# test\_probability\_calculator.py saved under the 5. Tests folder

‘’’

Tests:

1. Verify that the p-value is calculated correctly for a given alignment score.
2. Check if the p-value reflects the expected statistical significance.

Further **Tests but they relate to edge cases**:

1. Test the probability and p-value calculation with **extremely high** and **low alignment scores**.
2. Check for sequences with **perfect matches** or **completely different sequences**.

‘’’

import unittest

from src.probability\_calculator import calculate\_p\_value, generate\_score\_distribution

from Bio import SeqIO

from src.align\_sequences import align\_sequences

class TestProbabilityCalculator(unittest.TestCase):

def test\_probability\_calculation(self):

"""Test the p-value calculation based on a given alignment score."""

# Assuming the alignment score is 50 and we have a known score distribution

score\_distribution = [45, 47, 49, 50, 52, 55, 56]

p\_value = calculate\_p\_value(50, score\_distribution)

self.assertAlmostEqual(p\_value, 0.5, places=2) # Check that p-value is reasonable (depends on distribution)

def test\_p\_value\_extreme\_high\_score(self):

"""Test the p-value for an extremely high alignment score."""

score\_distribution = [45, 47, 49, 50, 52, 55, 56]

p\_value = calculate\_p\_value(100, score\_distribution)

self.assertLess(p\_value, 0.01) # High alignment score should lead to a low p-value

def test\_p\_value\_extreme\_low\_score(self):

"""Test the p-value for an extremely low alignment score."""

score\_distribution = [45, 47, 49, 50, 52, 55, 56]

p\_value = calculate\_p\_value(10, score\_distribution)

self.assertGreater(p\_value, 0.99) # Low alignment score should lead to a high p-value

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()