코로나19 확산에 따른 한국 넷플릭스 영화 스트리밍 소비 경향 변화

20171218 김다빈

1. 활용 데이터

- FlixPatrol 웹페이지의 “한국 월간 넷플릭스 10위 영화 콘텐츠” (flixpatrol.com)
  - ‘flixpatrol.com' 홈페이지에서 제공하는 2020.05~ 2021.06 ‘한국 넷플릭스 인기영화 10위 랭킹’에서 차트 정보와 영화 세부정보를 수집하였다. 해당 자료는 웹 크롤링을 통해 확보하였다.
  - ‘flixpatrol.com' 홈페이지의 인기영화 정보는 현 시간을 기준으로 지난 12개월의 정보만을 제공한다. 따라서 본 프로젝트는 2020.05 ~ 2021.06 기간에 한하여 데이터 수집과 가공을 진행하였다.
- KDX 한국 데이터거래소의 “코로나바이러스감염증-19(COVID-19) 현황 누적 데이터 (2021.06.12)”
  - 코로나19 바이러스 확산 정보는 KDX 한국 데이터거래소에서 제공하는 “코로나바이러스감염증-19(COVID-19) 현황 누적 데이터 (2021.06.12)” 무료자료에서 확보하였다. 해당 자료는 CSV 파일 형식으로, pandas 모듈의 read_csv 함수로 Jupyter Notebook에 불러왔다.

2. 사용 모듈

다음은 프로젝트 내에서 사용한 모듈이다. 대체로 ‘기초 빅데이터 프로그래밍’에서 학습한 모듈을 사용하였으나, 프로그램 작성 도중 봉착한 문제를 해결하기 위해 추가적인 초기설정을 더했다.

```
%matplotlib inline

import matplotlib
import matplotlib.pyplot as plt
from matplotlib import rc
import pandas as pd
import numpy as np
import re

# 시간계산을 위한 모듈
import time

# 웹 크롤링 전용 모듈
import urllib.request
from bs4 import *

# 연산시간이 오래 소요되는 메서드 진행현황 확인용 모듈
from tqdm.auto import tqdm

# MAC Matplotlib 한글 사용을 위한 초기설정
rc('font', family='AppleGothic')
plt.rcParams['axes.unicode_minus'] = False

# Matplotlib figuresize 초기설정
plt.rcParams["figure.figsize"] = (10,5)
```

- from tqdm.auto import tqdm
  - 다량의 DataFrame과 객체를 한번에 생성하여 연산시간이 오래 소요되는 메서드가 존재한다. 프로그램 실행 중 편의를 위해 해당 모듈을 사용하여 반복문의 ‘연산 진행현황’을 확인할 수 있도록 하였다.
- rc('font', family=‘AppleGothic’) …
  - MAC에서 Matplotlib 모듈로 생성한 차트 label에 한글이 깨지는 문제를 해결하기 위해 설정하였다.
- plt.rcParams["figure.figsize"] = (10,5)
  - Matplotlib 모듈로 생성한 차트의 기본 크기를 (10, 5)로 설정하였다.

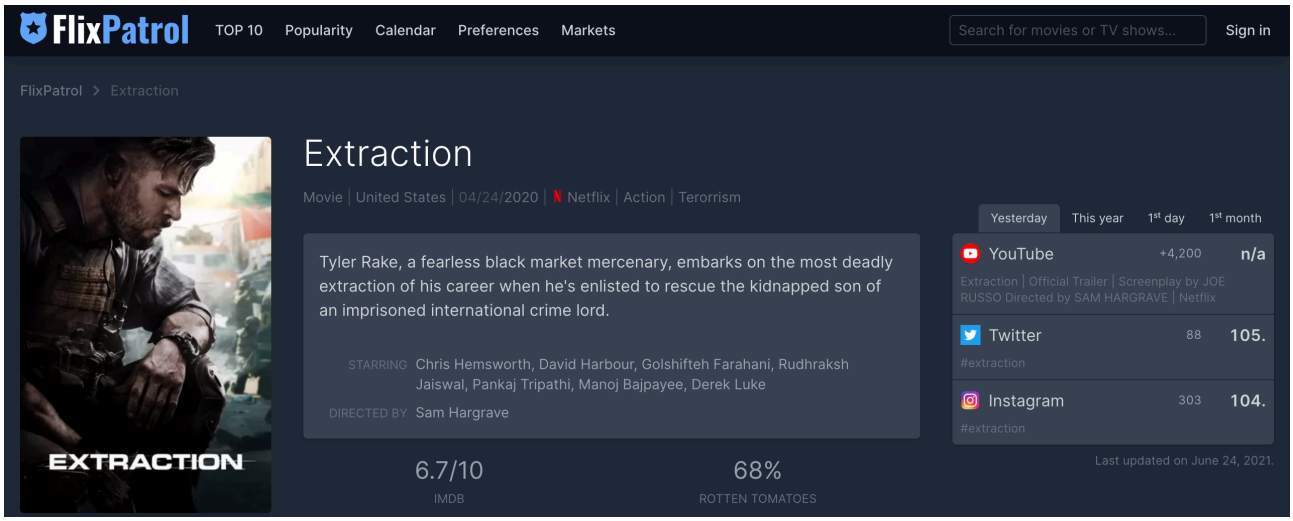
3. 데이터 수집

A. FlixPatrol 웹페이지

한국 월간 넷플릭스 10위 영화 콘텐츠 차트정보

- 위 사진은 https://flixpatrol.com/top10/netflix/south-korea/{year}-{month}/#netflix 꼴의 주소값을 지닌 웹페이지로, 다음 정보를 수집하였다.
  (1) 영화 제목
  (2) 영화 순위
  (3) 순위 정보와 시청률에 기반한 FlixPatrol Point (막대그래프 좌측 숫자정보)

넷플릭스 영화 콘텐츠 상세정보



- 위 사진은 앞서 웹페이지의 목록에서 연결되는 영화 상세정보 페이지로, 다음 정보를 수집하였다.
  (1) 제목
  (2) 기본 정보: 감독, 제작년도, IMDB 평가점수
  (3) 옵션 정보: 장르, 주제, 제작국가

- 여기에서 '옵션 정보'는 영화에 따라 나타나지 않는 경우도 있으며, 이로 인해 추출해야 할 정보가 빈번히 불규칙적인 배열로 나타난다. 이러한 옵션 정보를 안정적으로 추출하고 객체에 할당하기 위해 프로젝트 시작 전, 지난 12개월 중 10위 영화에 나타나는 모든 옵션정보를 수집했으며, 유니크한 원소로 구성된 다음 리스트를 마련하였다. 한 영화에 대한 옵션 정보는 모두 아래 클래스 변수에 대해 값 포함 여부를 확인하고 객체의 적절한 인스턴스 변수에 할당된다.

```
class MovieOptionalFeatureBox:
    # FlixPatrol 웹사이트의 영화 세부정보는 genre, subject, country 값을 불규칙하기 제공한다.
    # 일정한 형식의 틀에서 나타나는 데이터가 아니기 때문에 tag를 통해 원하는 값을 곧바로 추출하는 것이 불가능하다.
    # 따라서 'FlixPatrol_MovieInfo' 인스턴스에 대한 'optional movie info' 요소는
    # 'MovieOptionalFeatureBox' 내 세가지 리스트에 대한 포함 여부를 판단하여 할당한다.
    # 아래 데이터는 2020.06.01 부터 2021.05.31까지 한국 넷플릭스 10순위에 속한 모든 영화의
    # optional information을 중복되지 않도록 수집한 값이다.

    genreBOX = \
    ['Action','Action Crime','Action Heroes','Anime','Comedy','Documentary',
     'Drama','Fantasy','Fantasy Adventure','Horror','Independent Comics',
     'Postapocalypse','Rom Com','Romance','Science Fiction','Survival',
     'Thriller','Thrillers']
    subjectBOX = \
    ['Ancient Egypt','Ancient Greece','Buddies','Cars','Christmas','Conspiracy',
     'DC','Disapperance','Family','For Boys','For Girls','Gangsters','History',
     'Iraq War','Kung-fu','Marvel','Music','Musicians','Politics','Psychological',
     'Relationship','Revenge','Robots','Sd','Serial killer','Space','Superhero',
     'Terorrism','Treasure hunt','Vampires','War','Zombies','monster','sex']
    countryBOX = \
    ['Canada','China','France','Hong-Kong','Italy','Japan','Poland','South Korea',
     'United Kingdom']
```

B. 코로나19 현황 누적 데이터

다음은 KDX 한국 데이터거래소가 무료로 제공한 'Covid19InfState.csv' 파일이다. 본 프로젝트에서 해당 데이터호 확인하고자 하는 내용은 '코로나19 확산의 감소/증가 추세'이므로, '기준일', '누적 확진자 수', '사망자 수', '누적 확진률' 데이터만을 수집했다.

seq	stateDt	stateTime	decideCnt	clearCnt	examCnt	deathCnt	careCnt	resutlNegCnt	accExamCnt	accExamCom	accDefRate
1	20200101	18:00		1	1						
2	20200202	9:00	2	2	2	2					
3	20200203	9:00	15								
4	20200204	0:00									
5	20200205	9:00	18		129						
6	20200205	19:00	19	1	40						
7	20200206	9:00	23	1	169						
8	20200207	9:00	24	2	264						
9	20200207	16:00	24	2	327						
10	20200208	9:00	24	2	620						
11	20200208	16:00	24	2	939						
12	20200208	16:00	24	2	939						
13	20200208	16:00	24	2	939						
14	20200208	16:00	24	2	939						
15	20200209	9:00	25	3	960						
16	20200209	16:00	27	3	888						
17	20200210	9:00	27	3	809						
18	20200210	16:00	27	3	531						
19	20200211	9:00	28	4	865						
20	20200211	16:00	28	4	762						

필자는 이 프로젝트를 통해 코로나19의 감염 추세와 스트리밍 영화소비패턴 사이에 관련성이 존재하는지 확인하고자 하였다. 이를 위해 OOP 객체지향프로그래밍을 중심으로 다량의 코로나 및 영화 정보를 관리-가공했으며, 최종적으로 추세파악에 용이한 시각자료를 구현해 두 데이터를 비교분석했다.

4. 프로그램 구현 (1) - FlixPatrol 영화 데이터

해당 OOP 프로그램은 총 세 단계에 걸쳐 FlixPatrol 영화 데이터를 관리한다. 가장 하위의 FlixPatrol_MovieInfo 객체는 단 하나의 영화 상세정보를 관리한다. 중간단계에서 데이터를 관리하는 FlixPatrol_Top10 객체는 한 달의 10위 영화 정보를 다루며, 총 10개의 FlixPatrol_MovieInfo 객체를 저장한다. 마지막으로 최상위에서 지난 12개월의 모든 영화 정보를 관리하는 Annual_FlixPatrol_Top10 클래스는 총 12개의 FlixPatrol_MovieInfo 객체를 저장하며, 해당 정보를 활용해 시각자료를 출력한다.

A. 클래스 FlixPatrol_MovieInfo

* 초기화 구간

```
class FlixPatrol_MovieInfo:
    def __init__(self, movie_url):
        # basic movie info
        self._title = None
        self._director = None
        self._year = None
        self._IMDBscore = None
        # optional movie info
        self._genre = None
        self._subject = None
        self._country = None
        # movie url
        self._movie_url = movie_url
```

- 영화에 대한 상세정보는 모두 None으로 초기화하며, 영화정보를 담은 FlixPatrol 웹사이트 주소를 외부에서 할당받아 저장한다.

```
    # BASIC MOVIE INFO.
    @property
    def title(self):
        return self._title
    @property
    def director(self):
        return self._director
    @property
    def year(self):
        return self._year
    @property
    def IMDBscore(self):
        return self._IMDBscore
    # OPTIONAL MOVIE INFO.
    @property
    def genre(self):
        return self._genre
    @property
    def subject(self):
        return self._subject
    @property
    def country(self):
        return self._country
    # MOVIE URL
    @property
    def movie_url(self):
        return self._movie_url
    @movie_url.setter
    def movie_url(self, value):
        self._movie_url = value
```

- 각 인스턴스 변수는 @property 데코레이터를 활용해 핀리하게 추출할 수 있도록 하였다.

```
def read_basic_movie_info(self):
    # 영화의 감독, 제작년도, IMDB평가점수 불러오기
    html = urllib.request.urlopen(self._movie_url)
    bsObject = BeautifulSoup(html, 'html.parser')
    movieInfo = bsObject.find('div', {'class':'flex flex-wrap text-sm leading-6 text-gray-500'})

    participantsInfo = bsObject.find_all('div', {'class':'flex flex-col sm:flex-row items-baseline text-gray-400'})
    self._director = participantsInfo[-1].find_all('div')[-1].get_text()

    year_data = movieInfo.find('span', {'title':'Premiere'})
    if type(year_data) == type(None):
        # year 정보가 기입되지 않는 경우
        # ex. 'https://flixpatrol.com/title/the-gangster-the-cop-the-devil/'
        self._year = None
    else:                               # year 정보가 기입된 경우
        self._year = int(year_data.get_text()[-4:])
    self._title = bsObject.find('h1', {'class':'mb-3'}).get_text()
    IMDB = bsObject.find('div', {'class':'mb-1 text-2xl text-gray-400'}).get_text()[:-3]
    if IMDB == '':
        self._IMDBscore = 'None'
    else:
        self._IMDBscore = float(IMDB)
```

- read_basic_movie_info 인스턴스 메서드는 초기에 할당받은 주소값에 접근하여 영화의 기본 정보(제목, 감독, 제작년도, IMDB 평가점수)를 저장한다. '제작년도'와 'IMDB 평가점수'는 정보가 기록되어있지 않을 경우 각각 None과 'None'을 할당하였다.

```
def read_optional_movie_info(self):
    # 영화의 장르, 주제, 제작국가 불러오기
    html = urllib.request.urlopen(self._movie_url)
    bsObject = BeautifulSoup(html, 'html.parser')
    movieInfo = bsObject.find('div', {'class':'flex flex-wrap text-sm leading-6 text-gray-500'})

    features = [i.get_text().strip() for i in movieInfo.find_all('span')
                if re.search('[a-z]', i.get_text(), re.I)][1:]
                # Slicing을 통해 불필요한 'Movie' 정보 제외

    for feature in features:
        if feature in MovieOptionalFeatureBox.genreBOX:
            self._genre = feature
        if feature in MovieOptionalFeatureBox.subjectBOX:
            self._subject = feature
        if feature in MovieOptionalFeatureBox.countryBOX:
            self._country = feature
```

- read_optiona_movie_info 인스턴스 메서드는 할당받은 주소값에 접근하여 모든 옵션 정보를 수집하며, 각각의 내용을 MovieOptionalFeatureBox의 세가지 클래스변수에 포함 여부를 확인한 후, _genre, _subject, _country에 적절히 저장하였다.

```
def make_series(self):
    return pd.Series({'제목': self._title,
                      '감독': self._director,
                      '연도': self._year,
                      'IMDB': self._IMDBscore,
                      '장르': self._genre,
                      '주제': self._subject,
                      '국가': self._country})
```

- 저장한 정보는 make_series 인스턴스 메서드를 통해 pd.Series 형태로 반환될 수 있도록 하였다.

다음은 FlixPatrol_MovieInfo의 객체를 테스트하는 main1() 함수의 실행결과다.

```
# MAIN 1 : FlixPatrol_MovieInfo 객체 테스트
# FlixPatrol_MovieInfo 는 하나의 영화정보를 저장하는 클래스

@checkTime
def main1():
    # FlixPatrol_MovieInfo
    # 영화 하나에 대한 세부정보 불러오기

    # FlixPatrol 웹사이트에서 'We Can Be Heroes' 영화 세부정보 불러오기
    movie_info_A = FlixPatrol_MovieInfo('https://flixpatrol.com/title/we-can-be-heroes-2020/')
    movie_info_A.read_basic_movie_info()      # 영화의 감독, 제작년도, IMDB평가점수 불러오기
    movie_info_A.read_optional_movie_info()   # 영화의 장르, 주제, 제작국가 불러오기

    # 'We Can Be Heroes' 영화 세부정보 출력하기
    print(movie_info_A.title)     # 제목
    print(movie_info_A.genre)     # 장르정보 없음
    print(movie_info_A.subject)   # 주제
    print("-------------------------")

    # movie_info_A 영화정보에 대한 시리즈 반환
    print(movie_info_A.make_series())
```

```
if __name__ == "__main__":
    print("MAIN1 TEST : FlixPatrol_MovieInfo 객체 테스트\n")
    main1()
    print()
```

```
MAIN1 TEST : FlixPatrol_MovieInfo 객체 테스트

main 함수의 실행시간을 기록합니다.
We Can Be Heroes
None
Family
-------------------------
제목       We Can Be Heroes
감독       Robert Rodriguez
연도                   2020
IMDB                    4.9
장르                   None
주제                 Family
국가                   None
dtype: object
총 3.5893449783325195의 시간이 소요되었습니다.
```

B. 클래스 FlixPatrol_Top10

* 초기화 구간

```
class FlixPatrol_Top10:

    def __init__(self, year, month):
        self._year = year
        self._month = month
        self._top10 = {}
        self._base_url = 'https://flixpatrol.com'

        self._TOP10_MOVIE_INFO = []
```

- 외부로부터 year(연도)과 month(월) 값을 입력받아 저장한다.
- _top10 인스턴스 변수는 추후 10개의 차트정보를 저장하기 위해 빈 사전으로 초기화한다.
- _base_url 인스턴스 변수는 추후 10개의 FlixPatrol_MovieInfo 객체를 생성하기 위해 저장한 주소값이다.
- _TOP10_MOVIE_INFO 인스턴스 변수는 생성한 10개의 FlixPatrol_MovieInfo 객체를 저장하기 위해 빈 리스트로 초기화한다.

```
    @property
    def year(self):
        return self._year
    @year.setter
    def year(self, value):
        self._year = value
    @property
    def month(self):
        return self._month
    @month.setter
    def month(self, value):
        self._month = value
    @property
    def TOP10_MOVIE_INFO(self):
        return self._TOP10_MOVIE_INFO

    @staticmethod
    def default_url(year, month):
        return f'https://flixpatrol.com/top10/netflix/south-korea/{year}-{month:02d}/'
```

- year 과 month 값은 @property 데코레이터를 통해 쉽게 값을 할당하고 추출할 수 있도록 하였다.
- TOP10_MOVIE_INFO는 @property 데코레이터로 접근을 용이하게 하여 추후 인덱싱으로 FlixPatrol_MovieInfo 객체 값에 쉽게 접근할 수 있도록 한다.
- default_url 정적 메서드는 연도와 월 값을 입력받아 영화 차트정보가 나타나는 FlixPatrol 주소값을 반환한다.

```
    def read_top10_chart_info(self):
        # 10순위 영화 데이터 불러오기
        search_url = FlixPatrol_Top10.default_url(self._year, self._month)
        html = urllib.request.urlopen(search_url)
        bsObject = BeautifulSoup(html, 'html.parser')
        movieRankTable = bsObject.find_all('table')[0]

        for e in movieRankTable.find_all('tr', {'class':'table-group'}):
            info = [i for i in re.findall('[^\n\t]+', e.get_text()) if i.isspace()==False]
            info[2] = int(info[2])  # Point 값은 정수형(int)으로 저장
            additional_info_url = base_url + e.find('a')['href']
            info.append(additional_info_url)
            self._top10[info[0][:-1]] = info[1:]

    def load_top10_info(self):
        # 10순위 영화 데이터가 담긴 set 반환
        if self._top10 == {}:
            print('TOP 10 정보를 불러오지 않았습니다.')
        else:
            return self._top10
```

- read_top10_chart_info 는 입력받은 _yeat, _month 값을 통해 FlixPatrol 웹페이지를 크롤링한다. 차트 목록에 나타난 10개의 영화 정보와, 차트 목록을 클릭하게 접근 가능한 웹페이지 주소 10개를 _top10 사전에 등록한다.
- load_top10_info 는 불러온 영화 차트 정보를 사전(dict)의 형태로 반환한다.

```
    def show_the_Nst_movie(self, num):
        # N번째 순위 영화의 데이터 텍스트 출력
        if not (1 <= num <= 10):
            print('1과 10 사이의 정수를 입력해주세요.')
        else:
            if self._top10 == {}:
                print('TOP 10 정보를 불러오지 않았습니다.')
            else:
                movie_info = self._top10[str(num)]
                print(f'Title: {movie_info[0]}, Point: {movie_info[1]}, URL: {movie_info[2]}')
```

- show_the_Nst_movie 는 1 ~ 10 사이의 정수를 입력받은 후, 그 순위에 해당하는 영화 정보를 출력한다.

```
    def make_MovieChart_dataframe(self):
        # 'index: 영화 순위', 'columns: 정보 유형'의 DataFrame 반환
        if self._top10 == {}:
            print('TOP 10 정보를 불러오지 않았습니다.')
        else:
            print("MovieChart DataFrame을 생성하였습니다.")
            return pd.DataFrame(self._top10, index=['Title','Point','Additional_URL']).T


    def make_MovieInfo_dataframe(self):
        # TOP10_MOVIE_INFO에 저장된 FlixPatrol_MovieInfo 객체를 통해 월간 10순위 영화정보 DataFrame 반환
        # 약 30초 시간 소요
        if len(self._TOP10_MOVIE_INFO) == 0:
            print('FlixPatrol_MovieInfo 객체가 저장되어있지 않습니다.')
        else:
            DF = pd.DataFrame()
            for MovieInfo in self._TOP10_MOVIE_INFO:
                MovieInfo.read_basic_movie_info()
                MovieInfo.read_optional_movie_info()
                DF = pd.concat([DF, pd.DataFrame(MovieInfo.make_series())], axis=1)
            DF.columns = range(1,len(DF.columns)+1)
            DF.columns.name = '순위'
            DF.index.name = '상세정보'
            print("MovieInfo DataFrame을 생성하였습니다.")
        return DF.T
```

- make_MovieChart_dataframe 은 차트 목록에 나타난 10개의 영화 정보를 DataFrame으로 반환한다.
- make_MovieInfo_dataframe은 TOP10_MOVIE_INFO에 저장된 FlixPatrol_MovieInfo 객체 정보를 활용해 월간 10순위 영화 상제정보를 DataFrame으로 반환환다.

다음은 FlixPatrol_Top10의 객체를 테스트하는 main2() 함수의 실행결과다.

```
# MAIN 2 : FlixPatrol_Top10 객체 테스트
# FlixPatrol_Top10 은 한국 넷플릭스 월별 10순위에 속하는 FlixPatrol_MovieInfo 객체를 저장하는 클래스

@checkTime
def main2():
    # FlixPatrol_Top10
    # 한국 넷플릭스 월별 10순위의 '차트정보' 및 '영화 세부정보' 불러오기

    # FlixPatrol 웹사이트에서 2020년 5월 10순위 영화 '차트정보' 불러오기
    movie_top10_202005 = FlixPatrol_Top10(2020,5)  # 2020년 5월 FlixPatrol 차트 객체 생성
    movie_top10_202005.read_top10_chart_info()     # 10순위 차트정보 불러오기
    movie_top10_202005.generate_top10_MovieInfo()  # 10순위 차트정보의 10개 FlixPatrol_MovieInfo 객체 저장

    # 2020년 5월 N순위 영화 차트정보 출력
    movie_top10_202005.show_the_Nst_movie(1)
    movie_top10_202005.show_the_Nst_movie(2)
    movie_top10_202005.show_the_Nst_movie(3)
    print()

    # FlixPatrol_Top10에 저장된 개별 FlixPatrol_MovieInfo 객체 데이터 접근
    movie_top10_202005.TOP10_MOVIE_INFO[0].read_basic_movie_info()
    movie_top10_202005.TOP10_MOVIE_INFO[0].read_optional_movie_info()
    print("2020/05 1순위 영화 제목: ", movie_top10_202005.TOP10_MOVIE_INFO[0].title)
    print("2020/05 1순위 영화 장르: ", movie_top10_202005.TOP10_MOVIE_INFO[0].genre)
    print("2020/05 1순위 영화 주제: ", movie_top10_202005.TOP10_MOVIE_INFO[0].subject)
    print()

    # 2020년 5월 10순위 목록의 '차트정보' 및 '영화 세부정보' DataFrame 생성
    movie_chart_df = movie_top10_202005.make_MovieChart_dataframe()
    movie_info_df = movie_top10_202005.make_MovieInfo_dataframe()  # 약 30초 소요
```

```
if __name__ == "__main__":
#     print("MAIN1 TEST : FlixPatrol_MovieInfo 객체 테스트\n")
#     main1()
#     print()

    print("MAIN 2 : FlixPatrol_Top10 객체 테스트\n")
    main2()
    print()
```

```
MAIN 2 : FlixPatrol_Top10 객체 테스트

main 함수의 실행시간을 기록합니다.
Title: Extraction, Point: 272, URL: https://flixpatrol.com/title/extraction-2020/
Title: Time to Hunt, Point: 228, URL: https://flixpatrol.com/title/time-to-hunt/
Title: The Man Standing Next, Point: 187, URL: https://flixpatrol.com/title/the-man-standing-next/

2020/05 1순위 영화 제목:  Extraction
2020/05 1순위 영화 장르:  Action
2020/05 1순위 영화 주제:  Terorrism

MovieChart DataFrame을 생성하였습니다.
```

생성한 두가지 DataFrame (MovieChart DataFrame, MovieInfo DataFrame)의 출력결과는 다음과 같다.

```
# FlixPatrol_Top10 객체 DataFrame 생성 Test

movie_top10_202005 = FlixPatrol_Top10(2020,5)
movie_top10_202005.read_top10_chart_info()
movie_top10_202005.generate_top10_MovieInfo()

movie_chart_df = movie_top10_202005.make_MovieChart_dataframe()
movie_info_df = movie_top10_202005.make_MovieInfo_dataframe()
```

```
MovieChart DataFrame을 생성하였습니다.
MovieInfo DataFrame을 생성하였습니다.
```

MovieChart DataFrame 출력결과

```
# 2020년 5월 10순위 목록의 '차트정보' DataFrame
movie_chart_df
```

	Title	Point	Additional_URL
1	Extraction	272	https://flixpatrol.com/title/extraction-2020/
2	Time to Hunt	228	https://flixpatrol.com/title/time-to-hunt/
3	The Man Standing Next	187	https://flixpatrol.com/title/the-man-standing-...
4	Zodiac	113	https://flixpatrol.com/title/zodiac-2007/
5	Constantine	105	https://flixpatrol.com/title/constantine-2005/
6	Howls Moving Castle	101	https://flixpatrol.com/title/howls-moving-castle/
7	Fast & Furious Presents: Hobbs & Shaw	92	https://flixpatrol.com/title/fast-furious-pres...
8	The Shallows	91	https://flixpatrol.com/title/the-shallows/
9	Jason Bourne	53	https://flixpatrol.com/title/jason-bourne/
10	Spirited Away	52	https://flixpatrol.com/title/spirited-away/

MovieInfo DataFrame 출력결과

```
# 2020년 5월 10순위 목록의 '영화 세부정보' DataFrame
movie_info_df
```

상세정보 순위	제목	감독	연도	IMDB	장르	주제	국가
1	Extraction	Sam Hargrave	2020	6.7	Action	Terorrism	None
2	Time to Hunt	Yoon Sung-hyun	2020	6.3	Science Fiction	None	South Korea
3	The Man Standing Next	Woo Min-ho	2020	7.2	None	None	South Korea
4	Zodiac	David Fincher	2007	7.7	None	Serial killer	None
5	Constantine	Francis Lawrence, Michael L. Fink	2005	7	None	DC	None
6	Howls Moving Castle	Hayao Miyazaki	2004	8.2	Anime	None	Japan
7	Fast & Furious Presents: Hobbs & Shaw	David Leitch	2019	6	Action	Buddies	None
8	The Shallows	Jaume Collet-Serra	2016	6.4	Horror	None	None
9	Jason Bourne	Paul Greengrass	2016	6.8	Action	None	None
10	Spirited Away	Hayao Miyazaki	2001	8.6	None	None	Japan

C. 클래스 Annual_FlixPatrol_Top10

* 클래스 변수

```
class Annual_FlixPatrol_Top10:
    YEAR_MONTH_LIST = []
    FLIXPATROL_TOP10_LIST = []

    MOVIE_CHART_DATA_FRAMES = []
    MOVIE_INFO_DATA_FRAMES = []
```

- YEAR_MONTH_LIST 는 (year, month) 꼴로 이루어진 날짜 정보를 저장한다.
- FLIXPATROL_TOP10_LIST 는 총 12개의 FlixPatrol_Top10 객체를 날짜 정보와 함께 저장한다.
- MOVIE_CHART_DATA_FRAMES 는 12개의 FlixPatrol_Top10 객체로 생성한 MovieChart DataFrame 12가지를 저장한다.

```
    @classmethod
    def set_period(cls):
        # 프로그램을 실행하는 현재시간을 기준으로 지난 12개월을 기간으로 설정
        # (FlixPatrol 웹사이트 제공 데이터 범위 : 12개월)
        now = time.localtime()
        for month in range(now.tm_mon, 13):
            cls.YEAR_MONTH_LIST.append((now.tm_year-1, month))
        for month in range(1, now.tm_mon):
            cls.YEAR_MONTH_LIST.append((now.tm_year, month))
        print(f'{now.tm_year-1}/{now.tm_mon} 부터 {now.tm_year}/{now.tm_mon-1}까지 기간이 설정되었습니다.')
```

- set_period 클래스 메서드는 프로그램을 실행하는 현재시간을 기준으로 지난 12개월을 기간의 날짜 정보를 (year, month) 꼴로 YEAR_MONTH_LIST 클래스 변수에 저장한다.

```
    @classmethod
    def load_annual_movie_info(cls):
        if len(cls.YEAR_MONTH_LIST) == 0:
            print("set_period 메서드로 기간을 설정해주세요.")
        elif len(cls.YEAR_MONTH_LIST) > 12:
            print("이미 지난 12개월의 넷플릭스 Top10 영화정보가 저장되어 있습니다.")
        else:
            for year, mon in tqdm(cls.YEAR_MONTH_LIST):  # tqdm: 경과시간 확인
                month_data = FlixPatrol_Top10(year, mon)
                month_data.read_top10_chart_info()
                month_data.generate_top10_MovieInfo()
                cls.FLIXPATROL_TOP10_LIST.append((month_data, year, mon))
                # FlixPatrol_Top10 객체 및 시간정보 저장
            print("지난 12개월의 넷플릭스 Top10 영화정보를 불러왔습니다.")
```

- load_annual_movie_info 메서드는 YEAR_MONTH_LIST 에 저장된 날짜정보를 통해 총 12개의 FlixPatrol_Top10 객체를 YEAR_MONTH_LIST 클래스 변수에 저장한다.

```
    @classmethod
    def generate_MovieChart_dataframes(cls):
        if len(cls.FLIXPATROL_TOP10_LIST) == 0:
            print("load_annual_movie_info 메서드로 지난 12개월의 넷플릭스 Top10 영화정보를 불러와주세요.")
        else:
            annual_df = pd.DataFrame()
            for month_data, year, mon in tqdm(cls.FLIXPATROL_TOP10_LIST):  # tqdm: 경과시간 확인
                month_df = month_data.make_MovieChart_dataframe()
                month_df['Y/M'] = f'{year}/{mon:02d}'
                cls.MOVIE_CHART_DATA_FRAMES.append(month_df)
            print("MOVIE_CHART_DATA_FRAMES 리스트에 지난 12개월 MovieChart DataFrame을 저장하였습니다.")

    @classmethod
    def generate_MovieInfo_dataframes(cls):
        if len(cls.YEAR_MONTH_LIST) == 0:
            print("set_period 메서드로 기간을 설정해주세요.")
        else:
            annual_df = pd.DataFrame()
            for month_data, year, mon in tqdm(cls.FLIXPATROL_TOP10_LIST):  # tqdm: 경과시간 확인
                month_df = month_data.make_MovieInfo_dataframe()
                month_df['Y/M'] = f'{year}/{mon:02d}'
                cls.MOVIE_INFO_DATA_FRAMES.append(month_df)
            print("MOVIE_INFO_DATA_FRAMES 리스트에 지난 12개월 MovieInfo DataFrame을 저장하였습니다.")
```

- generate_MovieChart_dataframes 메서드는 12개의 FlixPatrol_Top10 객체로 생성한 MovieChart DataFrame 12가지를 MOVIE_CHART_DATA_FRAMES 클래스 변수에 저장한다.
- generate_MovieInfo_dataframes 메서드는 12개의 FlixPatrol_Top10 객체로 생성한 MovieInfo DataFrame 12가지를 MOVIE_INFO_DATA_FRAMES 클래스 변수에 저장한다.

```
    @classmethod
    def show_MovieChart_FlixPatrolPoint_chart(cls):
        # MovieChart의 FlixPatrol Point 값의 추세도 출력
        point_dict = {}
        for df in Annual_FlixPatrol_Top10.MOVIE_CHART_DATA_FRAMES:
            point_dict[df.iloc[0,-1]] = df.Point.median()
        pd.Series(point_dict).plot.bar(color='green')
        plt.title('한국 넷플릭스 선호 영화의 FlixPatrol 포인트 추세')
```

- show_MovieChart_FlixPatrolPoint_chart 메서드는 MOVIE_CHART_DATA_FRAMES에 저장된 MovieChart DataFrame을 활용하여 '한국 넷플릭스 선호 영화의 FlixPatrol 포인트 추세도' 시각자료를 출력한다.

```
    @classmethod
    def show_MovieInfo_feature_chart(cls, feature, cut=1):
        if feature in ['연도','주제','장르','국가']:
            total_columns = [df.iloc[0,-1] for df in Annual_FlixPatrol_Top10.MOVIE_INFO_DATA_FRAMES
            total_dicts = []
            for df in Annual_FlixPatrol_Top10.MOVIE_INFO_DATA_FRAMES:
                genre_dict = {}
                for i, j in df[feature].value_counts().items():
                    genre_dict[i] = j
                total_dicts.append(genre_dict)

            # DataFrame 생성을 위한 dict 생성
            raw_data = {col: {} for col in total_columns}
            for i, col in enumerate(total_columns):
                raw_data[col] = total_dicts[i]

            df = pd.DataFrame(raw_data).T
            for column in df.columns:
                if len(df[column].dropna()) <= cut:
                    # 12개월 중 n회 이하로 등장한 주제 삭제
                    df = df.drop([column], axis=1)
            df.plot.bar(stacked=True)
            plt.title(f"한국 넷플릭스 선호 영화의 {feature} 빈도 추세", size=15)
        else:
            print("'연도', '주제', '장르', '국가' 중 하나의 요소를 입력해주세요.")
```

- show_MovieInfo_feature_chart 메서드는 MOVIE_INFO_DATA_FRAMES에 저장된 MovieInfo DataFrame을 활용하여 시각자료를 출력한다.
- '연도', '주제', '장르', '국가' 중 한 요소를 입력해여 해당 정보에 대한 Stacked Bar 차트를 확인할 수 있으며, cut=n 값을 추가적으로 할당해 12개월 중 n회 이하로 나타난 주제를 삭제하여 시각화할 수 있다.

다음은 Annual_FlixPatrol_Top10 클래스 메서드를 테스트하는 main3() 함수의 실행결과다.

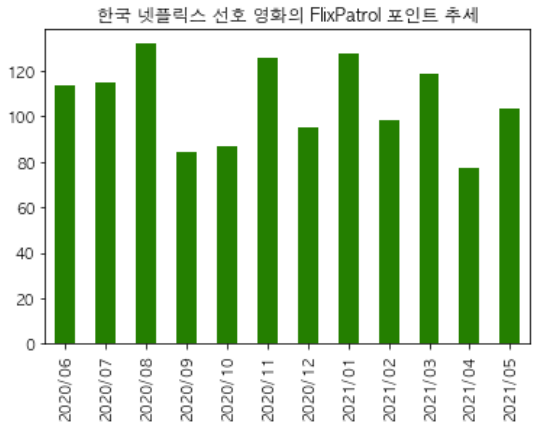
```
# MAIN 3 : Annual_FlixPatrol_Top10 클래스 테스트
# Annual_FlixPatrol_Top10 지난 12개월의 FlixPatrol_Top10 객체를 저장하는 클래스
# 해당 프로그램에서 FlixPatrol 넷플릭스 영화정보에 대한 모든 것을 관리

@checkTime
def main3():
    # Annual_FlixPatrol_Top10
    # 지난 12개월 한국 넷플릭스 10순위의 '차트정보' 및 '영화 세부정보' 불러오기

    # 현 시간을 기준으로 지난 12개월 기간 설정
    Annual_FlixPatrol_Top10.set_period()
    Annual_FlixPatrol_Top10.load_annual_movie_info()

    # 지난 12개월 한국 넷플릭스 10순위 '차트정보' DataFrame 생성
    Annual_FlixPatrol_Top10.generate_MovieChart_dataframes()

    # 지난 12개월 한국 넷플릭스 10순위 '차트정보'의 FlixPatrol Point 라인차트 출력
    Annual_FlixPatrol_Top10.show_MovieChart_FlixPatrolPoint_chart()
```

```
if __name__ == "__main__":
#     print("MAIN1 TEST : FlixPatrol_MovieInfo 객체 테스트\n")
#     main1()
#     print()

#     print("MAIN 2 : FlixPatrol_Top10 객체 테스트\n")
#     main2()
#     print()

    print("MAIN 3 : Annual_FlixPatrol_Top10 클래스 테스트\n")
    main3()
    print()
```

show_MovieChart_FlixPatrolPoint_chart 클래스 메서드 출력결과

- 한국 넷플릭스 선호 영화의 FlixPatrol 포인트 추세도이다.
- 해당 차트를 출력하기 위해서는 generate_MovieChart_dataframes 메서드를 통해 12개의 MovieChart DataFrame을 생성해야 한다.
- 해당 시각자료는 시간에 따라 변화하는 넷플릭스 인기 영화의 'FlixPatrol 평가점수' 중간값을 보여준다.
- 높은 막대 그래프는 좋은 평가를 받은 영화를 주로 시청하는 소비패턴을 반영하며, 낮은 막대 그래프는 평가점수가 낮은 영화 시청에 도전하는 소비패턴을 보여준다.

```
MAIN 3 : Annual_FlixPatrol_Top10 클래스 테스트

main 함수의 실행시간을 기록합니다.
2020/6 부터 2021/5까지 기간이 설정되었습니다.
이미 지난 12개월의 넷플릭스 Top10 영화정보가 저장되어 있습니다.

100% 12/12 [00:00<00:00, 21.00it/s]

MovieChart DataFrame을 생성하였습니다.
MovieChart DataFrame을 생성하였습니다.
MovieChart DataFrame을 생성하였습니다.
MovieChart DataFrame을 생성하였습니다.
MovieChart DataFrame을 생성하였습니다.
MovieChart DataFrame을 생성하였습니다.
MovieChart DataFrame을 생성하였습니다.
MovieChart DataFrame을 생성하였습니다.
MovieChart DataFrame을 생성하였습니다.
MovieChart DataFrame을 생성하였습니다.
MovieChart DataFrame을 생성하였습니다.
MovieChart DataFrame을 생성하였습니다.

MOVIE_CHART_DATA_FRAMES 리스트에 지난 12개월 MovieChart DataFrame을 저장하였습니다.
총 0.3846709728240967의 시간이 소요되었습니다.
```



show_MovieInfo_feature_chart 클래스 메서드 출력결과

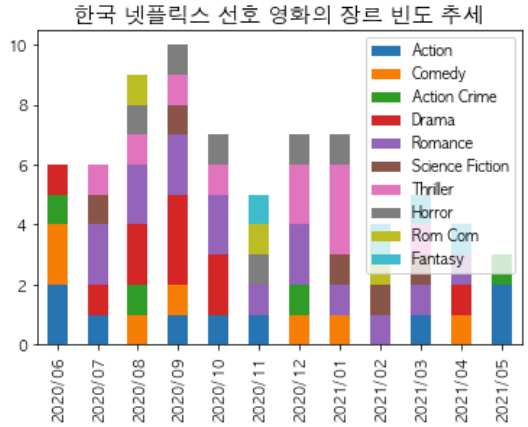
- '연도', '주제', '장르', '국가' 중 한 요소에 대한 Stacked Bar 차트를 출력한다.
- 해당 차트를 출력하기 위해서는 generate_MovieInfo_dataframes 메서드를 통해 12개의 MovieInfo DataFrame을 생성해야 한다.
- 아래 예시는 모두 cut=2 로 입력하여 12개월 중 2회 미만으로 나타난 영화정보를 무시하였다.

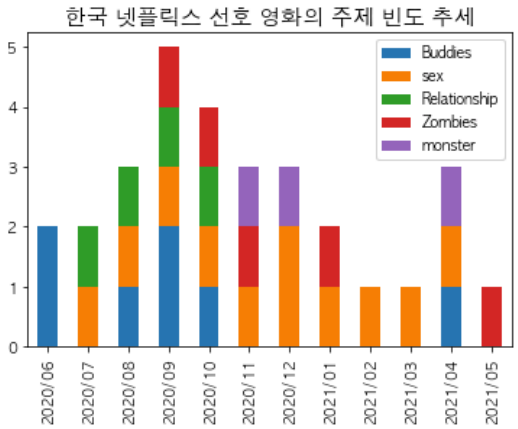
```
# MAIN 3 : Annual_FlixPatrol_Top10 클래스 테스트
# Annual_FlixPatrol_Top10 지난 12개월의 FlixPatrol_Top10 객체를 저장하는 클래스
# 해당 프로그램에서 FlixPatrol 넷플릭스 영화정보에 대한 모든 것을 관리

@checkTime
def main3():
    # Annual_FlixPatrol_Top10
    # 지난 12개월 한국 넷플릭스 10순위의 '차트정보' 및 '영화 세부정보' 불러오기

    # 현 시간을 기준으로 지난 12개월 기간 설정
    Annual_FlixPatrol_Top10.set_period()
    Annual_FlixPatrol_Top10.load_annual_movie_info()

    # 지난 12개월 한국 넷플릭스 10순위 '영화 세부정보' DataFrame 생성
    Annual_FlixPatrol_Top10.generate_MovieInfo_dataframes()

    # 지난 12개월 한국 넷플릭스 10순위 '영화 세부정보'의 stacked bar 차트 출력
    # '장르', '주제', '국가', '연도' 세부정보 선택 가능
    # cut=N 값을 설정하여 12개월 내 N개 이하의 빈도수로 나타난 값 삭제
    Annual_FlixPatrol_Top10.show_MovieInfo_feature_chart('장르', cut=2)
    Annual_FlixPatrol_Top10.show_MovieInfo_feature_chart('주제', cut=2)
    Annual_FlixPatrol_Top10.show_MovieInfo_feature_chart('국가', cut=2)
    Annual_FlixPatrol_Top10.show_MovieInfo_feature_chart('연도', cut=2)
```

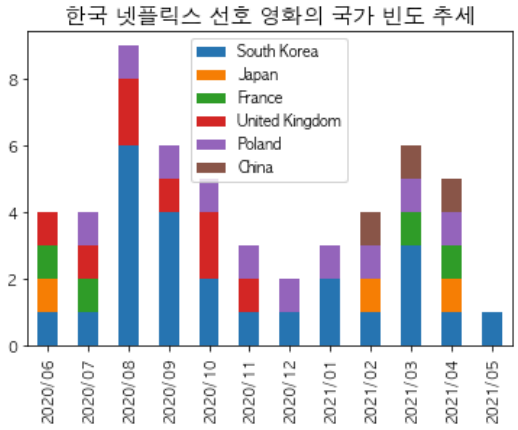
```
if __name__ == "__main__":
#     print("MAIN1 TEST : FlixPatrol_MovieInfo 객체 테스트\n")
#     main1()
#     print()

#     print("MAIN 2 : FlixPatrol_Top10 객체 테스트\n")
#     main2()
#     print()

    print("MAIN 3 : Annual_FlixPatrol_Top10 클래스 테스트\n")
    main3()
    print()
```

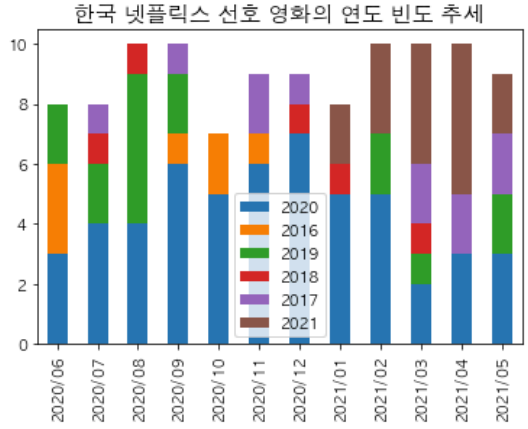
```
MAIN 3 : Annual_FlixPatrol_Top10 클래스 테스트

main 함수의 실행시간을 기록합니다.
2020/6 부터 2021/5까지 기간이 설정되었습니다.

100% ████████████████████████████████ 12/12 [06:31<00:00, 32.59s/it]


지난 12개월의 넷플릭스 Top10 영화정보를 불러왔습니다.

100% ████████████████████████████████ 12/12 [06:04<00:00, 30.38s/it]

MovieInfo DataFrame을 생성하였습니다.
MovieInfo DataFrame을 생성하였습니다.
MovieInfo DataFrame을 생성하였습니다.
MovieInfo DataFrame을 생성하였습니다.
MovieInfo DataFrame을 생성하였습니다.
MovieInfo DataFrame을 생성하였습니다.
MovieInfo DataFrame을 생성하였습니다.
MovieInfo DataFrame을 생성하였습니다.
MovieInfo DataFrame을 생성하였습니다.
MovieInfo DataFrame을 생성하였습니다.
MovieInfo DataFrame을 생성하였습니다.
MovieInfo DataFrame을 생성하였습니다.

MOVIE_INFO_DATA_FRAMES 리스트에 지난 12개월 MovieInfo DataFrame을 저장하였습니다.
총 390.03376603126526의 시간이 소요되었습니다.
```

```
MOVIE_INFO_DATA_FRAMES 리스트에 지난 12개월 MovieInfo DataFrame을 저장하였습니다.
총 390.03376603126526의 시간이 소요되었습니다.
```









5. 프로그램 구현 (2) - COVID 현황 누적 데이터 : 클래스 COVID_Data

코로나19 데이터는 COVID_Data 하나의 클래스로 관리한다. 해당 클래스는 'Covid19InfState.csv' 파일로 구현한 DataFrame을 활용하여 다양한 시각자료를 출력한다. COVID_Data가 출력할 수 있는 시각자료의 종류는 다음과 같다.

(1) '확진자 수', '사망자 수', '확진률'에 대한 개별 누적 라인차트

(2) '확진자 수', '사망자 수'에 대한 개별 변동률 라인차트 (월 별 코로나19 확산세 변화를 보다 쉽게 파악 가능)

(3) '확진자 수', '사망자 수'에 대한 누적 라인차트 동시 출력 (서로 다른 y축 척도)

(4) '확진자 수', '사망자 수'에 대한 변동률 라인차트 동시 출력 (서로 다른 y축 척도)

```
class COVID_Data:
    DataFrame = None

    @classmethod
    def read_covid_data(cls):
    # 'Covid19InfState.csv'의 decideCnt', 'deathCnt', 'accDefRate' 데이터를 DataFrame으로 읽기

        covid19_info = pd.read_csv('./Covid19InfState.csv', index_col='stateDt')
        covid19_info = pd.DataFrame(covid19_info,
                                    columns=['decideCnt', 'deathCnt', 'accDefRate']).fillna(0)[124:
        covid19_info.index = covid19_info.index.astype('str')
        covid19_info.index.name = '시간'
        covid19_info.columns = ['확진자 수', '사망자 수', '확진률']
        covid19_info.columns.name = '코로나 현황정보'
        cls.DataFrame = covid19_info
        print("COVID DataFrame을 생성하였습니다.")
```

- read_covid_data 클래스 메서드는 'Covid19InfState.csv' 파일의 decideCnt, deathCnt, accDefRate (누적 확진자 수, 누적 사망자 수, 누적 확진률) 데이터를 추출하여 DataFrame을 생성한다. 해당 자료는 클래스 변수 DataFrame에 저장된다.

```
    @classmethod
    def show_accumulated_feature_chart(cls, feature):
        # 확진자 수, 사망자 수, 확진률에 대한 누적 라인차트 출력
        if feature in ['확진자 수', '사망자 수', '확진률']:
            try:
                plt.plot(cls.DataFrame[feature])
                x = sorted(set(i[:-2]+'01' for i in cls.DataFrame.index))
                plt.xticks(x, rotation=90)
            except TypeError:
                print('read_covid_data 메서드로 DataFrame을 생성해주세요.')
        else:
            print("'확진자 수', '사망자 수', '확진률' 중 하나의 요소를 입력해주세요.")

    @classmethod
    def show_changing_feature_chart(cls, feature):
        # 확진자 수, 사망자 수에 대한 변동률 라인차트 출력
        if feature in ['확진자 수', '사망자 수']:
            try:
                plt.plot(cls.DataFrame[feature].pct_change())
                x = sorted(set(i[:-2]+'01' for i in cls.DataFrame.index))
                plt.xticks(x, rotation=90)
            except TypeError:
                print('read_covid_data 메서드로 DataFrame을 생성해주세요.')
        else:
            print("'확진자 수', '사망자 수' 중 하나의 요소를 입력해주세요.")
```

- show_changing_feature_chart 메서드는 '확진자 수', '사망자 수', '확진률' 중 하나의 요소를 입력받아 누적 라인차트를 출력한다. (.cumsum() 함수를 적용하지 않았으나, 'Covid19InfState.csv' 파일이 누적 데이터로 구성되어 있음)

- show_changing_feature_chart 메서드는 '확진자 수', '사망자 수' 중 하나의 요소를 입력받아 변동률 라인차트를 출력한다. (pct_change 메서드를 적용하였다.)

```
    @classmethod
    def show_mulitple_changing_feature_chart(cls):
        # 사망자 수, 확진자 수에 대한 변동률 라인차트 동시 출력 (서로 다른 y축 척도)
        try:
            figure, ax1 = plt.subplots(figsize=(10,5))
            xtick_info = {date:idx for idx, date in enumerate(cls.DataFrame.index)
                          if date.endswith('01')}
            ax1.plot(cls.DataFrame['확진자 수'].pct_change(), color='blue')
            ax1.set_xlabel('시간')
            ax1.set_ylabel('확진자 수', color='blue')
            ax1.tick_params(axis='y', labelcolor='blue')

            ax1.set_xticks(list(xtick_info.values()))
            ax1.set_xticklabels(list(xtick_info.keys()), rotation='vertical', fontsize=8)

            ax2 = ax1.twinx()
            ax2.plot(cls.DataFrame['사망자 수'].pct_change(), color='red')
            ax2.set_ylabel('사망자 수', color='red')
            ax2.tick_params(axis='y', labelcolor='red')

            ax2.set_xticks(list(xtick_info.values()))
            ax2.set_xticklabels(list(xtick_info.keys()), rotation='vertical', fontsize=8)

        except TypeError:
                print('read_covid_data 메서드로 DataFrame을 생성해주세요.')
        except:
                print('TypeError 이외의 예외가 발생하였습니다.')
```

- show_mulitple_accumulated_feature_chart 메서드는 사망자 수, 확진자 수에 대한 누적 라인차트를 서로 다른 y축 척도에서 동시에 출력한다.

```
    @classmethod
    def show_mulitple_accumulated_feature_chart(cls):
        # 사망자 수, 확진자 수에 대한 누적 라인차트 동시 출력 (서로 다른 y축 척도)
        try:
            figure, ax1 = plt.subplots(figsize=(10,5))
            # 월별 01일을 기준으로 나타나는 x축 눈금 설정
            xtick_info = {date:idx for idx, date in enumerate(cls.DataFrame.index)
                          if date.endswith('01')}
            ax1.plot(cls.DataFrame['확진자 수'], color='blue')
            ax1.set_xlabel('시간')
            ax1.set_ylabel('확진자 수', color='blue')
            ax1.tick_params(axis='y', labelcolor='blue')

            ax1.set_xticks(list(xtick_info.values()))
            ax1.set_xticklabels(list(xtick_info.keys()), rotation='vertical', fontsize=8)

            ax2 = ax1.twinx()
            ax2.plot(cls.DataFrame['사망자 수'], color='red')
            ax2.set_ylabel('사망자 수', color='red')
            ax2.tick_params(axis='y', labelcolor='red')

            ax2.set_xticks(list(xtick_info.values()))
            ax2.set_xticklabels(list(xtick_info.keys()), rotation='vertical', fontsize=8)

        except TypeError:
            print('read_covid_data 메서드로 DataFrame을 생성해주세요.')
        except:
            print('TypeError 이외의 예외가 발생하였습니다.')
```

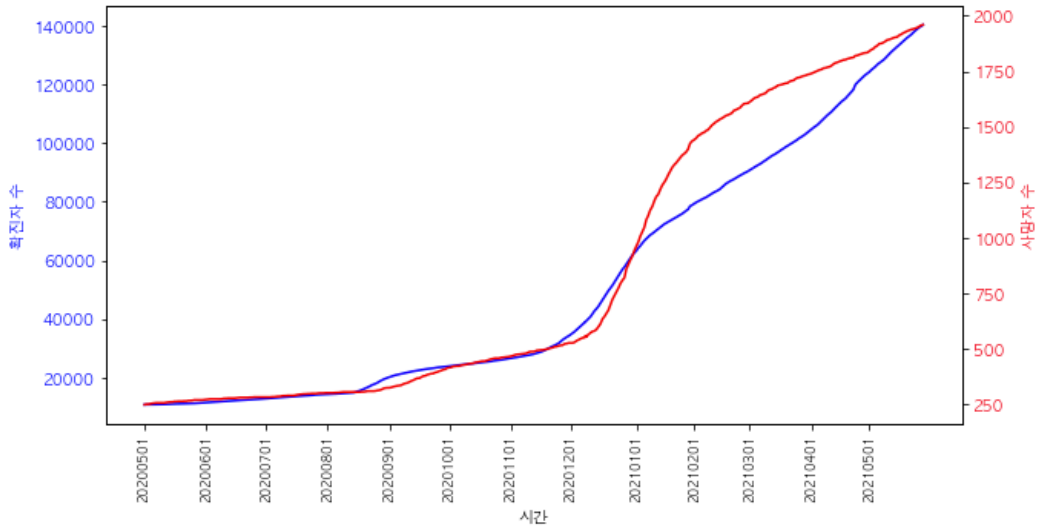
- show_mulitple_changing_feature_chart 메서드는 사망자 수, 확진자 수에 대한 변동률 라인차트를 서로 다른 y축 척도에서 동시에 출력한다.

다음은 COVID_Data의 두가지 동시출력 예시이다.

(1) show_mulitple_accumulated_feature_chart

```
COVID_Data.read_covid_data()
COVID_Data.show_mulitple_accumulated_feature_chart()
```

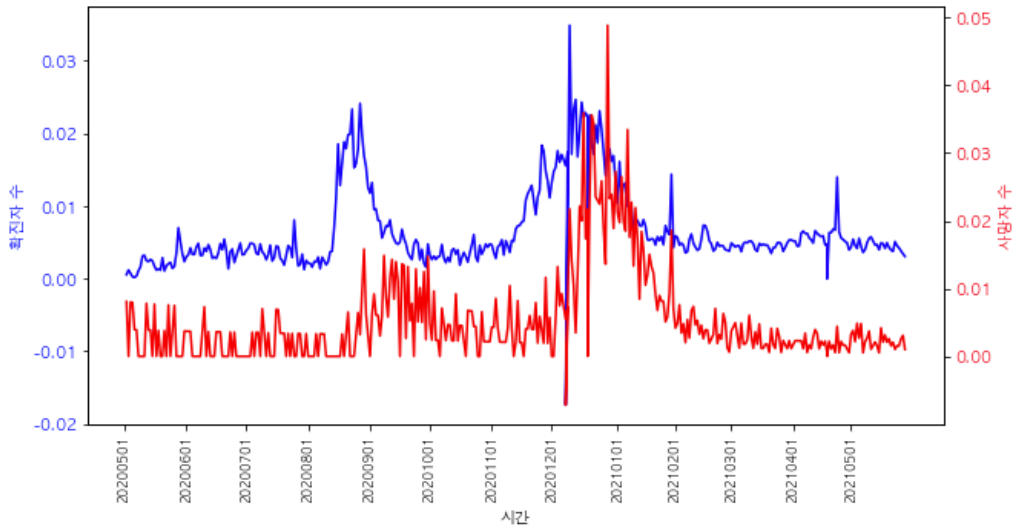
```
COVID DataFrame을 생성하였습니다.
```



(2) show_mulitple_accumulated_feature_chart

```
COVID_Data.read_covid_data()
COVID_Data.show_mulitple_changing_feature_chart()
```

```
COVID DataFrame을 생성하였습니다.
```



COVID_Data에 저장된 코로나 현황정보 DataFrame

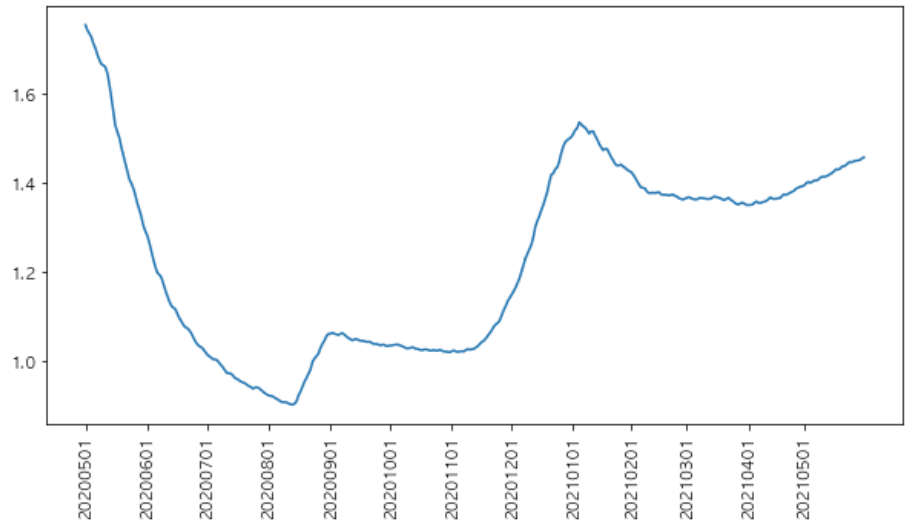
```
COVID_Data.DataFrame
```

코로나 현황정보 일일 시간표	확진자 수	사망자 수	확진률
20200501	10774.0	248.0	1.753626
20200502	10780.0	250.0	1.741333
20200503	10793.0	250.0	1.734136
20200504	10801.0	252.0	1.726102
20200505	10804.0	254.0	1.711175
...	...	...	...
20210527	138311.0	1943.0	1.449340
20210528	138898.0	1946.0	1.449938
20210529	139431.0	1951.0	1.450925
20210530	139910.0	1957.0	1.454196
20210531	140340.0	1959.0	1.456836

395 rows × 3 columns

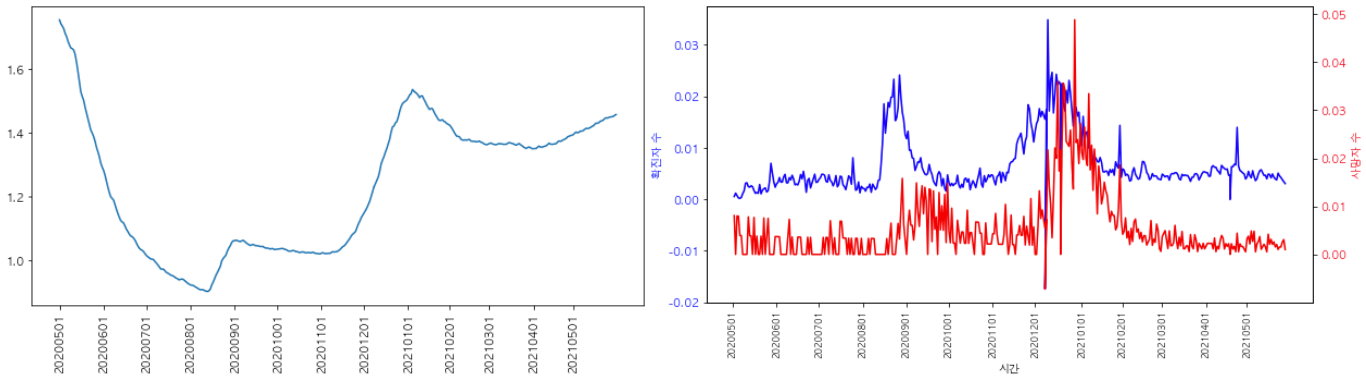
COVID_Data에 저장된 코로나 현황정보 DataFrame

```
COVID DataFrame을 생성하였습니다.
```



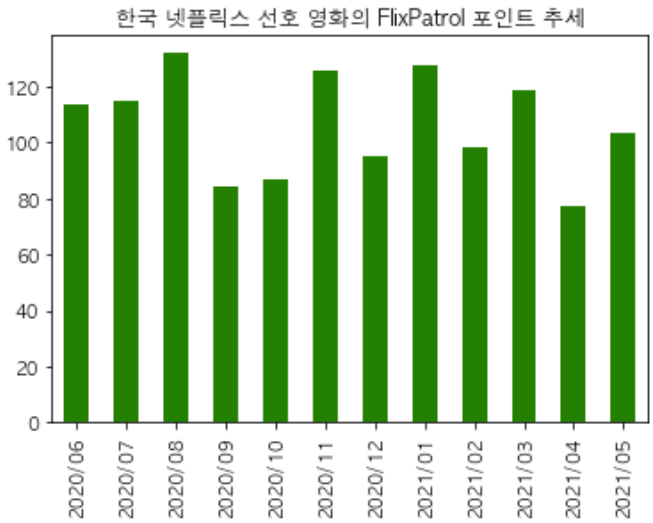
6. 자료분석

코로나 확진률 누적 라인차트 & 사망자 및 확진자의 변동률 라인차트



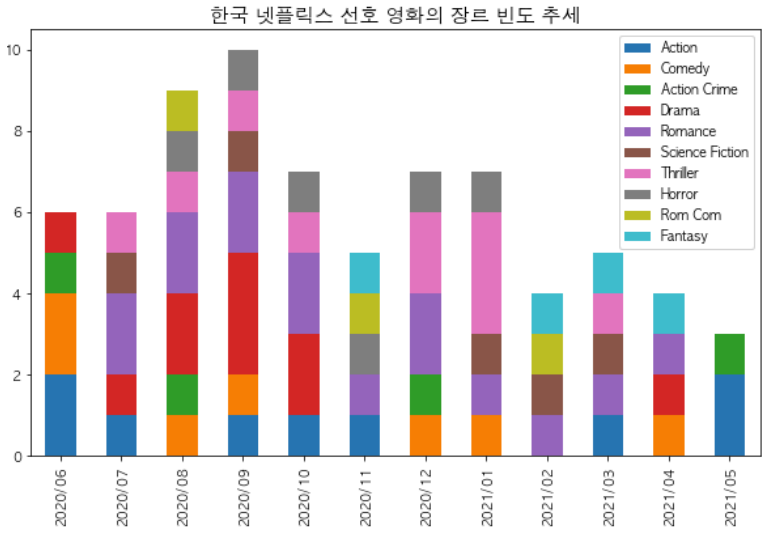
COVID_Data 클래스에서 확인한 두 라인차트에 따르면 코로나 19의 확산세가 급격히 빨라진 것은 2020년 9월과 2021년 1월이다. 그 중에서도 2021년 1월의 확산세가 훨씬 높은 수치를 보이며, 사망자 수 또한 2020년 9월보다 높은 비율로 나타나 사회적 불안감이 조성되었을 것을 추측된다.

한국 넷플릭스 선호 영화의 FlixPatrol 포인트 추세



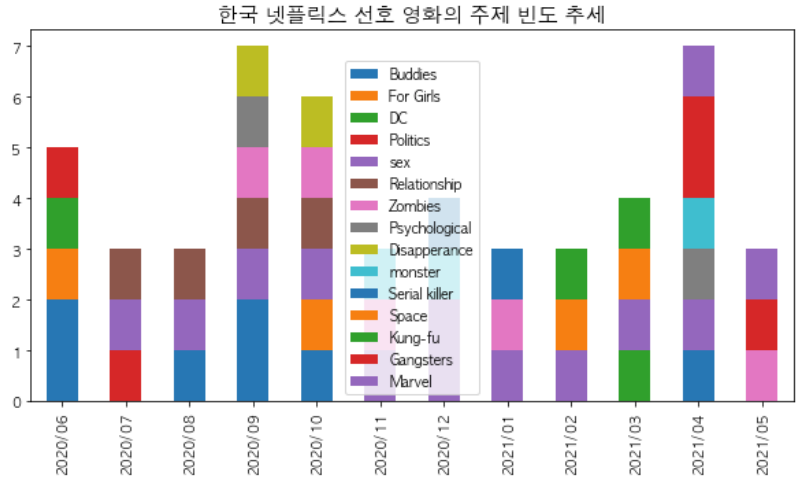
- 코로나 확산이 심각한 수준으로 나타났던 2020년 9월은 약 80 포인트, 2021년 1월은 약 120 포인트로 나타난다. 2020년 9월 코로나 확산에는 갑작스럽게 평이 좋지 않은 영화를 주로 찾은 반면, 2021년 1월 코로나 확산에는 가장 평이 좋은 영화들을 주로 소비하였다.
- 첫번째 코로나 대규모 확산에서는 '좋은 영화 콘텐츠'를 소비하는 데에 거부감을 느꼈으나, 두번째 코로나 확산에서는 선행 경험을 통해 '좋은 영화 콘텐츠'를 탐색하였다는 가설을 세울 수 있다. 단, 부족한 자료에서 도출한 결과인 만큼 충분한 검증단계가 필요하다.

한국 넷플릭스 선호영화의 장르 빈도 조사 (12개월 중 빈도 2회 이하 장르 제외)



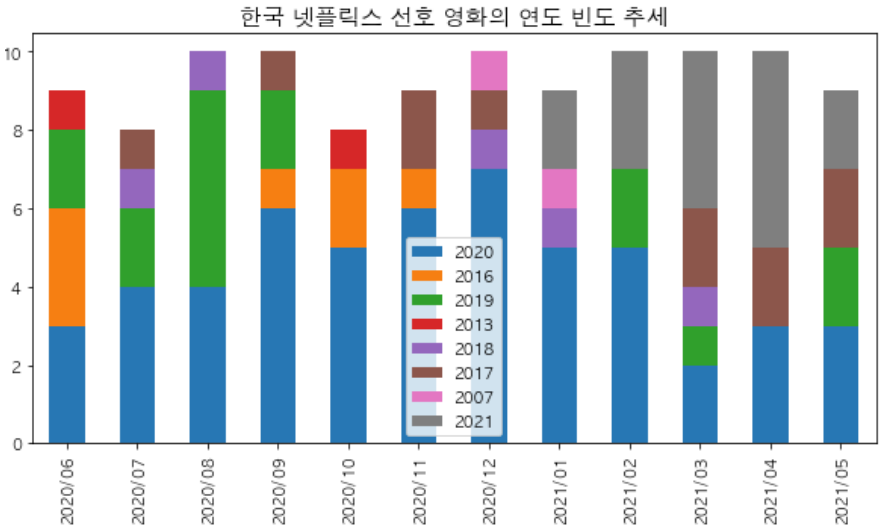
- 코로나가 급격히 확산된 2020년 9월은 12개월 중 가장 다양한 종류의 장르 영화가 소비되었다. 그 중 첫번째로 'Drama', 두번째로 'Romance' 장르가 가장 높은 빈도를 차지한다. 그 외에도 'Action', 'Comedy' 등 정신적 회복과 오락을 추구하는 장르가 빈번히 나타난다. 이 점을 고려했을 때, 2020년 9월 코로나 확산 시에는 정신적 스트레스 회복을 위한 영화장르 탐색이 적극적으로 이루어졌다고 분석할 수 있다.
- 반면 2021년 1월은 'Thriller' 장르의 빈도가 가장 높게 나타난다.
- 해당 결과에 의하면 첫번째 코로나 대규모 확산에서는 장기적인 정신적 회복을 위한 장르탐색이 주로 이루어졌으나 두번째 확산에서는 단기적인 스트레스 해소를 위한 오락장르 소비가 활발히 이루어졌다. 단, 데이터 표본의 규모가 작으므로 추가적인 검증과 분석이 필요하다.

한국 넷플릭스 선호영화의 주제 빈도 조사 (12개월 중 빈도 1회 이하 주제 제외)



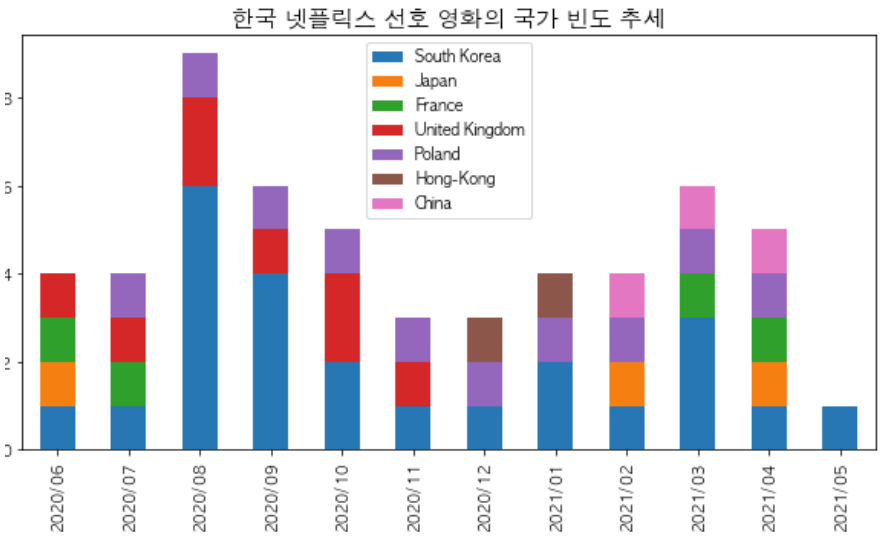
- 장르 차트와 마찬가지로 2020년 9월은 가장 다양한 주제가, 2021년 1월은 가장 저조한 주제가 나타난다.
- 위 그래프에 따르면 2020년에는 코로나 확산세가 심해질수록 다양한 주제의 영화를 탐색하지만, 2021년에 들어서는 코로나 확산세가 잦아들수록 다양한 주제의 영화를 탐색한다.

한국 넷플릭스 선호영화의 제작연도 빈도 조사



- 2020년의 영화가 2020년 12월을 중심으로 상승하고 하강하는 추세를 보인다. 좌측으로는 2019년 영화가 빈번히 나타나며, 우측으로는 2021년의 영화가 고빈도로 나타난다.
- 선호영화 제작연도에서는 코로나 19 확산 추세와 연관성을 찾기 어렵다.

한국 넷플릭스 선호영화의 제작국가 빈도 조사



- 2020년 9월에는 한국영화의 빈도가 높게 나타나나, 2021년 1월에는 빈약하게 나타난다. 그러나 전후 개월의 수치를 확인했을 때, 코로나 대확산 이전부터 지속되었던 경향이 반영된 결과로 보인다. 따라서 선호 영화의 제작국가에서도 코로나 19 확산 추세와 연관성을 찾기 어렵다.

7. 결론

위 자료분석에 따르면 2020년 9월 코로나 확산 시에는 드라마와 로멘스 영화 등 정신적 회복을 목적으로 영화를 시청하였다. 또한 평소에 접하지 않았던 다양한 종류의 장르와 주제의 영화를 탐색하였으며, 그 결과 비교적 낮은 평점의 영화들을 다량 소비하였다. 반면 2021년 1월 코로나 확산 시에는 스릴러 등 단기적인 스트레스 회복을 위한 영화시청이 이루어졌다. 동시에 높은 평점의 영화 콘텐츠를 선택적으로 시청한 것으로 보인다.

해당 분석결과에 따르면 질병 재난상황이 지속될수록 보다 한적적인 장르와 주제에서 영화소비가 이루어지며, 동시에 단기간에 높은 만족도를 보장하는 문화콘텐츠를 선호한다. 단, 본 프로젝트는 FlixPatrol 웹사이트에 누락된 수많은 상세정보를 고려하지 못하였고, 월별 10개의 영화에 한정된 소규모 표본으로 조사를 진행하여 빈약한 신뢰성을 보인다. 이를 해소하기 위해 FlixPatrol 외 웹사이트의 추가 데이터 분석을 진행하고 점진적인 일반화 작업을 진행한다면, 보다 가치있는 스트리밍 영화소비 경향성을 발견할 수 있을 것이라고 생각한다.