R을 이용한 부동산 데이터 분석 케이스 스터디

전희원 : freesearch.pe.kr

2016년 1월 30일

차 례

들어가며

우리는 매일 의사결정을 하면서 살고 있다. “점심에 뭘 먹지?”, “저녁 운동이 좋을까 아침 운동이 좋을까?”, “소개팅 여성분에게 애프터 신청을 할까 말까?” 등등 의식적 혹은 무의식적으로 많은 의사결정을 내리며 살아가고 있다. 그러한 의사결정들 중에서 인생에 큰 영향을 끼치는 의사결정은 결혼, 첫 주택 구매 등이 될 것이며, 특히나 인생에서 가장 큰 지름중에 하나인 주택 구매는 (아직까지는) 주거와 투자의 개념이 섞여 있는 현 주택시장을 볼때 매우 많은 고민을 안겨주는 결정중에 하나이다. 이러한 크고작은 의사결정을 매우 효과적으로 한다는건 복잡한 현실을 비춰볼때 어려운것 중에 하나임에는 틀림없으며, 그러한 의사결정에 가장 큰 영향을 끼치곤 하는 개인의 경험, 선입견과 같은 주과적인 측면들은 의사결정을 더욱 어렵게 하는 중요한 요인중에 하나이다. 그러나 의사결정에 영향을 끼치는 주관적인 측면이 항상 쓸모없는 것은 아니다. 문제는 정확한 객관적인 정보들이 없다는것이 문제이고, 이러한 객관적인 정보들이 주관적인 측면과 적절히 융합되어 있을때 매우 효과적인 의사결정을 할 수 있을 것이다. 필자는 이 글에서 인생의 가장 큰 의사결정중에 하나인 주택 구매와 관련한 객관적인 정보를 만들어 볼 것이며, 의사결정을 위한 학문인 통계를 이용해 이 과정을 서술하도록 하겠다. 그 과정은 오픈소스 R을 기반으로 수행하여, 다양한 패키지를 기반으로 아주 효과적인 분석을 수행할 수 있음을 보여주도록 하겠다.

이 글의 주된 목적은 부동산 매매 데이터를 기반으로 정량적인 분석을 통해 R기반 데이터 분석에 대한 이해를 도모하고 동기를 부여하기 위함이다. 더 나아가 독자분들이 이 글을 발판삼아 나름대로 각자 필요에 의한 자신만의 분석을 해볼 수 있으면 필자로서 큰 보람을 느낄거라 생각한다.

분석 데이터 수집과 데이터 얼개 살펴보기

분석을 위해 사용할 원천 데이터는 국토교통부 실거래가[1] 홈페이지에서 획득가능하며, 이 데이터를 수집하고 정리하는 코드는 아래와 같다.

참고로 웹 데이터를 수집하는 코드는 웹 페이지의 구조가 변경되면 반드시 바뀌어야 될 수 밖에 없는 운명을 가진 코드이다. 따라서 실거래가 홈페이지 자료제공 방식이 바뀌거나 혹은 조그마한 개편이라도 되면 코드를 자료제공 레이아웃에 맞게 바꿔야 된다는 것을 기억하길 바란다. `result_sales_dt.RData` 데이터 파일은 실습을 위해 GitHub[2] 링크에서 제공하고 있으니 코드를 실행하거나 이해해야 되는 부담은 일단 떨치길 바란다.

데이터 수집 코드가 동작하는 방식은 매매정보가 포함된 엑셀 파일이 존재하는 URL에 대해서 어떠한 순차적인 패턴이 있을거라는 판단을 하고 이 순차 패턴을 쭉 따라가 엑셀파일을 다운로드 받는 것이다. 이렇게 받은 엑셀 파일들은 분석 목적에 맞게 정리되 `data.table`객체로 저장되게 된다.

```
#웹 데이터 수집과 저장을 위한 패키지
library(rvest)
#rvest를 구성하는 저수준 API를 저장하고 있으며
#rvest에 대한 자세한 핸들링을 하기 위해 사용한다.
library(httr)
#문자열 처리에 대한 패키지로
library(stringi)
#국토교통부에서 원 데이터를 엑셀로 제공하기 때문에
#엑셀 데이터를 읽어들이기 위해서 필요함
library(XLConnect)
# data.frame보다 편리하고 효율적인 테이블 핸들링 패키지
library(data.table)



## 엑셀 데이터 다운로딩 부분(데이터는 working dir아래 data 디렉토리에 쌓인다.)
su <- file("succ.txt", "w")

#흡사 웹브라우저처럼 웹서버에서 인식하게 한다.
agent_nm <- paste0("Mozilla/5.0 (Macintosh;",
                   "Intel Mac OS X 10.10; rv:35.0) Gecko/20100101 Firefox/35.0")

#게시판 번호의 최대값을 가져온다.
maxidx <- html('http://rt.molit.go.kr/rtFile.do?cmd=list') %>%
 html_nodes('.notiWhite1 .notiBorad01') %>%
```

[1] http://rt.molit.go.kr

[2] https://github.com/haven-jeon/korea_real_estate_analysis

```
 html_text %>% as.numeric %>% max

#가져온 게시판 번호를 이용해 전체 실거래가 페이지를 방문해 파일을 다운로드한다.
for(i in maxidx:1){
  urls_view <- sprintf("http://rt.molit.go.kr/rtFile.do?cmd=view&seqNo=%d", i)
  r <- GET(urls_view,
           user_agent(agent_nm))
  htxt <- html(r, "text")
  html_nodes(htxt, "td.notiBorad14")[[1]] %>%
    html_text()
  #만일 페이지에 아래와 같은 텍스트가 존재하면
  #다음 페이지를 수집한다.
  if((html_nodes(htxt, "td.notiBorad14")[[2]]  %>%
      html_text()  %>% stri_trim_both) ==
       '첨부파일이 존재하지 않습니다.') next

  download_tags <- html_nodes(htxt, "td.notiBorad14")[[2]] %>%
    html_nodes('a[href^="javascript:jsDown"]')
  #페이지 내에 있는 다운로드 태그를 순회하며 태그 이름(파일명)과
  #링크 정보(파일 다운로드 링크)를 추출해 각각 저장한다.
  for(dtag in download_tags){

    dtag %>% html_attr("href") %>%
      stri_match_all_regex(pattern="javascript:jsDown\\('([0-9]+)','([0-9]+)'\\);") %>%
      .[[1]]  %>%
          {
            f_idx <<- .[2] %>% as.numeric
            s_idx <<- .[3] %>% as.numeric
          }

    f_nm <- dtag %>% html_text

    urls <- sprintf(paste0("http://rt.molit.go.kr/",
                           "rtFile.do?cmd=fileDownload&seq_no=%d&file_seq_no=%d"),
                    f_idx,s_idx)
    r <- GET(urls,
             user_agent(agent_nm))
    bin <- content(r, "raw")
```

```
    #1kb 미만의 데이터는 버림(에러 페이지?)
    if(length(bin) < 1000) next
    writeBin(bin, sprintf("data/%s",f_nm))
    cat(sprintf("%d, %d\n", f_idx,s_idx), file = su)
    print(sprintf("%d, %d", f_idx,s_idx))
  }
}

close(su)




## 엑셀 데이터에서 테이블을 추출해 하나의 아파트 매매 데이터로 통합하는 코드
## 연립 다세대, 단독 다가 데이터도 간단한 코드 변환으로 통합할 수 있다.


f_list <- list.files('data') %>%  stri_trans_nfc  %>% .[stri_detect_fixed(.,'매매아파트')]

total_list <- list()

#cnts <- 0

for(xlsf in f_list){
  wb <- loadWorkbook(paste0('data/',xlsf))
  sells <- list()
  fname <- stri_replace_last_fixed(xlsf, '.xls',"")
  yyyymm <- substring(fname, 1, 6)
  typenm <- substring(fname, 9)
  for(nm in getSheets(wb)) {
    df <- data.table(readWorksheet(wb, sheet = nm, header = TRUE))
    df[,`:=`(region=nm, yyyymm=yyyymm, typenm=typenm)]
    df[,`거래금액.만원.`:= stri_replace_all_fixed(`거래금액.만원.`, ',', '')]
    sells[[nm]] <- df
  }
  total_list[[paste0(yyyymm,typenm)]] <- rbindlist(sells)
  #cnts <- cnts + 1
  #if(cnts > 10) break
```

```
}

result_sales_dt <- rbindlist(total_list)



setnames(result_sales_dt, 1:10,
         c('si_gun_gu', 'm_bun', 's_bun', 'dangi', 'area',
           'cont_date', 'price', 'floor', 'year_of_construct', 'road_nm'))

result_sales_dt[,price:=as.numeric(price)]
result_sales_dt[,floor:=as.numeric(floor)]
result_sales_dt[,mm:=substr(yyyymm, 5,6)]
result_sales_dt[between(as.numeric(mm), 1, 3), qrt:='Q1']
result_sales_dt[between(as.numeric(mm), 4, 6), qrt:='Q2']
result_sales_dt[between(as.numeric(mm), 7, 9), qrt:='Q3']
result_sales_dt[between(as.numeric(mm), 10, 12), qrt:='Q4']
result_sales_dt[,yyyyqrt:=paste0(substr(yyyymm, 1,4), qrt)]
result_sales_dt[,yyyy:=factor(substr(yyyymm, 1,4))]
result_sales_dt[,yyyyqrt:=factor(yyyyqrt)]



#결과 데이터 저장
save(result_sales_dt, file='result_sales_dt.RData')
```

앞으로 코드를 이해하기 위해 필요한 패키지 설명을 간단하게 하겠다.

- `data.table` : `data.frame`과 같은 데이터 저장 클래스를 공유하며, `data.frame`에 비해 수십에서 수백배 빠른 데이터 처리 능력을 보여주며 간단한 코드로 다양한 데이터 전처리를 할 수 있게 한다. 오백만건 이상의 레코드의 데이터를 사용해야 되기 때문에 해당 패키지를 사용했음.
- `dplyr` : 이 글에서는 `data.table`과 함께 사용이 되며 파이프 연산자를 통해 코드의 가독성을 높여주고, `data.table` 혹은 `data.frame`이든 원본 소스에 상관없이 같은 전처리 코드로 다양한 소스 데이터를 다룰 수 있게 해준다.
- `ggplot2` : 대표적인 R기반 시각화 도구
- `lubridate` : R에서 다소 복잡한 시간에 관련된 데이터를 쉽게 다룰 수 있게 해주는 패키지
- `stringr` : 문자열 처리를 위한 패키지

이상의 패키지들이 대표적인 시각화 및 데이터 전처리 패키지들이다. 이정도 패키지들은 손에 익혀 두어야 어떠한 데이터든 빠르게 정리할 수 있다.

- forecast : 시계열 분석을 위한 패키지
- randtests : 램덤성을 검정하는 패키지

필자가 분석에 사용한 머신은 16GB의 메인 메모리를 가지고 있는 맥북 프로이다. 최대 약 5백만건의 데이터를 로딩해야 되기 때문에 메인 메모리 8GB 이상의 머신에서 실습하실걸 추천드린다. 참고로 수집한 5백만건의 매매 데이터의 메모리 로딩 크기는 약 633Mb이다.

```
# 수집한 매매 데이터 로딩
load('result_sales_dt.RData')
```

데이터가 주어지면 전체적으로 어떻게 구성이 되어 있는지 확인이 필요하다. 많은 경우 head 명령어나 str 명령어를 주로 사용하나 필자는 dplyr의 glimpse 명령어를 주로 사용한다.

```
glimpse(result_sales_dt, width=60)
```

```
## Observations: 5,289,885
## Variables: 17
## $ si_gun_gu          (chr) " 서울특별시 강남구 개포동", " 서울특별시 강남구...
## $ m_bun              (dbl) 12, 12, 12, 12, 12, 12, 12, 1...
## $ s_bun              (dbl) 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ dangi              (chr) "대청", "대청", "대청", "대청", "대청",...
## $ area               (dbl) 39.53, 39.53, 39.53, 51.12, 6...
## $ cont_date          (chr) "21~31", "1~10", "11~20", "11...
## $ price              (dbl) 22300, 24800, 23720, 33700, 4...
## $ floor              (dbl) 15, 5, 15, 8, 14, 7, 9, 1, 5,...
## $ year_of_construct (dbl) 1992, 1992, 1992, 1992, 1992,...
## $ road_nm            (chr) "개포로109길", "개포로109길", "개포로109...
## $ region             (chr) "서울", "서울", "서울", "서울", "서울",...
## $ yyyymm             (chr) "200601", "200601", "200601",...
## $ typenm             (chr) "아파트", "아파트", "아파트", "아파트", "...
## $ mm                 (chr) "01", "01", "01", "01", "01",...
## $ qrt                (chr) "Q1", "Q1", "Q1", "Q1", "Q1",...
## $ yyyyqrt            (fctr) 2006Q1, 2006Q1, 2006Q1, 2006...
## $ yyyy               (fctr) 2006, 2006, 2006, 2006, 2006...
```

si_gun_gu는 매매가 일어날 시군을 의미하며, m_bun, s_bun은 번지를 의미한다. area는 m^2 단위의 면적을 의미한다. cont_date는 계약일, price는 만원단위의 매매가격이다. road_nm은 도로명 주소, region은 지역, yyyymm 매매 년월을 의미한다.

관심을 두는 문제는 아래와 같은 질문에 대한 나름의 답을 구하는 것이다.

1. 아파트 매매 추이는 어떻게 되는가? 그리고 매매량 예측이 가능한가?

일단 분석전 일반적인 예상을 해보자면 부동산이라는건 가격을 결정하고 수요를 결정하는 많은 외부 요인들이 많기 때문에 예측이 어려울 것이라는 생각을 해본다. 하지만 그 불확실한 정보량이 어느정도 되는지 가늠해 보는것도 의미가 있을 것이라 생각한다.

매매 추이

ggplot2로 간단하게 분기별 아파트 매매건수를 시각화 해보겠다. 2015년 2분기는 아직 완전한 데이터가 수집된 상황이 아니므로 제거한다.

첫번째 라인이 처음 등장하는 data.table 문법인데, 아주 간단하게 data.table 문법을 SQL의 문(statement)으로 설명하자면 아래와 같다.

data.table명[where, select, group by]

SQL의 where절에 해당하는 곳은 데이터를 조건에 맞게 필터링 하는 곳이며, select는 어떠한 필드를 보여줄지 선택하는 곳이고, group by문은 어떠한 기준으로 데이터를 요약해서 보여줄지 결정하는 곳이다.

따라서 아래 구문에서는 쿼터별로 매매수(.N는 group by 조건에 해당되는 레코드 수를 리턴하는 함수)를 카운팅 해서 qrt_cnts라는 이름의 data.table객체를 만들게 된다. data.table객체는 data.frame객체를 입력받는 ggplot()과 같은 함수에 그대로 적용이 가능해서 별도의 변환작업 없이 활용이 가능하다는게 가장 큰 장점중에 하나이다.

ggplot2 패키지의 시각화 방식은 데이터와 그래프로 표현되는 미적(aesthetic)객체를 어떻게 매핑시키는지를 서술하는게 가장 기본이다. 그림1의 그래프의 경우 X축에 data.table 객체의 쿼터컬럼(yyyyqrt), Y축에 쿼터별 매매횟수(N)을 매핑 시키고 보여줄 시계열은 1종류라는것을 group 파라메터로 명시해 준다. 이런 매핑 정보를 기반으로 geom_point 함수나, 이후 + 연산자로 추가되는 모든 레이어관련 함수들이 하나의 그래프를 그리기 위해 동작하게 된다. 필자의 경우 이런 미적객체들의 정보를 추가 레이어들에 상속된다라고 설명을 하곤한다.

물론 각 레이어에서 별도의 미적매핑을 사용할 수 있는데, 이런 미적매핑은 해당 레이어 에서만 유효하게 된다. theme 명령어는 각 축이나 레이블에 다양한 표현을 하기 위해서 제공되는 명령어로 X축의 레이블 텍스트가 겹치는 현상을 없애기 위해 명령어로 텍스트를 90도 회전해 표현하게 했다. stat_smooth의 경우 X,Y 변수간의 선형, 혹은 비선형적인 패턴을 시각화 하기 위해 주로 쓰이며, 여기서는 선형회귀 모형으로 피팅된 값을 뿌려주도록 했다.

좀더 자세한 설명은 필자가 온라인으로 오픈해둔 R기반 데이터 시각화[3]라는 책을 참고하길 바란다.

```
qrt_cnts <- result_sales_dt[yyyyqrt != '2015Q2',.N,yyyyqrt]


ggplot(qrt_cnts, aes(x=yyyyqrt, y=N,group=1)) +
```

[3] http://freesearch.pe.kr/archives/3891

```
  geom_line() + xlab("년도분기") + ylab("매매건수") +
  theme(axis.text.x=element_text(angle=90)) + stat_smooth(method='lm')
```

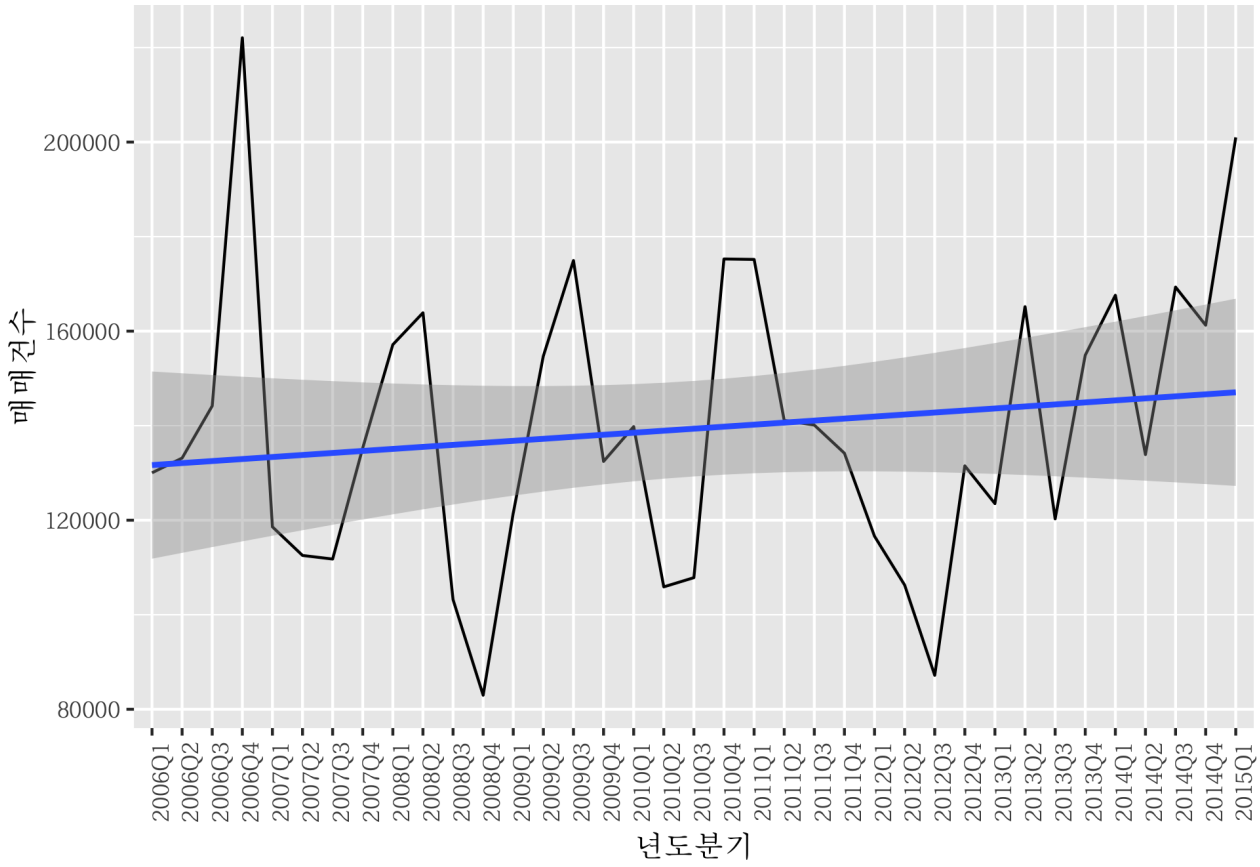

그림 1: 분기별 아파트 매매건수

그림1은 전체 추이를 분기별로 뿌려본 것이다. 부동산 활황기(2006년)에 엄청난 매매 건수 상승을 가져왔으며, 이후 크고 작은 매매건수 변동이 있었으나 추이에 큰 영향이 없다가 최근 2014년부터 점차 매매건수가 상승하는 추이를 보이고 있음을 알 수 있다. 물론 이런 추이가 어느 지역에서 발생하는지 확인해볼 필요가 있으니 지역별 추이를 시각화 해보도록 하자!

```
#group by 절에 region을 추가해서 쿼터별 지역별 매매량을 계산하게 함
region_cnts <- result_sales_dt[yyyyqrt != '2015Q2',.N,.(yyyyqrt,region)]

#지면 여건상 ` theme(axis.text.x = element_blank())`로 X 레이블을 제거했다.
ggplot(region_cnts, aes(yyyyqrt, N,group=region)) +
  geom_line() + facet_wrap(~region,scale='free_y', ncol=3) + stat_smooth(method = 'lm') +
  theme(axis.text.x = element_blank())
```

그림2는 지역별 분기별 아파트 매매량 추이에 대해서 간단하게 플로팅하고 간단한 선형모형을 피팅해 추세를

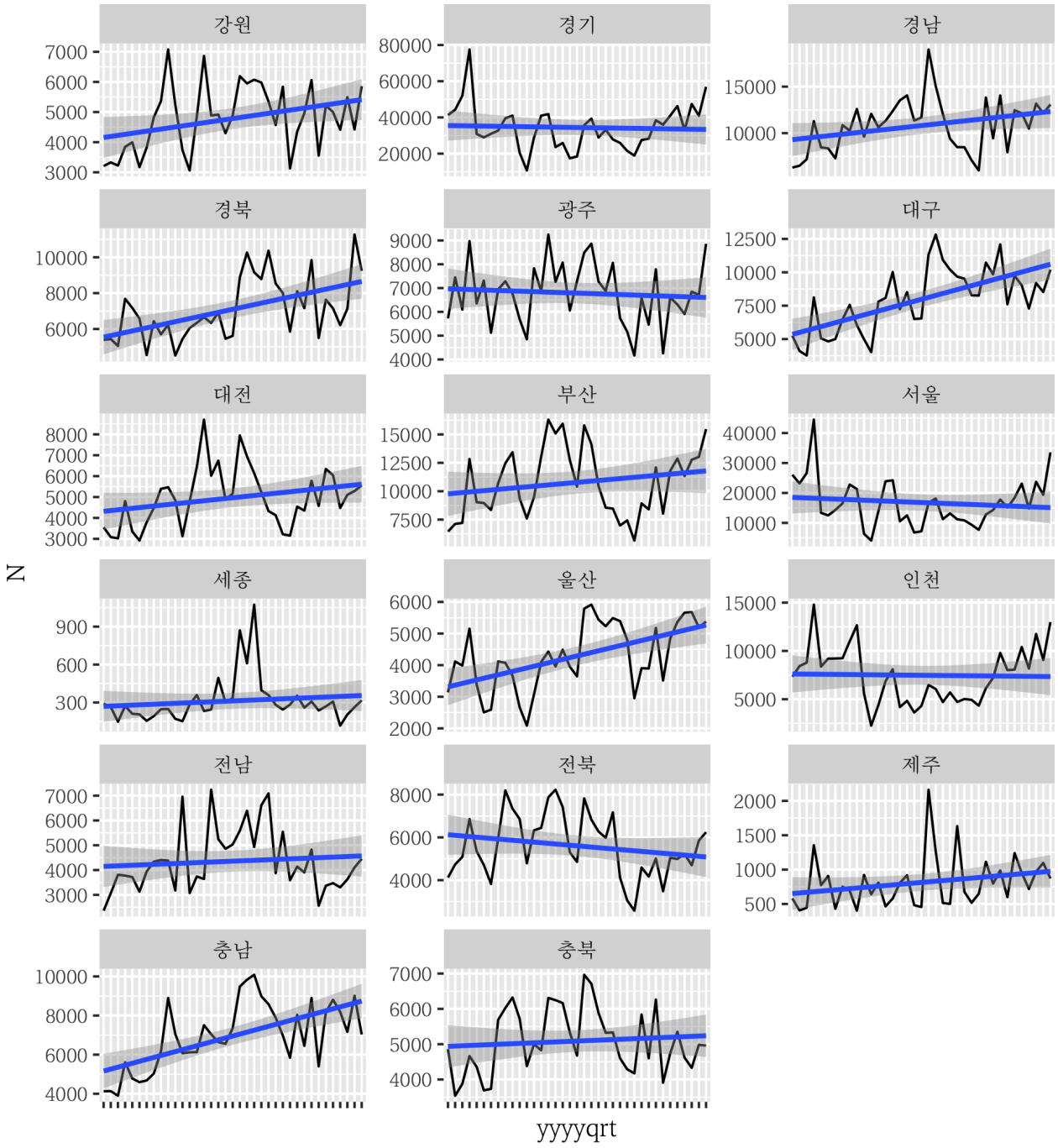

그림 2: 분기별 지역별 아파트 매매건수 추이

확인할 수 있도록 했다(Y축 척도가 그래프마다 다르다는 것을 주의하길 바란다). 눈으로 어림짐작해보면, 17개 지역중에서 47%인 약 8개 지역에서 매매량 상승 추이가 보이는 것을 볼 수 있다.

시계열에 패턴이 존재한다면 그 패턴이 미래에도 계속될 것이라는 가정을 기반으로 예측(forecasting)을 해볼 수 있을 것이다. 그렇다면 시계열에 패턴이 존재한다고 할 수 있다고 볼 수 있는가? 이를 검증하는 가장 간단한 방법은 눈으로 파악하는 것이고, 방법론적으로 가능성을 가늠해 보는 가장 간단한 방법은 시계열이 랜덤한지를 보는 랜덤성 검정을 통해서이다.

랜덤성을 검정하는 런 검정을 통해서 시계열이 랜덤한 시계열인지 간단하게 확인해 보도록 하자!

```
#월별 지역별 매매량
region_cnts <- result_sales_dt[,.N,.(yyyymm,region)]

#대표지역 추출
regions <- unique(region_cnts$region)



#각 지역별로 매매량의 랜덤성 검정 결과를 runs_p 변수에 추가
runs_p <- c()
for(reg in regions){
  runs_p <- c(runs_p, runs.test(region_cnts[region %chin% reg,N])$p.value)
}

ggplot(data.table(regions, runs_p), aes(x=regions, y=runs_p, group=1)) +
  geom_line() + geom_point() +
  ylab('P-value') + xlab('지역')
```

P-value의 의미는 년도별 지역별 아파트 매매량의 변동이 랜덤하다는 귀무가설이 참이라 가정할 때 관측값이 나올 확률을 의미한다. 그림3에서 보듯이 충북을 제외하고는 0.05미만의 유의미한 값이 나왔기 때문에 매매량의 패턴에는 랜덤하지 않은 뭔가 패턴이 있다는 것을 예상할 수 있다.

그렇다면 이중 서울 지역의 매매 추이 패턴을 시계열을 패턴 종류별로 분리해서 살펴보자!

```
seoul_cnts <- result_sales_dt[yyyymm != '201504' & region %chin% '서울',.N,.(yyyymm)]

tot_ts <- ts(seoul_cnts$N,start = c(2006,1), frequency = 12)

plot(stl(tot_ts,s.window = 'periodic'))
```

위에서 첫 data.table 문법에서는 2015년 4월을 제외하며 지역은 서울인 데이터셋에 대해서 월별(yyyymm)별 매매량(.N)을 카운팅해 seoul_cnts변수에 할당한다.

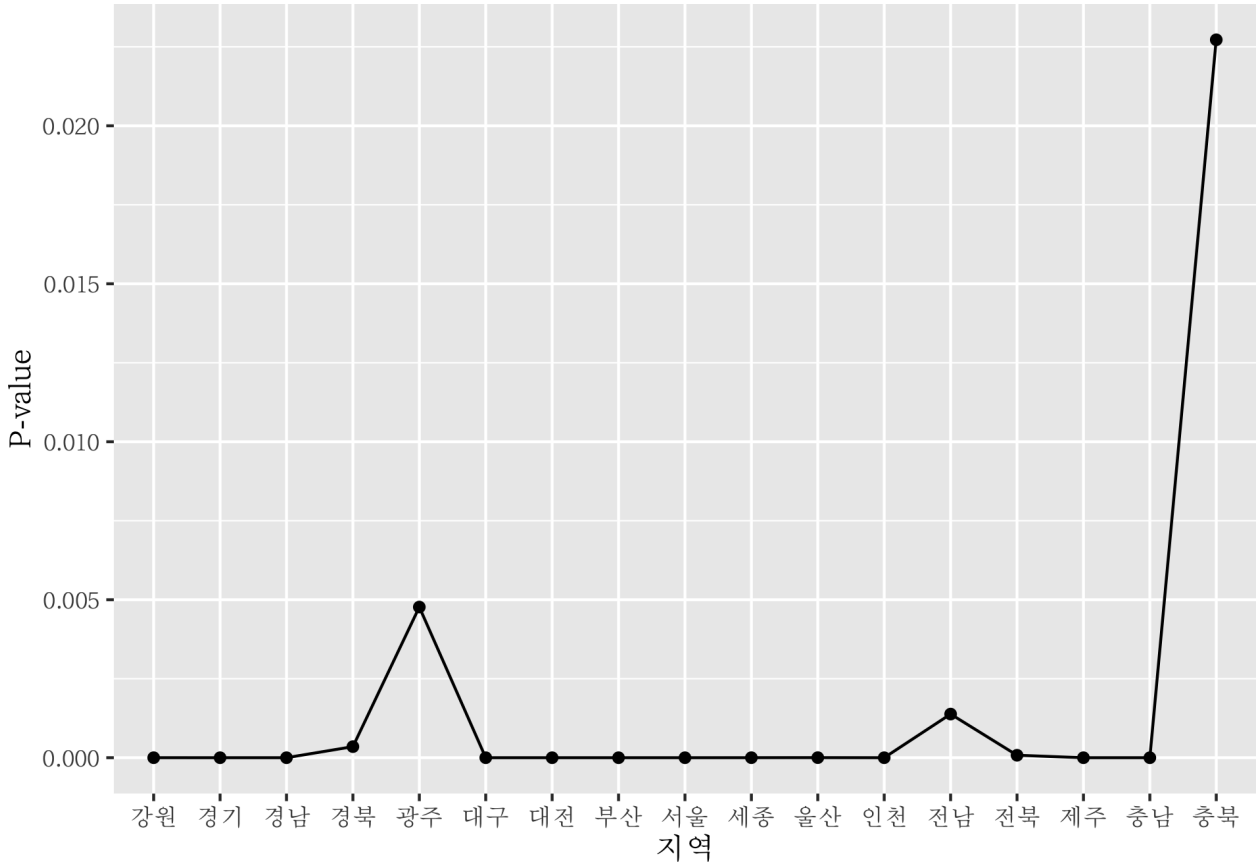

그림 3: 지역별 매매 추이의 램덤성 검정 시각화

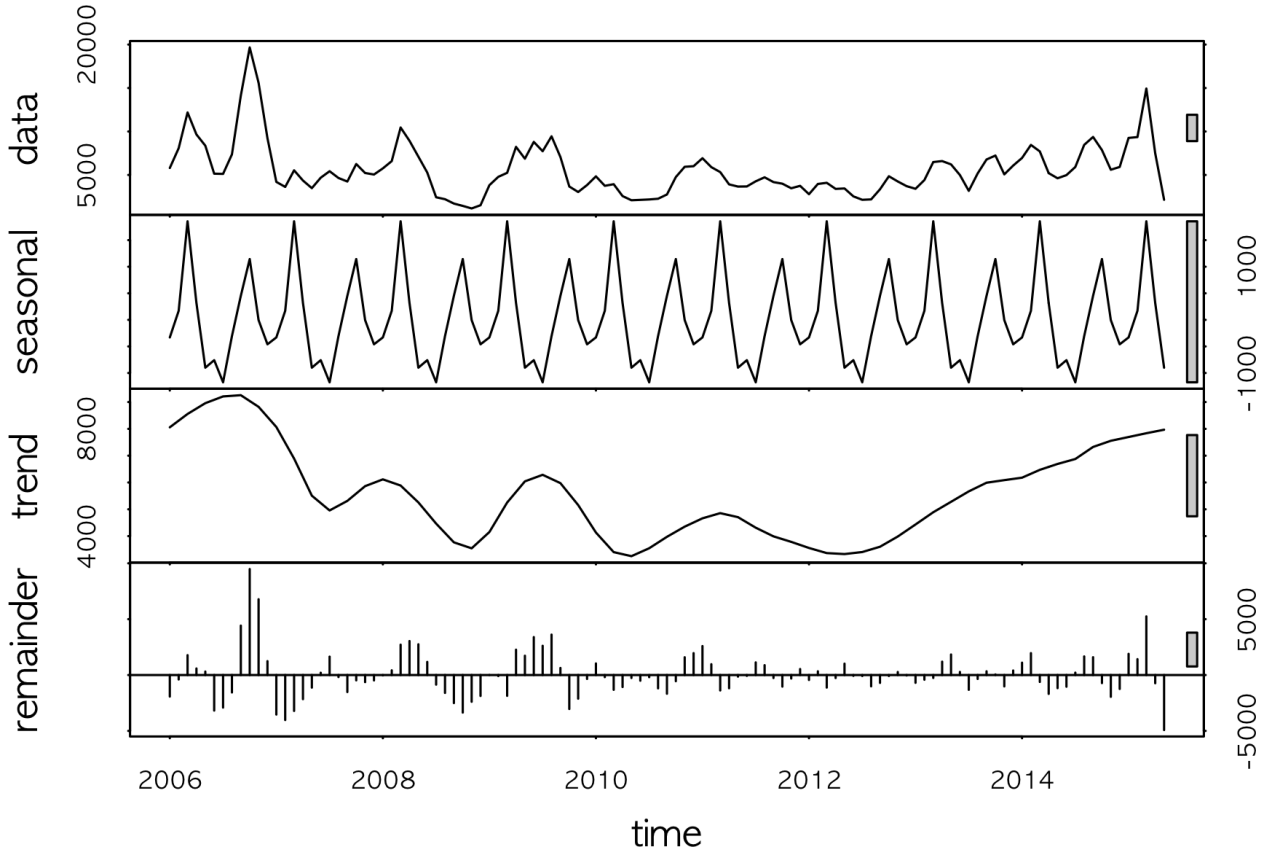

그림 4: 시계열 분할(decompose)

R에서 시계열 데이터를 다루기 위해서는 ts(time series)객체로 데이터를 변환해야 된다. 이를 위해 월별 매매량 데이터 벡터를 첫번째 변수로 입력하고 시작년월(2006,1)을 입력하고 년단위 12개월의 데이터라는 것을 명시한다.

시계열에는 대표적으로 세가지 패턴이 존재한다.

1. 트랜드(Trend) : 장기적으로 나타나는 변동 패턴
2. 시즈널(Seasonal) : 주,월,분기,반기 단위 등 우리가 알고 있는 시간의 주기로 나타나는 패턴
3. 주기(Cyclic) : 최소 2년 단위로 나타나는 고정된 기간이 아닌 장기적인 변동

위 세가지 중에서 1,2번을 시각화한 도표가 바로 그림4의 도표이다. 참고로 R에 `decompose`라는 `stl`함수와 비슷한 역할을 하는 함수가 존재하나 `stl`을 사용한 이유는 `decompose`가 이동평균에 의해서 시계열 분할을 하기 위해 시계열의 표현 기간이 줄어드는 특징을 가지고 있기 때문이다. 반면 `stl`의 경우는 `Loess`기반 비선형 추정을 하기때문에 표현기간이 줄어드는 현상이 없다.

그림4의 오른쪽 바는 각 시계열의 스케일을 맞추기 위해 표시한 것으로, 시즈널 패턴이 가장 변동량이 적다는 것을 보이고 있으며, 시즈널과 트랜드를 제외한 나머지(remainder) 변동(알수 없는)이 가장 크다는 것을 보여준다. 이것이 의미하는건 패턴기반 예측 모델링이 가능하나 에러가 매우 클 수 있다는 것을 알려주고 있다. 매매 추이만 가지고 예측을 하는건 에러가 매우 크며 좀더 정확한 예측을 위해서는 다른 정보가 필요하다는 것을 알 수 있다.

위에서 도출된 패턴을 기반으로 2015년 말까지의 서울지역 월단위 매매건수를 예측해 봤다.

```
arima_mdl <- auto.arima(tot_ts)


tsdiag(arima_mdl)
```

여기서 사용한 모델링 방법은 계절성(Seasonal) ARIMA 모형인데, 간단하게 과거의 관측값과 오차가 지금 현재의 시계열값을 결정한다는 ARMA모형에 불안정 시계열을 안정시계열로 만드는 I를 결합한 모형이다. 여기에 위의 개념을 계절적인 시차로 확장한게 계절성 ARIMA 모형이다. `AR(p)`모형의 p차수 `MA(q)`의 q차수 그리고 트랜드를 제거하여 안정시계열로 만들기 위한 `I(d)`의 차분 차수 d를 결정하기 위해 KPSS test[4], ACF, PACF를 그려 확인이 필요하나 필자의 경우 직접적으로 이런 과정을 거치는 것과 `auto.arima`를 사용해 자동으로 결정하게 하는것 사이의 차이를 가져오는 경우가 거의 없어서 자동으로 결정해주는 `auto.arima`를 사용해 모델링을 수행하였다.

`auto.arima`가 최적의 파라메터를 찾는 과정은 아래와 같다.

1. KPSS 검정을 통한 d 찾기
2. AIC를 최소로 하는 p,q 차수 순차탐색(stepwise search)

자동화된 모델링을 수행하기 때문에 이 함수는 대량의 예측을 자동으로 수행할 필요가 있을 때 주로 사용된다.

모델의 차수를 찾는과정도 중요하지만 생성된 모델이 추출 가능한 패턴들을 가능한한 모두 데이터로부터 추출을 하였는지 확인이 반드시 필요하다. 이는 모형이 모형이 초기에 가졌던 가정을 만족하는지 확인하는 과정으로 어찌보면 모델의 성능을 측정하는 것만큼 중요한 과정중에 하나이다.

[4] `https://en.wikipedia.org/wiki/KPSS_test`

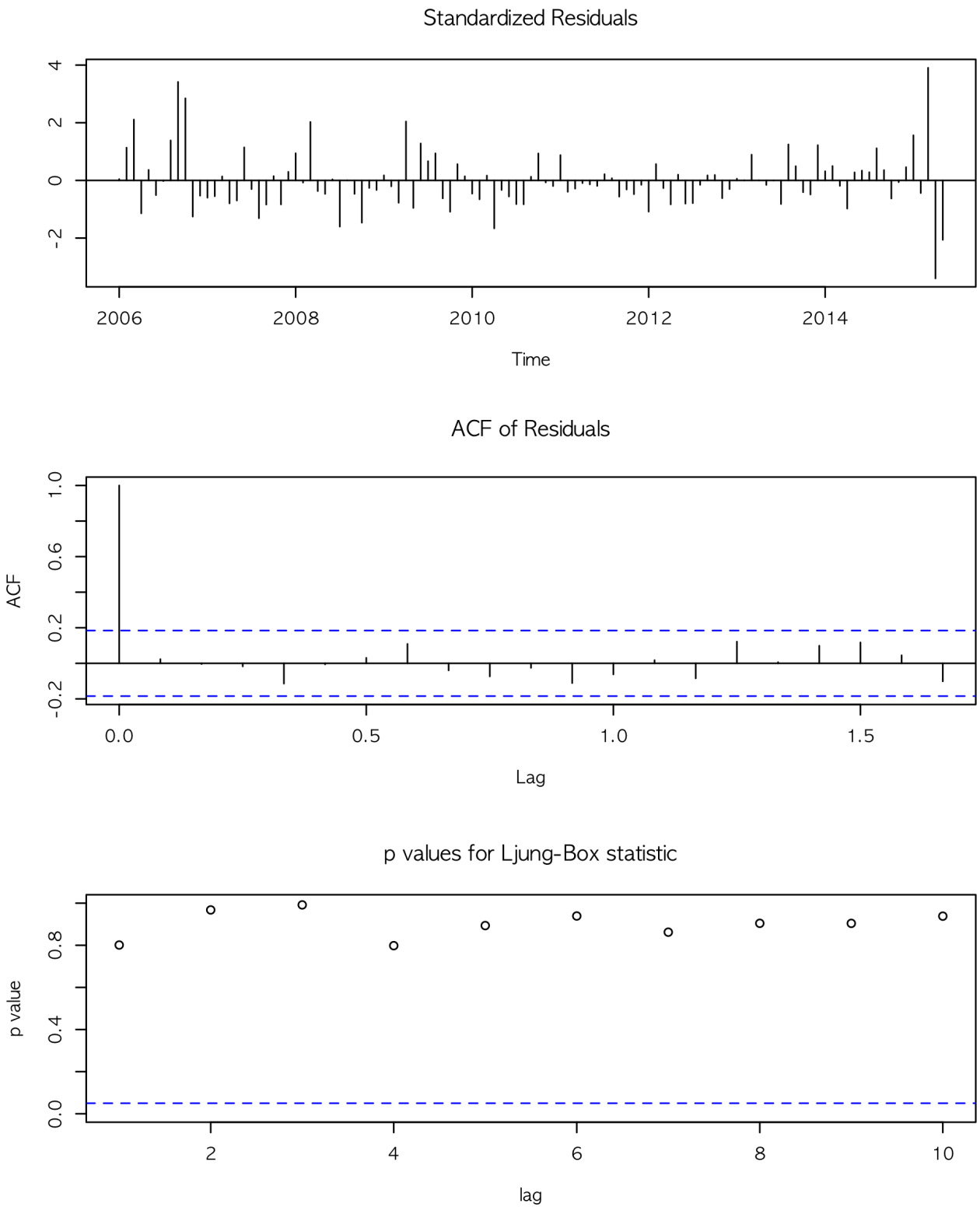

그림 5: ARIMA 모델 가정 만족 확인

그림5는 모형이 모형의 가정을 잘 만족하는지 보여주는데, 오차의 등분산성을 확인하기 위해 표준화 잔차(Standardized Residuals) 플롯을 살펴보니 가정을 대체적으로 잘 만족하는 것을 볼 수 있으며, 시차 상관을 살펴보기 위해 ACF, Ljung-Box를 살펴보면 10차 시차까지 시차 상관이 없다는 귀무가설을 기각하지 못해, 시차 상관이 없다고 볼 수 있다. 다만 지면 여건상 포함되지는 않았지만 오차의 정규성은 다소 위배되는 형태를 보이는데, 이때문에 예측구간, 신뢰구간을 기반으로 예측 결과를 해석할때 주의를 할 필요가 있음을 알 수 있다. 그러나 점추정은 상대적으로 의미가 있을 것으로 보인다.

```
#아래 명령어로 정확도등 다양한 지표가 확인 가능하다.
#accuracy(arima_mdl)
plot(forecast(arima_mdl,h=8))
```

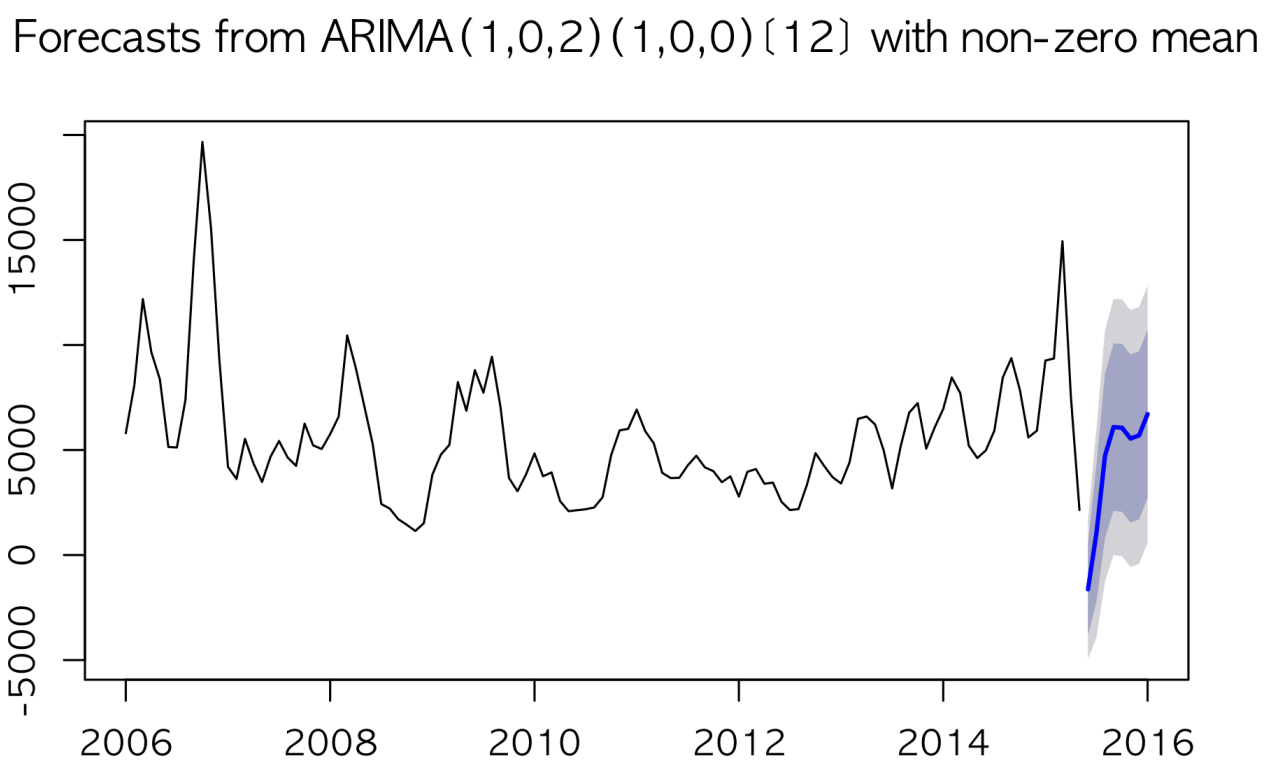

그림 6: 서울지역 아파트 매매량 예측(2015년 말까지)

그림6은 과거 시계열과 예측치들이 함께 그려진 그래프인데, 봄,가을에 거래가 늘어나는 계절적인 패턴과 더불어 2006년 이래로 상승과 하강을 반복하여 결국 추세적으로 큰 변동이 없는 그러한 패턴이 학습된 것으로 볼 수 있다. 이전 분석에서 이미 예상한 것처럼 평균 예측 에러율(MAPE[5])이 23%에 육박하여 예측력이 그다지 좋지는 못한 것을 알 수 있으나, 이곳에 추가하지는 못했지만 모형의 가정에 대해서는 모델이 잘 소화하고 있어서 시계열에서 추출되어야 되는 패턴들은 잘 추출이 된것으로 보이나 약 1/4 정도의 예측에 필요한 정보들은 이 시계열만으로는 채울 수 없는 오차인 것을 또한 확인할 수 있었다.

시계열 예측이라는 부분은 이미 알려진 패턴을 그 시계열에서 얼마나 잘 뽑아내느냐가 성패를 좌우한다. 그리고 이미 위에서 보는바와 같이 시계열적인 패턴은 대부분 뽑혀서 모델에 추가되었으나 안타깝게도 우리가 알지

[5] https://en.wikipedia.org/wiki/Mean_absolute_percentage_error

못하는 부동산에 영향을 미치는 외생변수의 존재가 매우 크다는 것도 어렴풋이나마 확인할 수 있었다. 하지만 모형의 에러가 크다고 쓰지 못할 모형은 아니다. 해당 모형은 최소 트랜드와 시즈널에 대한 패턴 추출은 아주 잘 하고 있어 예측의 출발점으로는 손색이 없다고 생각한다. 외생 변수로 부동산 매매 활성화에 도움이 되는 한국은행 기준 금리와 같은 변수가 추가 된다면 좀더 에러를 줄일 수 있지 않을까 하는 생각을 해본다.

전세가율 분석

전세가율이 높아지고 있다고 많은 미디어에서 이야기 하고 있다. 실제 전세가율이 시간이 지나면서 어떻게 변해가는지 데이터를 통해서 확인해보 필요가 있을것 같다.

이를 위해서는 국토교통부 실거래가사이트에서 전세 매매 관련 데이터를 가져와야 되는데, 위에서 설명한 수집 코드와 큰 차이가 없기 때문에 별도 설명은 하지 않을 것이다. 물론 분석에 추가적으로 필요한 데이터는 Github에 공개하도록 하겠다.

```
#전세 데이터 로딩
load('result_rents_dt.RData')
```

가져온 전세/월세 데이터의 얼개는 아래와 같다.

```
glimpse(result_rents_dt, width=60)
```

```
## Observations: 2,362,202
## Variables: 19
## $ si_gun_gu          (chr) " 서울특별시 강남구 개포동", " 서울특별시 강남구...
## $ m_bun              (dbl) 12, 12, 12, 12, 12, 12, 12, 1...
## $ s_bun              (dbl) 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ dangi              (chr) "대청", "대청", "대청", "대청", "대청",...
## $ type_of_rent       (chr) "월세", "전세", "전세", "전세", "전세",...
## $ area               (dbl) 39.53, 51.12, 60.00, 51.12, 5...
## $ cont_date          (chr) "21~31", "11~20", "21~31", "1...
## $ base_price         (dbl) 17000, 22000, 24000, 21000, 2...
## $ monthly_charge     (dbl) 15, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ floor              (dbl) 6, 4, 12, 2, 2, 5, 1, 1, 12, ...
## $ year_of_construct (dbl) 1992, 1992, 1992, 1992, 1992,...
## $ road_nm            (chr) "개포로109길", "개포로109길", "개포로109...
## $ region             (chr) "서울", "서울", "서울", "서울", "서울",...
## $ yyyymm             (chr) "201101", "201101", "201101",...
## $ typenm             (chr) "전월세아파트", "전월세아파트", "전월세아파트",...
## $ mm                 (chr) "01", "01", "01", "01", "01",...
```

```
## $ qrt                (chr) "Q1", "Q1", "Q1", "Q1", "Q1",...
## $ yyyyqrt            (fctr) 2011Q1, 2011Q1, 2011Q1, 2011...
## $ yyyy               (fctr) 2011, 2011, 2011, 2011, 2011...
```

type_of_rent 필드에서 전세인 경우만 사용할 것이며, base_price가 전세가가 된다. 나머지 필드는 매매 데이터와 비슷하니 별도 설명은 하지 않겠다.

두가지 데이터를 결합하기 위해서는 Join이라는 데이터 결합 방식을 사용해야 된다. 간단하게 설명하면 두 데이터셋에서 지정된 필드의 값이 같은 경우 지정된 필드들을 하나의 데이터로 결합하는 것을 의미한다.

조인(Join)을 수행할시 정확하게 하기 위해 주소, 년도/월, 층, 면적까지 맞추도록 하자!

참고로 전세 데이터는 2011년도 데이터부터 존재한다. 따라서 전세가율은 2011년도부터 확인 가능하다.

아래 코드를 확인해보면 데이터에 월세와 전세가 혼합되어 존재하기 때문에 전세데이터만을 추출하기 위해서 data.table의 where절에 필터 옵션으로 전세만 골라서 별도의 객체에 저장을 해두었다(1번 코드).

(2)코드를 살펴보면 다소 보기 어려웠던 %>%와 같은 연산자가 존재하는 것을 볼 수 있는데, 이 연산자는 리눅스의 파이프(|)연산자와 유사한 역할을 하는데, 이전 함수의 결과를 그대로 다음 함수의 첫번째 인자로 입력해주는 역할을 수행하는 직관적 문법(Syntactic sugar)을 제공한다. 이런 기법은 이전에 설명한 dplyr 패키지의 기본 동작 방식으로 다소 깔끔한 코드를 만들 수 있고 직관적인 코드 이해를 돕는데 큰 역할을 수행하기도 한다.

따라서 (2)코드는 (3)코드와 동일한 역할을 수행한다. 어떤게 더 직관적인 코드일까? 독자분들이 판단해서 더 직관적인 코드로 연습하면 될 문제이다.

```
rents <- result_rents_dt[type_of_rent %chin% '전세']  #--(1)


rent_sales <- rents %>%   # -- (2)
  inner_join(result_sales_dt,
    by=c('si_gun_gu', 'm_bun', 's_bun', 'dangi','area', 'yyyymm', 'floor')) %>%
  data.table   # data.table 객체로 반환한다.

# rent_sales <- data.table(     # ---(3)
#                 inner_join(rents,
#                   result_sales_dt,
#                   by=c('si_gun_gu', 'm_bun', 's_bun', 'dangi','area', 'yyyymm', 'floor')
#                   )
#                 )


ggplot(rent_sales[qrt.y %chin% 'Q1'] %>%
         sample_frac(0.5), aes(x=price, y=base_price, colour=yyyy.x)) +
```

```
  geom_point(alpha=0.7) +
  stat_smooth(method='gam', formula = y~ s(x, bs='cs'), size=1.7) +
  scale_color_brewer('yyyy',palette = "Set1") + xlab('매매가(만원)') + ylab('전세가(만원)')
```

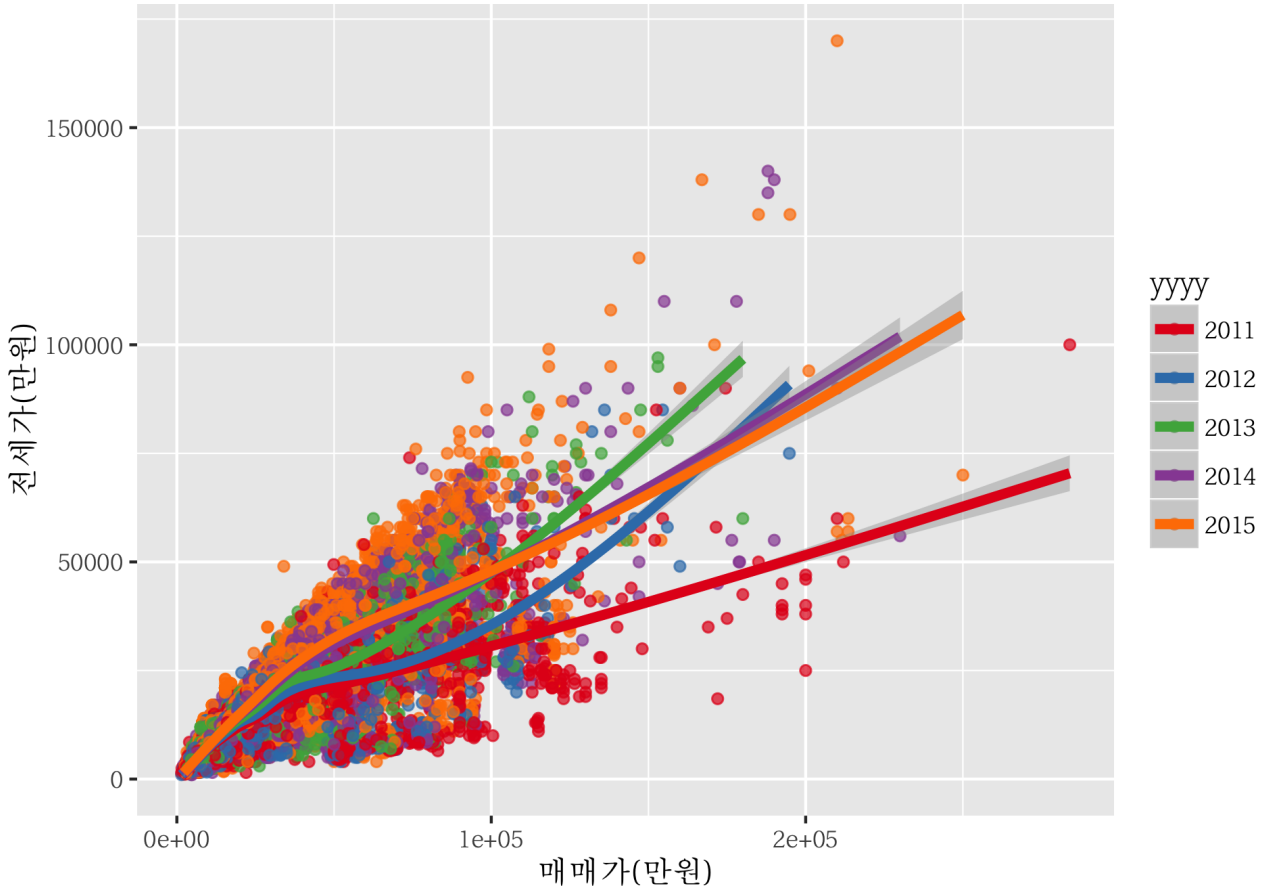

그림 7: 전국 매매가 대 전세가(1분기)

그림 7의 시각화를 위한 코드의 경우 `sample_frac` 함수를 사용해 다소 많은 데이터를 모두 시각화하지 않고 전체 50%인 약 5만건의 데이터만 시각화했다. 매매 가격에 따라 전세가의 변화를 좀더 자세히 보기 위해 선형 모형으로 피팅해 보기 보다는 비선형 모형으로 피팅해 이를 시각화된 결과와 함께 뿌려주었다. 계절적으로 매매가가 다른 패턴이 보이므로 매년의 `1분기(Q1)`만 시각화 해봤다.

보는바와 같이 매매가가 상대적으로 낮는 쪽에서는 2015년 전세가율이 2014년과 유사하고, 높은 쪽에서는 2013년과 유사해 2015년 전세가율이 높다는 것을 알 수 있다.

몇몇 미디어에서는 수도권의 전세가율에 대해서 이야기 하고 있는데, 같은 방식으로 경기지역만 그림8에서 보는바와 같이 확인해보면 경기지역의 2015년 전세가율이 사상 최대라는게 과장된 이야기는 아니라는 것을 알 수 있다.

아래 그림8은 전세가를 매매가로 나눈 전세가율 자체의 상자그림(boxplot) 보여준다. 상자그림은 박스를 해석하는게 중요한데, 전통적인 평균 분산을 매핑시켜 보면 좀 쉽게 접근이 가능하다. 상자의 중간에 있는 평행선을

중앙값(median)이라 하며 값을 순서대로 정렬해서 중간에 있는 값을 의미한다. 그리고 IQR은 25퍼센타일과 75퍼센타일의 차이를 의미하는데, 예를 들어 총 100개의 숫자를 정렬했을때 25번째 있는 값과 75번째 있는 값의 차이 구간을 나타낸다. 중앙값은 평균과 유사한 역할을 하며, IQR은 분산이나 표준편차와 같은 역할을 수행한다고 보면 된다. 정렬된 값을 기준으로 나타낸 통계량이기 때문에 이상치(outlier)에 좀더 강건하다고 알려져 있다. 좀더 상세한 내용을 알고 싶은 독자는 위키피디어[6]를 참고하길 바란다.

```
rent_sales_ratio <- rent_sales[,rent_ratio:=base_price/price]

ggplot(rent_sales_ratio, aes(x=region.x, y=rent_ratio)) +
  geom_boxplot(aes(fill=yyyy.x),outlier.size=0.5) +
  scale_y_continuous(breaks=seq(0,1.5, by=0.1), limits=c(0,1.5)) + xlab("지역") + ylab('전세가율') +
  scale_fill_discrete('년도')
```

```
## Warning: Removed 138 rows containing non-finite values (stat_boxplot).
```

위 코드에서 data.table의 select절 :=연산자는 새로운 전세가율(rent_ratio) 필드를 base_price, price를 이용해서 생성하라는 코드이다.

그림9 상자그림의 중앙값과 IQR을 참고해서 살펴보면 대부분 지역의 년도별 전세가율은 상승추세에 있는 것을 알 수 있다. 서울은 전세가율(중앙값)이 70%에 근접하고 있고, 경기도는 70%를 넘어가고 있는 것을 볼 수 있다. 2011년 제주도의 전세가율은 데이터에 뭔가 문제가 있는 것으로 보인다.

```
seouls_gu <- rent_sales[si_gun_gu %like% '서울특별시',
                        gu:=sapply(str_trim(si_gun_gu),
                              function(x){str_split_fixed(x,pattern = ' ', n=3)[1,2]})]

ggplot(seouls_gu, aes(gu, rent_ratio)) + geom_boxplot(aes(fill=yyyy.x),outlier.size=0.5) +
  scale_y_continuous(breaks=seq(0,1.2, by=0.1), limits=c(0,1.2)) + xlab("지역") +
  ylab('전세가율') + scale_fill_discrete('년도') + theme(axis.text.x=element_text(angle=45))
```

위 코드에서 구(gu)라는 새로운 필드를 만들기 위해 시군구(si_gun_gu)에 포함된 주소 정보를 기반으로 두번째에 구 정보가 포함되었다는 가정하에 첫번째 공백 다음에 나온 문자열을 gu 필드에 추가하는 코드이다. sapply함수로 구성한 이유는 str_split_fixed함수가 문자열 벡터를 입력받을때 matrix객체 값을 리턴하기 때문이다. 이는 새로운 필드에 이 객체를 할당하는게 맞지 않기 때문이며 벡터의 원소 하나하나를 처리해 별도로 리턴받게 하기 위해 sapply를 이용해 구성하였다. data.table의 경우 필드를 조합해 새로운 필드를 만들기 위해서는 같은 길이의 벡터를 리턴하는 함수를 쓰는게 원칙이기 때문이다.

그림10는 서울 지역내 구단위 전세가율에 대한 분포를 보여주는데, 예상했던 대로 전 구에서 전세가율이 상승 추이에 있다는 것을 볼 수 있으나, 강남, 강동구의 경우 각 년도별 분산이 크기 때문에 상승 추이에 있다고 보기

[6] http://en.wikipedia.org/wiki/Box_plot

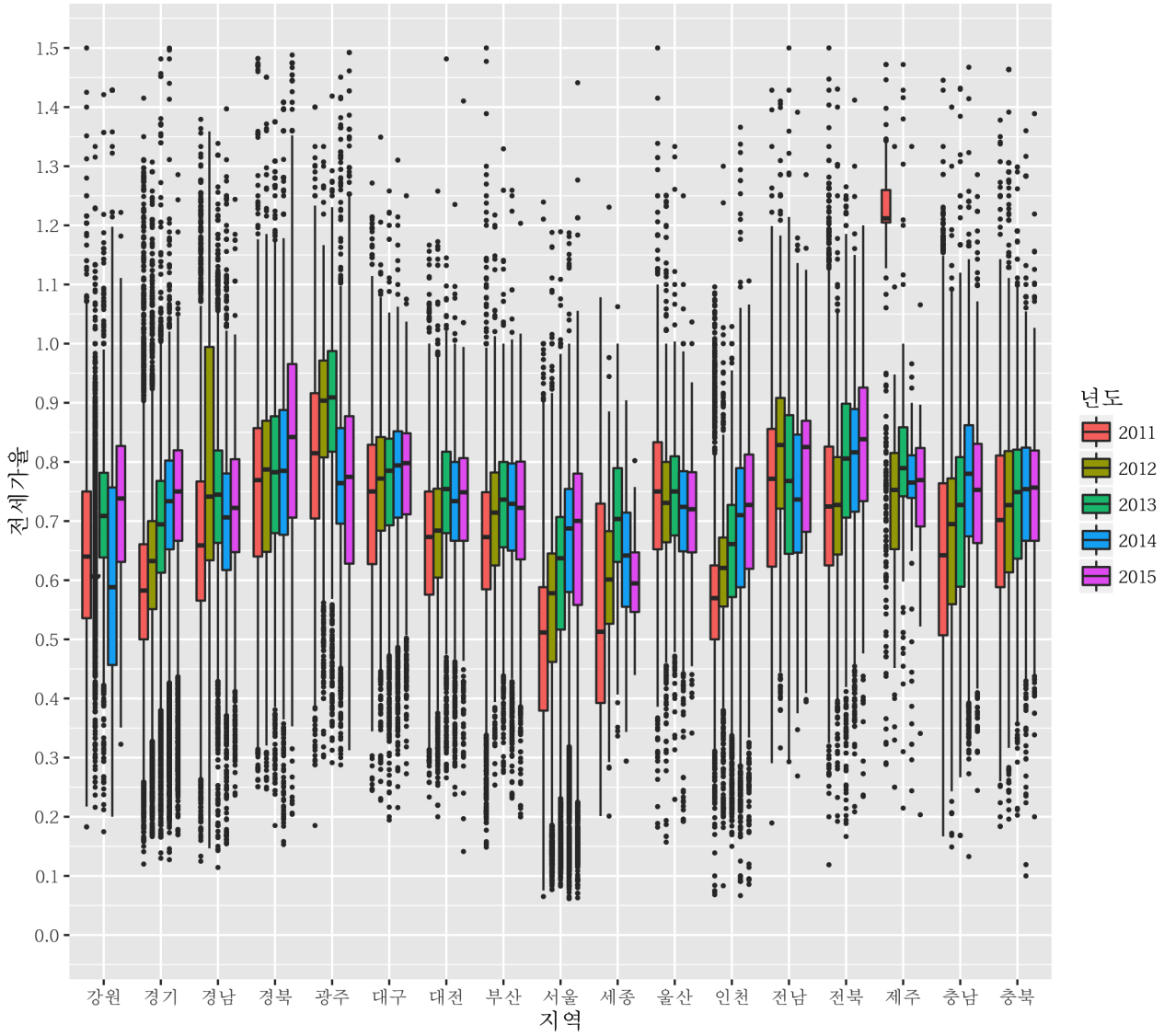

그림 8: 상자그림을 이용한 지역별 연도별 전세율

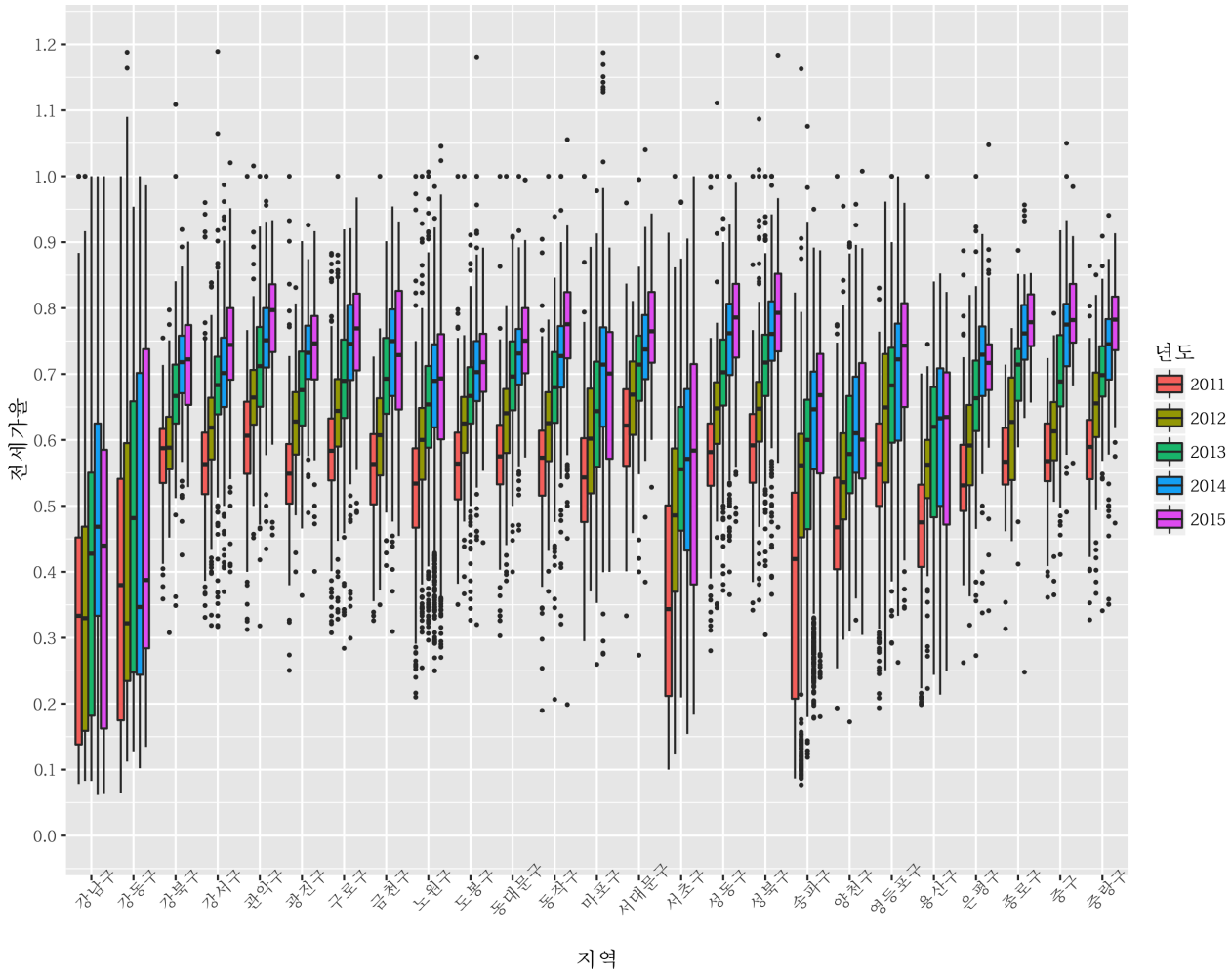

그림 9: 서울시 구별 전세가율 추이

힘들다. 전세가율의 변동이 크다는건 매매가의 변동이 커서 그런건지 전세가의 변동이 큰 이유인지 확인이 필요하지만 전세가로만 볼때 강남, 강동은 매우 변동이 큰 지역이라 볼 수 있다. 이 부분에 대해서는 강남의 대표적인 아파트 단지 한곳을 다음회에 상세하게 분석해 보면서 확인해볼 예정인데, 생각해 볼 수 있는 시나리오로는 학기초의 전세가와 학기중의 전세가의 차이가 커서 이런 현상이 일어날 수 있지 않을까 하는 예상을 해본다.

전세가율에 대해서 정리를 해보면 아래와 같다.

1. 전세가율은 2015년이 최고라고 이야기 할 수 있다.
2. 상승 추이는 수도권 이내가 가장 크다.
3. 서울시 전세가율 추이를 볼때 특징적으로 다른 몇개의 구가 존재하고 있다.

마치며

R을 이용해 웹 크롤링, 시각화, 데이터 전처리 그리고 모델링까지 다양한 작업을 할 수 있다는 것을 간략하게 부동산 매매 데이터를 이용해 살펴봤다. 필자는 수년간 R을 기반으로 하는 데이터 분석 실무를 통해 이 언어가 데이터 분석을 위한 최적의 도구라는건 이젠 의심할 수 없는 사실이라 자신있게 이야기 할 수 있을것 같다. 하지만 최근 R만 잘 하면 데이터 과학이 모두 잘 수행될 수 있을 거라는 오해를 하는 사람들을 자주 봐온다. 이런 분들의 공통점은 R을 다 배웠는데, 이걸로 뭘 하지?하는 반응이 대부분이다. R과 주어진 데이터로 적절한 질문을 하지 못하고, 또한 질문이 주어지더라도 질문에 대해서 답을 할 수 있는 통계적인 방법론에 익숙하지 않을 경우 R은 그냥 또 다른 프로그래밍 언어중에 하나로 전락하게 된다.

좋은 질문, 그리고 그에 합당한 통계적인 분석 방법 및 시각화 그리고 마지막으로 합당한 도구가 결합될 때 데이터 분석이라는 과정은 매우 의미있는 과정이 될 수 있다고 생각한다. 따라서 도구를 학습하는 것과 동시에 통계적인 분석 방법,시각화, 좋은 질문에 대한 고민 그리고 이를 검증할 수 있는 과학적인 문제 접근 방식에 대한 훈련도 게을리 하면 안될 것이다.