수능 한파는 실제로 존재할까?

[웹/파이썬프로그래밍] 컴퓨터공학과 2016104101 고승완

(1) 주제 선정 이유

수능한파(修能寒波) : 대학 수학 능력 시험을 치르는 당일 또는 그 시기가 되면 기온이 갑자기 내려가는 현상.

얼마전 갑자기 추워진 날씨에 매년 수능 즈음이 되면 날이 갑자기 추워졌던 기억이 떠올랐다. 문득 생각해보면 내가 수능 응원을 갔던 2014년 수능, 그리고 수능 시험을 보러 갔던 2015년 수능 때는 굉장히 추웠었다.
'왜 수능 시즌만 되면 갑자기 날씨가 추워지는 것일까?'라는 의문에서 시작된 이 Project는 실제로 수능 한파가 있는지를 확인하기 위해 수능이 시작된 1993년부터 2018년까지의 수능 '7일 전', '1일 전', '당일', '1일 후', '7일 후'의 최저 기온을 비교해 수능 시즌의 기온 변화를 확인할 것이다.
만약 실제로 수능 당일이나 그 시기가 평소보다 춥다면, 기온이 내려가는 것의 이유가 정말 신기하게 수능 때문인지 아니면 수능과는 상관없이 계절 변화로 인해서인지 확인해보고자 한다.

(2) 가설 정의

- 수능한파는 실제로 존재하지 않는다.
- 수능 시즌에 갑자기 추워진다고 느끼는 이유는 수능 때문이 아니라 단지 수능 시즌에 겨울이 다가와서 추워지는 기상 변화가 이루어졌기 때문이다.

(3) 인터넷을 통한 데이터 획득

1993년부터 2018년까지 수능 시즌의 기온 정보를 가져오기 위해 기상청의 '기상자료개방포털'의 Open-API를 이용했다.
'전체-기상관측-지상-종관기상관측' 분류의 '종관기상관측 일자료'를 python으로 가져오기 위해 서비스를 신청했고, Open API Key를 배당받았다.

```
import datetime
import urllib.request
import json
import time
import matplotlib.pyplot as plt

# [7일 전, 1일 전, 당일, 1일 후, 7일 후]를 따로 요청하니 시간이 너무 오래 걸림. 기간별로 받아서 해당 날짜의 데이터만 추출하는 함수
# 시작일과 종료일을 파라미터로 받아 요청 URL을 만들고, 자료를 요청함.
def apiCall(startDate, endDate):
    url = 'http://data.kma.go.kr/apiData/getData?type=json&dataCd=ASOS&dateCd=DAY&startDt='+startDate+'&endDt='+endDate+'&stnIds=108&scl
    #JSON 포맷 데이터를 파싱해서 사전 데이터로 만들어주는 라이브러리 사용
    request = urllib.request.Request(url)
    response = urllib.request.urlopen(request)
    response_body = response.read()

    data = json.loads(response_body)
    time.sleep(1)
    return data


#suDate = 역대 수능일자
suDate = [19931116,19941123,19951122,19961113,19971119,19981118,19991117,20001115,20011107,20021106,20031105,20041117,20051123,20061116,
#listMIN_TA = 각 수능별 [7일 전, 1일 전, 당일, 1일 후, 7일 후] 최저 기온
listMIN_TA = [[0 for j in range(0,5)]for i in range(0,len(suDate))]

for i in range(len(listMIN_TA)):
    # 수능 시즌별 최저 기온 데이터를 받기위한 날짜
    d = datetime.date(round(suDate[i]/10000), round(suDate[i]%10000/100), suDate[i]%100)
    startDate = (d + datetime.timedelta(days=-7)).strftime('%Y%m%d')
    endDate = (d + datetime.timedelta(days=+7)).strftime('%Y%m%d')

    #요청 URL을 만들고, 반복 요청을 통해 각 수능 시즌별 최저기온 데이터를 리스트에 저장함.
    frontData = apiCall(startDate, d.strftime('%Y%m%d')) # 7일 전~당일
    backData = apiCall((d + datetime.timedelta(days=+1)).strftime('%Y%m%d'), endDate) # 1일 후~7일 후

    listMIN_TA[i][0] = frontData[3]['info'][0]['MIN_TA'] # 7일 전 최저 기온
    listMIN_TA[i][1] = frontData[3]['info'][6]['MIN_TA'] # 1일 전 최저 기온
    listMIN_TA[i][2] = frontData[3]['info'][7]['MIN_TA'] # 당일 최저 기온
    listMIN_TA[i][3] = backData[3]['info'][0]['MIN_TA'] # 1일 후 최저 기온
    listMIN_TA[i][4] = backData[3]['info'][6]['MIN_TA'] # 1일 후 최저 기온

# 최저기온 리스트 출력
print(listMIN_TA)
print()
```

소스코드와 함께 데이터 획득 과정을 설명하겠다.

- Python에서 API 정보를 json 형식으로 불러오고 처리하기 위해 'urllib.request'와 'json' 모듈을 import했다.

- 역대 수능일자를 저장한 리스트 suDate가 있다. 각 수능일자는 'int'형으로 저장되어 있다.
- startDate와 endDate 값들은 URL에 기간을 정하는 것이다. 이 값들은 날짜 계산이 필요하다. 따라서 날짜 계산을 할 수 있는 'datetime' 모듈을 import했다.
- 날짜 계산 후, apiCall 함수에 startDate와 endDate 값을 넣어 api를 call한다. 각 수능 시즌의 서울특별시 기준 종관기상관측 일자료 정보를 두 번에 걸쳐 json 형식으로 받아온다. 두 번 Call하는 이유는 부를 수 있는 정보가 1주일 기간으로 한정되어 있기 때문이다. 따라서 7일 전~당일, 1일 후~7일 후로 나누어 Call하는 것이다.
- for문을 통해 서버에 자료를 요청할 때 Bad Request Error가 발생해, apiCall 함수에서는 data를 return하기 전에 time 모듈의 sleep함수를 사용했다. 너무 짧은 시간에 제한된 요청 수 이상을 요청해서 그런듯 하다.
  아래 사진은 apiCall함수를 호출한 이후의 예시로, '1993년 11월 16일 기준 ±7일' 종관기상관측 일자료 정보의 일부이다.(데이터 확인을 위해 잠시 코드 값을 수정한 상태이다.)

```
for i in range(1):
    # 수능 시즌별 최저 기온 데이터를 받기위한 날짜
    d = datetime.date(round(suDate[i]/10000), round(suDate[i]%10000/100), suDate[i]%100)
    startDate = (d + datetime.timedelta(days=-7)).strftime('%Y%m%d')
    endDate = (d + datetime.timedelta(days=+7)).strftime('%Y%m%d')

    #요청 URL을 만들고, 반복 요청을 통해 각 수능 시즌별 최저기온 데이터를 리스트에 저장함.
    frontData = apiCall(startDate, d.strftime('%Y%m%d')) # 7일 전~당일
    backData = apiCall((d + datetime.timedelta(days=+1)).strftime('%Y%m%d'), endDate) # 1일 후~7일 후

    print(frontData, backData)
```

[{'status': 200}, {'msg': 'success'}, {'stnIds': '108'}, {'info': [{'MAX_WD': 70, 'MIN_PS': 1026.9, 'TM': '1993-11-09', 'AVG_RHM': 45.8, 'STN_ID': 108, 'MIN_RHM_HRMT': 1533, 'MAX_INS_WS_WD': 70, 'AVG_TS': 11.4, 'MAX_INS_WS_HRMT': 2025, 'MAX_PS_HRMT': 900, 'AVG_PV': 6.6, 'SUM_SS_HR': 8, 'MIN_RHM': 29, 'SS_DUR': 10.4, 'AVG_PS': 1029.3, 'MAX_WS': 5, 'MIN_TG': 2.6, 'MAX_WS_WD': 70, 'MAX_TA_HRMT': 1420, 'AVG_TCA': 3.3, 'HR1_MAX_ICSR': 1.75, 'AVG_TD': 0.9, 'AVG_TA': 12.7, 'RNUM': 1, 'MAX_PS': 1031.1, 'MIN_PS_HRMT': 1540, 'MIN_TA': 9, 'STN_NM': '서울', 'MAX_TA': 17.8, 'HR24_SUM_RWS': 2853, 'HR1_MAX_ICSR_HRMT': 1100, 'MAX_INS_WS': 11, 'AVG_LMAC': 1.3, 'MIN_TA_HRMT': 704, 'SUM_GSR': 9.55, 'MAX_WS_HRMT': 1010, 'AVG_PA': 1018.7, 'AVG_WS': 3.3}, {'SUM_RN_DUR': 1.63, 'MIN_PS': 1024.3, 'MAX_WD': 70, 'HR1_MAX_RN_HRMT': 1118, 'TM': '1993-11-10', 'AVG_RHM': 55.8, 'STN_ID': 108, 'MIN_RHM_HRMT': 0, 'MAX_INS_WS_WD': 70, 'AVG_TS': 12.1, 'MAX_INS_WS_HRMT': 930, 'MAX_PS_HRMT': 0, 'AVG_PV': 8, 'SUM_SS_HR': 0.3, 'MIN_RHM': 48, 'SS_DUR': 10.3, 'AVG_PS': 1026.4, 'MAX_WS': 4.8, 'MIN_TG': 5.7, 'MAX_TA_HRMT': 1513, 'MAX_WS_WD': 70, 'AVG_TCA': 9, 'MI10_MAX_RN': 0.1, 'HR1_MAX_ICSR': 0.6, 'AVG_TD': 3.6, 'AVG_TA': 12, 'RNUM': 2, 'MAX_PS': 1029.9, 'MIN_PS_HRMT': 1455, 'MIN_TA': 9.7, 'STN_NM': '서울', 'MAX_TA': 14.2, 'HR24_SUM_RWS': 2246, 'MI10_MAX_RN_HRMT': 1130, 'HR1_MAX_ICSR_HRMT': 1200, 'MAX_INS_WS': 9.3, 'SUM_RN': 0.3, 'ISCS': '{비}1057-{비}{강도0}1200-1218. {비}1358-1410. {비}1629-1634.', 'AVG_LMAC': 4, 'MIN_TA_HRMT': 700, 'SUM_GSR': 3.93, 'MAX_WS_HRMT': 932, 'AVG_PA': 1015.8, 'HR1_MAX_RN': 0.2, 'AVG_W

서버에서 불러온 정보에서 최저 기온(Min_TA)을 추출한다.

```
for i in range(1):
    # 수능 시즌별 최저 기온 데이터를 받기위한 날짜
    d = datetime.date(round(suDate[i]/10000), round(suDate[i]%10000/100), suDate[i]%100)
    startDate = (d + datetime.timedelta(days=-7)).strftime('%Y%m%d')
    endDate = (d + datetime.timedelta(days=+7)).strftime('%Y%m%d')

    #요청 URL을 만들고, 반복 요청을 통해 각 수능 시즌별 최저기온 데이터를 리스트에 저장함.
    frontData = apiCall(startDate, d.strftime('%Y%m%d')) # 7일 전~당일
    backData = apiCall((d + datetime.timedelta(days=+1)).strftime('%Y%m%d'), endDate) # 1일 후~7일 후

    listMIN_TA[i][0] = frontData[3]['info'][0]['MIN_TA'] # 7일 전 최저 기온
    listMIN_TA[i][1] = frontData[3]['info'][6]['MIN_TA'] # 1일 전 최저 기온
    listMIN_TA[i][2] = frontData[3]['info'][7]['MIN_TA'] # 당일 최저 기온
    listMIN_TA[i][3] = backData[3]['info'][0]['MIN_TA'] # 1일 후 최저 기온
    listMIN_TA[i][4] = backData[3]['info'][6]['MIN_TA'] # 1일 후 최저 기온

    print(listMIN_TA[i])
```

```
[9, 9.9, 7.6, 11.1, -7]
```

- 예시를 보였으니, 이제 추출한 '최저 기온'을 저장해 각 수능 시즌별 최저 기온 변화양상을 나타내는 리스트를 만들어보자.

```
for i in range(len(listMIN_TA)):
    # 수능 시즌별 최저 기온 데이터를 받기위한 날짜
    d = datetime.date(round(suDate[i]/10000), round(suDate[i]%10000/100), suDate[i]%100)
    startDate = (d + datetime.timedelta(days=-7)).strftime('%Y%m%d')
    endDate = (d + datetime.timedelta(days=+7)).strftime('%Y%m%d')

    #요청 URL을 만들고, 반복 요청을 통해 각 수능 시즌별 최저기온 데이터를 리스트에 저장함.
    frontData = apiCall(startDate, d.strftime('%Y%m%d')) # 7일 전~당일
    backData = apiCall((d + datetime.timedelta(days=+1)).strftime('%Y%m%d'), endDate) # 1일 후~7일 후

    listMIN_TA[i][0] = frontData[3]['info'][0]['MIN_TA'] # 7일 전 최저 기온
    listMIN_TA[i][1] = frontData[3]['info'][6]['MIN_TA'] # 1일 전 최저 기온
    listMIN_TA[i][2] = frontData[3]['info'][7]['MIN_TA'] # 당일 최저 기온
    listMIN_TA[i][3] = backData[3]['info'][0]['MIN_TA'] # 1일 후 최저 기온
    listMIN_TA[i][4] = backData[3]['info'][6]['MIN_TA'] # 1일 후 최저 기온

print(listMIN_TA)
```

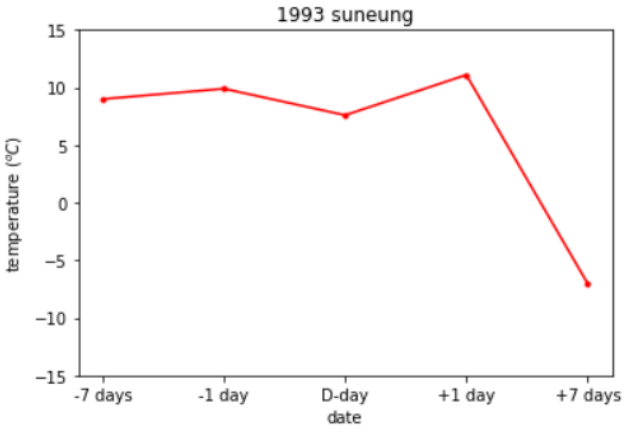
```
[[9, 9.9, 7.6, 11.1, -7], [7.7, 2.3, 2.9, 6.5, 1.2], [-0.4, -1.7, 3.5, -2.8, 0], [3, 5.6, 2.1, 0.7, 3.9], [11.4, -3.2, -3.2, 0.2, 3.
3], [5.8, -2, -5.3, -6.8, -2.5], [7.8, 2.9, 1, 2.3, 5.9], [0, 4.5, 7.9, 3.9, -3.8], [12.7, 0.8, -0.3, 3.2, -0.2], [1, -1.6, 5, 6.3,
0], [2.7, 3.4, 8.1, 9.6, 7.4], [13.6, 1.2, 4.9, 7.8, 5.9], [-1.1, 0.8, 3.5, 5.4, -0.8], [8.6, 1.8, -0.4, 0.5, 6], [7.5, 8.3, 4.6, 1.2,
-2.7], [12.5, 5.1, 5.3, 6.8, -5.3], [11.4, 6.4, 6, 8.2, -0.7], [3.2, 3, 1.9, 2.4, -1], [10, 12.5, 10.9, 11, 10.5], [2.7, 4.1, 6, 4.4,
-0.3], [5.5, 9, 7.5, 4, 3.1], [9.5, -1.3, -3.1, -1.4, 2.7], [8.6, 6.6, 10.2, 10.5, 6.7], [1.5, 0, 4, 4.9, -6.2], [-3.4, 0.6, -2.5, -2.
5, -5.2], [10.5, 4.8, 4.7, 4.2, -1.3]]
```

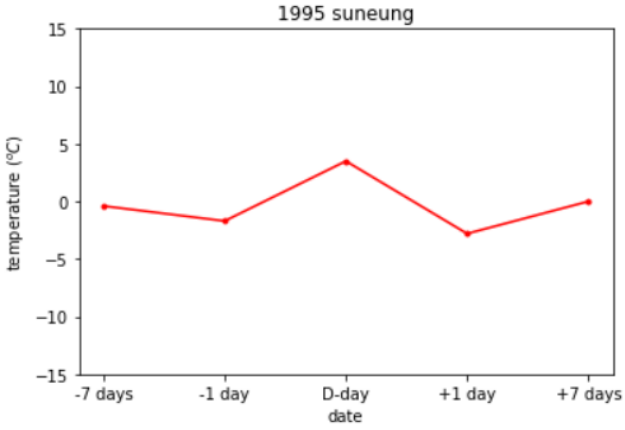
(4) 분석을 위한 데이터의 가공

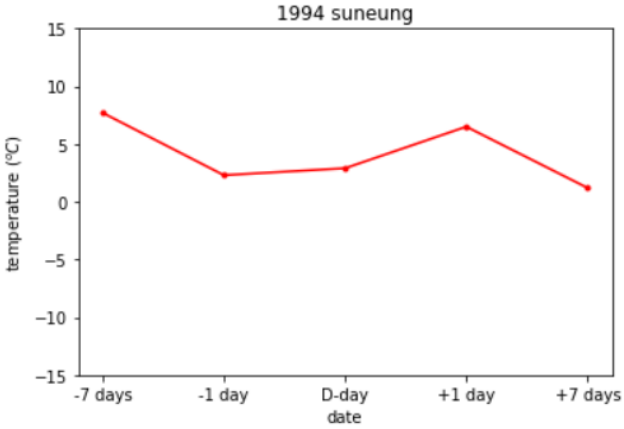
위 과정을 통해 수능 시즌 26년치의 최저 기온 리스트가 완성되었다.
이제 각 수능 시즌의 '7일 전-1일 전-당일-1일 후-7일 후'의 기온 변화가 어떤 양상을 띄는지 확인하기 위해 리스트 데이터를 그래프로 가공할 것이다.
그래프를 사용하기 위해 'matplotlib.pyplot'을 import 해주었다.

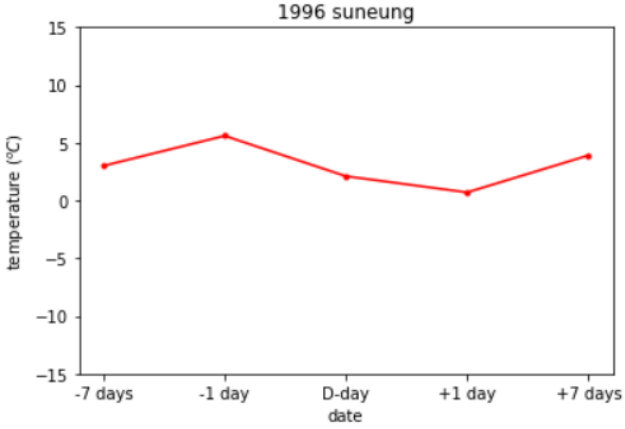
```
import matplotlib.pyplot as plt

for i in range(len(listMIN_TA)):
    y_value = listMIN_TA[i]
    x_name=('-7 days', '-1 day', 'D-day', '+1 day', '+7 days')

    plt.plot(x_name,y_value,'r.-')

    plt.xlabel('date')
    plt.ylabel('temperature ($^oC$)')
    plt.title(str(suDate[i])[:4]+' suneung')
    plt.ylim(-15, 15)
    plt.show()
```

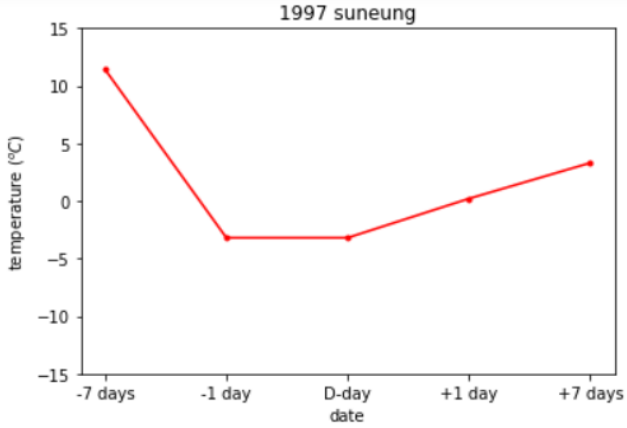
위 코드는 예시 코드이고, 아래 그래프들은 그 결과물이다.

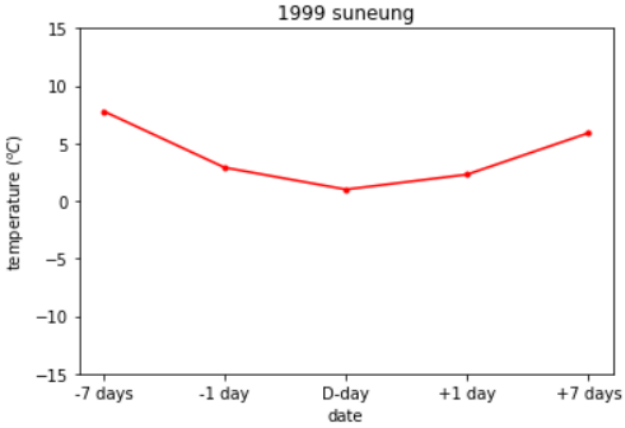

1993 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

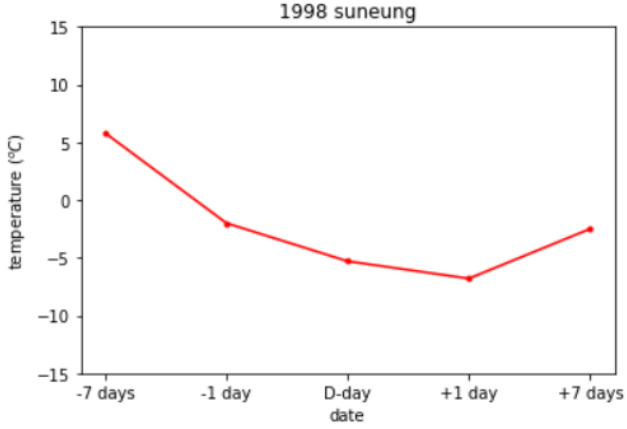

1995 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

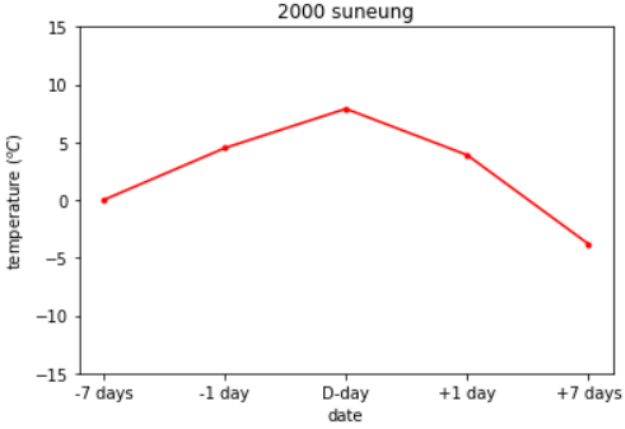

1994 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

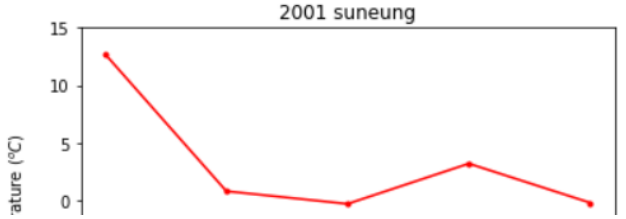

1996 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

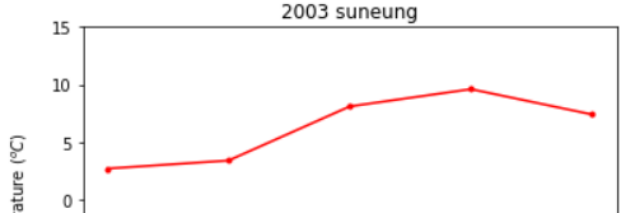

1997 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

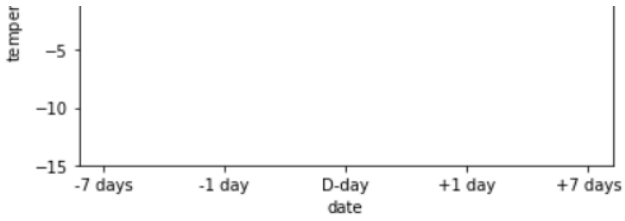

1999 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

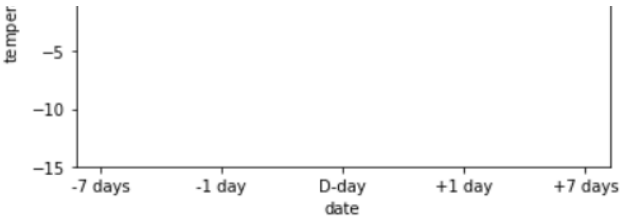

1998 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

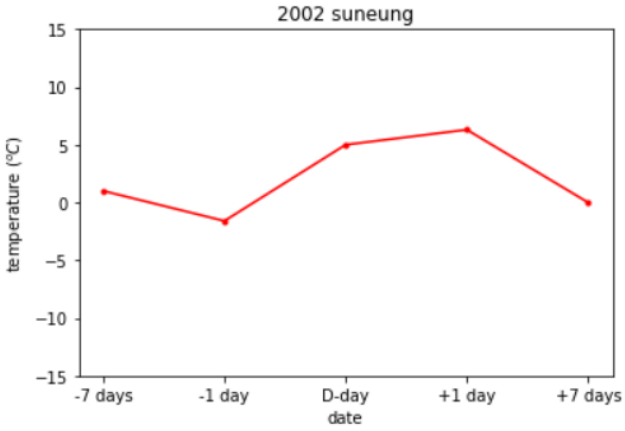

2000 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

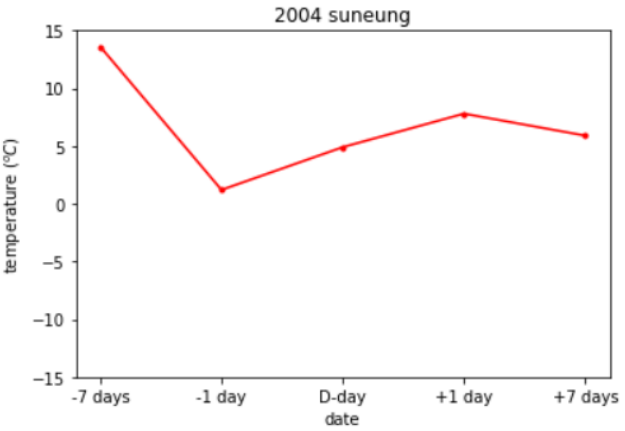

2001 suneung
15
10
5
0

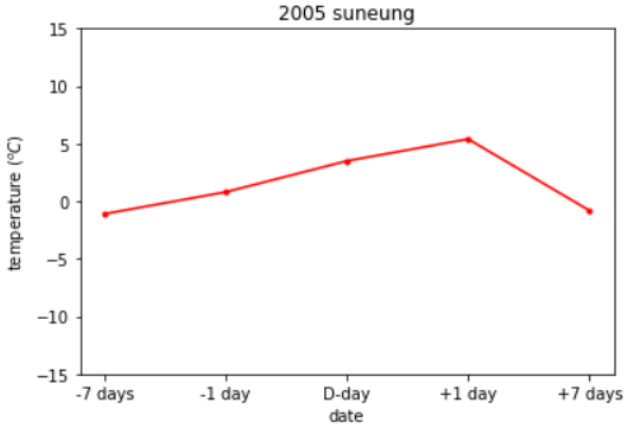

2003 suneung
15
10
5
0

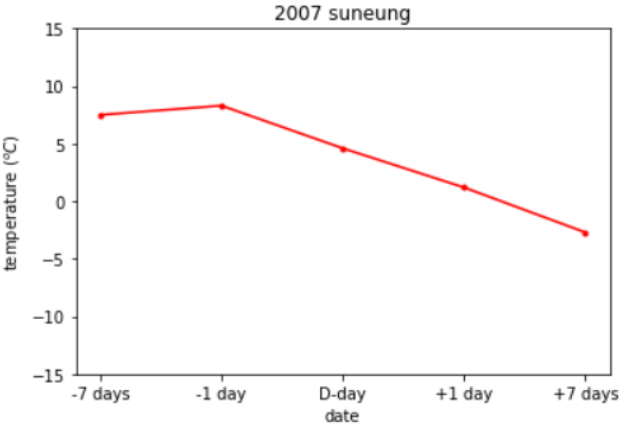

temper
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

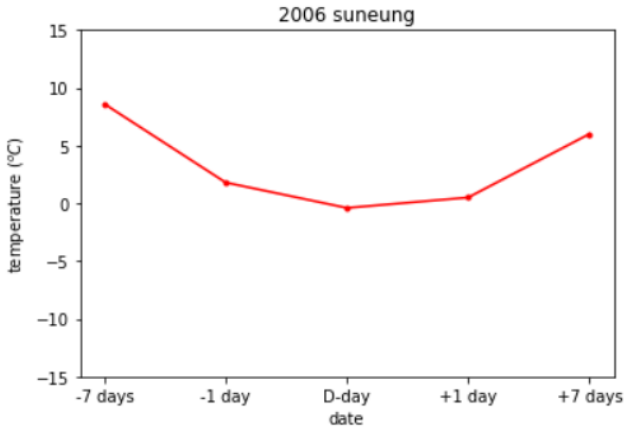

temper
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

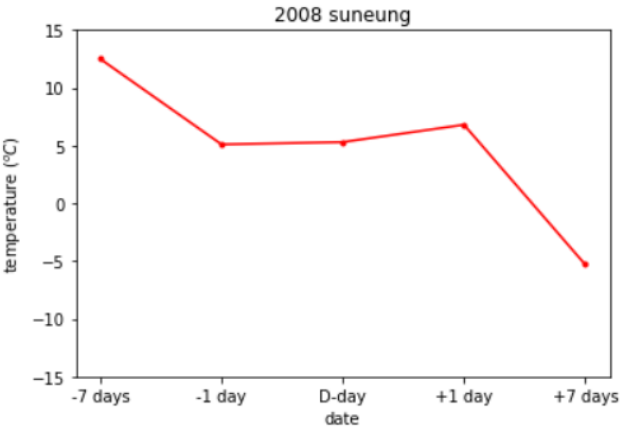

2002 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

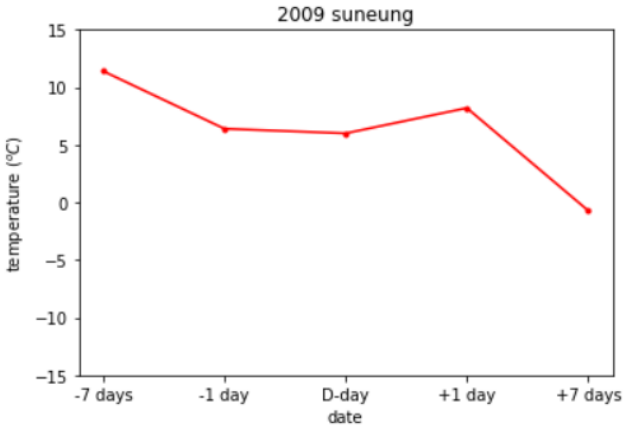

2004 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

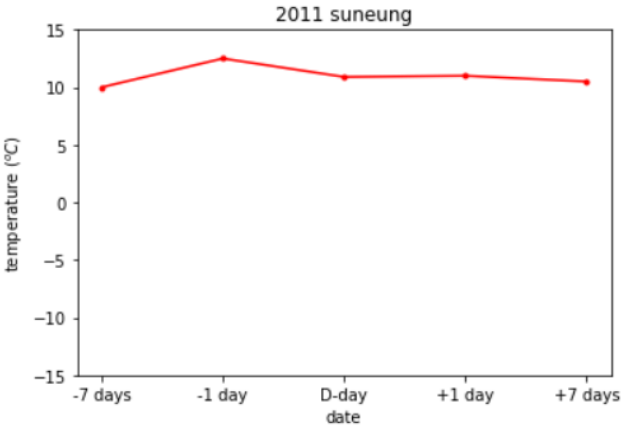

2005 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

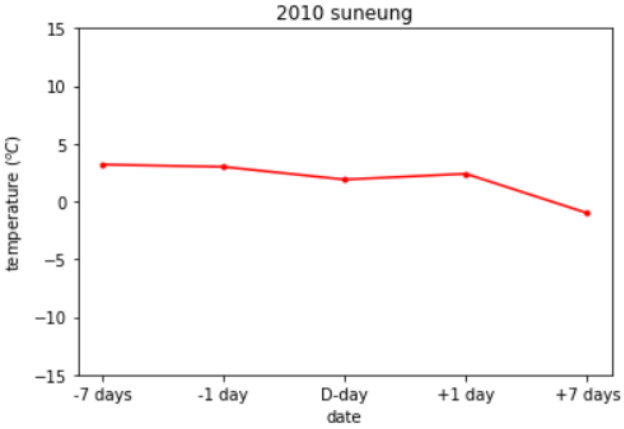

2007 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

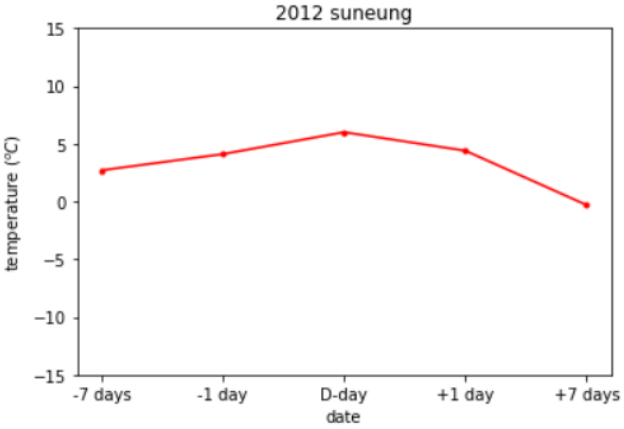

2006 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

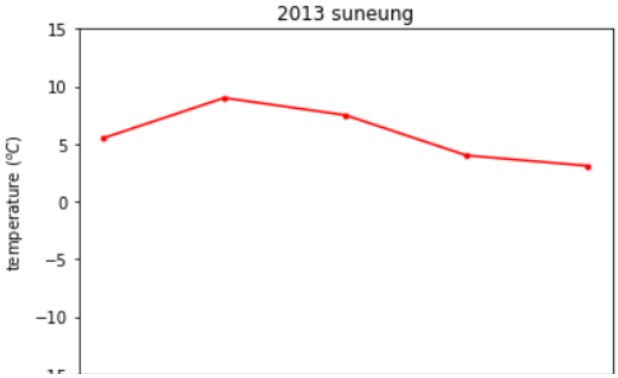

2008 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

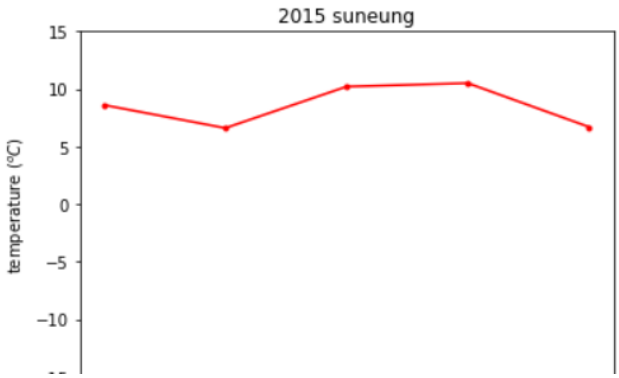

2009 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date


2011 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date


2010 suneung
temperature (℃)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date


2012 suneung
temperature (℃)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date


2013 suneung
temperature (℃)
15
10
5
0
−5
−10


2015 suneung
temperature (℃)
15
10
5
0
−5
−10

(5) 분석 결과 도출

각 수능 시즌별로 어떤 기온 변화를 보여왔는지 직접 분석해보자.
만약 수능 한파가 실존한다면 수능 당일이나 그 시기의 기온이 영하권이거나, 7일 전보다 수능에 가까워지면서 기온이 5℃ 이상 낮아질 것이다.
만약 전체 경우 중에서 60% 이상이 한파가 있었다라고 분석되면 수능 한파가 실존한다고 결론지을 것이다.

1993년

- 1993년 시즌은 대체적으로 평이한 날씨를 보였다. 수능 7일 후 날씨가 급격히 추워졌다.

1994년

- 수능 1일 전의 기온이 7일 전보다 5℃ 이상 하강했다. 이는 수능 한파라고 할 수 있다.

1995년

- 1995년 시즌은 전체적으로 0℃ 정도의 추운 날씨를 보였다. 하지만 날씨가 갑자기 추워지지 않았고, 오히려 수능 당일의 기온이 상승했기 때문에 수능 한파라고 볼 수 없다.

1996년

- 1996년 시즌은 0℃와 5℃ 사이의 날씨를 보였으나, 영하권의 날씨가 아니고 수능 시기에 특별하게 추웠다고 볼 수 없는 날씨를 보였다.

1997년

- 1997년 시즌은 7일 전에 비해 '수능 전날과 당일' 급격한 기온 하강이 있고, 수능 당일의 기온이 영하권이다. 따라서 수능 한파라고 볼 수 있다.

1998년

- 1998년 시즌은 7일 전에 비해 기온이 급락했고, 수능 1일 전부터 영하권의 날씨를 보이고있다. 따라서 수능 한파라고 볼 수 있다.

1999년

- 1999년 시즌은 전체적으로 기온이 영상이고, 기온이 5℃ 이상 하강한 모습이 없기 때문에 수능 한파라고 볼 수 없다.

2000년

- 2000년 시즌은 기온이 영상권이고, 그래프의 모양이 산 모양이다. 기준을 만족하지 못한다.

2001년

- 2001년 시즌의 경우 기온이 수능 전날 급하강했고, 이후로 쭉 0℃ 부근에서 온도 변화가 있었다. 영하권은 아니지만 이는 수능 한파라고 볼 수 있다.

2002년

- 2002년 시즌의 경우 기온이 대체적으로 영상권이고, 수능 당일의 기온이 5℃ 부근이기 때문에 수능 한파라고 볼 수 없다.

2003년

- 2003년 시즌의 경우 수능에 가까워짐에도 오히려 기온이 상승했다.

2004년

- 2004년 시즌의 경우 기온이 7일 전 13.6℃에서 수능 전날 1.2도로 급격히 하강했다. 비록 수능 당일 기온이 영하권은 아니나 이는 수능 한파라고 볼 수 있다.

2005년

- 2005년 시즌의 경우 수능 시기에 0℃ 이상의 기온을 보이고 있다. 수능 한파라고 볼 수 없다.

2006년

- 2006년 시즌의 경우 기온이 7일 전 8.6℃에서 수능 전날 1.8도로 급격히 하강했고 수능 당일 영하의 기온을 보였다. 이는 수능 한파라고 볼 수 있다.

2007년

- 2007년 시즌의 경우 수능 전날을 기점으로 기온이 하락세를 보이고 있다. 수능 한파는 '수능을 치루는 시기가 되면 기온이 갑자기 내려가는 현상'이기 때문에 이런 그래프도 수능 한파라고 봐야 한다.

2008년

- 2008년 시즌의 경우 7일 전 12.5℃에서 1일 전 5.1℃로 하강했고, 수능 후 기온이 급하강했다. 수능 시험 부근의 기온이 5℃ 이상이기는 하나, 7일 전에 비해 5℃ 이상 하강했으므로 이는 수능 한파라고 볼 수 있다.

2009년

- 2009년 시즌의 경우 7일 전 11.4℃에서 1일 전 6.4℃로 5℃ 하강했다. 그리고 수능 7일 후 -0.7℃로 급하강했다. 수능 시험 부근의 기온이 0℃ 이상이기는 하나, 7일 전에 비해 5℃ 하강했으므로 이는 수능 한파라고 볼 수 있다.

2010년

- 2010년 시즌의 경우 쭉 쌀쌀한 날씨였으나, 영상권이었기 때문에 이 경우는 수능 한파라고 볼 수 없다.

2011년

- 2011년 시즌의 경우 10℃ 이상을 유지하는 대체적으로 평이한 날씨였다.

2012년

- 2012년 시즌의 경우 전체적으로 기온이 0℃ 이상이고, 기온이 급강하한 모습이 없기 때문에 수능 한파라고 볼 수 없다.

2013년

- 2013년 시즌의 경우 전체적으로 기온이 5℃ 이상이다. 수능 1일 전을 기점으로 기온이 하락하고 있으나, 특별히 기온이 급락했다고 볼 수 없기 때문에 한파의 기준에 맞지 않는다.

2014 년

- 2014년 시즌의 경우 1일 전 기온이 7일 전 기온에 비해 10℃ 이상 급락했고, 수능 시기의 기온이 영하권이기 때문에 이는 수능 한파라고 볼 수 있다.

2015 년

- 2015년 시즌의 경우 전체적으로 6℃ 이상의 기온을 보였으며, 특별히 기온이 내려가는 추세의 그래프도 아니다.

2016 년

- 2016년 시즌의 경우 기온이 급락한 모습도 없고, 수능 시기의 기온이 영하권도 아니었다.

2017 년

- 2017년 시즌의 경우 약 0℃ 이하의, 굉장히 추운 시즌이었다. 따라서 수능 한파라고 볼 수 있다.

2018 년

(6) 결론

분석 결과 도출 후, 수능 한파의 기준에 맞는 데이터는 총 12개(94, 97, 98, 01, 04, 06, 07, 08, 09, 14, 17, 18년), 기준에 맞지않는 데이터는 14개(93, 95, 96, 99, 00, 02, 03, 05, 10, 11, 12, 13, 15, 16년)였다.
'수능 한파가 실존한다, 정기적으로 찾아오는 이벤트다'라고 하기 위해서 전체 데이터의 60%이 수능 한파 현상을 보여야한다고 가정했다. 총 26년치의 데이터 중 12개(46.2%)가 조건을 충족했고, 60%인 15.6개 이상(16개 이상)을 만족하지 못했다.
따라서 수능 시즌에 기온이 낮아지는 것은 수능과 관련이 없다고 볼 수 있다. 언론과 사람들이 그저 계절이 변해감에 따른 평범한 기상 현상을 수능과 엮어 프레임을 씌웠을 뿐, 수능 한파는 존재하지 않는다.

(7) 참고문헌

- 주피터 노트북 markdown 작성법
  https://heropy.blog/2017/09/30/markdown/ (https://heropy.blog/2017/09/30/markdown/)
  https://blog.naver.com/PostView.nhn?blogId=syg7949&logNo=221417147237&parentCategoryNo=&categoryNo=8&viewDate=&isShowPopularI (https://blog.naver.com/PostView.nhn?blogId=syg7949&logNo=221417147237&parentCategoryNo=&categoryNo=8&viewDate=&isShowPopularI
  https://hashcode.co.kr/questions/1772/%EB%A7%88%ED%81%AC%EB%8B%A4%EC%9A%B4-%EB%AC%B8%EB%B2%95-%EC%9E%91%EC%84%B1-%ED%8C%81 (https://hashcode.co.kr/questions/1772/%EB%A7%88%ED%81%AC%EB%8B%A4%EC%9A%B4-%EB%AC%B8%EB%B2%95-%EC%9E%91%EC%84%B1-%ED%8C%81)
- 기상청 API 사용법
  https://kingpodo.tistory.com/13 (https://kingpodo.tistory.com/13)
- OpenAPI활용가이드(기상청 기상관측자료(종관 일자료))_v1.0.docx : open api 신청 후 확인할 수 있는 오픈 API 현황의 첨부 문서이다.
- json 파싱
  https://www.fun-coding.org/crawl_basic3.html (https://www.fun-coding.org/crawl_basic3.html)
- Python datetime 날짜 계산
  https://yuddomack.tistory.com/entry/%ED%8C%8C%EC%9D%B4%EC%8D%AC-datetime-%EB%82%A0%EC%A7%9C-%EA%B3%84%EC%82%B0 (https://yuddomack.tistory.com/entry/%ED%8C%8C%EC%9D%B4%EC%8D%AC-datetime-%EB%82%A0%EC%A7%9C-%EA%B3%84%EC%82%B0)
- 그래프 그리기
  https://techreviewtips.blogspot.com/2017/10/04-04-bar.html (https://techreviewtips.blogspot.com/2017/10/04-04-bar.html)
  https://ratsgo.github.io/machine%20learning/2017/06/23/visual/ (https://ratsgo.github.io/machine%20learning/2017/06/23/visual/)
- 모바일(Android) 환경에서 jupyter notebook 사용하기
  https://stackoverflow.com/questions/49837474/how-do-i-install-jupyter-notebook-on-an-android-device (https://stackoverflow.com/questions/49837474/how-do-i-install-jupyter-notebook-on-an-android-device)

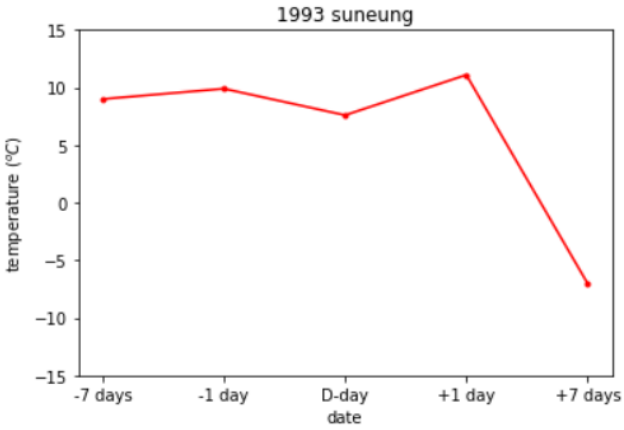
(8) 별첨1

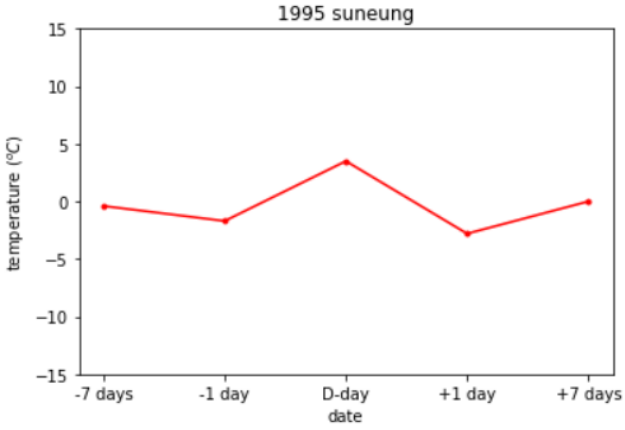
(3)의 획득한 데이터 원본

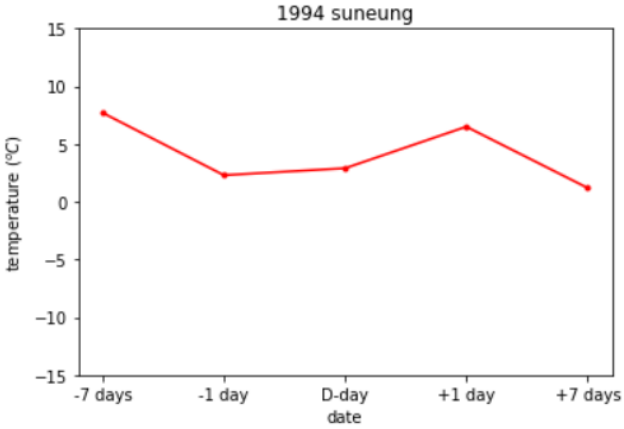
```
[[9, 9.9, 7.6, 11.1, -7], [7.7, 2.3, 2.9, 6.5, 1.2], [-0.4, -1.7, 3.5, -2.8, 0], [3, 5.6, 2.1, 0.7, 3.9], [11.4, -3.2, -3.2, 0.2, 3.3], [5.8, -2, -5.3, -6.8, -2.5], [7.8, 2.9, 1, 2.3, 5.9], [0, 4.5, 7.9, 3.9, -3.8], [12.7, 0.8, -0.3, 3.2, -0.2], [1, -1.6, 5, 6.3, 0], [2.7, 3.4, 8.1, 9.6, 7.4], [13.6, 1.2, 4.9, 7.8, 5.9], [-1.1, 0.8, 3.5, 5.4, -0.8], [8.6, 1.8, -0.4, 0.5, 6], [7.5, 8.3, 4.6, 1.2, -2.7], [12.5, 5.1, 5.3, 6.8, -5.3], [11.4, 6.4, 6, 8.2, -0.7], [3.2, 3, 1.9, 2.4, -1], [10, 12.5, 10.9, 11, 10.5], [2.7, 4.1, 6, 4.4, -0.3], [5.5, 9, 7.5, 4, 3.1], [9.5, -1.3, -3.1, -1.4, 2.7], [8.6, 6.6, 10.2, 10.5, 6.7], [1.5, 0, 4, 4.9, -6.2], [-3.4, 0.6, -2.5, -2.5, -5.2], [10.5, 4.8, 4.7, 4.2, -1.3]]
```

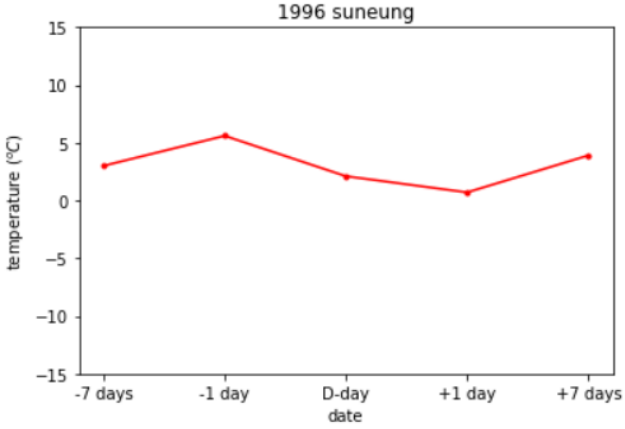
(9) 별첨2

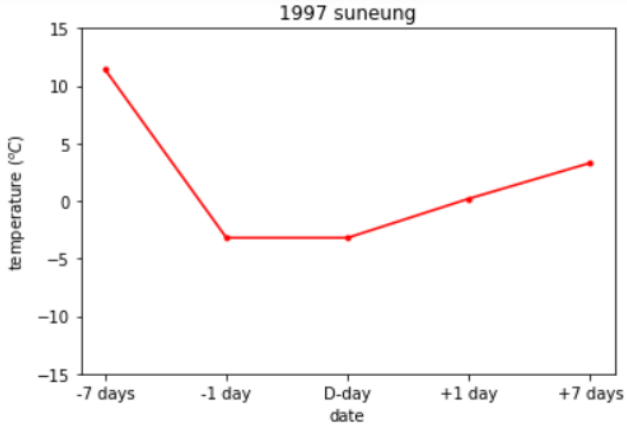
(4)의 가공된 데이터 원본

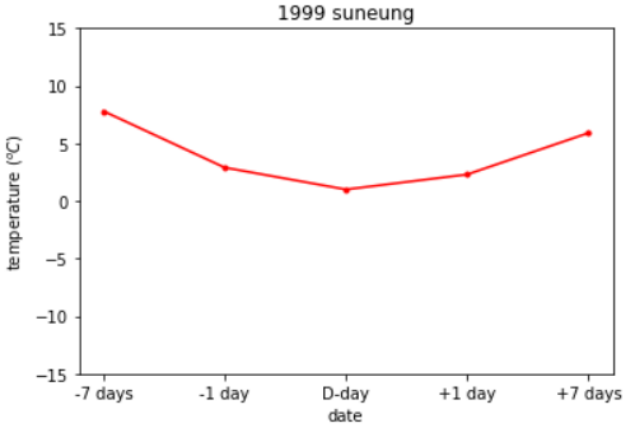

1993 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

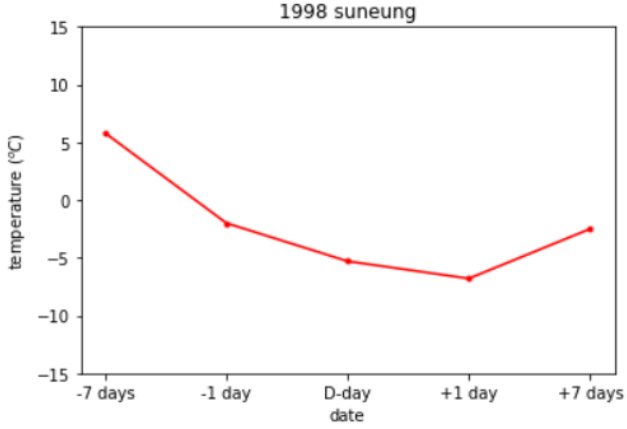

1995 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

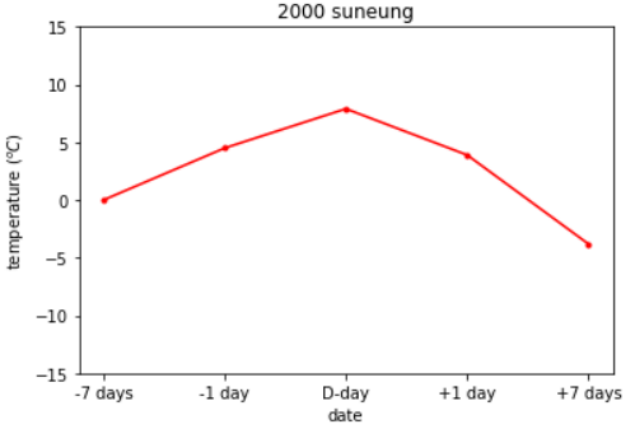

1994 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

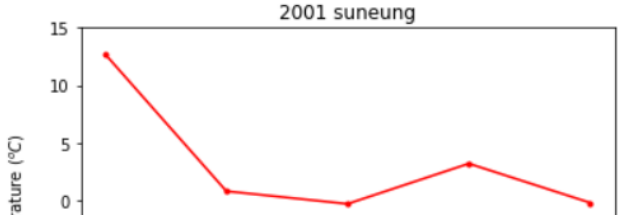

1996 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

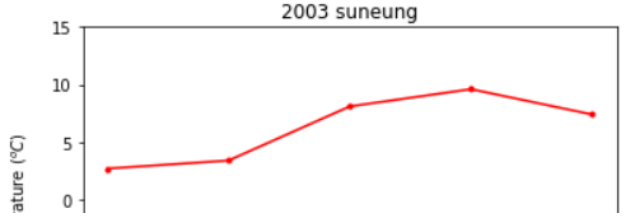

1997 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

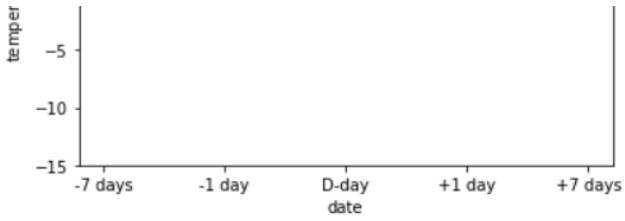

1999 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

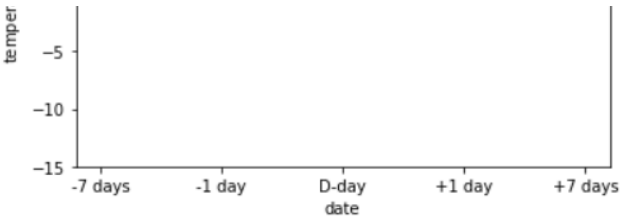

1998 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

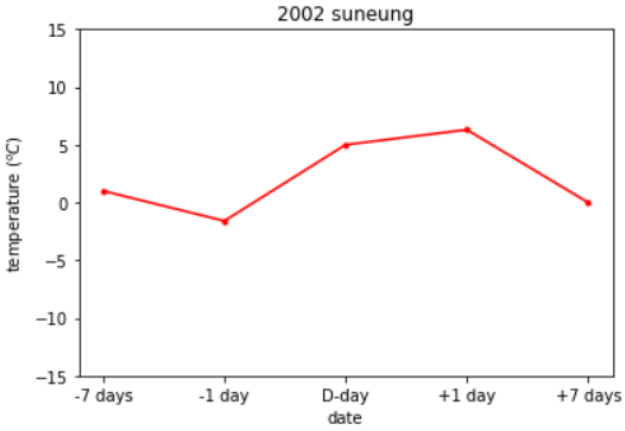

2000 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

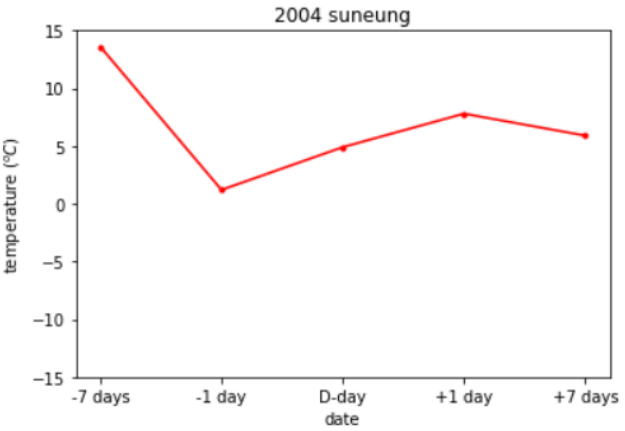

2001 suneung
15
10
5
0

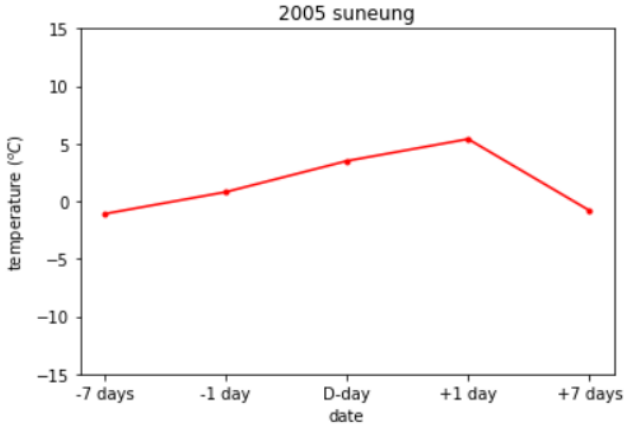

2003 suneung
15
10
5
0

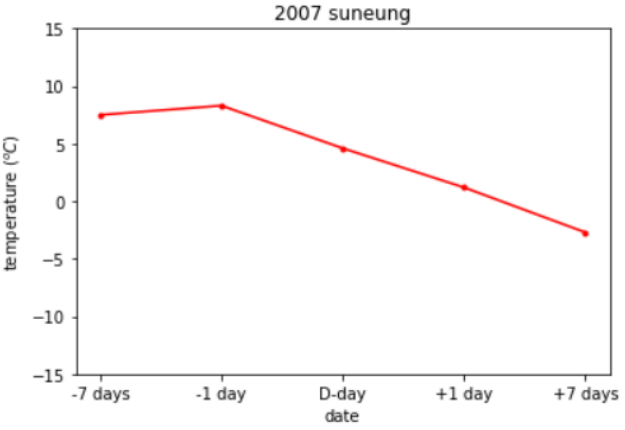

temper
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

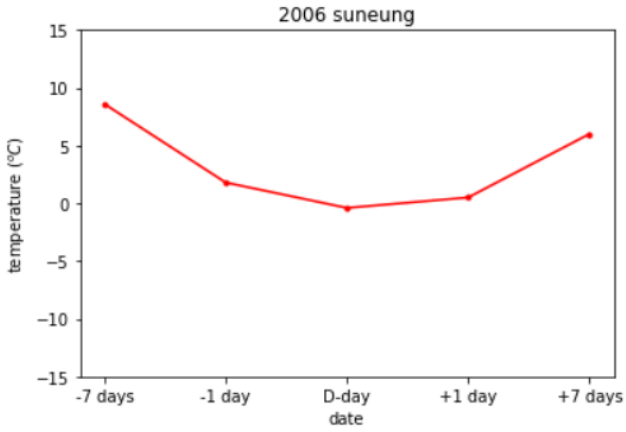

temper
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

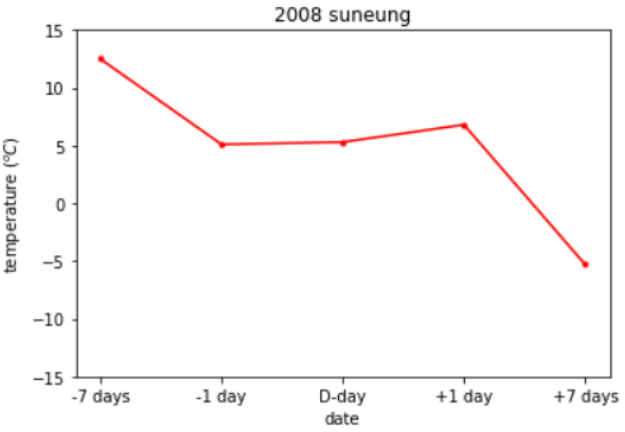

2002 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

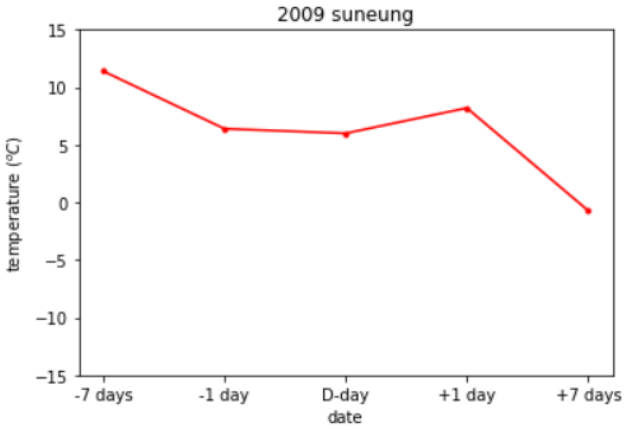

2004 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

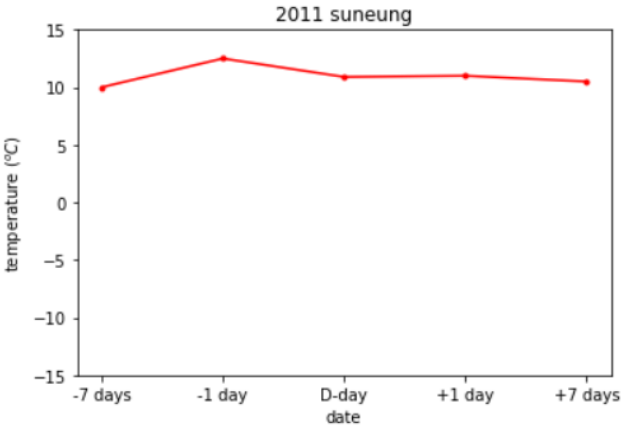

2005 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date

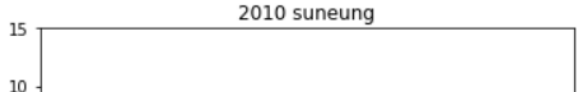

2007 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date


2006 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date


2008 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date


2009 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date


2011 suneung
temperature (°C)
15
10
5
0
−5
−10
−15
-7 days
-1 day
D-day
+1 day
+7 days
date


2010 suneung
15
10


2012 suneung
15
10

(10) 별첨3

```
import datetime
import urllib.request
import json
import time
import matplotlib.pyplot as plt

# [7일 전, 1일 전, 당일, 1일 후, 7일 후]를 따로 요청하니 시간이 너무 오래 걸림. 기간별로 받아서 해당 날짜의 데이터만 추출하는 함수
# 시작일과 종료일을 파라미터로 받아 요청 URL을 만들고, 자료를 요청함.
def apiCall(startDate, endDate):
    url = 'http://data.kma.go.kr/apiData/getData?type=json&dataCd=ASOS&dateCd=DAY&startDt='+startDate+'&endDt='+endDate+'&stnIds=108&schListCnt=10&pageIndex=1&apiKey=ptZTV0vvtZ0pecITeh0SBcmD3c7NI1KgoaIpHLmmH3G0hezv3uhvgLyIsna%2BKS2Q'
    #JSON 포맷 데이터를 파싱해서 사전 데이터로 만들어주는 라이브러리 사용
    request = urllib.request.Request(url)
    response = urllib.request.urlopen(request)
    response_body = response.read()

    data = json.loads(response_body)
    time.sleep(1)
    return data


#suDate = 역대 수능일자
suDate = [19931116,19941123,19951122,19961113,19971119,19981118,19991117,20001115,20011107,20021106,20031105,20041117,20051123,20061116,20071115,20081113,20091112,20101118,20111110,20121108,20131107,20141113,20151112,20161117,20171123,20181115]
#listMIN_TA = 각 수능별 [7일 전, 1일 전, 당일, 1일 후, 7일 후] 최저 기온
listMIN_TA = [[0 for j in range(0,5)]for i in range(0,len(suDate))]

for i in range(len(listMIN_TA)):
    # 수능 시즌별 최저 기온 데이터를 받기위한 날짜
    d = datetime.date(round(suDate[i]/10000), round(suDate[i]%10000/100), suDate[i]%100)
    startDate = (d + datetime.timedelta(days=-7)).strftime('%Y%m%d')
    endDate = (d + datetime.timedelta(days=+7)).strftime('%Y%m%d')

    #요청 URL을 만들고, 반복 요청을 통해 각 수능 시즌별 최저기온 데이터를 리스트에 저장함.
    frontData = apiCall(startDate, d.strftime('%Y%m%d')) # 7일 전~당일
    backData = apiCall((d + datetime.timedelta(days=+1)).strftime('%Y%m%d'), endDate) # 1일 후~7일 후

    listMIN_TA[i][0] = frontData[3]['info'][0]['MIN_TA'] # 7일 전 최저 기온
    listMIN_TA[i][1] = frontData[3]['info'][6]['MIN_TA'] # 1일 전 최저 기온
    listMIN_TA[i][2] = frontData[3]['info'][7]['MIN_TA'] # 당일 최저 기온
    listMIN_TA[i][3] = backData[3]['info'][0]['MIN_TA'] # 1일 후 최저 기온
    listMIN_TA[i][4] = backData[3]['info'][6]['MIN_TA'] # 1일 후 최저 기온

# 최저기온 리스트 출력
print(listMIN_TA)
print()
```

```
# 그래프 출력
for i in range(len(listMIN_TA)):
    y_value = listMIN_TA[i]
    x_name=('-7 days', '-1 day', 'D-day', '+1 day', '+7 days')

    plt.plot(x_name,y_value,'r.-')

    plt.xlabel('date')
    plt.ylabel('temperature ($^oC$)')
    plt.title(str(suDate[i])[:4]+' suneung')
    plt.ylim(-15, 15)
```