실험계획법 기말 프로젝트 레포트

‘기후상태와 음악 청취 간의 상관관계’

12182127 최상진
12183570 전동인
12190616 남윤호
12203125 김선호

목차

I. 배경

‘비가 오면 실제로 비와 관련된 노래를 많이 들을까’ 라는 단순한 생각에서 시작하게 되었다. 카페나 백화점에 방문하였을 때도 특정 기후에서 그와 관련된 노래를 듣고 이로 인해 사람들의 감수성도 자극이 되는 모습도 적지 않게 볼 수 있다. 이렇듯 노래는 사람들을 모으고 그 공간에서도 오래 머물게 하는 등 단순 배경 음악으로서의 용도를 넘어 상황에서 적절한 음악 선정은 좋은 마케팅 효과도 가지고 온다. 그렇다면 기후 상태에 따라 사람들의 음악 선호는 변화할까? 이런 의문에서 해당 연구를 진행하게 되었다.

음악을 크게 분류한다면 그것은 장르에 의한 분류일 것이다. 소위 그 음악이 가진 분위기, 구성, 사용된 악기의 종류 등을 포괄적으로 고려하여 분류할 수 있고 새로운 음악을 도전해보고 싶을 때 유사한 장르의 음악을 찾아본다면 긍정적인 인상을 가질 확률이 높다. 사람마다 하나의 장르만 좋아하기 보다는 여러 장르를 포괄적으로 선호하는 경우가 많고 상황이나 시기에 따라 듣고 싶은 장르를 취사선택하여 듣는 경우가 대부분이다.

현재 다양한 음원 스트리밍 사이트가 존재하고 음원 자체도 너무 많이 존재한다. 이런 스트리밍 사이트를 이용하게 되면 각 음원에 대한 순위를 볼 수 있는데 보통 100위에서 200위까지의 순위로 형성되어 있다. 사람들이 어떤 음악을 많이 듣는지는 쉽게 확인할 수 있다. 보통 상위권 (1~20위 정도)은 해당 시기의 인기있는 아티스트의 음원이나 영화 혹은 드라마에 나온, 즉 누구나 아는 소위 ‘유행인 음악’ 들이 대거 포진되어 있다. 하지만 순위가 전부 해당 시기의 인기로만 형성되었다고는 보기 힘들다. 가령 ‘벚꽃엔딩’ 과 같이 발매한지 수년이 지난 지금도 봄이 되면 다시금 등장하는 음원도 있고 비가 오거나 날이 급격히 더워지면 등장하는 음원들도 있기 때문이다.

그렇다면 이런 날씨(기상상태, 우천, 온도) 등이 사람들의 장르에 대한 선호에 영향을 주는 요인 중 하나라고 봐도 적절한가? 흔히 비가 오면 감수성이 풍부해지고 그럴 땐 신나는 락이나 댄스보다는 서정적인 발라드 음악을 많이 듣지 않을까라는 의문을 품지만 이런 의문을 보다 통계적인 결과를 통해 확인해보고 싶었다.

II. 기대효과

사람들이 음악을 선택하는 기준은 다양한 원인에 의해 영향을 받는 것처럼 보인다. 그러한 원인에는 날씨, 시간대, 처해진 상황, 좋아하는 가수 등 여러가지가 있을 것이다. 그렇기에 전문가들은 그러한 관계를 분석한 후 음악 선곡을 활용하여 이득을 누리고 있다. 예를들어 고급 레스토랑에서는 저녁시간대에 느린 템포의 음악을 틀어 매출을 10프로 상승시키거나, 백화점 할인행사때는 빠른 템포의 음악을 틀어 고객 회전율을 10프로 가량 높이고 있다.

우리는 다양한 요인 중 직관적으로 연관성이 높을 것이라고 생각한 날씨를 요인으로 음악 청취의 관계를 분석하려했다. 흔히 비가오는 날에는 비와 관련된 음악 플레이리스트 조회수 및 검색어 상승의 현상을 목격하기도 하고 사계절에 따라 특색에 맞는 장르들이 눈에 자주 들어오기도 한다.
이렇게 현상적으로 확인되고 직관적으로도 연관성 있다고 느껴지는 것들을, 수업시간에 배운 통계기법을 활용하여 유의미한 것인지 이해해보고 싶었다. 그렇게 분석을 통해, 거시적인 차원의 활용이 아니더라도 연구한 결과를 바탕으로 개인 사업 차원에서 카페 매장음악을 선곡 등에 활용하여 매출 상승 및 고객을 유인하는 효과를 얻을 수 있지 않을까 생각했다.

III. 데이터 설명

A. 기후 데이터

특정한 날에 대한 기상정보를 얻기 위해 기상청 기상자료개방포털에서 ‘종관기상관측’ 파일을 이용하였다. 종관기상관측이란 종관규모의 날씨를 파악하기 위하여 정해진 시각에 모든 관측소에서 같은 시각에 실시하는 지상관측을 말하는데, 여러 지역 중 대한민국의 수도인 서울의 데이터를 사용하였다. 데이터는 ‘일시’, ‘평균기온(°C)’, ‘일강수량(㎜)’, ‘평균 풍속(m/s)’, ‘평균 이슬점온도(°C)’를 비롯한 58가지의 열을 가지는데, 이 중 주제와 관련이 있는 ‘일시’, ‘평균기온(°C)’, ‘일강수량(㎜)’, ‘일 최심적설(㎝)’, ‘평균 전운량(1/10)’ 총 5가지의 열만 남기고 전처리 하였다. ‘일시’는 그 기상정보를 측정한 날짜이다. ‘평균기온(°C)’은 3시간 간격으로 측정한 기온을 평균 낸 값이다. ‘일강수량(㎜)’은 하루 동안의 강수량을 평균 낸 값이다. ‘일 최심적설(㎝)’은 하루 동안 눈이 가장 많이 쌓였을 때의 깊이를 의미한다. ‘평균 전운량(1/10)’은 그 날 구름의 양을 1부터 10단계로 나타낸 것으로 기상청은 전운량 0~5.4을 ‘맑음’, 5.5~8.4을 ‘구름많음’, 8.5~10을 ‘흐림’으로 분류하고 있다.

B. 장르 데이터

해당 연구를 진행하면서 사람들의 장르에 대한 선호는 일별 기준으로 하였을 때 200위까지의 순위에서 그 장르 수를 count하는 것으로 생각하였다. 예를 들어 24년 6월 2일에 맑음인 조건에서 장르 내에서 발라드를 count했을때 10이라는 값이 나오고 이후 비가 오는 6월 5일에 발라드 수가 차이가 있을 수 있겠다는 생각에 순위 내에서 장르별 count 수를 value 값으로 이용하기로 하였다. 이를 위하여 일별 탑 200위까지 음원별 장르에 대한 데이터가 필요했는데 이를 한번에 파일로 제공하는 사이트는 없었다. 또한 일별 장르를 제공하는 사이트도 찾기 어려웠는데 ‘지니뮤직’이 이런 해당 조건을 모두 만족하는 사이트였다. 또한 멜론과 유튜브 다음으로 이용하는 서비스였기에 타 스트리밍 사이트 대비 사람들의 선호라고 이야기 할 수 있을많큼 적지 않은 사람들이 이용한다고 판단하였다.

출처 : 콘텐츠 진흥원

지니 뮤직에서 필요한 데이터를 200위까지 크롤링 하는 방식으로 진행하였다. 제목과 아티스트 크롤링은 크게 어렵지 않았지만 장르는 해당 음원의 앨범아트를 눌러 다른 페이지로 넘어가야 했기에 크롤링 과정은 크게 3단계로 구성하였다. 제목과 아티스트를 하나의 데이터프레임으로 저장하고 반복문을 통해 200위까지의 앨범아트 클릭 후 장르를 크롤링하여 추가 데이터프레임으로 저장한 뒤 마지막으로 모든 데이터프레임을 결합하여 특정일로 저장하는 과정을 통해 진행하였다.

	Title	Artist	genre
0	Butter	방탄소년단	댄스 / 가요
1	헤픈 우연	헤이즈 (He	R&B/소울 / 가요
2	상상더하기	MSG워너비	댄스 / 가요
3	Dun Dun	오마이걸 (	전체 / 가요
4	Next Level	aespa	댄스 / 가요
5	체념	MSG워너비	댄스 / 가요
6	롤린 (Rolli	브레이브걸	댄스 / 가요

결론적으로 이런 식의 파일로 저장하여 장르 데이터에 대한 파일을 구축할 수 있었다.

C. 실험데이터 구성

위에서 얻은 기후 데이터에서 랜덤으로 일자를 뽑아 각 일별 필요한 장르 데이터를 크롤링하고 장르별 값을 count하고 value로 적용하여 최종 데이터 프레임을 형성하였다. 기후 데이터를 랜덤으로 뽑을 때 몇가지 고려한 내용이 있는데 우선 크리스마스와 같은 공유일은 차트 내에서 기후 차이를 보기 힘들 정도로 캐롤이 많이 포진되어 있을 수 있기 때문에 최대한 해당 날짜는 피하고 온도와 기상 상태가 같더라고 계절성을 고려해야 하기 때문에 연도와 계절을 기준으로 하여 기후 조건에 맞는 일자를 랜덤으로 뽑았다.

IV. 가설 및 모형에 대한 설명

A. 가설 1.

$$y_{ijk} = \mu + \tau_i + \beta_j + (\tau\beta)_{ij} + \epsilon_{ijk}$$

i : 1,2,3,
j : 1,2,3,4
k : 1,2,3,4,5,6

$3^1 \times 4^1$ Factorial design

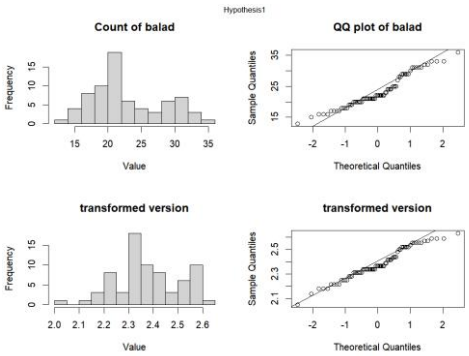
	-10~0			0~10			10~20			20~30		
맑음	2023	2021	2022									
	2018	2019	2020									
흐림												
강수												

가설1: 기온과 기후가 발라드 장르의 청취 수에 영향을 줄 것이다. (차트 top 200중 발라드의 수에 변화가 있을 것이다)

실험 방법: 교호작용까지 분석하고 싶고, 표본 수가 충분하기 때문에 라틴방격법보다 요인배치법이 적합할 것

결측치 처리: 2020년 흐림, -10~0도인 날이 없기에 NA처리하고 평균대체로 진행

정규성검정



BOX-COX 변환 후 조금 더 정규성을 띈다고 판단하여 실험진행

ANOVA

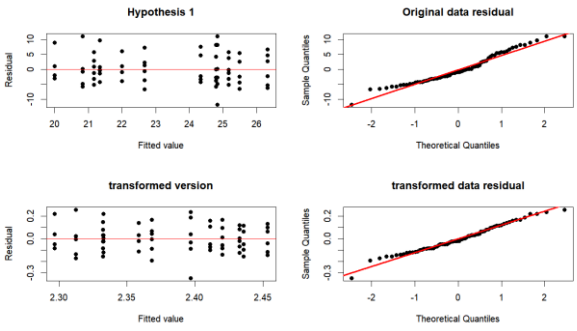
```
fit1 <- aov(value ~ weather * temp,data = final_data1)
anova(fit1)
```

```
Analysis of Variance Table

Response: value
             Df  Sum Sq  Mean Sq F value  Pr(>F)
weather       2 0.00869 0.004346  0.2706 0.76386
temp          3 0.16987 0.056625  3.5258 0.02015 *
weather:temp  6 0.00650 0.001084  0.0675 0.99871
Residuals    60 0.96361 0.016060
```

결과: 기온은 유의하지만, 기후는 유의하지 않음

잔차분석



B. 가설 2.

$$y_{ijk} = \mu + \tau_i + \beta_j + (\tau\beta)_{ij} + \epsilon_{ijk}$$

i : 1,2,3
j : 1,2,3
k : 1,2,3, 4, 5, 6

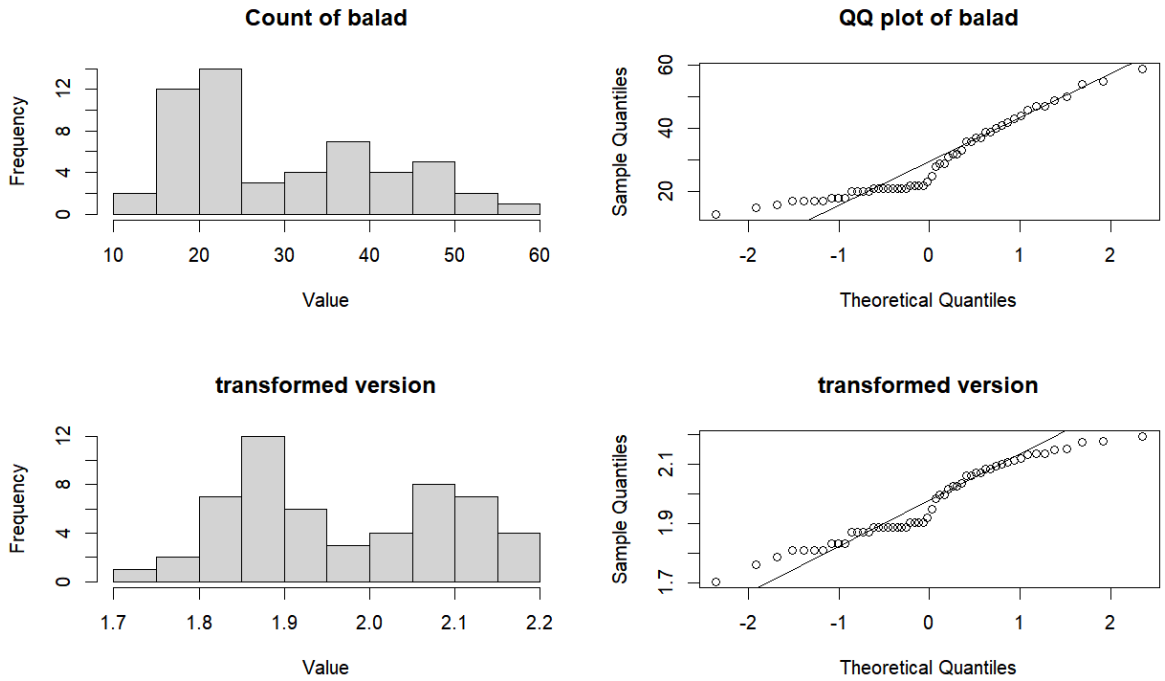
3^2 Factorial design

	0~10			10~20			20~30		
맑음	2023	2021	2022						
	2018	2019	2020						
흐림									
비									

가설2: 우천 여부와 기온이 발라드 장르의 청취 수에 영향을 줄 것이다. (차트 top 200중 발라드의 수에 변화가 있을 것이다)

실험 방법: 가설1과 동일한 요인배치법을 사용

정규성검정



BOX-COX 변환 후 조금 더 정규성을 띈다고 판단하여 실험진행

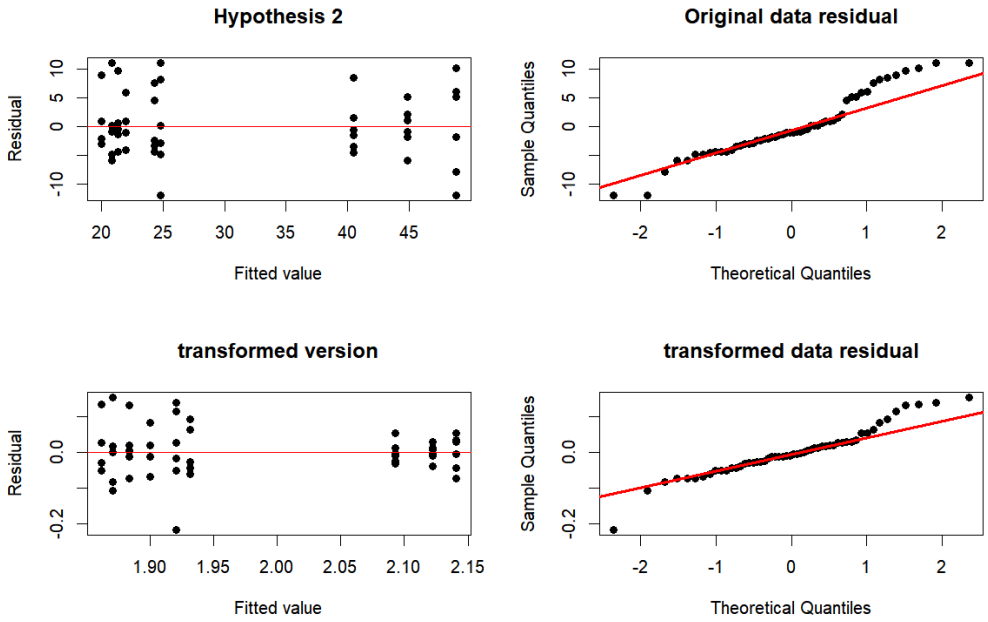
ANOVA

```
Analysis of Variance Table

Response: value
             Df  Sum Sq  Mean Sq F value    Pr(>F)
weather       2 0.60084 0.300421 58.2234 3.286e-13 ***
temp          2 0.02813 0.014067  2.7262   0.07628 .
weather:temp  4 0.00157 0.000394  0.0763   0.98911
Residuals    45 0.23219 0.005160
```

결과: 앞선 실험과는 다르게 날씨 부분의 P-VALUE값이 상당히 유의해진 것을 볼 수 있다.

잔차분석:



C. 가설 3.

$$y_{ijk} = \mu + \tau_i + \beta_j + +\epsilon_{ijk}$$

i : 1,2,3,4
j : 1,2,3,4
k : 1,2,3

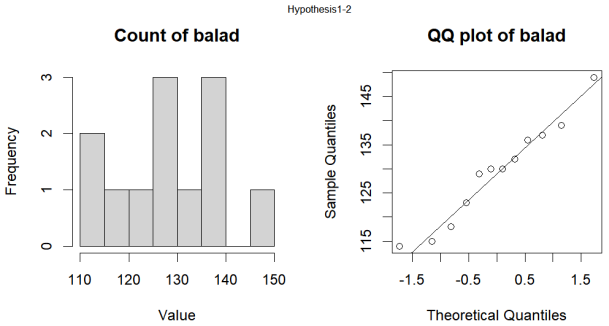
RCBD

2020	2021	2022	2023
봄 발라드 수	봄 발라드 수	봄 발라드 수	봄 발라드 수
여름 발라드 수	여름 발라드 수	여름 발라드 수	여름 발라드 수
가을 발라드 수	가을 발라드 수	가을 발라드 수	가을 발라드 수
겨울 발라드 수	겨울 발라드 수	겨울 발라드 수	겨울 발라드 수

가설3: 계절은 발라드 장르의 청취 수에 영향을 줄 것이다.

실험 방법: 년도를 블록으로 처리하여 RCBD를 사용하면 년도 간 변동을 줄일 수 있음

정규성검정



충분히 정규성을 따른다고 판단하여 추가적인 변환없이 원데이터로 실험 진행

ANOVA

```
fit1_2 <- aov(value ~ season + block,data = final_data1_2)
anova(fit1_2)

fit1_2_crbd <- aov(value ~ season, data = final_data1_2)
anova(fit1_2_crbd)
```

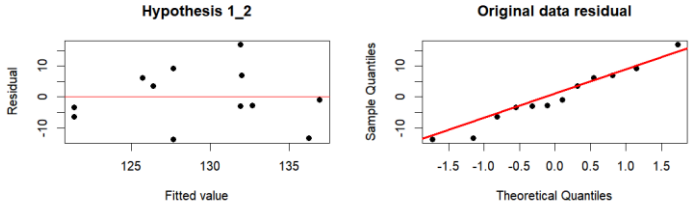
```
Analysis of Variance Table

Response: value
          Df Sum Sq Mean Sq F value Pr(>F)
season     3  66.00   22.00  0.1448 0.9293
block      2 223.17  111.58  0.7345 0.5184
Residuals  6 911.50  151.92
Analysis of Variance Table

Response: value
          Df  Sum Sq Mean Sq F value Pr(>F)
season     3   66.0   22.00  0.1551 0.9235
Residuals  8 1134.7  141.83
```

결과: p value값이 약 0.9로 유의하지 않음을 알 수 있다.

잔차분석:



D. 가설 4.

$$y_{ij} = \mu + \tau_i + \epsilon_{ij}$$

i : 1,2
j : 1,2,3,4,5,6

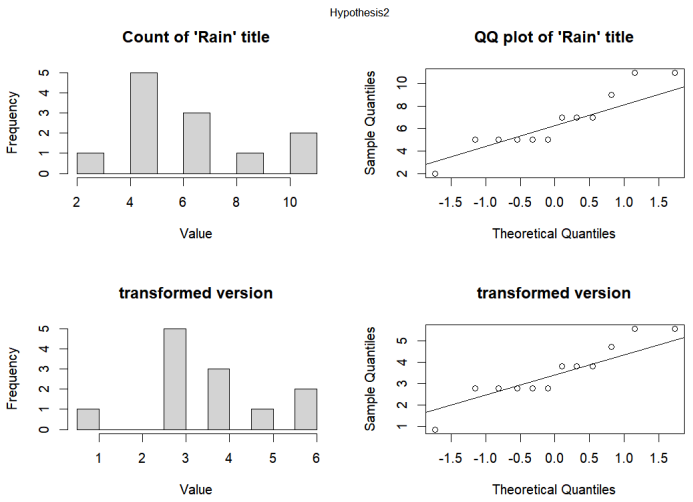
Single factor experiment

	1	2	3	4	5	6
비						
비 X						

가설4: 비가 내리는 날엔 비와 관련된 노래를 많이 들을 것이다. (차트 top 200중 '비'와 관련된 키워드를 포함한 노래 수의 변화가 있을 것이다.)

실험 방법: 요인이 강수여부 하나이므로 가장 단순한 1요인 실험 시도

정규성검정



원래의 데이터가 정규성을 많이 벗어나는 것 같아 boxcox변환을 통해 그나마 정규분포를 따르게끔 만들

었다.

ANOVA

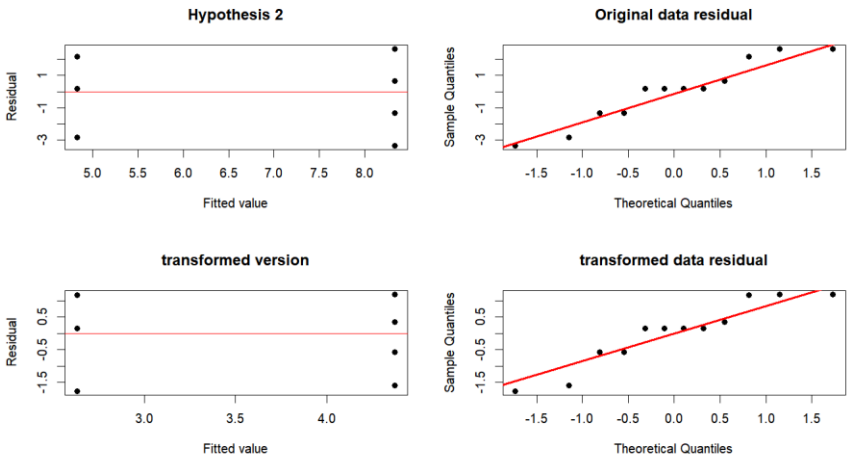
```
fit2 <- aov(value ~ rain,data = final_data2)
anova(fit2)

Analysis of Variance Table

Response: value
          Df  Sum Sq Mean Sq F value  Pr(>F)
rain       1  9.1357  9.1357  8.4907 0.01546 *
Residuals 10 10.7597  1.0760
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

결과: p value 값이 약 0.015로 $\alpha = 0.05$일 때 유의함을 확인할 수 있다.

잔차분석:



E. 가설 5.

$$y_{ijk} = \mu + \tau_i + \beta_j + (\tau\beta)_{ij} + \epsilon_{ijk}$$

i : 1,2
j : 1,2
k : 1,2,3,4,5,6

2^2 Factorial design

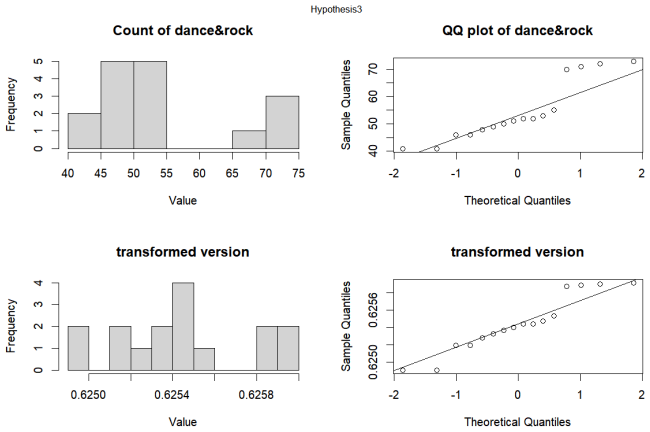
	여름			여름X		
비						
비X						

가설5: 여름이면서 비가 안 오는 날엔 댄스, 락 청취 수에 영향을 줄 것이다.

(이 실험에선 value값을 차트 top200중 댄스 장르의 수 + 락 장르의 수로 결정하였다.)

실험 방법: 여름과 여름이 아닌 조건이 추가됐음으로 2^2 Factorial design으로 변경하여 진행

정규성검정



이 데이터도 정규성을 많이 벗어나 boxcox변환을 통해 정규분포에 가깝게 만들어주었다.

ANOVA

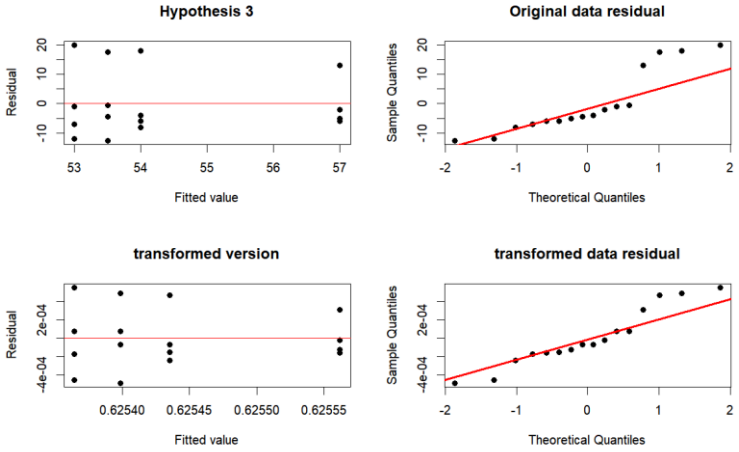
```
fit3 <- aov(value ~  rain*summer, data = final_data3)
anova(fit3)

Analysis of Variance Table

Response: value
            Df     Sum Sq    Mean Sq F value Pr(>F)
rain         1 8.5600e-09 8.5590e-09  0.0696 0.7963
summer       1 2.6000e-08 2.6003e-08  0.2116 0.6538
rain:summer  1 5.4770e-08 5.4770e-08  0.4457 0.5170
Residuals   12 1.4748e-06 1.2290e-07
```

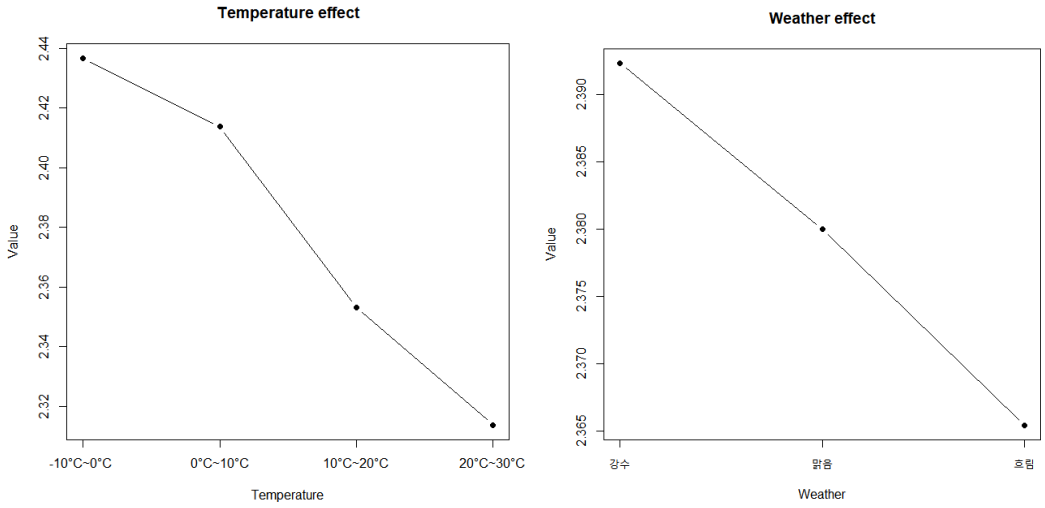
결과: rain의 p value는 약 0.8, summer의 p value는 약 0.65, 둘의 교호작용의 p value값은 약 0.5로 전부 유의하지 않음을 확인할 수 있었다.

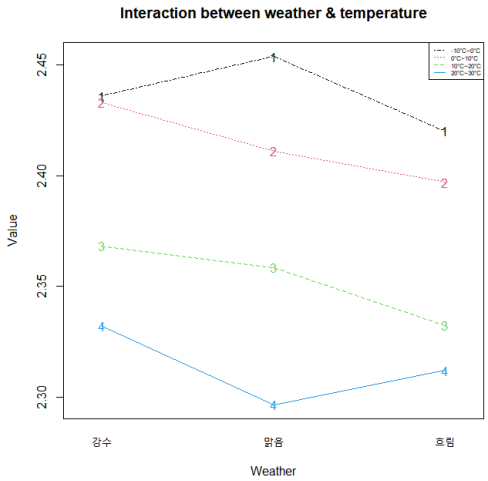
잔차분석:



V. 분석결과해석

가설1: 기온과 기후가 발라드 장르의 청취 수에 영향을 줄 것이다. (차트 top 200중 발라드의 수에 변화가 있을 것이다)

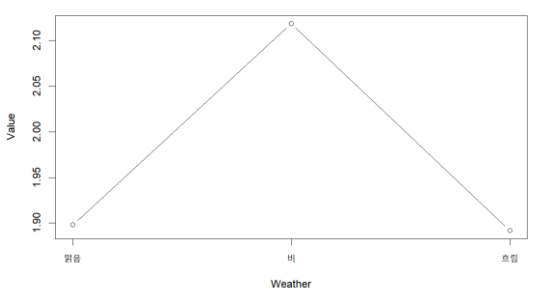


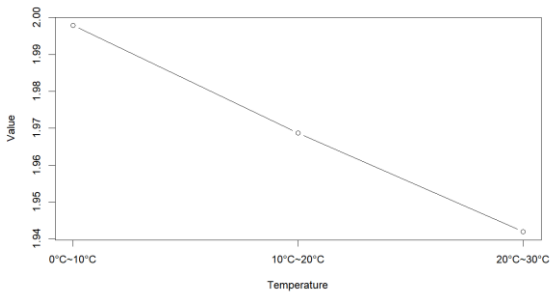


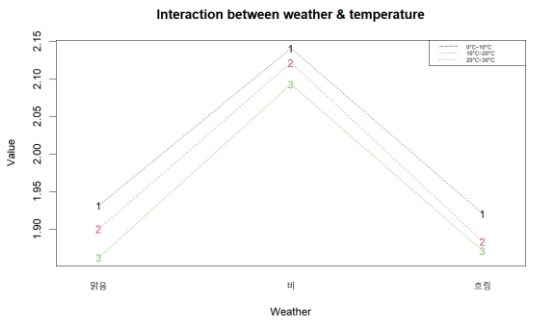
해석 : ANOVA 결과를 토대로 보았을 때 기온이 영향이 있고 기상상태가 영향이 없는 것으로 나온다. 그러나 이것을 기온만 영향이 있다고 보기는 어렵다. 날씨에서 전운량, 강수상태로 나눈 부분에서 강수상태에 비와 눈이 동시에 포함되어 있었기 때문에 비가 얼마나 영향을 주는지 추가 실험을 진행하였다.

가설2: 우천 여부와 기온이 발라드 장르의 청취 수에 영향을 줄 것이다. (차트 top 200중발라드의 수에 변화가 있을 것이다)

설정 이유 : 가설 1에서 날씨 요인을 정하는 과정에서 강수상태에 비와 눈이 전부 포함되어 있었기 때문에 순전히 비가 주는 영향을 보기 위해 진행





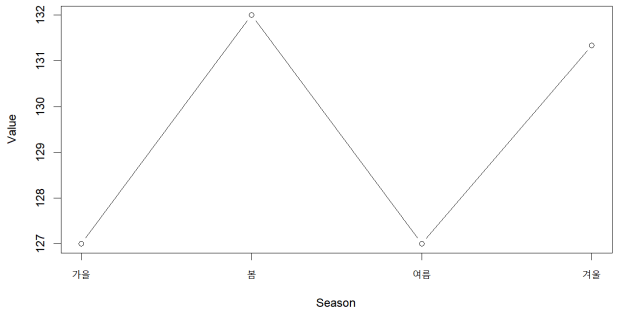


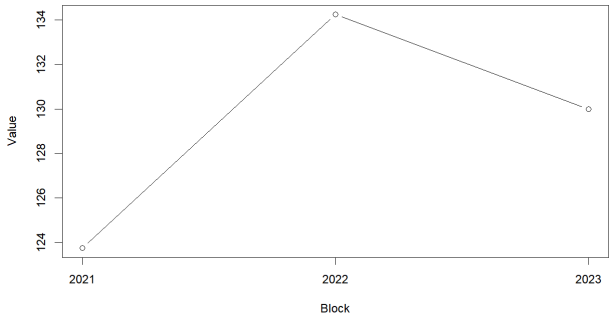
해석 : 가설 1의 실험과는 다르게 강수상태에서 눈을 제외한 순전히 비만 온 것을 가지고 다시 실험을 진행하였는데 비가 오는 상태 자체는 발라드 음악 청취에 영향을 준다고 볼 수 있었다. 즉 가설 1 실

험에서 눈과 비가 합쳐진 상태로 날씨 요인을 만들어서 유의미한 결과가 나오지 않았다고 생각할 수 있다. 온도에 대한 부분도 가설 1과 비슷하게 온도가 낮아질수록 발라드 청취수가 높아진다는 것을 볼 수 있었다. 또한 ANOVA TABLE에서 교호작용의 P-VALUE 값이 약 0.99정도로 큰 것을 확인했는데 그래프에서 역시 영향이 적은 것을 확인할 수 있었다.

가설3: 계절은 발라드 장르의 청취 수에 영향을 줄 것이다.

설정이유 : 앞선 두 실험에서 날씨가 추워질수록 발라드 청취 수에 영향을 주는 것을 확인했는데 온도는 계절성을 띄기 때문에 계절에 따른 청취 수 영향도 보고자 진행하게 되었다.

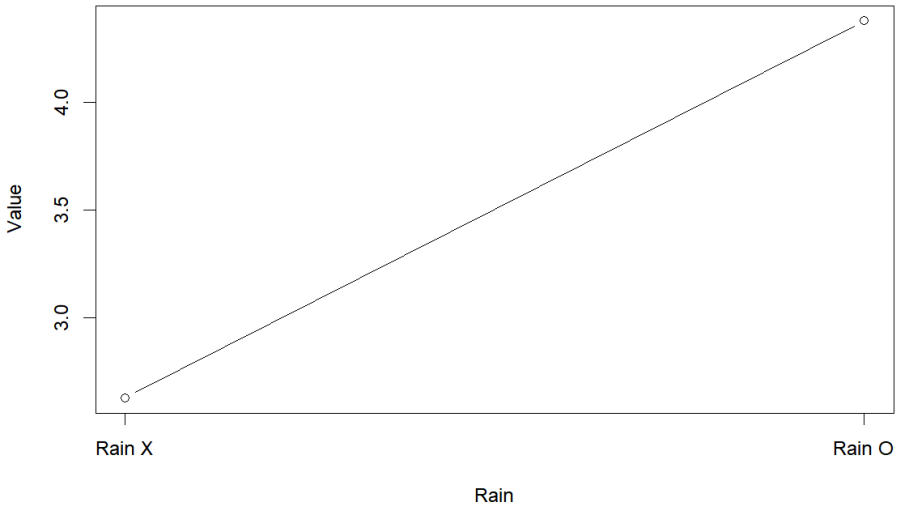




해석 : 위 실험에서 본 것처럼 온도가 상대적으로 낮은 봄과 겨울이 여름과 봄보다 영향이 더 크다는 것을 볼 수 있었다. 추가적으로 연도에 따라 블록을 했을 때 연도에 따른 차이도 볼 수 있었는데 그에 대한 이유를 추측해보기로는 한때 인기를 끌었던 트로트의 영향이라고 생각했다. 차트가 그 시기의 트랜드를 반영하기 때문에 선풍적으로 인기를 끌었던 트로트의 영향으로 생각했다. 트로트라는 장르를 뜨게 한 ‘내일은 미스트롯’과 ‘내일은 미스터 트롯’ 방영 시기의 공백기를 생각해보았을 때 미스트롯2 방영시기가 2020년 12월 ~ 2021년 3월, 미스터 트롯2의 방영시기가 2022년 12월 ~ 2023년 4월로 2022년 공백기가 생긴다. 이런 시기와 실험결과를 비교해보았을 때 연도별 차이를 트로트의 영향이라고 생각했다.

가설4: 비가 내리는 날엔 비와 관련된 노래를 많이 들을 것이다. (차트 top 200중 '비'와 관련된 키워드를 포함한 노래 수의 변화가 있을 것이다.)

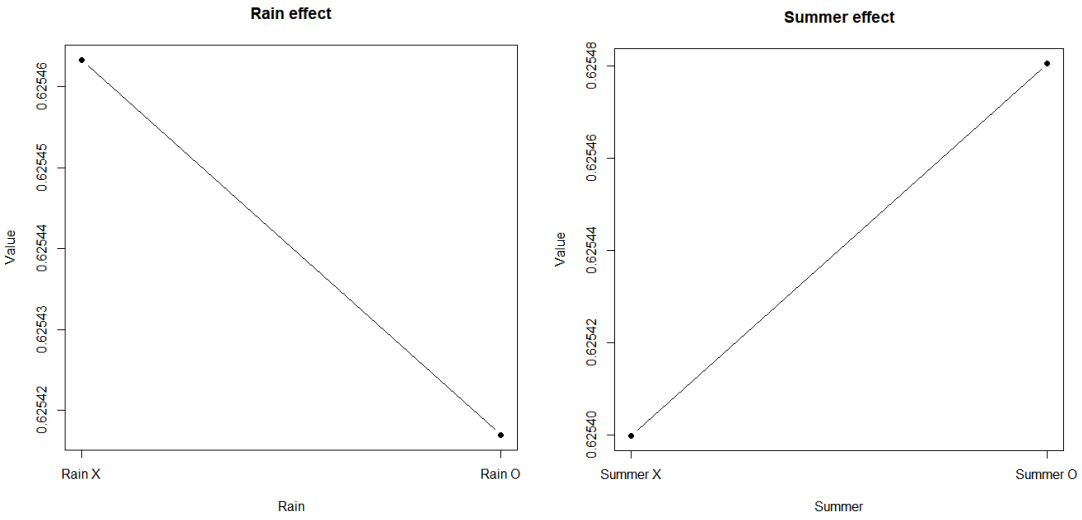
설정이유 : 앞선 실험에서 '날씨가 좋지 않을 때 잔잔한 음악을 들을 것이다' 라고 생각했는데 잔잔한 음악을 발라드 음악으로 생각해서 진행하였다. 비가 오는 상황과 그에 적합한 비와 관련된 용어가 제목에 들어간 음악 청취 수에 대한 영향을 확인해보고자 실험을 진행하였습니다.

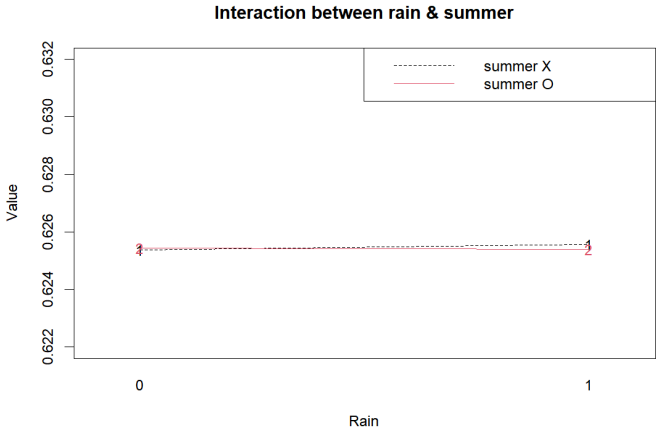


해석 : 비가 오는 날에는 제목에 비와 관련된 용어(비, 빗, 우산, rain, Rain)가 들어간 음악의 청취 수가 높아진다는 것을 확인할 수 있다.

가설5: 여름이면서 비가 안 오는 날엔 댄스, 락 청취 수에 영향을 줄 것이다.

설정이유 : 앞선 실험에선 발라드 장르에 한정하여 실험을 진행하였는데 반대로 날씨가 좋고 여름일수록 신나는 음악(댄스, 락)을 많이 듣지않을까라는 의문에서 진행하게 되었다.





해석 : 실험결과는 발라드를 활용했던 실험에 비해 유의미하다고 하기 어려웠다. 그 이유로는 현재 음원차트의 장르 구분에 있어서 신나는 음악이라고 할 음악이 실험에서 진행했던 댄스, 락과 같은 장르 이외에 구분되어 있어서 그렇다고 생각하였다. 인디, 랩/힙합 등에 장르에 흔히 신나는 음악이라고 할 음악이 포함되어 있을 수 있다. 다시 말해 신나는 음악이라는 것이 잔잔한 음악에 비해 주관적인 요소가 더 크고 아이유의 love poem이 락으로 구분되어 있는 것처럼 음원 차트의 장르 구분 자체가 애매하기 때문에 장르를 활용하여 여름이나 날씨가 좋을 때 들을 음악을 구분해내는 것은 어려울 수 있다는 생각을 하였다. 추가적인 이유로는 실험을 진행했던 연도가 2020~2023년 이었기에 코로나 영향으로 인해 여행이나 외부활동을 하며 들을 음악에 대한 수요가 적을 수도 있겠다라는 생각을 했다.

`

VI . 보완점 및 후속 연구

1. 보완점

A. 반복 수

가설3,5 실험에서 3~4개년 차트 데이터를 크롤링하여 활용했는데 이 시기에는 코로나, 트로트의 흥행 등 기후요인이 분명하게 드러나기 힘든 제한요소가 존재했다고 생각해서 확실한 영향을 위해선 더 많은 연도가 필요했다고 느꼈다. 특히 3,5 실험은 계절성에 대한 부분을 고려한 실험인데 더 많은 과거 데이터가 필요함을 느꼈다.

B. 장르의 다양성

위 실험에서는 가설 4을 제외하고는 음악을 장르에 따라 count하여 사용하였는데 target으로 삼은 음악적 특성을 장르만 활용하여 구분하기에는 특성 자체의 주관적 요인이 많아서 추가적으로 고려할 부분이 많았다. 신나는 음악을 함에 있어서 락, 댄스 이외에 추가적인 장르를 추가하거나 키워드 혹은 가사에서 나타나는 용어에 따른 접근이 음악적 특성을 활용한 실험을 하기에는 더 적합했을 것이란 생각이 든다.

2. 후속연구

A. 연도별 블록을 했을 때 기후요인 이외의 질병, 트랜드적인 요소 연도별 차이가 발생하는 것을 확인했는데 2021년~2023년까지의 트로트의 인기가 음원 차트에 준 영향’ 을 확인해보고 싶다.

B. 앞서 보완점 A에서 언급했던 연도 수에 대한 부분을 2010년 정도부터 시작하여 실험을

진행해보고 싶다.

C. 보완점 B에서 언급했던 것처럼 제목이 장르를 통한 실험보다는 더 유의미한 결과를 보였던 것으로 보아 가사의 NLP분석을 통해 나타난 기후상태를 나타낼 수 있는 키워드를 활용한 분석을 진행해보고 싶다.

VIII. 부록

코드 양이 너무 많은 관계로 항목별

- 크롤링

파이썬

- 패키지

```
import requests
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.common.by import By
import time
from tqdm.notebook import tqdm
import pandas as pd
```

- 완료 후 파일 형성

```
df['genre']=''
df2['genre']=''
df3['genre']=''
df4['genre']=''
```

```
df['genre']=genres
df2['genre']=genres2
df3['genre']=genres3
df4['genre']=genres4
```

```
df_ = pd.concat([df,df2,df3,df4], ignore_index=True)
```

```
df_.to_csv(f"{date1}.csv")
```

아티스트, 제목 추출

```
url = f"https://www.genie.co.kr/chart/top200?ditc=D&rtm=N&ymd={date1}"
url2 = f"https://www.genie.co.kr/chart/top200?ditc=D&ymd={date1}&hh=16&rtm=N&pg=2"
url3 = f"https://www.genie.co.kr/chart/top200?ditc=D&ymd={date1}&hh=16&rtm=N&pg=3"
url4 = f"https://www.genie.co.kr/chart/top200?ditc=D&ymd={date1}&hh=16&rtm=N&pg=4"


headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36"
}

response = requests.get(url, headers=headers)
if response.status_code == 200:
    soup = BeautifulSoup(response.content, "html.parser")

    track_data = []

    tracks = soup.select("#body-content > div.newest-list > div > table > tbody > tr")
    for track in tracks:
        title_element = track.select_one("a.title.ellipsis")
        artist_element = track.select_one("a.artist.ellipsis")


        if title_element and artist_element:
            title = title_element.text.strip()
            artist = artist_element.text.strip()


            track_data.append({
                "Title": title,
                "Artist": artist,

            })
        else:
            title = title_element.text.strip() if title_element else "N/A"
            artist = artist_element.text.strip() if artist_element else "N/A"


            print(f"Missing element - Title: {title}, Artist: {artist}")

    df = pd.DataFrame(track_data)
    print(df)
else:
    print(f"Failed to retrieve the webpage. Status code: {response.status_code}")


response2 = requests.get(url2, headers=headers)
if response2.status_code == 200:
    soup2 = BeautifulSoup(response2.content, "html.parser")

    track_data2 = []

    tracks2 = soup2.select("#body-content > div.newest-list > div > table > tbody > tr")
    for track2 in tracks2:
        title_element2 = track2.select_one("a.title.ellipsis")
        artist_element2 = track2.select_one("a.artist.ellipsis")
```

```
            if title_element2 and artist_element2:
                title2 = title_element2.text.strip()
                artist2 = artist_element2.text.strip()


                track_data2.append({
                    "Title": title2,
                    "Artist": artist2,

                })
            else:
                title2 = title_element2.text.strip() if title_element2 else "N/A"
                artist2 = artist_element2.text.strip() if artist_element2 else "N/A"


                print(f"Missing element - Title2: {title2}, Artist2: {artist2}")

        df2 = pd.DataFrame(track_data2)
        print(df2)
    else:
        print(f"Failed to retrieve the webpage. Status code: {response.status_code}")

    response3 = requests.get(url3, headers=headers)
    if response3.status_code == 200:
        soup3 = BeautifulSoup(response3.content, "html.parser")

        track_data3 = []

        tracks3 = soup3.select("#body-content > div.newest-list > div > table > tbody > tr")
        for track3 in tracks3:
            title_element3 = track3.select_one("a.title.ellipsis")
            artist_element3 = track3.select_one("a.artist.ellipsis")


            if title_element3 and artist_element3:
                title3 = title_element3.text.strip()
                artist3 = artist_element3.text.strip()


                track_data3.append({
                    "Title": title3,
                    "Artist": artist3,

                })
            else:
                title3 = title_element3.text.strip() if title_element3 else "N/A"
                artist3 = artist_element3.text.strip() if artist_element3 else "N/A"


                print(f"Missing element - Title3: {title3}, Artist3: {artist3}")

        df3 = pd.DataFrame(track_data3)
        print(df3)
    else:
        print(f"Failed to retrieve the webpage. Status code: {response.status_code}")

    response4 = requests.get(url4, headers=headers)
    if response4.status_code == 200:
        soup4 = BeautifulSoup(response4.content, "html.parser")

        track_data4 = []

        tracks4 = soup4.select("#body-content > div.newest-list > div > table > tbody > tr")
        for track4 in tracks4:
            title_element4 = track4.select_one("a.title.ellipsis")
            artist_element4 = track4.select_one("a.artist.ellipsis")


            if title_element4 and artist_element4:
                title4 = title_element4.text.strip()
                artist4 = artist_element4.text.strip()


                track_data4.append({
                    "Title": title4,
                    "Artist": artist4,

                })
            else:
                title4 = title_element4.text.strip() if title_element4 else "N/A"
                artist4 = artist_element4.text.strip() if artist_element4 else "N/A"


                print(f"Missing element - Title4: {title4}, Artist3: {artist4}")

        df4 = pd.DataFrame(track_data4)
        print(df4)
    else:
        print(f"Failed to retrieve the webpage. Status code: {response.status_code}")
```

장르(50개씩 크롤링하기에 4번진행)

```
driver = webdriver.Chrome()
driver.get(f'https://www.genie.co.kr/chart/top200?ditc=D&rtm=N&ymd={date1}')

genres = []
```

```
        for i in range(0, 50):
            more_info=driver.find_elements(By.CSS_SELECTOR,'.mask')
            more_info[i].click()
            time.sleep(2)
            genre=driver.find_element(By.XPATH, "//*[@id='body-content']/div[2]/div[2]/ul/li[2]/span[2]").text
            genres.append(genre)
            driver.back()
            time.sleep(2)
```

1. 전처리 코드

```
setwd('C:/Users/senio/OneDrive/바탕 화면/기상청 데이터')


library(dplyr)


#코드 실행전 각 파일의 이름을 20XX_data.csv로 변경하기

data_2018 <- read.csv("2018_data.csv", fileEncoding = "EUC-KR")

data_2019 <- read.csv("2019_data.csv", fileEncoding = "EUC-KR")

data_2020 <- read.csv("2020_data.csv", fileEncoding = "EUC-KR")

data_2021 <- read.csv("2021_data.csv", fileEncoding = "EUC-KR")

data_2022 <- read.csv("2022_data.csv", fileEncoding = "EUC-KR")

data_2023 <- read.csv("2023_data.csv", fileEncoding = "EUC-KR")


#각 년도별 데이터들 합치기

combined_data <- bind_rows(data_2018, data_2019, data_2020,

                           data_2021, data_2022, data_2023)


#필요한 열만 남기고 삭제

data <- combined_data %>%

  select(date = "일시", avg_temp = "평균기온..C.", daily_precip = "일강수량.mm.", avg_cloud = "평균.전운량.1.10.")


#write.csv(data, "weatherdata_2018-2023.csv", row.names = FALSE)


library(lubridate)


#맑음, -10°C ~ 0°C인 날짜 각 년도에서 하나씩 추출

filtered_data <- data %>%

  filter(avg_temp >= -10, avg_temp < 0, is.na(daily_precip), avg_cloud >= 0, avg_cloud < 5.5)


filtered_data <- filtered_data %>%

  mutate(year = year(date))


random_dates1 <- filtered_data %>%

  group_by(year) %>%

  slice_sample(n = 1) %>%

  ungroup() %>%

  select(date)


print(random_dates1)
```

```
#맑음, 0°C ~ 10°C인 날짜 각 년도에서 하나씩 추출
filtered_data <- data %>%
  filter(avg_temp >= 0, avg_temp < 10, is.na(daily_precip), avg_cloud >= 0, avg_cloud < 5.5)

filtered_data <- filtered_data %>%
  mutate(year = year(date))

random_dates2 <- filtered_data %>%
  group_by(year) %>%
  slice_sample(n = 1) %>%
  ungroup() %>%
  select(date)

print(random_dates2)

#맑음, 10°C ~ 20°C인 날짜 각 년도에서 하나씩 추출
filtered_data <- data %>%
  filter(avg_temp >= 10, avg_temp < 20, is.na(daily_precip), avg_cloud >= 0, avg_cloud < 5.5)

filtered_data <- filtered_data %>%
  mutate(year = year(date))

random_dates3 <- filtered_data %>%
  group_by(year) %>%
  slice_sample(n = 1) %>%
  ungroup() %>%
  select(date)

print(random_dates3)

#맑음, 20°C ~ 30°C인 날짜 각 년도에서 하나씩 추출
filtered_data <- data %>%
  filter(avg_temp >= 20, avg_temp < 30, is.na(daily_precip), avg_cloud >= 0, avg_cloud < 5.5)

filtered_data <- filtered_data %>%
  mutate(year = year(date))

random_dates4 <- filtered_data %>%
  group_by(year) %>%
  slice_sample(n = 1) %>%
  ungroup() %>%
  select(date)

print(random_dates4)

#흐림, -10°C ~ 0°C인 날짜 각 년도에서 하나씩 추출
```

```
filtered_data <- data %>%
  filter(avg_temp >= -10, avg_temp < 0, is.na(daily_precip), avg_cloud >= 5.5)

filtered_data <- filtered_data %>%
  mutate(year = year(date))

random_dates5 <- filtered_data %>%
  group_by(year) %>%
  slice_sample(n = 1) %>%
  ungroup() %>%
  select(date)

print(random_dates5)

#흐림, 0℃ ~ 10℃인 날짜 각 년도에서 하나씩 추출
filtered_data <- data %>%
  filter(avg_temp >= 0, avg_temp < 10, is.na(daily_precip), avg_cloud >= 5.5)

filtered_data <- filtered_data %>%
  mutate(year = year(date))

random_dates6 <- filtered_data %>%
  group_by(year) %>%
  slice_sample(n = 1) %>%
  ungroup() %>%
  select(date)

print(random_dates6)

#흐림, 10℃ ~ 20℃인 날짜 각 년도에서 하나씩 추출
filtered_data <- data %>%
  filter(avg_temp >= 10, avg_temp < 20, is.na(daily_precip), avg_cloud >= 5.5)

filtered_data <- filtered_data %>%
  mutate(year = year(date))

random_dates7 <- filtered_data %>%
  group_by(year) %>%
  slice_sample(n = 1) %>%
  ungroup() %>%
  select(date)

print(random_dates7)

#흐림, 20℃ ~ 30℃인 날짜 각 년도에서 하나씩 추출
filtered_data <- data %>%
```

```
  filter(avg_temp >= 20, avg_temp < 30, is.na(daily_precip), avg_cloud >= 5.5)

filtered_data <- filtered_data %>%
  mutate(year = year(date))

random_dates8 <- filtered_data %>%
  group_by(year) %>%
  slice_sample(n = 1) %>%
  ungroup() %>%
  select(date)

print(random_dates8)

#강수, -10℃ ~ 0℃인 날짜 각 년도에서 하나씩 추출
filtered_data <- data %>%
  filter(avg_temp >= -10, avg_temp < 0, daily_precip >= 0.1, avg_cloud >= 5.5)

filtered_data <- filtered_data %>%
  mutate(year = year(date))

random_dates9 <- filtered_data %>%
  group_by(year) %>%
  slice_sample(n = 1) %>%
  ungroup() %>%
  select(date)

print(random_dates9)

#강수, 0℃ ~ 10℃인 날짜 각 년도에서 하나씩 추출
filtered_data <- data %>%
  filter(avg_temp >= 0, avg_temp < 10, daily_precip >= 0.1, avg_cloud >= 5.5)

filtered_data <- filtered_data %>%
  mutate(year = year(date))

random_dates10 <- filtered_data %>%
  group_by(year) %>%
  slice_sample(n = 1) %>%
  ungroup() %>%
  select(date)

print(random_dates10)

#강수, 10℃ ~ 20℃인 날짜 각 년도에서 하나씩 추출
filtered_data <- data %>%
  filter(avg_temp >= 10, avg_temp < 20, daily_precip >= 0.1, avg_cloud >= 5.5)
```

```
filtered_data <- filtered_data %>%
  mutate(year = year(date))

random_dates11 <- filtered_data %>%
  group_by(year) %>%
  slice_sample(n = 1) %>%
  ungroup() %>%
  select(date)

print(random_dates11)

#강수, 20℃ ~ 30℃인 날짜 각 년도에서 하나씩 추출
filtered_data <- data %>%
  filter(avg_temp >= 20, avg_temp < 30, daily_precip >= 0.1, avg_cloud >= 5.5)

filtered_data <- filtered_data %>%
  mutate(year = year(date))

random_dates12 <- filtered_data %>%
  group_by(year) %>%
  slice_sample(n = 1) %>%
  ungroup() %>%
  select(date)

print(random_dates12)
```

2. 박스콕스

```
#boxcox 변수변환
boxcox_result <- boxcox(lm(final_data2$value ~ 1), lambda = seq(-2, 2, by = 0.1))
lambda <- boxcox_result$x[which.max(boxcox_result$y)]; lambda
transformed_data <- if (lambda == 0) log(final_data2$value) else (final_data2$value^lambda - 1) / lambda
final_data2$value <- transformed_data #변환된 데이터 value값으로 추가

final_data2$weather <- as.factor(final_data2$weather)
final_data2$temp <- as.factor(final_data2$temp)
```

3. 잔차분석

```
a_fit1 <- aov(value ~ weather * temp,data = data1)
fit1 <- aov(value ~ weather * temp,data = final_data1)

par(mfrow = c(2, 2))
plot(a_fit1$fitted.values,a_fit1$residuals,pch = 16,main = 'Hypothesis 1',xlab = 'Fitted value',ylab = 'Residual')
abline(h = 0, col = 'red')
qqnorm(a_fit1$residuals,pch = 16,main = 'Original data residual')
qqline(a_fit1$residuals,col = 'red',lwd = 2)
```

```
plot(fit1$fitted.values,fit1$residuals,pch = 16,main = 'transformed version',xlab = 'Fitted value',ylab = 'Residual')

abline(h = 0, col = 'red')

qqnorm(fit1$residuals, pch = 16,main = 'transformed data residual')

qqline(fit1$residuals,col = 'red',lwd = 2)
```

4. Main effect plot

```
fit1 <- aov(value ~ weather * temp,data = final_data1)


par(mfrow = c(1,1))

#main effect plot

main_effect_weather <- tapply(final_data1$value,final_data1$weather,mean)

plot(main_effect_weather,type = 'b',ylab = 'Value',xlab = 'Weather',xaxt = 'n')

axis(1,at = c(1,2,3),labels = c('강수','맑음','흐림'))


main_effect_temp <- tapply(final_data1$value,final_data1$temp,mean)

plot(main_effect_temp,type = 'b',ylab = 'Value',xlab = 'Temperature',xaxt = 'n')

axis(1,at = c(1,2,3,4),labels = c('-10℃~0℃ ','0℃~10℃','10℃~20℃','20℃~30℃'))


#interaction plot

interaction.plot(final_data1$weather,final_data1$temp,response = final_data1$value, xlab = 'Weather',ylab = 'Value', main = 'Interaction between weather & temperature',type = 'o',legend = F,col = c(1,2,3,4))

legend('topright',legend = c('-10℃~0℃','0℃~10℃','10℃~20℃','20℃~30℃'),cex = 0.5,lty = 4:1,col = c(1,2,3,4))
```