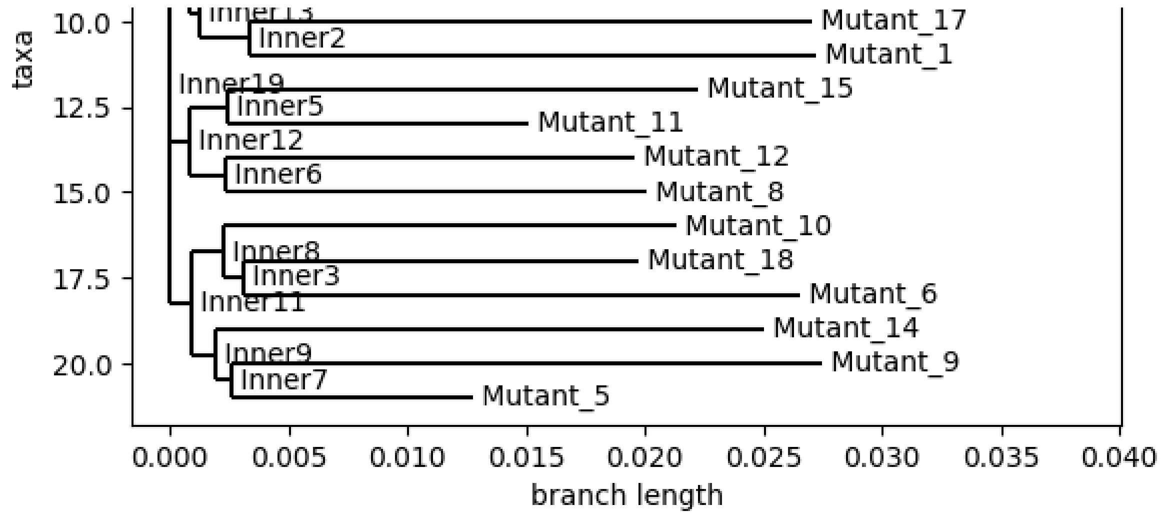
```
!pip install biopython
%pip install Bio ete3
!pip install cus
!apt-get install clustalw
```

```
Collecting ete3
  Downloading ete3-3.1.3.tar.gz (4.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.8/4.8 MB 15.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: biopython>=1.80 in /usr/local/lib/python3.10/dist-packages (from Bio) (1.84)
Collecting gprofiler-official (from Bio)
  Downloading gprofiler_official-1.0.0-py3-none-any.whl.metadata (11 kB)
Collecting mygene (from Bio)
  Downloading mygene-3.2.2-py2.py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from Bio) (2.1.4)
Requirement already satisfied: pooch in /usr/local/lib/python3.10/dist-packages (from Bio) (1.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from Bio) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from Bio) (4.66.4)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from biopython>=1.80->Bio) (1.26.4)
Collecting biothings-client>=0.2.6 (from mygene->Bio)
  Downloading biothings_client-0.3.1-py2.py3-none-any.whl.metadata (9.8 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas->Bio)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->Bio) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas->Bio) (2024.1)
Requirement already satisfied: platformdirs>=2.5.0 in /usr/local/lib/python3.10/dist-packages (from pooch->Bio) (4.2
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from pooch->Bio) (24.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->B
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->Bio) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->Bio) (2
  Stored in directory: /root/.cache/pip/wheels/a0/72/00/1982bd848e52b03079dbf800900120bc1c20e92e9a1216e525
Successfully built ete3
Installing collected packages: ete3, gprofiler-official, biothings-client, mygene, Bio
Successfully installed Bio-1.7.1 biothings-client-0.3.1 ete3-3.1.3 gprofiler-official-1.0.0 mygene-3.2.2
ERROR: Could not find a version that satisfies the requirement cus (from versions: none)
ERROR: No matching distribution found for cus
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  clustalx seaview
The following NEW packages will be installed:
  clustalw
0 upgraded, 1 newly installed, 0 to remove and 45 not upgraded.
Need to get 275 kB of archives.
After this operation, 818 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 clustalw amd64 2.1+lgpl-7 [275 kB]
Fetched 275 kB in 1s (310 kB/s)
Selecting previously unselected package clustalw.
(Reading database ... 123589 files and directories currently installed.)
Preparing to unpack .../clustalw_2.1+lgpl-7_amd64.deb ...
Unpacking clustalw (2.1+lgpl-7) ...
Setting up clustalw (2.1+lgpl-7) ...
Processing triggers for man-db (2.10.2-1) ...
```

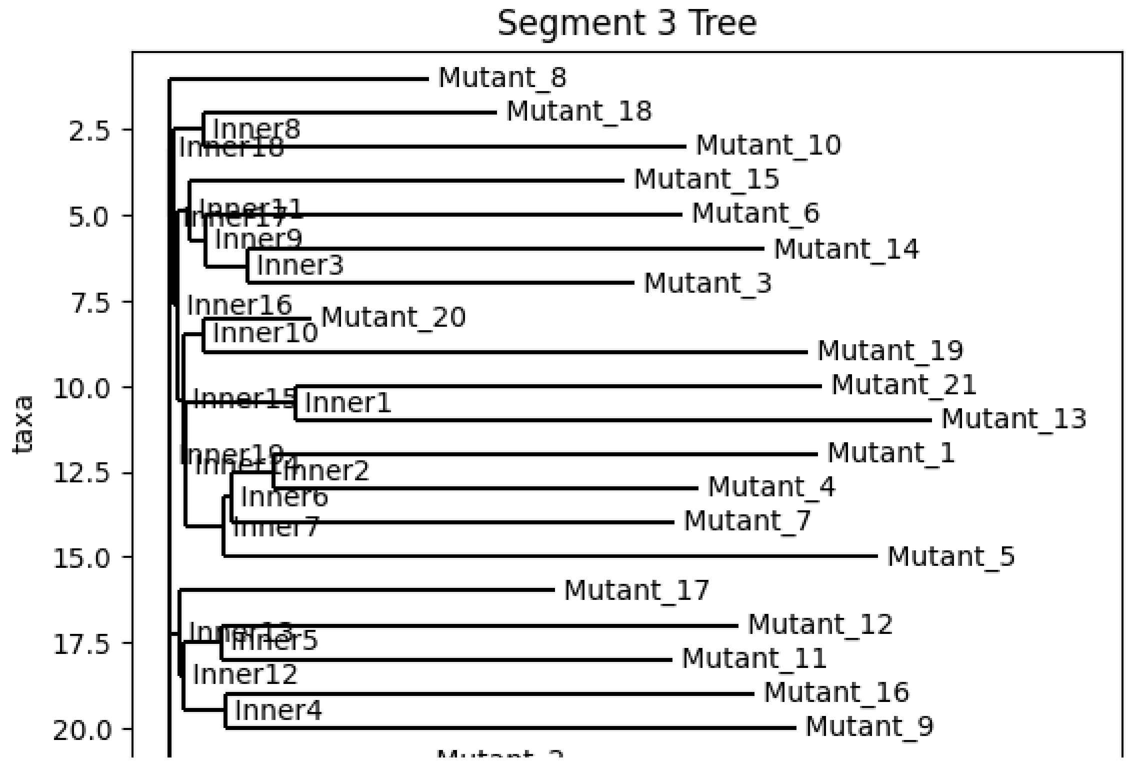
```
# Install ClustalW
!apt-get install clustalw

import random
from Bio import Phylo
from Bio.Align.Applications import ClustalwCommandline
from Bio.Phylo.TreeConstruction import DistanceTreeConstructor, DistanceCalculator
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
from Bio import AlignIO
from Bio import SeqIO
import matplotlib.pyplot as plt

# Function to generate random SNP mutants
```

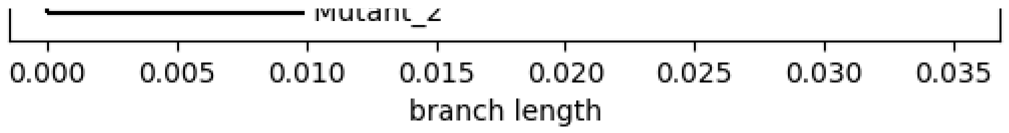
```
def generate_snp_mutants(sequence, num_mutants, mutation_rate):
    mutants = []
    for _ in range(num_mutants):
        mutant = list(sequence)
        for i in range(len(mutant)):
            if random.random() < mutation_rate[i % len(mutation_rate)]:
                mutant[i] = random.choice(['A', 'T', 'G', 'C'])
        mutants.append(SeqRecord(Seq("".join(mutant)), id=f"Mutant_{_+1}", description=""))
    return mutants

# Function to create phylogenetic tree
def create_phylogenetic_tree(sequences, tree_label):
    # Write sequences to a file
    SeqIO.write(sequences, "alignment.fasta", "fasta")

    # Perform multiple sequence alignment using ClustalW
    clustalw_cline = ClustalwCommandline("clustalw", infile="alignment.fasta")
    stdout, stderr = clustalw_cline()

    # Read the alignment result
    alignment = AlignIO.read("alignment.aln", "clustal")

    # Calculate distance matrix
    calculator = DistanceCalculator('identity')
    dm = calculator.get_distance(alignment)

    # Construct the phylogenetic tree
    constructor = DistanceTreeConstructor()
    tree = constructor.nj(dm)

    # Plot the phylogenetic tree
    plt.figure(figsize=(10, 5))
    Phylo.draw(tree, do_show=False)
    plt.title(tree_label)
    plt.savefig(f"{tree_label}.png")
    plt.show()

    return tree
```

```
# Function to calculate Robinson-Foulds distance between two trees
def calculate_robinson_foulds(tree1, tree2):
    rf_distance = tree1.robinson_foulds(tree2)
    return rf_distance

# Main script
coding_sequence = "ATG" * 300  # Example coding sequence
non_coding_sequence = "CGT" * 100  # Example non-coding sequence
combined_sequence = coding_sequence + non_coding_sequence

# Generate 21 SNP mutants
mutation_rates = [0.01, 0.01, 0.03, 0.05]
mutants = generate_snp_mutants(combined_sequence, 21, mutation_rates)

# Split sequences into three segments
segment1 = [SeqRecord(Seq(str(seq.seq[:400])), id=seq.id, description="") for seq in mutants]
segment2 = [SeqRecord(Seq(str(seq.seq[400:800])), id=seq.id, description="") for seq in mutants]
segment3 = [SeqRecord(Seq(str(seq.seq[800:])), id=seq.id, description="") for seq in mutants]

# Create and plot phylogenetic trees for each segment
tree1 = create_phylogenetic_tree(segment1, "Segment 1 Tree")
tree2 = create_phylogenetic_tree(segment2, "Segment 2 Tree")
tree3 = create_phylogenetic_tree(segment3, "Segment 3 Tree")

# Calculate Robinson-Foulds distances between the trees
rf_distance_1_2 = calculate_robinson_foulds(tree1, tree2)
rf_distance_1_3 = calculate_robinson_foulds(tree1, tree3)
rf_distance_2_3 = calculate_robinson_foulds(tree2, tree3)

print(f"Robinson-Foulds distance between Segment 1 Tree and Segment 2 Tree: {rf_distance_1_2}")
print(f"Robinson-Foulds distance between Segment 1 Tree and Segment 3 Tree: {rf_distance_1_3}")
print(f"Robinson-Foulds distance between Segment 2 Tree and Segment 3 Tree: {rf_distance_2_3}")
```


taxa
10.0
12.5
15.0
17.5
20.0
Inner13
Inner2
Inner19
Inner5
Inner12
Inner6
Inner8
Inner3
Inner11
Inner9
Inner7
Mutant_17
Mutant_1
Mutant_15
Mutant_11
Mutant_12
Mutant_8
Mutant_10
Mutant_18
Mutant_6
Mutant_14
Mutant_9
Mutant_5
0.000 0.005 0.010 0.015 0.020 0.025 0.030 0.035 0.040
branch length

```
<Figure size 1000x500 with 0 Axes>
```

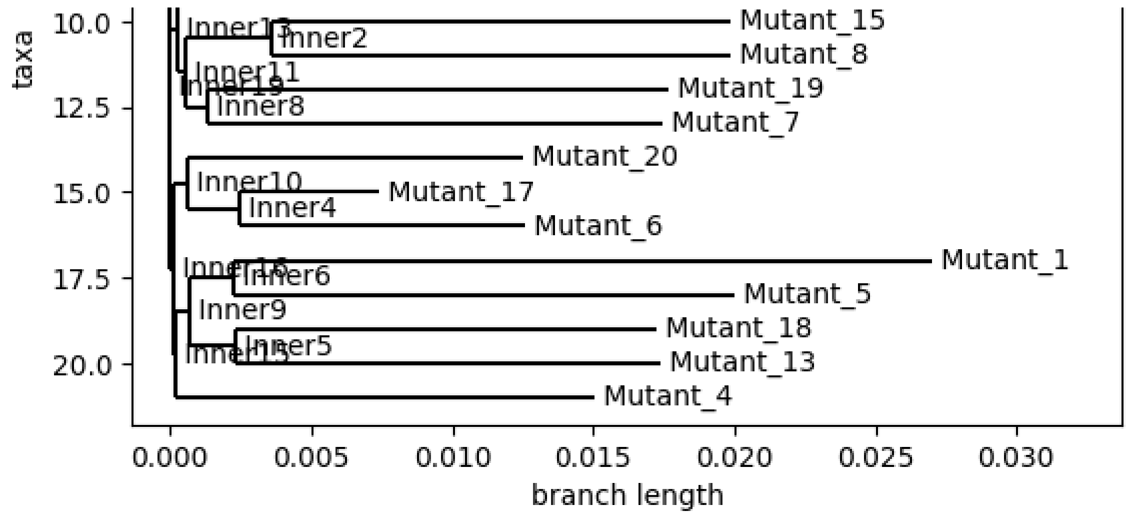

Segment 3 Tree
taxa
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Inner8
Inner18
Inner11
Inner17
Inner9
Inner3
Inner16
Inner10
Inner15
Inner1
Inner19
Inner14
Inner2
Inner6
Inner7
Inner13
Inner5
Inner12
Inner4
Mutant_8
Mutant_18
Mutant_10
Mutant_15
Mutant_6
Mutant_14
Mutant_3
Mutant_20
Mutant_19
Mutant_21
Mutant_13
Mutant_1
Mutant_4
Mutant_7
Mutant_5
Mutant_17
Mutant_12
Mutant_11
Mutant_16
Mutant_9



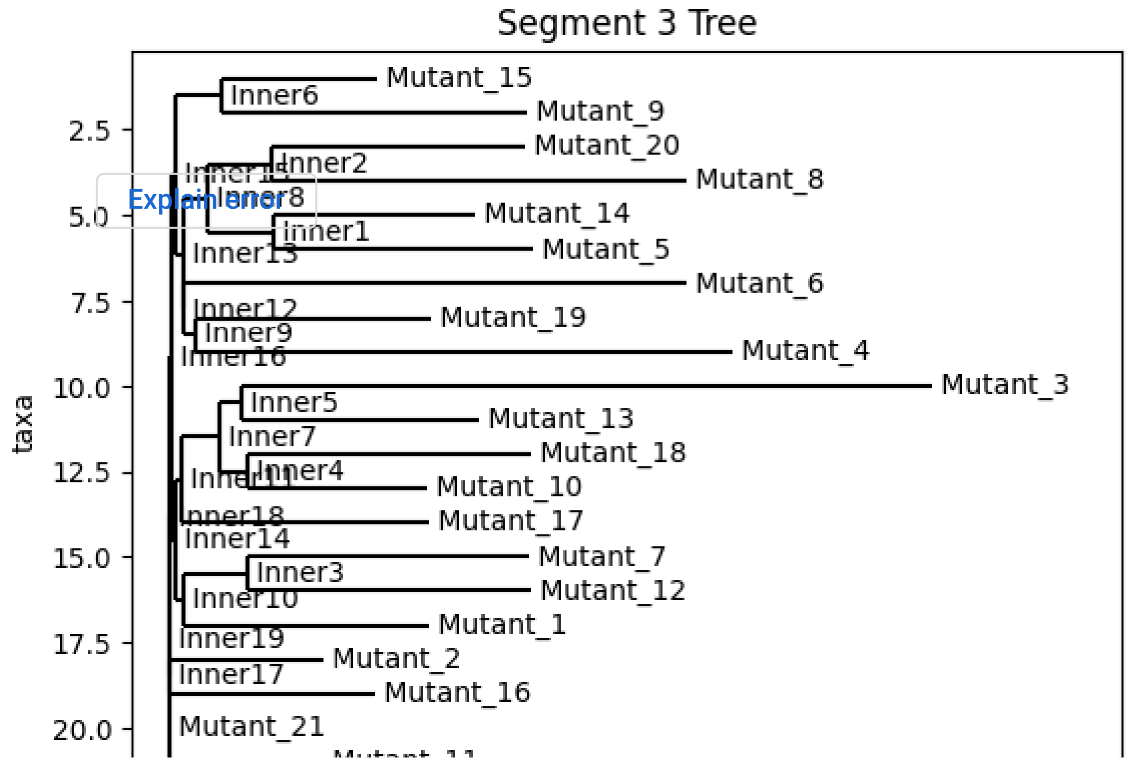
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-3-635db9e04cf3> in <cell line: 79>()
     77 
     78 # Calculate Robinson-Foulds distances between the trees
---> 79 rf_distance_1_2 = calculate_robinson_foulds(tree1, tree2)
     80 rf_distance_1_3 = calculate_robinson_foulds(tree1, tree3)
     81 rf_distance_2_3 = calculate_robinson_foulds(tree2, tree3)

<ipython-input-3-635db9e04cf3> in calculate_robinson_foulds(tree1, tree2)
     54 # Function to calculate Robinson-Foulds distance between two trees
     55 def calculate_robinson_foulds(tree1, tree2):
---> 56     rf_distance = tree1.robinson_foulds(tree2)
     57     return rf_distance
     58 
```

Next steps: Explain error

```
# Install ClustalW
!apt-get install clustalw

import random
from Bio import Phylo
from Bio.Align.Applications import ClustalwCommandline
from Bio.Phylo.TreeConstruction import DistanceTreeConstructor, DistanceCalculator
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
from Bio import AlignIO
from Bio import SeqIO
import matplotlib.pyplot as plt
from ete3 import Tree

# Function to generate random SNP mutants
def generate_snp_mutants(sequence, num_mutants, mutation_rate):
    mutants = []
    for _ in range(num_mutants):
        mutant = list(sequence)
        for i in range(len(mutant)):
            if random.random() < mutation_rate[i % len(mutation_rate)]:
                mutant[i] = random.choice(['A', 'T', 'G', 'C'])
        mutants.append(SeqRecord(Seq("".join(mutant)), id=f"Mutant_{_+1}", description=""))
    return mutants

# Function to create phylogenetic tree
def create_phylogenetic_tree(sequences, tree_label):
    # Write sequences to a file
    SeqIO.write(sequences, "alignment.fasta", "fasta")

    # Perform multiple sequence alignment using ClustalW
    clustalw_cline = ClustalwCommandline("clustalw", infile="alignment.fasta")
    stdout, stderr = clustalw_cline()

    # Read the alignment result
    alignment = AlignIO.read("alignment.aln", "clustal")

    # Calculate distance matrix
    calculator = DistanceCalculator('identity')
    dm = calculator.get_distance(alignment)

    # Construct the phylogenetic tree
    constructor = DistanceTreeConstructor()
    tree = constructor.nj(dm)

    # Save the tree to a file
    Phylo.write(tree, f"{tree_label}.xml", "phyloxml")

    # Plot the phylogenetic tree
    plt.figure(figsize=(10, 5))
    Phylo.draw(tree, do_show=False)
    plt.title(tree_label)
    plt.savefig(f"{tree_label}.png")
    plt.show()
    return tree

# Function to calculate Robinson-Foulds distance between two trees
def calculate_robinson_foulds(tree1, tree2):
    t1 = Tree(tree1.write(format=1), format=1)
    t2 = Tree(tree2.write(format=1), format=1)
    rf_distance, _, _, _, _ = t1.robinson_foulds(t2)
    return rf_distance

# Main script
coding_sequence = "ATG" * 300  # Example coding sequence
non_coding_sequence = "CGT" * 100  # Example non-coding sequence
combined_sequence = coding_sequence + non_coding_sequence

# Generate 21 SNP mutants
mutation_rates = [0.01, 0.01, 0.03, 0.05]
mutants = generate_snp_mutants(combined_sequence, 21, mutation_rates)

# Split sequences into three segments
segment1 = [SeqRecord(Seq(str(seq.seq[:400])), id=seq.id, description="") for seq in mutants]
segment2 = [SeqRecord(Seq(str(seq.seq[400:800])), id=seq.id, description="") for seq in mutants]
segment3 = [SeqRecord(Seq(str(seq.seq[800:])), id=seq.id, description="") for seq in mutants]

# Create and plot phylogenetic trees for each segment
tree1 = create_phylogenetic_tree(segment1, "Segment 1 Tree")
tree2 = create_phylogenetic_tree(segment2, "Segment 2 Tree")
tree3 = create_phylogenetic_tree(segment3, "Segment 3 Tree")

# Calculate Robinson-Foulds distances between the trees
rf_distance_1_2 = calculate_robinson_foulds(tree1, tree2)
rf_distance_1_3 = calculate_robinson_foulds(tree1, tree3)
rf_distance_2_3 = calculate_robinson_foulds(tree2, tree3)

print(f"Robinson-Foulds distance between Segment 1 Tree and Segment 2 Tree: {rf_distance_1_2}")
print(f"Robinson-Foulds distance between Segment 1 Tree and Segment 3 Tree: {rf_distance_1_3}")
print(f"Robinson-Foulds distance between Segment 2 Tree and Segment 3 Tree: {rf_distance_2_3}")
```



```
<Figure size 1000x500 with 0 Axes>
```

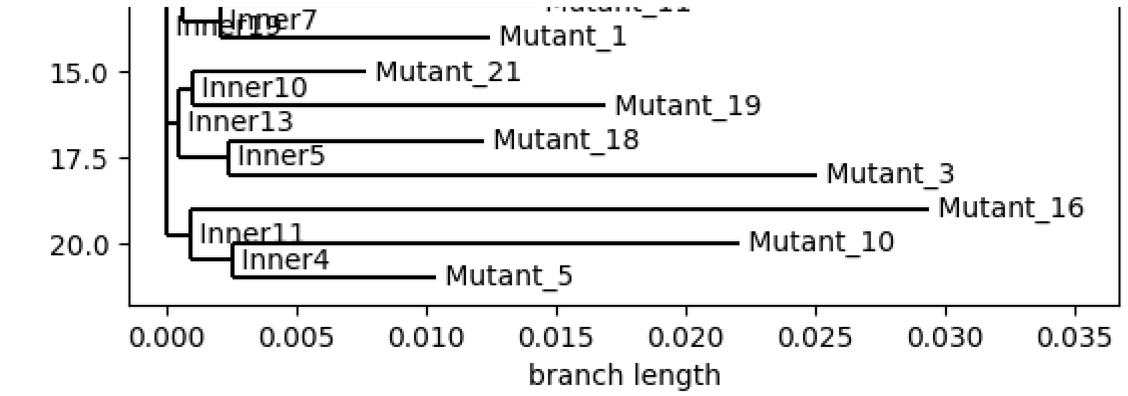


Next

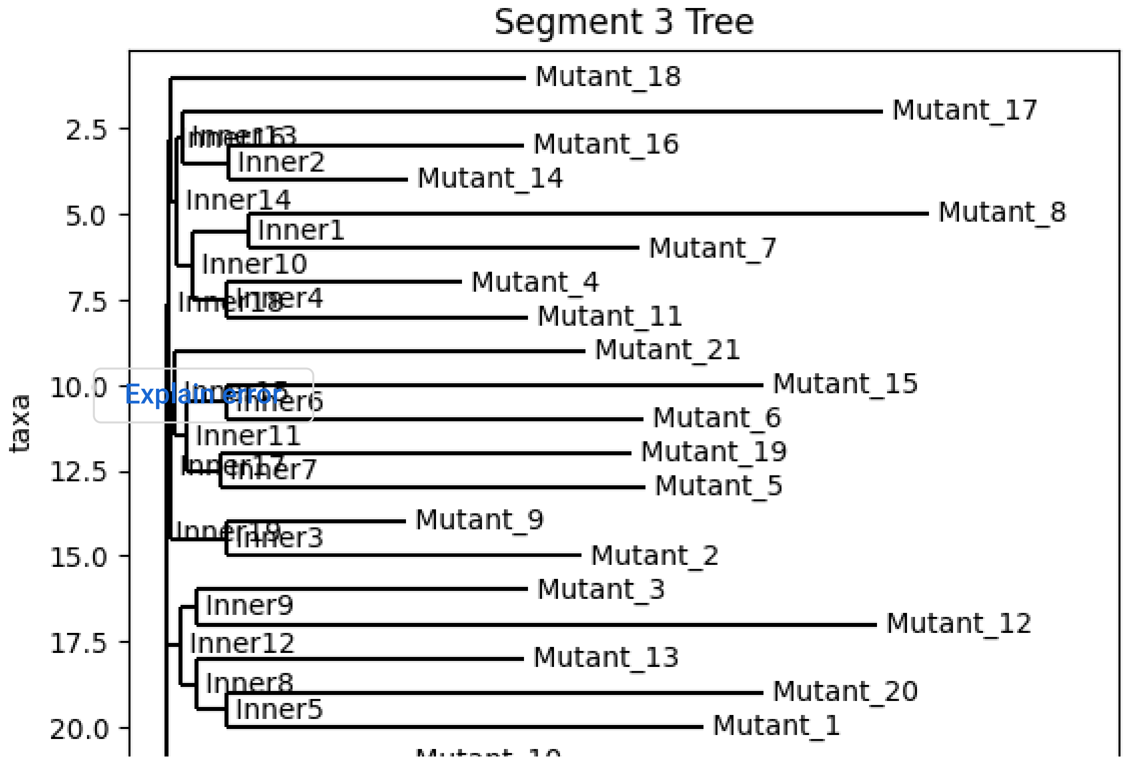
Explain error

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-4-dfbcc136b474> in <cell line: 85>()
     83 
     84 # Calculate Robinson-Foulds distances between the trees
---> 85 rf_distance_1_2 = calculate_robinson_foulds(tree1, tree2)
     86 rf_distance_1_3 = calculate_robinson_foulds(tree1, tree3)
     87 rf_distance_2_3 = calculate_robinson_foulds(tree2, tree3)

<ipython-input-4-dfbcc136b474> in calculate_robinson_foulds(tree1, tree2)
     58 # Function to calculate Robinson-Foulds distance between two trees
     59 def calculate_robinson_foulds(tree1, tree2):
---> 60     t1 = Tree(tree1.write(format=1), format=1)
     61     t2 = Tree(tree2.write(format=1), format=1)
     62     rf_distance, _, _, _, _ = t1.robinson_foulds(t2)
```

```
# Install ClustalW
!apt-get install clustalw

import random
from Bio import Phylo
from Bio.Align.Applications import ClustalwCommandline
from Bio.Phylo.TreeConstruction import DistanceTreeConstructor, DistanceCalculator
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
from Bio import AlignIO
from Bio import SeqIO
import matplotlib.pyplot as plt
from ete3 import Tree

# Function to generate random SNP mutants
def generate_snp_mutants(sequence, num_mutants, mutation_rate):
    mutants = []
    for _ in range(num_mutants):
        mutant = list(sequence)
        for i in range(len(mutant)):
            if random.random() < mutation_rate[i % len(mutation_rate)]:
                mutant[i] = random.choice(['A', 'T', 'G', 'C'])
        mutants.append(SeqRecord(Seq("".join(mutant)), id=f"Mutant_{_+1}", description=""))
    return mutants

# Function to create phylogenetic tree
def create_phylogenetic_tree(sequences, tree_label):
    # Write sequences to a file
    SeqIO.write(sequences, "alignment.fasta", "fasta")

    # Perform multiple sequence alignment using ClustalW
    clustalw_cline = ClustalwCommandline("clustalw", infile="alignment.fasta")
    stdout, stderr = clustalw_cline()

    # Read the alignment result
    alignment = AlignIO.read("alignment.aln", "clustal")

    # Calculate distance matrix
    calculator = DistanceCalculator('identity')
    dm = calculator.get_distance(alignment)

    # Construct the phylogenetic tree
    constructor = DistanceTreeConstructor()
    tree = constructor.nj(dm)

    # Save the tree to a file
    Phylo.write(tree, f"{tree_label}.xml", "phyloxml")

    # Plot the phylogenetic tree
    plt.figure(figsize=(10, 5))
    Phylo.draw(tree, do_show=False)
    plt.title(tree_label)
    plt.savefig(f"{tree_label}.png")
    plt.show()
    return tree

# Function to calculate Robinson-Foulds distance between two trees
def calculate_robinson_foulds(tree1, tree2):
    newick1 = tree1.format('newick')
    newick2 = tree2.format('newick')
    t1 = Tree(newick1, format=1)
    t2 = Tree(newick2, format=1)
    rf_distance, _, _, _, _ = t1.robinson_foulds(t2)
    return rf_distance

# Main script
coding_sequence = "ATG" * 300  # Example coding sequence
non_coding_sequence = "CGT" * 100  # Example non-coding sequence
combined_sequence = coding_sequence + non_coding_sequence

# Generate 21 SNP mutants
mutation_rates = [0.01, 0.01, 0.03, 0.05]
mutants = generate_snp_mutants(combined_sequence, 21, mutation_rates)

# Split sequences into three segments
segment1 = [SeqRecord(Seq(str(seq.seq[:400])), id=seq.id, description="") for seq in mutants]
segment2 = [SeqRecord(Seq(str(seq.seq[400:800])), id=seq.id, description="") for seq in mutants]
segment3 = [SeqRecord(Seq(str(seq.seq[800:])), id=seq.id, description="") for seq in mutants]

# Create and plot phylogenetic trees for each segment
tree1 = create_phylogenetic_tree(segment1, "Segment 1 Tree")
tree2 = create_phylogenetic_tree(segment2, "Segment 2 Tree")
tree3 = create_phylogenetic_tree(segment3, "Segment 3 Tree")

# Calculate Robinson-Foulds distances between the trees
rf_distance_1_2 = calculate_robinson_foulds(tree1, tree2)
rf_distance_1_3 = calculate_robinson_foulds(tree1, tree3)
rf_distance_2_3 = calculate_robinson_foulds(tree2, tree3)

print(f"Robinson-Foulds distance between Segment 1 Tree and Segment 2 Tree: {rf_distance_1_2}")
print(f"Robinson-Foulds distance between Segment 1 Tree and Segment 3 Tree: {rf_distance_1_3}")
print(f"Robinson-Foulds distance between Segment 2 Tree and Segment 3 Tree: {rf_distance_2_3}")
```



```
<Figure size 1000x500 with 0 Axes>
```



Next Explain error