기상청

[R을 활용한 분석교육실습]

AWS(방재기상관측) 자료를 활용한 고속도로 사고위험 분석사례

기상기후
빅데이터 분석 플랫폼

분석 개요

분석 교육 실습 주제인 AWS(방재기상관측)를 사용한 날씨 정보를 이용하여 날씨가 고속도로 교통사고에 미치는 영향에 대해 알아봅니다.

● 방재기상관측(AWS)[1)]

- 기상청은 서울기상관측소를 비롯하여 전국 95개소의 종관기상관측장비(ASOS)와 무인으로 운영되는 493개소의 자동기상관측장비(AWS)를 이용하여 지상기상관측업무를 수행하고 있습니다.
- 방재기상관측이란 지진 · 태풍 · 홍수 · 가뭄 등 기상현상에 따른 자연재해를 막기 위해 실시하는 지상관측을 말합니다.
- 관측 공백 해소 및 국지적인 기상 현상을 파악하기 위하여 전국 약 510여 지점에 자동기상관측장비(AWS)를 설치하여 자동으로 관측합니다.

● 교통사고[2)]

- 도로교통법 제2조의 규정에 의한 도로(도로법에 의한 도로, 유료도로법에 의한 유료도로, 그 밖의 불특정 다수의 통행을 위하여 공개된 장소)에서 차량의 운행중 인적인 피해가 발생한 사고를 말한다.
- 고속도로 교통사고 위험도 산출을 위해 아래와 같은 비기상 데이터를 수집합니다.
    - ① 교통류 데이터(시간단위) : 한국도로공사의 교통류 수집기 VDS(Vehicle Detection System)으로부터 수집된 시간단위 데이터
    - ② 도로기하구조 데이터 : 한국도로공사 직접 관리하는 25개 노선에 대한 도로의 평면선형, 종단경사 정보
    - ③ 교통류 데이터(분단위) : VDS의 분단위 데이터는 속도분산 임계치를 구하기 위해 사용되었으며, 사고다발구간 약 250일의 분단위 데이터

1) 출처:기상청 홈페이지

2) 출처:TAAS 교통사고분석시스템

분석 시나리오

실습할 예제는 AWS(방재기상관측) 기상 데이터와 비기상 데이터를 활용하여 날씨가 고속도로에 미치는 영향에 대한 모형을 구축해보는 시간을 가져 봅니다.

● 고속도로 사고 위험도 분석 모형 구축시나리오

[분석 활용 데이터]

기상 데이터

- 기온, 풍속 강수량

+

비기상 데이터

- 교통류 / 속도
- 도로기하구조

[Analytics]

교통사고 영향 변수 선택	
정의	고속도로의 각 1KM구간번호 및 콘존에 대한 기상(기온, 풍속, 강수의 평균, 최대, 최소, 편차)과 기하구조(종단, 평면, 경사, 방영) 그리고 각 자동차의 교통량 및 평균 속도
분석방법	Gradient Boosting Model(GBM) & 상관분석
분석결과	교통사고 발생

↓

교통사고 예측 모형 구축	
정의	머신러닝 기법(GBM)을 활용하여 날씨에 따른 교통사고를 예측하는 모형 구축
분석방법	Gradient Boosting Model(GBM)
분석결과	날씨에 따른 교통사고 분석 모형

분석 절차

실습은 데이터 로딩, 데이터 탐색, 데이터 처리, 모형 구축, 모형 검증의 단계에 따라 진행됩니다.

● 사고위험 예측 모형 구축절차

	[실습 설명]	[실습 단계]
1 데이터 로딩	분석환경을 설정하고 분석에 필요한 기상 데이터 및비기상데이터를 로딩하여 분석에 필요한 데이터를 준비하는단계	1. 분석 환경 설정 및 패키지로딩 2. 데이터 불러오기 3. 데이터 결합하기
2 데이터 탐색	분석 데이터의 요약 통계를 확인하는 단계	1. 요약 통계 보기 2. 탐색적 데이터 분석
3 데이터 처리	이상치를 처리하고 최종 데이터셋을 구성하는 단계	1. 이상치 처리
4 모형 구축	분석할 데이터를 선정하고, 날씨에 따른 고속도로 사고 모형을 구축하는 단계	1. 분석 데이터 셋 2. 모형 구축
5 모형 검증	최종 구축한 산출 모형의성능을 검증하는 단계	1. 변수 중요도 파악 2. 최종 모형 선택 3. 모형 성능 및 예측력 파악

※ 본 실습 교재는 교육용으로 기개발된 사고위험 예측 알고리즘을 최대한 단순화한 모형입니다. 기존 개발된 모형의 데이터와 과정이 일부 달라, 개발된 모형의 결과와는 다를 수 있습니다.

분석 데이터

고속도로 교통사고 위험도 전망 모형 구축에 사용된 파일 및 변수 정보를 확인합니다.

● 고속도로 교통사고 위험도 전망 모형 구축에 사용된 파일 및 변수정보

파일명	파일설명	변수명	변수설명	형식	예제
WEATHER	방제기상 관측(AWS)	TA_MAX_2_STD	이전 2시간 동안 최대 기온의 편차	Num	0.075
		WS_MAX_6_STD	이전 6시간 동안 최대 풍속의 편차	Num	0.281
		WS_MAX_2_MIN	이전 2시간 동안 최대 풍속의 최소	Num	2.45
		TA_MIN_6_MAX	이전 6시간 동안 최소 기온의 최대	Num	4.9
		WS_AVG_2_STD	이전 2시간 동안 평균 풍속의 편차	Num	0.1
		TA_MIN_6_STD	이전 6시간 동안 최소 기온의 편차	Num	0.476
		RN_3_MAX	이전 3시간 동안 최대 강수량	Num	0.125
TRAF	고속도로 교통량	AVG_SPEED	1시간 평균 속도	Num	85
		VOLUME_ALL	1시간 총 교통량	Num	2243
ROAD	고속도로 기하구조	GISID	1KM 구간 번호	Chr	00100000000000597
		CONZONE	콘존ID	Chr	0010CZE010
		BUSlane	버스전용차로 유무	Num	0
		JD_3KM_STD	이전 3KM 구간 동안 종단 편차	Num	0.816
		PM_3KM_STD	이전 3KM 구간 동안 평면 편차	Num	306
		JD_1KM_AVG	이전 1KM 구간 동안 종단 평균	Num	-3.63
		KS_3KM_AVG	이전 3KM 구간 동안 경사 평균	Num	2.66
		BY_3KM_STD	이전 3KM 구간 동안 방영 편차	Num	0.91
		BY_3KM_AVG	이전 3KM 구간 동안 방영 평균	Num	2.66
		SPEEDLIMIT	제한속도	Num	100
		Month	사고 발생 월	Int	11
		hour	사고 발생 시간	Int	01

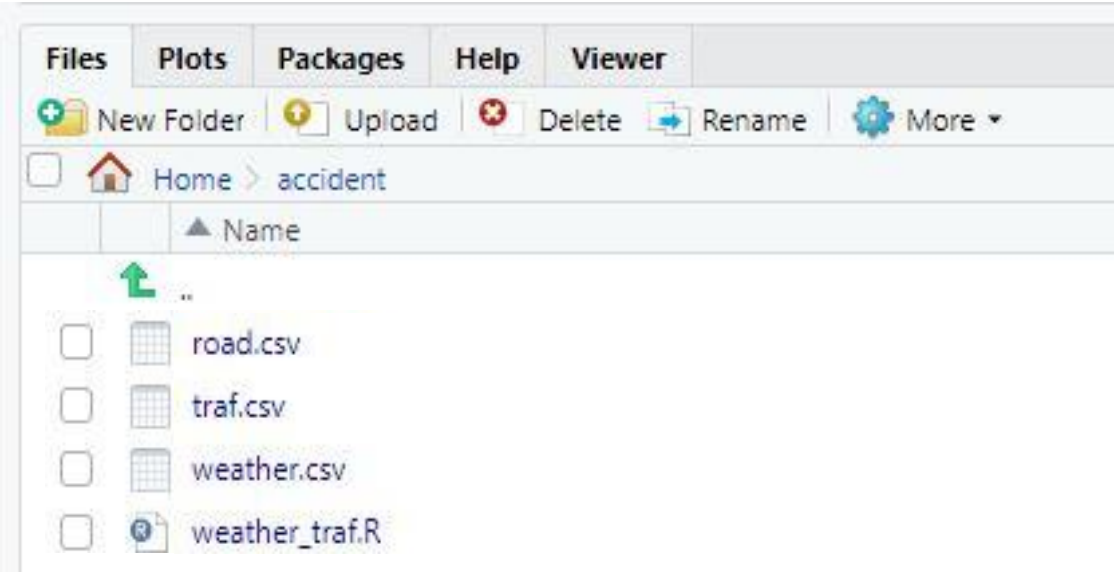
I. 데이터 로딩

1. 분석 환경 설정 및 패키지 로딩
2. 데이터 불러오기 :
    1. 기상데이터
    2. 도로기하구조 데이터
    3. 교통류 데이터
3. 데이터 결합하기

1. 분석 환경 설정 및 패키지 로딩 (1/2)

- **교육실습 R 소스**
  - 분석을 실행하기 위한 예제 소스 위치

  ./accident/weather_traf.R



- **패키지 설치 및 로딩**
  - 분석을 위해 사용할 패키지를 설치하고 R 메모리에 로딩

```
#=================================================================
# 패키지 로딩
#=================================================================
library(data.table)
library(h2o)
library(tidyverse)
library(gridExtra)
library(Metrics)
#=================================================================
```

  - **library()로 설치된 패키지를 R 메모리에 로딩**

1. 분석 환경 설정 및 패키지 로딩 (2/2)

- **분석 환경 설정**
  - **분석을 실행하기 전 메모리를 초기화하고 옵션들을 지정**

```
#============================================================
# 분석 환경 세팅
#============================================================
rm(list=ls()) #R메모리에 생성된 모든 객체 삭제(초기화)
cat("\014") # console clear
Sys.setenv(LANG="en_US.UTF-8") # 환경변수 설정
options(scipen=999)
options(digits=10)
#============================================================
# 작업 디렉터리 경로 설정
#============================================================
setwd("./accident")
getwd() #현재 위치한 디렉터리 경로 확인
```

- **rm(list=ls()) 로 R메모리 초기화**
- **cat("\014")으로 콘솔 창 지우기**
- **Sys.setenv()로 환경변수 설정**
- **options()로 분석 옵션을 세팅**
- **setwd()로 Default 디렉터리 위치를 확인**
- **getwd()로 현재 설정된 디렉터리 위치를 확인**

2. 데이터 불러오기 : 기상데이터

- **데이터를 불러온 뒤, 구조 확인하기**

```
#============================================================
# 기상데이터
weather <- read_csv("./weather.csv")
#============================================================
# 불러온 데이터 구조 확인하기
str(weather)
```

  - **read_csv()로 기상 데이터 불러오기**
  - **str()로 읽어 온 데이터 구조 확인**

> 실행 결과

```
> #============================================================
> # 기상데이터
> weather <- read_csv("./weather.csv")
Parsed with column specification:
cols(
  CONZONE = col_character(),
  GISID = col_character(),
  month = col_character(),
  hour = col_character(),
  TA_MAX_2_STD = col_double(),
  WS_MAX_6_STD = col_double(),
  WS_MAX_2_MIN = col_double(),
  TA_MIN_6_MAX = col_double(),
  WS_AVG_2_STD = col_double(),
  TA_MIN_6_STD = col_double(),
  RN_3_MAX = col_double()
)
>
> #============================================================
> # 불러온 데이터 구조 확인하기
> #============================================================
> str(weather)
tibble [220,064 x 11] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ CONZONE     : chr [1:220064] "0010CZE010" "0010CZE010" "0010CZE010" "0010CZE011" ...
 $ GISID       : chr [1:220064] "00100000000000597" "00100000597001000" "00100001000002000" "00100002000003000"
 $ month       : chr [1:220064] "01" "01" "01" "01" ...
 $ hour        : chr [1:220064] "00" "00" "00" "00" ...
 $ TA_MAX_2_STD: num [1:220064] 0.075 0.288 0.3 0.217 0.213 ...
 $ WS_MAX_6_STD: num [1:220064] 0.281 0.288 0.628 0.614 0.395 ...
 $ WS_MAX_2_MIN: num [1:220064] 2.45 1.98 3.53 2.67 2.27 ...
 $ TA_MIN_6_MAX: num [1:220064] 4.9 4.22 2.7 6.1 3.9 ...
 $ WS_AVG_2_STD: num [1:220064] 0.1 0.263 0.133 0.483 0.225 ...
 $ TA_MIN_6_STD: num [1:220064] 0.476 1.083 0.574 0.751 0.92 ...
 $ RN_3_MAX    : num [1:220064] 0 0.125 0 0 0 ...
 - attr(*, "spec")=
  .. cols(
  ..   CONZONE = col_character(),
  ..   GISID = col_character(),
  ..   month = col_character(),
  ..   hour = col_character(),
  ..   TA_MAX_2_STD = col_double(),
  ..   WS_MAX_6_STD = col_double(),
  ..   WS_MAX_2_MIN = col_double(),
  ..   TA_MIN_6_MAX = col_double(),
  ..   WS_AVG_2_STD = col_double(),
  ..   TA_MIN_6_STD = col_double(),
  ..   RN_3_MAX = col_double()
  .. )
```

2. 데이터 불러오기 : 도로기하 데이터

- **데이터를 불러온 뒤, 구조 확인하기**

```
#=============================================================
# 도로 기하데이터
traf <- read_csv("./input/traf.csv")

#=============================================================
# 불러온 데이터 구조 확인하기
str(traf)
```

- **read_csv()로 기상 데이터 불러오기**
- **str()로 읽어 온 데이터 구조 확인**

> 실행 결과

```
> #=============================================================
> # 도로 기하데이터
> traf <- read_csv("./traf.csv")
Parsed with column specification:
cols(
  CONZONE = col_character(),
  GISID = col_character(),
  month = col_character(),
  hour = col_character(),
  AVG_SPEED = col_double(),
  VOLUME_ALL = col_double()
)
>
> #=============================================================
> # 불러온 데이터 구조 확인하기
> #=============================================================
> str(traf)
tibble [220,064 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ CONZONE   : chr [1:220064] "0010CZE010" "0010CZE010" "0010CZE010" "0010CZE011" ...
 $ GISID     : chr [1:220064] "00100000000000597" "00100000597001000" "00100001000002000" "00100002000003000"
 $ month     : chr [1:220064] "01" "01" "01" "01" ...
 $ hour      : chr [1:220064] "00" "00" "00" "00" ...
 $ AVG_SPEED : num [1:220064] 85.7 82.8 82.3 96.9 90.6 ...
 $ VOLUME_ALL: num [1:220064] 2243 2190 1953 1993 1782 ...
 - attr(*, "spec")=
  .. cols(
  ..   CONZONE = col_character(),
  ..   GISID = col_character(),
  ..   month = col_character(),
  ..   hour = col_character(),
  ..   AVG_SPEED = col_double(),
  ..   VOLUME_ALL = col_double()
  .. )
```

2. 데이터 불러오기 : 교통류 데이터

- **데이터를 불러온 뒤, 구조 확인하기**

```
#============================================================
# 교통류 데이터
road <- read_csv( " ./input/road.csv")

#============================================================
# 불러온 데이터 구조 확인하기
str(road)
```

- **read_csv()로 기상 데이터 불러오기**
- **str()로 읽어 온 데이터 구조 확인**

> 실행 결과

```
> #============================================================
> # 교통류 데이터
> road <- read_csv("./road.csv")
Parsed with column specification:
cols(
  month = col_character(),
  hour = col_character(),
  Y = col_double(),
  GISID = col_character(),
  CONZONE = col_character(),
  BUSlane = col_double(),
  JD_3KM_STD = col_double(),
  PM_3KM_STD = col_double(),
  JD_1KM_AVG = col_double(),
  KS_3KM_AVG = col_double(),
  BY_3KM_STD = col_double(),
  BY_3KM_AVG = col_double(),
  SPEEDLIMIT = col_double()
)
>
> #============================================================
> # 불러온 데이터 구조 확인하기
> #============================================================
> str(road)
tibble [220,064 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ month     : chr [1:220064] "01" "01" "01" "01" ...
 $ hour      : chr [1:220064] "00" "00" "00" "00" ...
 $ Y         : num [1:220064] 2 4 3 3 4 7 4 3 12 5 ...
 $ GISID     : chr [1:220064] "00100000000000597" "00100000597001000" "00100001000002000" "00100002000003000"
 $ CONZONE   : chr [1:220064] "0010CZE010" "0010CZE010" "0010CZE010" "0010CZE011" ...
 $ BUSlane   : num [1:220064] 0 0 0 0 0 0 0 0 0 0 ...
 $ JD_3KM_STD: num [1:220064] 0 0.816 1.305 3.524 3.508 ...
 $ PM_3KM_STD: num [1:220064] 0 0 306 502 351 ...
 $ JD_1KM_AVG: num [1:220064] -2.444 -0.813 -4.01 4.533 -0.532 ...
 $ KS_3KM_AVG: num [1:220064] 2 1.5 2 2.33 2.33 ...
 $ BY_3KM_STD: num [1:220064] 0 0 1.414 1.7 0.816 ...
 $ BY_3KM_AVG: num [1:220064] 1 1 2 3.33 4 ...
 $ SPEEDLIMIT: num [1:220064] 100 100 100 100 100 100 100 100 100 100 ...
 - attr(*, "spec")=
  .. cols(
  ..   month = col_character(),
  ..   hour = col_character(),
  ..   Y = col_double(),
  ..   GISID = col_character(),
  ..   CONZONE = col_character(),
  ..   BUSlane = col_double(),
  ..   JD_3KM_STD = col_double(),
  ..   PM_3KM_STD = col_double(),
  ..   JD_1KM_AVG = col_double(),
  ..   KS_3KM_AVG = col_double(),
  ..   BY_3KM_STD = col_double(),
  ..   BY_3KM_AVG = col_double(),
  ..   SPEEDLIMIT = col_double()
  .. )
```

3. 데이터 결합하기

- **데이터 결합하기**
  - 날짜, 고속도로의 콘존ID와 1KM의 구간을 기준으로 대한 기상 교통류, 도로기하 데이터를 결합

```
#============================================================
# 테이블 결합 및 확인
#============================================================
weather_traf <- weather %>%
  left_join(road) %>%
  left_join(traf)
weather_traf
#============================================================
```

  - **left_join()로 기상 데이터 , 교통류, 도로기하 데이터를 결합**

> 실행 결과

```
> #============================================================
> # 테이블 결합 및 확인
> #============================================================
> weather_traf <- weather %>%
+   left_join(road) %>%
+   left_join(traf)
Joining, by = c("CONZONE", "GISID", "month", "hour")
Joining, by = c("CONZONE", "GISID", "month", "hour")
> weather_traf
# A tibble: 220,064 x 22
   CONZONE GISID month hour  TA_MAX_2_STD WS_MAX_6_STD WS_MAX_2_MIN TA_MIN_6_MAX WS_AVG_2_STD TA_MIN_6_STD RN_3_MAX
   <chr>   <chr> <chr> <chr>        <dbl>        <dbl>        <dbl>        <dbl>        <dbl>        <dbl>    <dbl>
 1 0010CZ~ 0010~ 01    00           0.075        0.281         2.45         4.9          0.1          0.476   0
 2 0010CZ~ 0010~ 01    00           0.288        0.288         1.98         4.22         0.263        1.08    0.125
 3 0010CZ~ 0010~ 01    00           0.3          0.628         3.53         2.7          0.133        0.574   0
 4 0010CZ~ 0010~ 01    00           0.217        0.614         2.67         6.1          0.483        0.751   0
 5 0010CZ~ 0010~ 01    00           0.213        0.395         2.28         3.9          0.225        0.920   0
 6 0010CZ~ 0010~ 01    00           0.193        0.422         2.26         2.7          0.271        0.890   0.0714
 7 0010CZ~ 0010~ 01    00           0.15         0.433         1.78         2            0.188        0.963   0
 8 0010CZ~ 0010~ 01    00           0.488        0.736         1.7          5            0.178        0.821   0
 9 0010CZ~ 0010~ 01    00           0.208        0.478         2.42         2.43         0.267        0.865   0
10 0010CZ~ 0010~ 01    00           0.310        0.493         1.9          0.460        0.200        1.01    0
# ... with 220,054 more rows, and 11 more variables: Y <dbl>, BUSlane <dbl>, JD_3KM_STD <dbl>, PM_3KM_STD <dbl>,
#   JD_1KM_AVG <dbl>, KS_3KM_AVG <dbl>, BY_3KM_STD <dbl>, BY_3KM_AVG <dbl>, SPEEDLIMIT <dbl>, AVG_SPEED <dbl>,
#   VOLUME_ALL <dbl>
```

기상청

II. 데이터 탐색

1. 타입변환
2. 탐색적 데이터 분석

1. 타입변환

```
#============================================================
# 타입 변환
#============================================================
weather_traf$Y <- as.numeric(weather_traf$Y)
weather_traf$AVG_SPEED <- as.numeric(weather_traf$AVG_SPEED)
weather_traf$VOLUME_ALL <- as.numeric(weather_traf$VOLUME_ALL)
weather_traf$TA_MAX_2_STD <- as.numeric(weather_traf$TA_MAX_2_STD)
weather_traf$WS_MAX_6_STD <- as.numeric(weather_traf$WS_MAX_6_STD)
weather_traf$WS_MAX_2_MIN <- as.numeric(weather_traf$WS_MAX_2_MIN)
weather_traf$TA_MIN_6_MAX <- as.numeric(weather_traf$TA_MIN_6_MAX)
weather_traf$WS_AVG_2_STD <- as.numeric(weather_traf$WS_AVG_2_STD)
weather_traf$TA_MIN_6_STD <- as.numeric(weather_traf$TA_MIN_6_STD)
weather_traf$RN_3_MAX <- as.numeric(weather_traf$RN_3_MAX)
weather_traf$month <- as.character(weather_traf$month)
weather_traf$hour <- as.character(weather_traf$hour)
weather_traf$GISID <- as.character(weather_traf$GISID)
weather_traf$CONZONE <- as.character(weather_traf$CONZONE)
weather_traf$BUSlane <- as.character(weather_traf$BUSlane)
weather_traf$JD_3KM_STD <- as.numeric(weather_traf$JD_3KM_STD)
weather_traf$PM_3KM_STD <- as.numeric(weather_traf$PM_3KM_STD)
weather_traf$JD_1KM_AVG <- as.numeric(weather_traf$JD_1KM_AVG)
weather_traf$KS_3KM_AVG <- as.numeric(weather_traf$KS_3KM_AVG)
weather_traf$BY_3KM_STD <- as.numeric(weather_traf$BY_3KM_STD)
weather_traf$BY_3KM_AVG <- as.numeric(weather_traf$BY_3KM_AVG)
weather_traf$SPEEDLIMIT <- as.numeric(weather_traf$SPEEDLIMIT)
```

- **as.numeric(), as.character() 함수를 사용해 각 데이터를 변환**

> 실행 결과

```
> #============================================================
> # 타입 변환
> #============================================================
> weather_traf$Y <- as.numeric(weather_traf$Y)
> weather_traf$AVG_SPEED <- as.numeric(weather_traf$AVG_SPEED)
> weather_traf$VOLUME_ALL <- as.numeric(weather_traf$VOLUME_ALL)
> weather_traf$TA_MAX_2_STD <- as.numeric(weather_traf$TA_MAX_2_STD)
> weather_traf$WS_MAX_6_STD <- as.numeric(weather_traf$WS_MAX_6_STD)
> weather_traf$WS_MAX_2_MIN <- as.numeric(weather_traf$WS_MAX_2_MIN)
> weather_traf$TA_MIN_6_MAX <- as.numeric(weather_traf$TA_MIN_6_MAX)
> weather_traf$WS_AVG_2_STD <- as.numeric(weather_traf$WS_AVG_2_STD)
> weather_traf$TA_MIN_6_STD <- as.numeric(weather_traf$TA_MIN_6_STD)
> weather_traf$RN_3_MAX <- as.numeric(weather_traf$RN_3_MAX)
> weather_traf$month <- as.character(weather_traf$month)
> weather_traf$hour <- as.character(weather_traf$hour)
> weather_traf$GISID <- as.character(weather_traf$GISID)
> weather_traf$CONZONE <- as.character(weather_traf$CONZONE)
> weather_traf$BUSlane <- as.character(weather_traf$BUSlane)
> weather_traf$JD_3KM_STD <- as.numeric(weather_traf$JD_3KM_STD)
> weather_traf$PM_3KM_STD <- as.numeric(weather_traf$PM_3KM_STD)
> weather_traf$JD_1KM_AVG <- as.numeric(weather_traf$JD_1KM_AVG)
> weather_traf$KS_3KM_AVG <- as.numeric(weather_traf$KS_3KM_AVG)
> weather_traf$BY_3KM_STD <- as.numeric(weather_traf$BY_3KM_STD)
> weather_traf$BY_3KM_AVG <- as.numeric(weather_traf$BY_3KM_AVG)
> weather_traf$SPEEDLIMIT <- as.numeric(weather_traf$SPEEDLIMIT)
```

2. 탐색적 데이터 분석 (1/15)

- **데이터 요약, 결측치 파악**

```
#============================================================
# 데이터 요약, 결측치 파악
#============================================================

# 데이터 요약
summary(weather_traf)

# 컬럼별 결측치
map_dfr(1:22, function(x) {
  data.frame(
    col = colnames(weather_traf[x]),
    count_na = sum(is.na(weather_traf[x]))
  )
})
```

- summary() 함수를 사용해 데이터의 요약통계 파악
  - Min: 최소값
  - 1st Qu.: 1분위수
  - Median: 중앙값
  - Mean: 평균
  - 3rd Qu: 3분위수
  - Max: 최대값
- Map_dfr()함수를 사용해 함수를 병렬처리 & data.frame 자동 row binding
  - 컬럼 col: colnames()함수를 사용해 해당 컬럼 지정
  - 컬럼 count_na: sum(), is.na()함수를 사용해 해당 컬럼의 결측값 개수 확인

2. 탐색적 데이터 분석 (2/15)

> 실행 결과

```
> # 데이터 요약
> summary(weather_traf)
   CONZONE             GISID              month               hour           TA_MAX_2_STD      WS_MAX_6_STD     WS_MAX_2_MIN      TA_MIN_6_MAX
 Length:220064      Length:220064      Length:220064      Length:220064      Min.   : 0.0000   Min.   :0.0000   Min.   : 0.000   Min.   :-18.60
 Class :character   Class :character   Class :character   Class :character   1st Qu.: 0.2000   1st Qu.:0.3898   1st Qu.: 1.214   1st Qu.:  5.55
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Median : 0.3333   Median :0.5258   Median : 1.850   Median : 14.93
                                                                            Mean   : 0.4325   Mean   :0.5600   Mean   : 2.014   Mean   : 14.26
                                                                            3rd Qu.: 0.5556   3rd Qu.:0.6924   3rd Qu.: 2.650   3rd Qu.: 23.26
                                                                            Max.   :12.7000   Max.   :4.5215   Max.   :11.000   Max.   : 37.00
  WS_AVG_2_STD      TA_MIN_6_STD        RN_3_MAX            Y             BUSlane           JD_3KM_STD       PM_3KM_STD        JD_1KM_AVG
 Min.   :0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 1.000   Length:220064      Min.   :0.0000   Min.   :    0.0   Min.   :-6.450
 1st Qu.:0.1143   1st Qu.: 0.7391   1st Qu.: 0.0000   1st Qu.: 3.000   Class :character   1st Qu.:0.6156   1st Qu.:  141.4   1st Qu.: 0.000
 Median :0.1786   Median : 1.1254   Median : 0.0000   Median : 4.000   Mode  :character   Median :2.1543   Median :  286.1   Median : 2.400
 Mean   :0.2022   Mean   : 1.3218   Mean   : 0.2315   Mean   : 4.578                      Mean   :2.0210   Mean   :  759.1   Mean   : 2.509
 3rd Qu.:0.2625   3rd Qu.: 1.7237   3rd Qu.: 0.1250   3rd Qu.: 6.000                      3rd Qu.:3.3041   3rd Qu.:  801.4   3rd Qu.: 6.500
 Max.   :1.9500   Max.   :13.9075   Max.   :38.7500   Max.   :19.000                      Max.   :5.4212   Max.   :11785.1   Max.   : 6.500
   KS_3KM_AVG      BY_3KM_STD       BY_3KM_AVG      SPEEDLIMIT      AVG_SPEED        VOLUME_ALL
 Min.   :1.000   Min.   :0.0000   Min.   :1.000   Min.   :100.0   Min.   :  3.54   Min.   :   1.0
 1st Qu.:2.000   1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:100.0   1st Qu.: 90.73   1st Qu.: 589.5
 Median :2.667   Median :0.9428   Median :2.000   Median :100.0   Median : 96.45   Median :1248.2
 Mean   :2.612   Mean   :0.9476   Mean   :1.994   Mean   :101.2   Mean   : 95.59   Mean   :1661.3
 3rd Qu.:3.333   3rd Qu.:1.8856   3rd Qu.:2.667   3rd Qu.:100.0   3rd Qu.:101.46   3rd Qu.:2411.0
 Max.   :4.000   Max.   :2.3570   Max.   :5.667   Max.   :110.0   Max.   :169.52   Max.   :8689.0
>
> # 컬럼별 결측치
> map_dfr(1:22, function(x) {
+   data.frame(
+     col = colnames(weather_traf[x]),
+     count_na = sum(is.na(weather_traf[x]))
+   )
+ })
           col count_na
1       CONZONE        0
2         GISID        0
3         month        0
4          hour        0
5  TA_MAX_2_STD        0
6  WS_MAX_6_STD        0
7  WS_MAX_2_MIN        0
8  TA_MIN_6_MAX        0
9  WS_AVG_2_STD        0
10 TA_MIN_6_STD        0
11     RN_3_MAX        0
12            Y        0
13      BUSlane        0
14   JD_3KM_STD        0
15   PM_3KM_STD        0
16   JD_1KM_AVG        0
17   KS_3KM_AVG        0
18   BY_3KM_STD        0
19   BY_3KM_AVG        0
20   SPEEDLIMIT        0
21    AVG_SPEED        0
22   VOLUME_ALL        0
```

2. 탐색적 데이터 분석 (3/15)

● 1KM 구간에 따른 사고 빈도 (기상)

```
#============================================================
# 1KM 구간에 따른 사고 빈도 (기상)
#============================================================
lst <- map(5:11, function(i) {
  mean_value <- colnames(weather_traf)[i]

  weather_traf %>%
    select(GISID, mean_value, Y) %>%
    rename(aa = mean_value) %>%
    group_by(GISID) %>%
    summarise(sum_y = sum(Y),
              mean_value = mean(aa)) %>%
    select(sum_y, mean_value) %>%
    rename(aa = mean_value) %>%
    ggplot(aes(x = aa,y = sum_y)) +
    geom_point(alpha=.2,color = "#208A2E") +
    labs(title = paste0(mean_value,"에 따른 사고 빈도"), x = mean_value, y = "사고 빈도") +
    theme_bw() +
    theme(legend.position = "bottom")
})

grid.arrange(grobs=lst, ncol=2)
```

'weather_traf ' 테이블에서 특정 변수만 추출 후, mean_value 변수명을 aa로 변경

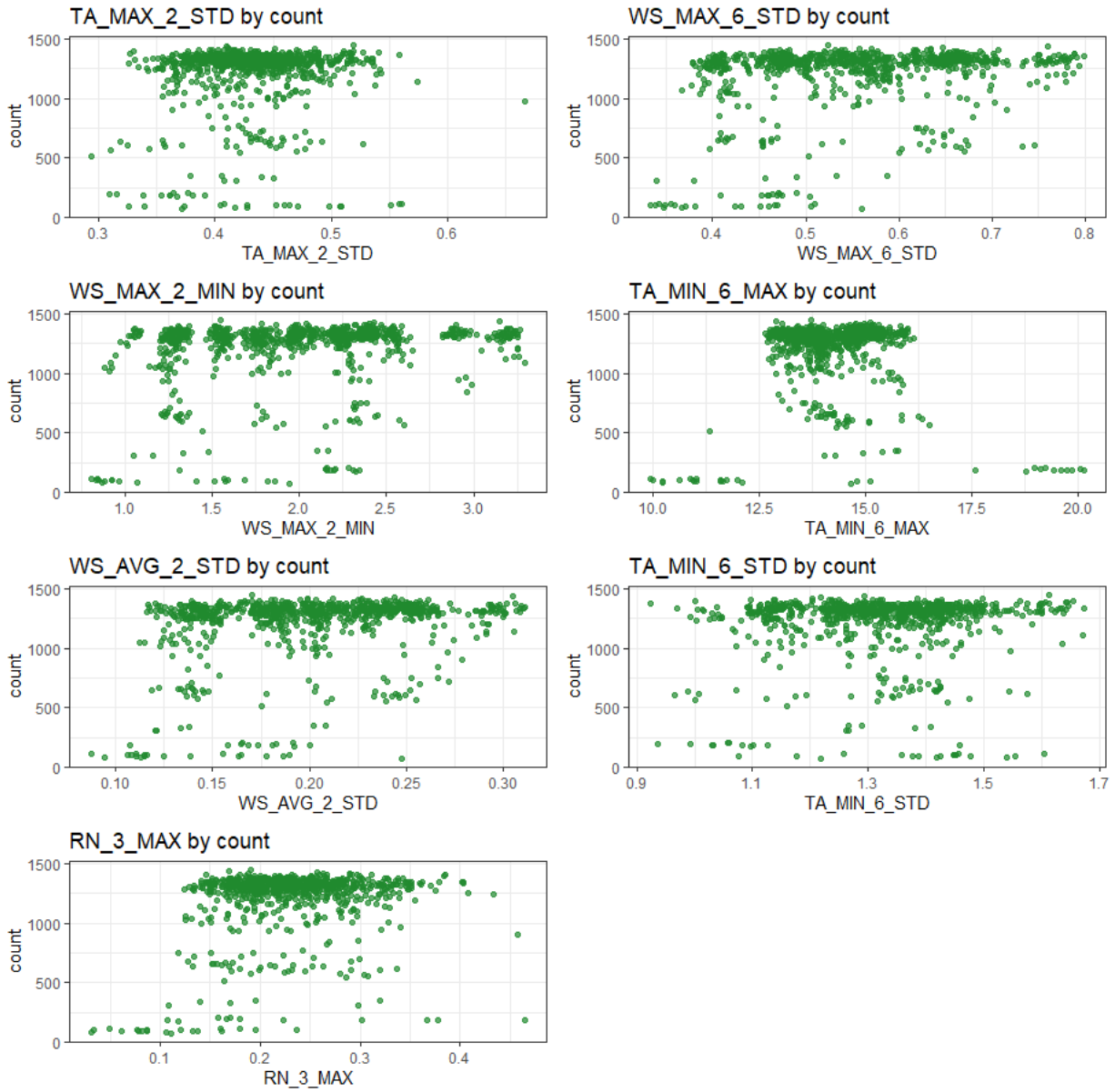
GISID로 group by한후 Y,바뀐 변수 aa에 대해 각각 sum, mean 값 계산

점색상 선택
그래프 제목 및 축 제목, 설정
범례위치 설정

- 1km 구간에 따른 사고 빈도 시각화 (기상)
- grid.arrange() 함수를 통해 기상에 대한 모든 시각화를 표현
- map() 함수를 사용하여 기상에 해당하는 시각화를 리스트로 생성
  - 기상에 대한 컬럼 5~11열에 대해 반복문 사용
- 각 반복문을 통해 변수명을 mean_value로 지정
- 'weather_traf'데이터에서 GISID, mean_value, Y 추출
- ggplot()에서 제목을 지정해주기 위해서 mean_value의 변수명을 aa로 재설정
- group_by진행
  - GISID
  - 각 기상에 대한 컬럼
  - Y
- ggplot(), geom_point()함수를 통해 점 그래프 생성

2. 탐색적 데이터 분석 (4/15)

> 실행 결과



2. 탐색적 데이터 분석 (5/15)

● 1KM 구간에 따른 사고 빈도 (기하구조)

```
#============================================================
# 1KM 구간에 따른 사고 빈도 (기하구조)
#============================================================
lst <- map(14:20, function(i) {
  mean_value <- colnames(weather_traf)[i]

  weather_traf %>%
    select(GISID, mean_value, Y) %>%
    rename(aa = mean_value) %>%
    group_by(GISID) %>%
    summarise(sum_y = sum(Y),
              mean_value = mean(aa)) %>%
    select(sum_y, mean_value) %>%
    rename(aa = mean_value) %>%
    ggplot(aes(x = aa,y = sum_y)) +
    geom_point(alpha=.2,color = "#A28C2A") +
    labs(title = paste0(mean_value,"에 따른 사고 빈도"), x = mean_value, y = "사고 빈도") +
    theme_bw() +
    theme(legend.position = "bottom")
})

grid.arrange(grobs=lst, ncol=2)
```

'weather_traf ' 테이블에서 특정 변수만 추출 후, mean_value 변수명을 aa로 변경

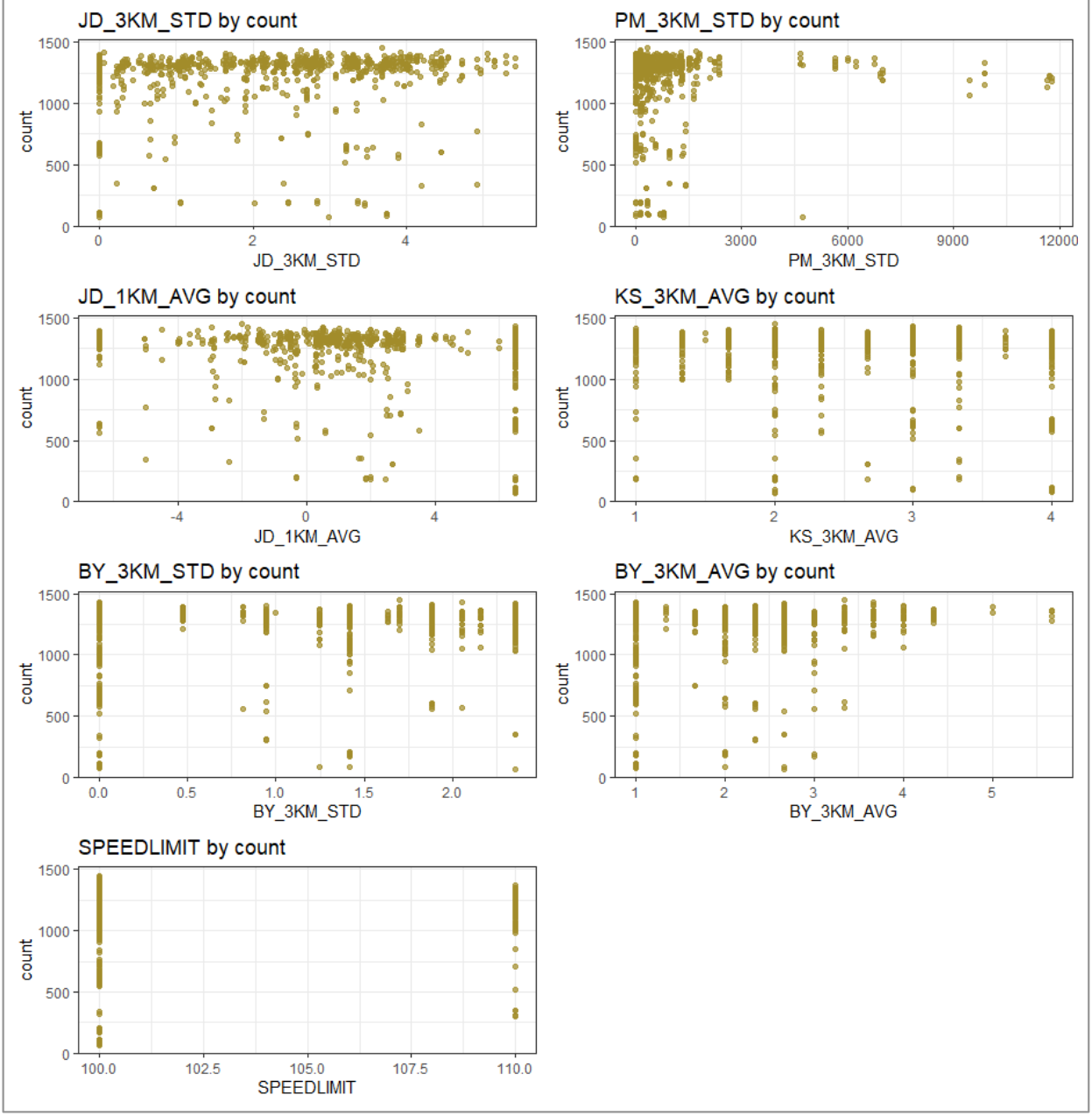
GISID로 group by한후 Y,바뀐 변수 aa에 대해 각각 sum, mean 값 계산

점색상 선택
그래프 제목 및 축 제목, 설정
범례위치 설정

- 1km 구간에 따른 사고 빈도 시각화 (기하구조)
- grid.arrange() 함수를 통해 기상에 대한 모든 시각화를 표현
- map() 함수를 사용하여 기상에 해당하는 시각화를 리스트로 생성
  - 기하구조에 대한 컬럼 14~20열에 대해 반복문 사용
- 각 반본문을 통해 변수명을 mean_value로 지정
- 'weather_traf'데이터에서 GISID, mean_value, Y 추출
- ggplot()에서 제목을 지정해주기 위해서 mean_value의 변수명을 aa로 재설정
- group_by진행
  - GISID
  - 각 기하구조에 대한 컬럼
  - Y
- ggplot(), geom_point()함수를 통해 점 그래프 생성

2. 탐색적 데이터 분석 (6/15)

> 실행 결과



2. 탐색적 데이터 분석 (7/15)

● 1KM 구간에 따른 사고 빈도 (교통류)

```
#============================================================
# 1KM 구간에 따른 사고 빈도 (교통류)
#============================================================
lst <- map(21:22, function(i) {
  mean_value <- colnames(weather_traf)[i]

  weather_traf %>%
    select(GISID, mean_value, Y) %>%
    rename(aa = mean_value) %>%
    group_by(GISID) %>%
    summarise(sum_y = sum(Y),
              mean_value = mean(aa)) %>%
    select(sum_y, mean_value) %>%
    rename(aa = mean_value) %>%
    ggplot(aes(x = aa,y = sum_y)) +
    geom_point(alpha=.2,color = "#2125A1") +
    labs(title = paste0(mean_value,"에 따른 사고 빈도"), x = mean_value, y = "사고 빈도") +
    theme_bw() +
    theme(legend.position = "bottom")
})

grid.arrange(grobs=lst, ncol=2)
```

'weather_traf ' 테이블에서 특정 변수만 추출 후, mean_value 변수명을 aa로 변경

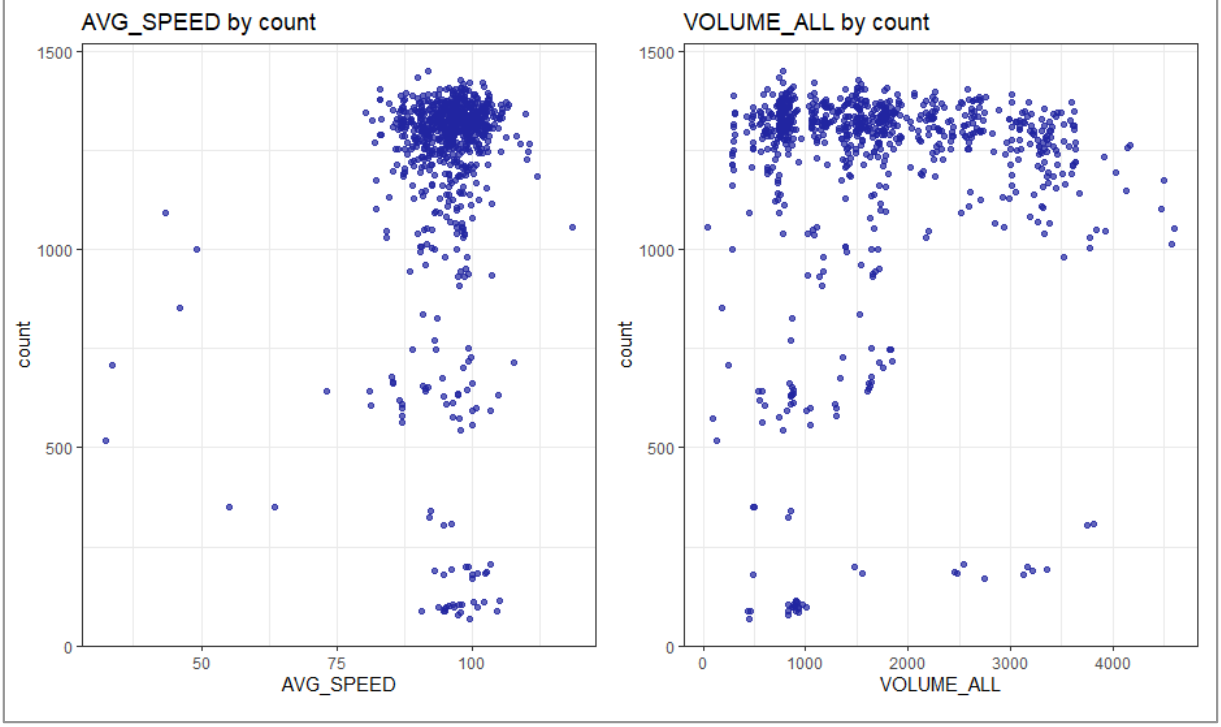
GISID로 group by한후 Y,바뀐 변수 aa에 대해 각각 sum, mean 값 계산

점색상 선택
그래프 제목 및 축 제목, 설정
범례위치 설정

- 1km 구간에 따른 사고 빈도 시각화 (교통류)
- grid.arrange() 함수를 통해 기상에 대한 모든 시각화를 표현
- map() 함수를 사용하여 기상에 해당하는 시각화를 리스트로 생성
  - 교통에 대한 컬럼 21~22열에 대해 반복문 사용
- 각 반본문을 통해 변수명을 mean_value로 지정
- 'weather_traf'데이터에서 GISID, mean_value, Y 추출
- ggplot()에서 제목을 지정해주기 위해서 mean_value의 변수명을 aa로 재설정
- group_by진행
  - GISID
  - 각 교통류에 대한 컬럼
  - Y
- ggplot(), geom_point()함수를 통해 점 그래프 생성

2. 탐색적 데이터 분석 (8/15)

> 실행 결과



2. 탐색적 데이터 분석 (9/15)

- **콘존에 따른 사고 빈도 (기상)**

```
#============================================================
# 콘존 따른 사고 빈도 (기상)
#============================================================
lst <- map(5:11, function(i) {
  mean_value <- colnames(weather_traf)[i]

  weather_traf %>%
    select(CONZONE, mean_value, Y) %>%
    rename(aa = mean_value) %>%
    group_by(CONZONE) %>%
    summarise(sum_y = sum(Y),
              mean_value = mean(aa)) %>%
    select(sum_y, mean_value) %>%
    rename(aa = mean_value) %>%
    ggplot(aes(x = aa,y = sum_y)) +
    geom_point(alpha=.3,color = "#208A2E") +
    labs(title = paste0(mean_value,"에 따른 사고 빈도"), x = mean_value, y = "사고 빈도") +
    theme_bw() +
    theme(legend.position = "bottom")
})

grid.arrange(grobs=lst, ncol=2)
```

'weather_traf ' 테이블에서 특정 변수만 추출 후, mean_value 변수명을 aa로 변경

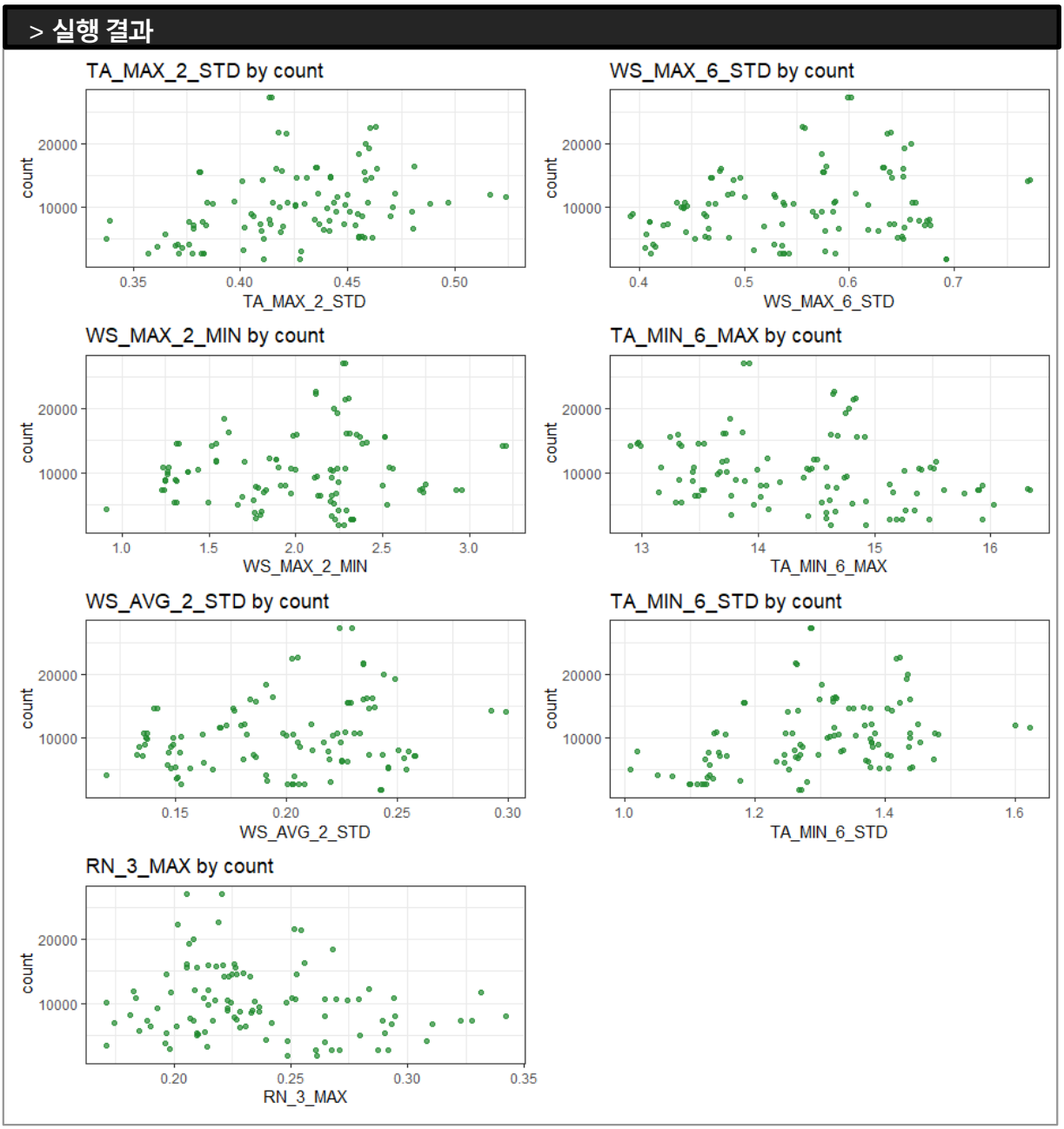
CONZONE으로 group by한후 Y,바뀐 변수 aa에 대해 각각 sum, mean 값 계산

점색상 선택
그래프 제목 및 축 제목, 설정
범례위치 설정

- 콘존에 따른 사고 빈도 시각화 (기상)
- grid.arrange() 함수를 통해 기상에 대한 모든 시각화를 표현
- map() 함수를 사용하여 기상에 해당하는 시각화를 리스트로 생성
  - 기상에 대한 컬럼 5~11열에 대해 반복문 사용
- 각 반본문을 통해 변수명을 mean_value로 지정
- 'weather_traf'데이터에서 GISID, mean_value, Y 추출
- ggplot()에서 제목을 지정해주기 위해서 mean_value의 변수명을 aa로 재설정
- group_by진행
  - CONZONE
  - 각 기상에 대한 컬럼
  - Y
- ggplot(), geom_point()함수를 통해 점 그래프 생성

2. 탐색적 데이터 분석 (10/15)

> 실행 결과



2. 탐색적 데이터 분석 (11/15)

● 콘존에 따른 사고 빈도 (기하구조)

```
#============================================================
# 콘존 따른 사고 빈도 (기하구조)
#============================================================
lst <- map(14:20, function(i) {
  mean_value <- colnames(weather_traf)[i]

  weather_traf %>%
    select(CONZONE, mean_value, Y) %>%
    rename(aa = mean_value) %>%
    group_by(CONZONE) %>%
    summarise(sum_y = sum(Y),
              mean_value = mean(aa)) %>%
    select(sum_y, mean_value) %>%
    rename(aa = mean_value) %>%
    ggplot(aes(x = aa,y = sum_y)) +
    geom_point(alpha=.3,color = "#A28C2A") +
    labs(title = paste0(mean_value,"에 따른 사고 빈도"), x = mean_value, y = "사고 빈도") +
    theme_bw() +
    theme(legend.position = "bottom")
})

grid.arrange(grobs=lst, ncol=2)
```

'weather_traf ' 테이블에서 특정 변수만 추출 후, mean_value 변수명을 aa로 변경

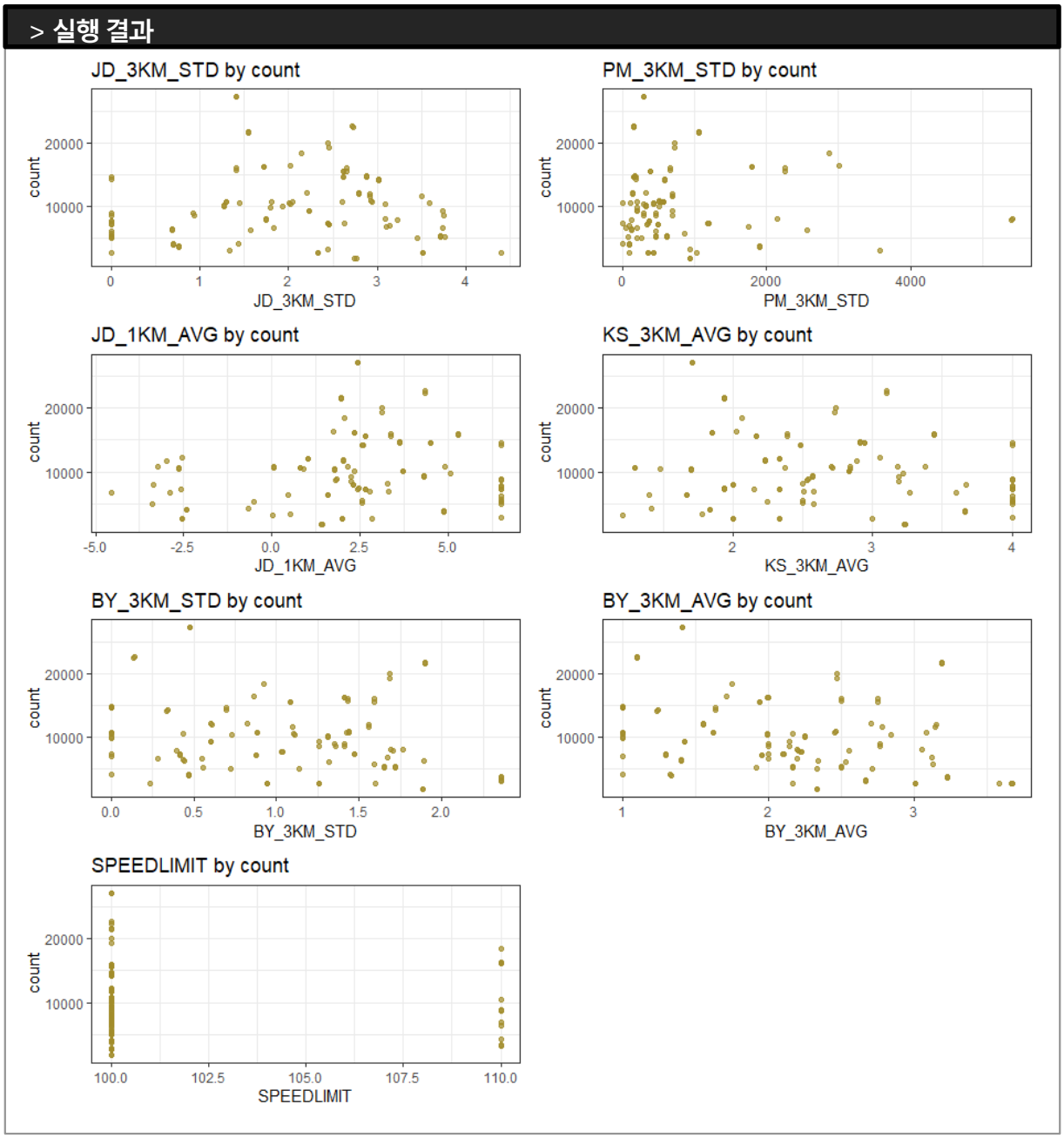
CONZONE으로 group by한후 Y,바뀐 변수 aa에 대해 각각 sum, mean 값 계산

점색상 선택
그래프 제목 및 축 제목, 설정
범례위치 설정

- 콘존에 따른 사고 빈도 시각화 (기하구조)
- grid.arrange() 함수를 통해 기상에 대한 모든 시각화를 표현
- map() 함수를 사용하여 기상에 해당하는 시각화를 리스트로 생성
  - 기하구조에 대한 컬럼 14~20열에 대해 반복문 사용
- 각 반본문을 통해 변수명을 mean_value로 지정
- 'weather_traf'데이터에서 GISID, mean_value, Y 추출
- ggplot()에서 제목을 지정해주기 위해서 mean_value의 변수명을 aa로 재설정
- Group_by진행
  - CONZONE
  - 각 기하구조에 대한 컬럼
  - Y
- ggplot(), geom_point()함수를 통해 점 그래프 생성

2. 탐색적 데이터 분석 (12/15)

> 실행 결과



2. 탐색적 데이터 분석 (13/15)

- **콘존에 따른 사고 빈도 (교통류)**

```
#============================================================
# 콘존 따른 사고 빈도 (교통류)
#============================================================
lst <- map(21:22, function(i) {
  mean_value <- colnames(weather_traf)[i]

  weather_traf %>%
    select(CONZONE, mean_value, Y) %>%
    rename(aa = mean_value) %>%
    group_by(CONZONE) %>%
    summarise(sum_y = sum(Y),
              mean_value = mean(aa)) %>%
    select(sum_y, mean_value) %>%
    rename(aa = mean_value) %>%
    ggplot(aes(x = aa,y = sum_y)) +
    geom_point(alpha=.3,color = "#2125A1") +
    labs(title = paste0(mean_value,"에 따른 사고 빈도"), x = mean_value, y = "사고 빈도") +
    theme_bw() +
    theme(legend.position = "bottom")
})

grid.arrange(grobs=lst, ncol=2)
```

'weather_traf ' 테이블에서 특정 변수만 추출 후, mean_value 변수명을 aa로 변경

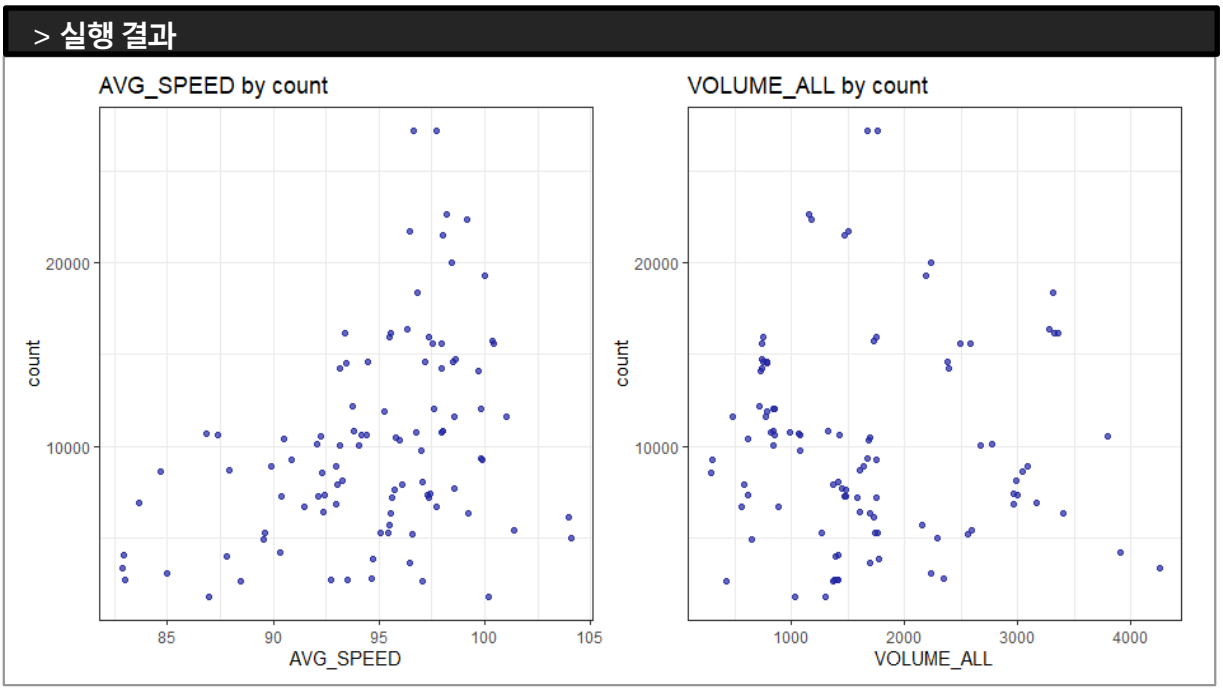
CONZONE으로 group by한후 Y,바뀐 변수 aa에 대해 각각 sum, mean 값 계산

점색상 선택
그래프 제목 및 축 제목, 설정
범례위치 설정

- 콘존에 따른 사고 빈도 시각화 (교통류)
- grid.arrange() 함수를 통해 기상에 대한 모든 시각화를 표현
- map() 함수를 사용하여 기상에 해당하는 시각화를 리스트로 생성
  - 교통류에 대한 컬럼 21~22열에 대해 반복문 사용
- 각 반본문을 통해 변수명을 mean_value로 지정
- 'weather_traf'데이터에서 GISID, mean_value, Y 추출
- ggplot()에서 제목을 지정해주기 위해서 mean_value의 변수명을 aa로 재설정
- Group_by진행
  - CONZONE
  - 각 교통류에 대한 컬럼
  - Y
- ggplot(), geom_point()함수를 통해 점 그래프 생성

2. 탐색적 데이터 분석 (14/15)

> 실행 결과



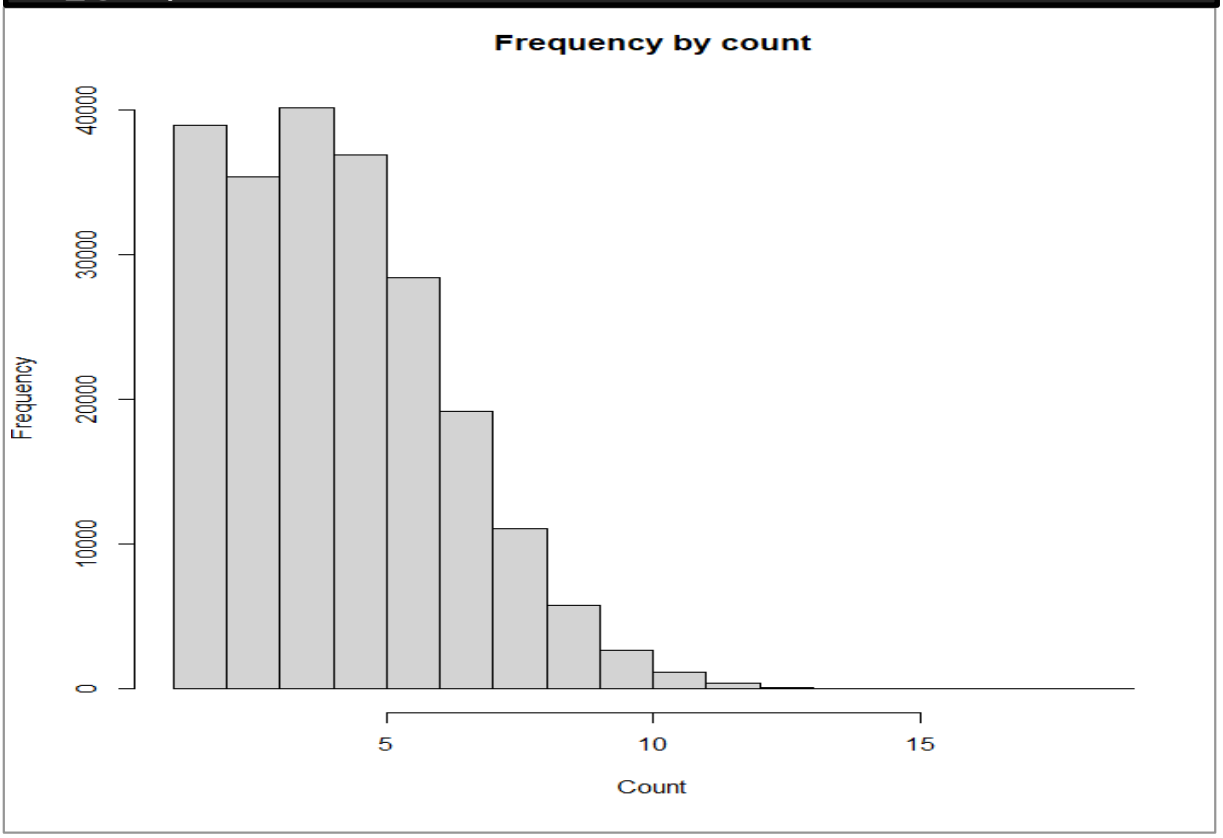
2. 탐색적 데이터 분석 (15/15)

- **히스토그램 (발생건수, 빈도)**

```
#============================================================
# 히스토그램 (발생건수, 빈도)
#============================================================
hist(weather_traf$Y,
     xlab = "Count",
     ylab = "Frequency",
     main = "Frequency by count")
```

- **hist() 함수를 사용해 히스토그램 생성**
  - **월, 일, 1km 구간, 콘존ID에 각각 의 발생 건수에 대한 빈도 시각화**

> 실행 결과



기상청

III. 데이터 처리

1. 이상치 제거

1. 이상치 제거 (1/3)

- **이상치 제거**

- 이상치 제거: summary()함수를 통해 이상치로 판단 되는 데이터는 제거

```
#============================================================
# 이상치 제거 전
#============================================================
summary(weather_traf)
```

> 실행 결과

```
> #============================================================
> # 이상치 제거 전
> #============================================================
> summary(weather_traf)
   CONZONE             GISID              month              hour          
 Length:220064      Length:220064      Length:220064      Length:220064     
 Class :character   Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Mode  :character   Mode  :character  


  TA_MAX_2_STD       WS_MAX_6_STD      WS_MAX_2_MIN      TA_MIN_6_MAX      WS_AVG_2_STD    
 Min.   : 0.0000   Min.   :0.0000   Min.   : 0.000   Min.   :-18.60   Min.   :0.0000  
 1st Qu.: 0.2000   1st Qu.:0.3898   1st Qu.: 1.214   1st Qu.:  5.55   1st Qu.:0.1143  
 Median : 0.3333   Median :0.5258   Median : 1.850   Median : 14.93   Median :0.1786  
 Mean   : 0.4325   Mean   :0.5600   Mean   : 2.014   Mean   : 14.26   Mean   :0.2022  
 3rd Qu.: 0.5556   3rd Qu.:0.6924   3rd Qu.: 2.650   3rd Qu.: 23.26   3rd Qu.:0.2625  
 Max.   :12.7000   Max.   :4.5215   Max.   :11.000   Max.   : 37.00   Max.   :1.9500  
  TA_MIN_6_STD       RN_3_MAX            Y             BUSlane        JD_3KM_STD     
 Min.   : 0.0000   Min.   : 0.0000   Min.   : 1.000   Min.   :0.000   Min.   :0.0000  
 1st Qu.: 0.7391   1st Qu.: 0.0000   1st Qu.: 3.000   1st Qu.:0.000   1st Qu.:0.6156  
 Median : 1.1254   Median : 0.0000   Median : 4.000   Median :0.000   Median :2.1543  
 Mean   : 1.3218   Mean   : 0.2315   Mean   : 4.578   Mean   :0.252   Mean   :2.0210  
 3rd Qu.: 1.7237   3rd Qu.: 0.1250   3rd Qu.: 6.000   3rd Qu.:1.000   3rd Qu.:3.3041  
 Max.   :13.9075   Max.   :38.7500   Max.   :19.000   Max.   :1.000   Max.   :5.4212  
   PM_3KM_STD        JD_1KM_AVG        KS_3KM_AVG      BY_3KM_STD       BY_3KM_AVG    
 Min.   :    0.0   Min.   :-6.450   Min.   :1.000   Min.   :0.0000   Min.   :1.000  
 1st Qu.:  141.4   1st Qu.: 0.000   1st Qu.:2.000   1st Qu.:0.0000   1st Qu.:1.000  
 Median :  286.1   Median : 2.400   Median :2.667   Median :0.9428   Median :2.000  
 Mean   :  759.1   Mean   : 2.509   Mean   :2.612   Mean   :0.9476   Mean   :1.994  
 3rd Qu.:  801.4   3rd Qu.: 6.500   3rd Qu.:3.333   3rd Qu.:1.8856   3rd Qu.:2.667  
 Max.   :11785.1   Max.   : 6.500   Max.   :4.000   Max.   :2.3570   Max.   :5.667  
   SPEEDLIMIT       AVG_SPEED        VOLUME_ALL     
 Min.   :100.0   Min.   :  3.54   Min.   :   1.0  
 1st Qu.:100.0   1st Qu.: 90.73   1st Qu.: 589.5  
 Median :100.0   Median : 96.45   Median :1248.2  
 Mean   :101.2   Mean   : 95.59   Mean   :1661.3  
 3rd Qu.:100.0   3rd Qu.:101.46   3rd Qu.:2411.0  
 Max.   :110.0   Max.   :169.52   Max.   :8689.0  
```

1. 이상치 제거 (2/3)

```
#============================================================
# 이상치 제거
#============================================================
weather_traf <- weather_traf %>%
  filter(TA_MAX_2_STD < 7) %>%
  filter(WS_MAX_2_MIN < 7) %>%
  filter(TA_MIN_6_STD < 10) %>%
  filter(RN_3_MAX < 20) %>%
  filter(PM_3KM_STD < 10000)
```

- **이상치 제거: summary()함수를 통해 이상치로 판단 되는 데이터는 제거**
- **TA_MAX_2_STD < 7**
  - **이전 2시간 동안 최대 기온의 편차: 7보다 큰값 제거**
- **WS_MAX_2_MIN < 7**
  - **이전 2시간 동안 최대 풍속의 최소: 7보다 큰값 제거**
- **TA_MIN_6_STD < 10**
  - **이전 6시간 동안 최소 기온의 편차: 10보다 큰값 제거**
- **RN_3_MAX < 20**
  - **이전 3시간 동안 강수의 최대: 20보다 큰값 제거**
- **PM_3KM_STD < 10000**
  - **이전 3KM 구간 동안 평면 편차: 10000보다 큰값 제거**

> 실행 결과

```
> #============================================================
> # 이상치 제거
> #============================================================
> weather_traf <- weather_traf %>%
+   filter(TA_MAX_2_STD < 7) %>% # 이전 2시간 동안 최대 기온의 편차
+   filter(WS_MAX_2_MIN < 7) %>% # 이전 2시간 동안 최대 풍속의 최소
+   filter(TA_MIN_6_STD < 10) %>% # 이전 6시간 동안 최소 기온의 편차
+   filter(RN_3_MAX < 20) %>% # 이전 3시간 동안 강수의 최대
+   filter(PM_3KM_STD < 10000)  # 이전 3KM 구간 동안 평면 편차
```

1. 이상치 제거 (3/3)

```
#============================================================
# 이상치 제거 후
#============================================================
summary(weather_traf)
```

> 실행 결과

```
> #============================================================
> # 이상치 제거 후
> #============================================================
> summary(weather_traf)
   CONZONE             GISID              month               hour
 Length:218245      Length:218245      Length:218245      Length:218245
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character


  TA_MAX_2_STD      WS_MAX_6_STD      WS_MAX_2_MIN     TA_MIN_6_MAX      WS_AVG_2_STD
 Min.   :0.0000   Min.   :0.0000   Min.   :0.000   Min.   :-18.60   Min.   :0.0000
 1st Qu.:0.2000   1st Qu.:0.3894   1st Qu.:1.214   1st Qu.:  5.54   1st Qu.:0.1143
 Median :0.3333   Median :0.5249   Median :1.850   Median : 14.93   Median :0.1786
 Mean   :0.4318   Mean   :0.5588   Mean   :2.010   Mean   : 14.27   Mean   :0.2019
 3rd Qu.:0.5563   3rd Qu.:0.6908   3rd Qu.:2.643   3rd Qu.: 23.27   3rd Qu.:0.2625
 Max.   :6.9175   Max.   :4.5215   Max.   :6.975   Max.   : 37.00   Max.   :1.9500
  TA_MIN_6_STD       RN_3_MAX            Y              BUSlane        JD_3KM_STD
 Min.   :0.0000   Min.   : 0.0000   Min.   : 1.000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.7394   1st Qu.: 0.0000   1st Qu.: 3.000   1st Qu.:0.0000   1st Qu.:0.6156
 Median :1.1255   Median : 0.0000   Median : 4.000   Median :0.0000   Median :2.1437
 Mean   :1.3209   Mean   : 0.2293   Mean   : 4.582   Mean   :0.2539   Mean   :2.0178
 3rd Qu.:1.7234   3rd Qu.: 0.1250   3rd Qu.: 6.000   3rd Qu.:1.0000   3rd Qu.:3.3039
 Max.   :9.6049   Max.   :19.5000   Max.   :19.000   Max.   :1.0000   Max.   :5.4212
   PM_3KM_STD       JD_1KM_AVG       KS_3KM_AVG      BY_3KM_STD       BY_3KM_AVG
 Min.   :   0.0   Min.   :-6.450   Min.   :1.000   Min.   :0.0000   Min.   :1.000
 1st Qu.: 141.4   1st Qu.: 0.000   1st Qu.:2.000   1st Qu.:0.0000   1st Qu.:1.000
 Median : 286.1   Median : 2.173   Median :2.667   Median :0.9428   Median :2.000
 Mean   : 677.3   Mean   : 2.480   Mean   :2.609   Mean   :0.9371   Mean   :1.989
 3rd Qu.: 795.2   3rd Qu.: 6.500   3rd Qu.:3.333   3rd Qu.:1.8856   3rd Qu.:2.667
 Max.   :9871.2   Max.   : 6.500   Max.   :4.000   Max.   :2.3570   Max.   :5.667
   SPEEDLIMIT       AVG_SPEED        VOLUME_ALL
 Min.   :100.0   Min.   :  3.54   Min.   :   1
 1st Qu.:100.0   1st Qu.: 90.76   1st Qu.: 589
 Median :100.0   Median : 96.48   Median :1246
 Mean   :101.2   Mean   : 95.61   Mean   :1663
 3rd Qu.:100.0   3rd Qu.:101.48   3rd Qu.:2417
 Max.   :110.0   Max.   :169.52   Max.   :8689
```

기상청

Ⅳ. 모형 구축

1. 분석 데이터 셋
2. 모형 구축

1. 분석 데이터셋 (1/2)

- **H2O 서버 세팅**

- H2O는 빅데이터 처리를 위한 분산처리 프로세스를 통해 빠른 처리속도를 지원하고 다양한 머신러닝 분석 기술을 제공하는 패키지

- H2O GBM(Gradient Boosting Machine)을 활용하여분석

```
#============================================================
# 로컬에H2O 가상서버 설정하기
#============================================================
h2o.init(nthreads = -1) # h2o 서버 세팅
h2o.removeAll()
#============================================================
```

- **h2o.init()으로 H2O 자바가상머신을세팅**

> 실행 결과

```
> #============================================================
> # 로컬에 H2O 가상서버 설정하기
> #============================================================
> h2o.init(nthreads = -1) # h2o 서버 세팅
 Connection successful!

R is connected to the H2O cluster:
    H2O cluster uptime:         33 minutes 33 seconds
    H2O cluster timezone:       Asia/Seoul
    H2O data parsing timezone:  UTC
    H2O cluster version:        3.34.0.3
    H2O cluster version age:    27 days
    H2O cluster name:           H2O_started_from_R_owen_vzf373
    H2O cluster total nodes:    1
    H2O cluster total memory:   7.99 GB
    H2O cluster total cores:    12
    H2O cluster allowed cores:  12
    H2O cluster healthy:        TRUE
    H2O Connection ip:          localhost
    H2O Connection port:        54321
    H2O Connection proxy:       NA
    H2O Internal Security:      FALSE
    H2O API Extensions:         Amazon S3, Algos, AutoML, Core V3, TargetEncoder, Core V4
    R Version:                  R version 4.0.2 (2020-06-22)

> h2o.removeAll()
```

1. 분석 데이터셋 (2/2)

- **데이터 셋 분할**

- 모델 학습 및 최적화, 테스트를 위해 데이터 셋을 train_data, valid_data, test_data로 나눔

```
#============================================================
# 데이터 분할
#============================================================
# set x and y
df_h2o_1 <- as.h2o(weather_traf)
x        <- which(colnames(df_h2o_1)!="Y")
y        <- which(colnames(df_h2o_1)=="Y")
colnames(df_h2o_1)[x]
colnames(df_h2o_1)[y]

# split dataset
split <- h2o.splitFrame(df_h2o_1, ratios = c(0.6, 0.2), seed=1234)
train <- split[[1]]
valid <- split[[2]]
test <- split[[3]]

#============================================================
h2o_train <- as.h2o(train)
h2o_valid <- as.h2o(valid)
h2o_test <- as.h2o(test)
```

- **sample()으로 데이터 셋을 분할**
  1) train → 0.6, valid → 0.2, test → 0.2
- **as.h2o()로 분석 데이터셋을 H2OFrame구조의 데이터로 변환**

> 실행 결과

```
> #============================================================
> # 데이터 분할
> #============================================================
> # set x and y
> df_h2o_1 <- as.h2o(weather_traf)
  |============================================================================================================================================================| 100%
> x        <- which(colnames(df_h2o_1)!="Y")
> y        <- which(colnames(df_h2o_1)=="Y")
> colnames(df_h2o_1)[x]
 [1] "CONZONE"      "GISID"        "month"        "hour"         "TA_MAX_2_STD" "WS_MAX_6_STD" "WS_MAX_2_MIN" "TA_MIN_6_MAX" "WS_AVG_2_STD" "TA_MIN_6_STD"
[11] "RN_3_MAX"     "BUSlane"      "JD_3KM_STD"   "PM_3KM_STD"   "JD_1KM_AVG"   "KS_3KM_AVG"   "BY_3KM_STD"   "BY_3KM_AVG"   "SPEEDLIMIT"   "AVG_SPEED"
[21] "VOLUME_ALL"
> colnames(df_h2o_1)[y]
[1] "Y"
>
> # split dataset
> split <- h2o.splitFrame(df_h2o_1, ratios = c(0.6, 0.2), seed=1234)
> train <- split[[1]]
> valid <- split[[2]]
> test <- split[[3]]
>
> #============================================================
> h2o_train <- as.h2o(train)
> h2o_valid <- as.h2o(valid)
> h2o_test <- as.h2o(test)
```

2. 모형구축(1/4)

- **하이퍼 파라미터 최적화(Hyper-parameter Optimization)**

- GBM 모델의 하이퍼 파라미터를 조정하여 RMSE(Root Mean Square Error)가 가장 높은 모형 선택

```
#============================================================
# 모형 튜닝 자동화
# ============================================================
# catesian grid search
grid_params <- list(learn_rate = c(0.1, 0.05, 0.001),
                    max_depth = c(12,15,18))

# Train and validate a random grid of GBMs
gbm_grid <- h2o.grid("gbm",
                     x = x,
                     y = y,
                     grid_id = "gbm_grid",
                     training_frame = train,
                     validation_frame = valid,
                     hyper_params = grid_params,
                     col_sample_rate = 0.85,
                     col_sample_rate_change_per_level = 0.9,
                     col_sample_rate_per_tree = 0.6,
                     histogram_type = "UniformAdaptive",
                     min_rows = 16384.0,
                     min_split_improvement = 0.0,
                     nbins = 1024,
                     nbins_cats = 2048,
                     ntrees = 148,
                     seed = 1234
)

# RMSE가 낮은순으로 정렬하기
gbm_gridperf <- h2o.getGrid(grid_id = "gbm_grid",
                            sort_by = "rmse",
                            decreasing = FALSE)

gbm_gridperf
```

- **List()로 하이퍼 파라미터 조합 리스트를 생성**
- **h2o.grid()로 하이퍼 파라미터 조합들을 활용하여 모형을 구축한 "gbm_grid" 생성**
- **h2o.getGrid()로 "gbm_grid"의 모형 구축 결과를 RMSE가 낮은 순으로 정렬**

2. 모형구축(2/4)

> 실행 결과

```
> #============================================================
> # 모형 튜닝 자동화
> #============================================================
> # catesian grid search
> grid_params <- list(learn_rate = c(0.1, 0.05, 0.001),
+                     max_depth = c(12,15,18))
>
> # Train and validate a random grid of GBMs
> gbm_grid <- h2o.grid("gbm",
+                      x = x,
+                      y = y,
+                      grid_id = "gbm_grid",
+                      training_frame = train,
+                      validation_frame = valid,
+                      hyper_params = grid_params,
+                      col_sample_rate = 0.85,
+                      col_sample_rate_change_per_level = 0.9,
+                      col_sample_rate_per_tree = 0.6,
+                      histogram_type = "UniformAdaptive",
+                      min_rows = 16384.0,
+                      min_split_improvement = 0.0,
+                      nbins = 1024,
+                      nbins_cats = 2048,
+                      ntrees = 148,
+                      seed = 1234
+ )
  |=================================================| 100%
>
> # RMSE가 낮은순으로 정렬하기
> gbm_gridperf <- h2o.getGrid(grid_id = "gbm_grid",
+                             sort_by = "rmse",
+                             decreasing = FALSE)
> gbm_gridperf
H2O Grid Details
================

Grid ID: gbm_grid
Used hyper parameters:
  -  learn_rate
  -  max_depth
Number of models: 9
Number of failed models: 0

Hyper-Parameter Search Summary: ordered by increasing rmse
  learn_rate max_depth        model_ids              rmse
1        0.1        12 gbm_grid_model_1 1.8811574761339835
2        0.1        15 gbm_grid_model_4 1.8811574761339835
3        0.1        18 gbm_grid_model_7 1.8811574761339835
4       0.05        12 gbm_grid_model_2 1.9310361854700078
5       0.05        15 gbm_grid_model_5 1.9310361854700078
6       0.05        18 gbm_grid_model_8 1.9310361854700078
7      0.001        12 gbm_grid_model_3  2.135202932922323
8      0.001        15 gbm_grid_model_6  2.135202932922323
9      0.001        18 gbm_grid_model_9  2.135202932922323
```

2. 모형 구축(3/4)

- **최종 모형 선택**

- 모형 성능을 RMSE(Root Mean Square Error) 기준으로 모델 선택.

- max_depth = 12, learn_rate = 0.1의 모델 선택

```
#==========================================================
# 최종 모형 선택
#==========================================================
# create Model
Traffic_GBM_Model <- h2o.gbm(training_frame = train
                        ,validation_frame = valid
                        ,col_sample_rate = 0.85
                        ,col_sample_rate_change_per_level = 0.9
                        ,col_sample_rate_per_tree = 0.6
                        ,histogram_type = "UniformAdaptive"
                        ,max_depth = 12
                        ,learn_rate = 0.1
                        ,min_rows = 16384.0
                        ,min_split_improvement = 0.0
                        ,nbins = 1024
                        ,nbins_cats = 2048
                        ,sample_rate = 0.8
                        ,model_id = "Traffic_GBM_Model"
                        ,ntrees = 148
                        ,x = x
                        ,y = y
                        ,seed = 1234
                        )
# __Model Summary ----
Traffic_GBM_Model
```

- **h2o.gbm()로 활용하여 모형을 구축한 "Traffic_GBM_Model"를 생성**

> 실행 결과

```
> #==========================================================
> # 모형 선택
> #==========================================================
> # create Model
> Traffic_GBM_Model <- h2o.gbm(training_frame = train
+                         ,validation_frame = valid
+                         ,col_sample_rate = 0.85
+                         ,col_sample_rate_change_per_level = 0.9
+                         ,col_sample_rate_per_tree = 0.6
+                         ,histogram_type = "UniformAdaptive"
+                         ,max_depth = 12
+                         ,learn_rate = 0.1
+                         ,min_rows = 16384.0
+                         ,min_split_improvement = 0.0
+                         ,nbins = 1024
+                         ,nbins_cats = 2048
+                         ,sample_rate = 0.8
+                         ,model_id = "Traffic_GBM_Model"
+                         ,ntrees = 148
+                         ,x = x
+                         ,y = y
+                         ,seed = 1234
+                         )
  |=============================================================================================| 100%
```

2. 모형 구축(4/4)

> 실행 결과

```
> # __Model Summary ----
> Traffic_GBM_Model
Model Details:
==============

H2ORegressionModel: gbm
Model ID:  Traffic_GBM_Model
Model Summary:
  number_of_trees number_of_internal_trees model_size_in_bytes min_depth max_depth mean_depth min_leaves
1             148                      148               18222         3         5    3.79730          4
  max_leaves mean_leaves
1          6     5.13514


H2ORegressionMetrics: gbm
** Reported on training data. **

MSE:  3.689727135
RMSE:  1.920866246
MAE:  1.536235075
RMSLE:  0.371914751
Mean Residual Deviance :  3.689727135


H2ORegressionMetrics: gbm
** Reported on validation data. **

MSE:  3.652697819
RMSE:  1.911203239
MAE:  1.528952159
RMSLE:  0.3712770604
Mean Residual Deviance :  3.652697819
```

기상청

Ⅴ. 모형 검증

1. 변수 중요도 파악
2. 최종 모형 선택
3. 모형 성능 및 예측력 파악

1. 변수 중요도 파악 (1/2)

- **생성된 모델에 대해 가장 많은 영향을 주는 변수 중요도 파악**

```
#============================================================
# 변수 중요도 파악
#============================================================
feature_list <- h2o.varimp(Traffic_GBM_Model)

feature_list

h2o.varimp_plot(Traffic_GBM_Model)
```

  - **h2o.varimp()함수를 사용해 변수 중요도 테이블 생성**
  - **생성된 테이블은 feature_list로 지정**
  - **h2o.varimp_plot()을 사용해 변수 중요도 plot 생성**

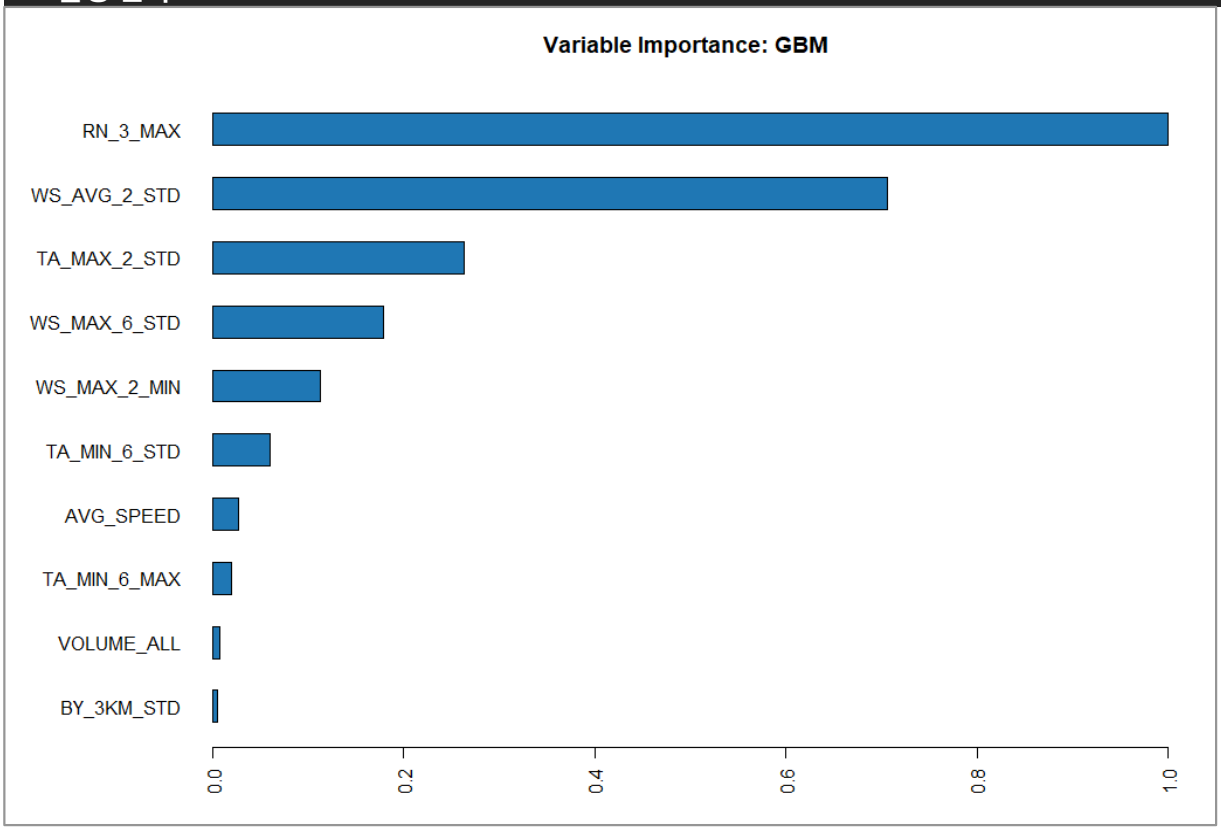
> 실행 결과

```
> #============================================================
> # __변수 중요도 파악 ----
> #============================================================
> feature_list <- h2o.varimp(Traffic_GBM_Model)
> feature_list
Variable Importances:
        variable relative_importance scaled_importance percentage
1       RN_3_MAX       257505.046875          1.000000   0.419386
2  WS_AVG_2_STD       177028.468750          0.687476   0.288318
3  TA_MAX_2_STD        69871.601562          0.271341   0.113796
4  WS_MAX_6_STD        44494.917969          0.172792   0.072467
5  WS_MAX_2_MIN        34576.597656          0.134275   0.056313
6  TA_MIN_6_STD        12870.435547          0.049981   0.020961
7     AVG_SPEED         6731.551758          0.026141   0.010963
8  TA_MIN_6_MAX         4752.227051          0.018455   0.007740
9    VOLUME_ALL         2204.141602          0.008560   0.003590
10   JD_3KM_STD         1363.685181          0.005296   0.002221
11   BY_3KM_STD         1041.142334          0.004043   0.001696
12   BY_3KM_AVG          838.109802          0.003255   0.001365
13   PM_3KM_STD          439.797424          0.001708   0.000716
14   JD_1KM_AVG          206.174698          0.000801   0.000336
15   KS_3KM_AVG           81.047997          0.000315   0.000132
16   SPEEDLIMIT            0.000000          0.000000   0.000000
```

*** R과 Python의 랜덤 효과가 상이하여 모형 결과가 다를 수 있습니다.**

1. 변수 중요도 파악 (2/2)

> 실행 결과



*** R과 Python의 랜덤 효과가 상이하여 모형 결과가 다를 수 있습니다.**

2. 최종 모형 선택 (1/3)

- 생성된 변수 중요도에 따른 모델 평가
  - 변수 중요도 상위 3,5,7,9개별 4개 모델의 RMSE 평가

```
#============================================================
#  최종 모형 선택
#============================================================
feature_test <- map_dfr(c(3,5,7,9), function(z) {
  train_feature <- train[,c("CONZONE","GISID", "month", "hour", "Y",
                            feature_list$variable[1:z])]
  valid_feature <- valid[,c("CONZONE","GISID", "month", "hour", "Y",
                            feature_list$variable[1:z])]

  x         <- which(colnames(train_feature)!="Y")
  y         <- which(colnames(valid_feature)=="Y")

  # create Model
  Traffic_GBM_Model_feature <- h2o.gbm(training_frame = train_feature
                                       ,validation_frame = valid_feature
                                       ,col_sample_rate = 0.85
                                       ,col_sample_rate_change_per_level = 0.9
                                       ,col_sample_rate_per_tree = 0.6
                                       ,histogram_type = "UniformAdaptive"
                                       ,max_depth = 12
                                       ,learn_rate = 0.1
                                       ,min_rows = 16384.0
                                       ,min_split_improvement = 0.0
                                       ,nbins = 1024
                                       ,nbins_cats = 2048
                                       ,sample_rate = 0.8
                                       ,model_id = "Traffic_GBM_Model"
                                       ,ntrees = 148
                                       ,x = x
                                       ,y = y
                                       ,seed = 1234
  )

  # __Model rmse  ----
  data.frame(
    feature = z,
    rmse = Traffic_GBM_Model_feature@model$validation_metrics@metrics$RMSE
  )
})

feature_test
```

*** R과 Python의 랜덤 효과가 상이하여 모형 결과가 다를 수 있습니다.**

2. 최종 모형 선택 (2/3)

- 3,5,7,9개의 변수 선택 위해 반복문 사용
- map_dfr()함수를 사용해 반복분 사용
  - 상위 3,5,7,9 개의 변수
- 반복문으로 생성된 각각의 train/valid set는 train_feature/valid_feature라는 변수 생성
- H2o.gbm()함수를 사용해 각각의 모델 생성후 Traffic_GBM_Model_feature라는 모델 생성
- 각 모델에 대한 RMSE를 data.frame()함수를 통해 생성
- 모델의 RMSE는 다음의 코드 사용
  - Traffic_GBM_Model_feature@model$validation_metrics@metrics$RMSE
- RMSE 비교를 위해 최종적으로 feature_test라는 데이터 테이블 생성

> 실행 결과

```
> #============================================================
> # __최종 모형 선택 ----
> #============================================================
> feature_test <- map_dfr(c(3,5,7,9), function(z) {
+   train_feature <- train[,c("CONZONE","GISID", "month", "hour", "Y",
+                             feature_list$variable[1:z])]
+   valid_feature <- valid[,c("CONZONE","GISID", "month", "hour", "Y",
+                             feature_list$variable[1:z])]
+
+   x        <- which(colnames(train_feature)!="Y")
+   y        <- which(colnames(valid_feature)=="Y")
+
+   # create Model
+   Traffic_GBM_Model_feature <- h2o.gbm(training_frame = train_feature
+                                        ,validation_frame = valid_feature
+                                        ,col_sample_rate = 0.85
+                                        ,col_sample_rate_change_per_level = 0.9
+                                        ,col_sample_rate_per_tree = 0.6
+                                        ,histogram_type = "UniformAdaptive"
+                                        ,max_depth = 12
+                                        ,learn_rate = 0.1
+                                        ,min_rows = 16384.0
+                                        ,min_split_improvement = 0.0
+                                        ,nbins = 1024
+                                        ,nbins_cats = 2048
+                                        ,sample_rate = 0.8
+                                        ,model_id = "Traffic_GBM_Model"
+                                        ,ntrees = 148
+                                        ,x = x
+                                        ,y = y
+                                        ,seed = 1234
+   )
+
+   # Model rmse
+   data.frame(
+     feature = z,
+     rmse = Traffic_GBM_Model_feature@model$validation_metrics@metrics$RMSE
+   )
+ })
  |======================================================================================================================| 100%
  |======================================================================================================================| 100%
  |======================================================================================================================| 100%
  |======================================================================================================================| 100%
```

*** R과 Python의 랜덤 효과가 상이하여 모형 결과가 다를 수 있습니다.**

2. 최종 모형 선택 (3/3)

- 변수별 RMSE 결과
  - 상위 변수 3개: 1.945368
  - 상위 변수 5개: 1.926327
  - 상위 변수 7개; 1.916569
  - 상위 변수 9개: 1.914549
- 따라서 상위 변수 3개를 사용하여 최종 모델 선정
  - RN_3_MAX: 이전 3시간 동안 강수의 최대
  - WS_AVG_2_STD: 이전 2시간 동안 평균 풍속의 편차
  - TA_MAX_2_STD: 이전 2시간 동안 최대 기온의 편차
  - WS_MAX_6_STD: 이전 6시간 동안 최대 풍속의 편차
  - WS_MAX_2_MIN: 이전 2시간 동안 최대 풍속의 최소
  - TA_MIN_6_STD: 이전 6시간 동안 최소 기온의 편차
  - AVG_SPEED: 1시간 평균 속도
  - TA_MIN_6_MAX: 이전 6시간 동안 최소 기온의 최대
  - VOLUME_ALL: 1시간 총 교통량

> 실행 결과

```
> feature_test
  feature        rmse
1       3 1.945368638
2       5 1.926327761
3       7 1.916569710
4       9 1.914549754
```

*** R과 Python의 랜덤 효과가 상이하여 모형 결과가 다를 수 있습니다.**

3. 모형 성능 및 예측력 파악(1/4)

- **모형 성능 및 예측력 검증**

- 최종 선택한 변수를 대상으로 테스트 데이터를 예측하고, RMSE(Root Mean Square Error)로 결과 확인

```
#============================================================
# 모형 성능
#============================================================
train_feature <- train[,c("CONZONE","GISID", "month", "hour", "Y",
                          feature_list$variable[1:9])]
test_feature <- valid[,c("CONZONE","GISID", "month", "hour", "Y",
                          feature_list$variable[1:9])]

x       <- which(colnames(train_feature)!="Y")
y       <- which(colnames(test_feature)=="Y")

# create Model
Traffic_GBM_Model_final <- h2o.gbm(training_frame = train_feature
                                   ,validation_frame = test_feature
                                   ,col_sample_rate = 0.85
                                   ,col_sample_rate_change_per_level = 0.9
                                   ,col_sample_rate_per_tree = 0.6
                                   ,histogram_type = "UniformAdaptive"
                                   ,max_depth = 12
                                   ,learn_rate = 0.1
                                   ,min_rows = 16384.0
                                   ,min_split_improvement = 0.0
                                   ,nbins = 1024
                                   ,nbins_cats = 2048
                                   ,sample_rate = 0.8
                                   ,model_id = "Traffic_GBM_Model"
                                   ,ntrees = 148
                                   ,x = x
                                   ,y = y
                                   ,seed = 1234
)
# __Model Summary ----
Traffic_GBM_Model_final
```

- **h2o.gbm()로 활용하여 모형을 구축한 "Traffic_GBM_Model_final"를 생성**
- **변수는 상위 9개를 선택하여 새로 train_feature/test_feature로 생성**

3. 모형 성능 및 예측력 파악(2/4)

> 실행 결과

```
> #============================================================
> # 모형 성능
> #============================================================
> train_feature <- train[,c("CONZONE","GISID", "month", "hour", "Y",
+                           feature_list$variable[1:9])]
> test_feature <- valid[,c("CONZONE","GISID", "month", "hour", "Y",
+                          feature_list$variable[1:9])]
>
> x        <- which(colnames(train_feature)!="Y")
> y        <- which(colnames(test_feature)=="Y")
>
> # create Model
> Traffic_GBM_Model_final <- h2o.gbm(training_frame = train_feature
+                                    ,validation_frame = test_feature
+                                    ,col_sample_rate = 0.85
+                                    ,col_sample_rate_change_per_level = 0.9
+                                    ,col_sample_rate_per_tree = 0.6
+                                    ,histogram_type = "UniformAdaptive"
+                                    ,max_depth = 12
+                                    ,learn_rate = 0.1
+                                    ,min_rows = 16384.0
+                                    ,min_split_improvement = 0.0
+                                    ,nbins = 1024
+                                    ,nbins_cats = 2048
+                                    ,model_id = "Traffic_GBM_Model"
+                                    ,ntrees = 148
+                                    ,x = x
+                                    ,y = y
+                                    ,seed = 1234
+ )
  |=============================================================================================================| 100%

> # __Model Summary ----
> Traffic_GBM_Model_final
Model Details:
==============

H2ORegressionModel: gbm
Model ID:  Traffic_GBM_Model
Model Summary:
  number_of_trees number_of_internal_trees model_size_in_bytes min_depth max_depth mean_depth min_leaves
1             148                      148               20585         3         6    4.67568          5
  max_leaves mean_leaves
1          7     6.43919


H2ORegressionMetrics: gbm
** Reported on training data. **

MSE:  3.605666478
RMSE:  1.898859257
MAE:  1.518613489
RMSLE:  0.3675048967
Mean Residual Deviance :  3.605666478


H2ORegressionMetrics: gbm
** Reported on validation data. **

MSE:  3.570140433
RMSE:  1.889481525
MAE:  1.510972831
RMSLE:  0.3668833225
Mean Residual Deviance :  3.570140433
```

3. 모형 성능 및 예측력 파악(3/4)

- **모형 성능 및 예측력 확인**

-최종 선택한 모형으로 테스트 데이터를 예측하고, RMSE(Root Mean Square Error)로 결과 확인

```
#============================================================
# 모형 성능
#============================================================
# Predict on test set
Pred_conversion <- h2o.predict(object = Traffic_GBM_Model_final, newdata =test_featurel)

Pred_conversion <- pred_conversion %>%
  as_tibble()

comp <- cbind(as_tibble(test_feature$Y),pred_conversion)

# RMSE
rmse(comp$Y, comp$predict)
```

- **as.h2o() 함수를 사용해 h2o모형에 맞는 데이터 프레임 생성**
- **h2o.predict()함수를 사용해 모델 평가**
- **rmse()함수를 사용해 모델의 rmse 파악**
  - **예측값에 대한 RMSE = 1.88948**

> 실행 결과

```
> #============================================================
> # 모형 성능
> #============================================================
> # Predict on test set
> pred_conversion <- h2o.predict(object = Traffic_GBM_Model_final, newdata = test_feature)
  |=============================================================================================================| 100%
>
> pred_conversion <- pred_conversion %>%
+   as_tibble()
>
> comp <- cbind(as_tibble(test_feature$Y),pred_conversion)
>
> # __RMSE ----
> rmse(comp$Y, comp$predict)
[1] 1.889481525
```

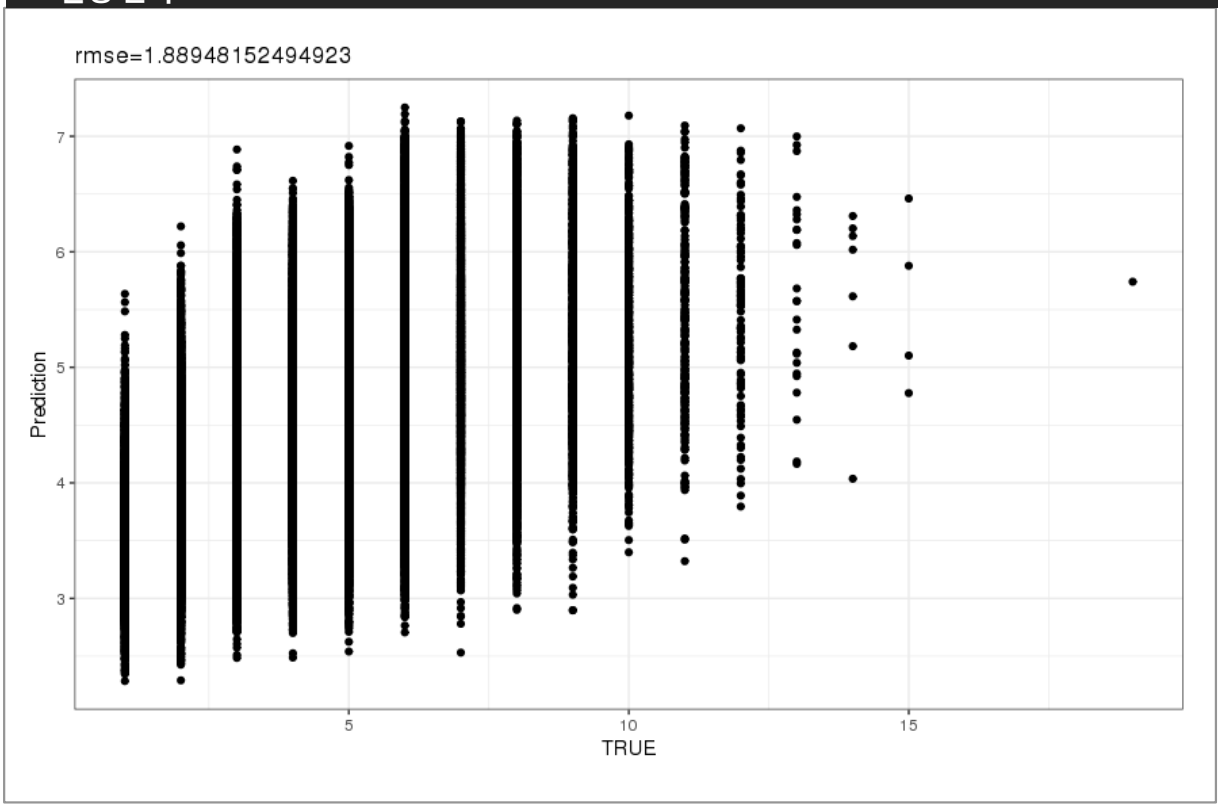
3. 모형 성능 및 예측력 파악(4/4)

- **RMSE 그래프 파악**

```
#=============================================================
#  RMSE
#=============================================================
comp %>%
  ggplot(aes(x = Y, y = predict)) +
  geom_point() +
  labs(title= paste0("rmse=", rmse(comp$Y, comp$predict)), x = "TRUE", y = "Prediction") +
  theme_bw()
```

  - **RMSE 그래프 파악**
  - **ggplot을 사용하여 실제값과 예측값의 분포 해석**

> 실행 결과



본 문서의 내용은 기상청 날씨마루(http://big.kma.go.kr)의
분석 플랫폼 활용을 위한 R 프로그래밍 교육 자료입니다.