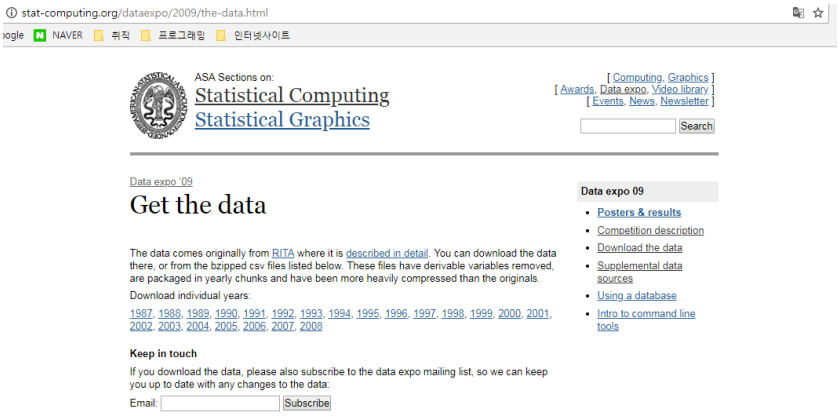
1. 데이터를 수집 한다.(공공데이터, 서울시 공공데이터)
2. 데이터를 빅데이터 시스템(하둡 및 데이터베이스) 저장
3. R을 통해 데이터를 탐색 및 분석 한다
4. 데이터 분석 화면을 구현 한다.

1. 데이터를 수집 한다.(공공데이터, 서울시 공공데이터)



→미국 airline_delay 2006~2008까지 데이터 수집

1. 데이터를 빅데이터 시스템(하둡 및 데이터베이스) 저장

태이블 생성

```
Logging initialized using configuration in
/lib/hive-common-1.0.1.jar!/hive-log4j.pro
hive> CREATE TABLE airline_delay(
    > Year INT,
    > MONTH INT,
    > DayofMonth INT,
    > DayofWeek INT,
    > DepTime INT,
    > CRSDepTime INT,
    > ArrTime INT,
    > CRSArrTime INT,
    > UniqueCarrier STRING,
    > FlightNum INT,
    > TailNum STRING,
    > ActualElapsedTime INT,
    > CRSElapsedTime INT,
```

DATA load 를 해준다

```
Loading data to table default.airline_delay partition (delayyear=20
08)
Partition default.airline_delay{delayyear=2008} stats: [numFiles=1,
 numRows=0, totalSize=689413344, rawDataSize=0]
OK
Time taken: 13.471 seconds
hive> LOAD DATA LOCAL INPATH '/root/airline/2007.csv'
    > OVERWRITE INTO TABLE airline_delay
    > PARTITION (delayYear='2007');
Loading data to table default.airline_delay partition (delayyear=20
07)
Partition default.airline_delay{delayyear=2007} stats: [numFiles=1,
 numRows=0, totalSize=702878193, rawDataSize=0]
OK
Time taken: 14.966 seconds
hive>
```

Hive --service hiveserver2 를 실행시켜 연동을 시킨다

```
e CPU 13.04 sec
2018-03-22 16:44:34,675 Stage-1 map = 100%  reduce = 67% Cumulati
ve CPU 16.07 sec
2018-03-22 16:44:42,753 Stage-1 map = 100%  reduce = 78% Cumulati
ve CPU 16.07 sec
2018-03-22 16:44:43,765 Stage-1 map = 100%  reduce = 100% Cumulat
ive CPU 17.51 sec
MapReduce Total cumulative CPU time: 17 seconds 510 msec
Ended Job = job_201803221529_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 3  Reduce: 3   Cumulative CPU: 17.51 sec   HDFS
 Read: 672110523 HDFS Write: 56 SUCCESS
Total MapReduce CPU Time Spent: 17 seconds 510 msec
OK
```

2. R을 통해 데이터를 탐색 및 분석 한다

```
library(rJava)
library(RJDBC)
library(DBI)
library(ggplot2)
library(dplyr)

drvName <- 'org.apache.hive.jdbc.HiveDriver';
id <- 'root';
pwd <- '111111';
url <- 'jdbc:hive2://192.168.111.100:10000';
```

→library를 임포트해준다, drvName,id,pwd,url을 입력하여준다

```
#라이브러리 디렉토리를 클래스 path로 지정합니다.
hive_lib <- 'C:\\java_hive_lib';
.jinit();
#hive lib에 있는 파일을 클래스파일로 하겠다.
.jaddClassPath(dir(hive_lib, full.names = T));
.jclassPath();
```

```
# DB 접근 과정(Java도 똑같음)
# 1. Driver Loading
drv <- JDBC(driverClass = drvName,
            classPath = 'hive-jdbc-1.0.1.jar')
# 2. Connection
conn <- dbConnect(drv, url, id, pwd)
```

→주석참조

```
# 3. Statement                    # sql문 작성 전송
sqlstr <- 'SELECT Year,Month,COUNT(*)
FROM airline_delay
where Month IN (1,2,3,4,6,7,8,9,10,11,12)
AND ArrDelay >0
GROUP BY Year,Month';
# 4. ResultSet                    # 값 받기  이 때, data.frame으로 받음
air <- dbGetQuery(conn, sqlstr);
# 5. Close
dbDisconnect(conn)
```

→년도별,월별,월별 딜레이 카운터를 출력하여 air객체에 담는다.

```
air_06 <- air[air$year == '2006',]
air_07 <- air[air$year == '2007',]
air_08 <- air[air$year == '2008',]
```

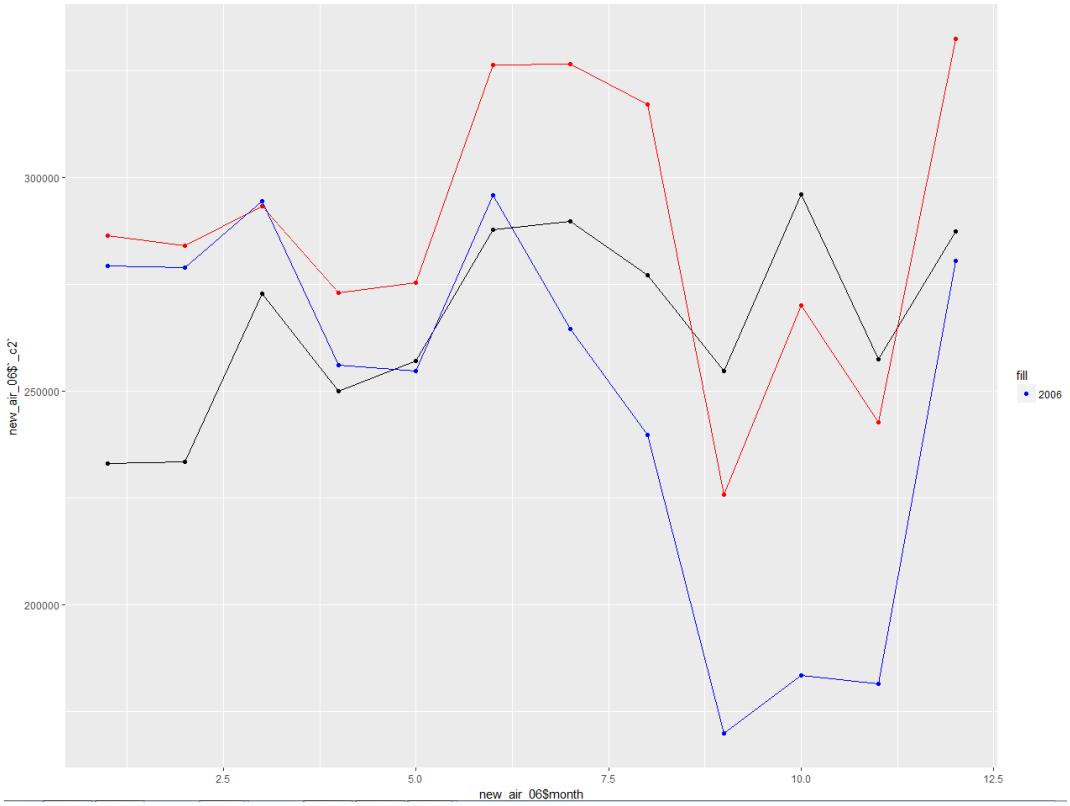
```
new_air_06 <- air_06[,c(-1)]
new_air_07 <- air_07[,c(-1)]
new_air_08 <- air_08[,c(-1)]
```

→연도,월,딜레이 count 칼럼 dataFrame을 만든 후, 각 연도에 따른 월, 딜레이 count칼럼을 dataFranme을 추출

4. 데이터 분석 화면을 구현 한다.

```
ggplot(data=new_air_06, aes(x=new_air_06$month, y=new_air_06$`_c2`,fill="2006"))+
  geom_line()+geom_point()+
  geom_line(data=new_air_07, aes(x=new_air_07$month, y=new_air_07$`_c2`), colour="red")+
  geom_point(data=new_air_07, aes(x=new_air_07$month, y=new_air_07$`_c2`), colour="red")+
  geom_line(data=new_air_08, aes(x=new_air_08$month, y=new_air_08$`_c2`), colour="blue")+
  geom_point(data=new_air_08, aes(x=new_air_08$month, y=new_air_08$`_c2`), colour="blue")
}
```

→그래프를 그려준다 geom_line과 point를 사용하여 한층 보기쉽게 작성



→해석: 8월~9월달은 대체로 delay비율이 작고 1월~7월 까지 delay가 잦다. 이 그래프를 통해 알수 있는점은 겨울에 폭설로 인해 딜레이가 잦은것으로 보인다. 그에 반해 여름의 delay 비율비 적은 것으로 추측된다.