hw 7 solution

Statistical Computing, Jieun Shin

Autumn 2022

문제 1.

$n = 10$일 때, 자유도가 각 1 (코시분포), 5, 15, 10의 조합에서 신뢰구간이 참값 0을 포함하는 상대도수를 구해보면 다음과 같다. 결과를 보면 자유도가 커질수록 0.95에 가까워짐을 확인할 수 있다.

R 코드

```
set.seed(123)
N = 10000
dfs = c(1, 5,15,30)
for(i in 1:4){
  t_rep = replicate(N, expr = {
    n = 10
    x = rt(n, df = dfs[i]) # t분포
    ci = mean(x) + c(-1, 1) * qt(0.975, df = n-1) * sd(x) / sqrt(n)
    C = ifelse(ci[1] <= 0 & ci[2] >= 0, 1, 0)  # 신뢰구간이 0을 포함하면 1
    C
  })
  cat("T분포 df:", dfs[i], "상대도수:", mean(t_rep),"\n")
}
```

```
## T분포 df: 1 상대도수: 0.9791
## T분포 df: 5 상대도수: 0.9551
## T분포 df: 15 상대도수: 0.952
## T분포 df: 30 상대도수: 0.9484
```

파이썬 코드

```
import numpy as np
from numpy import random
from scipy.stats import cauchy
from scipy.stats import t
import matplotlib.pyplot as plt

N = 10000
def T_confi(df):
    n = 10

    x = t.rvs(df, loc=0, scale=1, size=n)
    ci_lower = np.mean(x) + (-1) * t.ppf(0.975,df = n-1) * np.std(x) / np.sqrt(n)
    ci_upper = np.mean(x) + (+1) * t.ppf(0.975,df = n-1) * np.std(x) / np.sqrt(n)
    if ci_lower <= 0 and ci_upper >=0:
        return 1
    else:
        return 0
```

```
dfs = [1, 5,15,30]
for df in dfs:
    print("T분포 df:", df, "상대도수:", np.mean([T_confi(df) for _ in range(N)]))
```

```
## T분포 df: 1 상대도수: 0.9744
## T분포 df: 5 상대도수: 0.943
## T분포 df: 15 상대도수: 0.9429
## T분포 df: 30 상대도수: 0.9422
```

문제 2.

표본크기와 상관계수에 따른 유의수준을 비교하면 다음과 같다. 추정된 상관계수가 0.5근처에서 나타나는데, 상관계수의 절댓값이 클수록 0.5에서 다소 멀어지는 경향을 보인다.

```
library(mvtnorm)
set.seed(123)
n = c(5, 15, 30)
rho = seq(-0.8, 0.8, 0.4)
N = 10000

result_mat = matrix(0, nrow = length(n), ncol = length(rho))
i=1;j=1
for(i in 1:length(n)){
  for(r in 1:length(rho)){
    rep = replicate(N, expr = {
        x = rmvnorm(n[i], c(0, 0), matrix(c(1, rho[r], rho[r], 1), 2, 2))
        D = x[,1] - x[,2]
        ci = mean(D) + c(-1, 1) * qt(0.975, df = n[i]-1) * sd(D) / sqrt(n[i])
        C = ifelse(ci[1] <= 0 & ci[2] >= 0, 0, 1)  # 기각할 경우 1
        C
    })
    result_mat[i, r] = mean(rep)
  }
}

colnames(result_mat) = rho
rownames(result_mat) = n
result_mat
```

```
##      -0.8   -0.4      0    0.4    0.8
## 5  0.0504 0.0489 0.0497 0.0509 0.0490
## 15 0.0464 0.0481 0.0498 0.0488 0.0501
## 30 0.0479 0.0515 0.0516 0.0556 0.0489
```

파이썬 코드

```
n = [5,15,30]
rho = np.arange(-0.8,0.81,0.4)
N = 10000

result_mat = np.zeros((len(n),len(rho)))

def MVN_confi(i,j):
    x = random.multivariate_normal(np.zeros((2)), np.array([[1,rho[j]],[rho[j],1]]),n[i])
    D = x[:,0] - x[:,1]
```

```
    ci_lower = np.mean(D) + (-1) * t.ppf(0.975,df = n[i]-1) * np.std(D) / np.sqrt(n[i])
    ci_upper = np.mean(D) + (+1) * t.ppf(0.975,df = n[i]-1) * np.std(D) / np.sqrt(n[i])
    if ci_lower <= 0 and ci_upper >=0: #기각할 경우
        return 0
    else:
        return 1


for i in range(len(n)):
    for j in range(len(rho)):
        result_mat[i,j] = np.mean([MVN_confi(i,j) for _ in range(N)])


print(result_mat)
```

```
## [[0.0665 0.0682 0.0652 0.07   0.0705]
##  [0.0616 0.0592 0.0564 0.0604 0.0532]
##  [0.0535 0.0531 0.0524 0.0578 0.0514]]
```