**深 圳 大 学 实 验 报 告**

**课程名称：­ 概率论与数理统计**

**实验项目名称： 随机概率和德摩根定理的验证**

**学院： 电子与信息工程学院**

**专业： 电子信息工程**

**指导教师： 陈昌盛**

**报告人： 陈应权 学号： 2022280297**

**班级： 06**

**实验时间： 2023年9月20日——9月25日**

**实验报告提交时间： 9月28日**

**教务处制**

|  |
| --- |
| Aim of Experiment:   1. Use python programming to verify whether the probability of heads under the event of coin throwing is 1/2. 2. Verify the relevant theorems in the set, including de Morgan's theorem, inclusion-exclusion principle. |
| Experiment Content:  1.Probability  In this excercise you will write code to estimate the probability that n flips of a fair coin will result in number of "heads" between k1 and k2.  You should write the body of two functions:  ①seq\_sum(n): generates a random sequence of coin flips and counts the number of heads.  ②estimate\_prob(n,k1,k2,m): Using calls to seq\_sum , estimate the probability of the number of heads being between k1 and k2.  2.Set  Verify the relevant theorems in the set, including de Morgan's theorem, inclusion-exclusion principle.   1. Write the function **complement\_of\_union** that first determines A∪B and then evaluates the complement of this set. Output the tuple: (A∪B,(A∪B)c)   De Morgan's first law states the following for any two sets A and B  (A∪B)c=Ac∩Bc   1. In the following two exercises we calculate (A∪B)c in two different ways. Both functions must take A, B and the universal set U as their inputs. 2. Two methods are used to obtain the module of AUB. |
| Experiment Process：  1.Probability  (1)Write a function, seq\_sum(n), which generates n random coin flips from a fair coin and then returns the number of heads. A fair coin is defined to be a coin where P(heads)=12  The output type should be a numpy integer, **hint:** use np.random.rand()  def seq\_sum(n):  """ input: n, generate a sequence of n random coin flips  output: return the number of heads  Hint: For simplicity, use 1,0 to represent head,tails  """  coin\_list=np.random.rand(n) # Generates n random variables between 0 and 1  head=sum(coin\_list>0.5) # Count the number greater than 0.5, and interpret the number greater than 0.5 as the coin facing up, and use the sum function to calculate the number of coins facing up  return head  (2)Write a function, estimate\_prob(n,k1,k2,m), that uses seq\_sum(n) to estimate the following probability:  P(k1<=number of heads in n flips<k2)  The function should estimate the probability by running m different trials of seq\_sum(n), probably using a for loop. In order to receive full credit estimate\_prob MUST call seq\_sum  def estimate\_prob(n,k1,k2,m):  success=0 # The number of successful falls between K1-K2 is counted and initialized to 0  for i in range(m): # Conduct m experiments  heads=seq\_sum(n) # Call the seq sum function, generate n coins, and return the number of heads  if k1<=heads<=k2-1:  success+=1 # If the number of upturned coins is between k1 and k2, the number of successes +1 '  probability=success\*1.0/m # Because the floating-point number and the success rate are required to be returned, \*1.0.  return probability  2.Set  (1) Write the function **complement\_of\_union** that first determines A∪B and then evaluates the complement of this set. Output the tuple: (A∪B,(A∪B)c).  def complement\_of\_union(A, B, U):  # inputs: A, B and U are of type 'set'  # output: a tuple of the type (set, set)  A\_and\_B=A.union(B) # Using the union function, generate A and B  U.difference\_update(A\_and\_B) # Generating complement  tuple\_result=(set(A\_and\_B),set(U)) # Output the collection as a tuplereturn tuple\_result   1. Write the function intersection\_of\_complements that first determines Ac and Bc and then evaluates the intersection of their complements. Output the tuple: (Ac,Bc∩Ac)   def intersection\_of\_complements(A, B, U):  # inputs: A, B and U are of type 'set'  # output: a tuple of the form (set, set)  A\_c=U.difference(A) # A\_c , B\_c represents the complement of A and B in U  B\_c=U.difference(B) # It can be solved directly by a difference function  temp\_set=B\_c.copy() #Create a copy of B complement  B\_c.difference\_update(A\_c) # B complement minus A complement  temp\_set.difference\_update(B\_c) # And then B complement minus B which is specific to b is the intersection of the two  result\_tuple=(set(A\_c),set(temp\_set))  return result\_tuple   1. The inclusion-exclusion principle states that for two sets A and B,   |A∪B|=|A|+|B|−|A∩B|.  Write the following functions to determine |A∪B| in two different ways.  A function union that determines first A∪B and then evaluates the union's size. Output the ordered pair (A∪B,|A∪B|).  def union(A, B):  # inputs: A and B are of type 'set'  # output: a tuple of the type (set, set\_length)    A\_and\_B=A.union(B) # Use the union function to generate A and B set\_length=len(A\_and\_B) #len function calculate length  return(set(A\_and\_B),set\_length)   1. A function **inclusion\_exclusion** that first deterimines |A|, |B|, A∩B, and |A∩B|, and then uses the inclusion-exclusion formula to determine |A∪B|. Output the tuple (|A|,|B|,|A∩B|,|A∪B|)   def inclusion\_exclusion(A, B):  # inputs: A and B are of type 'set'  # output: a tuple of four integers  A\_length=len(A)  B\_length=len(B)  B\_A=B.difference(A) # Find the B-A event  AB=B.difference(B\_A) # Find A∩B  AB\_length=len(AB) #We can actually use intersection later to find  A\_and\_B=A\_length + B\_length - AB\_length  tuple\_result=(A\_length,B\_length,AB\_length,A\_and\_B)  return tuple\_result   1. The inclusion-exclusion principle says that for three sets A, B and C,   |A∪B∪C|=|A|+|B|+|C|−|A∩B|−|A∩C|−|B∩C|+|A∩B∩C|  We will write the following functions to determine |A∪B∪C| in two different ways.  Write function **union3** that first determines A∪B∪C and then evaluates the size of this union. Output the tuple (A∪B∪C,|A∪B∪C|).  def union3(A, B, C):  # inputs: A, B and C are of type 'set'  # output: a tuple of the type (set, set\_length)  union\_set = A.union(B, C)  union\_size = len(union\_set)  return (union\_set, union\_size)   1. A function **inclusion\_exclusion3** that first deterimines the sizes of A, B, C and their mutual intersections, and then uses the inclusion-exclusion principle to determine the size of the union. Output the tuple (|A∩B∩C|,|A∪B∪C|). Note that for brevity we are asking you to output the intermediate answer just for A∩B∩C, but you need to calculate all.   def inclusion\_exclusion3(A, B, C):  # inputs: A, B and C are of type 'set'  # output: a tuple of two integers    #  # YOUR CODE HERE  #  A\_length=len(A) #len function calculate length  B\_length=len(B)  C\_length=len(C)  AB\_intersection=A.intersection(B) #find intersection  AC\_intersection=A.intersection(C)  BC\_intersection=B.intersection(C)  intersection\_set = A.intersection(B, C)  intersection\_length=len(intersection\_set)  AB\_intersection\_length=len(AB\_intersection) #find length  AC\_intersection\_length=len(AC\_intersection)  BC\_intersection\_length=len(BC\_intersection)  Union\_ABC=A\_length + B\_length + C\_length - AB\_intersection\_length - AC\_intersection\_length - BC\_intersection\_length + intersection\_length  result\_tuple=(intersection\_length, Union\_ABC)  return result\_tuple |
| Data Logging and Processing:  （1）Graph analysis of coin probabilities：        (2)  Through this experiment, I deepened my understanding of frequency and probability in probability theory, consolidated the laws related to sets, and more importantly, improved my ability to handle sets using python.  For example:   1. Application of union function，which can directly generates the union of two sets.   Just like in experience 2-6:  union\_set = A.union(B, C)  If we don’t use this function, we have to make it step by step, just like:  temp\_set = set()  for i in A:  for j in B:  if j != i:  temp\_set.add(j)  A.update(temp\_set)  It is tedious by comparison！！！   1. Application of union function，which can directly generates the intersection of two sets.   Just like in experience 2-7:  AB\_intersection=A.intersection(B) #求交集  If we don’t use this function, we have to make it step by step, just like:  temp\_set=B\_c.copy() #创造一个B补的副本  B\_c.difference\_update(A\_c) #求得B补-A补  temp\_set.difference\_update(B\_c) #再用B补减去B特有的，就是两者交集  It is tedious by comparison！！！   1. Finally, and most importantly, I learned that we can use python to draw a graph of coordinates to depict the distribution of probabilities, such as the handling of the coin results in Experiment 1 |
| Experimental Results and Analysis:  Experimental Results:   1. After analyzing the results of the coin toss experiment, we know that over many experiments, the frequency of the silver coin facing up tends to probability 1/2 2. Through experiment 2-1, we successfully prove the correctness of De Morgan's theorem by the operation of the set 3. We successfully obtain the module of A∪B by two different methods, and extend it to the union of three sets |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。