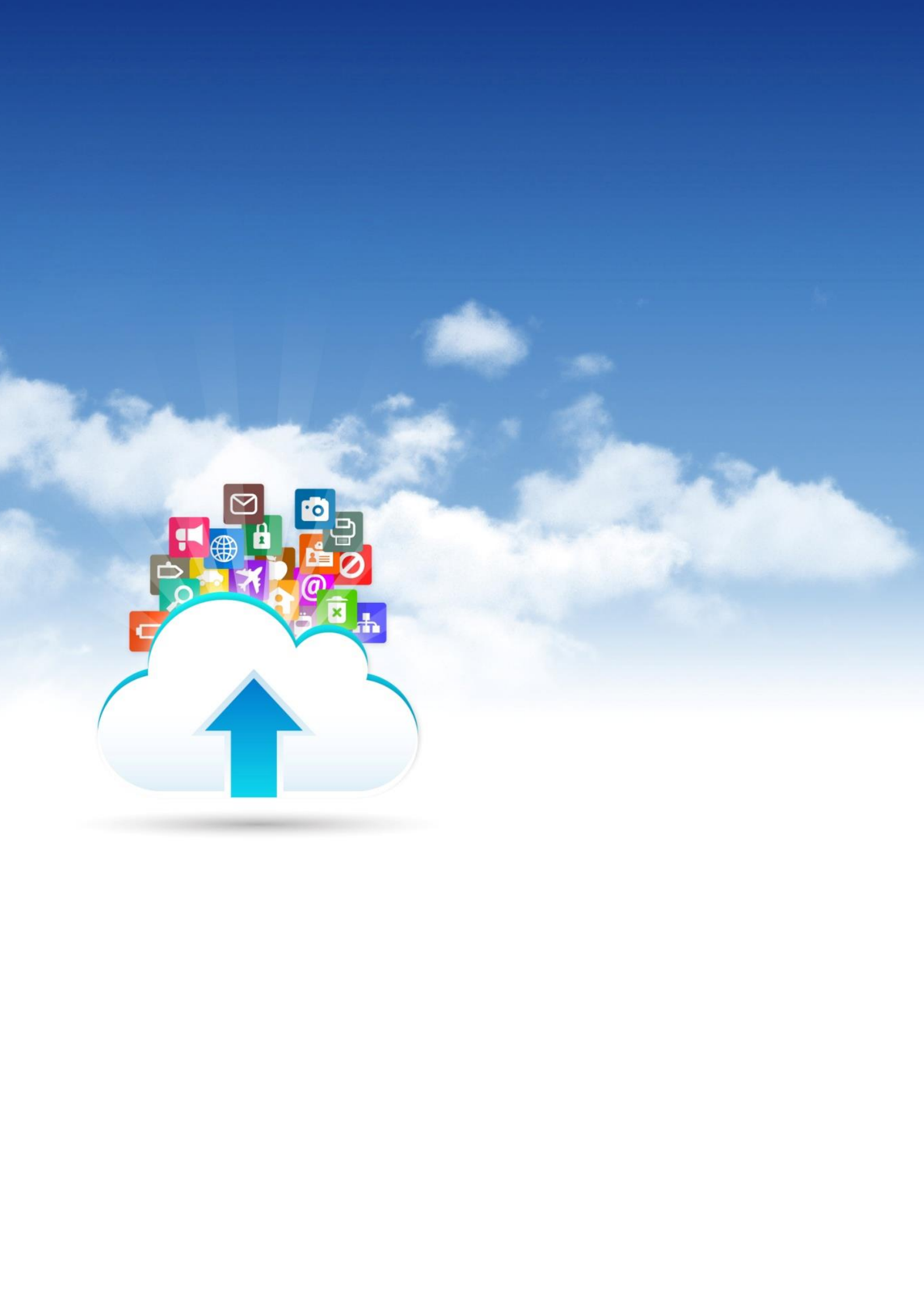
빅데이터 분석

: 농산물편

미래창조과학부 NIA 한국정보화진흥원 K-ICT 빅데이터센터

CONTENTS



Ⅰ. 개요

I 개요

학습목표

원 데이터에서는 유의미한 정보 추출이 불가능하므로 농축산물 가격 데이터와 기상 데이터 등을 분석 가능한 형태의 데이터로 변환한다.

농축산물 가격 데이터를 이용한 데이터 시각화 및 데이터간의 연관성 분석 과정을 상세히 소개하고, 그 의미를 살펴본다.

고급 클러스터링 기법을 이용한 연관성 분석을 실시하고 날씨 및 뉴스 데이터를 분석하여 농축산물 가격변동의 원인을 유추해 본다.

- 필요 패키지 : `plyr`, `ggplot2`, `TSclust`, `zoo`, `gridExtra`, `urca`, `stringr`, `corrplot`, `RColorBrewer`, `gtable`
- 필요 데이터 : `product.csv`, `weather.csv`, `code.csv`

Ⅱ. 상관 분석을 통한 지역별 돼지고기 가격 연관성

1. 필요 패키지 불러오기
2. 데이터 불러오기
3. 데이터 가공하기
4. 상관분석 및 데이터 시각화
5. 데이터 저장하기

Ⅱ 상관 분석을 통한 지역별 돼지고기 가격 연관성 분석

한국농수산식품유통공사에서 제공받은 2011 년~2013 년 농축산물의 소매가격 정보 중 지역별 돼지고기 평균 소매가격 자료를 이용하여 소매가격의 지역간 상관관계를 분석하고 이를 시계열 그림을 통하여 시각화한다.

원 데이터를 분석 가능한 형태의 데이터로 변환하는 방법, 농축산물 변수의 상관관계 분석 및 데이터 시각화 등을 실습한다.

1. 필요 패키지 불러오기

아래 R 명령어는 여러 개의 패키지를 로딩하는데 유용한 수행방법이다.

```
library1 <- c("plyr", "ggplot2", "stringr", "zoo", "corrplot", "RColorBrewer")
unlist(lapply(library1, require, character.only = TRUE ))
```

➢ 각 패키지의 역할을 간략히 설명하면 다음과 같음

패키지 명	설명
plyr	데이터 핸들링을 하기 위한 라이브러리
ggplot2	시각화 기능 라이브러리
stringr	문자열 핸들링을 하기 위한 라이브러리
zoo	문자형 데이터를 데이트 형식으로 변환하기 위한 라이브러리
corrplot	상관분석을 위한 라이브러리
RcolorBrewer	색상 처리 기능 라이브러리

2. 데이터 불러오기

CSV 형태의 농축산물 데이터와 기상 데이터는 다음과 같이 읽어 들인다.

```
product <- read.csv("Data/product.csv", header=T, fileEncoding="UTF-8")
code <- read.csv("Data/code.csv", header=T, fileEncoding="UTF-8")
```

- read.csv() : 이 함수는 csv 파일을 읽는 기능을 함

3. 데이터 가공하기

농축산물 데이터는 한국농수산식품유통공사에서 제공받은 2011 년~2013 년 농축산물 데이터를 수집하여 분석 목적에 맡게 품목, 지역 및 마트를 비식별화 처리를 통해 분석에 용이하게 편집하여 제공한다.

농산물 소매가격정보는 일반 농산물, 친환경 농산물 소매가격으로 일자, 부류, 품목, 지역, 마트 별 구분으로 구성이 되어 있다.

```
head(product, n=10)
```

```
##           일자 부류코드 품목코드 지역코드 마트코드   가격
## 1  2011-01-03      100      111     3511   350401 39600
## 2  2011-01-03      100      111     3311   330401 40000
## 3  2011-01-03      100      111     1101   110251 37000
## 4  2011-01-03      100      111     1101   110401 43900
## 5  2011-01-03      100      111     1101   110402 39800
## 6  2011-01-03      100      111     1101   110403 39600
## 7  2011-01-03      100      111     1101   110405 42000
## 8  2011-01-03      100      111     1101   110406 43800
## 9  2011-01-03      100      111     2100   210022 42000
## 10 2011-01-03      100      111     2100   210401 36800
```

```
tail(product, n=10)
```

```
##               일자 부류코드 품목코드 지역코드 마트코드 가격
## 253409 2013-12-31      500      515     2401   240401 6500
## 253410 2013-12-31      500      515     2401   240403 5500
## 253411 2013-12-31      500      515     2501   250112 4000
## 253412 2013-12-31      500      515     2501   250401 5500
## 253413 2013-12-31      500      515     2501   250402 6990
## 253414 2013-12-31      500      515     2601   260401 6500
## 253415 2013-12-31      500      515     3111   310001 4330
## 253416 2013-12-31      500      515     3111   310402 5570
## 253417 2013-12-31      500      515     3113   312401 6500
## 253418 2013-12-31      500      515     3211   320001 5210
```

- product 을 구성하는 변수에 대한 설명은 다음과 같음
  - 부류코드 : 농축산물의 범주에 따라 분류한 코드. 100(식량작물), 200(채소류). 300(특용작물), 400(과일류), 500(축산물)

– 품목코드 : 품목에 따른 코드
– 지역코드 : 지역에 따른 코드
– 마트코드 : 마트에 따른 코드

- head() : 데이터의 앞부분을 출력함

- tail() : 데이터의 뒷부분을 출력함

R 서버에서 원활하게 데이터에 접근하기 위해 원시 데이터의 변수명을 한글에서 영어로 변환해준다.
데이터 product 의 변수명은 date(일자), category(부류코드), item(품목코드), region(지역코드), mart(마트코드), price(가격)로 변환된다.

```
colnames(product) <- c('date','category','item','region','mart','price')
```

- colnames() : 데이터 프레임의 열의 이름을 지정함

일별 농축산물의 평균 소매가격을 추출하기 위해 품목별 코드 번호를 확인하기 위한 category 오브젝트를 생성한다. 여기서 category 에는 code 데이터에서 구분코드설명이 품목코드에 해당하는 값만 추출되어 저장된다.

```
category <- subset(code, code$구분코드설명=="품목코드")
category
```

```
##    구분코드 구분코드설명 분류코드 분류코드설명
## 6      1200     품목코드      111          쌀
## 7      1200     품목코드      211         배추
## 8      1200     품목코드      214         상추
## 9      1200     품목코드      224         호박
## 10     1200     품목코드      245         양파
## 11     1200     품목코드      256     파프리카
## 12     1200     품목코드      312         참깨
## 13     1200     품목코드      411         사과
## 14     1200     품목코드      514     돼지고기
## 15     1200     품목코드      515       닭고기
```

- subset() : 데이터의 특정부분에 대해 조건을 만족하는 값을 반환해주는 함수임

R 서버에서 원활하게 데이터에 접근하기 위해 원시 데이터의 변수명을 한글에서 영어로 변환해준다.
데이터 category 의 변수명은 code(구분코드), exp(구분코드설명), item(분류코드), name(분류코드설명)으로 변환된다.

```
colnames(category) <- c('code', 'exp', 'item', 'name')
```

- colnames() : 데이터 프레임의 열의 이름을 지정함

분석 대상은 돼지고기 소매가격이므로 일반농산물 소매가격 데이터 파일 product 에서 품목코드(item)가 514 인 돼지고기에 대한 데이터만 추출하여 total.pig 데이터를 생성한다.

```
total.pig <-product[which(product$item==514),]
head(total.pig, n=10)
```

```
##           date category item region   mart price
## 211 2011-01-03      500  514   2200 220063  1500
## 212 2011-01-03      500  514   2200 220401  1380
## 213 2011-01-03      500  514   3911 390401  1980
## 214 2011-01-03      500  514   3511 350401  1380
## 215 2011-01-03      500  514   3311 330401  1890
## 216 2011-01-03      500  514   3211 320401  1280
## 217 2011-01-03      500  514   3145 310403  1280
## 218 2011-01-03      500  514   3111 310402  2050
## 219 2011-01-03      500  514   2601 260401  1980
## 220 2011-01-03      500  514   2501 250403  1380
```

- which() : 벡터 또는 배열에서 주어진 조건을 만족하는 값이 있는 곳의 색인을 찾음

- head() : 데이터의 앞부분을 출력함

지역별 돼지고기의 평균 소매가격을 추출하기 위해 지역별 코드 번호를 확인하기 위한 region 오브젝트를 생성한다.
여기서 region 에는 code 데이터에서 구분코드설명이 "지역코드"에 해당하는 값만 추출되어 저장된다.

```
region <- subset(code, code$구분코드설명=="지역코드")
region
```

```
##    구분코드 구분코드설명 분류코드 분류코드설명
## 16     1300     지역코드     1101         서울
## 17     1300     지역코드     2100          부산
## 18     1300     지역코드     2200          대구
## 19     1300     지역코드     2300          인천
## 20     1300     지역코드     2401          광주
## 21     1300     지역코드     2501          대전
## 22     1300     지역코드     2601          울산
## 23     1300     지역코드     3111          수원
## 24     1300     지역코드     3113        의정부
## 25     1300     지역코드     3145          용인
## 26     1300     지역코드     3211          춘천
## 27     1300     지역코드     3311          청주
## 28     1300     지역코드     3511          전주
## 29     1300     지역코드     3613          순천
## 30     1300     지역코드     3711          포항
## 31     1300     지역코드     3714          안동
## 32     1300     지역코드     3814          창원
## 33     1300     지역코드     3911          제주
```

- `subset()` : 데이터의 특정부분에 대해 조건을 만족하는 값을 반환해주는 함수임

R 서버에서 원활하게 데이터에 접근하기 위해 원시 데이터의 변수명을 한글에서 영어로 변환해준다.
데이터 region 의 변수명은 code(구분코드), exp(구분코드설명), region(분류코드), name(분류코드설명)으로 변환된다.

```
colnames(region) <- c('code', 'exp', 'region', 'name')
```

- colnames() : 데이터 프레임의 열의 이름을 지정함

지역코드에 대한 데이터 region 과 전체 돼지고기 가격에 대한 데이터 total.pig 를 지역변수(region)를 기준으로 하나의 데이터 day.pig 으로 만든다.

```
day.pig <- merge(total.pig, region, by="region", all=T)
head(day.pig,n=10)
```

```
##    region       date category item   mart price code      exp  name
## 1    1101 2012-07-18      500  514 110251  1800 1300 지역코드 서울
## 2    1101 2012-06-19      500  514 110402  1580 1300 지역코드 서울
## 3    1101 2012-07-05      500  514 110406  1780 1300 지역코드 서울
## 4    1101 2012-04-30      500  514 110407  1490 1300 지역코드 서울
## 5    1101 2011-09-05      500  514 110403  1980 1300 지역코드 서울
## 6    1101 2011-10-31      500  514 110251  1800 1300 지역코드 서울
## 7    1101 2012-04-30      500  514 110403  1750 1300 지역코드 서울
## 8    1101 2011-01-21      500  514 110406  2340 1300 지역코드 서울
## 9    1101 2012-03-26      500  514 110402  1750 1300 지역코드 서울
## 10   1101 2012-06-21      500  514 110406  1780 1300 지역코드 서울
```

- merge() : 두 데이터 프레임을 공통된 값을 기준으로 묶는 함수임

- head() : 데이터의 앞부분을 출력함

day.pig 데이터를 일별로 정렬한 후, 지역별로 돼지고기의 평균가격을 구하여 생성한 데이터 프레임을 지역별 이름으로 나누어 total.pig.mean 이라는 리스트 형태의 데이터를 생성한다.

```
total.pig.mean <- dlply(ddply(ddply(day.pig, .(date), summarise, name=na
me, region=region, price=price),.(date, name), summarise, mean.price=mea
n(price)), .(name))
```

- dlply() : 데이터 프레임 형태를 품목별로 **list** 형태로 출력함

- ddply() : 데이터 프레임을 분리하여 함수를 적용시킨 후 데이터 프레임 형태로 출력하는 함수임
  - **summarise** 는 다음에 나오는 식의 결과를 추가 변수로 출력하는 함수임

아래의 예제를 통해 함수 ddply()를 이해해 보도록 한다.

```
x <- data.frame(
    Date=as.Date(c('2013-10-01', '2013-10-02', '2013-10-02', '2013-10-02
', '2013-10-01', '2013-10-02', '2013-10-02')),
    Category=factor(c('First', 'First', 'First', 'Second', 'Third', 'Thi
rd', 'Second')),
    Frequency=c(10, 15, 5, 2, 14, 20, 3))
head(x)
```

```
##         Date Category Frequency
## 1 2013-10-01    First        10
## 2 2013-10-02    First        15
## 3 2013-10-02    First         5
## 4 2013-10-02   Second         2
## 5 2013-10-01    Third        14
## 6 2013-10-02    Third        20
```

- 예제에 사용될 data frame 형태인 **x** 를 생성함
  - 3 개의 변수로 이루어져 있음
  - Date 값이 2013-10-02 이고 Category 가 First 인 자료가 두 개이며 이를 합쳐서 하나의 자료로 나타내야 함

데이터 프레임 **x** 의 자료 중에 Date 와 Category 가 같은 자료들의 Frequency 값을 합하여 하나의 자료 값으로 만든 후 데이터 프레임 형태로 출력해보도록 한다.

```
ddply(x, .(Date, Category), summarize, Sum_F=sum(Frequency))
```

```
##         Date Category Sum_F
## 1 2013-10-01    First    10
## 2 2013-10-01    Third    14
## 3 2013-10-02    First    20
## 4 2013-10-02   Second     5
## 5 2013-10-02    Third    20
```

다음은 함수 dlply()에 대한 예제이다.

데이터 프레임 **x** 를 생성한 후 dlply() 함수를 적용하여 리스트 형태로 출력해 본다.

```
x <- data.frame(
    Date=as.Date(c('2013-10-01', '2013-10-02', '2013-10-02', '2013-10-02
', '2013-10-01', '2013-10-02', '2013-10-02')),
    Category=factor(c('First', 'First', 'First', 'Second', 'Third', 'Thi
rd', 'Second')),
    Frequency=c(10, 15, 5, 2, 14, 20, 3))
```

- 예제에 사용될 data frame 형태인 **x** 를 생성함
  - 3 개의 변수로 이루어져 있음

```
dlply(x, .(Date), summarize, Sum_F=sum(Frequency))
```

```
## $`2013-10-01`
##   Sum_F
## 1    24
##
## $`2013-10-02`
##   Sum_F
## 1    45
##
## attr(,"split_type")
## [1] "data.frame"
## attr(,"split_labels")
##         Date
## 1 2013-10-01
## 2 2013-10-02
```

각 지역별 데이터의 크기를 확인하기 위해 다음과 같은 절차를 시행한다.

```
for (i in 1 : length(total.pig.mean)){
  cat(names(total.pig.mean)[i],"의 데이터의 길이는", nrow(total.pig.mean
[[i]]),"이다","\n")
}
```

```
## 광주 의 데이터의 길이는 745 이다
## 대구 의 데이터의 길이는 745 이다
## 대전 의 데이터의 길이는 745 이다
## 부산 의 데이터의 길이는 745 이다
## 서울  의 데이터의 길이는 745 이다
## 수원 의 데이터의 길이는 745 이다
## 순천 의 데이터의 길이는 477 이다
## 안동 의 데이터의 길이는 468 이다
## 용인 의 데이터의 길이는 268 이다
## 울산 의 데이터의 길이는 745 이다
## 의정부 의 데이터의 길이는 477 이다
## 인천 의 데이터의 길이는 745 이다
## 전주 의 데이터의 길이는 745 이다
## 제주 의 데이터의 길이는 745 이다
## 창원 의 데이터의 길이는 477 이다
## 청주 의 데이터의 길이는 745 이다
## 춘천 의 데이터의 길이는 743 이다
## 포항 의 데이터의 길이는 476 이다
```

- cat() : 주어진 값을 output 으로 출력함
- names() : 리스트로 부터 변수명을 얻어 출력함
- nrow() : 행렬형태 데이터에 대한 row 수를 읽어주는 기능을 함

day.pig 데이터에서 데이터 길이가 맞지 않은 일곱 지역을 제거하여 day.pig 데이터를 새롭게 생성한다.

```
day.pig  <- day.pig [! day.pig$name %in% c("의정부","용인","창원","안동",
"포항","순천","춘천" ),]
```

- 데이터[! 변수 %in% 조건,] : 데이터에서 변수의 조건에 맞는 열을 제거함
  - %in% : 조건에 대해 일치여부를 boolean 형으로 출력하는 기능의 연산자임

day.pig 데이터를 지역(region), 일자(date)별로 돼지고기의 평균가격을 구하여 pig.region.daily.mean 데이터를 생성한다.

```
pig.region.daily.mean <- ddply(day.pig, .(name, region, date), summaris
e,  mean.price=mean(price))
head(pig.region.daily.mean, n=10)
```

```
##    name region       date mean.price
## 1  광주   2401 2011-01-03       1610
## 2  광주   2401 2011-01-04       1610
## 3  광주   2401 2011-01-05       1610
## 4  광주   2401 2011-01-06       1390
## 5  광주   2401 2011-01-07       1390
## 6  광주   2401 2011-01-10       1390
## 7  광주   2401 2011-01-11       1390
## 8  광주   2401 2011-01-12       1390
## 9  광주   2401 2011-01-13       1635
## 10 광주   2401 2011-01-14       1735
```

- ddply() : 데이터 프레임을 분리하여 함수를 적용시킨 후 데이터 프레임 형태로 출력하는 함수임
  - **summarise** 는 다음에 나오는 식의 결과를 추가 변수로 출력하는 함수임
- pig.region.daily.mean 을 구성하는 변수에 대한 설명은 다음과 같음
  - name : 지역이름
  - region : 지역코드
  - date : 일자
  - mean.price : 전체지역의 일별 평균 가격

date 에서 month 만 추출하여 지역(region), 월(month)별로 돼지고기의 평균가격을 구하여 pig.region.monthly.mean 데이터를 생성한다.

```
pig.region.monthly.mean <- ddply(pig.region.daily.mean,
    .(name, region, month=str_sub(pig.region.daily.mean$date,1,7)),
    summarise, mean.price=mean(mean.price))
head(pig.region.monthly.mean, n=10)
```

```
##    name region   month mean.price
## 1  광주   2401 2011-01   1782.619
## 2  광주   2401 2011-02   2073.235
## 3  광주   2401 2011-03   1771.364
## 4  광주   2401 2011-04   1785.000
## 5  광주   2401 2011-05   1956.750
## 6  광주   2401 2011-06   2357.857
## 7  광주   2401 2011-07   2452.976
## 8  광주   2401 2011-08   2253.182
## 9  광주   2401 2011-09   1997.375
## 10 광주   2401 2011-10   1731.375
```

- str_sub() : 문자열에서 뽑아내고자 하는 값만 출력함
  - str_sub(date,1,7) : date 데이터 내에서 각 개체의 1~7 번째 값(연~월)을 추출함
- pig.region.daily.mean 을 구성하는 변수에 대한 설명은 다음과 같음
  - name : 지역이름
  - region : 지역코드
  - month : 월
  - mean.price : 전체지역의 월별 평균 가격

date 에서 year 만 추출하여 지역(region), 연(year)별로 돼지고기의 평균가격을 구하여 pig.region.yearly.mean 데이터를 생성한다.

```
pig.region.yearly.mean <- ddply(pig.region.daily.mean,
    .(name, region, year=str_sub(pig.region.daily.mean$date,1,4)),
    summarise, mean.price=mean(mean.price))
head(pig.region.yearly.mean, n=10)
```

```
##    name region year mean.price
## 1  광주   2401 2011   1990.312
## 2  광주   2401 2012   1658.922
## 3  광주   2401 2013   1573.522
## 4  대구   2200 2011   2031.474
## 5  대구   2200 2012   1714.234
## 6  대구   2200 2013   1533.105
## 7  대전   2501 2011   1969.667
## 8  대전   2501 2012   1691.904
## 9  대전   2501 2013   1527.339
## 10 부산   2100 2011   2185.958
```

- str_sub() : 문자열에서 뽑아내고자 하는 값만 출력함
  - str_sub(date,1,4) : date 데이터 내에서 각 개체의 1~4 번째 값(연도)을 추출함

4. 상관분석 및 데이터 시각화

4.1 월별 돼지고기 가격 시각화

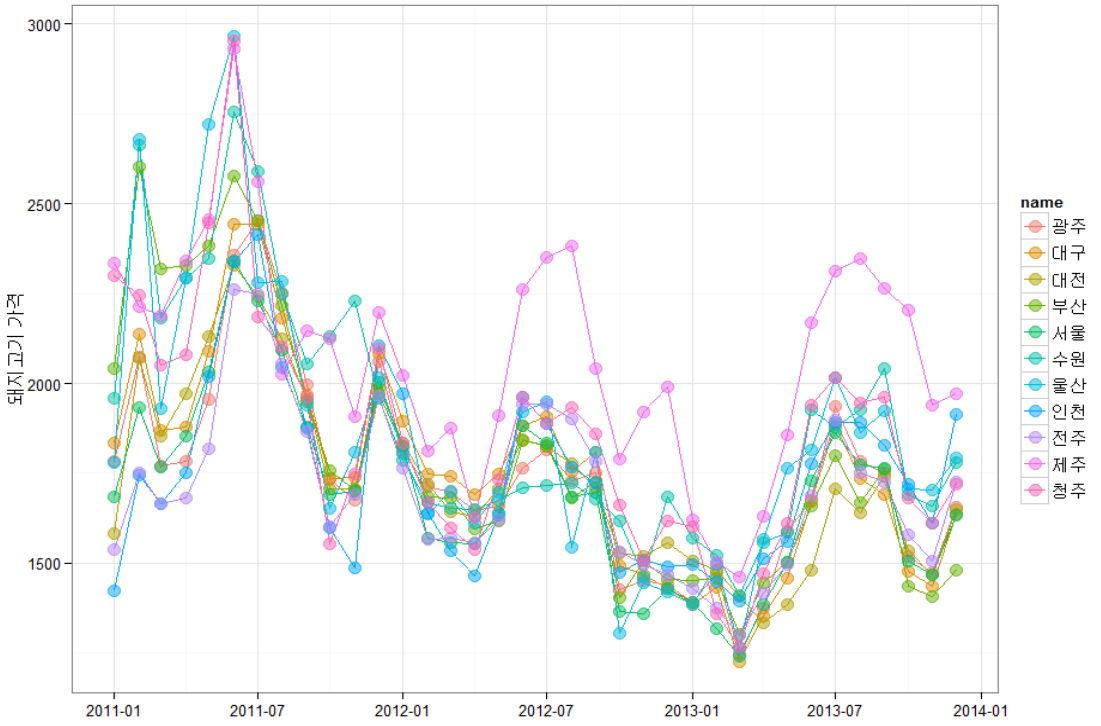
시계열도를 그리기 전에 시각화를 위해 월별 데이터를 가공한다.

```
pig.region.monthly.mean$month <- as.Date(as.yearmon(pig.region.monthly.mean$month, "%Y-%m"))
```

- `as.yearmon()` : **factor** 타입의 데이터를 월별 시계열로 변환함
  - `as.Date()` : 변환된 시계열 데이터를 **date** 타입으로 변환함

2011 년부터 2013 년까지 월별 돼지고기 가격의 변화를 지역별로 시계열 그림을 통해 시각화한다.

```
ggplot(pig.region.monthly.mean, aes(x=month, y=mean.price, colour=name, group=name)) +
    geom_line() + theme_bw() + geom_point(size=6, shape=20, alpha=0.5) +

    ylab("돼지고기 가격") + xlab("")
```



- `ggplot()` : 메인 함수로 데이터 셋과 표현할 데이터 변수명을 정의함

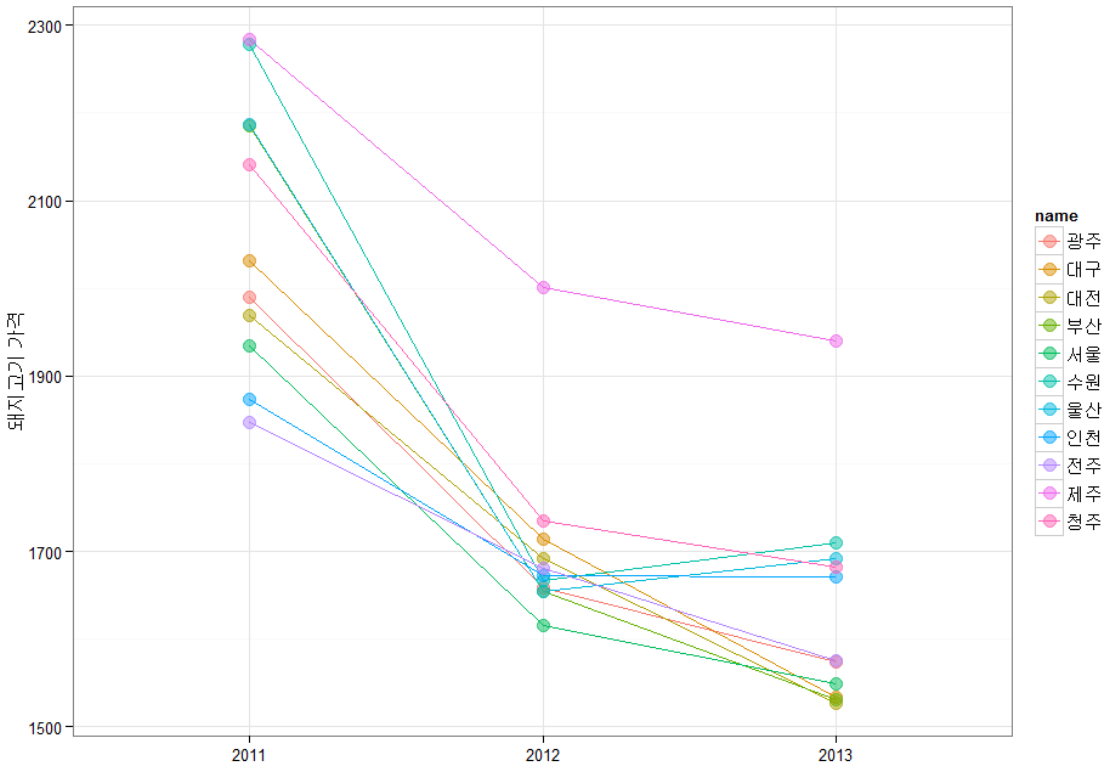
- aes(x=month, y=mean.price, colour=name, group=name))
  : 월별 가격데이터를 지역별로 표현함

- geom_line() : 데이터를 line 형태로 시각화함
- geom_point() : 데이터를 point 형태로 시각화함
- theme_bw() : 흰색 배경에 검은 색 눈금 선에 테마를 적용함
- xlab(), ylab() : x 축 또는 y 축의 이름을 지정함

4.2 지역별 연간 돼지고기 평균가격 시각화

2011 년부터 2013 년까지 지역별 돼지고기 연평균 가격의 변화를 지역별로 시각화한다.

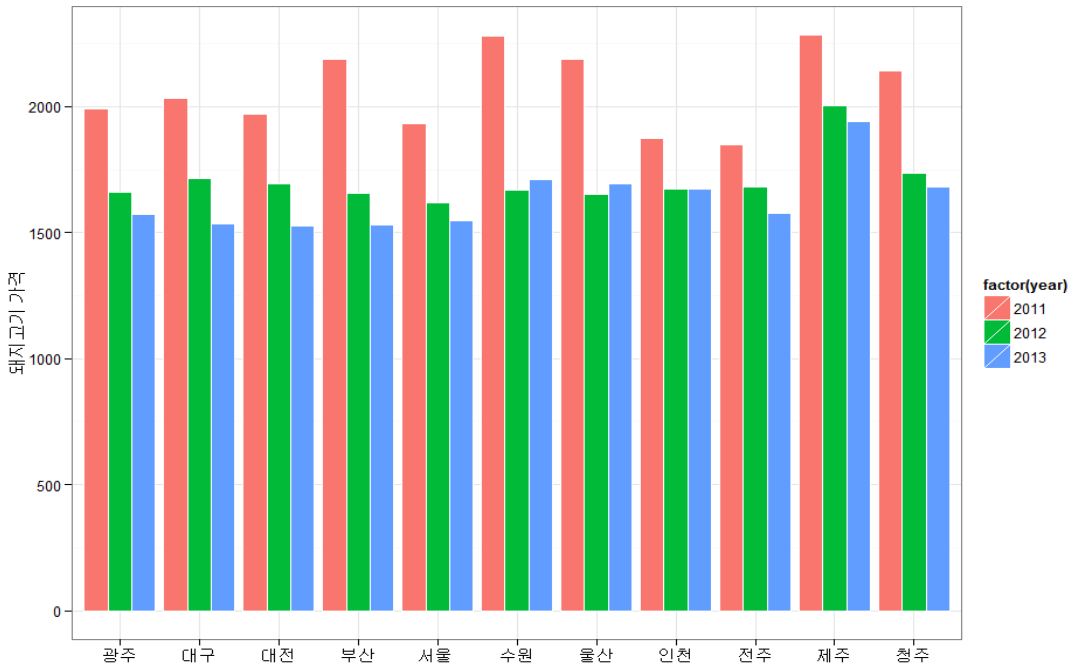
```
ggplot(pig.region.yearly.mean, aes(x=year, y=mean.price, colour=name, gr
oup=name)) +
    geom_line() + theme_bw() + geom_point(size=6, shape=20, alpha=0.5) +

    ylab("돼지고기 가격") + xlab("")
```



- ggplot() : 메인 함수로 데이터 셋과 표현할 데이터 변수명을 정의함
  - aes(x=year, y=mean.price, colour=name, group=name))
    : 연별 가격데이터를 품목별로 표현함
- geom_line() : 데이터를 line 형태로 시각화함
- geom_point() : 데이터를 point 형태로 시각화함
- theme_bw() : 흰색 배경에 검은 색 눈금 선에 테마를 적용함
- xlab(), ylab() : x 축 또는 y 축의 이름을 지정함

2011 년부터 2013 년까지 지역별 돼지고기 연평균 가격의 변화를 막대그래프를 이용하여 시각화한다.

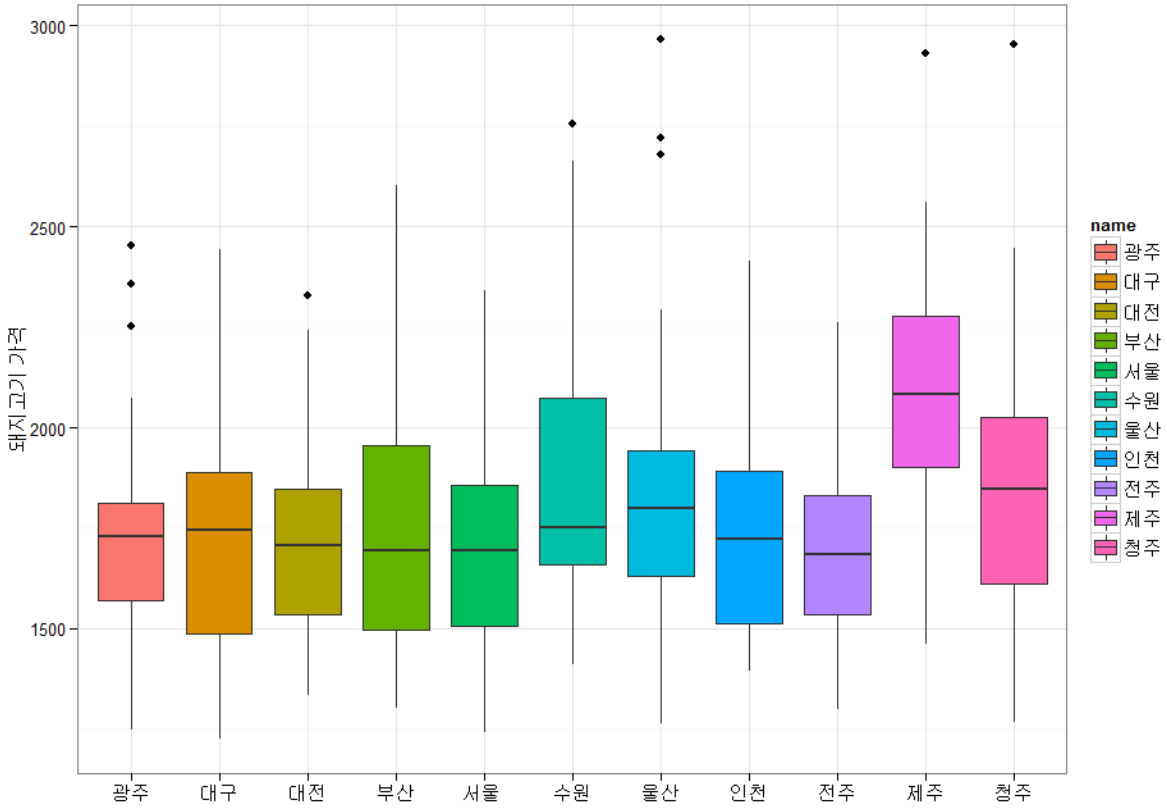
```
ggplot(pig.region.yearly.mean, aes(x=name, y=mean.price, fill=factor(year))) +
    theme_bw() + geom_bar(stat="identity", position="dodge", colour="white") +
    ylab("돼지고기 가격") + xlab("")
```



- ggplot() : 메인 함수로 데이터 셋과 표현할 데이터 변수명을 정의함
- geom_bar() : 막대그래프로 시각화하는 함수임
  - stat="identity" : 데이터 내에 y 축에 해당하는 값이 포함되어있을 때 사용. 즉, 이 명령어가 없으면 히스토그램 형태로 출력됨.
  - position="dodge" : 층이 겹치는 지점에 대해 구분하여 표시. 즉, 이 명령어가 없으면 각 지역별로 막대그래프 하나에 층으로 연도가 구분됨.
  - colour="white" : 배경화면에 대한 설정. white 는 하얀 바탕에 검은 테두리를 사용함
- xlab(), ylab() : x 축 또는 y 축의 이름을 지정함

2011 년부터 2013 년까지 지역별 돼지고기 일 평균 가격의 분포를 상자그림을 이용하여 시각화한다.

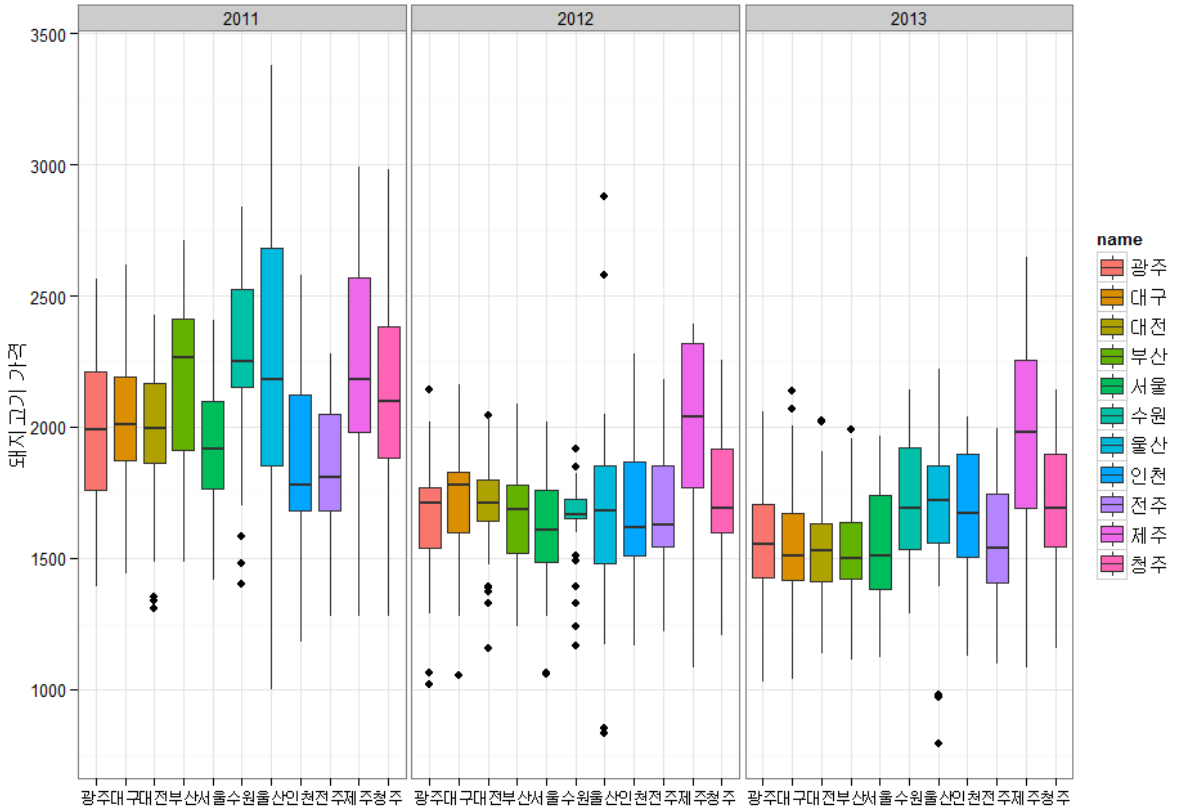
```
ggplot(pig.region.monthly.mean, aes(x=name, y=mean.price, fill=name)) +
    theme_bw() + geom_boxplot() +
    xlab("") + ylab("돼지고기 가격")
```



- `ggplot()` : 메인 함수로 데이터 셋과 표현할 데이터 변수명을 정의함
- `geom_boxplot()` : 상자그림으로 시각화하는 함수임
- `theme_bw()` : 흰색 배경에 검은 색 눈금 선에 테마를 적용함
- `xlab(), ylab()` : x 축 또는 y 축의 이름을 지정함

2011 년부터 2013 년까지 지역별 돼지고기 일 평균 가격의 연도별 분포를 상자그림을 이용하여 시각화한다.

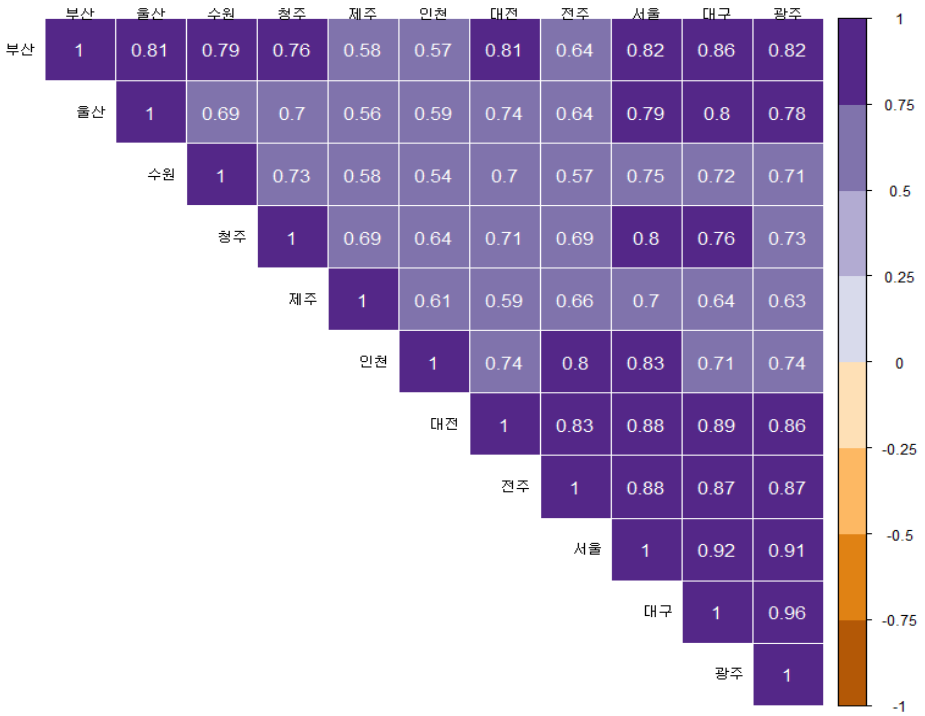
```
ggplot(year.pig , aes(x=name, y=mean.price, fill=name)) + geom_boxplot() +
    theme_bw() + facet_wrap(~year, scales='fixed') +
    xlab("") + ylab("돼지고기 가격") + theme(axis.text.x=element_text(size=
9))
```



- `geom_boxplot()` : 상자그림으로 시각화하는 함수임
- `facet_wrap()` : 격자 진열로 도식화하는 함수임
  - `~year` : 연도별로 나누어 도식화
  - `scales='fixed'` : 데이터 분포와 상관없이 고정된 y 축을 사용
- `theme(axis.text.x=element_text(size=9))` : x 축 값의 글씨 크기를 9 로 조절하는 함수임

한 번에 가격변화를 비교하기 어려우므로 도시들 간의 상관관계를 분석하여 상관관계가 높은 도시들을 묶어 가격변화를 살펴보도록 하겠다.

```
temp <- dlply(pig.region.daily.mean, .(name), summarise, mean.price)
pig.region <- data.frame(서울=unlist(temp$서울),
    부산=unlist(temp$부산),
    대구=unlist(temp$대구),
    인천=unlist(temp$인천),
    광주=unlist(temp$광주),
    대전=unlist(temp$대전),
    울산=unlist(temp$울산),
    수원=unlist(temp$수원),
    청주=unlist(temp$청주),
    전주=unlist(temp$전주),
    제주=unlist(temp$제주))
cor_pig <- cor(pig.region)
corrplot(cor_pig, method="color", type="upper", order="hclust", addCoef.
col = "white", tl.srt=0, tl.col="black", tl.cex=0.7, col=brewer.pal(n=8,
 name="PuOr"))
```



- `cor()` : 행렬의 분산공분산을 계산하는 함수임
- `corrplot()` : 상관행렬을 도식화하는 함수임
  - `method` : 상관행렬을 도식화하는데 사용되는 특징. circle(원), square(사각), ellipse(타원), ...
  - `type` : 상관행렬을 표시하는 특징. full(전체), upper(상삼각), lower(하삼각)

- `order` : 상관행렬의 정렬방식을 나타내는 특징. original(데이터 순), hclust(계층 군집분석 순) , …
- `addCoef.col` : 상관계수가 표시되는 색
- `tl.srt` : 텍스트 문자열의 표시 각도
- `tl.col` : 텍스트 문자열의 색
- `tl.cex` : 텍스트 문자열 크기
- `col=brewer.pal(n=8, name="PuOr")` : 그래프 색 지정. `brewer.pal` 을 사용하여 팔레트를 탐색함. PuOr 이름의 팔레트에서 8 가지 색을 추출하는 기능을 함

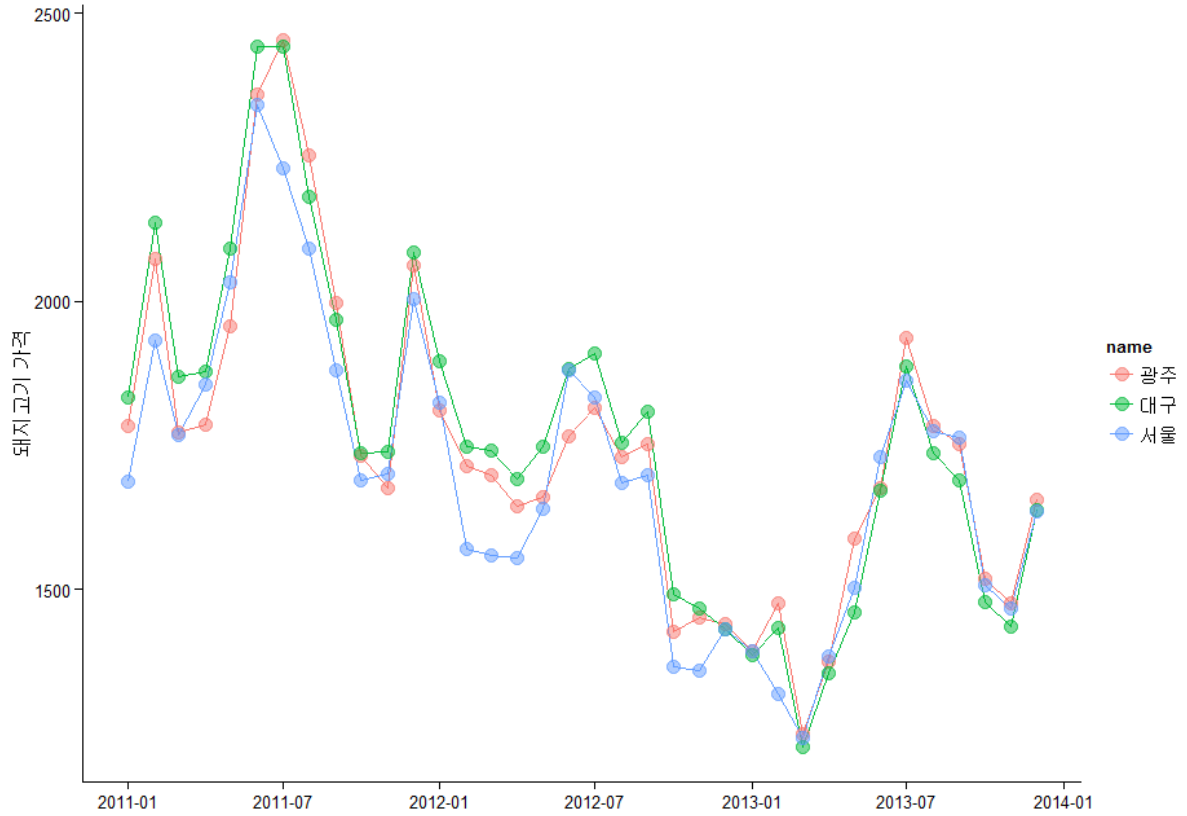
시계열도를 그리기 전에 월별 데이터를 가공한다.

```
pig.region.monthly.mean$month <- as.Date(as.yearmon(pig.region.monthly.mean$month, "%Y-%m"))
```

- `as.yearmon()` : **factor** 타입의 데이터를 월별 시계열로 변환함
  - `as.Date()` : 변환된 시계열 데이터를 **date** 타입으로 변환함

광주, 대구, 서울 세 지역의 2011 년~2013 년 월별 돼지고기 가격 시계열 그림을 통해 가격 변화가 매우 유사함을 확인할 수 있다.

```
ggplot(pig.region.monthly.mean[pig.region.monthly.mean$region %in% c(240
1,2200,1101),],
    aes(x=month, y=mean.price, colour=name, group=name)) +
    geom_line() + theme_classic() + geom_point(size=6, shape=20, alpha=
0.5) +
    ylab("돼지고기 가격") + xlab("")
```

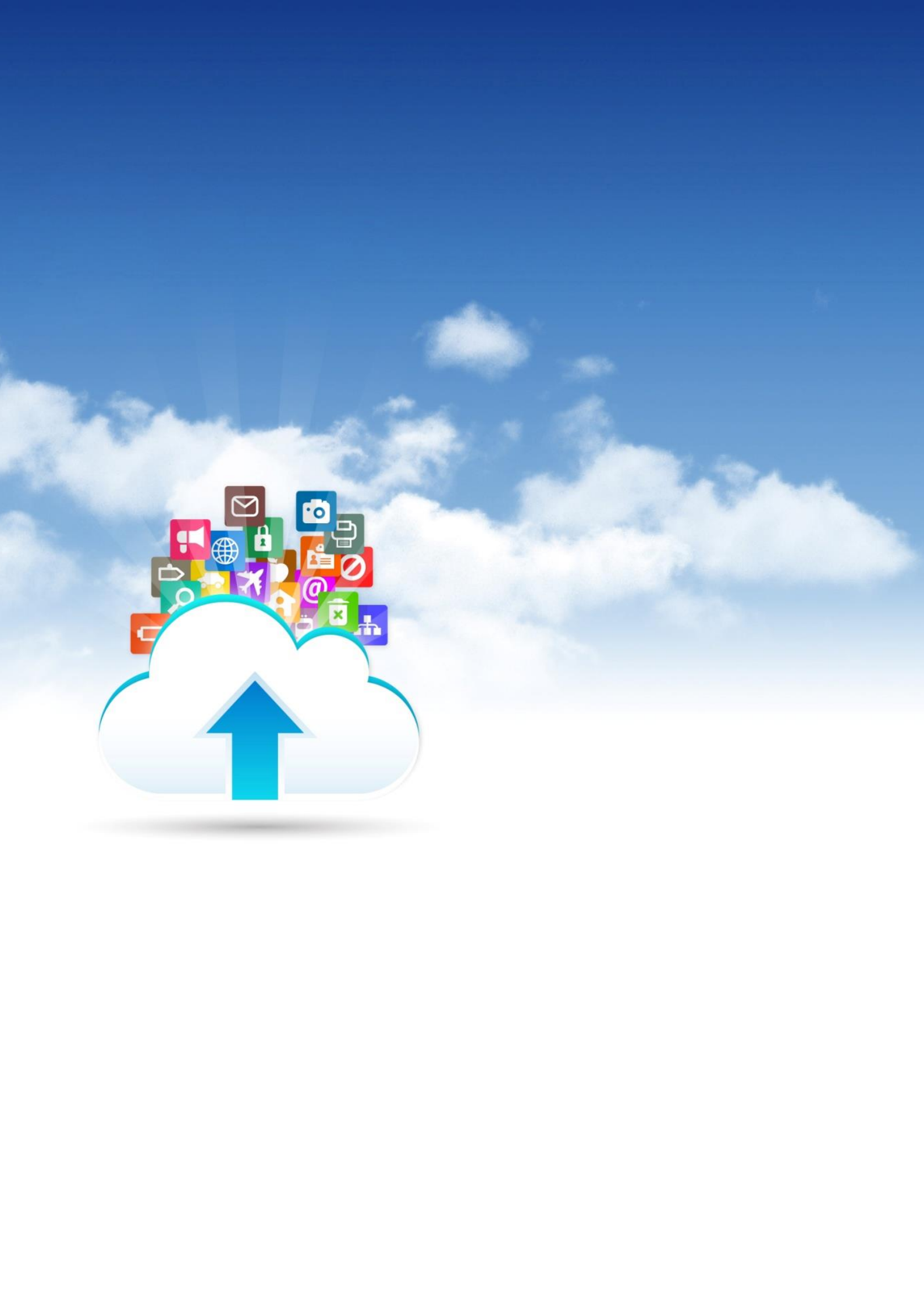


- `ggplot()` : 메인 함수로 데이터 셋과 표현할 데이터 변수명을 정의함
  - `pig.region.monthly.mean[pig.region.monthly.mean$region %in% c(2401,2200,1101),]` : 지역코드가 광주, 대구와 서울인 데이터만 추출함
  - `%in%` : 조건에 대해 일치여부를 boolean 형으로 출력하는 기능을 하는 연산자임
  - `aes(x=month, y=mean.price, colour=name, group=name))` : 월별 가격데이터를 품목별로 표현함

5. 데이터 저장하기

추후 분석에 필요한 데이터를 csv 파일로 저장한다.

```
write.csv(pig.region, "Data/pig.region.csv", fileEncoding="UTF-8")
write.csv(pig.region.monthly.mean, "Data/pig.region.monthly.mean.csv", f
ileEncoding="UTF-8")
```



Ⅲ. 공적분 검정을 통한 농축산물 소매가격 연관성

1. 필요 패키지 불러오기
2. 데이터 불러오기
3. 데이터 가공하기
4. 공적분 검정
5. 데이터 시각화
6. 데이터 저장

Ⅲ 공적분 검정을 통한 농축산물 소매가격 연관성 분석

한국농수산식품유통공사에서 제공받은 2011 년~2013 년 농축산물의 소매가격 정보 중 일별 축산물의 소매가격과 일별 농산물의 소매가격 자료를 이용하여 공적분 검정 시행 후 공적분이 있는 품목들을 시각화하고 의미를 찾는다.

1. 필요 패키지 불러오기

아래 R 명령어는 여러 개의 패키지를 로딩하는데 유용한 수행방법이다.

```
library2 <- c("plyr", "stringr", "urca", "ggplot2", "zoo", "gridExtra")
unlist(lapply(library2, require, character.only = TRUE ))
```

➢ 각 패키지의 역할을 간략히 설명하면 다음과 같음

패키지 명	설명
plyr	데이터 핸들링을 하기 위한 라이브러리
stringr	문자열 핸들링을 하기 위한 라이브러리
urca	공적분 검정을 하기 위한 라이브러리
ggplot2	시각화 기능 라이브러리
zoo	문자형 데이터를 데이트 형식으로 변환하기 위한 라이브러리
gridExtra	그래프 배열을 위한 라이브러리

2. 데이터 불러오기

CSV 형태의 농축산물 데이터와 코드설명 데이터는 다음과 같이 읽어 들인다.

```
product <- read.csv("Data/product.csv", header=T, fileEncoding="UTF-8")
code    <- read.csv("Data/code.csv", header=T, fileEncoding="UTF-8")
```

➢ read.csv() : 이 함수는 csv 파일을 읽는 기능을 함

3. 데이터 가공하기

함수 read.csv 에 읽혀 생성된 R 오브젝트 product 를 출력하면 다음과 같다.

```
str(product)
```

```
## 'data.frame':    253418 obs. of  6 variables:
##  $ 일자    : Factor w/ 745 levels "2011-01-03","2011-01-04",..: 1 1 1
 1 1 1 1 1 1 1 ...
##  $ 부류코드: int  100 100 100 100 100 100 100 100 100 100 ...
##  $ 품목코드: int  111 111 111 111 111 111 111 111 111 111 ...
##  $ 지역코드: int  3511 3311 1101 1101 1101 1101 1101 1101 2100 2100
 ...
##  $ 마트코드: int  350401 330401 110251 110401 110402 110403 110405 110
406 210022 210401 ...
##  $ 가격    : int  39600 40000 37000 43900 39800 39600 42000 43800 420
00 36800 ...
```

➢ `str(product)` : 변수가 6 개, 자료의 수가 253418 인 **data.frame** 형태임

함수 read.csv 에 읽혀 생성된 R 오브젝트 product 의 앞부분을 출력하면 다음과 같다.

```
head(product, n=10)
```

```
##           일자 부류코드 품목코드 지역코드 마트코드  가격
## 1  2011-01-03      100      111     3511   350401 39600
## 2  2011-01-03      100      111     3311   330401 40000
## 3  2011-01-03      100      111     1101   110251 37000
## 4  2011-01-03      100      111     1101   110401 43900
## 5  2011-01-03      100      111     1101   110402 39800
## 6  2011-01-03      100      111     1101   110403 39600
## 7  2011-01-03      100      111     1101   110405 42000
## 8  2011-01-03      100      111     1101   110406 43800
## 9  2011-01-03      100      111     2100   210022 42000
## 10 2011-01-03      100      111     2100   210401 36800
```

➢ `head(product, n=10)` : product 데이터의 앞부분을 출력함

R 서버에서 원활하게 데이터에 접근하기 위해 원시 데이터의 변수명을 한글에서 영어로 변환해준다.
데이터 product 의 변수명은 date(일자), category(부류코드), item(품목코드), region(지역코드), mart(마트코드), price(가격)로 변환된다.

```
colnames(product) <- c('date','category','item','region','mart','price')
```

➢ `colnames()` : 데이터 프레임의 열의 이름을 지정함

R 함수를 이용하여 product 를 품목(item), 일자(date)별로 평균가격을 구하여 데이터를 생성하기 위해서는 다음과 같은 명령어를 수행한다.

```
temp <- ddply(product, .(item, date), summarise, mean.price=mean(price))
head(temp, n=10)
```

```
##    item       date mean.price
## 1   111 2011-01-03   40953.33
## 2   111 2011-01-04   40993.33
## 3   111 2011-01-05   41153.33
## 4   111 2011-01-06   41020.00
## 5   111 2011-01-07   41013.33
## 6   111 2011-01-10   41180.00
## 7   111 2011-01-11   41080.00
## 8   111 2011-01-12   41146.67
## 9   111 2011-01-13   41146.67
## 10  111 2011-01-14   41146.67
```

- `ddply()` : 데이터 프레임 형태를 주어진 조건에 따라 정렬할 후 **summarise** 한 값을 **data frame** 형태로 출력함
  - `mean()` : 품목별로 정렬된 일별 전체 지역의 가격에 대한 평균을 구함
  - `summarise` : 출력하고자 하는 값만을 요약함

일별 농축산물의 평균 소매가격을 추출하기 위해 품목별 코드 번호를 확인하기 위한 category 오브젝트를 생성한다.
여기서 category 에는 code 데이터에서 구분코드설명이 품목코드에 해당하는 값만 추출되어 저장된다.

```
category <- subset(code, code$구분코드설명=="품목코드")
category
```

```
##    구분코드 구분코드설명 분류코드 분류코드설명
## 6      1200     품목코드      111          쌀
## 7      1200     품목코드      211         배추
## 8      1200     품목코드      214         상추
## 9      1200     품목코드      224         호박
## 10     1200     품목코드      245         양파
## 11     1200     품목코드      256     파프리카
## 12     1200     품목코드      312         참깨
## 13     1200     품목코드      411         사과
## 14     1200     품목코드      514     돼지고기
## 15     1200     품목코드      515       닭고기
```

- subset() : 데이터의 특정부분에 대해 조건을 만족하는 값을 반환해주는 함수임

R 서버에서 원활하게 데이터에 접근하기 위해 원시 데이터의 변수명을 한글에서 영어로 변환해준다.
데이터 category 의 변수명은 code(구분코드), exp(구분코드설명), item(분류코드), name(분류코드설명)으로 변환된다.

```
colnames(category) <- c('code', 'exp', 'item', 'name')
```

- colnames() : 데이터 프레임의 열의 이름을 지정함

merge() 함수를 이용하여 temp 데이터와 category 데이터를 공통변수 item 으로 합쳐 date.item.mean 테이터를 생성한다.

```
date.item.mean <- merge(temp, category, by="item")
head(date.item.mean, n=10)
```

```
##    item       date mean.price code      exp name
## 1   111 2011-01-03   40953.33 1200 품목코드   쌀
## 2   111 2011-01-04   40993.33 1200 품목코드   쌀
## 3   111 2011-01-05   41153.33 1200 품목코드   쌀
## 4   111 2011-01-06   41020.00 1200 품목코드   쌀
## 5   111 2011-01-07   41013.33 1200 품목코드   쌀
## 6   111 2011-01-10   41180.00 1200 품목코드   쌀
## 7   111 2011-01-11   41080.00 1200 품목코드   쌀
## 8   111 2011-01-12   41146.67 1200 품목코드   쌀
## 9   111 2011-01-13   41146.67 1200 품목코드   쌀
## 10  111 2011-01-14   41146.67 1200 품목코드   쌀
```

- head() : 데이터의 앞부분을 출력함

월별 데이터를 필요한 변수들만 구성하여 생성한다.

```
month.item.mean <- ddply(date.item.mean, .(name, item, month=str_sub(as.
character.Date(date),1,7)),
  summarise, mean.price=mean(mean.price))
head(month.item.mean, n=10)
```

```
##    name   item   month   mean.price
## 1  닭고기  515  2011-01   5517.809
## 2  닭고기  515  2011-02   6612.114
## 3  닭고기  515  2011-03   6983.201
## 4  닭고기  515  2011-04   6911.101
## 5  닭고기  515  2011-05   6004.349
## 6  닭고기  515  2011-06   5242.317
## 7  닭고기  515  2011-07   5971.546
## 8  닭고기  515  2011-08   6314.089
## 9  닭고기  515  2011-09   5890.317
## 10 닭고기  515  2011-10   5751.018
```

- str_sub() : 문자열에서 뽑아내고자 하는 값만 출력함
  - str_sub(date,1,7) : date 데이터 내에서 각 개체의 1~7 번째 값(연~월)을 추출함
- month.item.mean 을 구성하는 변수에 대한 설명은 다음과 같음
  - name : 품목이름
  - item : 품목코드
  - date : 일자
  - month : 월
  - mean.price : 전체지역의 월별 평균 가격

공적분 검정을 위한 데이터를 만들기 위해 다음과 같은 명령어를 실행한다.
다음은 일간 품목별 평균 데이터이다.

```
temp <- dlply(date.item.mean, .(name), summarise, mean.price)
daily.product <- data.frame(쌀=unlist(temp$쌀),
    배추=unlist(temp$배추),
    상추=unlist(temp$상추),
    호박=unlist(temp$호박),
    양파=unlist(temp$양파),
    파프리카=unlist(temp$파프리카),
    참깨=unlist(temp$참깨),
    사과=unlist(temp$사과),
    돼지고기=unlist(temp$돼지고기),
    닭고기=unlist(temp$닭고기))
head(daily.product, n=10)
```

```
##             쌀     배추     상추     호박     양파  파프리카     참깨
## ..11  40953.33 4251.818 642.9024 1123.889 1903.333 1015.2963 12133.33
## ..12  40993.33 4168.182 633.3095 1134.259 1877.667 1015.1071 12133.33
## ..13  41153.33 4375.833 629.5227 1145.370 1864.333 1015.1071 12133.33
## ..14  41020.00 4609.167 651.7826 1231.321 2070.000 1043.1071 12133.33
## ..15  41013.33 4609.167 653.9565 1227.925 2071.667 1043.1071 12133.33
## ..16  41180.00 4399.167 648.2391 1275.849 2075.000 1022.7500 12133.33
## ..17  41080.00 4422.500 642.3696 1277.222 2075.000 1025.2857 12133.33
## ..18  41146.67 4495.000 627.4783 1303.774 2067.667 1017.8621 12133.33
## ..19  41146.67 4521.111 635.1522 1350.185 1856.333 1006.8276 12040.00
## ..110 41146.67 4521.111 638.8478 1365.472 1843.000  997.5172 12040.00
##           사과 돼지고기   닭고기
## ..11  24904.00 1653.000 5262.524
## ..12  24904.00 1683.000 5228.773
## ..13  25176.73 1687.000 5263.318
## ..14  25358.55 1504.333 5289.696
## ..15  25994.91 1516.333 5130.136
## ..16  26176.73 1530.333 5201.000
## ..17  27027.64 1512.667 5306.045
## ..18  27482.18 1512.667 5306.045
## ..19  27320.09 1752.000 5656.955
## ..110 27320.09 1748.000 5647.087
```

- `dlply()` : 데이터 프레임 형태를 품목별로 **list** 형태로 출력함
  - `unlist()` : **list** 형태를 **vector** 형태로 변환함
  - `data.frame` : **vector** 형태의 데이터를 **data frame** 형태로 변환함

다음은 위와 같은 방법으로 월간 품목별 평균 데이터를 생성한다.

```
temp <- dlply(month.item.mean, .(name), summarise, mean.price)
monthly.product <- data.frame(쌀=unlist(temp$쌀),
```

```
    배추=unlist(temp$배추),
    상추=unlist(temp$상추),
    호박=unlist(temp$호박),
    양파=unlist(temp$양파),
    파프리카=unlist(temp$파프리카),
    참깨=unlist(temp$참깨),
    사과=unlist(temp$사과),
    돼지고기=unlist(temp$돼지고기),
    닭고기=unlist(temp$닭고기))
head(monthly.product, n=10)
```

```
##             쌀     배추      상추     호박     양파  파프리카     참깨
## ..11  41286.35 4728.411  640.2479 1444.737 2000.267 1025.0198
12075.56
## ..12  41957.45 4804.518  641.9159 1544.894 2254.595 1079.0852
11934.12
## ..13  42763.86 4683.778  616.6619 1461.145 2180.592 1008.6646
11814.55
## ..14  44331.37 3301.397  589.1966 1090.387 1635.318  988.8949
11812.00
## ..15  44702.17 1430.533  666.4199 1060.357 1313.533  885.4103
11812.00
## ..16  45076.56 1216.767  631.5808 1196.714 1190.547  849.5205
11812.00
## ..17  44678.94 2268.429 1298.1798 1709.303 1383.319  891.4756
11812.00
## ..18  44059.64 4051.221 1470.3333 2125.239 1457.447  984.2294
11825.64
## ..19  44021.13 3300.982 1048.3100 1702.826 1519.013 1363.8384
11792.00
## ..110 43514.47 2264.531  762.5600 1324.929 1402.435 1341.4683
11712.00
##           사과 돼지고기   닭고기
## ..11  27870.33 1780.313 5517.809
## ..12  28694.18 2131.667 6612.114
## ..13  27635.08 1899.864 6983.201
## ..14  26181.89 1958.794 6911.101
## ..15  26041.14 2138.150 6004.349
## ..16  26796.51 2459.933 5242.317
## ..17  27567.90 2359.587 5971.546
## ..18  23466.68 2151.652 6314.089
## ..19  23415.32 1944.283 5890.317
## ..110 19180.17 1737.617 5751.018
```

4. 공적분 검정

돼지고기와 다른 품목들 간의 공적분 관계가 있는지 검정을 한다.

공적분의 개념을 직관적으로 이해하기 위하여, 먼저 간단한 두 시계열 변수의 경우를 생각하자. 두 시계열 X_{t1}과 X_{t2}에 대하여 적당한 상수 a, b가 존재하여 관계식 $aX_{t1} + bX_{t2} \sim 0$가 성립하면 두 시계열이 공적분관계가 있음을 의미한다.

이 때 두 상수 a, b의 부호가 같은 경우 (음의 공적분)에는 두 시계열은 서로 다른 방향으로 움직임을 의미하며 부호가 다른 경우 (양의 공적분)에는 서로 같은 방향으로 움직임을 의미한다.

공적분 분석에 대한 자세한 내용은 금융계량분석(조담 저, 2006, 도서출판 청람)을 참고하기 바란다.

```
for (i in 1:9){
  for (j in 1:9){
    if ((i+j) < 11){
      jc <- ca.jo(data.frame(daily.product[,i], daily.product[,i+j]), ty
pe="trace", K=2, ecdet="const")
      if (jc@teststat[1] > jc@cval[1]) {
        if(jc@V[1,1]*jc@V[2,1]>0){
          cat( colnames(monthly.product)[i],"와" , colnames(monthly.prod
uct)[i+j], ": 음의 공적분 관계가 있다.", "\n")
        } else {
          cat( colnames(monthly.product)[i],"와" , colnames(monthly.prod
uct)[i+j], ": 양의 공적분 관계가 있다.","\n")
        }}}}}

## 상추 와 호박 : 양의 공적분 관계가 있다.
## 상추 와 사과 : 음의 공적분 관계가 있다.
## 상추 와 돼지고기 : 양의 공적분 관계가 있다.
## 상추 와 닭고기 : 음의 공적분 관계가 있다.
## 호박 와 닭고기 : 음의 공적분 관계가 있다.
## 사과 와 닭고기 : 음의 공적분 관계가 있다.
```

- `ca.jo()` : 두 변수간의 **요한슨 공적분 검정**을 시행함.
  - data.frame() : 공적분 검정을 시행할 두 변수를 데이터 프레임 형태로 변환함

- type="trace" : Test type is trace statistic , without linear trend and constant in cointegration
- K=2 : The lag order of the series (levels) in the VAR.
- ecdet="const" : for constant term in cointegration

공적분 검정을 더 알아보기 위해 아래 상추와 호박의 공적분 분석 결과를 살펴보자.

```
output <- ca.jo(data.frame(daily.product[,3], daily.product[,4]),
    type="trace", K=2, ecdet="const")
summary(output)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , without linear trend and constant in coi
ntegration
##
## Eigenvalues (lambda):
## [1] 1.576451e-02 1.090154e-02 3.469447e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 |  8.14  7.52  9.24 12.97
## r = 0  | 19.95 17.85 19.96 24.60
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                        daily.product...3..l2 daily.product...4..l2
## daily.product...3..l2              1.00000           1.00000000
## daily.product...4..l2             -1.05189           0.06474024
## constant                         673.25720        -980.34611931
##                             constant
## daily.product...3..l2        1.00000
## daily.product...4..l2      -25.44892
## constant                -278264.55826
##
## Weights W:
## (This is the loading matrix)
##
##                       daily.product...3..l2 daily.product...4..l2
## daily.product...3..d          -0.003869091          -0.01182244
## daily.product...4..d           0.019079144          -0.01172547
##                             constant
## daily.product...3..d 4.097721e-21
## daily.product...4..d 6.898609e-22
```

위 결과에서 필요한 분석 결과를 추출하면 아래와 같다.

```
output@teststat[1]
```

```
## [1] 8.144317
```

```
output@cval[1]
## [1] 7.52
output@V[1,1]
## [1] 1
output@V[2,1]
## [1] -1.05189
```

- `output@teststat` 그리고 `output@cval`:
  - r<=1 이라는 귀무가설 하에서 10% 유의수준으로 teststat(**8.14**) > cval(10pct)(**7.52**)이면 공적분 관계가 있다고 본다.
- `output@V` :
  - [1,1]의 값 1.00000 과 [2,1]의 값 -1.05189 의 곱이 음수이므로 양의 공적분 관계가 있다고 본다.

5. 데이터 시각화

시계열도를 그리기 전에 시각화를 위해 월별 데이터를 가공한다.

```
month.item.mean$month <- as.Date(as.yearmon(month.item.mean$month, "%Y-%
m"))
```

- as.yearmon() : **factor** 타입의 데이터를 월별 시계열로 변환함
  - as.Date() : 변환된 시계열 데이터를 **date** 타입으로 변환함

공적분 관계가 존재하는 변수들의 가격을 시각화 한다.
같은 방향으로 가격이 변화하는 품목들은 빨간색과 주황색으로 나타내고, 반대 방향으로 가격이 변화하는 품목들은 파란색과 하늘색으로 나타낸다.

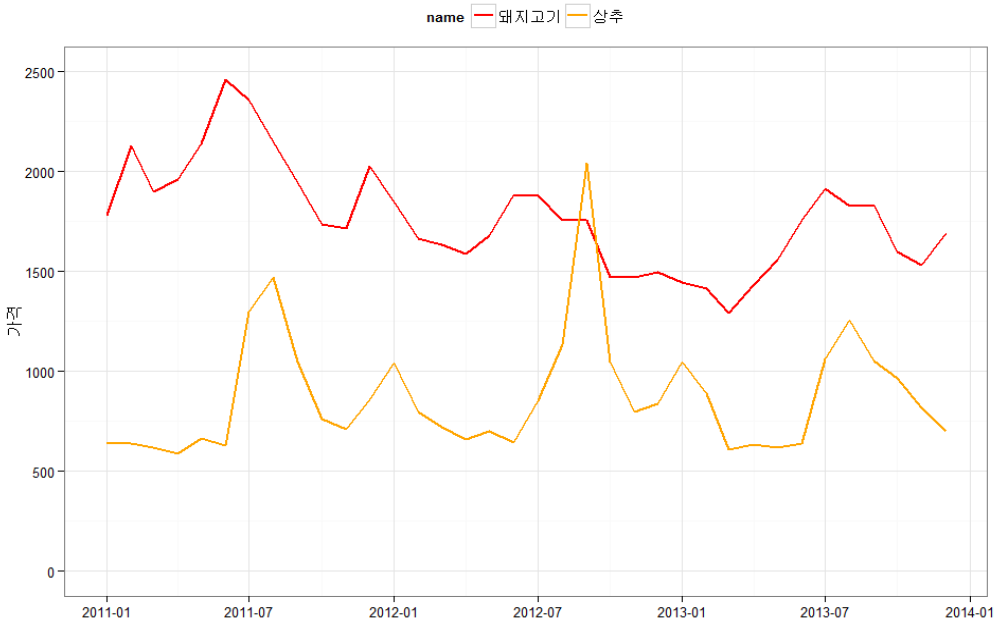
시계열 그림에 의하여 돼지고기는 상추 및 호박과 같은 방향으로 가격이 움직이고 있음을 확인할 수 있으며, 이와 반대로 닭고기는 상추 및 호박과 반대 방향으로 가격이 움직이고 있음을 확인할 수 있다.

이로서 돼지고기와 닭고기는 서로 반대 방향으로 가격이 변화하고 있음을 유추해 볼 수 있다.

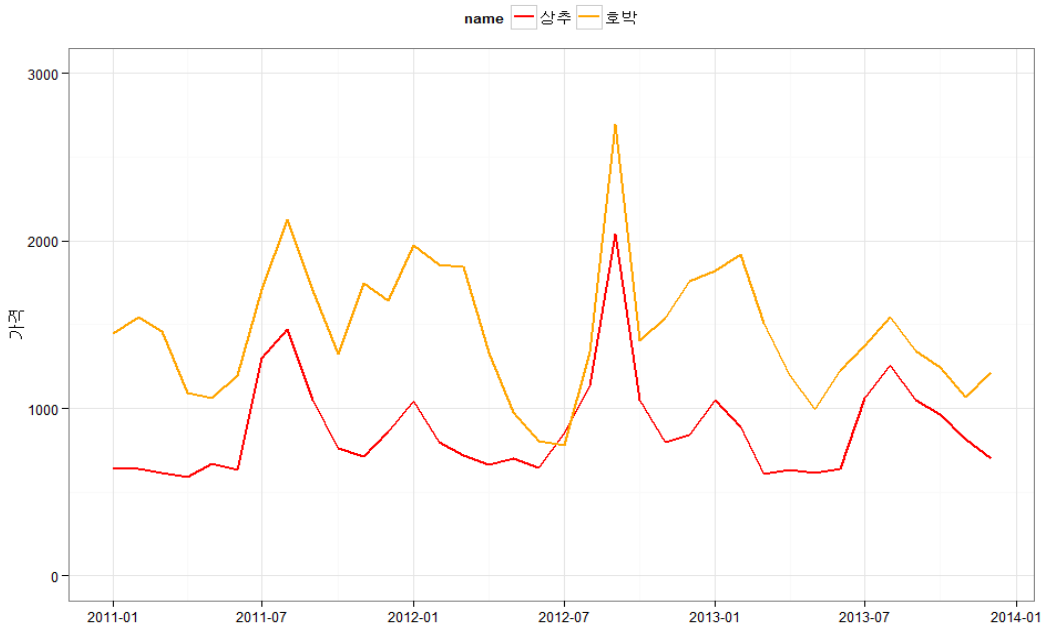
```
p1 <- ggplot(month.item.mean[month.item.mean$name %in% c("돼지고기", "상
추"),], aes(x=month, y=mean.price, colour=name, group=name)) +
  geom_line() + scale_y_continuous(name="가격",limits=c(0,2500)) +
  theme_bw() + xlab("")

p2 <- ggplot(month.item.mean[month.item.mean$name %in% c("상추","호박
"),], aes(x=month, y=mean.price, colour=name, group=name)) +
  geom_line() + scale_y_continuous(name="가격",limits=c(0,3000)) +
  theme_bw() + xlab("")

month.item.mean[month.item.mean$name %in% c("상추"),]$mean.price <- mont
h.item.mean[month.item.mean$name %in% c("상추"),]$mean.price+5000

p4 <- ggplot(month.item.mean[month.item.mean$name %in% c("닭고기","상추
"),], aes(x=month, y=mean.price, colour=name)) +
  geom_line() + scale_y_continuous(name="가격",limits=c(5000,8000)) +
  theme_bw() + xlab("")

month.item.mean[month.item.mean$name %in% c("호박"),]$mean.price <- mont
```

```
h.item.mean[month.item.mean$name %in% c("호박"),]$mean.price+5000

p5 <- ggplot(month.item.mean[month.item.mean$name %in% c("닭고기","호박
"),], aes(x=month, y=mean.price, colour=name)) +
  geom_line() + scale_y_continuous(name="가격",limits=c(5000,8000)) +
  theme_bw() + xlab("")
```

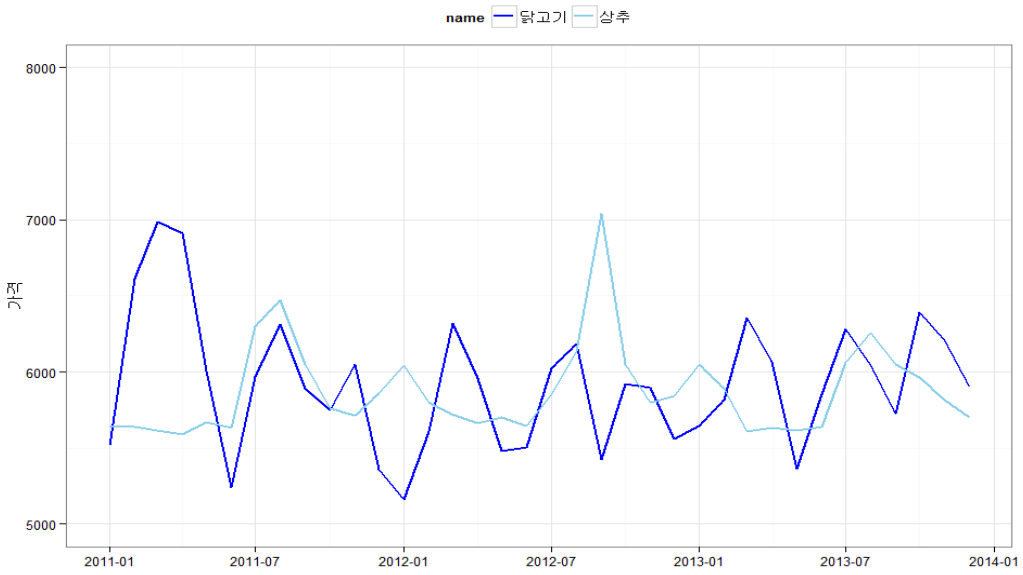
```
p1 + theme(legend.position="top") + scale_color_manual(values=c("red", "
orange")) +
  geom_line(size=1.0)
```



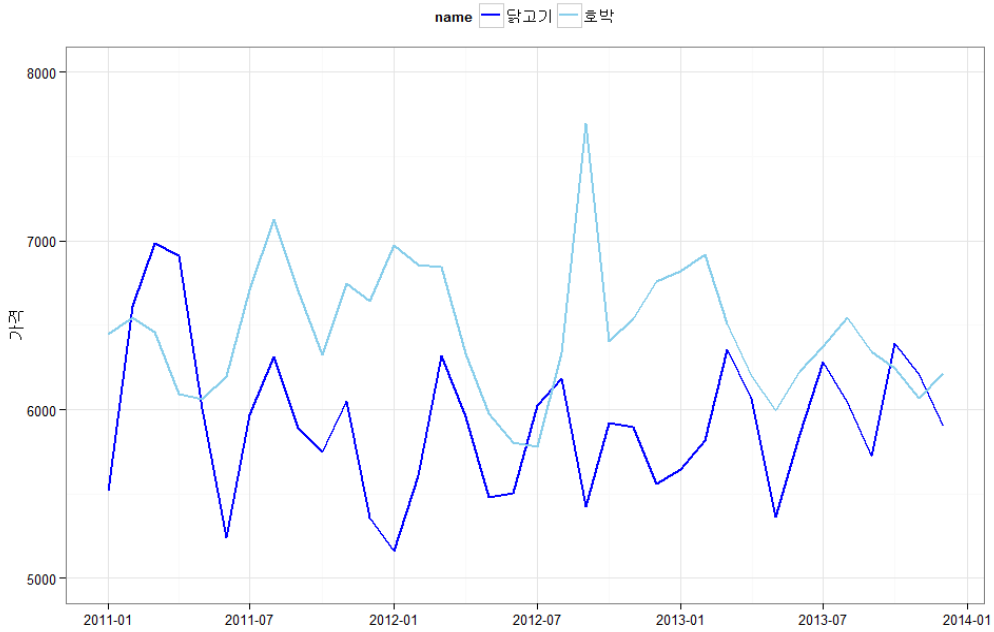
```
p2 + theme(legend.position="top") + scale_color_manual(values=c("red","o
range")) +
  geom_line(size=1.0)
```




```
p4 + theme(legend.position="top") + scale_color_manual(values=c("blue","
skyblue")) +
  geom_line(size=1.0)
```




```
p5 + theme(legend.position="top") + scale_color_manual(values=c("blue","
skyblue")) +
  geom_line(size=1.0)
```



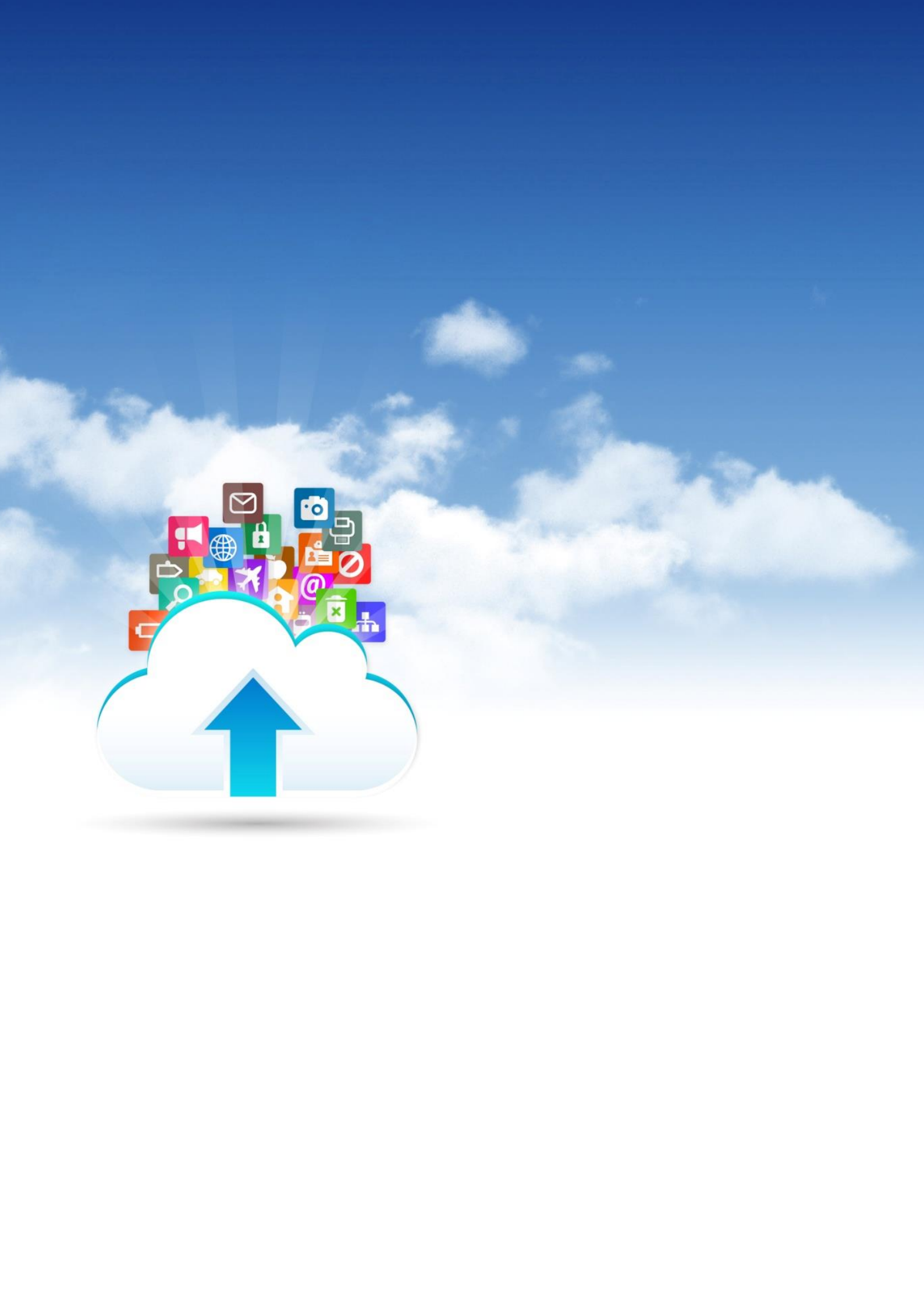
- `theme(legend.position="top")` : 범례의 위치를 상단으로 설정함
- `scale_color_manual()` : 그래프에 대한 색을 설정함
- `geom_line(size=0.7)` : line 의 두께를 설정함
- `grid.arrange()` : ggplot 그래프를 정렬해서 나타냄

6. 데이터 저장하기

추후에 군집분석에 필요한 데이터를 따로 CSV 파일로 저장한다.

```
write.csv(date.item.mean, "Data/date.item.mean.csv", fileEncoding="UTF-8
")
write.csv(month.item.mean, "Data/month.item.mean.csv", fileEncoding="UTF
-8")
```

- `write.csv()` : 이 함수는 R 데이터를 **csv** 파일로 저장하는 기능을 함



Ⅳ. 클러스터링 기법을 이용한 농축산물 데이터 분석

1. 필요 패키지 불러오기
2. 데이터 불러오기
3. 데이터 가공하기
4. 클러스터링 분석 및 데이터 시각화

IV 클러스터링 기법을 이용한 농축산물 데이터 분석

한국농수산식품유통공사에서 제공받은 2011 년~2013 년 농축산물의 소매가격 정보 중 일별 농산물 평균 소매가격 자료와 지역별 돼지고기 평균 소매가격 자료를 이용하여 군집분석 시행 후 유사한 가격변화를 나타내는 그룹을 찾고 이를 시계열 그림으로 시각화 한다.

클러스터링 기법에 대한 자세한 내용은 논문 TSclust: An R Package for Time Series Clustering (Journal of Statistical Software, 2014, Pablo Montero 와 Jose A. Vilar)을 참고하기 바란다.

1. 필요 패키지 불러오기

아래 R 명령어는 여러 개의 패키지를 로딩하는데 유용한 수행방법이다.

```
library3 <- c("plyr", "TSclust", "zoo", "ggplot2" )
unlist(lapply(library3, require, character.only = TRUE ))
```

➢ 각 패키지의 역할을 간략히 설명하면 다음과 같음

패키지 명	설명
plyr	데이터 핸들링을 하기 위한 라이브러리
TSclust	군집 분석을 하기 위한 라이브러리
ggplot2	시각화 기능 라이브러리
zoo	문자형 데이터를 데이트 형식으로 변환하기 위한 라이브러리

2. 데이터 불러오기

Ⅱ장에서 가공한 지역별 돼지고기 일간 평균 소매가격에 대한 데이터 pig.region 와 월간 평균 소매가격에 대한 데이터 pig.region.monthly.mean 을 불러들여온다.

```
pig.region <- read.csv("Data/pig.region.csv", header=T, fileEncoding="UT
F-8")[,-1]
head(pig.region, n=10)
```

```
##         서울   부산   대구 인천 광주 대전 울산 수원 청주 전주 제주
## 1  1738.333 1937.5 1627.5 1280 1610 1335 1980 2050 1890 1380 1980
## 2  1738.333 1937.5 1627.5 1280 1610 1560 1980 2050 1890 1380 1980
## 3  1738.333 1937.5 1627.5 1280 1610 1560 1980 2050 2010 1380 1980
## 4  1538.333 1692.5 1442.5 1380 1390 1310 1000 2050 2010 1380 1980
## 5  1538.333 1692.5 1492.5 1380 1390 1310 1000 2050 2010 1380 2140
## 6  1543.333 1692.5 1492.5 1380 1390 1310 1000 2110 2340 1380 2140
## 7  1543.333 1692.5 1492.5 1380 1390 1310 1000 1580 2340 1380 2140
## 8  1543.333 1692.5 1492.5 1380 1390 1310 1000 1580 2340 1380 2140
## 9  1676.667 2080.0 1760.0 1380 1635 1660 2380 1580 2340 1380 2140
## 10 1676.667 2080.0 1860.0 1380 1735 1635 1380 1580 2340 1380 2320
```

```
pig.region.monthly.mean  <- read.csv("Data/pig.region.monthly.mean.csv",
 header=T, fileEncoding="UTF-8")[,-1]
head(pig.region.monthly.mean, n=10)
```

```
##    name region      month mean.price
## 1  광주   2401 2011-01-01   1782.619
## 2  광주   2401 2011-02-01   2073.235
## 3  광주   2401 2011-03-01   1771.364
## 4  광주   2401 2011-04-01   1785.000
## 5  광주   2401 2011-05-01   1956.750
## 6  광주   2401 2011-06-01   2357.857
## 7  광주   2401 2011-07-01   2452.976
## 8  광주   2401 2011-08-01   2253.182
## 9  광주   2401 2011-09-01   1997.375
## 10 광주   2401 2011-10-01   1731.375
```

- read.csv() : 이 함수는 csv 파일을 읽는 기능을 함
- head() : 데이터의 앞부분을 출력함

일간 품목별 평균 데이터 date.item.mean 와 월간 품목별 평균 데이터 month.item.mean 을 불러들여온다.

```
date.item.mean <- read.csv("Data/date.item.mean.csv", header=T, fileEnco
ding="UTF-8")[,-1]
head(date.item.mean, n=10)
```

```
##    item       date mean.price code      exp name
## 1   111 2011-01-03   40953.33 1200 품목코드   쌀
## 2   111 2011-01-04   40993.33 1200 품목코드   쌀
## 3   111 2011-01-05   41153.33 1200 품목코드   쌀
## 4   111 2011-01-06   41020.00 1200 품목코드   쌀
## 5   111 2011-01-07   41013.33 1200 품목코드   쌀
## 6   111 2011-01-10   41180.00 1200 품목코드   쌀
## 7   111 2011-01-11   41080.00 1200 품목코드   쌀
## 8   111 2011-01-12   41146.67 1200 품목코드   쌀
## 9   111 2011-01-13   41146.67 1200 품목코드   쌀
## 10  111 2011-01-14   41146.67 1200 품목코드   쌀
```

```
month.item.mean <- read.csv("Data/month.item.mean.csv", header=T, fileEn
coding="UTF-8")[,-1]
head(month.item.mean)
```

```
##     name item      month mean.price
## 1 닭고기  515 2011-01-01   5517.809
## 2 닭고기  515 2011-02-01   6612.114
## 3 닭고기  515 2011-03-01   6983.201
## 4 닭고기  515 2011-04-01   6911.101
## 5 닭고기  515 2011-05-01   6004.349
## 6 닭고기  515 2011-06-01   5242.317
```

3. 데이터 가공하기

농산물간의 군집 분석을 위해 축산물을 제외하고 농산물 데이터만을 추출하여 farm.product 데이터를 생성한다.

```
temp <- dlply(date.item.mean, .(name), summarise, mean.price)
farm.product <- data.frame(쌀=unlist(temp$쌀),
    배추=unlist(temp$배추),
    상추=unlist(temp$상추),
    호박=unlist(temp$호박),
    양파=unlist(temp$양파),
    파프리카=unlist(temp$파프리카),
    참깨=unlist(temp$참깨),
    사과=unlist(temp$사과))
head(farm.product, n=10)
```

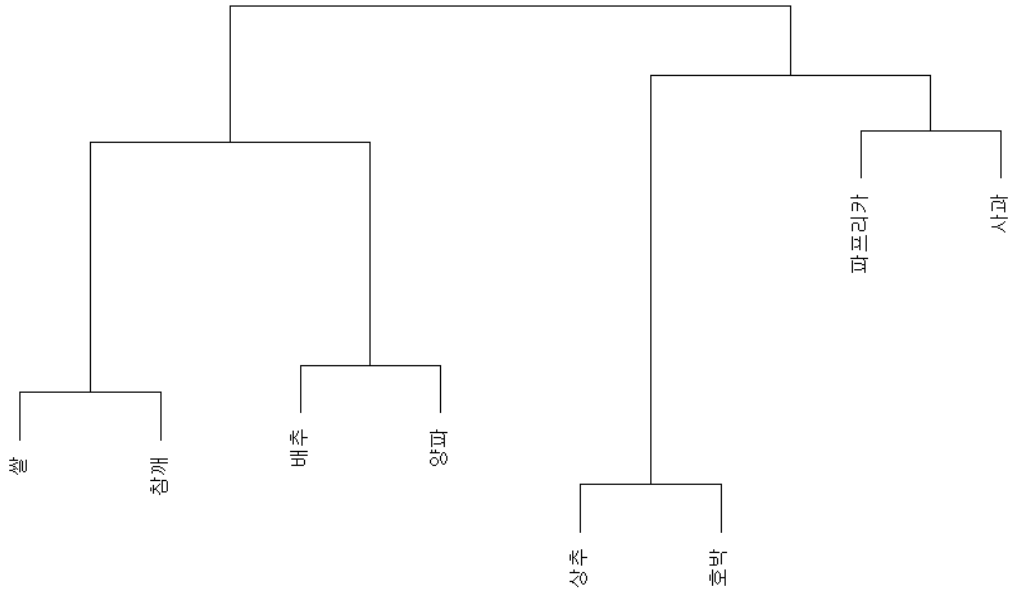
```
##             쌀     배추     상추     호박     양파  파프리카     참깨
## ..11  40953.33 4251.818 642.9024 1123.889 1903.333 1015.2963 12133.33
## ..12  40993.33 4168.182 633.3095 1134.259 1877.667 1015.1071 12133.33
## ..13  41153.33 4375.833 629.5227 1145.370 1864.333 1015.1071 12133.33
## ..14  41020.00 4609.167 651.7826 1231.321 2070.000 1043.1071 12133.33
## ..15  41013.33 4609.167 653.9565 1227.925 2071.667 1043.1071 12133.33
## ..16  41180.00 4399.167 648.2391 1275.849 2075.000 1022.7500 12133.33
## ..17  41080.00 4422.500 642.3696 1277.222 2075.000 1025.2857 12133.33
## ..18  41146.67 4495.000 627.4783 1303.774 2067.667 1017.8621 12133.33
## ..19  41146.67 4521.111 635.1522 1350.185 1856.333 1006.8276 12040.00
## ..110 41146.67 4521.111 638.8478 1365.472 1843.000  997.5172 12040.00
##           사과
## ..11  24904.00
## ..12  24904.00
## ..13  25176.73
## ..14  25358.55
## ..15  25994.91
## ..16  26176.73
## ..17  27027.64
## ..18  27482.18
## ..19  27320.09
## ..110 27320.09
```

- dlply() : 데이터 프레임 형태를 품목이름별로 **list** 형태로 출력함
  - unlist() : list 형태를 vector 형태로 변환함
  - data.frame : vector 형태의 데이터를 data frame 형태로 변환함

4. 클러스터링 분석 및 데이터 시각화

농산물 자료에 대하여 군집분석을 시행한다.

```
plot(hclust(diss(farm.product,"COR")), axes = F, ann = F)
```



- `diss(farm.product,"COR")` : 데이터 프레임 형태의 farm.product 를 correlation 방법으로 계산함
  - `hclust()` : 계산된 오브젝트를 군집화함
  - `plot()` : 군집된 결과를 그림으로 나타냄
  - `axes = F` : 모든 축을 출력하지 않도록 설정함
  - `ann = F` : 모든 축에 대한 설명을 출력하지 않도록 설정함

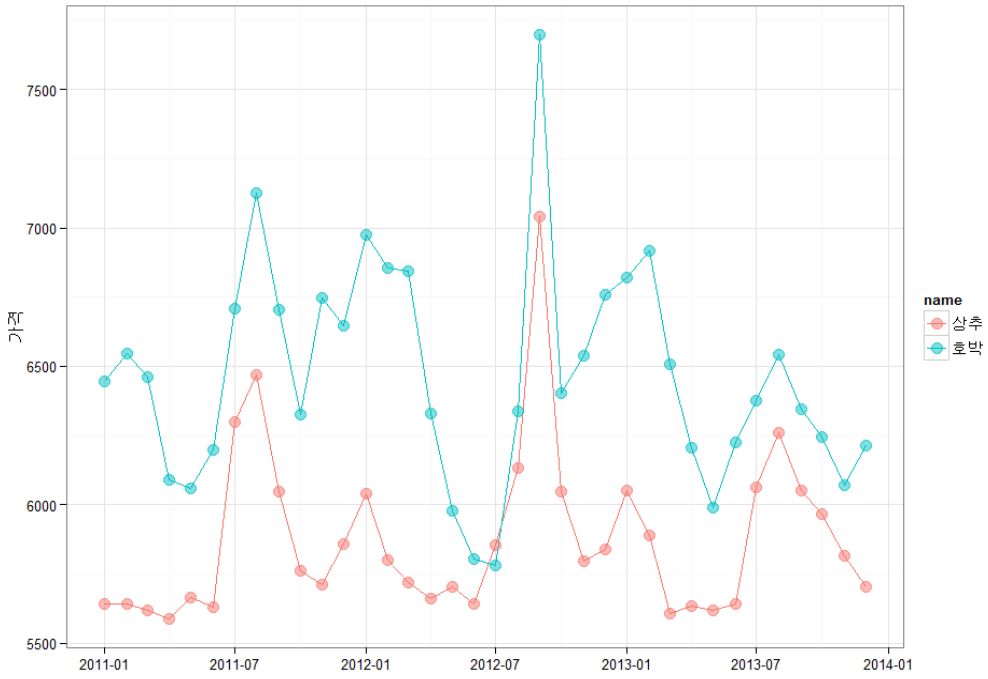
시계열도를 그리기 전에 시각화를 위해 month 데이터를 가공한다.

```
month.item.mean$month <- as.Date(as.yearmon(month.item.mean$month, "%Y-%
m"))
```

- `as.yearmon()` : **factor** 타입의 데이터를 월별 시계열로 변환함
  - `as.Date()` : 변환된 시계열 데이터를 **date** 타입으로 변환함

가장 유의한 군집으로 형성된 상추와 호박에 대한 시계열 그림을 그려본 결과, 가격변화가 매우 유사함을 확인할 수 있다.

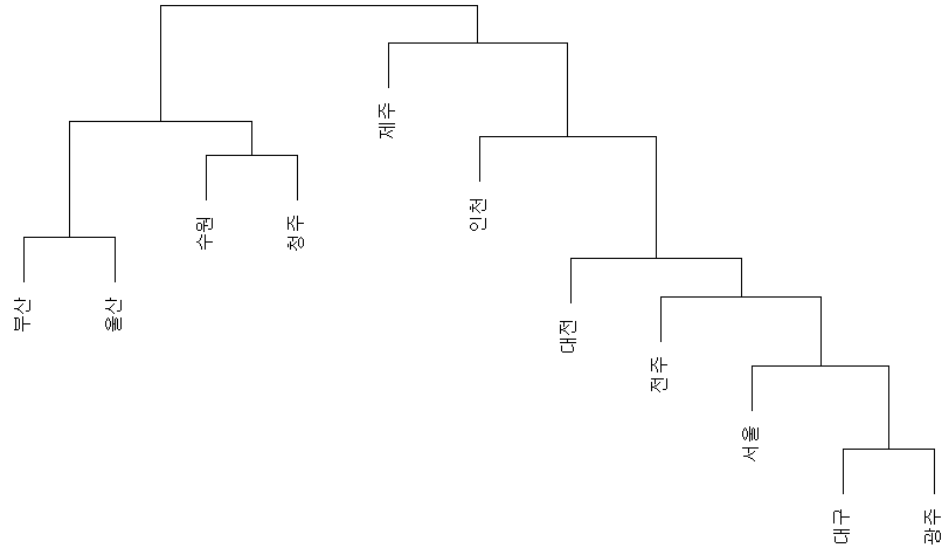
```
ggplot(month.item.mean[month.item.mean$name %in% c("상추", "호박"),],
    aes(x=month, y=mean.price, colour=name, group=name)) + geom_line() +

    theme_bw() + geom_point(size=6, shape=20, alpha=0.5) +
    ylab("가격") + xlab("")
```



- ggplot() : 메인 함수로 데이터 셋과 표현할 데이터 변수명을 정의함
  - month.item.mean[month.item.mean$name %in% c("상추", "호박"),] : 품목 이름이 상추와 호박인 데이터만 추출함
  - %in% : 조건에 대해 일치여부를 boolean 형으로 출력하는 기능의 연산자임
  - aes(x=month, y=mean.price, colour=name, group=name)) : 월별 가격데이터를 품목별로 표현함
- geom_line() : 데이터를 line 형태로 시각화함
- geom_point() : 데이터를 point 형태로 시각화함
- theme_bw() : 흰색 배경에 검은 색 눈금 선에 테마를 적용함
- xlab(), ylab() : x 축 또는 y 축의 이름을 지정함

돼지고기 자료에 대하여 군집분석을 시행한다.

```
plot(hclust(diss(pig.region,"COR")), axes = F, ann = F)
```



- `diss(farm.product,"COR")` : 데이터 프레임 형태의 farm.product 를 **correlation** 방법으로 계산함
  - `hclust()` : 계산된 오브젝트를 군집화함
  - `plot()` : 군집된 결과를 그림으로 나타냄
  - `axes = F` : 모든 축을 출력하지 않도록 설정함
  - `ann = F` : 모든 축에 대한 설명을 출력하지 않도록 설정함

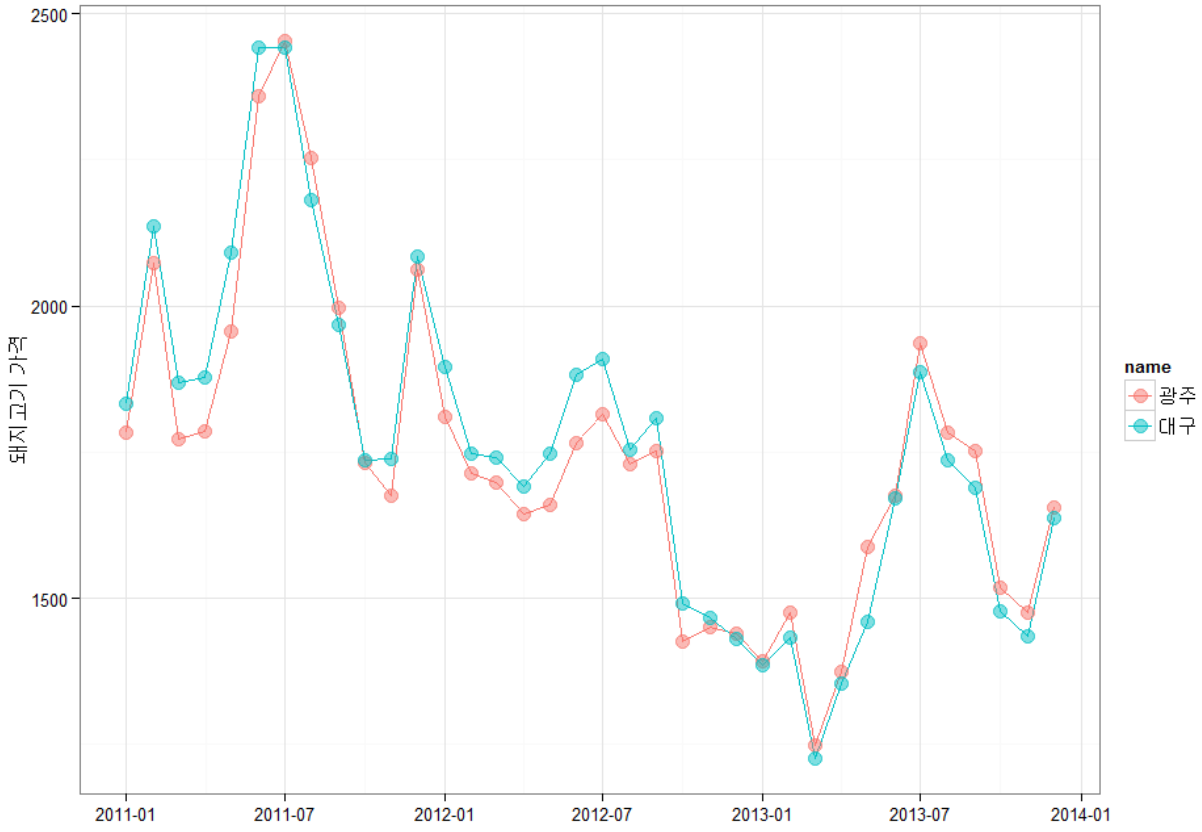
시계열도를 그리기 전에 시각화를 위해 월별 데이터를 가공한다.

```
pig.region.monthly.mean$month <- as.Date(as.yearmon(pig.region.monthly.m
ean$month, "%Y-%m"))
```

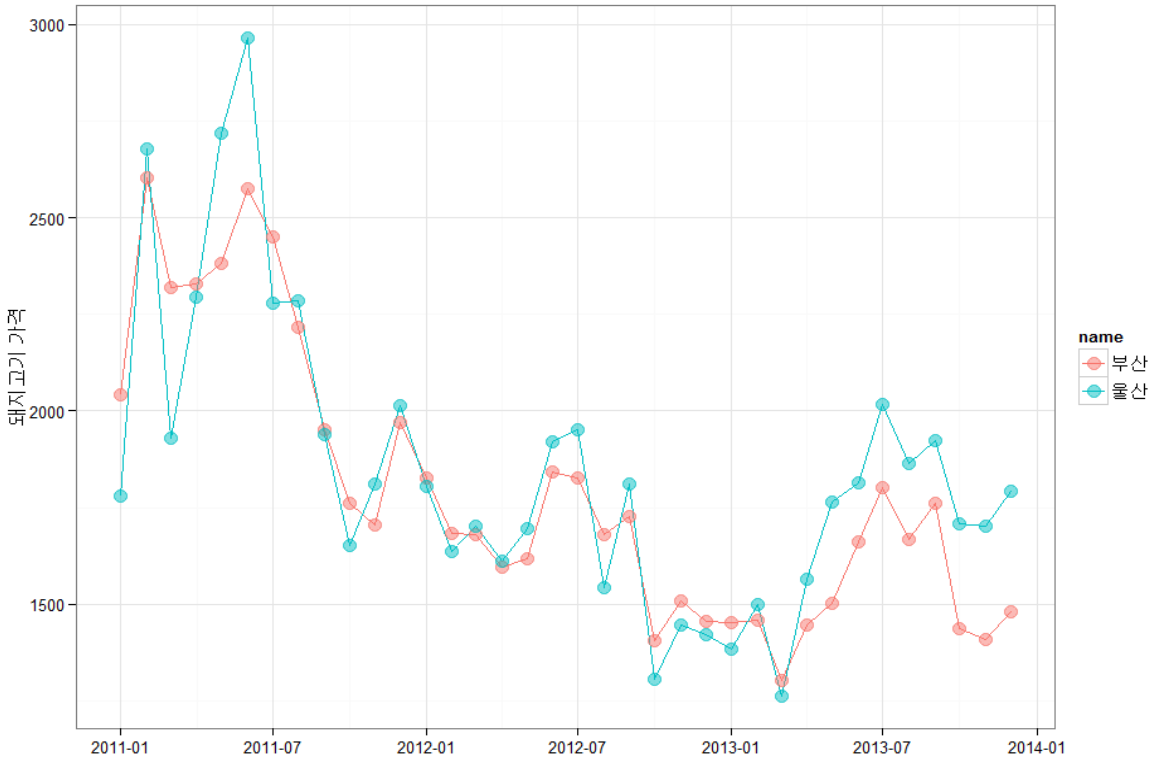
- `as.yearmon()` : **factor** 타입의 데이터를 월별 시계열로 변환함
  - `as.Date()` : 변환된 시계열 데이터를 **date** 타입으로 변환함

가장 유의하게 군집으로 형성된 대구와 광주, 부산과 울산의 시계열 그림을 그려본 결과 가격 변화가 매우 유사함을 확인할 수 있다.

```
ggplot(pig.region.monthly.mean[pig.region.monthly.mean$region %in% c(2200,2401),],
    aes(x=month, y=mean.price, colour=name, group=name)) +
geom_line() + theme_bw() +
geom_point(size=6, shape=20, alpha=0.5) +
ylab("돼지고기 가격") + xlab("")
```



```
ggplot(pig.region.monthly.mean[pig.region.monthly.mean$region %in% c(210
0,2601),],
    aes(x=month, y=mean.price, colour=name, group=name)) +
geom_line() + theme_bw() +
geom_point(size=6, shape=20, alpha=0.5) +
ylab("돼지고기 가격") + xlab("")
```



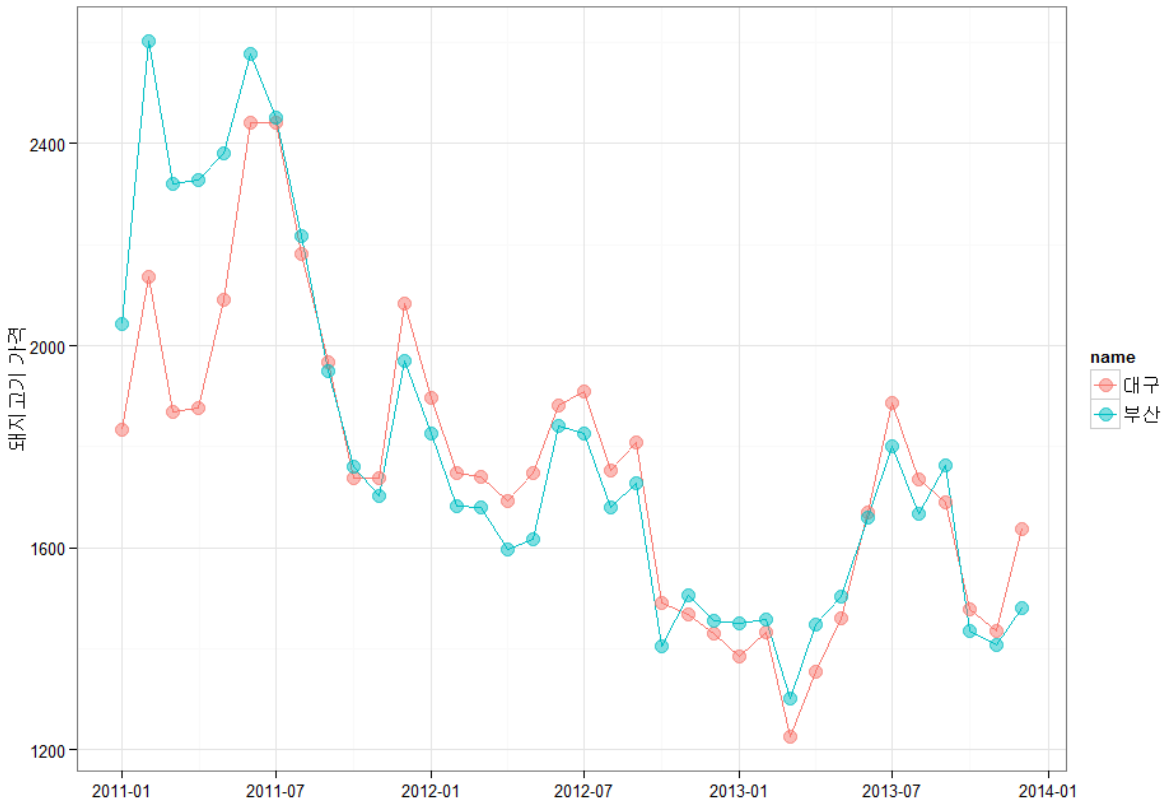
서로 다른 그룹으로 묶인 대구와 부산의 돼지고기 가격에 대한 시계열 그림을 그려본 결과, 초기에 서로 가격변화가 다른데, 이는 구제역 발생 후 대구는 큰 영향을 받았으나 부산은 그렇지 않았기 때문으로 추정된다.

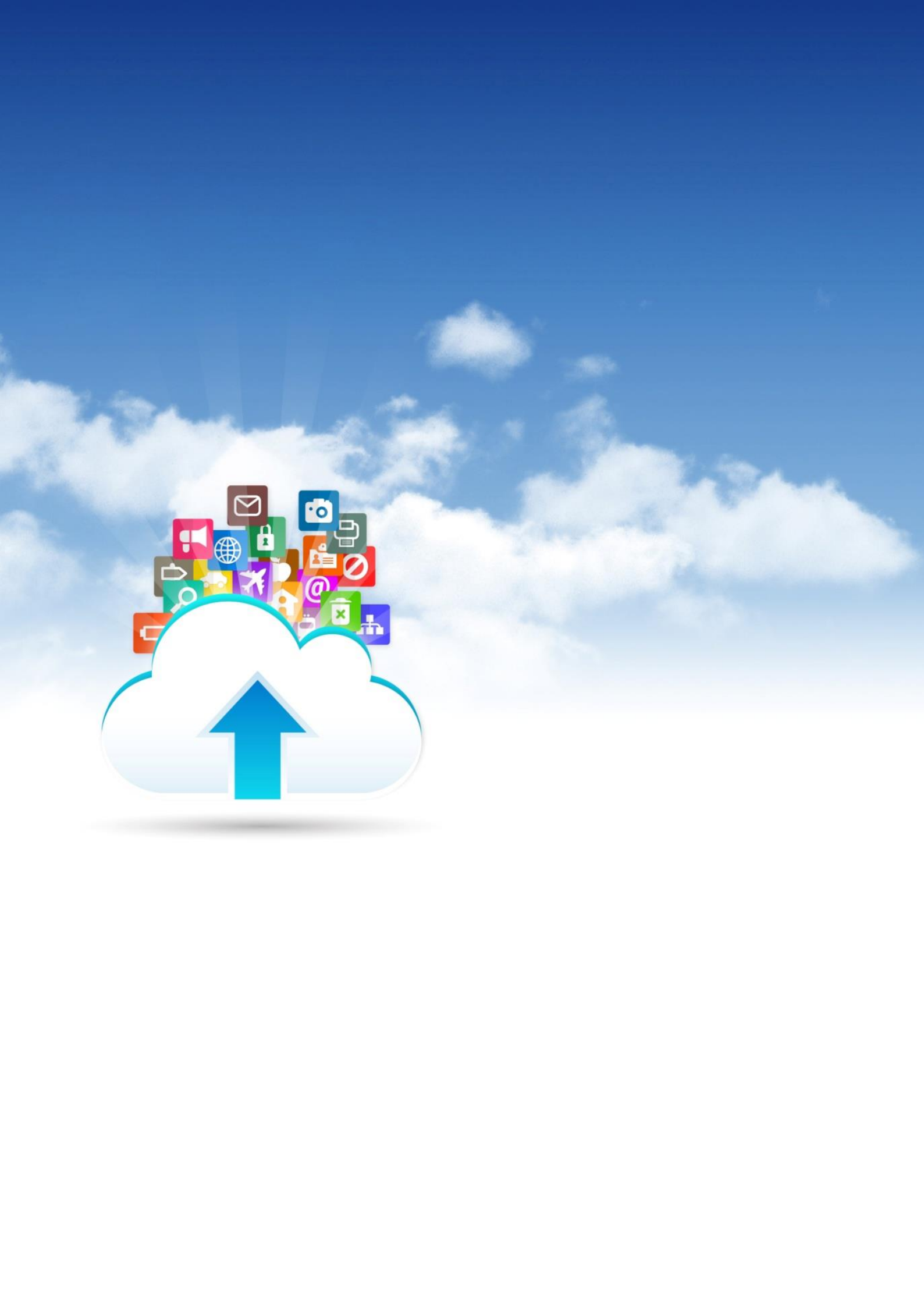
2011 년 초반에 돼지고기 가격의 변동성이 컸던 점을 활용해 뉴스를 검색해 본 결과 다음과 같은 구제역 관련 뉴스를 찾을 수 있었다.

- (http://news.kmib.co.kr/article/view.asp?arcid=0004564454&code=11151100)
- (http://blog.daum.net/sun6377/5061880)
- (http://blog.naver.com/PostView.nhn?blogId=giant50&logNo=140143825979)

```
ggplot(pig.region.monthly.mean[pig.region.monthly.mean$region %in% c(220
0,2100),],
    aes(x=month, y=mean.price, colour=name, group=name)) + geom_line() +
```

```
theme_bw() + geom_point(size=6, shape=20, alpha=0.5) +
ylab("돼지고기 가격") + xlab("")
```





V. 날씨 자료와 농산물 자료의 인과관계 분석

1. 필요 패키지 불러오기
2. 데이터 불러오기
3. 데이터 가공하기
4. 데이터 시각화

V 날씨 자료와 농산물 자료의 인과관계 분석

한국농수산식품유통공사에서 제공받은 2011 년~2013 년 농축산물의 소매가격 정보 중 상추와 호박의 서울 평균 소매가격 자료와 2010 년~2014 기상 정보 중 서울 지역의 일별 평균 강수량 자료를 시각화 하여 날씨와 농산물 가격간의 인과관계를 분석한다.

1. 필요 패키지 불러오기

아래 R 명령어는 여러 개의 패키지를 로딩하는데 유용한 수행방법이다.

```
library4 <- c("plyr", "stringr","dygraphs", "zoo", "xts")
unlist(lapply(library4, require, character.only = TRUE ))
```

➢ 각 패키지의 역할을 간략히 설명하면 다음과 같음

패키지 명	설명
plyr	데이터 핸들링을 하기 위한 라이브러리
stringr	문자열 핸들링을 하기 위한 라이브러리
dypraphs	시계열 시각화 기능 라이브러리
zoo	문자형 데이터를 데이트 형식으로 변환하기 위한 라이브러리
xts	시계열 오브젝트를 생성하기 위한 라이브러리

2. 데이터 불러오기

CSV 형태의 농축산물 데이터와 기상 데이터는 다음과 같이 읽어 들인다.

```
product <- read.csv("Data/product.csv", header=T, fileEncoding="UTF-8")
weather <- read.csv("Data/weather.csv", header=T, fileEncoding="UTF-8")
code <- read.csv("Data/code.csv", header=T, fileEncoding="UTF-8")
```

➢ read.csv() : 이 함수는 csv 파일을 읽는 기능을 함

3. 데이터 가공하기

3.1 농축산물 데이터 가공

서울지역 코드번호를 확인하기 위해 code 오브젝트에서 지역코드만 추출하여 확인한다.

```
subset(code, 구분코드설명 %in% c("지역코드"))
```

```
##    구분코드 구분코드설명 분류코드 분류코드설명
## 16     1300     지역코드     1101         서울
## 17     1300     지역코드     2100          부산
## 18     1300     지역코드     2200          대구
## 19     1300     지역코드     2300          인천
## 20     1300     지역코드     2401          광주
## 21     1300     지역코드     2501          대전
## 22     1300     지역코드     2601          울산
## 23     1300     지역코드     3111          수원
## 24     1300     지역코드     3113        의정부
## 25     1300     지역코드     3145          용인
## 26     1300     지역코드     3211          춘천
## 27     1300     지역코드     3311          청주
## 28     1300     지역코드     3511          전주
## 29     1300     지역코드     3613          순천
## 30     1300     지역코드     3711          포항
## 31     1300     지역코드     3714          안동
## 32     1300     지역코드     3814          창원
## 33     1300     지역코드     3911          제주
```

- `subset()` : 전체 데이터에서 특정 조건을 만족하는 값을 출력함
  - `%in%` : 조건에 대해 일치여부를 boolean 형으로 출력하는 기능의 연산자임

일별 농축산물의 평균 소매가격을 추출하기 위해 품목별 코드 번호를 확인하기 위한 category 오브젝트를 생성한다.
여기서 category 에는 code 데이터에서 구분코드설명이 품목코드에 해당하는 값만 추출되어 저장된다.

```
category <- subset(code, code$구분코드설명=="품목코드")
category
```

```
##    구분코드 구분코드설명 분류코드 분류코드설명
## 6      1200     품목코드      111          쌀
## 7      1200     품목코드      211         배추
## 8      1200     품목코드      214         상추
## 9      1200     품목코드      224         호박
## 10     1200     품목코드      245         양파
## 11     1200     품목코드      256      파프리카
## 12     1200     품목코드      312         참깨
## 13     1200     품목코드      411         사과
## 14     1200     품목코드      514      돼지고기
## 15     1200     품목코드      515        닭고기
```

- subset() : 데이터의 특정부분에 대해 조건을 만족하는 값을 반환해주는 함수임

R 서버에서 원활하게 데이터에 접근하기 위해 원시 데이터의 변수명을 한글에서 영어로 변환해준다.

```
colnames(product) <- c('date','category','item','region','mart','price')
colnames(category) <- c('code', 'exp', 'item', 'name')
```

- colnames() : 데이터 프레임의 열의 이름을 지정함
  - product 의 변수명은 date, category, item, region, mart, price 가 됨
- head() : 데이터의 앞부분을 출력함

서울지역의 가격만 추출하여 품목(item), 일자(date)별로 평균 가격을 구하여 품목에 대한 데이터 category 데이터와 merge 하여 seoul.item 데이터를 생성한다.

일별 평균 가격을 생성하여 품목별로 정렬한 seoul.item.mean 데이터를 생성한다.

```
seoul.item <- merge(ddply(product[which(product$region==1101),], .(item,
 date),
 summarise, mean.price=mean(price)), category, by="item", all=T)
head(seoul.item, n=10)
```

```
##    item       date mean.price code      exp name
## 1   111 2011-01-03   41016.67 1200 품목코드   쌀
## 2   111 2011-01-04   41016.67 1200 품목코드   쌀
## 3   111 2011-01-05   41350.00 1200 품목코드   쌀
## 4   111 2011-01-06   41350.00 1200 품목코드   쌀
## 5   111 2011-01-07   41316.67 1200 품목코드   쌀
## 6   111 2011-01-10   41316.67 1200 품목코드   쌀
## 7   111 2011-01-11   41316.67 1200 품목코드   쌀
## 8   111 2011-01-12   41316.67 1200 품목코드   쌀
## 9   111 2011-01-13   41316.67 1200 품목코드   쌀
## 10  111 2011-01-14   41316.67 1200 품목코드   쌀
```

- `merge()` : 두 데이터 프레임을 공통된 값 **item** 을 기준으로 묶는 함수
- `ddply()` : 데이터 프레임 형태를 주어진 조건에 따라 정렬할 후 **summarise** 한 값을 **data frame** 형태로 출력함
  - `product[which(product$region==1101),]` : 서울지역코드에 맞는 조건을 만족하는 값이 있는 곳을 추출함
- `head()` : 데이터의 앞부분을 출력함

```
seoul.item.mean <- ddply(seoul.item, .(item, date), summarise, name, mea
n.price)
```

R 서버에서 원활하게 데이터에 접근하기 위해 seoul.item.mean 의 변수명을 한글에서 영어로 변환해준다.

```
colnames(seoul.item.mean) <- c('item', 'date','item.name', 'mean.price')
head(seoul.item.mean , n=10)
```

```
##    item       date item.name mean.price
## 1   111 2011-01-03        쌀   41016.67
## 2   111 2011-01-04        쌀   41016.67
## 3   111 2011-01-05        쌀   41350.00
## 4   111 2011-01-06        쌀   41350.00
## 5   111 2011-01-07        쌀   41316.67
## 6   111 2011-01-10        쌀   41316.67
## 7   111 2011-01-11        쌀   41316.67
## 8   111 2011-01-12        쌀   41316.67
## 9   111 2011-01-13        쌀   41316.67
## 10  111 2011-01-14        쌀   41316.67
```

- colnames() : 데이터 프레임의 열의 이름을 지정함
- head() : 데이터의 앞부분을 출력함

3.2 기상 데이터 가공

함수 read.csv 에 읽혀 생성된 R 오브젝트 weather 를 출력하면 다음과 같다.

```
str(weather)
```

```
## 'data.frame':    691920 obs. of  4 variables:
##  $ 지역    : Factor w/ 93 levels "강릉(청)","강진군(공)",..: 27 27 27
27 27 27 27 27 27 27 ...
##  $ 기상구분: Factor w/ 2 levels "강수량","평균기온": 1 1 1 1 1 1 1 1
1 ...
##  $ 측정값  : num  NA 3 NA 8.5 37.5 NA NA 3 0.4 NA ...
##  $ 일자    : Factor w/ 1860 levels "2010-01-01","2010-01-02",..: 157
188 219 250 281 312 343 3 34 65 ...
```

- str(product) : 변수가 4 개, 자료의 수가 691920 인 **data.frame** 형태임

weather 데이터는 전 지역 강수량과 평균기온이 입력된 데이터이다.

```
head(weather, n=10)
```

```
##          지역 기상구분 측정값       일자
## 1  백령도(기)   강수량     NA 2010-06-02
## 2  백령도(기)   강수량    3.0 2010-07-02
## 3  백령도(기)   강수량     NA 2010-08-02
## 4  백령도(기)   강수량    8.5 2010-09-02
## 5  백령도(기)   강수량   37.5 2010-10-02
## 6  백령도(기)   강수량     NA 2010-11-02
## 7  백령도(기)   강수량     NA 2010-12-02
## 8  백령도(기)   강수량    3.0 2010-01-03
## 9  백령도(기)   강수량    0.4 2010-02-03
## 10 백령도(기)   강수량     NA 2010-03-03
```

```
tail(weather, n=10)
```

```
##              지역 기상구분 측정값       일자
## 691911 백령도(기)   강수량    0.2 2010-08-01
## 691912 백령도(기)   강수량   96.0 2010-09-01
## 691913 백령도(기)   강수량     NA 2010-10-01
## 691914 백령도(기)   강수량     NA 2010-11-01
## 691915 백령도(기)   강수량     NA 2010-12-01
## 691916 백령도(기)   강수량    1.0 2010-01-02
## 691917 백령도(기)   강수량     NA 2010-02-02
## 691918 백령도(기)   강수량     NA 2010-03-02
## 691919 백령도(기)   강수량     NA 2010-04-02
## 691920 백령도(기)   강수량     NA 2010-05-02
```

- `head()` : 데이터의 앞부분을 출력함
- `tail()` : 데이터의 뒷부분을 출력함

R 서버에서 원활하게 데이터에 접근하기 위해 원시 데이터의 변수명을 한글에서 영어로 변환해준다.

```
colnames(weather) <- c('region','category','value','date')
```

- `colnames()` : 데이터 프레임의 열의 이름을 지정함

weather 데이터를 지역(region)별 이름순으로 나누어 region.weather 리스트 형식의 데이터를 생성한다.

```
region.weather <- dlply(weather, .(region))
```

➢ dlply() : 데이터 프레임 형태를 품목별로 **list** 형태로 출력함

region.weather 에서 서울에 대한 기상데이터만 쓰기 위해 지역별로 나누어진 리스트의 이름들을 확인한다.

```
names(region.weather)
```

```
##  [1] "강릉(청)"   "강진군(공)" "강화(관)"   "거제(관)"   "거창(기)"
##  [6] "경주(공)"   "고산(기)"   "고창(구)"   "고흥(관)"   "광양(공)"
## [11] "광주(청)"   "구미(기)"   "군산(기)"   "금산(관)"   "김해시(공)"
## [16] "남원(기)"   "남해(관)"   "대관령(기)" "대구(구)"   "대구(기)"
## [21] "대전(청)"   "동두천(기)" "동해(기)"   "목포(기)"   "문경(관)"
## [26] "밀양(관)"   "백령도(기)" "보령(기)"   "보성군(공)" "보은(관)"
## [31] "봉화(관)"   "부산(청)"   "부안(관)"   "부여(관)"   "북강릉(청)"
## [36] "북창원(공)" "산청(관)"   "상주(기)"   "서귀포(기)" "서산(기)"
## [41] "서울(청)"   "성산(기)"   "속초(기)"   "수원(기)"   "순창(공)"
## [46] "순천(구)"   "순천(기)"   "안동(기)"   "양산(공)"   "양평(관)"
## [51] "여수(기)"   "영광(공)"   "영덕(관)"   "영월(기)"   "영주(관)"
## [56] "영천(관)"   "완도(기)"   "울릉도(기)" "울산(기)"   "울진(기)"
## [61] "원주(기)"   "의령군(공)" "의성(관)"   "이천(기)"   "인제(관)"
## [66] "인천(기)"   "임실(관)"   "장수(관)"   "장흥(관)"   "전주(기)"
## [71] "정선군(공)" "정읍(기)"   "제주(청)"   "제천(관)"   "진도군(공)"
## [76] "진주(기)"   "창원(기)"   "천안(기)"   "철원(기)"   "청송군(공)"
## [81] "청주(기)"   "추풍령(기)" "춘천(기)"   "충주(기)"   "태백(관)"
## [86] "통영(기)"   "파주(기)"   "포항(기)"   "함양군(공)" "합천(관)"
## [91] "해남(관)"   "홍천(관)"   "흑산도(기)"
```

➢ names() : 리스트 명을 출력함

서울에 대한 기상데이터를 확인한다.

```
head(region.weather[[41]], n=10)
```

```
##      region category value       date
## 1  서울(청) 평균기온  -6.8 2011-01-01
## 2  서울(청) 평균기온  -0.2 2011-02-01
## 3  서울(청) 평균기온   0.5 2011-03-01
## 4  서울(청) 평균기온   9.1 2011-04-01
## 5  서울(청) 평균기온  12.5 2011-05-01
## 6  서울(청) 평균기온  18.0 2011-06-01
## 7  서울(청) 평균기온  25.1 2011-07-01
```

```
## 8  서울(청) 평균기온  25.6 2011-08-01
## 9  서울(청) 평균기온  27.0 2011-09-01
## 10 서울(청) 평균기온  12.7 2011-10-01
```

```
tail(region.weather[[41]], n=10)
```

```
##        region category value       date
## 7431 서울(청)   강수량   9.5 2010-03-31
## 7432 서울(청)   강수량    NA 2010-04-31
## 7433 서울(청)   강수량   1.5 2010-05-31
## 7434 서울(청)   강수량    NA 2010-06-31
## 7435 서울(청)   강수량    NA 2010-07-31
## 7436 서울(청)   강수량   2.0 2010-08-31
## 7437 서울(청)   강수량    NA 2010-09-31
## 7438 서울(청)   강수량    NA 2010-10-31
## 7439 서울(청)   강수량    NA 2010-11-31
## 7440 서울(청)   강수량    NA 2010-12-31
```

- `head()` : 데이터의 앞부분을 출력함
- `tail()` : 데이터의 뒷부분을 출력함

분석 대상은 서울의 강수량 데이터이므로, category 가 강수량인 값들만 추출하여 init.seoul.rain 데이터를 생성한다.

```
init.seoul.rain <- region.weather[[41]][which(region.weather[[41]][,2]==
"강수량"),]
head(init.seoul.rain, n=10)
```

```
##       region category value       date
## 373 서울(청)   강수량    NA 2011-01-01
## 374 서울(청)   강수량    NA 2011-02-01
## 375 서울(청)   강수량   2.3 2011-03-01
## 376 서울(청)   강수량    NA 2011-04-01
## 377 서울(청)   강수량   0.0 2011-05-01
## 378 서울(청)   강수량  21.5 2011-06-01
## 379 서울(청)   강수량    NA 2011-07-01
## 380 서울(청)   강수량   0.5 2011-08-01
## 381 서울(청)   강수량    NA 2011-09-01
## 382 서울(청)   강수량    NA 2011-10-01
```

- which() : 주어진 조건을 만족하는 값이 있는 것을 추출함
- head() : 데이터의 앞부분을 출력함

데이터를 날짜순대로 배열하여 sort.seoul.rain 데이터를 생성한다.

```
sort.seoul.rain <- dlply(init.seoul.rain, .(date))
head(sort.seoul.rain, n=10)
```

```
## $`2010-01-01`
##     region category value       date
## 1 서울(청)   강수량    NA 2010-01-01
## 2 서울(청)   강수량    NA 2010-01-01
##
## $`2010-01-02`
##     region category value       date
## 1 서울(청)   강수량   1.4 2010-01-02
## 2 서울(청)   강수량   1.4 2010-01-02
##
## $`2010-01-03`
##     region category value       date
## 1 서울(청)   강수량    NA 2010-01-03
## 2 서울(청)   강수량    NA 2010-01-03
##
## $`2010-01-04`
```

```
##     region category value       date
## 1 서울(청)   강수량  14.2 2010-01-04
## 2 서울(청)   강수량  14.2 2010-01-04
##
## $`2010-01-05`
##     region category value       date
## 1 서울(청)   강수량     0 2010-01-05
## 2 서울(청)   강수량     0 2010-01-05
##
## $`2010-01-06`
##     region category value       date
## 1 서울(청)   강수량    NA 2010-01-06
## 2 서울(청)   강수량    NA 2010-01-06
##
## $`2010-01-07`
##     region category value       date
## 1 서울(청)   강수량    NA 2010-01-07
## 2 서울(청)   강수량    NA 2010-01-07
##
## $`2010-01-08`
##     region category value       date
## 1 서울(청)   강수량    NA 2010-01-08
## 2 서울(청)   강수량    NA 2010-01-08
##
## $`2010-01-09`
##     region category value       date
## 1 서울(청)   강수량   0.4 2010-01-09
## 2 서울(청)   강수량   0.4 2010-01-09
##
## $`2010-01-10`
##     region category value       date
## 1 서울(청)   강수량    NA 2010-01-10
## 2 서울(청)   강수량    NA 2010-01-10
```

- `dlply()` : 데이터 프레임 형태를 품목별로 **list** 형태로 출력함
- `head()` : 데이터의 앞부분을 출력함

일별로 데이터 값들이 2 개씩 중복되어 있으므로 하나의 값만 추출하여 `resort.seoul.rain` 데이터를 생성한다.

```
resort.seoul.rain <- lapply(1:length(sort.seoul.rain), function(x) sort.
seoul.rain[[x]][1,])
head(resort.seoul.rain, n=10)
```

```
## [[1]]
##     region category value       date
## 1 서울(청)   강수량    NA 2010-01-01
##
## [[2]]
```

```
##     region category value       date
## 1 서울(청)   강수량   1.4 2010-01-02
##
## [[3]]
##     region category value       date
## 1 서울(청)   강수량    NA 2010-01-03
##
## [[4]]
##     region category value       date
## 1 서울(청)   강수량  14.2 2010-01-04
##
## [[5]]
##     region category value       date
## 1 서울(청)   강수량     0 2010-01-05
##
## [[6]]
##     region category value       date
## 1 서울(청)   강수량    NA 2010-01-06
##
## [[7]]
##     region category value       date
## 1 서울(청)   강수량    NA 2010-01-07
##
## [[8]]
##     region category value       date
## 1 서울(청)   강수량    NA 2010-01-08
##
## [[9]]
##     region category value       date
## 1 서울(청)   강수량   0.4 2010-01-09
##
## [[10]]
##     region category value       date
## 1 서울(청)   강수량    NA 2010-01-10
```

- `lapply()` : **list** 형태의 `sort.seoul.rain` 오브젝트를 정의한 함수 **function(x)** 으로 처리 후 **list** 형태로 출력함
- `head()` : 데이터의 앞부분을 출력함

반복문을 통해 `resort.seul.rain` 데이터에서 date 와 value 값만 추출하여 `seoul.rain` 데이터를 생성한다.

```
seoul.rain <- data.frame(date=unlist(lapply(1:length(resort.seoul.rain),
 function(x) resort.seoul.rain[[x]][,4])), rain=unlist(lapply(1:length(r
esort.seoul.rain), function(x) resort.seoul.rain[[x]][,3])))
head(seoul.rain, n=10)
```

```
##          date rain
## 1  2010-01-01   NA
## 2  2010-01-02  1.4
## 3  2010-01-03   NA
## 4  2010-01-04 14.2
## 5  2010-01-05  0.0
## 6  2010-01-06   NA
```

```
## 7  2010-01-07   NA
## 8  2010-01-08   NA
## 9  2010-01-09  0.4
## 10 2010-01-10   NA
```

- `lapply()` : **list** 형태의 `resort.seoul.rain` 오브젝트를 정의한 함수 **function(x)** 으로 처리 후 **list** 형태로 출력함
  - `unlist()` : **list** 형태를 **vector** 형태로 변환함
- `head()` : 데이터의 앞부분을 출력함

분석을 용이하게 하기 위하여 NA 로 입력된 강수량을 0 으로 변환한다.

```
seoul.rain[,2][is.na(seoul.rain[,2])] <- 0
head(seoul.rain, n=10)
```

```
##          date rain
## 1  2010-01-01  0.0
## 2  2010-01-02  1.4
## 3  2010-01-03  0.0
## 4  2010-01-04 14.2
## 5  2010-01-05  0.0
## 6  2010-01-06  0.0
## 7  2010-01-07  0.0
## 8  2010-01-08  0.0
## 9  2010-01-09  0.4
## 10 2010-01-10  0.0
```

- `is.na()` : 어떤 변수에 NA 가 저장되어 있는지 확인함
- `변수[is.na(변수)] <- 0` : NA 값을 0 으로 변환함

3.3 농산물 데이터와 기상 데이터의 가공

농산물의 소매가격은 2011 년부터 2013 년까지의 데이터이고, 기상데이터는 2010 년부터 2014 년까지의 데이터이다.

함께 분석하기 위해서 date 에 따라 두 데이터를 합쳐서 `seoul.item.rain` 데이터를 생성한다.

```
seoul.item.rain <- merge(seoul.rain, seoul.item.mean, by="date", all=T)
head(seoul.item.rain)
```

```
##         date rain item item.name mean.price
## 1 2010-01-01  0.0   NA      <NA>         NA
## 2 2010-01-02  1.4   NA      <NA>         NA
## 3 2010-01-03  0.0   NA      <NA>         NA
## 4 2010-01-04 14.2   NA      <NA>         NA
## 5 2010-01-05  0.0   NA      <NA>         NA
## 6 2010-01-06  0.0   NA      <NA>         NA
```

- `merge()` : 두 데이터 프레임을 공통된 값 **item** 을 기준으로 묶는 함수
- `head()` : 데이터의 앞부분을 출력함

NA 값으로 표현된 2010 년 농산물 데이터 열을 제거하고 2011 년부터 2013 년까지의 데이터를 생성한다.

```
seoul.item.rain <- ddply(seoul.item.rain[!seoul.item.rain$mean.price %i
n% NA,], .(item.name))
head(seoul.item.rain)
```

```
##         date rain item item.name mean.price
## 1 2011-01-03    0  515    닭고기   4980.000
## 2 2011-01-04    0  515    닭고기   5686.667
## 3 2011-01-05    0  515    닭고기   5686.667
## 4 2011-01-06    0  515    닭고기   6113.333
## 5 2011-01-07    0  515    닭고기   5620.000
## 6 2011-01-10    0  515    닭고기   5620.000
```

- `ddply()` : 데이터 프레임 형태를 주어진 조건에 따라 정렬할 후 **summarise** 한 값을 **data frame** 형태로 출력함
- 데이터`[!` 변수 `%in% NA,]` : 데이터에서 변수에 NA 가 있는 열을 제거함
  - `%in%` : 조건에 대해 일치여부를 boolean 형으로 출력하는 기능의 연산자임
- `head()` : 데이터의 앞부분을 출력함

4. 데이터 시각화

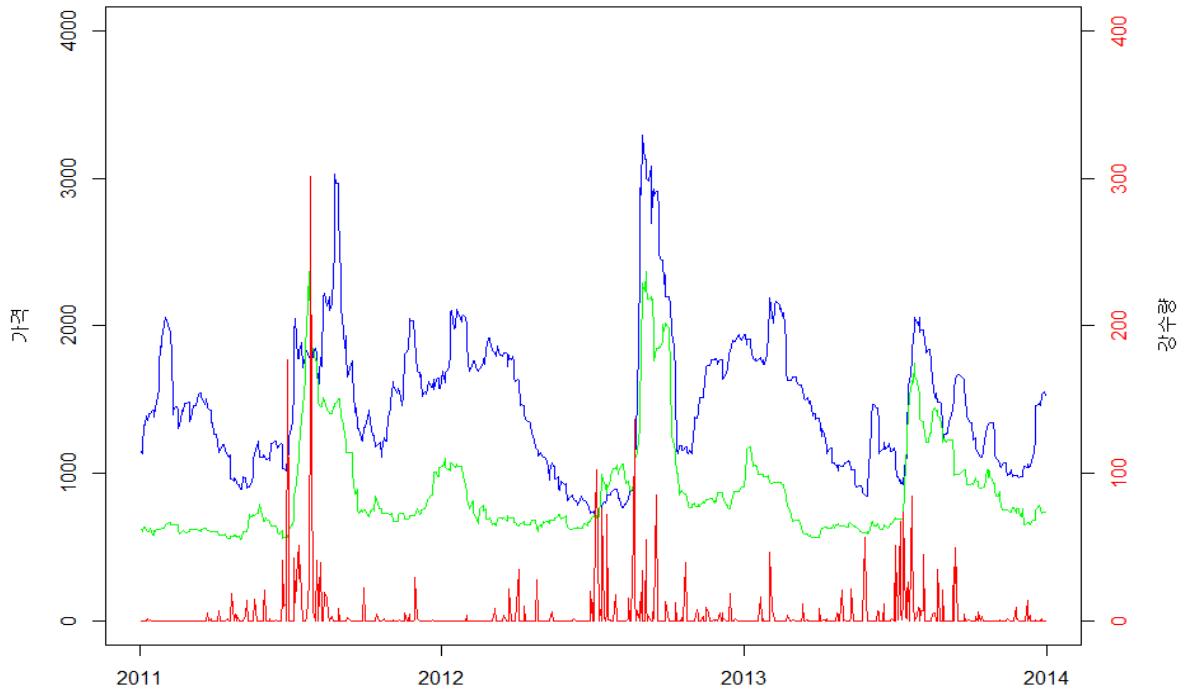
2011 년부터 2013 년까지 서울의 강수량 변화에 따른 상추의 가격 변화를 시각화하여 비교해본다.

빨간색으로 표현된 강수량이 크게 증가할 수록 상추와 호박의 가격이 크게 상승하는 것을 확인할 수 있다.

이로서 상추와 호박의 가격이 강수량에 큰 영향을 받고 있음을 알 수 있다.

4.1 plot()함수를 이용한 Graph Overlap

```
par(mar=c(3, 5, 3, 5))
plot(as.Date(seoul.item.rain[seoul.item.rain$item.name %in% c("호박"),]$dat
e),
 seoul.item.rain[seoul.item.rain$item.name %in% c("호박"),]$mean.price,
 type="l", col="blue", xlab="", ylab="", ylim=c(0,4000))
mtext("가격", side=2, line=3)

par(new=TRUE)
plot(as.Date(seoul.item.rain[seoul.item.rain$item.name %in% c("상추"),]$dat
e),
    seoul.item.rain[seoul.item.rain$item.name %in% c("상추"),]$mean.price,
    type="l", col="green", xlab="", ylab="", ylim=c(0,4000), axes=FALSE)

par(new=TRUE)
plot(as.Date(seoul.item.rain[seoul.item.rain$item.name %in% c("상추"),]$dat
e),
  seoul.item.rain[seoul.item.rain$item.name %in% c("상추"),]$rain,
  type="l", col="red", xlab="", ylab="", ylim=c(0,400), axes=FALSE)
axis(4, ylim=c(0,400), col.axis="red", las=3)
mtext("강수량", side=4, line=3)
```



- `par(mar)` : 현재 그래픽 장치의 그래픽 parameter 의 텍스트 라인을 수정함
  - 순서는 bottem-left-top-right 임
- `par(new=T)` : 첫 번째 plot 을 지우지 않고 두번째 plot 을 겹쳐 그림
- `mtext()` : 현재 그래픽의 네 가장자리 중 하나에 텍스트를 기록함
  - side=2 는 왼쪽 영역을 지정하고 side=4 는 오른쪽 영역을 지정함
- `plot()` : 데이터를 그래프로 나타냄
  - `seoul.item.rain[seoul.item.rain$item.name %in% c("호박"),]$date` : 품목 이름이 호박인 데이터에서 날짜만 추출함
  - `seoul.item.rain[seoul.item.rain$item.name %in% c("호박"),]$mean.price` : 품목 이름이 호박인 데이터에서 가격만 추출
  - `seoul.item.rain[seoul.item.rain$item.name %in% c("상추"),]$date` : 품목 이름이 상추인 데이터에서 날짜만 추출함
  - `seoul.item.rain[seoul.item.rain$item.name %in% c("상추"),]$mean.price` : 품목 이름이 상추인 데이터에서 가격만 추출함

빅데이터 분석 : 농산물편

2015 년 12 월 인쇄
2016 년 1 월 발행

발 행 처 한국정보화진흥원 K-ICT 빅데이터센터
집　　필 김성현, 최대우, 강기훈, 정석오, 이태욱,
이석호, 양성준, 김이환, 신은비
주　　소 대구광역시 동구 첨단로 53
연 락 처 (053) 230-1400
ISBN : 978-89-8483-236-7

빅데이터 분석
: 농산물편

NIA 한국정보화진흥원
NATIONAL INFORMATION SOCIETY AGENCY

(41068) 대구광역시 동구 첨단로 한국정보화진흥원
연락처 (053) 230-1114
www.nia.or.kr

비매품
93000
9 788984 832367
ISBN 978-89-8483-236-7