III. 시계열 Visualization

시계열 데이터는 일반적으로 가로축(X axis)에는 시간 인덱스가 위치하고 세로축(Y axis)에 값이 표현되는 라인 plot이 많이 사용된다. 라인 plot은 시간의 흐름에 따라 발생하는 추세(Trend), 계절성(Seasonality) 등의 시계열 특성을 표현하고 파악하는데 효율적이다. 시계열 데이터 분석에는 시계열 데이터의 직접적인 라인 그래프 외에도 자기 상관 함수(ACF : Autocorrelation Function), 편자기상관함수(PACF : Partial Autocorrelation Function) 등의 다양한 plot들이 사용되지만 이 장에서는 data.frame, ts, xts, tsibble등 시계열 객체 별로 plotting을 작성하는 방법을 설명한다.

# data.frame : ggplot2 패키지

ggplot2 패키지는 시계열 데이터 이외에 다양한 plot을 그리는 R의 대표적인 plot 패키지로 널리 사용되고 있다. ggplot2는 R-Studio의 수석 데이터 사이언티스트인 Hadley Wickham이 주도적으로 개발한 패키지로 2005년 발간된 Leland Wilkinson의 ’The Grammar of Graphics’을 토대로 개발되었다.

’The Grammar of Graphics’는 데이터를 효과적으로 표현하기 위해 다음과 같은 7가지 요소를 통해 표현할 것을 제안하였는데[[1]](#footnote-1) ggplot2의 plot를 그리는 요소로 사용하고 있다. ggplot2의 plot은 ggplot() 를 호출하면서 시작하는데 다음의 요소들을 + 기호를 사용하여 연결시켜가면서 전체 plot를 생성한다.

ggplot2의 plot 7 요소

1. data

* plot에서 표현해야 할 데이터를 지정함. 하나의 ggplot plot 안에는 하나 이상의 데이터를 포함시킬 수 있는데 기본적으로 하나의 데이터는 ggplot()에 포함되어야 함.
* 추가적인 데이터를 표현하기 위해서는 +로 연결되는 geom\_ 함수에 설정하여 사용할 수 있음.
* Ggplot은 데이터 프레임을 지원하는데 데이터프레임이 아닌 객체가 설정되면 내부적으로 데이터프레임으로 변환(fortify()를 활용)하여 사용함.

1. Aesthetics

* 데이터의 컬럼을 시각적 속성에 매핑 시키는 요소로써 Aesthetics는 ggplot()와 geom\_함수로 표현되는 개별 레이어에서 aes 매개변수를 이용하여 매핑 가능함.
* 매핑 가능한 시각적 속성은 X 축, Y 축, color, size 등임.

1. Geometry

* Geometry는 plot에서 실질적으로 표현되는 point, line, bar 등의 그래픽 요소를 설정함.
* 일반적으로 하나의 Geometries는 하나의 layer에 표현되는데 여러 Geometry layer가 겹쳐져서 하나의 plot을 생성하게 됨.

1. Facet

* plot에 표현되는 데이터가 일변량(univariate)이 아닌 다변량(multivariate)인 경우 하나의 plot에 다변량을 데이터가 표현되어 여러 Geometry가 겹쳐지면 데이터의 분석이 어려움.
* 이런 경우 다변량을 일변량화하여 일변량 plot을 여러개 표현할 때 사용하는 요소임.

1. Statistics

* mean, median 등 표현할 데이터의 통계값을 표현할 필요가 있을 때 사용하는 요소임.

1. Coordinates

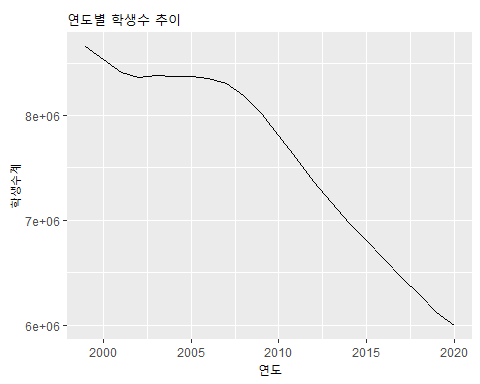
* plot에 표현되는 2차원 좌표계를 설정하기 위한 요소임. X 축과 Y 축의 limit, label 등을 설정할 수 있음.

1. Theme

* plot 제목, 축 제목, 축 단위, legend 등 plot의 전반적인 디자인을 꾸며줄 수 있음.
* 미리 정의된 Theme을 적용하여 설정할 수 있음.

ggplot으로 시계열 plot를 표현하기 위해서는 위의 7개 요소 중 Geometry는 line으로 사용하는 경우가 많다. Aesthetic의 X 축은 시간 인덱스로 주로 설정되는데 ggplot()를 사용하여 2장에서 저장한 students 데이터프레임을 시계열 plot으로 다음과 같이 그릴 수 있다.

library(ggplot2)  
students %>%  
 ggplot(aes(x = 연도, y = 학생수계)) +  
 geom\_line(aes(group = 1)) +   
 labs(title = '연도별 학생수 추이')

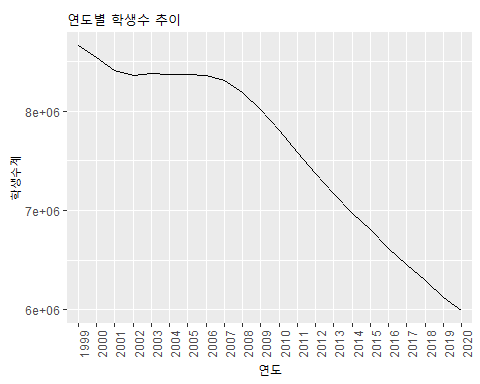


코드 설명

* ggplot를 사용하여 플로팅하기 위한 데이터프레임을 %>%으로 ggplot()에 전달.
* Aesthetic 요소를 설정하기 위해 aes() 를 사용함. x축에 바인딩 될 컬럼은 연도, y축에 바인딩 될 컬럼은 학생수로 설정(aes(x = 연도, y = 학생수계))
* geom\_line()을 사용하여 라인 geometry를 표현하는 layer를 생성. 라인 geometry는 여러 라인이 표기될 수 있으므로 같은 라인으로 표현될 데이터를 묶어주는데 그룹이 하나인 경우는 1로 설정(aes(group = 1)).
* labs()를 사용하여 plot 제목을 설정(title =)

위의 예제에서 X 축에 연도 컬럼을 연결했는데 students 데이터프레임에 연도가 다 나타나지 않는다. 연도 컬럼은 date 클래스이기 때문에 ggplot()에서 적절한 표기 범위를 잡는데 이를 1년 단위로 표기하기 위해서는 다음과 같이 그릴 수 있다.

ggplot(data = students, aes(x = as.factor(lubridate::year(연도)), y = 학생수계)) +  
 geom\_line(aes(group = 1)) +   
 theme(axis.text.x=element\_text(angle=90,hjust=1)) +   
 labs(title = '연도별 학생수 추이', x = '연도')

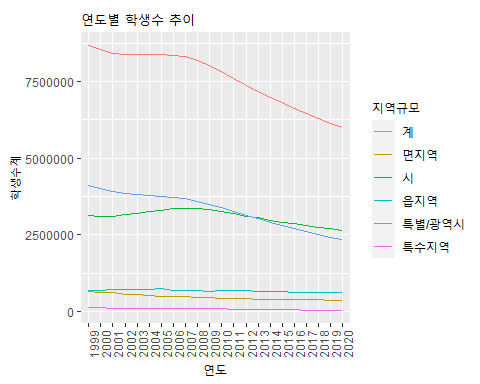


코드 설명

* 앞선 코드에서 사용한 파이프라인(%>%)을 사용하지 않으면 ggplot()에 직접 사용할 데이터프레임을 지정해야 함(data =).
* date 클래스로 년-월-일로 설정된 연도 컬럼에서 lubridate 패키지의 year()를 이용하여 연도만 추출(lubridate::year(연도))하고 이를 factor()를 사용하여 factor로 변환
* theme()을 사용하여 theme 요소를 추가. 추가된 요소는 X축에 표기되는 텍스트(axis.text.x)를 element\_text()를 사용하여 설정하는데 텍스트 방향을 90도 회전(angle=90)시키고 정렬을 맞춤(hjust=1)
* labs()를 사용하여 X축 제목을 추가로 설정(x =)

이번에는 다중 선 plot을 그려본다. 앞 장에서 생성했던 데이터 중에 students.all 데이터 프레임을 사용한다. 앞에서 설명했듯이 라인 geometry는 여러 그룹들을 같이 그릴 수 있는데 지역규모별로 plot을 생성하려면 아래와 같이 그릴 수 있다.

ggplot(data = students.all, aes(x = 연도, y = 학생수계)) +  
 geom\_line(aes(group = 지역규모, color = 지역규모)) +   
 theme(axis.text.x=element\_text(angle=90,hjust=1)) +   
 labs(title = '연도별 학생수 추이', x = '연도')

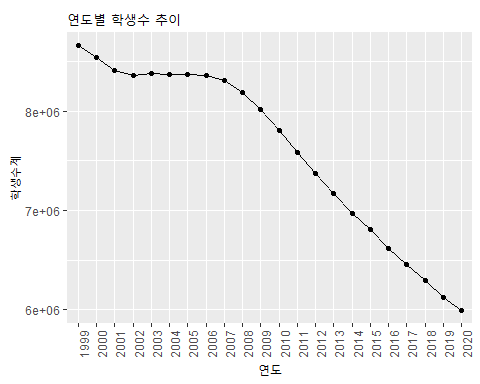


코드 설명

* geom\_line()을 사용하여 라인 Geometries Layer를 설정. 라인 Geometries로 표시할 다변량 라인 plot은 그룹핑이 가능한 컬럼인 지역규모를 group =로 설정하고 색상 변경을 설정(color =)

위에서 그려진 plot은 전체적인 흐름을 보기에는 편리하지만 연도별 데이터의 위치를 가늠하기에는 조금 어렵다. 각 연도에 점을 찍어주면 더욱 보기가 편리할 듯 하다. 이럴 때에는 점(point) Geometry layer를 추가하면 되는데 geom\_point()를 사용하면 된다.

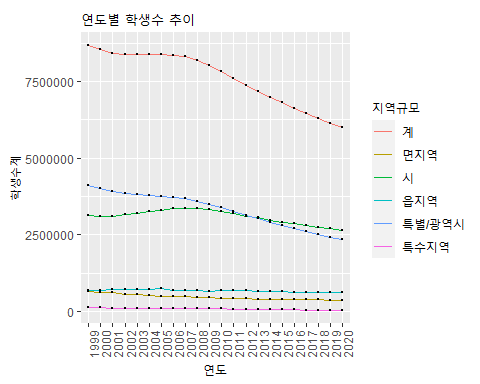
ggplot(data = students, aes(x = as.factor(lubridate::year(연도)), y = 학생수계)) +  
 geom\_line(aes(group = 1)) +   
 geom\_point(shape = 'circle') +   
 theme(axis.text.x=element\_text(angle=90,hjust=1)) +   
 labs(title = '연도별 학생수 추이', x = '연도')



코드 설명

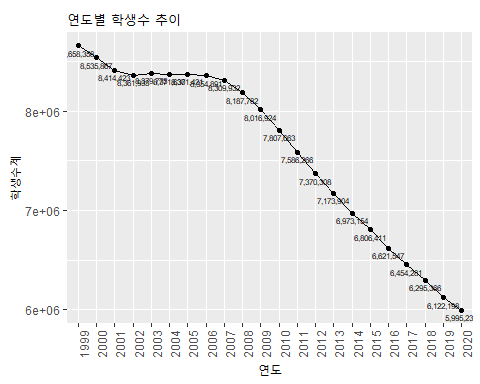
* geom\_point()를 사용하여 점 geometry layer를 추가. 점 모형은 circle로 설정(shape = 'circle')

ggplot(data = students.all, aes(x = 연도, y = 학생수계)) +  
 geom\_line(aes(group = 지역규모, color = 지역규모)) +   
 geom\_point(shape = 'circle', size = 0.5) +   
 theme(axis.text.x=element\_text(angle=90,hjust=1)) +   
 labs(title = '연도별 학생수 추이', x = '연도')



점을 표기하면 좀 더 데이터를 이해하기가 쉬워진다. 하지만 구체적인 값을 표기하면 더 이해하기가 좋아질 것이다. 다만 이 과정에서 조심해야할 것은 값을 표기하다 보면 숫자들이 너무 많아져 거꾸로 plot이 전체적으로 보기가 어려워 질 수 있다.

ggplot(data = students, aes(x = as.factor(lubridate::year(연도)), y = 학생수계)) +  
 geom\_line(aes(group = 1)) +   
 geom\_point(shape = 'circle') +   
 geom\_text(aes(label = scales::number(학생수계, big.mark = ',')), size = 2, vjust = 1.5) +   
 theme(axis.text.x=element\_text(angle=90,hjust=1)) +   
 labs(title = '연도별 학생수 추이', x = '연도')

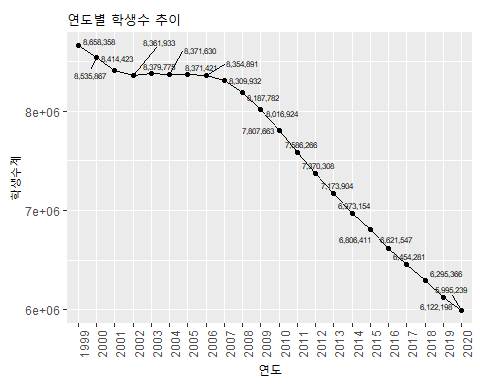


코드 설명

* geom\_text()를 사용하여 각 데이터 포인트에 데이터 값을 표기(aes(label =)). 표기되는 값은 scales 패키지의number()를 사용하여 숫자 형태로 표기하되 천 단위 콤마를 표기(big.mark = ',')하고 크기는 2(size = 2)로, 정렬은 수직 위치로 1.5(vjust = 1.5)만큼 이동

위의 plot에서도 2000년에서 2006년까지는 데이터 값을 보기가 어렵다. 이럴 때는 ggrepel 패키지의 geom\_text\_repel()을 이용하면 겹치지 않게 라벨을 표시할 수 있다.

library(ggrepel)  
ggplot(data = students, aes(x = as.factor(lubridate::year(연도)), y = 학생수계)) +  
 geom\_line(aes(group = 1)) +   
 geom\_point(shape = 'circle') +   
 geom\_text\_repel(aes(label = scales::number(학생수계, big.mark = ',')), size = 2, vjust = 1.5) +   
 theme(axis.text.x=element\_text(angle=90,hjust=1)) +   
 labs(title = '연도별 학생수 추이', x = '연도')

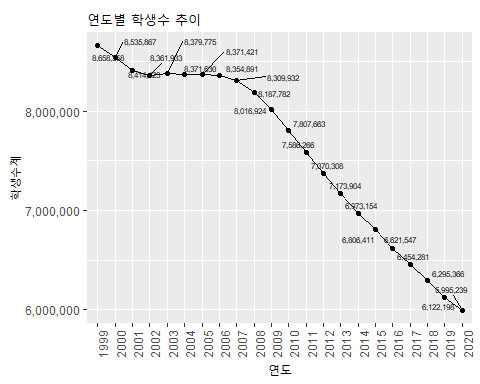


코드 설명

* ggrelpe 패키지의 geom\_text\_repel()을 사용하여 각 데이터 포인트에 데이터 값을 표기. 사용법은 geom\_text()와 동일.

ggplot()은 X 축과 Y 축의 표현 설정을 지정하지 않으면 자체적으로 판단하여 설정하는데 표현되는 값이 클 경우 지수 형태로 표기된다. 앞의 전체 학생수 plot이 지수 형태로 표기된 Y 축 라벨이 보이는데 이렇게 표현되면 값을 알아보기가 어렵다. 이 부분을 수정해보자.

ggplot(data = students, aes(x = as.factor(lubridate::year(연도)), y = 학생수계)) +  
 geom\_line(aes(group = 1)) +  
 geom\_point(shape = 'circle') +   
 geom\_text\_repel(aes(label = scales::number(학생수계, big.mark = ',')), size = 2, vjust = 1.5) +   
 theme(axis.text.x=element\_text(angle=90,hjust=1)) +   
 labs(title = '연도별 학생수 추이', x = '연도') +  
 scale\_y\_continuous(labels = scales::number\_format(big.mark = ','))

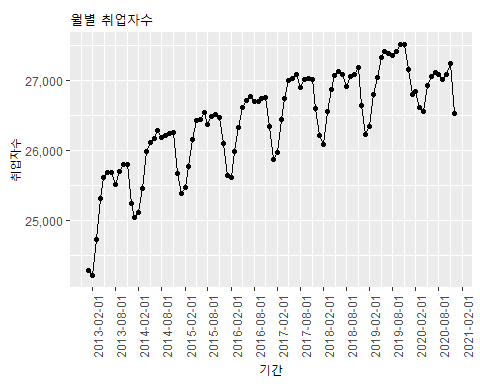


코드 설명

* scale\_y\_continuous()를 사용하여 Y 축 특성을 설정. 설정하는 특성은 라벨 특성으로 sacles 패키지의 number\_format()을 사용하는데 천 단위의 콤마를 찍어주도록 설정

ggplot이 아닌 다른 시계열 패키지에서 제공하는 plot은 ggplot에서 제공하는 것만큼 다양한 기능을 제공하지 않는다. 디자인이 중요한 plot을 작성하기 위해서는 ggplot의 다양한 기능을 사용하여 plot을 작성하는 것이 바람직하다. 이외에 ggplot에 관련한 자세한 사항은 여러 다른 책들에서 소개하고 있다. 아래의 코드는 전체 취업자 데이터와 코로나 발생 데이터를 ggplot()을 사용하여 그린 예제이다.

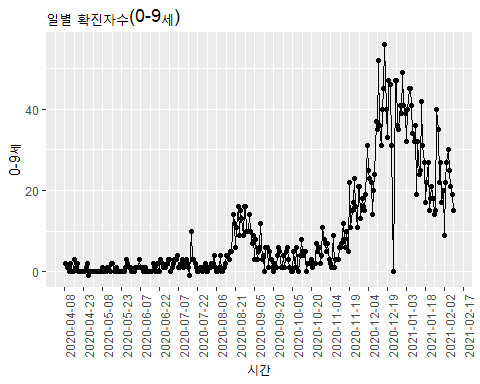
ggplot(data = employees, aes(x = time, y = total)) +  
 geom\_line(aes(group = 1)) +   
 geom\_point(shape = 'circle') +  
 labs(title = '월별 취업자수', x = '기간', y = '취업자수') +  
 scale\_y\_continuous(labels = scales::number\_format(big.mark = ',')) +  
 scale\_x\_date(breaks = '6 month') +  
 theme(axis.text.x=element\_text(angle=90,hjust=1))



코드 설명

* scale\_x\_date()을 사용하여 X 축의 라벨 표기점을 설정. X 축으로 바인딩 된 컬럼이 date 클래스이므로 scale\_x\_date()를 사용할 수 있는데 표기점을 6개월마다로 설정(breaks = '6 month')

ggplot(data = covid19, aes(x = date, y = `0-9세`)) +  
 geom\_line(aes(group = 1)) +   
 geom\_point(shape = 'circle') +  
 labs(title = '일별 확진자수(0-9세)', x = '시간') +  
 scale\_y\_continuous(labels = scales::number\_format(big.mark = ',')) +   
 scale\_x\_date(breaks = '15 day') +  
 theme(axis.text.x=element\_text(angle=90,hjust=1))



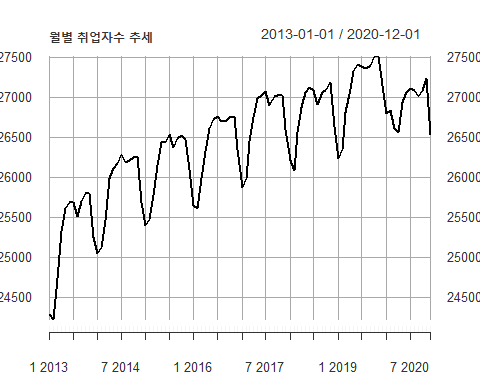
# xts : xts 패키지

xts는 시계열 데이터를 다루는 데이터 클래스로 xts 패키지를 로딩해야 사용할 수 있다. xts 패키지에서는 시간 처리 함수, 시계열 데이터 plotting 함수 등을 제공한다. 따라서 xts 패키지에서 제공하는 함수는 xts 클래스 객체를 대상으로 한다.

xts 패키지를 사용하여 plot을 작성하려면 plot.xts()를 사용한다. plot.xts()로 작성한 plot에 추가적인 데이터의 라인 plot이나 범례 등을 추가할 수 있다.

xts 패키지의 plot.xts()를 활용하는 것은 디자인적 측면에서 ggplot보다 못 미치지만 사용이 간편하다는 점에서 큰 장점이 있다.

library(xts)  
plot.xts(employees.xts$total, main = '월별 취업자수 추세', xlab = '월, 년', ylab = '취업자수')

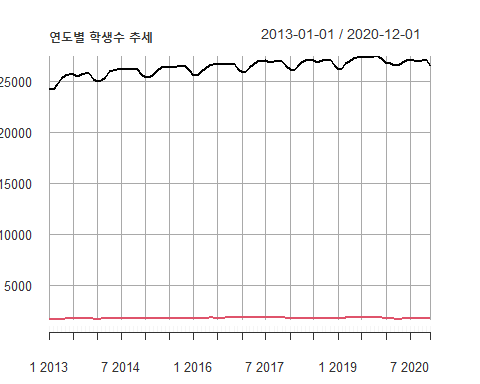


코드 설명

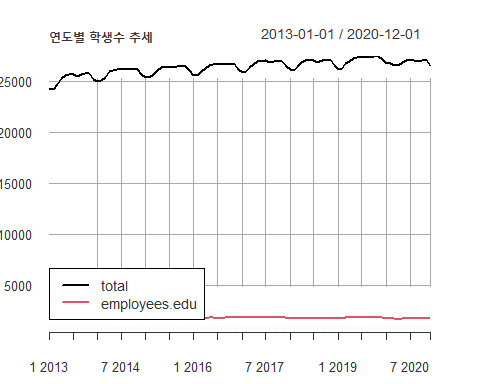
* xts 데이터를 다루기 위해 xts 패키지를 로딩
* plot.xts()을 사용하여 전체 취업자수(employees$total)의 plot을 생성
* plot의 제목(main =), X 축 제목(xlab =), Y 축 제목(ylab =)을 설정

위에서 본 것과 같이 xts 클래스 데이터의 단변량 plot은 plot.xts()로 간단히 그릴 수 있다. 그렇다면 다변량 plot은 어떻게 그릴 수 있을까? 또 다변량 plot은 반드시 범례를 같이 표기해야 하는데 이는 어떻게 추가할까? 같은 xts 클래스 데이터에 여러 컬럼으로 저장된 다변량 plot은 단지 plot.xts()에 xts 객체를 전달할 때 그리고자 하는 컬럼을 전달해주면 같이 그려지고 범례는 addLegend()를 사용하면 추가할 수 있다.

plot.xts(employees.xts, main = '연도별 학생수 추세', xlab = '년', ylab = '학생수', yaxis.right=FALSE)



addLegend('bottomleft', ncol = 1, bg = 'white', lty=c(rep(1, 12)), lwd=c(rep(2, 12)), bty="o")

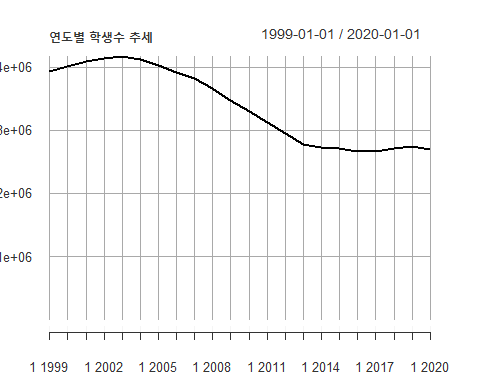


코드 설명

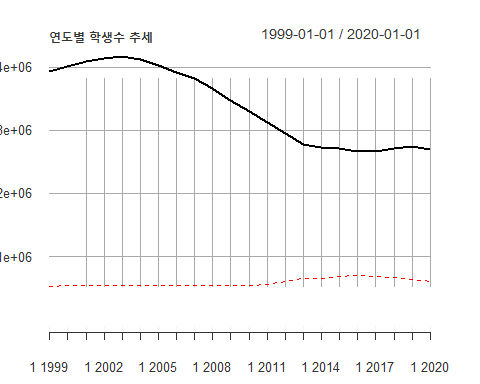
* plot.xts()에 두개의 시계열 데이터가 저장된 employees.xts를 전달함으로써 다변량 plot을 그림. 좌우 Y축에 모두 값이 표기되므로 우측 Y축에는 값을 제거

만약 같이 그리고자 하는 다변량 데이터가 다른 데이터 프레임에 있거나 컬럼을 같이 전달하기가 어려울 경우에는 다음과 같이 lines()를 사용하여 겹쳐서 그릴 수 있다.

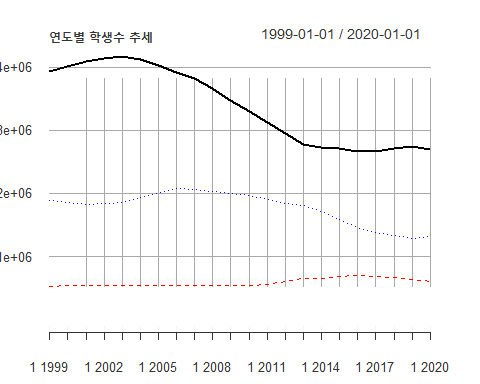
plot.xts(students.xts$초등학교, main = '연도별 학생수 추세', xlab = '년', ylab = '학생수', yaxis.right=FALSE, ylim = c(0, max(students.xts$초등학교)), col = 'black')



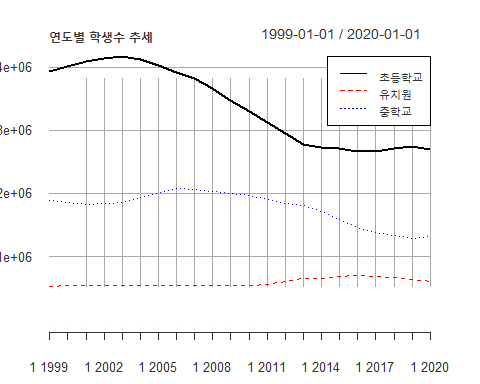
lines(students.xts$유치원, lty = 2, col = 'red')



lines(students.xts$중학교, lty = 3, col = 'blue')



addLegend('topright', ncol = 1, , legend.names = c('초등학교', '유치원', '중학교'), col = c('black', 'red', 'blue'), lty=c(1, 2, 3), bg = 'white', bty="o")

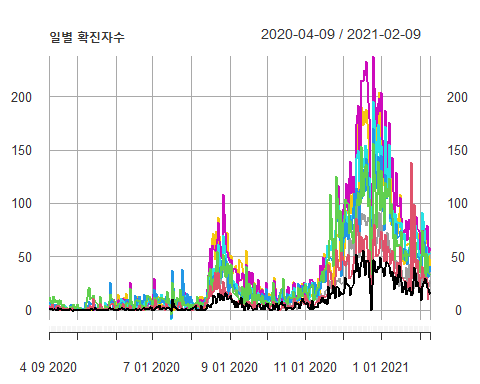


코드 설명

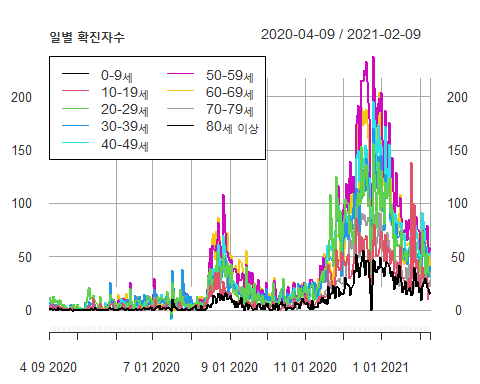
* plot.xts()를 사용하여 students.xts$초등학교 plot을 그림. plot 제목, 축 제목들을 적절히 설정. Y 축 값의 범위를 0부터 students.xts$초등학교의 최대값까지 설정하였는데 범위가 적절히 설정되지 않으면 뒤에 그려지는 라인들이 범위를 벗어나서 보이지 않을 수 있음.
* lines()를 사용하여 students.xts$유치원 라인 plot을 생성, 라인 타입(lty = 2)과 색(col = 'red')을 설정
* lines()를 사용하여 students.xts$중학교 라인 plot을 생성, 라인 타입(lty = 3)과 색(col = 'blue')을 설정
* addLegend()를 사용하여 범례를 생성. 범례의 행은 1개, 범례에 사용하는 item 이름(legend.names = c('초등학교', '유치원', '중학교')), 색(col = c('black', 'red', 'blue')), 라인 타입(lty=c(1, 2, 3))을 설정, 배경색은 흰색(bg = 'white'), 범례 배경 타입은 불투명(bty="o")으로 설정

plot.xts()를 사용하여 코로나 확진자에 대한 plot은 다음과 같이 그릴 수 있다.

plot.xts(covid19.xts, main = '일별 확진자수', xlab = '날짜', ylab = '확진자수')



addLegend('topleft', ncol = 2, , legend.names = c('0-9세', '10-19세', '20-29세', '30-39세', '40-49세', '50-59세', '60-69세', '70-79세', '80세 이상'), lty = 1, bg = 'white', bty="o")



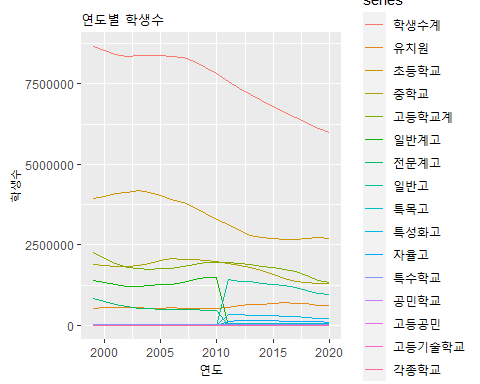
# ts : forecast 패키지

forecast 패키지는 시계열 자료의 시각화와 분석 방법과 도구를 제공하는 패키지로 호주의 Rob Hyndman교수가 주도하여 개발하였다.[[2]](#footnote-2) 이 패키지는 단변량(Univariate) 시계열 데이터에 적합하도록 설계되었고 plotting은 ggplot2 패키지를 기반으로 사용하고 있다.

forecast 패키지에서는 시계열 plotting 방법으로 autoplot()를 제공하고 있다. 사실 forecast 패키지에서 사용하는 autoplot() 함수는 ggplot2에서 제공하는 함수이지만 ggplot2에서는 ts 데이터 클래스를 지원하지 않는다. 따라서 autoplot() 함수에 전달된 데이터가 ts 클래스인 경우는 ggplot2 패키지의 autoplot() 함수가 아닌 forecast에서 제공하는 autoplot() 함수(autoplot.ts() 또는 autoplot.mts())를 호출하여 plot을 생성한다.[[3]](#footnote-3)

autoplot()는 ggplot2와 마찬가지로 Geometry layer를 겹쳐가면서 최종 plot를 생성하는데 + 기호를 사용하여 autolayer() 함수를 사용해 layer를 겹칠 수 있다.

library(forecast)  
autoplot(students.ts[,-1], main = '연도별 학생수', xlab = '연도', ylab = '학생수')



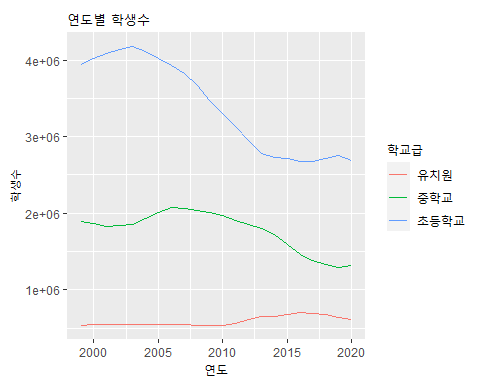
코드 설명

* autoplot()을 사용하여 plot을 생성. students.ts의 첫번째 컬럼은 연도이므로 제외(students.ts[,-1])하고 plot 제목(main =) , X 축 제목(xlab =), Y 축 제목(ylab = )을 설정.

위의 plot에서는 students.ts의 전체 컬럼을 모두 plotting하였다. 하지만 일부만 그리고 싶다면 어떻게 할까? autoplot()에서도 앞선 plot.xts()와 유사한 방법을 사용한다.

먼저 autoplot()으로 처음 그리고 싶은 데이터를 설정하여 그리고 + autolayer()를 사용하여 추가적인 layer를 겹쳐서 그린다. 위의 plot에서 유치원, 초등학교, 중학교만 그리면 다음과 같다. 단 여기서 주의해야 할 것은 ts 객체에서는 $를 사용한 열 이름에 접근할 수 없기 때문에 []에 열 번호를 넣어 접근해야한다.

autoplot(students.ts[, 4], main = '연도별 학생수', xlab = '연도', ylab = '학생수', series = '초등학교') +   
 autolayer(students.ts[, 3], series = '유치원') +   
 autolayer(students.ts[, 5], series = '중학교') +   
 labs(colour = "학교급")

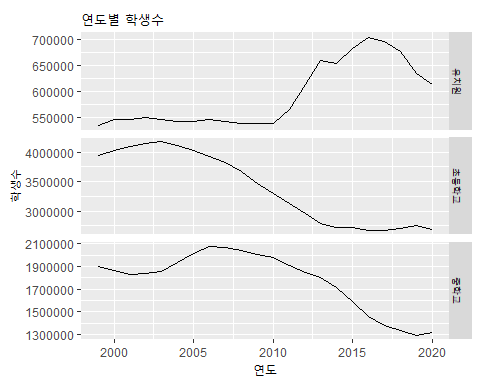


코드 설명

* autoplot()을 사용하여 students.ts의 네번째 컬럼(초등학교) plot을 생성. plot 제목(main =) , X 축 제목(xlab =), Y 축 제목(ylab = ), 범례 아이템 이름(series =)을 설정.
* autolayer()를 사용하여 students.ts의 세번째 컬럼(유치원) plot layer를 추가하고 범례 아이템 이름(series =)을 설정.
* autolayer()를 사용하여 students.ts의 다섯번째 컬럼(중학교) plot layer를 추가하고 범례 아이템 이름(series =)을 설정.
* labs()를 사용하여 범레 제목을 설정(colour =)

이번에는 다변량 plot을 하나의 plot이 아닌 여러 plot으로 그려보겠다. ggplot에서도 facet 요소를 사용하여 다중 plot을 생성하는데 autoplot()에서도 facet 매개변수를 사용하여 다중 plot을 그릴 수 있다.

autoplot(students.ts[, 3:5], main = '연도별 학생수', xlab = '연도', ylab = '학생수', facet = TRUE)

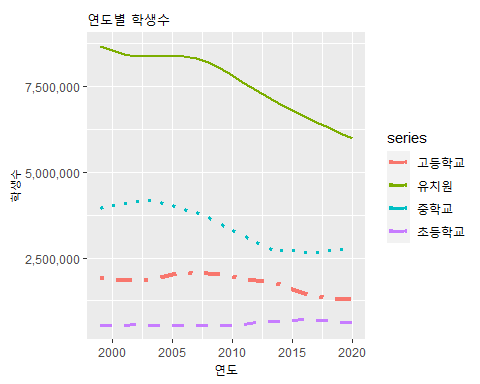


코드 설명

* autoplot()을 사용하여 students.ts의 세번째, 네번째, 다섯번째 컬럼(유치원, 초등학교, 중학교) plot을 생성. plot 제목(main =) , X 축 제목(xlab =), Y 축 제목(ylab = ), 범례 아이템 이름(series =)을 설정하고 다중 plot을 사용하도록 설정(facet = TRUE)

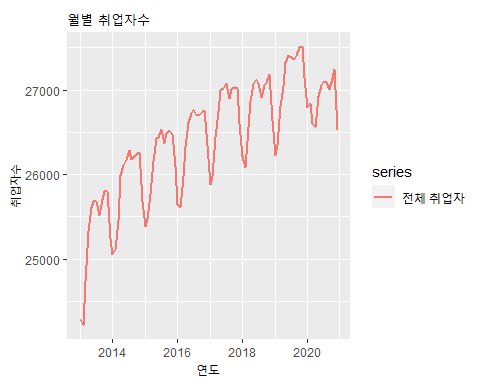
그 외 세부적인 plot의 변경은 ggplot에서 사용하는 방법을 준하여 사용할 수 있다.

autoplot(students.ts[,2], main = '연도별 학생수', xlab = '연도', ylab = '학생수', series = '유치원', lty = 1, lwd = 1) +  
 autolayer(students.ts[,3], series = '초등학교', lty = 2, lwd = 1.2) +  
 autolayer(students.ts[,4], series = '중학교', lty = 3, lwd = 1.4) +  
 autolayer(students.ts[,5], series = '고등학교', lty = 4, lwd = 1.6) +   
 scale\_y\_continuous(labels=scales::number\_format(big.mark = ','))

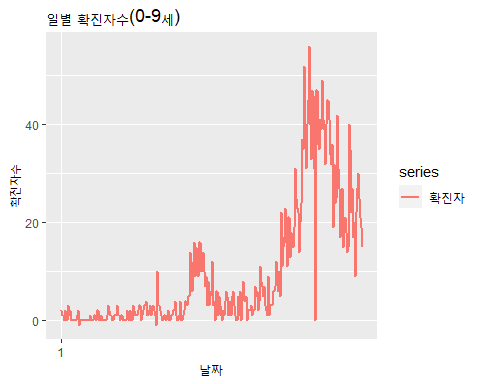


autoplot()을 사용해서 전체 취업자수와 코로나 확진자수 plot은 다음과 같이 그린다.

autoplot(employees.ts[,2], main = '월별 취업자수', xlab = '연도', ylab = '취업자수', series = '전체 취업자', lty = 1, lwd = 1)



autoplot(covid19.ts[,2], main = '일별 확진자수(0-9세)', xlab = '날짜', ylab = '확진자수', series = '확진자', lty = 1, lwd = 1)

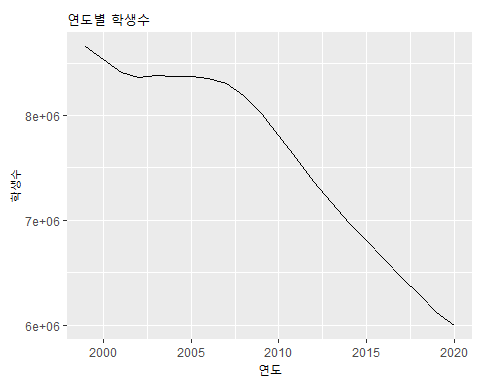


# tsibble : feasts 패키지

tsibble 객체를 분석하고 plotting 하기 위해서 feasts 패키지가 제공된다. feasts는 ’Feature Extraction And Statistics for Time Series’의 준말이다. tsibble 객체를 대상으로 시계열 처리, 분해, 통계 산출, 시각화를 위한 각종 함수들을 제공한다.

feasts도 앞서 ts를 다루는 forecast 패키지의 개발자인 Rob Hyndman 교수가 제작한 패키지이기 때문에 forecast와 유사한 함수들이 있다. 우선 tsibble 객체의 가장 간단한 plot을 생성하는 방법은 forecast 패키지와 같이 autoplot()을 사용하는 방법이다.

library(feasts)  
library(dplyr)  
students.tsibble %>% autoplot(학생수계)+   
 labs(title = '연도별 학생수', x = '연도', y = '학생수')



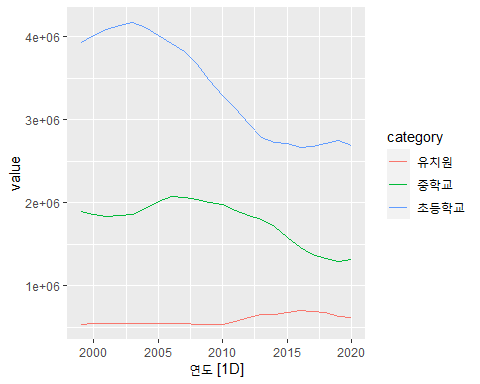
코드 설명

* %>%를 사용하여 students.tsibble을 autoplot()에 전달하고 students.tsibble의 ‘학생수계’ 열의 데이터를 plotting함. labs()를 사용하여 plot 제목(title =), X 축 제목(x =), Y 축 제목(y = )을 설정

하지만 forecast와는 달리 feasts에서는 autolayer()를 통해서 원하는 데이터의 layer를 추가할 수 없다. 따라서 다중(multivariate) 시계열 plot을 그리기 위해서는 긴(long)형태의 tsibble 객체로 변환하여야 한다.

students.tsibble %>% select(1, 3, 4, 5) %>%   
 tidyr::gather(category, value, 2:4) %>% autoplot()

Plot variable not specified, automatically selected `.vars = value`

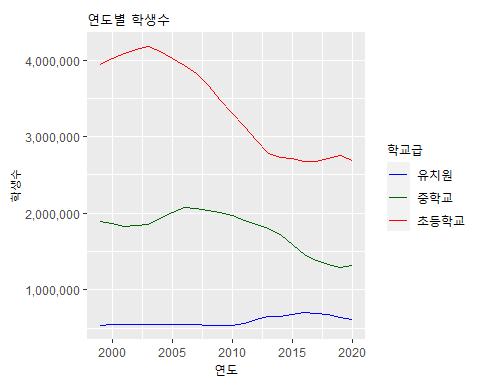


코드 설명

* %>%를 students.tsibble의 첫번째, 세번째, 네번째, 다섯번째 열을 선택하고 이 열들을 gather()를 사용하여 category 열로 모으고 해당 데이터를 value 열로 모음.
* 이 후 autoplot()을 사용하여 다중 시계열 plot을 생성.

위와 같이 tidy에서 제공하는 몇가지 방법을 사용하면 원하는 plot을 생성할 수 있지만 범례 설정이나 세부적인 설정이 여간 곤혹스러운 것이 아니다. 따라서 feasts에서 제공하는 plotting 방법보다는 차라리 ggplot을 사용하는 것이 오히려 편하다.

ggplot(students.tsibble, aes(x = 연도)) +  
 geom\_line(aes(y = 초등학교, group = 1, color = '초등학교')) +  
 geom\_line(aes(y = 유치원, group =1, color = '유치원')) +   
 geom\_line(aes(y = 중학교, group =1, color = '중학교')) +   
 labs(title = '연도별 학생수', x = '연도', y = '학생수', color = '학교급') +   
 scale\_y\_continuous(labels = scales::number\_format(big.mark = ',')) +  
 scale\_colour\_manual(values = c('초등학교' = "red", '유치원' = "blue", '중학교' = 'darkgreen'))

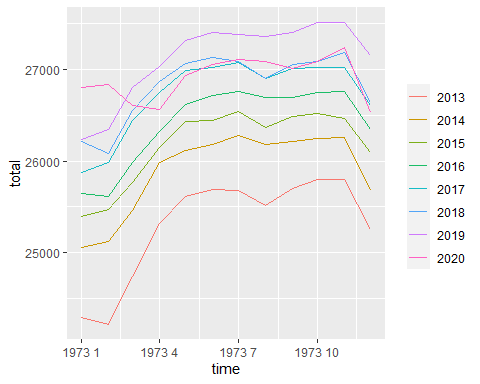


코드 설명

* ggplot()으로 원본 데이터인 students.tsibble을 plot 대상 데이터로 설정하고 aes()로 공통으로 사용될 X축 설정만 해줌.
* geom\_line으로 첫번째 라인 geometry layer를 생성. aes()로 Y축을 초등학교에 바인딩. 초등학교 열은 단일 데이터 그룹핑(group = 1), color를 ’초등학교’로 매핑(color = '초등학교', 범례에 매핑해줄 이름).
* geom\_line()으로 두번째 라인 geometry layer를 생성. aes()로 Y축을 유치원에 바인딩. 유치원 열은 단일 데이터 그룹핑(group = 1), color를 ’유치원’으로 매핑.
* labs()를 사용해 plot 제목, X축 제목, Y축 제목, 범례 제목을 설정
* scale\_y\_continuous()를 사용하여 Y축의 라벨을 숫자형(천 단위 콤마)으로 조절
* scale\_color\_manual()을 사용하여 컬러 매핑값을 실제 컬러값으로 설정

feasts 패키지에서는 시계열 분석에 사용하는 계절성 plot 함수인 gg\_season() 함수와 gg\_subseries() 함수를 제공한다.

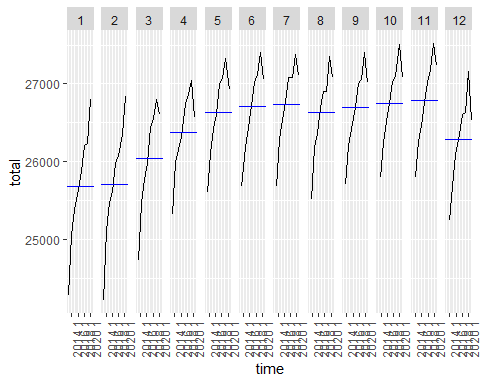
employees.tsibble %>% mutate(time = yearmonth(employees.tsibble$time)) %>%  
 gg\_season(total)



코드 설명

* employees.tsibble은 월별 데이터이기 때문에 gg\_season()을 사용하기 위해서는 yearmonth()를 사용하여 시간 인덱스(time) 열을 yearmonth 데이터 클래스로 변환하여 사용.

employees.tsibble %>% mutate(time = yearmonth(employees.tsibble$time)) %>%  
 gg\_subseries(total)



# data.frame : timetk 패키지

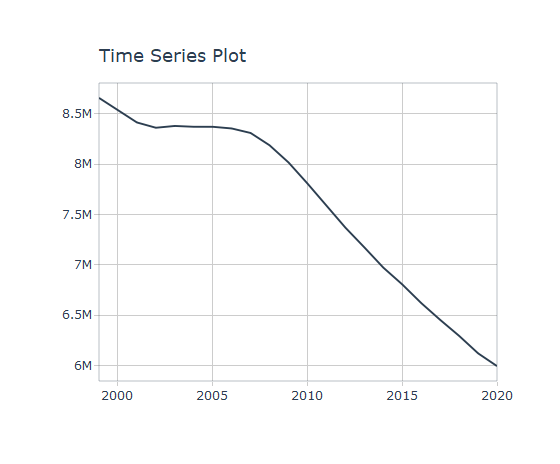
timetk 패키지는 미래 예측, 머신러닝 예측을 위해 시계열 데이터의 시각화와 데이터 처리를 하기 위한 패키지이다. 이 패키지는 dplyr, xts, forecast 등의 패키지를 기반으로 설계되었고 xts패키지와 달리 tibble이나 data.frame을 기반으로 작동한다. [[4]](#footnote-4)

timetk에서 제공하는 데이터 plot 함수는 plot\_time\_series() 함수가 유일하다. 이외에도 시계열 특성을 확인하기 위한 자기상관관계, 계절성, 이상치 plotting을 위한 함수들을 추가적으로 제공한다.

timetk plot이 다른 plot에 비해 장점이 여러가지가 있는데 필자가 생각하는 가장 큰 장점은 반응형(interactive) plot이 자동적으로 생성된다는 점이다. plot\_time\_series() 함수는 plot\_ly 패키지를 기반으로 plotting하기 때문에 plot\_ly 패키지에서 제공하는 plot의 장점을 모두 사용할 수 있다. 마우스를 plot위로 가져가면 해당 마우스 포인트에 대응하는 정보가 표현되거나 드래그하여 plot을 zoom in 할 수 있는 기능을 기본적으로 제공한다.

plot\_time\_series()를 사용하여 시계열 plot을 만드는 방법은 다음과 같다. 아래의 예제에서 .value로 설정된 하나의 컬럼을 plotting되어야 하지만 실제는 두개의 선이 나타난다. 검정색 선은 실제 데이터를 표현하는 line plot이고 파랑색 선은 ‘LOESS’ 평활화된 line plot이 나타난다. 평활화 line은 .smooth 매개변수로 조절 가능하다.

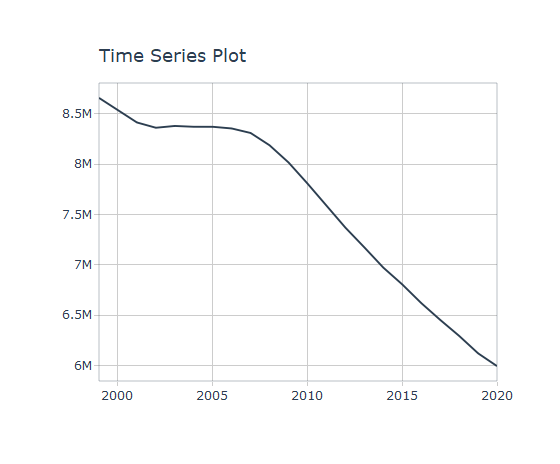
library(timetk)  
students %>%   
 plot\_time\_series(.date\_var = 연도, .value = 학생수계, .smooth = FALSE)



코드 설명

* timetk 패키지를 로딩
* %>%을 사용하여 students를 plot\_time\_series()에 전달. 시간인덱스로 사용할 열은 ’연도’열(.date\_var =)로, 데이터로 사용할 열을 ’학생수계’열(.value =)로 설정하고 추세선을 없앰(.smooth =)

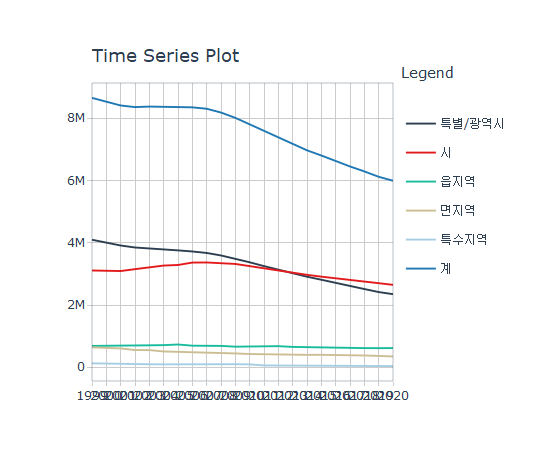
students %>%  
plot\_time\_series(.date\_var = 연도, .value = 학생수계, .smooth = FALSE)



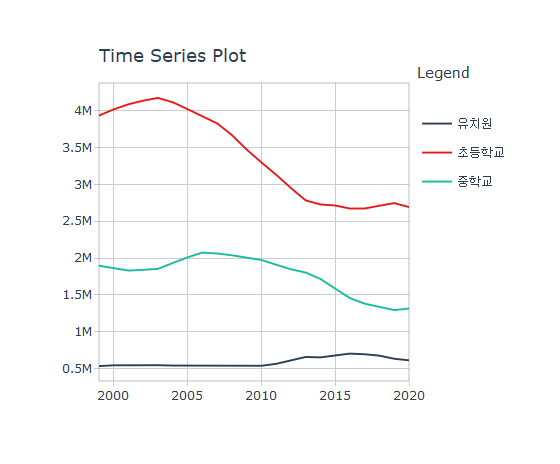
plot\_time\_series()를 사용하여 다변량 plot을 하기 위해서는 ggplot와 같이 다변량으로 데이터를 분류해줄 정보가 포함된 열이 필요하다. 즉 긴(long) 형태의 데이터가 필요하다는 것이다.

다변량 데이터는 컬러로 구분되어 표현되며 자동적으로 범례가 오른쪽에 표시된다.

students.all %>%  
plot\_time\_series(.date\_var = 연도, .value = 학생수계, .color\_var = 지역규모, .smooth = F)

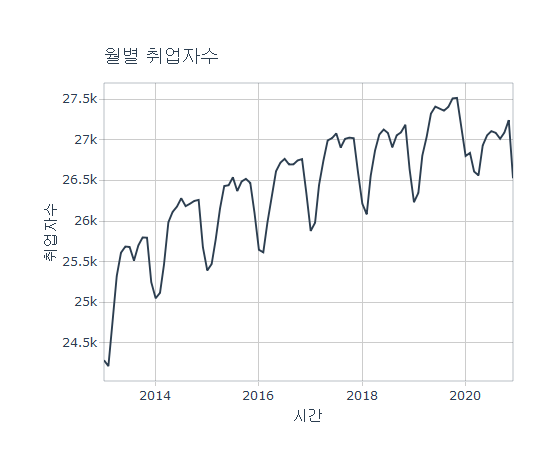


students %>% select(1, 3, 4, 5) %>%   
 tidyr::gather(category, value, 2:4) %>%  
 plot\_time\_series(.date\_var = 연도, .value = value, .color\_var = category, .smooth = F)

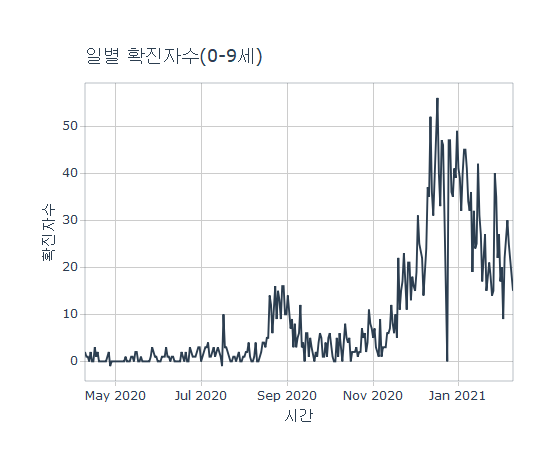


그 외 추가적인 매개변수는 timetk의 plot\_time\_series() 함수 매뉴얼을 참조하면 더 추가적인 정보를 얻을 수 있다.[[5]](#footnote-5)

employees %>%  
 plot\_time\_series(.date\_var = time, .value = total, .smooth = F, .title = '월별 취업자수', .x\_lab = '시간', .y\_lab = '취업자수')



covid19 %>%  
 plot\_time\_series(.date\_var = date, .value = `0-9세`, .smooth = F, .title = '일별 확진자수(0-9세)', .x\_lab = '시간', .y\_lab = '확진자수')



1. <https://ramnathv.github.io/pycon2014-r/visualize/ggplot2.html> [↑](#footnote-ref-1)
2. <https://cran.r-project.org/web/packages/forecast/forecast.pdf> [↑](#footnote-ref-2)
3. 이런 형태의 함수를 랩핑(wrapping) 함수라고 한다. [↑](#footnote-ref-3)
4. <https://business-science.github.io/timetk/> [↑](#footnote-ref-4)
5. <https://business-science.github.io/timetk/reference/plot_time_series.html> [↑](#footnote-ref-5)