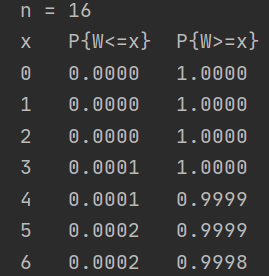
**비모수통계학 중간과제**

1. 다음 검정들의 귀무분포를 구하는 코드를 작성하여라.(Python 사용)

1) Signed-rank test (n=16)

from itertools import product  
import numpy as np  
def null\_signed(n):  
 ranks = [\*range(1, n + 1, 1)]  
 sum\_ranks=dict()  
 for i in product([0,1],repeat=n):  
 indices = [a for a, x in enumerate(i) if x == 1]  
 if not indices:  
 temp\_sum=0  
 else:  
 temp\_sum=sum([ranks[n-1-b] for b in indices])  
 if temp\_sum not in sum\_ranks.keys():  
 sum\_ranks[temp\_sum]=1  
 else:  
 sum\_ranks[temp\_sum]+=1  
 total = sum([sum\_ranks[i] for i in range(0,int(n\*(n+1)/2)+1)])  
 smaller = [];  
 greater = []  
 small = 0;  
 big = total  
 for i in range(int(n\*(n+1)/2)+1):  
 small += sum\_ranks[i];  
 smaller.append('{0:.4f}'.format(small / total))  
 greater.append('{0:.4f}'.format(big / total))  
 big -= sum\_ranks[i]  
 result = np.column\_stack((smaller, greater))  
 x = list(sum\_ranks.keys())  
 print(" n =", n)  
 print(" x P{W<=x} P{W>=x}")  
 for i in range(int(n\*(n+1)/2)+1):  
 print('{:3d}'.format(x[i]) + " " + result[i][0] + " " + result[i][1])  
null\_signed(16)

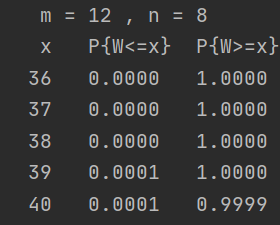
부호순위검정에서 귀무분포를 구하는 코드다. 인자는 하나로, n을 넣어주면 다음과 같이 출력이 된다.



2) Wilcoxon rank sum test (m=12, n=8)

from itertools import combinations  
import numpy as np  
def null\_ranksum(m,n):  
 ranks = [\*range(1, m+n + 1, 1)]  
 nCr = combinations(ranks, n)  
 rank\_list=dict()  
 for seq in nCr:  
 sum\_seq=sum(seq)  
 if sum\_seq not in rank\_list.keys():  
 rank\_list[sum\_seq]=1  
 else:  
 rank\_list[sum\_seq]+=1  
 total=sum([rank\_list[i] for i in range(int(n\*(n+1)/2),int(n\*(n+1)/2)+m\*n+1)])  
 smaller=[];greater=[]  
 small=0;big=total  
 for i in range(int(n\*(n+1)/2),int(n\*(n+1)/2)+m\*n+1):  
 small+=rank\_list[i];  
 smaller.append('{0:.4f}'.format(small/total))  
 greater.append('{0:.4f}'.format(big / total))  
 big -= rank\_list[i]  
 result = np.column\_stack((smaller, greater))  
 x=list(rank\_list.keys())  
 print(" m =",m,", n =",n)  
 print(" x P{W<=x} P{W>=x}")  
 for i in range(0,m\*n+1):  
 print('{:3d}'.format(x[i])+" "+result[i][0]+" "+result[i][1])  
null\_ranksum(12,8)

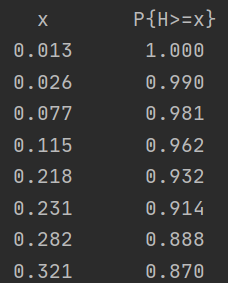
윌콕슨 순위합검정에서 귀무분포를 구하는 코드다. 인자는 두개로, m과 n을 넣어주면 다음과 같이 출력이 된다.



3) Kruskal-Wallis test (n1=3, n2=3, n3=6)

from itertools import combinations  
def null\_ks(\*args):  
 total\_len=0  
 for i in range(len(args)):  
 total\_len+=args[i]  
 ranks=[\*range(1,total\_len+1,1)]  
 h\_list=dict()  
 perm\_rank=combinations(ranks,args[0])  
 for perm in perm\_rank:  
 dup\_ranks=ranks[:]  
 for ele in perm:  
 dup\_ranks.remove(ele)  
 next\_comb=combinations(dup\_ranks,args[1])  
 for next\_perm in next\_comb:  
 rs = 0  
 group\_sums = []  
 group\_sums.append(sum(perm))  
 group\_sums.append(sum(next\_perm))  
 group\_sums.append((total\_len) \* (total\_len + 1) / 2 - group\_sums[0] - group\_sums[1])  
 for i in range(len(args)):  
 rs += pow(group\_sums[i], 2) / args[i]  
 h = 12.0 / (total\_len \* (total\_len + 1)) \* rs - 3 \* (total\_len + 1)  
 h=round(h,4)  
 if h not in h\_list.keys():  
 h\_list[h] = 1  
 else:  
 h\_list[h] += 1  
 res=sorted(h\_list.items())  
 total=sum([res[i][1] for i in range(0,len(res))])  
 cum=total  
 print(" x P{H>=x}")  
 for i in range(len(res)):  
 print('{0:6.3f}'.format(res[i][0]) + " " + str('{0:.3f}'.format(cum/total)))  
 cum-=res[i][1]  
  
null\_ks(3,3,6)

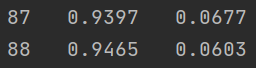
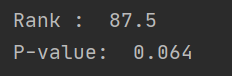
크루스칼 월리스 검정에서 귀무분포를 구하는 코드다. 이때, 처리는 3가지일 때만 사용할 수 있으며 세 그룹의 블록 수를 인자로 넣어주면 다음과 같이 출력이 된다.



1. 문제풀이에 대한 코드(wilcox.test 등이 아닌 코드 구현)를 작성한 후 각 데이터에 대한 검정을 실시하라

1) 부호순위검정을 이용하여 = 0.05에서 평균의 값이 160보다 큰가에 대하여 검정하여라 (단측 검정).

from itertools import product  
def null\_signed(n,w):  
 ranks = [\*range(1, n + 1, 1)]  
 sum\_ranks=dict()  
 for i in product([0,1],repeat=n):  
 indices = [a for a, x in enumerate(i) if x == 1]  
 if not indices:  
 temp\_sum=0  
 else:  
 temp\_sum=sum([ranks[n-1-b] for b in indices])  
 if temp\_sum not in sum\_ranks.keys():  
 sum\_ranks[temp\_sum]=1  
 else:  
 sum\_ranks[temp\_sum]+=1  
 total = sum([sum\_ranks[i] for i in range(0,int(n\*(n+1)/2)+1)])  
 smaller = [];  
 greater = []  
 small = 0;  
 big = total  
 for i in range(int(n\*(n+1)/2)+1):  
 small += sum\_ranks[i];  
 smaller.append('{0:.4f}'.format(small / total))  
 greater.append('{0:.4f}'.format(big / total))  
 big -= sum\_ranks[i]  
 stand=n\*(n+1)/4  
 if w>stand:  
 pval=(w-int(w))\*(float(greater[int(w)+1])-float(greater[int(w)]))+float(greater[int(w)])  
 else:  
 pval=(w-int(w))\*(float(smaller[int(w)+1])-float(smaller[int(w)]))+float(smaller[int(w)])  
 return pval  
def rankdata(list):  
 temp\_rank=dict()  
 for i in sorted(list):  
 if i not in temp\_rank.keys():  
 temp\_rank[i]=1  
 else:  
 temp\_rank[i]+=1  
 rank=dict();r=0  
 for i in temp\_rank.keys():  
 if temp\_rank[i]==1:  
 rank[i]=r+1;r+=1  
 else:  
 rank[i]=temp\_rank[i]\*(r+1+r+temp\_rank[i])/(2\*temp\_rank[i])  
 r+=temp\_rank[i]  
 final\_rank=[]  
 for i in list:  
 final\_rank.append(rank[i])  
 return final\_rank  
def rank\_sum(x,m):  
 abs\_remove\_x=[abs(a-m) for a in x if a!=m]  
 remove\_x=[a-m for a in x if a!=m]  
 abs\_x = [abs(x - m) for x in x]  
 t=rankdata(abs\_x)  
 table=dict()  
 for i in sorted(t):  
 if i not in table.keys():  
 table[i]=1  
 else:  
 table[i]+=1  
 rankSum=0  
 final\_rank=rankdata(abs\_remove\_x)  
 for rank, i in zip(final\_rank,remove\_x):  
 if i >0:  
 rankSum+=rank  
 print("Rank : ", rankSum)  
 pval=null\_signed(len(remove\_x),rankSum)  
 print("P-value: ",pval)  
  
x=[176.9, 158.3, 152.1, 158.8, 172.4, 169.8, 159.7, 162.7, 156.6, 174.5, 184.4, 165.2, 147.6, 177.8, 160.0, 160.5]  
rank\_sum(x,160)

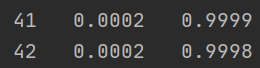
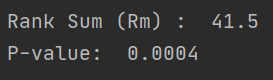


**(평균 값이 160이다) vs (평균 값이 160보다 크다)**

위 코드는 부호순위검정에서 단측 p값을 반환한다. 동점이 있어 부호순위검정통계량을 계산할 때는 귀무분포에서 선형보간을 해서 반환한다. alpha=0.05에서 부호순위검정을 이용해 구한 부호순위검정통계량은 87.5이다. 에 대한 선형보간된 p-value는 0.064>0.05이므로 을 기각할 증거가 부족하다. 즉, 평균의 값이 160보다 크다고 말할 수 없다. (선형보간을 원하지 않을 때는, 1번에서 작성한 귀무분포 함수에 동점인 160 하나를 제거해서, n=15를 주면 p값들을 확인할 수 있다. P(W>=87)=0.0677, P(W>=88)=0.0603이므로 동일한 결과를 얻는다)

2) 다음은 A 집단과 B 집단의 점수를 나타내는 데이터다. 윌콕슨 순위합검정을 이용하여 = 0.05에서 두 집단 사이에 점수의 차이가 있는지 검정하여라.

from itertools import combinations  
def null\_ranksum(n,m,w):  
 ranks = [\*range(1, m+n + 1, 1)]  
 nCr = combinations(ranks, n)  
 rank\_list=dict()  
 for seq in nCr:  
 sum\_seq=sum(seq)  
 if sum\_seq not in rank\_list.keys():  
 rank\_list[sum\_seq]=1  
 else:  
 rank\_list[sum\_seq]+=1  
 total=sum([rank\_list[i] for i in range(int(n\*(n+1)/2),int(n\*(n+1)/2)+m\*n+1)])  
 smaller=[];greater=[]  
 small=0;big=total  
 for i in range(int(n\*(n+1)/2),int(n\*(n+1)/2)+m\*n+1):  
 small+=rank\_list[i];  
 smaller.append('{0:.4f}'.format(small/total))  
 greater.append('{0:.4f}'.format(big / total))  
 big -= rank\_list[i]  
 stand=n\*(n+m+1)/2  
 x=list(rank\_list.keys())  
 t=w  
 w-=x[0]  
 if t<stand:  
 pval=(w-int(w))\*(float(smaller[int(w)+1])-float(smaller[int(w)]))+float(smaller[int(w)])  
 else:  
 pval=(w-int(w))\*(float(greater[int(w)+1])-float(greater[int(w)]))+float(greater[int(w)])  
 return pval  
  
def rankdata(list):  
 temp\_rank=dict()  
 for i in sorted(list):  
 if i not in temp\_rank.keys():  
 temp\_rank[i]=1  
 else:  
 temp\_rank[i]+=1  
 rank=dict();r=0  
 for i in temp\_rank.keys():  
 if temp\_rank[i]==1:  
 rank[i]=r+1;r+=1  
 else:  
 rank[i]=temp\_rank[i]\*(r+1+r+temp\_rank[i])/(2\*temp\_rank[i])  
 r+=temp\_rank[i]  
 final\_rank=[]  
 for i in list:  
 final\_rank.append(rank[i])  
 return final\_rank  
def rank\_sum(x,y):  
 total\_set=[]  
 which\_set=[]  
 for i in sorted(x):  
 total\_set.append(i)  
 which\_set.append('X')  
 for i in sorted(y):  
 total\_set.append(i)  
 which\_set.append('Y')  
  
 x\_ranks = []; y\_ranks = []  
 ranks=rankdata(total\_set)  
  
 for rank, i in zip(ranks,which\_set):  
 if i == 'X':  
 x\_ranks.append(rank)  
 else:  
 y\_ranks.append(rank)  
  
 x\_rankTotal = sum(x\_ranks)  
 y\_rankTotal = sum(y\_ranks)  
  
 if len(x) <= len(y):  
 rankSum = x\_rankTotal  
 m = len(x); n = len(y)  
 else:  
 rankSum = y\_rankTotal  
 m=len(y); n=len(x)  
 pval=null\_ranksum(m,n,rankSum)  
  
 print("Rank Sum (Rm) : ", rankSum)  
 print("P-value: ",2\*pval)  
  
a = [70, 80, 72, 76, 76, 76, 72, 78, 82, 92, 68, 84]  
b = [68, 72, 62, 70, 66, 68, 52, 64]  
  
rank\_sum(a,b)

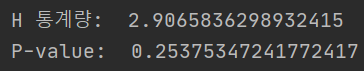


**두 집단의 위치모수는 같다(두 집단 사이에 점수 차이가 없다) vs : 두 집단의 위치모수는 다르다(두 집단 사이에 점수 차이가 있다)**

위 코드는 윌콕슨 순위합검정에서 양측 p값을 반환한다. 동점이 있어 순위합검정통계량을 계산할 때는 귀무분포에서 선형보간을 해서 반환한다. alpha=0.05에서 순위합검정을 이용해 구한 순위합검정통계량은 41.5이다. 에 대한 p-value는 0.0004<0.05이므로 을 기각한다. 즉, 두 집단 A와 B 사이에 점수의 차이가 있다. (선형보간을 원하지 않을 때는, 1번에서 작성한 귀무분포 함수에 m=12, n=8를 주면 p값들을 확인할 수 있다. P(W<=41)=0.0002, P(W<=42)=0.0002이므로 동일한 결과를 얻는다)

3) 크러스칼-월리스 검정을 이용하여 = 0.05에서 세 그룹 사이에 차이가 있다고 할 수 있는지 검정하여라 (코드 output으로는 검정통계량만 보이고 p-value는 안 보여도 됩니다).

from itertools import combinations  
import numpy as np  
def null\_ks(\*args,h\_val=1):  
 total\_len=0  
 for i in range(len(args)):  
 total\_len+=args[i]  
 ranks=[\*range(1,total\_len+1,1)]  
 h\_list=dict()  
 perm\_rank=combinations(ranks,args[0])  
 for perm in perm\_rank:  
 dup\_ranks=ranks[:]  
 for ele in perm:  
 dup\_ranks.remove(ele)  
 next\_comb=combinations(dup\_ranks,args[1])  
 for next\_perm in next\_comb:  
 rs = 0  
 group\_sums = []  
 group\_sums.append(sum(perm))  
 group\_sums.append(sum(next\_perm))  
 group\_sums.append((total\_len) \* (total\_len + 1) / 2 - group\_sums[0] - group\_sums[1])  
 for i in range(len(args)):  
 rs += pow(group\_sums[i], 2) / args[i]  
 h = 12.0 / (total\_len \* (total\_len + 1)) \* rs - 3 \* (total\_len + 1)  
 h=round(h,4)  
 if h not in h\_list.keys():  
 h\_list[h] = 1  
 else:  
 h\_list[h] += 1  
 res=sorted(h\_list.items())  
 total=sum([res[i][1] for i in range(0,len(res))])  
 cum=total  
 for i in range(len(res)):  
 if float(res[i][0])==h\_val:  
 pval=cum/total  
 if float(res[i][0])<h\_val and float(res[i+1][0])>h\_val:  
 ratio=(float(res[i+1][0])-h\_val)/(float(res[i+1][0])-float(res[i][0]))  
 pval=ratio\*(cum/total-(cum-res[i][1])/total)+(cum-res[i][1])/total  
 break  
 cum -= res[i][1]  
 return pval  
def rankdata(list):  
 temp\_rank=dict()  
 for i in sorted(list):  
 if i not in temp\_rank.keys():  
 temp\_rank[i]=1  
 else:  
 temp\_rank[i]+=1  
 rank=dict();r=0  
 for i in temp\_rank.keys():  
 if temp\_rank[i]==1:  
 rank[i]=r+1;r+=1  
 else:  
 rank[i]=temp\_rank[i]\*(r+1+r+temp\_rank[i])/(2\*temp\_rank[i])  
 r+=temp\_rank[i]  
 final\_rank=[]  
 for i in list:  
 final\_rank.append(rank[i])  
 return final\_rank  
def kruskal\_wallis(\*args):  
 groups=[]  
 n\_group=len(args)  
 for i in range(n\_group):  
 groups.append(args[i])  
 len\_of\_group = []  
 total\_len=0  
 for i in range(n\_group):  
 len\_of\_group.append(len(groups[i]))  
 total\_len+=len(groups[i])  
  
 total\_data = np.concatenate(groups)  
 ranked = rankdata(total\_data)  
 rank\_count=dict()  
 for i in ranked:  
 if i not in rank\_count.keys():  
 rank\_count[i]=1  
 else:  
 rank\_count[i]+=1  
  
 temp=sum([x \*\* 3 - x for x in rank\_count.values()])  
 if len(rank\_count.keys()) < 2:  
 ties=1  
 else:  
 ties=1 - temp / (total\_len \*\* 3 - total\_len)  
  
 rs = 0; lim=0  
 for i in range(n\_group):  
 rs += pow(sum(ranked[lim:lim+len\_of\_group[i]]),2) / len\_of\_group[i]  
 lim+=len\_of\_group[i]  
  
 h = 12.0 / (total\_len \* (total\_len + 1)) \* rs - 3 \* (total\_len + 1)  
 h /= ties  
 pval=null\_ks(len\_of\_group[0],len\_of\_group[1],len\_of\_group[2],h\_val=h)  
 print("H 통계량: ", h)  
 print("P-value: ",pval)  
  
a=[56,60,60]  
b=[48,57,53]  
c=[56,60,63,54,58,52]  
kruskal\_wallis(a,b,c)

**(세 그룹 사이에 차이가 없다) vs : (세 그룹 사이에 차이가 있다)**

위 코드는 크러스칼-월리스검정에서 p값을 반환한다. 동점이 있어 순위합검정통계량을 계산할 때는 귀무분포에서 선형보간을 해서 반환한다. alpha=0.05에서 크러스칼-월리스검정을 이용해 구한 H통계량은 2.906이다. 에 대한 p-value는 0.253>0.05이므로 을 기각할 증거가 부족하다. 즉, 세 그룹 사이에 차이가 있다고 말할 수 없다. (선형보간을 원하지 않을 때는, 1번에서 작성한 귀무분포 함수에 (3,3,6)를 주면 p값들을 확인할 수 있다. P(W=>2.897)=0.254, P(W=>2.987)=0.249이므로 동일한 결과를 얻는다)