소주 한 병은 몇 ‘ml’가 적절할까?

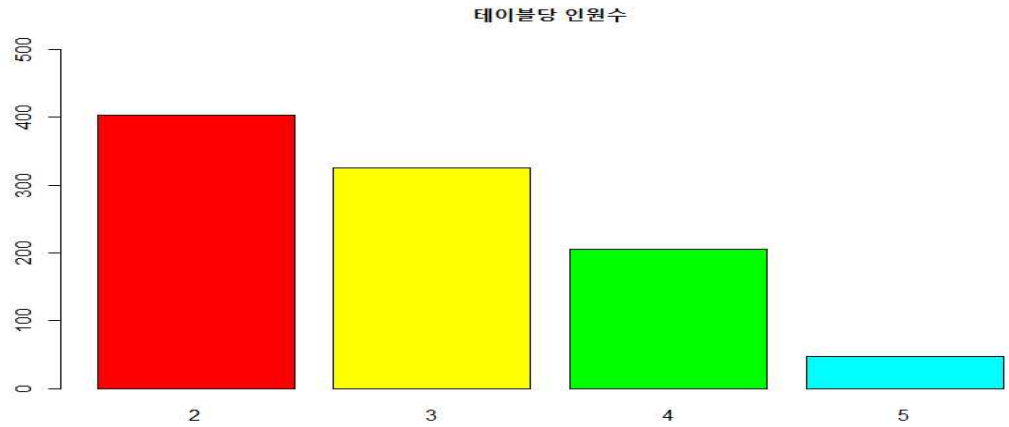
1. 개요

소주와 관련된 오래된 속설인 ‘1병당 7잔으로 설정한 것이 더 많은 매출을 올리게 한다.’ 의 진위 여부를 수학적인 방법으로 알아본다. 그리고 이 방법을 바탕으로 기업의 이윤을 현재보다 증가 시킬 수 있는 방법을 찾아본다. 현재의 소주 한 병의 용량에 대한 기업의 이윤이 최대치인지 알아본 뒤, 그렇지 않다면 소주 한 병의 용량을 어떻게 조절을 하였을 때, 기업의 이윤을 극대화되는지 알아본다.

2. 개발 내용 및 주요성과

과제를 진행하기 위해서 몇 가지 변수 및 가정을 설정하였다. 독립 변수는 테이블 당 인원수로 종속 변수는 음주량, 한 병당 나오는 잔의 수로 설정 하였다. 음주량은 한국 복지 패널에서 1인 음주량을 조사한 데이터를 찾아 활용하였다. 이 데이터들을 통해 주량의 평균과 표준편차를 구했다. 그리고 개별 음주량은 이 평균과 표준편차로 정규분포를 가질 것이라고 가정을 하였다. 계산 결과 음주량의 평균은 4.64잔, 표준편차는 2.61잔이 나왔다. 테이블 당 인원수는 사전에 조사된 정보가 존재하지 않았다. 1000개의 표본 집단이 존재하면 우리가 원하는 모집단을 대변할 수 있으므로 직접 근처 술집들을 돌아다니면서 조사를 하거나, 설문 조사를 활용해서 1000개의 테이블 당 인원수를 조사하였다.
테이블 당 인원수 중 1명인 경우는 너무 적게 나타나서 무시를 해도 무관해서 결과에 반영을 하지 않았으며, 6명이상인 단체 테이블의 경우에는 너무 많은 변수들이 존재를 해서 고려하지 않았다. 이것을 해결하기 위하여 1명인 테이블 수를 2명인 테이블 수에 6명인 테이블은 5명인 테이블 수에 합쳤다. 따라서 사용한 데이터는 다음과 같다.

	2명	3명	4명	5명
테이블 수	411	325	206	58



기본적으로 세운 가정은 사람들은 모두 자신의 주량에 맞춰서 술을 마신다. 즉 자신의 주량보다 더 적게 혹은 더 많이 마시는 경우는 없다고 한다. 그러므로 1잔을 더 마시기 위해 1병을 시키는 경우도 존재한다는 것이다.
한 병당 나오는 잔의 수에 따른 이윤의 최대를 구하기 위해서 세운 가정으로는 남은 소주의 양이 최대가 될 경우에 기업의 이윤이 최대가 된다고 가정을 하였다. 이를 구하기 위해서 python 프로그래밍을 사용하였으며, 다음 코드 및 내용은 랜덤으로 테이블 당 인원수와 음주량을 설정하고 한 병당 나오는 잔의 수를 조절해가면서 이윤이 최대가 되는 잔의 수를 구하는 것이다.

3, 코드 구현

사용한 python 라이브러리로는 pandas, numpy math, random을 사용했으며, 이후 조사한 1000개의 데이터에서 k번의 비복원추출을 통하여 k개의 가상 테이블을 설정하였다. 이 경우에는 1000개의 가상 테이블을 만들었다.

```
import pandas as pd
import numpy as np
import math
data = open('C:/Projects/keras_talk/table1.txt')
table = [int(num) for num in data.read().split()]
from random import choices
table_choice = choices(table, k=1000)
```

음주량은 사전에 조사한 1인 음주량의 데이터를 가지고 구한 평균과 표준편차를 가지는 정규분포를 따른다고 가정했었다. 이 정규분포에서 랜덤으로 뽑았을 때, 주량이 소수 값이 나오고 혹은 음수 값이 나오는 경우도 존재하게 되는데 소수 값이 나올 경우에는 올림을 해서 정수 값으로 만들어줬으며 음수일 경우는 주량이 0잔인 경우로 설정을 하였다. 다음은 앞서 k개의 가상 테이블이 있을 때, 각 테이블 별로 인원의 주량을 랜덤하게 정해주고 이를 합해 테이블 당 음주량을 구해주는 코드이다.

```
def drinking_capacity_table(n):
    a=[]
    for i in range(n) :
        b = math.ceil(np.random.normal(4.64,2.61))
        if b > 0 :
            a.append(b)
        else :
            b = 0
            a.append(b)
    return sum(a)
```

구한 테이블 당 음주량을 활용하여 소주 한 병이 6잔일 경우, 7잔일 경우 등 다양한 수를 넣어서 남는 총 소주의 양을 알아보았다. 이것을 가지고 소주 한 병을 몇 잔이 나오게 했을 때 가장 많은 양의 소주가 남게 되는지 알아보았다.

```
def conclusion(table_choice,zan):
    a = []
    for i in table_choice :
        b = drinking_capacity_table(i)
        c = zan - (b % zan)
        if c == zan :
            c = 0
        a.append(c)
    return math.ceil(sum(a)/zan)

def choose_proper(A,n):
    x = []
    for i in A :
        y = []
        for k in range(n) :
            a = conclusion(table_choice,i)
            y.append(a)
        x.append(sum(y)/n)
    print(x)
    b = max(x)
    c = A[x.index(b)]
    return b , c

choose_proper(list(range(6,16)),50)
[418.2, 431.86, 440.54, 445.68, 449.9, 449.36, 446.0, 441.36, 441.46, 441.98]
(449.9, 10)
```

다음과 같이 한 병이 6잔인 경우부터 16잔일 경우까지 넣어서 가장 남는 양이 많이 나오는 값을 보았을 때 10잔이 나오는 것을 알 수 있다. 하지만 이 과정을 여러 번 해보게 되면 항상 10잔이 아닌 11잔인 경우도 나오는 것을 알게 되었다. 그래서 이것을 해결하기 위하여 위와 같은 과정을 한 번이 아닌 여러 번 수행하도록 만들었다.

```
def choose_proper_maxzzan(A,n):
    a = []
    for i in range(n) :
        (x,y) = choose_proper(A,50)
        a.append(y)
```

```
    return a

choose_proper_maxzzan(list(range(6,16)),20)
[10, 10, 11, 11, 10, 11, 10, 11, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10]
```

다음과 같이 50개의 테이블을 가지고 20번의 과정을 반복시켜보았다. 이 경우에도 10잔인 경우가 더 많이 나오는 것을 알 수 있었다. 이후 총 1000번의 반복을 통하여 결과를 알아보았을 때에도, 10잔이 가장 많이 나오는 것을 알 수 있었다. 따라서 우리는 한 병당 10잔이 나올 경우에 남는 소주의 양이 가장 많이 나오게 되는 것으로 결론을 내렸다.

3. 및 기대효과

기업의 입장에서 남는 소주의 양이 많아질수록 이윤이 최대가 될 것이다. 이 가정을 바탕으로 내용을 진행을 하였을 때, 한 병당 10~11잔일 때 기업의 이윤이 최대가 됨을 알 수 있었다. 하지만 내용 진행을 하기 위해 잡은 가정들이 있었다. 모든 경우에 사람들은 정해진 주량을 딱 맞춰서 그리고 가득 채운 잔을 마시도록 설정을 했다. 하지만 일반적으로 잔을 가득 채워서 마시지 않고, 정해진 주량을 꼭 맞춰서 먹는 경우도 적다. 이를 통해 기업은 우리가 얻은 결과를 바탕으로 소주 한 병의 용량을 조절해서 이윤이 최대가 되는 것을 기대하기는 어렵다. 따라서 사람들의 일반적인 심리들까지 반영해서 이 과정들을 수행했다면 조금 더 유의미한 결과 값을 얻을 수 있을 것이다.