코로나로 인한 서울 주말 지하철 유동인구 분석

2019111569_경영정보학과_허재원

2020년 1월 20일 국내에서 첫 코로나 확진자가 생겼고 1차 대유행, 2차 대유행, 3차 대유행을 겪으며 거리두기 3단계까지 겪으며 현재는 수도권의 경우 10시 이후에는 셧다운 상황에 이르렀습니다. 분석을 실시한 현재 6월 15일까지도 매일 꾸준하게 400~500명의 확진자가 나오는 상황이고 수도권에서도 300~400명의 확진자가 나오고 있는 상황입니다. 저는 이러한 상황을 기반으로 코로나가 시작된 이후로부터 서울의 누적확진자가 가장 많은 지역 top5 중에도 유동인구가 가장 많은 각각의 지하철역의 하차승객수가 어떻게 변했는지를 살펴보고자 합니다.

분석 개요 및 활용데이터

먼저, 저는 현재까지 서울의 자치구별 누적 코로나 확진자 수를 통해 가장 누적확진자 수가 높은 5개의 자치구를 걸러낸 후, 각각의 역들을 추출해 교통카드 데이터를 통해 top5 자치구 별 2020년 1월부터 현재까지 누적 하차승객 수가 가장 높은 역들을 하나씩 추출해 낼 계획입니다. 그리고 그 역들에서 현재까지의 하차 승객수의 시계열 데이터를 분석해보고자 합니다.

이 분석을 통해서 코로나가 대유행한 시점에서 사람들의 주말 지하철 탑승객 수는 어떻게 변화했는지 살펴볼 것이고, 현재는 어떤 방향으로 그 추세가 진행중인지도 알아보고자 합니다. 이를 통해, 단순한 시계열적인 흐름뿐만 아니라 추세를 확인해 어떤 구의 어떤 역에 방역을 어떻게 실시해야 할지도 간단하게 알아볼 수 있을 것이라고 생각합니다.

데이터 정보

공개일자	2014.12.16	최신수정일자	2021.02.18
갱신주기	비정기(자료변경시)	분류	교통
원본시스템	서울교통공사 지하철정보 바로가기	저작권자	서울교통공사
제공기관	서울교통공사	제공부서	IT기획처
담당자	김성은 (02-6311-9345)		
원본형태	File	제3저작권자	없음
라이센스	저작권자표시(BY) : 이용이나 변경 및 2차적 저작물의 작성을 포함한 자유이용을 허락합니다.		
관련태그	역, 지하철, 서울, 1호선, 2호선, 3호선, 4호선		

데이터 정보

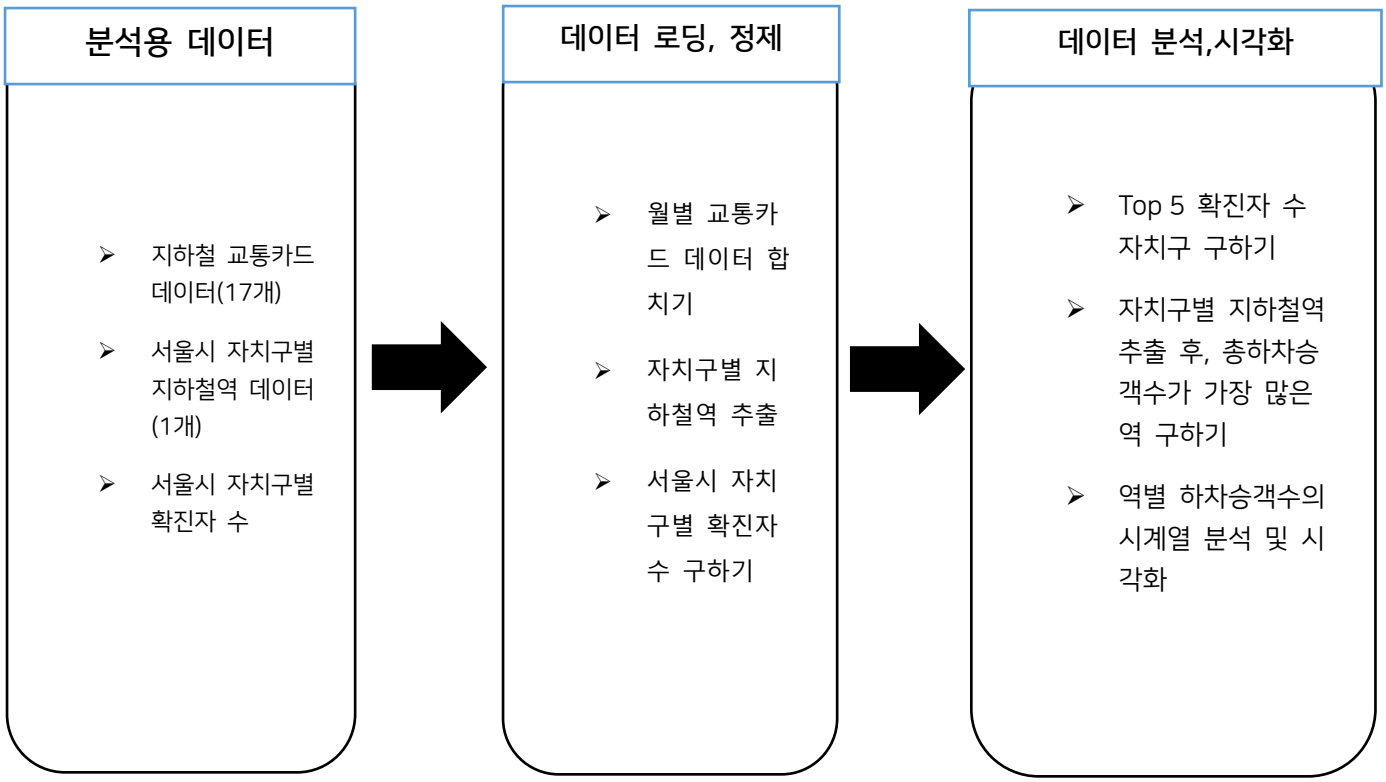
공개일자	2016.01.04	최신수정일자	2021.06.15
갱신주기	매일	분류	교통
원본시스템	교통카드 정산시스템	저작권자	서울특별시, 코레일, 공항철도
제공기관	서울특별시	제공부서	도시교통실 교통기획관 교통정책과
담당자	이윤정 (02-2133-2236)		
원본형태	DB	제3저작권자	없음
라이센스	저작권자표시(BY) : 이용이나 변경 및 2차적 저작물의 작성을 포함한 자유이용을 허락합니다.		
관련태그	지하철, 교통, 통계, 지하철역, 승차, 하차, 지하철호선, 중앙선, 경전철, 경의선		

데이터 정보

공개일자	2020.08.13	최신수정일자	2021.06.15
갱신주기	매일	분류	보건
원본시스템	서울시 홈페이지 바로가기	저작권자	서울특별시
제공기관	서울특별시	제공부서	시민소통기획관 뉴미디어담당관
담당자	정혜선 (02-2133-6507)		
원본형태	DB	제3저작권자	없음
라이센스	저작권자표시(BY) : 이용이나 변경 및 2차적 저작물의 작성을 포함한 자유이용을 허락합니다.		
관련태그	코로나, 코로나19		

차례대로, 분석에 활용한 데이터 셋은 서울교통공사에서 제공한 자치구별 지하철역 정보입니다. 이를 통해, 누적확진자 수가 높은 자치구들을 선별한 후 자치구에 있는 지하철역들을 추출해낼 수 있습니다. 두번째는 교통카드 정산시스템으로 2020년 1월부터 2021년 6월까지 서울내의 지하철역에서의 승하차 승객수가 기록된 데이터 셋입니다. 이를 통해 역별 총 하차 승객 수를 추출해낼 수 있습니다. 마지막은 서울시의 자치구 별로 매일 기록된 확진자 수, 경로, 확진일 등이 담긴 데이터 셋입니다. 이를 통해 현재까지의 누적확진자 수가 가장 높은 자치구들을 선별해 낼 수 있을 것입니다.

분석프로세스



자치구별 코로나 확진자 수 top5 구하기

먼저 서울시의 확진자 현황을 담고 있는 데이터를 로딩하여 데이터셋을 head, tail, str 함수를 통해 간단하게 살펴보았습니다. 살펴본 결과 14개의 컬럼과 46732개의 데이터가 있음을 알 수 있었습니다.

각각의 컬럼들은 id와 비슷한 역할을 하는 연변, 확진일, 지역, 접촉력, 등록일, 이동경로, 상태 등을 나타내고 있음을 알 수 있습니다. 그 중에서 환자번호, 국적, 환자번호 같은 경우에는 NA, 즉 결측치로 나타내져있음을 알 수 있습니다. 사용하지 않는 컬럼이기에 신경쓰지 않아도 됩니다.

```
C:/r_project/r_project/data/
> setwd('C:\\r_project\\r_project\\data')
> seoul <- read.csv("서울시.csv", header = T)
> head(seoul)
    연번       확진일 환자번호 국적 환자정보
1 46732 2021-06-14       NA   NA       NA
2 46731 2021-06-14       NA   NA       NA
3 46730 2021-06-14       NA   NA       NA
4 46729 2021-06-14       NA   NA       NA
5 46728 2021-06-14       NA   NA       NA
6 46727 2021-06-14       NA   NA       NA
     지역       여행력                 접촉력
1  타시도                감염경로 조사중
2  성동구                감염경로 조사중
3  중랑구                감염경로 조사중
4  강동구                감염경로 조사중
5  광진구 카자흐스탄     감염경로 조사중
6  강동구              타시도 확진자 접촉
   조치사항 상태 이동경로           등록일
1        NA    -       NA 2021-06-15 9:52
2        NA    -       NA 2021-06-15 9:52
3        NA    -       NA 2021-06-15 9:52
4        NA    -       NA 2021-06-15 9:52
5        NA    -       NA 2021-06-15 9:52
6        NA    -       NA 2021-06-15 9:52
            수정일 노출여부
1 2021-06-15 9:52        Y
2 2021-06-15 9:52        Y
3 2021-06-15 9:52        Y
4 2021-06-15 9:52        Y
5 2021-06-15 9:52        Y
6 2021-06-15 9:52        Y

> str(seoul)
'data.frame':   46732 obs. of  14 variables:
 $ 연번     : int  46732 46731 46730 46729 46
728 46727 46726 46725 46724 46723 ...
 $ 확진일   : chr  "2021-06-14" "2021-06-14"
 "2021-06-14" "2021-06-14" ...
 $ 환자번호: logi  NA NA NA NA NA NA ...
 $ 국적     : logi  NA NA NA NA NA NA ...
 $ 환자정보: logi  NA NA NA NA NA NA ...
 $ 지역     : chr  "타시도" "성동구" "중랑구" "강
동구" ...
 $ 여행력   : chr  "" "" "" "" ...
 $ 접촉력   : chr  "감염경로 조사중" "감염경로 조
사중" "감염경로 조사중" "감염경로 조사중" ...
 $ 조치사항: logi  NA NA NA NA NA NA ...
 $ 상태     : chr  "-" "-" "-" "-" ...
 $ 이동경로: logi  NA NA NA NA NA NA ...
 $ 등록일   : chr  "2021-06-15 9:52" "2021-06-
15 9:52" "2021-06-15 9:52" "2021-06-15 9:52"
 ...
 $ 수정일   : chr  "2021-06-15 9:52" "2021-06-
15 9:52" "2021-06-15 9:52" "2021-06-15 9:52"
 ...
 $ 노출여부: chr  "Y" "Y" "Y" "Y" ...
```

이러한 데이터들을 불러온 후엔 seoul2라는 변수를 만들어 seoul변수에서 dplyr 패키지의 group_by 함수를 이용해 지역별로 묶어준 후에 빈도수를 구하는 n()함수를 통해 각각 자치구별 누적확진자 수를 요약해 구한 후 arrange 함수로 확진자 수를 기준으로 내림차순 정렬한 후

head함수를 통해 5개만 추출해 담았습니다. 확인해본 결과 현재까지 누적 확진자 수는 송파구, 강남구, 강서구, 타시도, 노원구가 가장 높게 나왔음을 알 수 있습니다. 서울 자치구만 살펴보기로 했으므로 타시도는 제외시켜주기로 하겠습니다.

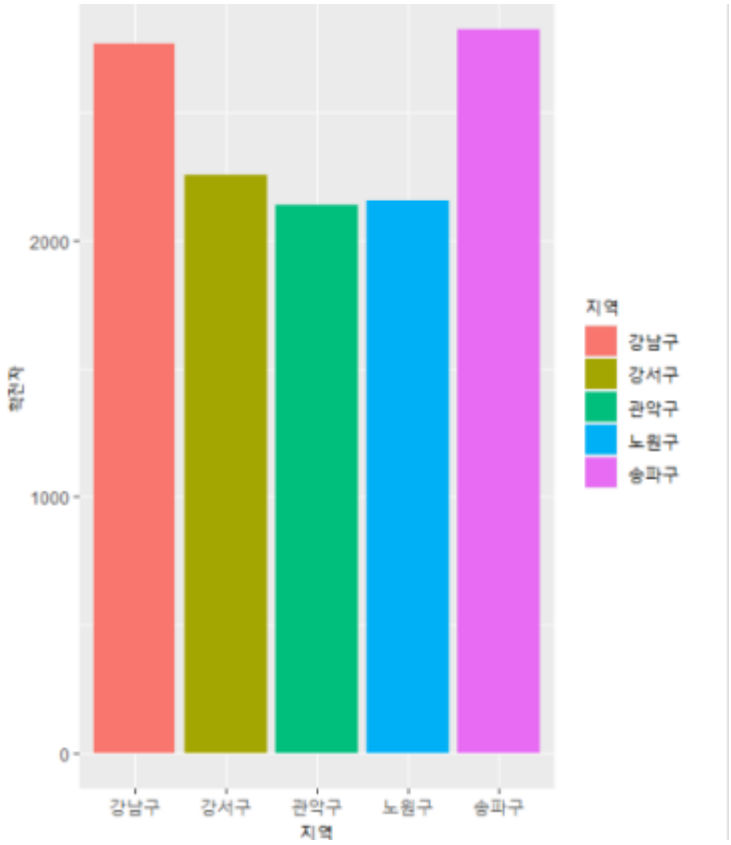
```
> seoul2 <- seoul %>%
+     group_by(지역) %>%
+     summarise(확진자 = n()) %>%
+     arrange(desc(확진자)) %>%
+     head(5)
> head(seoul2)
# A tibble: 5 x 2
  지역   확진자
  <chr>   <int>
1 송파구   2825
2 강남구   2769
3 강서구   2255
4 타시도   2190
5 노원구   2155
```

Filter 함수를 통해 지역에서 타시도는 제외하고 다시 내림차순 정렬 후 5개의 자치구를 뽑았습니다. 결과를 보면 누적 확진자 수가 높은 자치구는 각각 송파구, 강남구, 강서구, 노원구, 관악구임을 알 수 있습니다.

```
> seoul3 <- seoul %>%
+     group_by(지역) %>%
+     filter(지역 != '타시도') %>%
+     summarise(확진자 = n()) %>%
+     arrange(desc(확진자)) %>%
+     head(5)
> seoul3
# A tibble: 5 x 2
  지역   확진자
  <chr>   <int>
1 송파구   2825
2 강남구   2769
3 강서구   2255
4 노원구   2155
5 관악구   2138
```

Ggplot2 패키지의 ggplot을 활용해 x 축은 지역, y축은 확진자를 두고 지역별로 그래프의 색깔을 다르게 설정해 시각화를 해보았습니다.

```
> ggplot(seoul3, aes(x = 지역, y = 확진자, fi
ll = 지역)) + geom_bar(stat = 'identity', yl
im = c(0, 5000))
```

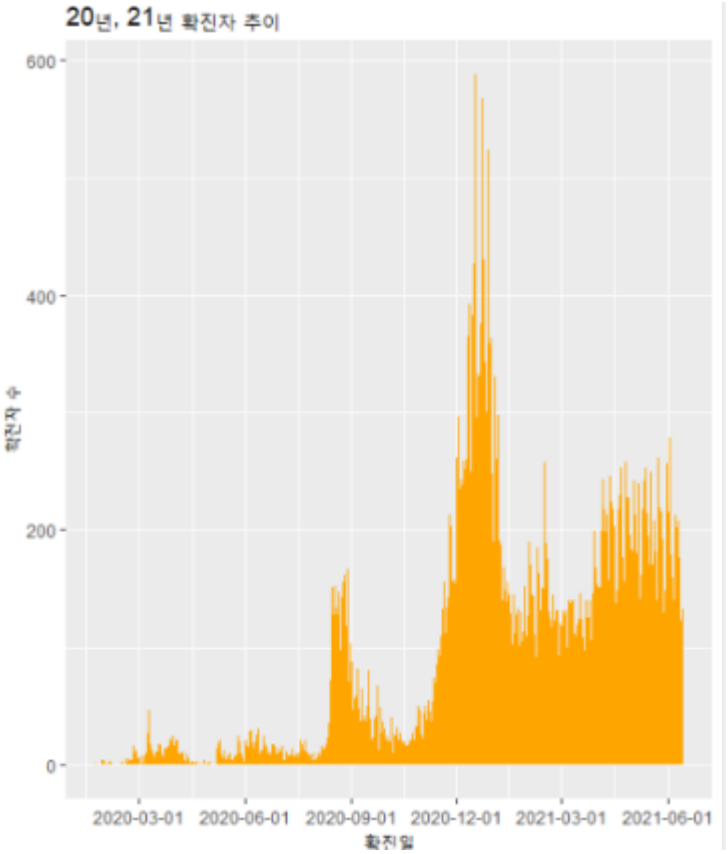


이 데이터를 활용해 서울의 전체 일자별 확진자 추이도 함께 보았습니다. 먼저 seoul 데이터프레임의 확진일이 int 형이었으므로 lubridate 패키지의 ymd를 통해 Date 형식으로 변환해주었습니다. 그 후 df 변수를 만들어 확진일, 지역, 접촉력 별로 그룹화 하고 count 변수에 n()함수로 나온 빈도 값들을 넣어주었습니다.

```
> seoul$확진일 <- ymd(seoul$확진일)
> df <- seoul %>%
+     group_by(확진일, 지역, 접촉력) %>%
+     summarise(count = n())
`summarise()` has grouped output by '확진일',
 '지역'. You can override using the `.groups
` argument.
> df
# A tibble: 22,070 x 4
# Groups:   확진일, 지역 [8,222]
   확진일     지역     접촉력          count
   <date>     <chr>    <chr>           <int>
 1 2020-01-24 강서구   해외유입            1
 2 2020-01-30 마포구   해외유입            1
 3 2020-01-30 종로구   종로구 집단~        1
 4 2020-01-30 중랑구   해외유입            1
 5 2020-01-31 성북구   기타 확진자 ~       1
 6 2020-01-31 종로구   종로구 집단~        2
```

이렇게 만든 데이터 셋으로 ggplot에서의 geom_col()을 이용해 20년부터 21년까지의 서울 모든 지역의 확진자 추이를 시각해보았습니다. 여기서 scale_x_date()를 통해 그래프에 3개월 단위로 나타내고자 했습니다. 이를 통해 서울 전제에서 흐름을 보면, 첫번째로 3월, 6월에 증가했고 9월에 소폭 증가했으며 2020년 말부터 2021년 초까지는 이전보다 훨씬 많이 그 확진자 수가 증가했음을 알 수 있었습니다.

```
> ggplot(data = df) +
+     geom_col(aes(x = 확진일, y = count), fi
ll = 'orange')+
+     scale_x_date(breaks = "3 month")+
+     labs(x = '확진일', y = '확진자 수', titl
e = '20년, 21년 확진자 추이')
> |
```



Top 5 자치구에 속하는 지하철역들 추출

먼저, working directory를 다시 한번 데이터가 들어있는 파일로 재 설정한 후 readxl 패키지의 read_excel()함수를 통해 자치구별 지하철역 현황을 subway_station 변수에 담았습니다. 그 다음 str()와 dim() 함수를 통해 278개의 객체와 7개의 변수가 존재함을 알 수 있었습니다.

```
> setwd("C:\\r_project\\r_project\\data")
> subway_station <- read_excel("역별_주소현황.x
lsx", sheet = 1, col_names = T)
New names:
* `` -> ...7
> str(subway_station)
tibble[,7] [278 x 7] (S3: tbl_df/tbl/data.fra
me)
 $ 연번     : num [1:278] 1 2 3 4 5 6 7 8 9 10
 ...
 $ 역명     : chr [1:278] "서울" "시청" "종각"
 "종로3가" ...
 $ 호선     : chr [1:278] "1" "1" "1" "1" ...
 $ 구       : chr [1:278] "중구" "중구" "종로구"
 "종로구" ...
 $ 동       : chr [1:278] "봉래동" "정동" "종로1
가" "종로3가" ...
 $ 상세주소: chr [1:278] "서울특별시 중구 봉래동2
가 122 서울역(1호선)" "서울특별시 중구 정동 5-5 시
청역(1호선)" "서울특별시 종로구 종로1가 54 종각지하
철역사" "서울특별시 종로3가 10-5  1호선 종로3가역(1
호선)" ...
 $ ...7     : chr [1:278] "6110-1331" "6110-13
21" "6110-1311" "6110-1301" ...
> dim(subway_station)
[1] 278   7
```

여기서 우리가 필요한 컬럼은 역명과 구 컬럼이므로 2열과 4열을 추출해 df_subway라는 새로운 변수에 담아준다. 후에 filter 함수를 통해 구가 강남, 강서, 관악, 노원, 송파구에 해당하는 데이터들만 추출한다. head 함수를 통해 확인해보면 잘 추출되어 있음을 알 수 있다.

```
> df_subway <- subway_station[c(2, 4)]
> df_subway_top5 <- df_subway %>%
+     filter(구 %in% c('강남구', '강서구', '관악
구', '노원구', '송파구'))
> head(df_subway_top5)
# A tibble: 6 x 2
  역명                 구
  <chr>                <chr>
1 잠실나루             송파구
2 잠실(송파구청)       송파구
3 잠실새내             송파구
4 종합운동장           송파구
5 낙성대               관악구
6 서울대입구(관악구청) 관악구
```

지하철 교통카드 데이터

이 다음엔 지하철 교통카드 데이터를 불러오고자 한다. 먼저 setwd()를 통해 지하철 교통카드 데이터가 담긴 폴더를 설정한 후 flowlist에 dir()함수를 통해 확인해본 데이터 변수명들을 확인해 본다.

```
C:/r_project/r_project/subway/
> setwd('C:\\r_project\\r_project\\subway')
> flowlist <- dir('C:\\r_project\\r_project
\\subway')
> flowlist
 [1] "CARD_SUBWAY_MONTH_202001.csv"
 [2] "CARD_SUBWAY_MONTH_202002.csv"
 [3] "CARD_SUBWAY_MONTH_202003.csv"
 [4] "CARD_SUBWAY_MONTH_202004.csv"
 [5] "CARD_SUBWAY_MONTH_202005.csv"
 [6] "CARD_SUBWAY_MONTH_202006.csv"
 [7] "CARD_SUBWAY_MONTH_202007.csv"
 [8] "CARD_SUBWAY_MONTH_202008.csv"
 [9] "CARD_SUBWAY_MONTH_202009.csv"
[10] "CARD_SUBWAY_MONTH_202010.csv"
[11] "CARD_SUBWAY_MONTH_202011.csv"
[12] "CARD_SUBWAY_MONTH_202012.csv"
[13] "CARD_SUBWAY_MONTH_202101.csv"
[14] "CARD_SUBWAY_MONTH_202102.csv"
[15] "CARD_SUBWAY_MONTH_202103.csv"
[16] "CARD_SUBWAY_MONTH_202104.csv"
[17] "CARD_SUBWAY_MONTH_202105.csv"
```

flow_nm이라는 변수에는 sub_str()함수를 통해 flowlist에 담겨있는 값들 중 각각 뒤에 붙어있는 .csv문자열을 빼고 추출해서 새로 담았다.

```
> flow_nm <- substr(flowlist, 1, nchar(flowli
st)-4)
> flow_nm
 [1] "CARD_SUBWAY_MONTH_202001"
 [2] "CARD_SUBWAY_MONTH_202002"
 [3] "CARD_SUBWAY_MONTH_202003"
 [4] "CARD_SUBWAY_MONTH_202004"
 [5] "CARD_SUBWAY_MONTH_202005"
 [6] "CARD_SUBWAY_MONTH_202006"
 [7] "CARD_SUBWAY_MONTH_202007"
 [8] "CARD_SUBWAY_MONTH_202008"
 [9] "CARD_SUBWAY_MONTH_202009"
[10] "CARD_SUBWAY_MONTH_202010"
[11] "CARD_SUBWAY_MONTH_202011"
[12] "CARD_SUBWAY_MONTH_202012"
[13] "CARD_SUBWAY_MONTH_202101"
[14] "CARD_SUBWAY_MONTH_202102"
[15] "CARD_SUBWAY_MONTH_202103"
[16] "CARD_SUBWAY_MONTH_202104"
[17] "CARD_SUBWAY_MONTH_202105"
```

교통카드 데이터를 불러오는 과정이다. 월별로 나눠져 있기에 for문을 이용해서 불러올 수도 있으나 파일이 저장되어있던 인코딩 형식이 각각 ANSI 또는 UTF-8-BOM으로 다르기에 하나씩 불러오도록 한다. 이때 불러오면서 데이터프레임의 변수명들은 flow_nm에 담겨있는 값들과 똑같이 설정해준다.

```
> CARD_SUBWAY_MONTH_202009 = read.csv('CARD_
SUBWAY_MONTH_202009.csv', fileEncoding = "CP
949")
> View(CARD_SUBWAY_MONTH_202009)
>
> CARD_SUBWAY_MONTH_202010 = read.csv('CARD_
SUBWAY_MONTH_202010.csv', fileEncoding = "UT
F-8-BOM")
> View(CARD_SUBWAY_MONTH_202010)
>
> CARD_SUBWAY_MONTH_202011 = read.csv('CARD_
SUBWAY_MONTH_202011.csv', fileEncoding = "UT
F-8-BOM")
> View(CARD_SUBWAY_MONTH_202011)
>
> CARD_SUBWAY_MONTH_202012 = read.csv('CARD_
SUBWAY_MONTH_202012.csv', fileEncoding = "UT
F-8-BOM")
> View(CARD_SUBWAY_MONTH_202012)
>
> CARD_SUBWAY_MONTH_202101 = read.csv('CARD_
SUBWAY_MONTH_202101.csv', fileEncoding = "UT
F-8-BOM")
> View(CARD_SUBWAY_MONTH_202101)
>
> CARD_SUBWAY_MONTH_202102 = read.csv('CARD_
SUBWAY_MONTH_202102.csv', fileEncoding = "UT
F-8-BOM")
> View(CARD_SUBWAY_MONTH_202102)
>
> CARD_SUBWAY_MONTH_202103 = read.csv('CARD_
SUBWAY_MONTH_202103.csv', fileEncoding = "UT
F-8-BOM")
> View(CARD_SUBWAY_MONTH_202103)
>
> CARD_SUBWAY_MONTH_202104 = read.csv('CARD_
SUBWAY_MONTH_202104.csv', fileEncoding = "UT
F-8-BOM")
> View(CARD_SUBWAY_MONTH_202104)
>
> CARD_SUBWAY_MONTH_202105 = read.csv('CARD_
SUBWAY_MONTH_202105.csv', fileEncoding = "UT
F-8-BOM")
```

잘 담겨져 있는지 확인하기 위해 ls()함수를 통해 메모리상에 저장된 변수명들을 확인한 후 for문을 돌려 각각 몇 행과 몇 열로 이뤄져있는지 보았다. 6열과 7열로 나누어져있는데 여기서 7열인

경우는 필요없는 열이기에 없애야 하는 열이다. 결과들을 확인해보면 다 잘 담겨있음을 알 수 있다.

```
> ls()
 [1] "CARD_SUBWAY_MONTH_202001"
 [2] "CARD_SUBWAY_MONTH_202002"
 [3] "CARD_SUBWAY_MONTH_202003"
 [4] "CARD_SUBWAY_MONTH_202004"
 [5] "CARD_SUBWAY_MONTH_202005"
 [6] "CARD_SUBWAY_MONTH_202006"
 [7] "CARD_SUBWAY_MONTH_202007"
 [8] "CARD_SUBWAY_MONTH_202008"
 [9] "CARD_SUBWAY_MONTH_202009"
[10] "CARD_SUBWAY_MONTH_202010"
[11] "CARD_SUBWAY_MONTH_202011"
[12] "CARD_SUBWAY_MONTH_202012"
[13] "CARD_SUBWAY_MONTH_202101"
[14] "CARD_SUBWAY_MONTH_202102"
[15] "CARD_SUBWAY_MONTH_202103"
[16] "CARD_SUBWAY_MONTH_202104"
[17] "CARD_SUBWAY_MONTH_202105"
[18] "df"
[19] "df_subway"
[20] "df_subway_top5"
[21] "flow_nm"
[22] "flowlist"
[23] "seoul"
[24] "seoul2"
[25] "seoul3"
[26] "subway_station"
>
> for(ivar in flow_nm){
+     cat(ivar, "->", dim(get(ivar))[1],
 '행', dim(get(ivar))[2], '열', '\n')
+ }
CARD_SUBWAY_MONTH_202001 -> 18312 행 6 열
CARD_SUBWAY_MONTH_202002 -> 17118 행 6 열
CARD_SUBWAY_MONTH_202003 -> 18265 행 6 열
CARD_SUBWAY_MONTH_202004 -> 17722 행 6 열
CARD_SUBWAY_MONTH_202005 -> 18327 행 7 열
CARD_SUBWAY_MONTH_202006 -> 17739 행 7 열
CARD_SUBWAY_MONTH_202007 -> 18329 행 7 열
CARD_SUBWAY_MONTH_202008 -> 18363 행 7 열
CARD_SUBWAY_MONTH_202009 -> 17882 행 7 열
CARD_SUBWAY_MONTH_202010 -> 18537 행 7 열
CARD_SUBWAY_MONTH_202011 -> 17943 행 7 열
CARD_SUBWAY_MONTH_202012 -> 18518 행 7 열
```

마지막 7열의 경우 빈값들로 채워진 컬럼명이 GG인 열이기에 각각 select()를 통해 'GG'열을 빼고 추출한 후 다시 저장해준다.

```
> CARD_SUBWAY_MONTH_202005 <- CARD_SUBWAY_MON
TH_202005 %>% select(-('GG'))
>
> CARD_SUBWAY_MONTH_202006 <- CARD_SUBWAY_MON
TH_202006 %>% select(-('GG'))
>
> CARD_SUBWAY_MONTH_202007 <- CARD_SUBWAY_MON
TH_202007 %>% select(-('GG'))
>
> CARD_SUBWAY_MONTH_202008 <- CARD_SUBWAY_MON
TH_202008 %>% select(-('GG'))
>
> CARD_SUBWAY_MONTH_202009 <- CARD_SUBWAY_MON
TH_202009 %>% select(-('GG'))
>
> CARD_SUBWAY_MONTH_202010 <- CARD_SUBWAY_MON
TH_202010 %>% select(-('GG'))
>
> CARD_SUBWAY_MONTH_202011 <- CARD_SUBWAY_MON
TH_202011 %>% select(-('GG'))
>
> CARD_SUBWAY_MONTH_202012 <- CARD_SUBWAY_MON
TH_202012 %>% select(-('GG'))
>
> CARD_SUBWAY_MONTH_202101 <- CARD_SUBWAY_MON
TH_202101 %>% select(-('GG'))
>
> CARD_SUBWAY_MONTH_202102 <- CARD_SUBWAY_MON
TH_202102 %>% select(-('GG'))
>
> CARD_SUBWAY_MONTH_202103 <- CARD_SUBWAY_MON
TH_202103 %>% select(-('GG'))
>
> CARD_SUBWAY_MONTH_202104 <- CARD_SUBWAY_MON
TH_202104 %>% select(-('GG'))
>
> CARD_SUBWAY_MONTH_202105 <- CARD_SUBWAY_MON
TH_202105 %>% select(-('GG'))
```

데이터 정제를 끝낸 후, 각각의 데이터 프레임들은 열 이름, 개수가 같아졌기에 행으로 합쳐준다. 합친 후에는 View()와, str()로 잘 되었는지 확인해준다. 변수는 모두 6개로 똑 같은 열을 기준으로 합쳐졌음을 알 수 있고 총 307572개의 레코드가 들어왔음을 확인할 수 있다.

```
> df_subway_count <- bind_rows(CARD_SUBWAY_MO
NTH_202001, CARD_SUBWAY_MONTH_202002, CARD_SU
BWAY_MONTH_202003, CARD_SUBWAY_MONTH_202004,
 CARD_SUBWAY_MONTH_202005, CARD_SUBWAY_MONTH_
202006, CARD_SUBWAY_MONTH_202007, CARD_SUBWAY
_MONTH_202008, CARD_SUBWAY_MONTH_202009, CARD
_SUBWAY_MONTH_202010, CARD_SUBWAY_MONTH_20201
1,CARD_SUBWAY_MONTH_202012, CARD_SUBWAY_MONTH
_202101, CARD_SUBWAY_MONTH_202102, CARD_SUBWA
Y_MONTH_202103, CARD_SUBWAY_MONTH_202104, CAR
D_SUBWAY_MONTH_202105)
> View(df_subway_count)
> str(df_subway_count)
'data.frame':   307572 obs. of  6 variables:
 $ 사용일자    : int  20200101 20200101 2020010
1 20200101 20200101 20200101 20200101 2020010
1 20200101 20200101 ...
 $ 노선명      : chr  "1호선" "1호선" "우이신설
선" "우이신설선" ...
 $ 역명        : chr  "종각" "시청" "신설동" "보
문" ...
 $ 승차총승객수: num  20427 12126 892 917 2010
 ...
 $ 하차총승객수: num  16301 10516 828 855 2363
 ...
 $ 등록일자    : int  20200104 20200104 2020010
4 20200104 20200104 20200104 20200104 2020010
4 20200104 20200104 ...
>
```

Top 5 자치구별 역 추출

위에서 구한 df_subway_top5, 즉 top5 자치구별 역들이 담겨있는 데이터에서 filter로 송파구와 관련된 역들만 추출한 후 df_subway_songpa 변수에 담는다. 송파구에 있는 역명을 as.vector()를 통해 벡터형으로 변환한 후 다시 변수에 저장한다.

```
> df_subway_songpa <- df_subway_top5 %>%
+   filter(구 == '송파구') %>%
+   select(역명)
> df_subway_songpa <- as.vector(df_subway_son
gpa$역명)
> df_subway_songpa
 [1] "잠실나루"
 [2] "잠실(송파구청)"
 [3] "잠실새내"
 [4] "종합운동장"
 [5] "가락시장"
 [6] "경찰병원"
 [7] "오금"
 [8] "올림픽공원(한국체대)"
 [9] "방이"
[10] "오금"
[11] "개롱"
[12] "거여"
[13] "마천"
[14] "몽촌토성(평화의문)"
[15] "잠실(송파구청)"
[16] "석촌"
[17] "송파"
[18] "가락시장"
[19] "문정"
[20] "장지"
[21] "복정"
```

df_subway_songpa_count에는 월별 총하차승객수가 담겨있는 df_subway_count에서 filter로 위에서 구한 df_subway_songpa 벡터에 담겨있는 역들만 추출해서 넣어준다. Str()함수로 확인해보면 12573개의 레코드가 들어왔음을 알 수 있다.

```
> df_subway_songpa_count <- df_subway_count %
>%
+     filter(역명 %in% df_subway_songpa)
> str(df_subway_songpa_count)
'data.frame':   12573 obs. of  6 variables:
 $ 사용일자    : int  20200101 20200101 2020010
1 20200101 20200101 20200101 20200101 2020010
1 20200101 20200101 ...
 $ 노선명      : chr  "9호선2~3단계" "9호선2~3단
계" "9호선2~3단계" "8호선" ...
 $ 역명        : chr  "올림픽공원(한국체대)" "석
촌" "종합운동장" "복정" ...
 $ 승차총승객수: num  2371 3354 2212 4130 10784
 ...
 $ 하차총승객수: num  2475 2750 1780 3672 10821
 ...
 $ 등록일자    : int  20200104 20200104 2020010
4 20200104 20200104 20200104 20200104 2020010
4 20200104 20200104 ...
```

Df_subway_songpa_count변수에서 역명별로 그룹화한 다음 월별 하차승객수를 모두 더한 후 내림차순 정렬 후 5개의 데이터만 확인해보았다. 그 결과 누적 총하차승객수가 가장 많은 곳은 잠실역임을 알 수 있었다. 즉 잠실역에서 내리는 승객수가 가장 많다는 뜻은 잠실역에서의 유동인구가 가장 많다는 뜻임을 알 수 있다.

```
top1_songpa <- df_subway_songpa_count %>%
    group_by(역명) %>%
    summarise(summit = sum(하차총승객수)) %>%
    arrange(desc(summit)) %>%
    head(5)
top1_songpa
A tibble: 5 x 2
역명              summit
<chr>              <dbl>
잠실(송파구청) 34999927
잠실새내        8862032
문정            8714789
가락시장        7470545
장지            7028023
```

Df_subway_jamsil 변수에는 df_subway_songpa_count에서 잠실역과 관련된 행들만 추출한 후 분석 및 시각화에 사용할 사용일자, 역명, 하차총승객수 컬럼만 추출한다. 총 1034개의 레코드가 추출됐음을 알 수 있다.

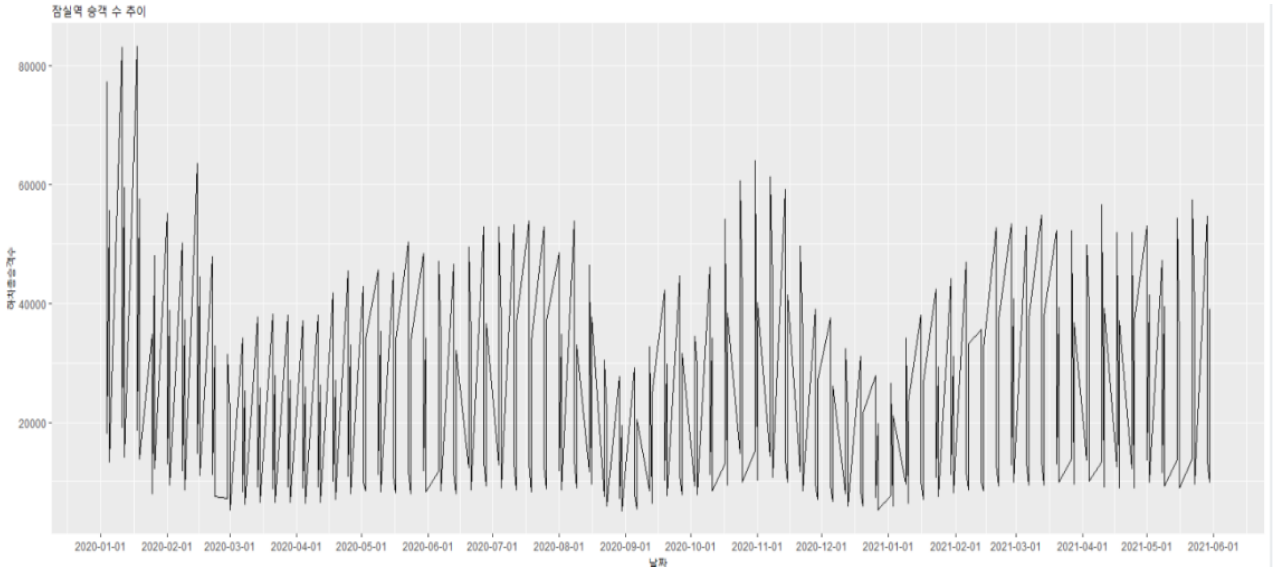
```
> df_subway_jamsil <- df_subway_songpa_count
 %>%
+     filter(역명 == '잠실(송파구청)') %>%
+     select(사용일자, 역명, 하차총승객수)
> head(df_subway_jamsil)
  사용일자            역명 하차총승객수
1 20200101 잠실(송파구청)        13225
2 20200101 잠실(송파구청)        48425
3 20200102 잠실(송파구청)        86659
4 20200102 잠실(송파구청)        19917
5 20200103 잠실(송파구청)        92634
6 20200103 잠실(송파구청)        21398
> str(df_subway_jamsil)
'data.frame':   1034 obs. of  3 variables:
 $ 사용일자    : int  20200101 20200101 2020010
2 20200102 20200103 20200103 20200104 2020010
4 20200105 20200105 ...
 $ 역명        : chr  "잠실(송파구청)" "잠실(송파
구청)" "잠실(송파구청)" "잠실(송파구청)" ...
 $ 하차총승객수: num  13225 48425 86659 19917 9
2634 ...
> df_subway_jamsil$사용일자 <- ymd(df_subway_ja
msil$사용일자)
>
> df_subway_jamsil$weekday <- wday(df_subway_
jamsil$사용일자, label = T)
>
> df_jamsil_wday <- df_subway_jamsil %>%
+     filter(weekday %in% c('토', '일'))
> View(df_jamsil_wday)
> head(df_jamsil_wday)
    사용일자            역명 하차총승객수
1 2020-01-04 잠실(송파구청)        77313
2 2020-01-04 잠실(송파구청)        18111
3 2020-01-05 잠실(송파구청)        55702
4 2020-01-05 잠실(송파구청)        13278
5 2020-01-11 잠실(송파구청)        83099
6 2020-01-11 잠실(송파구청)        19103
  weekday
1      토
2      토
3      일
4      일
5      토
:      :
```

추출한 df_subway_jamsil에서 사용일자를 lubridate의 ymd()를 통해 Date 형으로 변환한 후, lubridate의 wday()를 통해 요일을 표시한 weekday라는 새로운 컬럼을 추가해준다. 그런 다음,

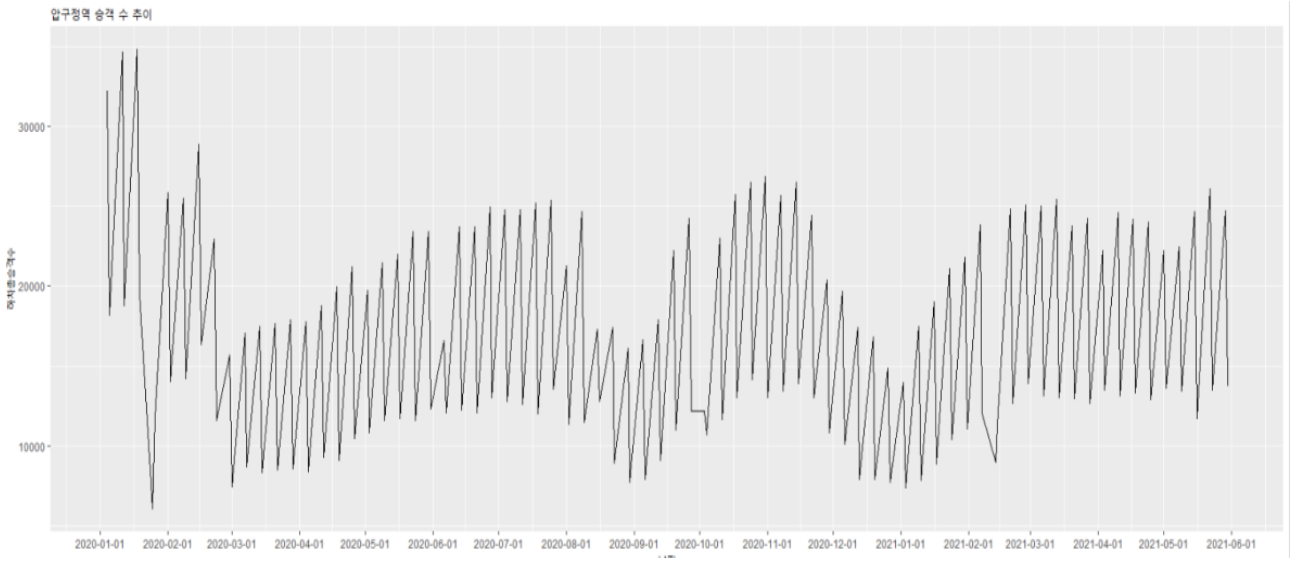
df_jamsil_wday라는 변수에 토요일 일요일만 포함된 행들로 재구성한 후 살펴보았더니 토, 일에 해당하는 잠실역의 하차승객수가 포함된 데이터셋이 만들어졌음을 알 수 있다.

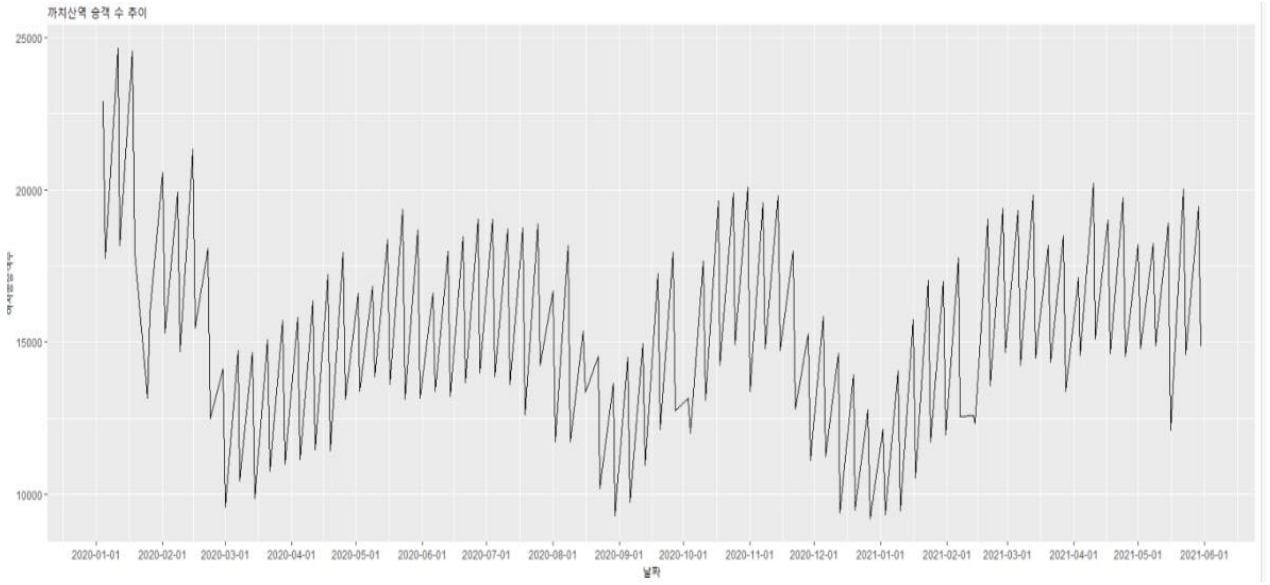
Ggplot의 geom_line()을 통해 주말이 포함된 월별 하차총승객수를 그래프로 나타내보았다.

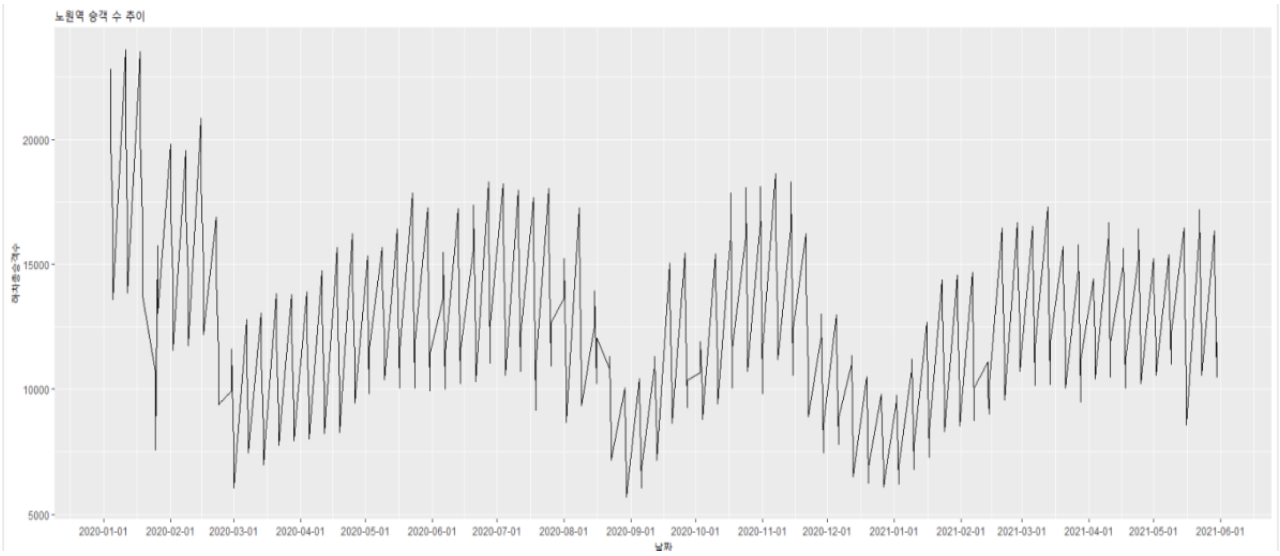
```
ggplot(data = df_jamsil_wday)+
    geom_line(aes(x = 사용일자, y = 하차총승객수), fill = 'orange') +
    scale_x_date(breaks = "1 month")+
    labs(x = '날짜', '하차 승객 수', title = '잠실역 승객 수 추이')
```

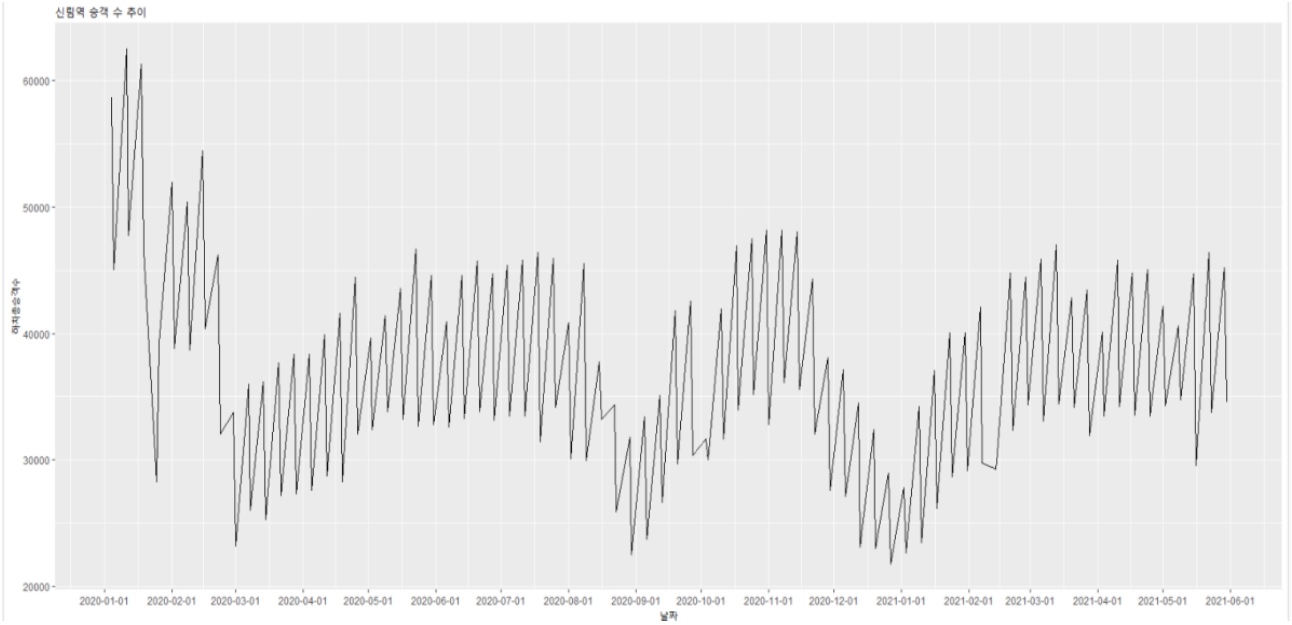


똑 같은 방식으로 남은 강남구, 강서구, 노원구, 관악구를 분석한 후 각각 유동인구가 많은 압구정역, 까치산역, 노원역, 신림역을 기준으로 시각화해보았다.




까치산역 승객 수 추이
하차승객수
25000
20000
15000
10000
2020-01-01
2020-02-01
2020-03-01
2020-04-01
2020-05-01
2020-06-01
2020-07-01
2020-08-01
2020-09-01
2020-10-01
2020-11-01
2020-12-01
2021-01-01
2021-02-01
2021-03-01
2021-04-01
2021-05-01
2021-06-01
날짜


노원역 승객 수 추이
하차승객수
20000
15000
10000
5000
2020-01-01
2020-02-01
2020-03-01
2020-04-01
2020-05-01
2020-06-01
2020-07-01
2020-08-01
2020-09-01
2020-10-01
2020-11-01
2020-12-01
2021-01-01
2021-02-01
2021-03-01
2021-04-01
2021-05-01
2021-06-01
날짜


신림역 승객 수 추이
하차승객수
60000
50000
40000
30000
20000
2020-01-01
2020-02-01
2020-03-01
2020-04-01
2020-05-01
2020-06-01
2020-07-01
2020-08-01
2020-09-01
2020-10-01
2020-11-01
2020-12-01
2021-01-01
2021-02-01
2021-03-01
2021-04-01
2021-05-01
2021-06-01
날짜

결론

분석 및 시각화를 통해 도출할 수 있는 결론은 2020년 1월 1일 이전에는 각각의 역을 통해 놀러가는 사람들이 많다가 2월 3월 9월 2021년 1월에 각각 지하철 하차 승객수가 감소하면서 최저점을 찍는 경향을 보였다. 이는 모든 역에서 똑 같은 그래프의 변화를 통해 확인할 수 있었다. 신천지, 사랑제일교회, 이태원발 클럽 감염, 2020년 연말부터 현재까지 이어져오는 수도권에서의 산발적인 감염을 고려했을 때 수도권에서의 대유행시에 주말에 놀러나가거나 지하철을 타는 승객들의 수가 적음을 알 수 있다. 하지만 현재, 언론보도에서도 코로나 소식을 접하기 어렵고 백신 접종과 같은 상황 때문에 2021년 6월 현재까지 다시 서서히 증가하고 있음을 알 수 있다.

우리는 이를 통해 각각 하차승객수, 즉 각 자치구별로 유동인구가 가장 많은 역들을 중심으로 방역거점을 세우고 지하철이용과 관련된 방역규정도 다시 세울필요가 있다고 본다. 앞으로 언제 끝날지 모르는 코로나지만 만약 코로나가 계속 산발적인 감염을 통해 수도권에서 유행하는 상황에서 그래프가 계속 증가 추세를 보이게 된다면 각별한 주의가 필요하다고도 생각한다.

Wrap up

```
> sessionInfo()
R version 4.0.5 (2021-03-31)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 19041)

Matrix products: default

locale:
[1] LC_COLLATE=Korean_Korea.949  LC_CTYPE=Korean_Korea.949
[3] LC_MONETARY=Korean_Korea.949 LC_NUMERIC=C
[5] LC_TIME=Korean_Korea.949

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] ggplot2_3.3.3    stringr_1.4.0    lubridate_1.7.10 readxl_1.3.1
[5] dplyr_1.0.5

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.6        cellranger_1.1.0 pillar_1.6.0     compiler_4.0.5
 [5] tools_4.0.5       digest_0.6.27    lifecycle_1.0.0  tibble_3.1.1
 [9] gtable_0.3.0      pkgconfig_2.0.3  rlang_0.4.11     DBI_1.1.1
[13] cli_2.5.0         rstudioapi_0.13  xfun_0.22        withr_2.4.2
[17] generics_0.1.0    vctrs_0.3.8      grid_4.0.5       tidyselect_1.1.1
[21] glue_1.4.2        R6_2.5.0         fansi_0.4.2      purrr_0.3.4
[25] farver_2.1.0      magrittr_2.0.1   scales_1.1.1     ellipsis_0.3.2
[29] assertthat_0.2.1 colorspace_2.0-1 labeling_0.4.2   utf8_1.2.1
[33] tinytex_0.31      stringi_1.5.3    munsell_0.5.0    crayon_1.4.1
>
```

Sessioninfo()함수를 통해 분석에 사용한 도구들을 확인해보았다. package로는 그래프를 그리기 위한 ggplot2, 문자열 추출 및 분석을 위한 stringr, 날짜 데이터를 분석하고 정제하기 위한 lubridate, 데이터 전처리를 위한 dplyr, 엑셀 파일을 불러오기 위한 readxl 패키지를 사용했다.