II. 시계열 데이터 객체

# 시계열 데이터 타입 및 Import

시계열 데이터는 시간을 인덱스로 사용해야하기 때문에 반드시 시간형 데이터 클래스가 존재하거나 이에 준하는 정보가 포함되어야 한다. R에서는 시간을 인덱스로 허용하는 데이터 클래스를 몇가지 제공하고 있는데 그 클래스에 따라 데이터 처리 함수, plot 함수, 모델 생성 함수 등의 사용법이 다르기 때문에 주의 깊게 설정해야 한다.

또한 각각의 데이터 클래스들이 장단점을 가지기 때문에 데이터 간의 변환을 통해 사용해야할 경우도 있다. 이 장에서는 R에서 많이 사용하는 날짜/시간 데이터 클래스와 시계열 전용 객체를 소개하고 데이터를 R로 불러들여 시계열 객체로 저장하는 방법에 대해 알아본다. 여기서 생성된 시계열 데이터를 사용하여 앞으로 시계열 모델링, 예측에 사용할 것이다.

## 날짜/시간 데이터 클래스

앞서 설명한 것과 같이 시계열 분석을 위한 시계열 데이터에는 반드시 시간이 표현된 데이터가 포함되어야 한다. 따라서 R이 특정 데이터를 시간으로 이해하기 위해서는 R이 이해할 수 있는 형태로 시간을 표기해야 한다. R이 이해하는 시간 데이터의 형태는 date, POSIXct, POSIXlt, yearmon, yearqt 클래스 등이 있다.

### date 클래스

R 에서 기본적으로 제공하는 시간 표현 클래스로 1970년 이후의 달력 상의 날짜 수를 기록하는 데이터 클래스이다. 양수일 경우 1970년 이후의 날짜이며 음수인 경우 1970년 이전의 날짜를 표시한다. date class는 실제 R내부에 저장 시에는 정수 형태로 저장되지만 출력 시에는 ‘yyyy-mm-dd’ 형태로 표현된다.

date 클래스는 특별한 패키지가 필요하지 않고 바로 사용할 수 있다. date 타입의 데이터를 생성하기 위해서는 아래와 같이 as.Date()를 사용한다.

(date <- as.Date(c('2021-01-31', '2021-02-28', '2021-03-31')))

[1] "2021-01-31" "2021-02-28" "2021-03-31"

(date <- as.Date(c('21/01/31', '21/02/28', '21/03/31'), format = '%y/%m/%d'))

[1] "2021-01-31" "2021-02-28" "2021-03-31"

unclass(date)

[1] 18658 18686 18717

코드 설명

* as.Date()는 벡터로 전달된 데이터를 date 클래스로 변환
* 변환하기 위해서는 전달된 벡터가 date format을 갖추어야 하나 갖추지 못한 경우는 format 매개변수로 설정 가능하며 많이 사용되는 format 지정자는 다음 절에 설명
* date class의 클래스를 해체(unclass())하면 내부적으로 저장된 1970년 이후의 날짜수가 보임

### POSIXct, POSIXlt class

date 클래스는 1970년 이후 일의 수를 기록하는 클래스이지만 POSIXct 와 POSIXlt 클래스는 1970년 이후 시간을 초 단위로 기록하는 데이터 클래스이다. POSIXct는 date 클래스와 같이 1970년이후의 시간을 초 단위의 정수로 기록하는 클래스이고 POSIXlt는 년, 월, 일, 시, 분, 초의 정보를 리스트형태로 기록하는 클래스이다. POSIXlt는 1900년 이후로 계산되어 리스트가 만들어 진다.

# character를 POSIXct class 로 변환  
as.POSIXct('2021-01-31 12:34:56')

[1] "2021-01-31 12:34:56 KST"

# POSIXct를 해제하면 정수  
unclass(as.POSIXct('2021-01-31 12:34:55'))

[1] 1612064095  
attr(,"tzone")  
[1] ""

# character를 POSIXlt class 로 변환  
as.POSIXlt('2021-01-31 12:34:56')

[1] "2021-01-31 12:34:56 KST"

# POSIXlt를 해제하면 list  
unclass(as.POSIXlt('2021-12-31 12:34:56')

# POSIXlt에서 1900년 이후 연도를 추출  
as.POSIXlt('2021-12-31 12:34:56')$year

[1] 121

코드 설명

* as.POSIXct()는 벡터로 전달된 데이터를 POSIXct 클래스로 변환
* POSIXct class를 해체(unclass())하면 내부적으로 저장된 1970년 이후의 해당 시간까지의 초 수가 보임
* as.POSIXlt()는 벡터로 전달된 데이터를 POSIXlt 클래스로 변환
* POSIXlt class를 해체하면 리스트로 저장된 날짜 속성이 보임
* POSIXlt에서 날짜 속성을 뽑아내려면 $속성명을 사용

### yearmon, yearqtr class

yearmon과 yearqtr 클래스는 모두 zoo 패키지에서 제공하는 클래스이다. 따라서 두 클래스를 사용하기 위해서는 zoo 패키지를 설치하고 로딩해야 한다.

yearmon과 yearqtr 클래스는 연, 월로 표현되거나 년, 분기로 표현된 시간 데이터가 있을 때 사용한다. yearmon 클래스는 연, 월별 데이터를 표현하는 클래스이고 yearqtr는 연, 분기 데이터를 표현하는 클래스이다. yearmon 클래스는 1월을 0으로 2월을 1/12 = 0.083, 12월을 11/12 = 0.917로 표기하고 yearqtr 클래스는 각 분기마다 0.25씩 더해서 저장되지만 표현될 때는 우리가 쓰는 시간 형태로 표현된다.

if(!require(zoo)) {  
 install.packages('zoo')  
 library(zoo)  
}  
# character를 yearmon class로 변환  
as.yearmon("2007-02")

[1] "2 2007"

# yearmon class를 해제하면 double  
unclass(as.yearmon("2007-02"))

[1] 2007.083

# 날짜가 있어도 yearmon은 년, 월까지만 인식  
as.yearmon("2007-02-01")

[1] "2 2007"

# character를 yearqtr class로 변환(1분기)  
as.yearqtr("2007-01")

[1] "2007 Q1"

# yearqtr class를 해제하면 double  
unclass(as.yearqtr("2007-04"))

[1] 2007.75

코드 설명

* yearmon, yearqtr 클래스를 사용하기 위해서는 zoo 패키지를 먼저 로딩해야 함.
* as.yearmon()로 날짜 문자열이나 date 클래스를 yearmon 클래스로 변환
* yearmon 클래스를 해체하면 정수 부분이 연도이고 소수점부분이 월(month/12)로 표현된 실수로 보임
* as.yearmon()에 연월일을 전달해도 연월만 인식
* as.yearqtr()로 날짜 문자열이나 date 클래스를 yearqtr 클래스로 변환
* yearmon 클래스를 해체하면 정수 부분이 연도이고 소수점부분이 분기(분기/4)로 표현된 실수로 보임

### 날짜, 시간 포맷

날짜와 시간을 표현하는 방법은 여러가지가 있다. R에서는 as.Date(), as.POSIXct()등과 같은 시간 클래스 생성 함수에서 format을 사용하여 다양한 날짜와 시간의 표현을 지원한다. format 매개변수에 의해 표현되는 날짜, 시간 표기는 날짜 표현 방법, 시간 클래스의 지역 설정(timezone)에 따라 달라진다. 예를 들어 우리나라 지역 설정은 %B, %b가 ‘January’, ‘Jan’ 이 아닌 ’1월’로 표기된다.

standard date format codes

|  |  |  |
| --- | --- | --- |
| Code | Value | Example |
| %d | 월의 날짜(정수) | 23 |
| %m | 월(정수) | 12 |
| %B | 전체 월 이름 | January |
| %b | 축약형 월 이름 | Jan |
| %Y | 4자리 연도 | 2010 |
| %y | 2자리 연도 | 10 |
| %a | 요일 | 수 |

as.Date('01/12/2010', format = '%d/%m/%Y')

[1] "2010-12-01"

Sys.setlocale("LC\_ALL", "English")

[1] "LC\_COLLATE=English\_United States.1252;LC\_CTYPE=English\_United States.1252;LC\_MONETARY=English\_United States.1252;LC\_NUMERIC=C;LC\_TIME=English\_United States.1252"

as.Date('01jan21', format = '%d%b%y')

[1] "2021-01-01"

Sys.setlocale("LC\_ALL", "Korean")

[1] "LC\_COLLATE=Korean\_Korea.949;LC\_CTYPE=Korean\_Korea.949;LC\_MONETARY=Korean\_Korea.949;LC\_NUMERIC=C;LC\_TIME=Korean\_Korea.949"

as.Date('011월21', format = '%d%b%y')

[1] "2021-01-01"

코드 설명

* format이 %d/%m/%Y형태로 표시된 문자열을 date형태로 변환
* timezone을 영국으로 설정
* format이 %d%b%y형태로 표시된 문자열을 date형태로 변환
* timezone을 한국으로 설정
* format이 %d%b%y형태로 표시된 문자열을 date형태로 변환

## 시계열 데이터 객체(Object)

R에서는 일반적으로 데이터 프레임에 데이터를 저장하는 경우가 많다. 시계열 데이터도 데이터프레임에 저장할 수 도 있다. 하지만 시계열 데이터는 시간 인덱스를 기반으로 검색, 분할, 집계, 병합과 같은 데이터 핸들링이 이루어지기 때문에 데이터 프레임보다는 시계열 데이터를 처리하는 객체에서 처리하는 것이 효과적이다.

따라서 R에서는 시간 인덱스를 기반으로 데이터를 처리하기 위한 특별한 데이터 객체를 제공하고 있다.[[1]](#footnote-1) R에서 사용할 수 있는 시계열 데이터 객체는 R에서 자체적으로 에서 제공하는 객체도 있지만 시계열 패키지를 통해 사용하는 객체도 있다.

### ts : R base(stats)

ts 객체(Object)는 R에서 가장 기본적으로 활용되는 시계열 데이터 타입이다. ts 객체는 ‘stats’ 패키지를 로딩해야 사용할 수 있지만 R이 실행될 때 기본적으로 로딩되기 때문에 바로 활용할 수 있다. 또한 stats 패키지에 포함된 다양한 시계열 데이터 처리 함수에서 다루는 객체이기 때문에 많이 사용된다.[[2]](#footnote-2)

ts 객체는 앞서 설명한 바와 같이 R에서 기본적으로 제공하는 시계열 객체라는 장점이 있지만 사용상의 단점도 있다. 우리는 시계열 데이터를 사용할 때 외부에서 생성된 시계열 데이터를 R로 불러들여 사용하는 경우가 일반적이다. 이 경우 외부에서 생성된 데이터는 일반적으로 데이터의 시간이 같이 기록되어 있고 사용자는 기록된 시간을 불러들여 사용하고 싶을 것이다. 하지만 ts 객체는 생성시에 데이터에 기록된 시간을 사용하는 것이 아닌 객체 생성시에 시계열 데이터의 주기(1은 연도 데이터, 12는 월 데이터 등)를 설정해야 하고 시작일, 종료일을 설정하여 날짜를 자체 설정하도록 규정하고 있다. 따라서 외부에서 생성된 데이터에 결측치가 존재하는 경우 후처리 과정이 필요할 수도 있다.

ts 데이터 객체는 다음과 같이 생성할 수 있다.

ts(1:10, frequency = 4, start = c(1959, 2))

Qtr1 Qtr2 Qtr3 Qtr4  
1959 1 2 3  
1960 4 5 6 7  
1961 8 9 10

코드 설명

* 1959년 2분기(start = c(1959, 2))부터 주기가 4(분기,frequency = 4)인 ts 객체 생성

### xts

xts는 extensible time-series의 준말로 xts 패키지를 로딩해야 활용할 수 있는 데이터 객체이다. xts는 뒤에서 다룰 모델링 패키지(forecast, fable, modeltime)에서 사용되지 않는 시계열 객체이다. 하지만 시계열 데이터 핸들링을 위한 다양한 함수를 제공하기 때문에 간단히 데이터를 확인하거나 데이터를 원하는 형태로 변환하는데 쉬운 함수들을 제공한다. 그래서 xts 객체를 사용하여 원하는 형태로 데이터를 만들고 다른 시계열 객체로 변환하는 것도 좋은 방법이다. xts로 데이터를 핸들링 하는 방법은 다음 장에서 설명한다.

xts 클래스를 사용하기 위해서는 먼저 xts 패키지를 설치하고 로딩해야 한다. 필자는 패키지를 로딩 할 때 다음과 같이 설치, 로딩을 한다.

if(!require(xts)) {  
 install.packages('xts')  
 library(xts)  
}

코드 설명

- require()는 library()와 같이 패키지를 로딩하는 함수이지만 다른 함수 안에서 사용하는 것을 목적으로 만들어진 함수임. 로딩에 성공하면 TRUE, 로딩에 실패하면 FALSE를 반환함.

- 만약, 패키지 로딩에 실패하면(!require(xts)) 패키지 설치 (install.packages('xts'))하고 로딩(library(xts))함.

xts 데이터 객체 생성은 다음과 같이 생성할 수 있다.

xts(rnorm(5), as.Date("2008-08-01") + 0:4)

[,1]  
2008-08-01 -0.9156415  
2008-08-02 -1.9974701  
2008-08-03 1.5285146  
2008-08-04 -1.5784030  
2008-08-05 1.3657558

as.xts(rnorm(5), as.Date("2008-08-01") + 0:4)

[,1]  
2008-08-01 0.7145142  
2008-08-02 -0.5901271  
2008-08-03 0.7750809  
2008-08-04 -0.7239417  
2008-08-05 0.6222250

코드 설명

- 2008년 8월 1일 부터 5일치(as.Date("2008-08-01") + 0:4) 데이터를 가지는 xts object 생성

- 2008년 8월 1일 부터 5일치 데이터를 가지는 xts 객체로 변환

xts()는 xts 객체를 생성하는 함수이나 as.xts()는 timeSeries, ts, irts, fts, matrix, data.frame, zoo 객체를 xts객체로 변환하는 함수이다.[[3]](#footnote-3) 다음과 같은 차이가 발생한다.

ts <- ts(1:10, frequency = 4, start = c(1959, 2))  
xts(ts)

Error in xts(ts): order.by requires an appropriate time-based object

as.xts(ts)

[,1]  
1959 Q2 1  
1959 Q3 2  
1959 Q4 3  
1960 Q1 4  
1960 Q2 5  
1960 Q3 6  
1960 Q4 7  
1961 Q1 8  
1961 Q2 9  
1961 Q3 10

코드 설명

* xts를 생성해야하나 인덱스가 정해지지 않아 에러 발생
* ts 객체의 특성을 해석하여 인덱스를 설정하여 xts로 변환

ts, xts는 모두 시계열 데이터를 다루는 객체이지만 xts가 보다 시계열 데이터를 다루는데 유연한 함수가 많다. 최근에는 ts 객체에도 동일하게 적용할 수 있는 xts 함수가 제공되고 있는데, xts 매뉴얼에 의하면 as.xts와 reclass를 활용하는 것이 xts를 사용하는 것보다 이익이 있다고 명기하고 있다.[[4]](#footnote-4)

# 시계열 데이터 형태로 보이지 않음  
head(ts)

[1] 1 2 3 4 5 6

# 시계열 형태로 보임  
head(as.xts(ts))

[,1]  
1959 Q2 1  
1959 Q3 2  
1959 Q4 3  
1960 Q1 4  
1960 Q2 5  
1960 Q3 6

### tsibble

최근 R에서 데이터를 처리하는 방법으로 많이 사용되는 방법이 tidyverse 를 사용하는 방법이다. 이 방법을 시계열 데이터 작업에도 사용하기 할 수 있도록 해주는 방법으로 제공되는 것이 tidyverts 패밀리 패키지들을 사용하는 것이다.

tidyverts는 tidyverse처럼 단독으로 제공되는 패키지는 아니지만 시계열 데이터를 저장할 수 있는 객체 패키지인 tsibble, 시계열 예측을 위한 fable, 시계열 특성 추출과 통계를 위한 feast, 최근에 페이스 북에서 개발된 prophet 모델을 사용하기 위한 fable.prophet 패키지 등이 포함한다.[[5]](#footnote-5)

tsibble 객체는 tsibble 패키지를 통해 제공되는 시계열 데이터 객체로써 tidy 데이터 원칙[[6]](#footnote-6)을 준용하여 시계열 데이터를 다룰 수 있도록 한 tibble 객체를 시계열 데이터에서 사용할 수 있도록 확장한 객체이다.

tsibble 객체는 tibble에서 각 관찰치(observation)를 고유하게 식별할 수 있는 컬럼 혹은 컬럼의 집합인 key와 시간의 순서가 지정되는 index를 필요로 한다. 즉 tibble 객체에서는 key로 특정 관찰치를 식별할 수 있지만 tsibble에서는 key를 통해 관찰치를 고유하게 식별하고 관찰치들은 다시 시간(index)에 따라 구분될 수 있다. 따라서 tsibble은 key와 index를 사용하여 고유한(unique) 관찰치의 고유한 데이터를 식별하게 된다.[[7]](#footnote-7)

tsibble 객체를 사용하기 위해서는 우선 tsibble 패키지를 설치하고 로딩해야한다. 다음과 같이 설치는 install.package 함수를 사용하고 로딩은 library 함수를 사용한다.

if(!require(tsibble)) {  
 install.packages('tsibble')  
 library(tsibble)  
 }

tsibble 패키지를 로딩 한 후에 tsibble 객체를 생성할 수 있다. tsibble 객체를 생성하는 방법은 tsibble() 을 사용해서 직접 tsibble 객체를 생성할 수도 있고 데이터 프레임으로 생성된 객체를 as.tsibble()를 사용해 tsibble 객체로 변환하는 방법이 있다. 이 중 데이터 프레임을 tsibble 객체로 변환하는 방법은 다음과 같다.

library(dplyr)  
x <- data.frame(date = as.Date('2008-01-01') + 0:9, id = 1:10, x1 = rnorm(10), x2= rep('a', 10))  
as\_tsibble(x, key = id, index = date)

| date | id | x1 | x2 |
| --- | --- | --- | --- |
| 2008-01-01 | 1 | -1.68231709 | a |
| 2008-01-02 | 2 | 0.63976279 | a |
| 2008-01-03 | 3 | 2.09829689 | a |
| 2008-01-04 | 4 | -0.70308962 | a |
| 2008-01-05 | 5 | -2.41780887 | a |
| 2008-01-06 | 6 | -1.19424856 | a |
| 2008-01-07 | 7 | 0.22373477 | a |
| 2008-01-08 | 8 | -0.49181182 | a |
| 2008-01-09 | 9 | -0.05175469 | a |
| 2008-01-10 | 10 | -1.38208444 | a |

as\_tsibble(x, index = date)

| date | id | x1 | x2 |
| --- | --- | --- | --- |
| 2008-01-01 | 1 | -1.68231709 | a |
| 2008-01-02 | 2 | 0.63976279 | a |
| 2008-01-03 | 3 | 2.09829689 | a |
| 2008-01-04 | 4 | -0.70308962 | a |
| 2008-01-05 | 5 | -2.41780887 | a |
| 2008-01-06 | 6 | -1.19424856 | a |
| 2008-01-07 | 7 | 0.22373477 | a |
| 2008-01-08 | 8 | -0.49181182 | a |
| 2008-01-09 | 9 | -0.05175469 | a |
| 2008-01-10 | 10 | -1.38208444 | a |

코드설명

* tsibble로 변환할 데이터 프레임 x를 생성. date 컬럼은 ‘2008년 1월 1일’(as.Data('2008-01-01'))부터 10일후 까지( + 0:9)이고 id 컬럼은 1부터 10까지(1:10), x1은 정규분포 랜덤값 10개(rnorm(10)), x2는 ‘a’(rep('a', 10))로 채움
* as.tsibble()로 x를 tsibble로 변환하는데 key값은 id(key = id), index 값은 date(index = date)로 설정.
* as.tsibble()로 x를 tsibble로 변환하는데 key값은 생략하고, index 값은 date(index = date)로 설정.

## 시계열 데이터 Import

시계열(Time Series) 데이터를 R에서 직접 생성할 수도 있으나 대부분의 경우는 다양한 파일 포맷(Excel, CSV, SPSS 등)으로 저장된 파일에서 읽어 들이는 방법이 일반적이다. 일반적으로 데이터 파일에서 데이터를 읽어 들여 시계열 데이터로 저장하는 방법은 다음과 같은 과정을 거친다.

1. read\_excel(), read.csv()을 사용하여 엑셀 파일을 읽어 데이터를 데이터 프레임에 저장
2. 읽어온 데이터가 적절한 데이터 타입으로 불러들여졌는지 확인하고 적절치 않은 데이터 타입으로 설정된 경우 적절히 변환
3. 읽어온 데이터 중 시간을 기록한 데이터 컬럼을 as.Date()를 사용하여 date 클래스로 변환
4. 시간 컬럼이 생성된 데이터 프레임을 as.ts(), as.xts(), as.tsibble() 등을 사용하여 사용하기 원하는 시계열 객체로 변환

여기에서는 다양한 파일 포맷 중 Excel과 CSV파일에서 시계열 데이터를 읽어 들여 활용하는 방법을 설명한다.

### 엑셀 파일

데이터를 엑셀 파일에서 읽어 들여 데이터 프레임에 저장 후 저장된 데이터 프레임을 시계열 데이터 타입으로 변환할 수 있다.

다음 예제에서 사용하는 자료는 연도별 학교급별 학생수 자료로 한국교육개발원 교육통계 서비스 홈페이지[[8]](#footnote-8)에서 다운로드하였으며 연도별 시계열 데이터 샘플로 사용한다.

read\_excel()을 통해 excel 파일을 바로 읽어 들일 수 있다. read\_excel()을 사용하기 위해서는 먼저 readxl 패키지가 필요하다. 앞에서 소개한 엑셀 데이터를 데이터 프레임으로 읽어 들이는 방법은 아래와 같다. 하나 주의해야하는 점은 col\_type으로 적절한 데이터 타입을 미리 알려주지 않으면 엉뚱한 데이터가 들어온다는 것이다. 이 데이터 파일에서의 문제는 숫자에 천 단위 구분 기호(,)가 포함되어 있기 때문에 해당 컬럼이 numeric이라고 설정해주지 않으면 엉뚱한 데이터 타입(POSIXct)으로 불러들인다는 점이다.

library(readxl)  
students.all <- read\_excel("./students.xlsx", skip = 16, na = '-', sheet = 1, col\_types = c('text', 'text', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric'))  
students <- students.all %>%   
 filter(지역규모 == '계') %>% select(-지역규모)  
head(students) # 데이터 확인

| 연도 | 학생수계 | 유치원 | 초등학교 | 중학교 | 고등학교계 | 일반계고 | 전문계고 | 일반고 | 특목고 | 특성화고 | 자율고 | 특수학교 | 공민학교 | 고등공민 | 고등기술학교 | 각종학교 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1999 | 8658358 | 534166 | 3935537 | 1896956 | 2251140 | 1399389 | 851751 | 0 | 0 | 0 | 0 | 23490 | 150 | 511 | 8399 | 8009 |
| 2000 | 8535867 | 545263 | 4019991 | 1860539 | 2071468 | 1324482 | 746986 | 0 | 0 | 0 | 0 | 23605 | 150 | 297 | 6601 | 7953 |
| 2001 | 8414423 | 545142 | 4089429 | 1831152 | 1911173 | 1259975 | 651198 | 0 | 0 | 0 | 0 | 23769 | 140 | 302 | 5408 | 7908 |
| 2002 | 8361933 | 550256 | 4138366 | 1841030 | 1795509 | 1220146 | 575363 | 0 | 0 | 0 | 0 | 23453 | 145 | 156 | 4911 | 8107 |
| 2003 | 8379775 | 546531 | 4175626 | 1854641 | 1766529 | 1224452 | 542077 | 0 | 0 | 0 | 0 | 24119 | 105 | 150 | 4324 | 7750 |
| 2004 | 8371630 | 541713 | 4116195 | 1933543 | 1746560 | 1232010 | 514550 | 0 | 0 | 0 | 0 | 23876 | 106 | 147 | 3457 | 6033 |

코드 설명

* readxl 패키지를 로딩
* ‘지역규모’ 컬럼이 ’계’인 데이터만 사용할 것이므로 filter()를 사용하여 원하는 데이터만 걸러내고 더이상 지역규모 컬럼이 필요 없으니 제거
* read\_excel()을 이용하여 ‘students.xlsx’파일의 데이터를 읽음. 데이터 파일의 처음 16줄은 skip,(skip = 16)’-’로 표기된 데이터는 NA로 처리(na = '-'), 첫번째 엑셀 시트 데이터를 읽고(sheet = 1), 컬럼별 데이터 타입을 설정

불러들인 데이터는 데이터프레임으로 저장된다. 데이터 프레임도 시계열 객체를 담아 사용할 수 있는데 이를 위해서는 반드시 시간 인덱스로 사용할 컬럼이 필요하다. 불러들인 데이터에는 시간 인덱스로 사용할 수 있는 컬럼이 ‘연도’ 컬럼이다. 이 컬럼을 date 컬럼으로 바꾸는 과정은 다음과 같다.

students$연도 <- as.Date(paste0(students$연도, '-01-01'))

코드 설명

* 연도만 있는 데이터를 date형태로 바꾸려면 먼저 월, 일을 추가해야하기 때문에 paste0()를 사용하여 연도에 ‘-01-01’ 문자열을 붙임.
* ’연도-01-01’로 만들어지 문자열을 as.Date()로 date 클래스로 변환
* 변환된 데이터를 student.from.excel$연도에 저장

연도 컬럼을 date 클래스로 변환한 데이터 프레임을 사용하여 ts, xts, tsibble 시계열 데이터로 변환한다. 하나 주의해야할 사항은 각각의 클래스로 변환하는 함수명이 다르다.

students.ts <- ts(students, frequency = 1, start = 1999)  
students.xts <- as.xts(students[,-1], order.by = students$연도)  
students.tsibble <- students %>%   
 mutate(연도 = yearmonth(paste0(students$연도, '-01-01')))  
students.tsibble <- as\_tsibble(students.tsibble, index = 연도)

코드 설명

* ts()로 student.from.excel 데이터 프레임의 데이터를 ts 객체로 변환하는데 연간 데이터이므로 주기를 1(frequency = 1)로 시작년을 1999년(start = 1999)으로 설정
* as.xts()로 student.from.excel 데이터 프레임의 데이터를 xts 객체로 변환하는데 시간 인덱스를 student.from.excel$연도로 설정(order.by = student.from.excel$연도)
* tsibble로 변환할 데이터프레임을 미리 만들어두는데 시간인덱스로 사용할 컬럼을 yearmonth 클래스로 변환해 둠
* as\_tsibble()을 사용하여 students.tsibble 객체를 tsibble 객체로 변환

### CSV 파일

CSV 파일은 데이터 필드의 구분자를 ’,’로 사용하는 형태의 텍스트 파일로 Comma Seperated Value의 준말이다. 데이터간의 호환을 위해 광범위하게 사용되는 파일 포맷이다.

R에서 CSV 파일을 읽을 수 있는 함수는 여러가지가 있지만 여기서는 read.csv() 을 사용하는 방법을 설명한다.

CSV 파일로 데이터를 읽어 들이는 예제는 월별 데이터는 통계청 KOSIS 홈페이지의 경제활동인구조사의 산업별 취업자수에서 다운받은 월간 취업자수와 교육서비스업 취업자수 데이터를 사용한다. 이 데이터는 통계청 KOSIS 홈페이지에서 다운로드[[9]](#footnote-9)받을 수 있는데 ’국내통계 > 노동 > 경제활동인구조사 > 취업자 > 산업별 취업자’에서 시점을 2013.1월부터 2020.12월까지, 산업구분을 전체 계와 교육서비스업으로 설정 후 행렬전환 기능을 사용하여 행에 기간, 열에 산업을 설정한 후 다운로드 받았는데 필자의 github에서 [다운](https://github.com/2stndard/concept-of-time-series/blob/master/%EC%82%B0%EC%97%85%EB%B3%84_%EC%B7%A8%EC%97%85%EC%9E%90_20210206234505.csv')받을 수 있다.(csv 파일을 다운로드 받는 방법은 다음의 사이트를 참조하길 바란다. [<https://technical-support.tistory.com/54>](https://technical-support.tistory.com/54))

아래와 같이 read.csv()를 통해 csv파일을 읽어들 일수 있다.

employees <- read.csv('./산업별\_취업자\_20210206234505.csv', header = TRUE, na = '-', strip.white = TRUE, stringsAsFactors = TRUE)  
colnames(employees) <- c('time', 'total', 'employees.edu')

코드설명

* read.csv()를 사용하여 csv 파일의 데이터를 읽어오는데 csv 파일의 첫줄은 헤더줄로 설정(header = TRUE), ’-’로 기록된 데이터는 NA로 설정(na = '-'), 빈칸은 제거(strip.white = TRUE), 문자열은 factor로 설정(stringsAsFactors = TRUE)
* 컬럼명을 time, total, employees.edu 로 설정

읽어 들인 데이터가 저장된 employees 데이터프레임에는 시계열 객체에서 사용할 수 있는 시간 인덱스 컬럼이 없다. 따라서 time 컬럼을 date 클래스로 바꿔준다.

employees$time <- as.Date(paste0(employees$time, '. 01'), format = '%Y. %m. %d')

코드설명

* employees$time 컬럼에 저장된 데이터 형태는 ‘연도네자리. 월두자리’의 형태로 되어있기 때문에 우선 ’연도네자리. 월두자리. 일두자리’로 맞추기 위해 paste0()를 사용하여’. 01’을 붙여줌
* 생성된 문자열을 date 클래스로 바꾸기 위해 as.Date()를 사용하는데 문자열 형태가 date 클래스의 기본형태가 아니기 때문에 R이 년월일을 제대로 읽을 수 있도록 format 매개변수를 사용하여 문자열 형태에서 날짜 정보를 추출하기 위한 정보(‘%Y. %m. %d’)를 전달

시간 인덱스로 사용될 date 클래스가 생성되었으므로 ts, xts, tsibble 객체로 변환한다.

employees.ts <- ts(employees, start = c(2013, 01), frequency = 12)  
employees.xts <- xts(employees[,2:3], order.by = employees[,1])  
employees.tsibble <- as\_tsibble(employees, index = time)

코드설명

* ts()를 사용하여 employees를 ts 클래스로 변환하는데 시작월을 2013년 1월로 설정(start = c(2013, 01)), 월별 데이터이기 때문에 시계열 주기를 12(frequency = 12)로 설정
* xts()를 사용하여 employees를 xts 클래스로 변환하는데 사용하는 데이터는 employees의 두번쨰와 세번째 컬럼만을 선택(employees[,2:3])하고 시간인덱스로 첫번째 컬럼(order.by = employees[,1])을 사용한다.
* as\_tsibble()을 사용하여 employees를 tsibble 클래스로 변환하는데 인덱스 컬럼을 time컬럼으로 설정

### 추가 실습 데이터 생성

앞서 생성한 두개의 데이터 셋(students, employees)에 더하여 하나의 추가 데이터 셋을 더 생성할 것이다. 일별 데이터로써 코로나 확진자 수 데이터 셋을 생성할 것이다. 코로나 데이터 셋은 통계청 KOSIS 홈페이지의 코로나 현황 홈페이지에서 다운로드 받았다. 앞서 생성한 연별 데이터(students), 월별 데이터(employees)에 추가적으로 일별 데이터를 생성한다.

#### 일별 데이터 - 코로나 확진자 수

일별 코로나 확진자 수는 통계청 KOSIS 홈페이지에서 다운로드[[10]](#footnote-10)받았고 필자의 github에서 다운로드 받을 수 있다. 이 데이터 셋을 사용하기 위해서는 spread()를 사용하여 긴(long) 형태의 데이터 프레임을 넓은(wide) 형태의 데이터프레임이 필요하다.

먼저 read.csv()를 이용하여 파일의 데이터를 읽어 들인다.

covid19 <- read.csv('./covid19.csv', header = TRUE, na = '-', strip.white = TRUE, stringsAsFactors = TRUE)  
colnames(covid19) <- c('category', 'status', 'date', 'value')  
covid19 <- covid19[, c(3, 1, 2, 4)]

코드 설명

* read.csv()를 사용하여 csv 파일의 데이터를 읽어 오는데 csv 파일의 첫줄은 헤더 줄로 설정(header = TRUE), ’-’로 기록된 데이터는 NA로 설정(na = '-'), 빈칸은 제거(strip.white = TRUE), 문자열은 factor로 설정(stringsAsFactors = TRUE)
* 컬럼명을 ‘category’, ‘status’, ‘date’, ’value’로 설정
* 데이터프레임을 보기쉽게 하기 위해 컬럼의 순서를 바꿈

날짜 컬럼을 만들기 위해 date 컬럼의 데이터를 date 클래스로 바꾼다.

covid19$date <- as.Date(covid19$date, "%Y. %m. %d")

코드 설명

* 읽어들인 데이터의 date 컬럼의 데이터 클래스를 바꾸기 위해 as.Date()를 사용하는데 문자열 형태가 date 클래스의 기본형태가 아니기 때문에 R이 년월일을 제대로 읽을 수 있도록 format 매개변수를 사용하여 문자열 형태에서 날짜 정보를 추출하기 위한 정보(‘%Y. %m. %d’)를 전달

읽어 들인 데이터 중에 연령대별 데이터만 사용하기 위해 category 컬럼에서 ’세’를 포함한 행만 선택하는데 지역구분중에 ’세종’이 있기 때문에 ’세종’만 추가적으로 제외한다.그 다음 데이터에 NA로 기록된 결측치를 처리한다. 결측치를 처리하는 방법으로 여러가지가 있지만 여기서는 일단 0을 넣어주는 방법으로 결측치를 처리하였다.

covid19 <- covid19 %>%   
 filter(grepl('세', category)) %>%   
 filter(category != '세종')  
covid19$value <- ifelse(is.na(covid19$value), 0, covid19$value)

코드 설명

* grepl()은 문자열에서 특정 문자가 존재하면 TRUE를 반환하는 함수임. category에서 ’세’문자가 존재하는 행을 골라내고, 다시 그 결과 중에 ’세종’이 아닌 행만 골라내서 covid19에 저장함
* is.na()로 covid1$value이 NA인지 검사하고 만약(ifelse) NA라면 0을 넣고 아니면 원래값(covid19$value)을 넣어 줌.

이제 사용해야할 데이터의 처리가 완료되었다. 시계열 데이터 처리에는 긴(long) 형태의 데이터보다는 넓은(wide) 형태의 데이터가 좋기 때문에 spread()를 사용하여 긴 형태를 좁은 형태로 바꾸었다.

covid19 <- tidyr::spread(covid19, category, value)

코드 설명

* covid19 데이터프레임의 category의 변수들을 컬럼화하고 해당 컬럼에 넣을 데이터는 value로 채워 넣음

데이터 정리가 다 끝났으므로 ts, xts, tsibble 객체로 변환한다.

covid19.ts <- ts(covid19[, 2:10], frequency = 365)  
covid19.xts <- as.xts(covid19[, 3:10], order.by = covid19$date)  
covid19.tsibble <- as\_tsibble(covid19, index = date)

코드 설명

* ts()를 사용하여 covid19를 ts로 변환하는데 covid19의 두번째부터 10번째컬럼(covid19[, 2:10])까지를 사용하고 일별 데이터이기 때문에 시계열 주기를 365로 설정(frequency = 365)
* xts()를 사용하여 covid19를 xts로 변환하는데 covid19의 세번째부터 10번째컬럼(covid19[, 3:10])까지를 사용하고 시간인덱스를 covid19$date로 설정(order.by = covid19\$date)
* as\_tsibble()을 사용하여 covid19를 tsibble로 변환하는데 시간인덱스를 date로 설정(index = date)

1. Eric Zivo, Working with Financial Time Series Data in R, 2014, <https://faculty.washington.edu/ezivot/econ424/Working%20with%20Time%20Series%20Data%20in%20R.pdf> [↑](#footnote-ref-1)
2. ts, R document , <https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/ts>, 2021.1.31 [↑](#footnote-ref-2)
3. <https://www.rdocumentation.org/packages/xts/versions/0.12.1/topics/as.xts.methods> [↑](#footnote-ref-3)
4. <https://cran.r-project.org/web/packages/xts/xts.pdf> [↑](#footnote-ref-4)
5. <https://tidyverts.org/> [↑](#footnote-ref-5)
6. <https://tidyr.tidyverse.org/articles/tidy-data.html> [↑](#footnote-ref-6)
7. <https://github.com/tidyverts/tsibble> [↑](#footnote-ref-7)
8. <https://kess.kedi.re.kr/userfile/6711778> [↑](#footnote-ref-8)
9. <https://kosis.kr/statisticsList/statisticsListIndex.do?parentId=D.1&menuId=M_01_01&vwcd=MT_ZTITLE&parmTabId=M_01_01&entrType=btn#content-group> [↑](#footnote-ref-9)
10. <https://kosis.kr/covid/covid_index.do> [↑](#footnote-ref-10)