1회차 파이통!

기초통계학 스터디 정리자료

10 연관성 분석

01 연관성 분석의 기본

- `연관성 분석`: 조사 대상에서 수집한 자료의 척도를 기준으로 변수들 간에 어느 정도 밀접한 관계가 있는지를 판단하기 위한 분석 방법.
    - 자료의 척도를 기준으로 연관성을 파악=> 척도에 따라 분석 방법도 달라짐.
- 척도에 따른 연관성 분석 구분표

구분	척도	분석 방법	기타 변수 개입 여부
상관분석	등간척도, 비율척도	편상관분석	o
		피어슨 상관분석	x
	서열척도	스치어만 서열 상관분석	-
교차분석	명목척도	교차분석	-

    - 연속형 변수들 사이의 연관성 분석법
        - 기타변수 개입O: 편상관분석
        - 기타변수 개입X: 피어슨 상관분석
    - 범주형 변수
        - 서열로 구성된 변수: 스피어만 서열 상관분석
        - 명목척도로 구성: 교차분석을 통한 검정

02 상관분석

상관분석의 개념

- 상관분석: 조사 목적에 맞게 구성된 변수들 간의 연관성을 분석하는 방법
- 상관관계 2개의 변수를 기준으로 양, 음의 방량으로 일정한 규칙이 나타나는 선형관계의 형태와 연관 정도를 수치로 나타냄
- 상관계수: 연관성을 나타내는 수치
    - 반드시 두 변수 간의 1:1관계가 수치로 나타남.

산포도

- 산포도: 2개의 변수를 x와 y의 그래프로 나타내 분포 정도를 확인 한것
- 산포도를 나타내는 지표: 분산, 표준편차, 범위, 사분위수, 백분위수
    - 이러한 수치들은 표본내 흩어진 정도-> 변수들간 연관성 알수 X
    - 따라서 연관성을 수치적으로 알기 위해 공분산솨 상관계수에 대해 알아야한다.
- 공분산: 두 가지 확률변수에 대한 흩어짐 정도가 동일한 방향인 양의 방향인지, 반대방향인 음의 방향인지를 나타내는 수치
    - X와 Y값의 단위가 다른경우 공분산 값을 둘의 표준편차의 곱으로 나누어 단위의 효과를 상쇄하여 표준화한다. 이 값이 상관계수이다.

X와 Y라는 2개의 변수 존재, 변수가 평균$(\bar{X}, \bar{Y})$로부터 떨어져 있는 정도는
$\sum(X-\bar{X})(Y-\bar{Y})$를 표본 개수로 나누어 계산하고, 이를 공분산이라 한다.

03 공분산과 상관계수

공분산

- 공분산: 두 확률변수의 흩어진 정도가 양의 방향인지 반대인 음의 방향인지를 나타내는 수치

$$\text{공분산은 } \mathrm{x}\text{와 } \mathrm{y}\text{의 변화정도 표현, 범위: } -\infty \leq Cov \leq \infty$$
$$\text{따라서 정도의 차이만 파악, 강도는 확인 불가능 하다.}$$

- 확률변수 X,Y에 대해 흩어짐의 정도가 산포도이며, 이를 분산으로 표시가능.
  - 이 두 확률변수의 공통점이 공분산이며, 이에 대한 분석을 공분산 분석이라 함.
  - 아래식에서 X에 대한 평균편차와 Y에 대한 평균편타의 곱을 모두 합하여 총 관측치의 수로 나눈 값이 공분산이다.

$$\text{공분산은 } Cov(X,Y)\text{로 표시하며 다음과 같이 표현된다.}$$
$$\begin{aligned} Cov(X,Y) &= \frac{\sum_{i=1}^{N}(X-\bar{X})(Y-\bar{Y})}{N} \\ (\text{공분산}) &= \frac{[((\text{개별}X\text{측정치})-(X\text{의평균}))*((\text{개별}Y\text{측정치})-(Y\text{의평균}))]}{(\text{조합을 이루는 개수})} \\ &= \frac{[(X\text{의평균편차})*(Y\text{의평균편차})]\text{의총합}}{(\text{조합을 이루는 개수})} \end{aligned}$$

공분산의 개념

- X와 Y가 같은 방향: 똑같이 +값이거나 -값으로 서로 대응하는 경우 공분산은 커짐.
- X와 Y가 다른 방향: 서로 +값과 -값으로 대응하는 경우 공분산은 작아진다.
- X와 Y가 일정한 규칙 없이 대응: 공분산은 0에 가까워진다.

상관계수

- 상관계수: 공분산을 표준화한 값으로, 강도를 알기 위해서.
  - 상관 계수의 범위는 -1=<Corr=<1로 하안과 상한이 고정 따라서 양과 음의 정도에 대한 파악과 함께 연관성의 강도까지 확인할 수 있다.
  - 공분산만 분석했을 때 관계를 정확히 파악하기 어렵기 때문에 이러한 문제의 극복을 위해 표준화하며 표준화된 공분산 계수를 상관계수라 한다.

$$\begin{aligned} Cov(X,Y) &= \frac{\sum_{i=1}^{N}(X-\bar{X})(Y-\bar{Y})}{N} \\ Corr(x,y) &= \frac{Cov(x,y)}{\sqrt{\frac{\sum(x_i-\bar{x})^2}{n}*\frac{\sum(y_i-\bar{y})^2}{n}}} \\ &= \frac{Cov(x,y)}{S.D(x)*S.D(y)}(\mathrm{S.D(x)}:\mathrm{x}\text{의 표준편차}, \mathrm{S.D(y)}:\mathrm{y}\text{의 표준편차}) \\ (\text{상관계수}) &= \frac{(\text{공분산})}{x\text{의표준편차}*y\text{의표준편차}}\text{공분산을 } \mathrm{x}\text{의 표준편차와 } \mathrm{y}\text{의 표준편차를 곱한 값으로 나눈 값이 상관계수이다.} \end{aligned}$$

상관계수의 개념

- x와 y가 서로 양의 상관관계: 0<p(x,y)=<1의 상관계수 값을 가짐
- x와 y가 서로 음의 상관관계: -1=<p(x,y)<0의 값을 가짐
- x와 y가 일정한 규칙 없이 양,음값이 동시에 대응하면 상관계수=0

상관계수의 가설검정

- 사실관계를 나타내기 위해 표본을 통해 표본상관계수(r)로 모상관계수(p)를 추정하기 위해 추가분석을 하는 과정을 상관계수의 가설 검정이라 한다.

가설 수립

- 상관계수는 0으로 갈수록 상관관계가 0,-1이나 1로 갈수록 연관성 높다.

$$H_0 : p = 0 => \text{연관성이없다.}$$
$$H_1 : p \neq 0 => \text{연관성이있다.}$$

검정통계량

- 상관계수의 검정통계량은 t분포를 이용 (분산을 알지 못하는 상황에서 상관계수의 평균에 대한 표본분포를 확인하기 때문)

$$t_{(n-2)} = \frac{r}{\sqrt{\frac{1-r^2}{n-2}}} = r\sqrt{\frac{n-2}{1-r^2}}$$

  검정통계량 t값이 임계치보다 작으면 귀무가설을 채택, 임계치보다 크면 귀무가설을 기각하고 대립가설을 채택.

예제

$$H_0 : p = 0 => \text{연관성이없다.}$$
$$H_1 : p \neq 0 => \text{연관성이있다.}$$

n=28, 임계치= 2.0484=> t값이 임계치보다 작으면 귀무가설을 채택, 크면 대립가설 채택

$$t_{(28)} = 0.888\sqrt{\frac{28}{1-0.888^2}} = 10.218$$

임계치보다 검정통계량이 크므로 귀무가설을 기각하고 대립가설을 채택, 연관성이 있다.

04 교차분석

교차분석과 카이제곱 검정

- 교차분석: 범주형 척도로 구성된 자료들 간의 연관관계를 확인하기 위해 교차표를 만들어 관계를 분석하는 방법
  - 변수들의 빈도를 확인, 빈도를 이용하여 상호 연관성을 판단
  - 검정통계량으로 카이제곱 검정을 이용.

교차표

- 교차표: 2개의 조사 요인에 대한 자료값을 각각 행과 열로 배열하여 교차되는 항목에 대한 빈도를 나타낸 표

관측빈도와 기대빈도

- 관측빈도: 실제로 수집된 데이터의 빈도 O_ij로 표기
- 기대빈도: 전체 빈도 n에 대하여 행과 열의 합을 기준으로 각 교차되는 셀에 몇번의 빈도가 확인될 수 있을지를 예상하는 기대값.

$$E_{ij} = \frac{n_i * n_j}{n} (E_{ij} = \frac{n_i}{n}, n_j = \text{열의 빈도})$$

카이제곱 통계량

$$\chi^2 = \sum \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

- 관측빈도와 기대빈도 편차의 제곱을 기대빈도로 나눈값의 합

카이제곱분포의 자유도

$$\text{카이제곱 검정은 범주형 변수를 대상으로 연관성을 판단}$$
$$d.f(\text{자유도}) = k - 1(k = \text{범주형 변수의 수})$$

- 자유도와 유의수준을 기준으로 작으면 채택, 크면 기각한다.

적합도 검정

- 양자택일의 기대빈도는 50:50으로 예상이 가능하다
- 기대빈도와 관측빈도의 차이가 적으면 적을수록 적합한 기대.
- 카이제곱 분포를 이용하여 차이가 있는지 없는지를 검정할 수 있다.
- ex)

$$H_0 : \text{바다에 대한 선호도}=\text{산에 대한 선호도}$$
$$H_1 : \text{바다에 대한 선호도} \neq \text{산에 대한 선호도}$$
$$\chi^2 = \sum \frac{(O_{ij} - E_{ij})^2}{E_{ij}} = \frac{(68-50)^2}{50} + \frac{(32-50)^2}{50} = 12.96$$

  임계치는 3.8이므로 검정값이 임계치보다 큰 기각역에 속하여 귀무가설을 기각, 대립가설을 채택한다.

독립성 검정

- 독립성 검정: 여러가지 범주를 대상으로 각 범주가 독립적인지를 판단하는 검정법.
- 독립성 검정에서의 자유도는 다음과 같다.

$$d.f = (R-1)(C-1)\ (\text{R:행의 수}, \text{C:열의 수})$$

- ex)

$$H_0 : \text{지역과 구매 의사는 독립적이다.}$$
$$H_1 : \text{지역과 구매 의사는 독립적이지 않다.}$$
$$d.f = (2-1)(2-1) = 1$$
$$\chi^2 = 14.407, \alpha = 6.63$$

  카이제곱값이 임계값보다 크므로 기각역에 속하여 귀무가설을 기각하고 대립가설을 채택, 지역과 구매 의사는 독립적이지 않다.

연습문제

4. 공분산은 -11.18로 음의 상관관계를 나타낸다.
5. 상관계수는 -0.867로 음의 상관관계이다. n=23, t=8.344이므로 임계치 2.0686보다 크므로 연관성이 있다고 볼수 있다.
6. $$H_0 : \text{폴더블 선호도}=\text{롤러블 선호도}$$
   $$H_1 : \text{폴더블 선호도} \neq \text{롤러블 선호도}$$
   $$E_{ij} = 7.5, d.f = 1, \chi^2 = 0.6$$

   임계치는 3.8415로 카이제곱값보다 크다. 따라서 귀무가설을 채택하므로 둘의 선호도는 같다.
7. cell을 보자

파이썬 스터디 정리자료

파이썬 기초 실습

6장

6-1 내가 프로그램을 만들 수 있을까?

`프로그램을 만들려면 가장 먼저 '입력'과 '출력'을 생각하라.`

EX> 구구단 2단 만들기

0. 틀 만들기

```
함수이름: Gu
입력받는 값: 2 (구구단 2단 만드니까)
출력하는 값: 2단
결과 저장 형태: 연속된 자료형으로 나타나니까 리스트
```

1.
```
#Gu라는 함수에 2를 넣으면 결과가 나와야한다.
result=Gu(2)
```

2. 리스트 형으로 결과값을 받는다.
3. Gu 함수 짜기

```
def Gu(n):
    print(n)

Gu(2)
2 #입력값이 잘 출력 되는지를 확인 하는 과정이다.
```

4. 결괏값을 담을 리스트 생성하기

```
def Gu(n):
    result=[]
```

5. append함수를 사용해 리스트에 요소를 추가하는 방법도 있지만 2*n이라는 반복이 이루어 지므로 반복문을 사용하자

```
def Gu(n):
    result=[]#결괏값을 리스트 형으로 받는다.
    i=1
    while i<10:#i가 1~9까지 반복한다.
        result.append(n*i)#result= n*i의 값을 리스트로 추가한다.
        i=i+1#위의 일을 수행한 후 i에 1을 더한다.
    return result#result를 돌려준다
```

6. 마지막 테스트 해보기

```
print(Gu(2))
[2, 4, 6, 8, 10, 12, 14, 16, 18]
```

6-2 3과 5의 배수 합하기

문제:

```
10 미만의 자연수에서 3과 5의 배수를 구하면 3,5,6,9이다. 이들의 총합은 23이다. 1000미만의 자연수에서 3의 배수와 5의 배수의 총합을 구하라.
```

0. 틀짜기

```
입력받는 값: 1-999까지 (1000미만이므로)
출력하는 값: 3의 배수와 5의 배수의 총합
생각할 점
1. 3의 배수와 5의 배수를 찾기
2. 3의 배수와 5의 배수 합치기
```

1. 함수짜기

```
#1~999까지 숫자 입력하는 방법
result=0 #result값을 0으로 지정
for n in range(1,1000): #n은 1~999까지의 수 진행
    if n%3 or n%5==0: #3,5의 배수이므로 n을 3,5으로 나누면 나머지가 없다.
        result+=n #reslut값은 n의 합이다.
```

2. 결과 출력해보기

```
print(result)
365832
```

문제점

: 15의 배수는 3으로도 나뉘고 5로도 나뉘어 이중으로 값이 출력되고, 더해지지 않을 수도 있으므로, or 연산자를 사용한건데, 다음과 같이 코딩시 이중으로 더해 진다.

```
#2중으로 더해지는 문제점
result=0
for n in range(1,1000):
    if n%3==0:
        result=n
    if n%5==0:
        result+=n
print(result)
```

6-3 게시판 페이징하기

1.
```
총페이지수=(총 건수/한페이지당 보여줄 건수)+1
```

2.
```
def page(m,n): #m=게시물의 수, n=페이지당 보여 줄 게시물 수
    if m%n==0: # m/n실행후 나머지가 0이면
        return m//n #소수점을 버리는 몫을 돌려준다.
    else:
        return m//n+1
```

6-4 간단한 메모장 만들기

0. 틀짜기

```
필요한 기능: 메모장에 쓰고 읽기
입력받을 값: 내용, 실행 옵션
출력값: memo.txt
```

1. 메모 출력하는 코드

```
#C:/doit/memo.py
import sys #sys함수를 불러옴

option=sys.argv[1]
memo=sys.argv[2]
#sys.argv는 프로그램을 실행시 입력된 값을 읽어 들일 수 있는 파이썬 라이브러리
print(option)
print(memo)
```

2. memo.py저장 이후 다음명령 시행

```
C:\doit>python memo.py -a "life is too short"
-a
life is too short
```

입력 옵션과 메모 내용이 그대로 출력되는 것을 확인하기 위한 과정

3. 메모를 파일에 쓰도록 수정

```
import sys

option=sys.argv[1]

if option=='-a': #option이 -a일경우
    memo=sys.argv[2]
    f=open('memo.txt','a')#memo.txt에 파일 마지막에 새로운 내용을 추가
    f.write(memo)
    f.write('\n')
    f.close()
```

4. 입력해보기

```
C:\doit>python memo.py -a "life is too short"
C:\doit>python memo.py -a "you need python"
```

직접확인해보면 정확히 써져 있는 것을 확인 할 수 있다. 프롬프트에서 확인하려면 다음과 같이 실행해 주면 된다.

```
C:\doit>type memo.txt
    #나는 안되더라
```

5. 메모의 내용을 출력하기

```
import sys

option=sys.argv[1]

if option=='-a': #option이 -a일경우
    memo=sys.argv[2]
    f=open('memo.txt','a')#memo.txt에 파일 마지막에 새로운 내용을 추가
    f.write(memo)
    f.write('\n')
    f.close()
elif option=='-v':
    f=open('memo,txt')
    memo=f.read()
    f.close
    print(memo)
```

6. 실행해보기

```
C:\doit>python memo.py -v
#입력 내용 출력, 나는 안되더라 memo.txt가 C:/doit에 있는데
```

6-5 탭을 4개의 공백으로 바꾸기

탭을 공백 4개로 바꾸어 주는 스크립트를 작성해 보자.

```
python tabto4.py src dst
# tabto4.py는 우리가 작성해야 할 파이썬 프로그램 이름
# src는 탭을 포함하고 있는 원본 파일 이름
# dst는 파일 안의 탭을 공백 4개로 변환한 결과를 저장할 파일
ex) a.txt 파일에 있는 탭을 4개의 공백으로 바꾸어 b.txt파일에 저장하고 싶으면
python tabto4.by a.txt b.txt라 쓰면 된다.
```

1. tabto4.py파일 작성

```
#c:/doit/tabto4.py
import sys

src=sys.argv[1]
dst=sys.argv[2]

print(src)
print(dst)
#sys.argv를 사용하여 입력값을 확인하도록 만든 코드이다.
```

2. 정상 출력 확인

```
#in CMD
C:\doit> python tabto4.py a.txt b.txt
a.txt #src가 a.txt를 받아 print한 결과
b.txt #dst가 b.txt를 받아 print한 결과
```

3. 테스트 과정

```
#a.txt작성
life    is  too short
you need    python
```

4. 탭 문자 공백 4개로 변환하는 코드 짜기

```
import sys

src=sys.argv[1]
dst=sys.argv[2]

f=open(src) #src에 해당되는 파일 열기
tab_content=f.read() #tab_content가 공백을 읽은 것
f.close()

space_content=tab_content.replace("\t",""*4) #\t=tab, ""=space공백
print(space_content)
```

5. 출력 확인하기

```
python tabto4.py a.txt b.txt
life    is    too    short
you    need    python
```

6. 변경내용 b.txt에 저장하기

```
import sys

src=sys.argv[1]
dst=sys.argv[2]

f=open(src)
tab_content=f.read()
f.close()
space_content=tab_content.replace("\t"," "*4)
print(space_content)

f=open(dst,'w') #dst를 열고 w=글을 쓰기 위에 연다
f.write(space_content) #space_content내용을 적는다.
f.close
```

7. 최종 실행

```
python tabto4.py a.txt b.txt
#실행시 b.txt파일이 형성되며 4space로 나뉜 문장이 적혀 있다.
```

6-6 하위 디렉터리 검색하기

특정 디렉터리에서 하위 모든 파일 중 파이썬 파일만 출력해주는 프로그램 만들기

1.
```
#C:/doit/sub_dir_search.py

def search(dirname):
    print(dirname)

search("C:/")
#search함수를 만들고 시작 디렉터리를 입력받도록 코드를 작성
```

2. 파일 검색할 수 있도록 변경하기

```
#C:/doit/sub_dir_search.py
import os

def search(dirname):
    filenames=os.listdir(dirname)
    for filename in filenames:
        full_filename=os.path.join(dirname,filename)
        print(full_filename)

search("C:/")
```

os.listdir를 사용하면 해당 디렉터리에 있는 파일들의 리스트를 구할 수 있다.

구하는 파일 리스트는 파일 이름만 포함되어 있으므로 경로를 포함한 파일 이름을 구하기 위해 입력 받는 dirname을 앞에 붙여줌.

os 모듈에는 디렉터리와 파일 이름을 이어주는 os.path.join함수가 있으므로 이 함수를 사용하면 디렉터리를 포함한 전체 경로를 쉽게 구할 수 있다.

3. 파이썬 파일만 검색하도록 변경하기

```
#C:/doit/sub_dir_search.py
import os

def search(dirname):
    filenames=os.listdir(dirname)
    for filename in filenames:
        full_filename=os.path.join(dirname,filename)
        ext=os.path.splitext(full_filename)[-1]
        if ext=='.py':
            print(full_filename)

search("C:/")
```

   - 파일 이름에서 확장자만 추출하기 위해 사용하는 함수로 os.path.splitext는 파일 확장자를 기준으로 두 분분으로 나누어 주기 때문에 (full_filename)[-1]은 해당 파일의 확장자 이름이 된다.

4. 하위 디렉터리에서도 검색이 가능하도록 변경하기

```
#C:/doit/sub_dir_search.py
import os

def search(dirname):
    try:
        filenames=os.listdir(dirname)
        for filename in filenames:
```

```
            full_filename=os.path.join(dirname,filename)
            if os.path.isdir(full_filename):
                search(full_filename)
            else:
                ext=os.path.splitext(full_filename)[-1]
                if ext=='.py':
                    print(full_filename)
    except PermissionError:
        pass

search("C:/")
```

- try...except PermissionError로 함수 전체를 감싸면 os.listdir를 수행시 권한이 없는 디렉터리에 접근해도 그냥 수행되도록 하기 위해서
- os.path.isdir= full_filename이 디렉터리인지 파일인지 구분
  - 디렉터리 일경우=> search
  - 아닐경우-> .py파일을 찾아서 출력한다.

쉽게 만들기

os.walk= 시작 디렉터리부터 시작하여 그 하위 모든 디렉터리를 차례대로 방문하는 함수

```
import os

for(path,dir,files)in os.walk("C:/"):
    for filenane in files:
        ext=os.path.splitext(filename)[-1]
        if ext=='.py':
            print("%s/%s"%(path,filename))
```