

# 퀀트 투자 족북: R을 이용한 투자 포트폴리오 만들기

이현열

2019-06-13



# Contents

여는 글	5
<b>1 퀀트 투자의 심장: 데이터와 프로그래밍</b>	<b>7</b>
1.1 데이터 구하기	8
1.2 퀀트 투자와 프로그래밍	9
1.3 R 프로그램	10
1.4 퀀트 투자에 유용한 R 패키지	10
<b>2 크롤링을 위한 기본 지식</b>	<b>13</b>
2.1 인코딩의 이해와 R에서 UTF-8 설정하기	13
2.2 웹의 동작 방식	15
2.3 HTML과 CSS 이해하기	16
2.4 파이프 오퍼레이터 (%>%)	25
2.5 오류에 대한 예외처리	26
<b>3 API를 이용한 데이터 수집</b>	<b>29</b>
3.1 API를 이용한 Quandl 데이터 다운로드	29
3.2 getSymbols() 함수를 이용한 API 다운로드	31
<b>4 크롤링 이해하기</b>	<b>39</b>
4.1 GET과 POST 방식 이해하기	39
4.2 크롤링 예제	43
<b>5 금융 데이터 수집하기 (기본)</b>	<b>57</b>
5.1 한국거래소의 산업별 현황 및 개별지표 크롤링	57
5.2 WICS 기준 섹터정보 크롤링	65
<b>6 금융 데이터 수집하기 (심화)</b>	<b>71</b>
6.1 수정주가 크롤링	71
6.2 재무제표 및 가치지표 크롤링	77
6.3 야후 파이낸스 데이터 구하기	86



# 여는 글

본 책의 목적은 다음과 같습니다.

1. R 내에서 API와 크롤링을 이용하여 주가, 재무제표, 가치지표 등의 데이터를 수집합니다.
2. 팩터 모델을 이용한 종목선정과 포트폴리오 최적화에 대해 알아보도록 합니다
3. 백테스트 및 성과를 평가하도록 합니다.

책 내용의 효율적인 전달을 위해 R 혹은 프로그래밍에 대한 지식이 있는 분을 대상으로 하였습니다. 따라서 R과 R Studio 설치, 기본적인 프로그래밍 내용 등은 배제하였으며, 프로그래밍이 처음 이신분은 해당 내용을 먼저 익히신 후 본 책을 읽으시길 추천드립니다.



# Chapter 1

## 퀀트 투자의 심장: 데이터와 프로그래밍

몇 년 전까지만 하더라도 **퀀트 투자**는 일반 투자자들에게 매우 낯선 영역이었던 반면, 최근에는 각종 커뮤니티와 매체를 통해 이제는 익숙한 단어가 되었습니다. 퀀트 투자에서 말하는 **퀀트**란 모형을 기반으로 금융상품의 가격을 산정하거나, 이를 바탕으로 투자를 하는 사람을 말합니다. **퀀트Quant** 라는 단어가 **계량적**을 의미하는 **퀀티타티브Quantitative**에서 앞 글자를 따온 점을 생각하면 쉽게 이해가 될 것입니다.

퀀트 투자 역시 이와 비슷한 의미입니다. 사람들이 일반적으로 투자법이라 알고 있는 산업과 기업을 분석하여 가치를 매기는 **정성적인** 투자법과는 달리, 수학과 통계를 기반으로 투자 전략을 만들고 이를 바탕으로 투자하는 **정량적인** 투자법을 의미합니다.

이처럼 데이터의 수집과 가공, 이를 바탕으로 모델을 만든 후 실행하는 단계는 데이터 사이언스의 업무 흐름도와 매우 유사합니다. **해들리 위컴**<sup>1</sup>Hadley Wickham에 따르면<sup>1</sup>, 데이터 사이언스의 업무 과정은 다음과 같습니다.

데이터 사이언스는 프로그래밍을 통해 데이터를 불러온 후 이를 정리하고, 원하는 결과를 찾기 위해 데이터를 변형하거나, 시각화하거나, 모델링을 합니다. 그 후 이러한 결과를 바탕으로 타인과 소통하는 일련의 과정을 의미합니다.

퀀트 투자의 단계 역시 이와 매우 유사합니다. 투자에 필요한 주가, 재무제표 등의 데이터를 수집하여 정리한 후, 필요한 지표를 얻기 위해 가공을 합니다. 그 후 투자 모형을 이용한 투자 종목을 선택하거나 백테스트를 수행하며, 이를 바탕으로 실제 투자를 하고 성과 평가를 하게 됩니다. 따라서 퀀트 투자는 데이터 사이언스가 금융에 응용된 사례라고도 볼 수 있으며, 그 중심에는 역시 데이터와 프로그래밍이 있습니다.

<sup>1</sup>R을 활용한 데이터 과학(해들리 위컴, 개럿 그롤몬드 저/김설기, 최혜민 역)

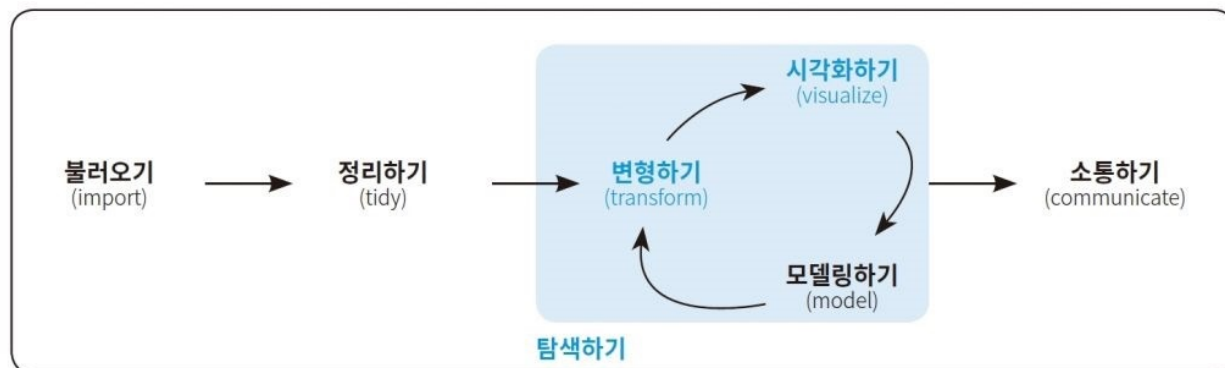


Figure 1.1: 데이터 사이언스 업무 과정

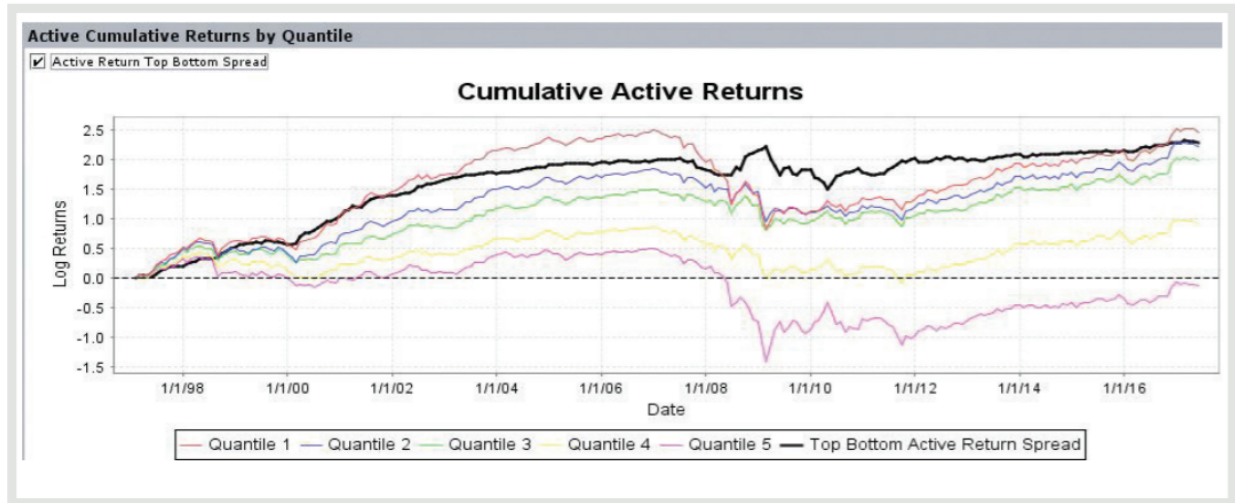


Figure 1.2: ClariFI®의 백테스트 기능

본 책에서도 위의 단계와 동일하게 데이터를 불러오기, 각 데이터 별로 정리하고 가공하기, 시각화를 통해 데이터의 특징 파악하기, 퀀트 모델을 이용하여 종목 선택하기, 백테스트를 실시한 후 성과 및 위험 평가하기에 대해 알아볼 예정입니다. 이에 앞서 본 장에서는 퀀트 투자의 심장이라고 볼 수 있는 데이터는 어떻게 얻을 수 있는지, 왜 프로그래밍을 해야 하는지, 그 중에서도 R은 무엇인지에 대해 간략히 살펴보겠습니다.

## 1.1 데이터 구하기

퀀트 투자에 필요한 데이터의 경우, 여러 데이터 제공업체들이 제공하는 서비스를 이용한다면 매우 손쉽게 구할 수 있습니다. 글로벌 데이터 수집에는 블룸버그 혹은 Factset, 국내 데이터 수집에는 DataGuide가 흔히 사용됩니다. 물론 비용을 더 지불한다면 단순 데이터 수집뿐만 아니라 주식에서 백테스트 및 성과 평가까지 가능합니다. Factset에서 판매하는 Alpha Testing 기능, 혹은 S&P Global에서 판매하는 ClariFI®를 사용한다면, 전세계 주식을 대상으로 원하는 전략의 백테스트 결과를 마우스 클릭과 몇 번의 동작만으로 수행할 수 있습니다.

이러한 데이터 제공업체들을 이용하는 방법의 최대 단점은 바로 비용입니다. 블룸버그 단말기의 경우 1년 사용료가 대리 한 명의 연봉과 비슷하여, 흔히들 **불대리**라 부르기도 합니다. 국내 데이터 업체의 경우 이보다 저렴하기는 하지만, 역시 1년 사용료가 수백 만원 정도로, 일반 개인 투자자들이 감당하기에는 부담이 가는 비용입니다.

해외데이터의 경우 Quandl<sup>2</sup>이나 tiingo<sup>3</sup>등의 업체가 제공하는 서비스를 통해 상대적으로 저렴한 가격으로 데이터를 구할 수 있기도 합니다. 물론 대형 데이터 제공업체에 비해 데이터의 종류나 기간은 짧은 편이지만, 대부분의 퀀트 투자자들이 주식을 대상으로 한다는 점을 생각해보면 충분한 데이터를 얻을 수 있습니다. tiingo에서는 전세계 64,386개 주식의 30년 이상 가격 정보, 21,352개 주식의 12년 이상 재무정보를 월 \$10에 받을 수 있으며, 한정된 종목과 용량에 한해서는 무료로 데이터를 받을 수도 있습니다. 더군다나 API를 통해 프로그램 내에서 직접 데이터를 받을 수 있다는 편의성도 존재합니다.

그러나 아쉽게도 이러한 데이터에서 한국 시장의 정보는 소외가 되어있습니다. 따라서 돈을 들이지 않고 국내 데이터를 얻기 위해서는 직접 발품을 파는 수밖에 없습니다. 야후 파이낸스<sup>4</sup>에서 제공하는 금융데이터를 API를 통해 받을 수도 있으며, 국내 금융 사이트들에서 제공하는 정보를 크롤링하여 데이터를 가공할 수도 있습니다.

아래의 표는 국내의 금융 데이터를 구할 수 있는 사이트의 주소들입니다. 해당 사이트의 정보를 잘만 활용한다면 장기간의 주가 및 재무정보를 무료로 수집할 수 있습니다. 물론 데이터 제공업체가 제공하는 깔끔한 형태의 데이터가 아니므로 클렌징 작업이 필요하다는 점, 그리고 상장폐지된 기업의 경우 데이터를 구하는 점이 힘들다는 단점이 있습

<sup>2</sup><https://www.quandl.com/>

<sup>3</sup><https://www.tiingo.com/>

<sup>4</sup><https://finance.yahoo.com/>





Figure 1.3: NAVER 금융 제공 재무정보

니다. 그러나 비용이 들지 않는다는 점, 그리고 현재 시점에서 투자 종목을 선택할 때는 상장폐지된 기업이 필요하지 않는다는 점을 고려하면, 이는 큰 문제가 되지 않는다 생각합니다.

## 1.2 퀀트 투자와 프로그래밍

우리가 구한 데이터는 연구나 투자에 바로 사용할 수 있는 형태로 주어지는 경우가 거의 없기 때문에 이를 목적에 맞게 처리하는 과정을 거쳐야 하며, 이를 흔히 데이터 클렌징 작업이라 합니다. 또한, 데이터가 정제된 이후 이를 활용한 투자 전략의 백테스트나 종목 선정을 위해서도 프로그래밍은 필수입니다. 물론 모든 퀀트 투자에서 프로그래밍이 필수인 것은 아닙니다. 엑셀을 이용하여도 간단한 형태의 백테스트 및 종목 선정은 얼마든지 가능합니다. 그러나 응용성 및 효율성의 측면에서 엑셀을 이용하는 것은 매우 비효율적입니다.

데이터를 수집하고 클렌징 작업을 하는 경우, 몇 종목 되지 않는다면 엑셀을 이용하여도 충분히 가능합니다. 그러나 종목 수가 수 천 종목을 넘어갈 경우, 데이터를 손으로 일일이 처리하는 것은 사실상 불가능에 가깝습니다. 이러한 단순 반복 작업의 경우 프로그래밍을 이용한다면 훨씬 효율적으로 작업을 수행할 수 있습니다.

백테스트에서도 프로그래밍을 사용하는 것이 훨씬 효율적입니다. 과거 12개월 누적 수익률이 높은 30종목에 투자하는 모멘텀 전략의 백테스트를 한다고 가정합니다. 처음에는 엑셀을 통해 백테스트를 하는 것이 편하다고 생각될 수 있습니다. 그러나 만일 12개월이 아닌 6개월 누적 수익률로 백테스트를 하고자 한다면 어떨까요? 다시 6개월 누적 수익률을 구하는 작업을 위해 명령어를 바꾸고 드래그를 해야 할 것입니다. 그러나 프로그래밍을 이용한다면  $n = 12$  였던 부분을  $n = 6$ 으로 변경한 후, 단지 클릭을 하면 새로운 백테스트가 완료됩니다.

전체 데이터가 100MB 정도라 가정할 때, 투자 전략이 계속해서 늘어날 경우는 어떨까요? 엑셀에서 A라는 전략을 백테스트 하기 위해서는 해당 데이터를 이용하여 작업을 한 후 저장합니다. 그 후 B라는 전략을 새롭게 백테스트 하려면 해당 데이터를 새로운 엑셀 파일에 복사하여 작업한 후 저장해야 합니다. 결과적으로 10개의 전략만 백테스트 하더라도 100MB 짜리 엑셀파일이 10개, 즉 1GB 정도의 엑셀 파일이 쌓이게 됩니다. 만일 데이터가 바뀔 경우, 다시 10개 엑셀 시트의 데이터를 일일이 바꿔야 하는 귀찮음도 감수해야 합니다. 물론 하나의 엑셀 파일 내에서 모든 전략을 수행할 수도 있지만, 이는 엄청난 속도 저하의 문제가 있습니다.

그러나 프로그래밍을 이용한다면 어떨까요? 백테스트를 수행하는 프로그래밍 스크립트는 불과 몇 KB에 불과합니다. 10개의 전략에 대한 스크립트 파일을 합해도 1MB가 되지 않습니다. 데이터가 바뀌더라도 원본 데이터 파일 하나만

수정해주면 됩니다.

물론 대부분의 사람들에게 프로그래밍은 매우 낯선 도구입니다. 그러나 퀀트 투자에 필요한 프로그래밍은 매우 한정적이고 반복적이기에, 몇 개의 단어와 구문만 익숙해지면 사용하는데 큰 어려움이 없습니다. 또한 개발자들의 프로그래밍에 비하면 상당히 쉬운 수준이므로, 비교적 빠른 시간 내에 원하는 전략을 테스트하고 수행하는 정도의 능력을 갖출 수도 있습니다.

최근에는 프로그래밍을 공부할 수 있는 좋은 온라인 사이트 및 학원도 늘어나고 있으므로, 프로그래밍을 처음 시작하기에도 큰 어려움이 없습니다. DataCamp<sup>5</sup>는 사람들이 즐겨 이용하는 온라인 프로그래밍 학원으로써 **R**, **Python**, **SQL**과 같은 언어를 단계별로 배워 나갈 수 있으며, 저렴한 사용료로 모든 과정을 수강할 수 있습니다. 직접 코드를 입력해야 과정이 진행된다는 점에서 초보자들도 능동적으로 학습할 수 있는 장점이 있습니다.

## 1.3 R 프로그램

인간이 사용하는 언어의 종류가 다양하듯이, 프로그램 언어의 종류 역시 매우 다양합니다. 대략 700여개 이상의 프로그램 언어<sup>6</sup> 사람들이 대중적으로 사용하는 언어는 그리 많지 않으므로, 대중성과 효율성을 위해 사용량이 많은 언어를 이용하는 것이 좋습니다.

아래 그림은 프로그래밍 언어의 사용 통계 순위<sup>7</sup>입니다. 이 중 R과 Python은 금융에서 대표적으로 사용되는 언어로써, 매우 대중적인 언어이기도 합니다. 해당 언어가 많이 사용되는 가장 큰 이유는 무료인 점, 그리고 일반인들도 사용하기에 매우 편한 형태로 언어가 구성되어 있다는 점입니다.

이러한 프로그래밍 언어 중 본 책에서는 R을 이용하였습니다. R의 장점은 무료라는 점 이외에도, 타 언어는 비교할 수 없는 다양한 패키지입니다. 두터운 사용자 층을 기반으로 하여 R에는 상상할 수 없을 정도로 다양한 패키지가 존재하며, 특히 통계나 계량분석과 관련된 패키지는 독보적이라 할 수 있습니다.

## 1.4 퀀트 투자에 유용한 R 패키지

여러 연구자 및 실무자들의 헌신적인 노력과 함께, R에는 금융 연구와 퀀트 투자를 위한 다양한 패키지들이 만들어져 있으며, 누구나 무료로 이용이 가능합니다. 그 중 해당 책의 내용에 사용되는 패키지 중 중요하다고 생각되는 것 위주로 소개하도록 하겠습니다. 각 패키지에 대한 자세한 설명은 구글에서 패키지명으로 검색한 후, PDF 파일을 통해 확인할 수 있습니다.

- **quantmod**: 이름에서 알 수 있듯이 퀀트 투자에 매우 유용한 패키지입니다. API를 이용하여 데이터를 다운로드 받는 `getSymbols()` 함수는 너무나 많이 사용되는 함수이며, 이 외에도 볼린저밴드, 이동평균선, 상대 강도 지수 RSI 등 여러 기술적 지표들을 주가 차트에 나타낼 수도 있습니다.
- **PerformanceAnalytics**: 포트폴리오의 성과와 위험을 측정하는데 역시나 매우 유용한 패키지입니다. 백테스트에 사용되는 `Return.portfolio()` 함수는 포트폴리오 단위의 백테스트에 필수적인 함수입니다. 또한 성과를 측정하는 상당히 많은 함수를 이용하여, 포트폴리오의 성과뿐만 아니라 위험을 파악할 수 있습니다.
- **xts**: 기본적으로 금융 데이터는 시계열 형태이며, 해당 패키지는 여러 데이터들을 시계열 형태 `eXtensible Time Series`로 변형시켜 줍니다. 일별 수익률을 월별 수익률 혹은 연도별 수익률로 변환하는 `apply.monthly()`와 `apply.yearly()` 함수, 데이터들의 특정 시점을 찾아주는 `endpoints()` 함수 역시 백테스트에 필수적으로 사용되는 함수입니다. 해당 패키지는 **PerformanceAnalytics** 패키지 설치 시 자동으로 설치됩니다.
- **zoo**: 해당 패키지 역시 시계열 데이터를 다루는데 유용함 함수가 존재합니다. `rollapply()` 함수는 `apply()` 함수를 전체 데이터가 아닌 롤링-윈도우 기법으로 활용할 수 있게 해주며, na 데이터를 채워주는 `na.locf()` 함수는 시계열 데이터의 결측치를 보정할 때 매우 유용합니다.

<sup>5</sup><https://www.datacamp.com/>

<sup>6</sup>[https://en.wikipedia.org/wiki/List\\_of\\_programming\\_languages](https://en.wikipedia.org/wiki/List_of_programming_languages)

<sup>7</sup><https://www.tiobe.com/tiobe-index/>

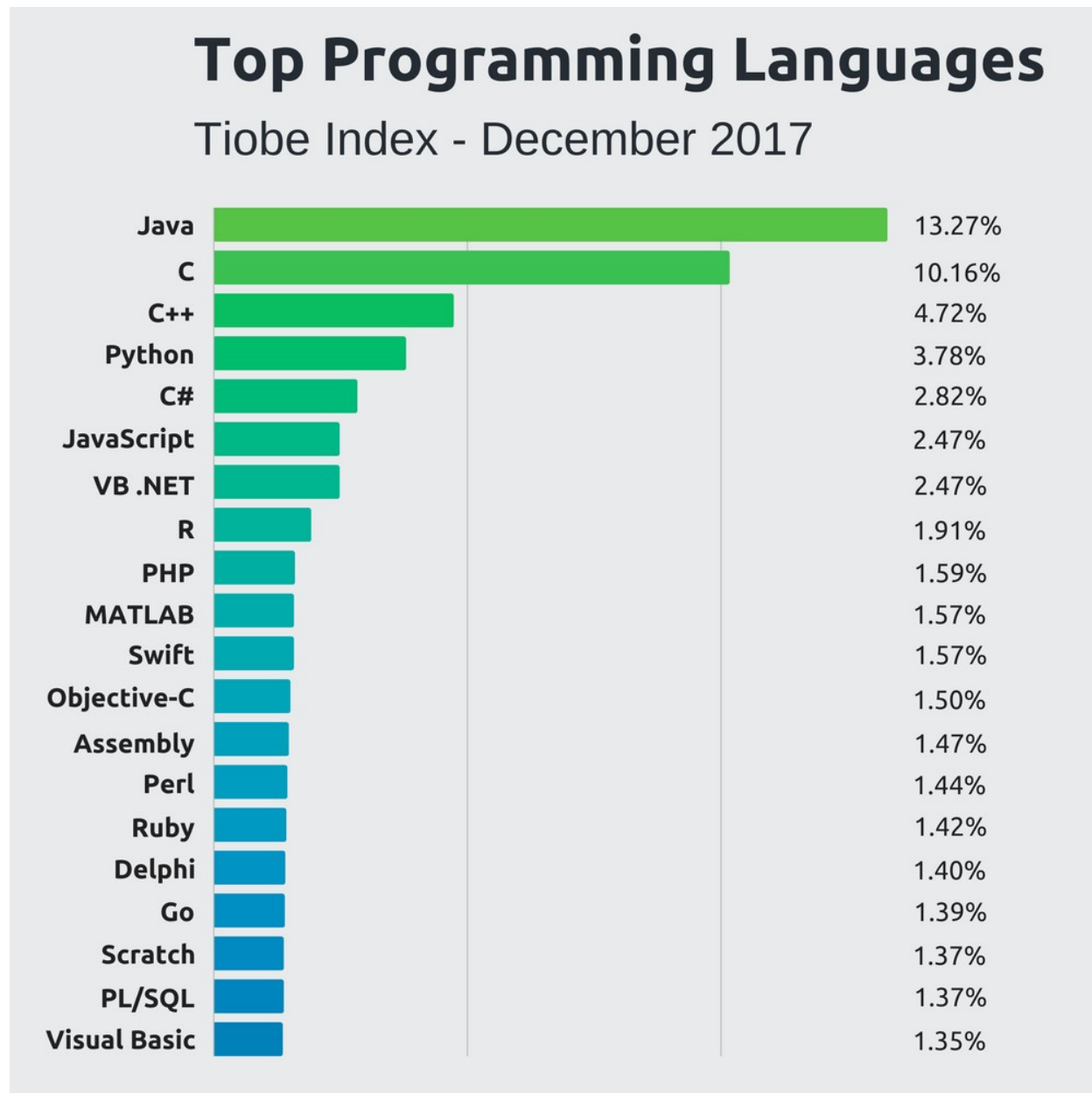


Figure 1.4: 2017년 기준 프로그래밍 언어 사용 통계 순위

- **httr & rvest**: 데이터를 웹에서 직접 수집하기 위해서는 크롤링이 필수이며, **httr**과 **rvest**는 이에 사용되는 패키지입니다. **httr**의 경우 http의 표준 요청을 수행해주는 패키지로써 단순히 데이터를 받는 **GET()** 함수와 사용자가 필요한 값을 선택하여 요청하는 **POST()** 함수가 대표적으로 사용됩니다. **rvest**의 경우 html 문서의 데이터를 가져오는데 사용되는 패키지이며, 웹 페이지에서 데이터를 크롤링 한 후 원하는 데이터만 뽑는데 필요한 여러 함수들이 포함되어 있습니다.
- **dplyr**: 해당 패키지는 데이터 처리에 특화된 패키지로써, R을 이용한 데이터 과학 분야에서 가장 많이 사용되는 패키지 중 하나입니다. C++로 작성되어 매우 빠른 처리속도를 보이며, API나 크롤링을 통해 수집한 데이터들을 정리할 때도 매우 유용하게 사용됩니다.
- **ggplot2**: 데이터를 시각화할 때 가장 많이 사용되는 패키지입니다. 물론 R에서 기본적으로 내장된 **plot()** 함수를 이용하여도 시각화가 가능하지만, 해당 패키지를 이용할 경우 훨씬 다양하고 깔끔하게 데이터를 그림으로 표현할 수 있습니다.

## Chapter 2

# 크롤링을 위한 기본 지식

## 2.1 인코딩의 이해와 R에서 UTF-8 설정하기

### 2.1.1 인간과 컴퓨터 간 번역의 시작, ASCII

R에서 스크립트를 한글로 작성하여 저장한 후 이를 다시 불러올 때, 혹은 한글로된 데이터를 크롤링하면 오류가 뜨거나 읽을 수 없는 문자로 나타나는 경우가 종종 있습니다. 이는 한글 인코딩 때문에 발생하는 문제이며, 이러한 현상을 흔히 **인코딩이 깨졌다**고 표현합니다. 인코딩이란 사람이 사용하는 언어를 컴퓨터가 사용하는 0과 1로 변환하는 과정을 말하며, 이와 반대의 과정을 디코딩이라고 합니다.

이러한 사람과 컴퓨터간의 번역을 위해 최초로 사용된 방식이 아스키(ASCII: American Standard Code for Information Interchange)입니다. 0부터 127까지 총 128개 바이트에 알파벳과 숫자, 그리고 자주 사용되는 특수문자 값을 부여하고, 글자가 입력되면 이에 대응되는 바이트가 저장됩니다. 그러나 아스키의 American이라는 이름에서 알 수 있듯이 이는 영어의 알파벳이 아닌 다른 언어를 표현하는데는 한계가 있으며, 이를 보완하기 위한 여러 방법들이 나오게 되었습니다.

### 2.1.2 한글 인코딩의 종류

인코딩에 대한 전문적인 내용의 경우 해당 책에서 다룰 분야도 아니며, 한글을 인코딩하는데 쓰이는 EUC-KR과 CP949, 그리고 UTF-8 정도만 알고 넘어가도 충분합니다. 만일 **알**이라는 단어를 인코딩하기 위해서는 어떠한 방법이 있을까요? 먼저 **알**이라는 문자 자체에 해당하는 코드를 부여하여 나타내는 방법이 있습니다. 아니면 이를 구성하는 모음과 자음을 나누어 **ㅇ**, **ㄴ**, **ㄹ** 각각에 해당하는 코드를 부여하고 이를 조합할 수도 있습니다. 전자와 같이 완성된 문자 자체로 나타내는 방법을 완성형, 후자와 같이 각 문자로 나타내는 방법을 조합형이라고 합니다.

먼저 한글 인코딩 중 완성형으로 가장 대표적인 방법은 EUC-KR이며, 이는 현대 한글에서 많이 쓰이는 글자 2,350개에 번호를 붙인 방법입니다. 그러나 2,350개 글자로 모든 한글의 조합을 표현하기는 부족하였으며, 이를 보완하고자 마이크로소프트사가 도입한 방법이 CP949입니다. CP949는 11,720개 한글에 번호를 붙인 방법으로 기존 EUC-KR보다 나타낼 수 있는 한글의 갯수가 훨씬 많아졌습니다. 윈도우의 경우 기본 인코딩이 CP949로 되어 있습니다.

조합형의 대표적 방법으로는 UTF-8이 있습니다. 이는 모음과 자음 각각에 코드를 부여한 후 조합하여 한글을 나타냅니다. 조합형의 경우 한글뿐만이 아니라 다양한 언어에 적용할 수 있다는 장점으로 인해 전세계 웹사이트의 대부분이 UTF-8로 만들어 지고 있습니다.

### 2.1.3 R에서 UTF-8 설정하기

위에서 언급했듯이 윈도우에서는 기본 인코딩이 CP949로 이루어져 있으며, 일부 국내 홈페이지는 EUC-KR로 인코딩이 된 경우도 있습니다. 반면 R의 여러 함수들은 인코딩이 UTF-8로 이루어져 있기에, 이러한 인코딩 방식의

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	null	32	20	space	64	40	@	96	60	`
1	01	start of heading	33	21	!	65	41	A	97	61	a
2	02	start of text	34	22	"	66	42	B	98	62	b
3	03	end of text	35	23	#	67	43	C	99	63	c
4	04	end of transmit	36	24	\$	68	44	D	100	64	d
5	05	enquiry	37	25	%	69	45	E	101	65	e
6	06	acknowledge	38	26	&	70	46	F	102	66	f
7	07	audible bell	39	27	'	71	47	G	103	67	g
8	08	backspace	40	28	(	72	48	H	104	68	h
9	09	horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	line feed (¶n)	42	2A	*	74	4A	J	106	6A	j
11	0B	vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	carriage return (¶r)	45	2D	-	77	4D	M	109	6D	m
14	0E	shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	shift in	47	2F	/	79	4F	O	111	6F	o
16	10	data link escape	48	30	0	80	50	P	112	70	p
17	11	device control 1	49	31	1	81	51	Q	113	71	q
18	12	device control 2	50	32	2	82	52	R	114	72	r
19	13	device control 3	51	33	3	83	53	S	115	73	s
20	14	device control 4	52	34	4	84	54	T	116	74	t
21	15	neg acknowledge	53	35	5	85	55	U	117	75	u
22	16	synchronous idle	54	36	6	86	56	V	118	76	v
23	17	end of trans. block	55	37	7	87	57	W	119	77	w
24	18	cancel	56	38	8	88	58	X	120	78	x
25	19	end of medium	57	39	9	89	59	Y	121	79	y
26	1A	substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	escape	59	3B	;	91	5B	[	123	7B	{
28	1C	file separator	60	3C	<	92	5C	₩	124	7C	
29	1D	group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	unit separator	63	3F	?	95	5F	_	127	7F	DEL

Figure 2.1: 아스키 코드 표

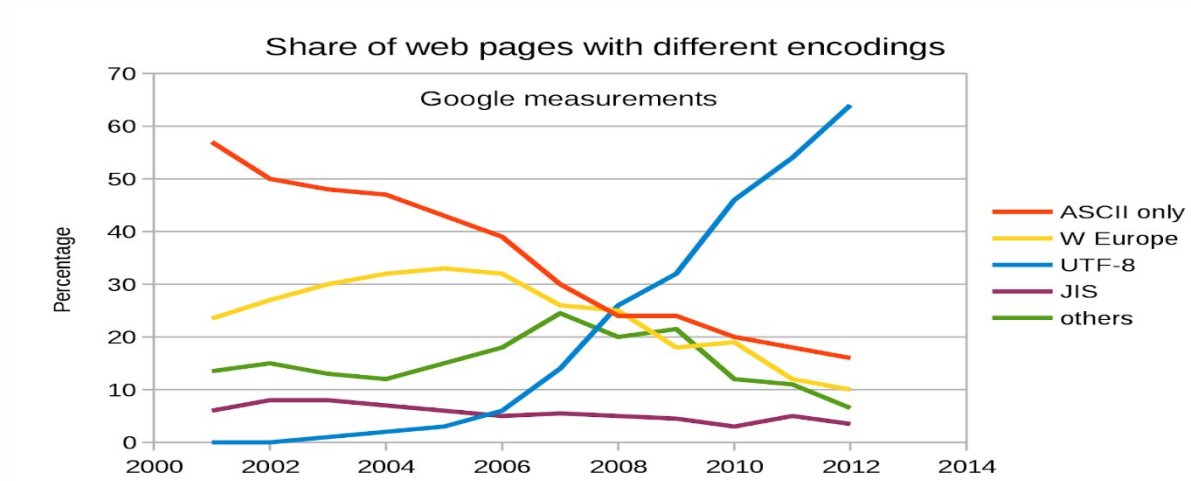


Figure 2.2: 웹페이지에서 사용되는 인코딩 비율



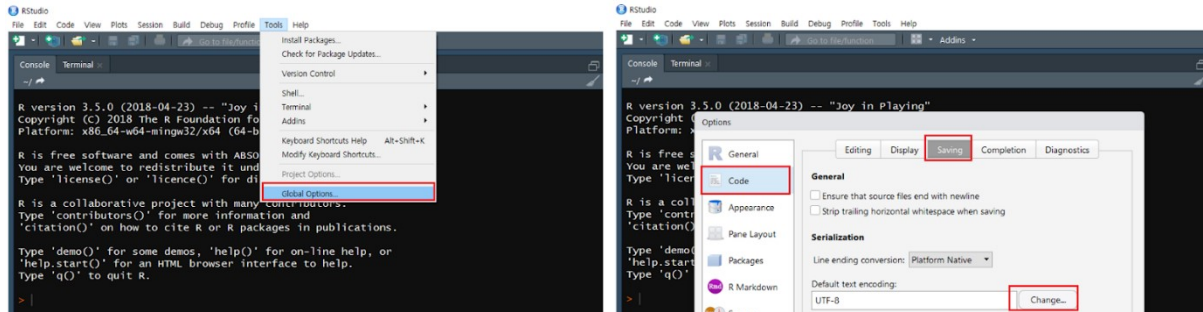


Figure 2.3: 인코딩 설정 항목

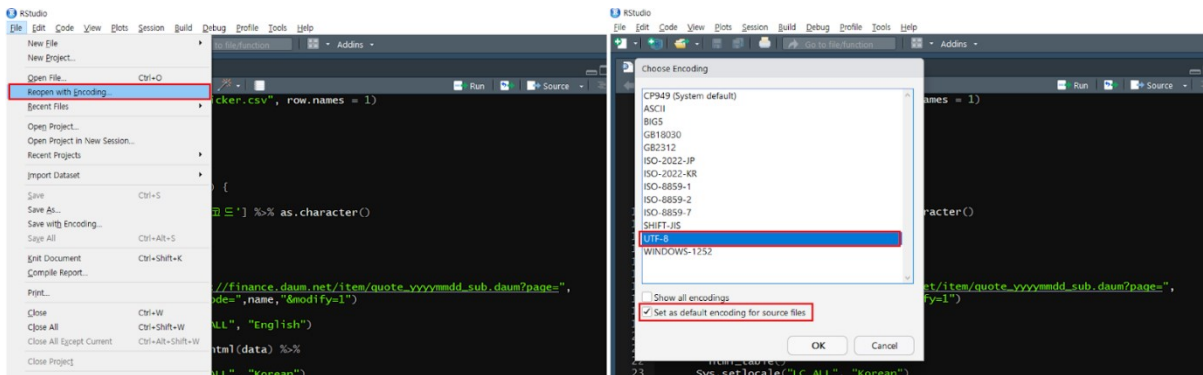


Figure 2.4: 인코딩 설정 항목

차이로 인해 스크립트 작성 및 크롤링 과정에서 오류가 발생하는 경우가 종종 있습니다. 윈도우 컴퓨터에서는 CP949가 기본으로 설정되어 있습니다.

만일 CP949 인코딩을 그대로 사용할 경우, 미리 저장되었던 한글 스크립트가 깨져 나오는 일이 발생할 수 있습니다. 이를 위해 기본 인코딩을 UTF-8로 변경해주는 것이 좋습니다. R Studio의 Tools → Global Options 메뉴에서 Code → Saving 항목 중 Default text encodings 항목을 통해 기본 인코딩을 UTF-8로 변경해주시도록 합니다.

해당 방법으로도 해결되지 않을 경우 File → Reopen with Encoding 메뉴에서 UTF-8 항목을 선택, Set as default encoding for source files 항목을 선택한 후 OK를 누르면 UTF-8로 인코딩이 설정된 후 파일을 다시 열게 됩니다.

## 2.2 웹의 동작 방식

크롤링은 웹사이트의 정보를 수집하는 과정이니 만큼, 웹이 어떻게 동작하는지 알아둘 필요가 있습니다.

먼저 클라이언트는 여러분의 데스크탑이나 휴대폰과 같은 장치, 그리고 이런 장치의 크롬이나 파이어폭스와 같은 소프트웨어를 의미합니다. 반대로 서버는 웹사이트, 앱을 저장하는 컴퓨터를 의미합니다. 클라이언트가 특정 정보를 요구하는 과정을 **요청**이라 하며, 서버가 해당 정보를 제공하는 과정을 **응답**이라고 합니다. 그러나 클라이언트와 서버가 연결되어 있지 않다면 둘 간에 정보를 주고 받는 것은 불가능하며, 이를 연결해주는 공간이 바로 인터넷입니다. 또한 건물에도 고유의 주소가 있는 것처럼, 각 서버에도 고유의 주소가 있으며, 이것이 인터넷주소 혹은 URL 입니다.

여러분이 네이버에서 경제 기사를 클릭하는 경우를 생각해 봅시다. 클라이언트는 사용자인 여러분, 서버는 네이버이며, URL은 [www.naver.com](http://www.naver.com) 이 됩니다. 경제 기사를 클릭하는 과정이 요청이며, 클릭 후 해당 페이지를 보여주는 과정이 응답입니다.

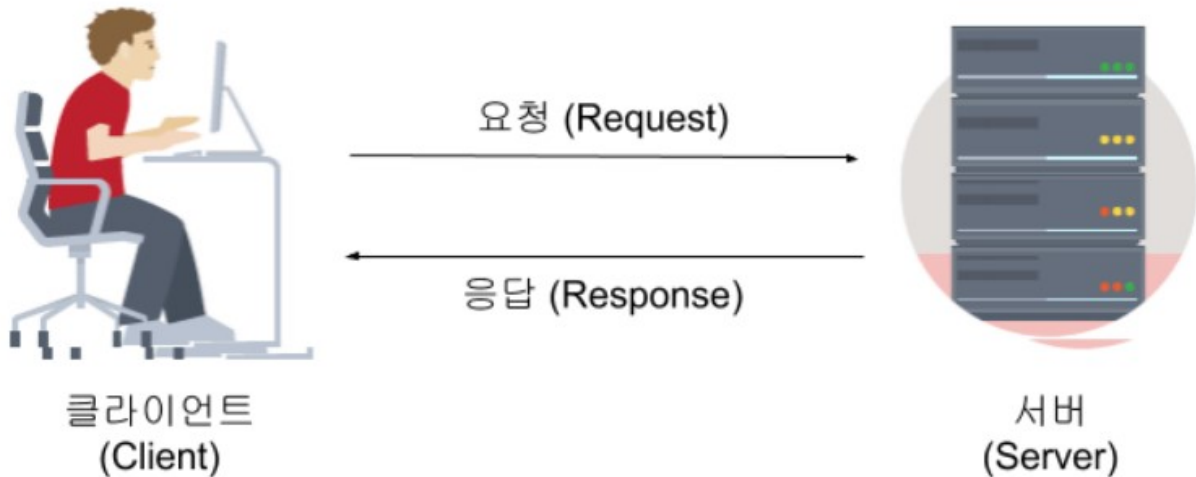


Figure 2.5: 웹 환경 구조

Table 2.1: HTTP 요청 방식과 설명

요청방식	주소
GET	특정 정보 조회
POST	새로운 정보 등록
PUT	기존 특정 정보 갱신
DELETE	기존 특정 정보 삭제

### 2.2.1 HTTP

클라이언트가 각기 다른 방법으로 데이터를 요청한다면, 서버는 해당 요청을 알아듣지 못할 것입니다. 이를 방지하기 위해 규정된 약속이나 표준에 맞추어 데이터를 요청해야하며, 이러한 약속을 HTTP(HyperText Transfer Protocol)라 합니다.

클라이언트가 서버에게 요청의 목적이나 종류를 알리는 방법을 HTTP 요청 방식(HTTP Request Method)이라고 합니다. 이는 크게 GET, POST, PUT, DELETE 4가지로 나눌 수 있지만 크롤링에는 GET과 POST 방식이 대부분 사용되므로 이 두가지만 아는 것도 충분합니다. GET 방식과 POST 방식에 대한 차이 및 크롤링 방법은 데이터 수집 파트에서 자세하게 다루도록 하겠습니다.

인터넷을 사용하다 보면 한번쯤 **이 페이지를 볼 수 있는 권한이 없습니다.(HTTP 오류 403 - 사용할 수 없음)** 혹은 **페이지를 찾을 수 없음(HTTP 오류 404 - 파일을 찾을 수 없음)**이라는 오류가 발생한 적이 있을 겁니다. 여기서 403과 404이라는 숫자는 클라이언트의 요청에 대한 서버의 응답 상태를 나타내는 코드이며, 이를 **HTTP 상태 코드**라 합니다.

HTTP 상태 코드는 100번대 부터 500번대 까지 있으며, 성공적으로 응답을 받을 시 200번 코드를 받게 됩니다. 각 코드에 대한 내용은 HTTP 상태 코드를 검색하면 확인할 수 있으며, 크롤링 과정에서 오류가 발생할 시 해당 코드를 통해 어떤 부분에서 오류가 발생하였는지 확인이 가능합니다.

## 2.3 HTML과 CSS 이해하기

클라이언트와 서버가 데이터를 주고 받을때는 디자인이라는 개념이 필요하지 않습니다. 그러나 응답 받은 정보를 사람이 확인하기 위해서는 보기 편한 방식으로 바꾸어 줄 필요가 있으며, 웹페이지가 그러한 역할을 합니다. 웹페이지의 제목, 단락, 목록 등 레이아웃을 잡아주는데 쓰이는 대표적인 마크업 언어가 HTML(HyperText Markup Language)



Table 2.2: HTTP 상태 코드 그룹 별 내용

코드	주소	내용
1xx	Informational (조건부 응답)	리퀘스트를 받고, 처리 중에 있음
2xx	Success (성공)	리퀘스트를 정상적으로 처리함
3xx	Redirection (리디렉션)	리퀘스트 완료를 위해 추가 동작이 필요함
4xx	Client Error (클라이언트 오류)	클라이언트 요청을 처리할 수 없어 오류 발생
5xx	Server Error (서버 오류)	서버에서 처리를 하지 못하여 오류 발생

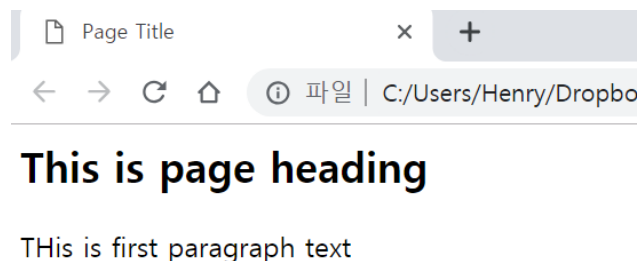


Figure 2.6: HTML 기본 구조

입니다. HTML을 통해 잡혀진 뼈대에 글자의 색상이나 폰트, 배경 색, 배치 등 화면을 꾸며주는 역할을 하는 것이 CSS(Cascading Style Sheets) 입니다.

우리의 목적은 웹페이지를 만드는 것이 아니기에 HTML과 CSS에 대해 지나치게 자세히 알 필요는 없습니다. 그러나 크롤링 하고자 하는 데이터가 페이지의 어떤 태그 내에 위치하고 있는지, 어떻게 크롤링을 하면 될지 파악하기 위해서는 HTML과 CSS에 대한 기본적인 지식은 알아둘 필요가 있습니다.

### 2.3.1 HTML 기본 구조

HTML은 크게 메타 데이터를 나타내는 <head> 부분과 본문을 나타내는 <body> 부분으로 나누어집니다. <head>에서 <title>은 웹페이지에서 나타나는 제목을 나타내며 <body> 내에는 본문에 들어갈 각종 내용들이 포함되어 있습니다.

```
<html>
<head>
<title>Page Title</title>
</head>

<body>
<h2> This is page heading </h2>
<p> THIS is first paragraph text </p>
</body>
</html>
```

### 2.3.2 태그와 속성

HTML 코드는 태그와 속성, 그리고 내용으로 이루어져 있습니다. 크롤링한 데이터에서 특정 태그의 데이터만을 찾는 방법, 특정 속성의 데이터만을 찾는 방법, 짧은 자료에서 내용만을 찾는 방법등의 내용을 찾는 방법이 모두 다르기 때문에 이러한 요소에 대해 좀더 자세히 살펴보도록 하겠습니다.

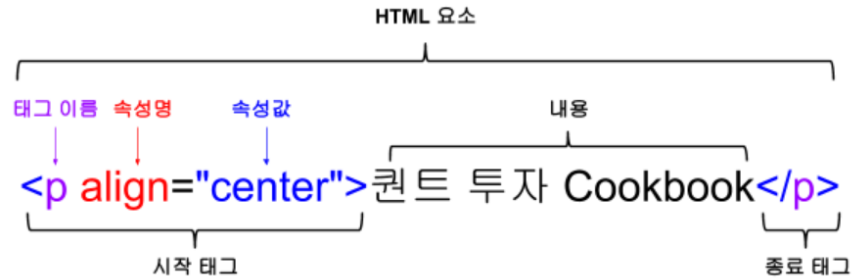


Figure 2.7: HTML 구성 요소 분석

꺾쇠(<>)로 감싸져 있는 부분을 태그라 부르며, 여는 태그 <>가 있으면 반드시 이를 닫아주는 태그인 </>가 쌍으로 존재해야 합니다. 속성은 해당 태그에 대한 추가적인 정보를 제공해주는 것으로써, 뒤에 속성값이 따라와야 합니다. 내용은 우리가 눈으로 보는 텍스트 부분을 의미합니다. 위의 HTML 코드는 문단을 나타내는 <p> 태그, 정렬을 나타내는 align 속성과 center를 통해 가운데 정렬을, 내용에는 ‘퀀트 투자 Cookbook’을, 그리고 태그를 </p>를 통해 태그를 마쳤습니다.

### 2.3.3 h 태그와 p 태그

h 태그는 폰트의 크기를 나타내는 태그이며, p 태그는 문단을 나타내는 태그입니다. 이를 사용한 간단한 예제는 다음과 같습니다. h 태그의 숫자가 작을수록 텍스트의 크기는 커지는 것이 확인되며, 숫자는 1에서 6까지 지원이 됩니다. p 태그를 사용할 경우 각각의 문단이 만들어지는 것이 확인됩니다.

```
<html>
<body>

<h1>Page heading: size 1</h1>
<h2>Page heading: size 2</h2>
<h3>Page heading: size 3</h3>

<p>Quant Cookbook</p>
<p>By Henry</p>

</body>
</html>
```

### 2.3.4 리스트: ul과 ol 태그

ul과 ol 태그는 리스트(글머리 기호)를 만들 때 사용되며, ul은 경우 순서가 없는(unordered list), ol의 경우 순서가 있는(ordered list)를 만듭니다.

```
<html>
<body>

<h2> Unordered List</h2>
<ul>
  <li>Price</li>
  <li>Financial Statement</li>
  <li>Sentiment</li>
```

# Page heading: size 1

## Page heading: size 2

### Page heading: size 3

Quant Cookbook

By Henry

Figure 2.8: h 태그와 p 태그 예제

```

</ul>

<h2> Ordered List</h2>
<ol>
  <li>Import</li>
  <li>Tidy</li>
  <li>Understand</li>
  <li>Communicate</li>
</ol>

</body>
</html>

```

ul 태그로 감싸진 부분은 글머리 기호가 순서가 없는 •으로 표현되었으며, ol 태그로 감싸진 부분은 숫자가 순서대로 표현되었습니다. 각각의 리스트는 li를 통해 생성하게 됩니다.

### 2.3.5 table 태그

table 태그는 표를 만드는 태그입니다.

```

<html>
<body>

<h2>Major Stock Indices and US ETF</h2>

<table>
  <tr>
    <th>Country</th>
    <th>Index</th>
    <th>ETF</th>
  </tr>
  <tr>
    <td>US</td>
    <td>S&P 500</td>
    <td>IVV</td>
  </tr>
</table>

```

## Unordered List

- Price
- Financial Statement
- Sentiment

## Ordered List

1. Import
2. Tidy
3. Understand
4. Communicate

Figure 2.9: 리스트 관련 태그 예제

```

</tr>
<tr>
  <td>Europe</td>
  <td>Euro Stoxx 50</td>
  <td>IEV</td>
</tr>
<tr>
  <td>Japan</td>
  <td>Nikkei 225</td>
  <td>EWJ</td>
</tr>
<tr>
  <td>Korea</td>
  <td>KOSPI 200</td>
  <td>EWY</td>
</tr>
</table>

</body>
</html>

```

table 태그 내의 tr 태그는 각 행을 의미합니다. 각 셀의 구분은 th 혹은 td 태그를 통해 구분이 가능하며, th 태그는 진하게 표현되므로 주로 테이블의 제목에, td 태그는 테이블의 내용에 사용됩니다.

### 2.3.6 a, src 태그와 속성

a 태그와 src 태그는 다른 태그와는 다르게, 혼자 쓰이기 보다는 속성과 결합하여 사용됩니다. 먼저 a 태그는 href 속성과 결합하여 다른 페이지의 링크를 걸 수 있습니다. src 태그는 img 속성과 결합하여 이미지를 불러옵니다.

## Major Stock Indices and US ETF

Country	Index	ETF
US	S&P 500	IVV
Europe	Euro Stoxx 50	IEV
Japan	Nikkei 225	EWJ
Korea	KOSPI 200	EWY

Figure 2.10: table 태그 예제

### a tag & href attribute

HTML links are defined with the a tag. The link address is specified in the href attribute:

[Henry's Quantopia](https://henryquant.blogspot.com/)

### img tag & src attribute

HTML images are defined with the img tag, and the filename of the image source is specified in the src attribute:



Figure 2.11: a 태그와 src 태그 예제

```
<html>
<body>

<h2>a tag & href attribute</h2>
<p>HTML links are defined with the a tag. The link address is specified in the href attribute:</p>

<a href="https://henryquant.blogspot.com/">Henry's Quantopia</a>

<h2>img tag & src attribute</h2>
<p>HTML images are defined with the img tag,
and the filename of the image source is specified in the src attribute:</p>



</body>
</html>
```

a 태그 뒤 href 속성에 대한 속성값으로 연결하고자 하는 웹사이트의 주소를 입력한 후, 내용을 입력하면, 해당 텍스트에 웹사이트의 링크가 추가됩니다. img 태그 뒤 src 속성에는 불러오고자 하는 이미지의 주소를 입력하며, width 속성과 height 속성을 통해 가로와 세로 길이를 조절할 수도 있습니다. 페이지 내에서 링크된 주소를 모두 찾거나, 혹은 모든 이미지를 저장하는 작업을 하고자 할 시, 이러한 속성값을 찾으려면 손쉽게 원하는 작업을 할 수 있습니다.

### 2.3.7 div 태그

div 태그는 화면의 전체적인 틀(레이아웃)을 만들 때 주로 사용하는 태그입니다. 단독으로도 사용될 수 있으며, 꾸밈을 담당하는 style 속성과 결합되어 사용되기도 합니다.



Figure 2.12: div 태그 예제

```
<html>
<body>

<div style="background-color:black;color:white">
  <h5>First Div</h2>
  <p>Black backgrond, White Color</p>
</div>

<div style="background-color:yellow;color:red">
  <h5>Second Div</h2>
  <p>Yellow backgrond, Red Color</p>
</div>

<div style="background-color:blue;color:grey">
  <h5>Second Div</h2>
  <p>Blue backgrond, Grey Color</p>
</div>

</body>
</html>
```

div 태그를 통해 총 3개의 레이아웃으로 나누어졌음이 확인됩니다. style 속성 중 background-color는 배경 색상을, color는 글자 색상을 의미하며, 각 레이아웃 마다 다른 스타일이 적용되었습니다.

### 2.3.8 CSS

CSS는 앞서 설명했듯이 웹페이지를 꾸며주는 역할을 합니다. head 부분에서 각 태그에 CSS 효과를 입력할 경우, 본문의 해당 태그들은 모두 CSS 효과가 적용됩니다. 이처럼 페이지를 꾸미기 위해 특정 요소에 접근하는 것을 선택터(Selector)라 합니다.

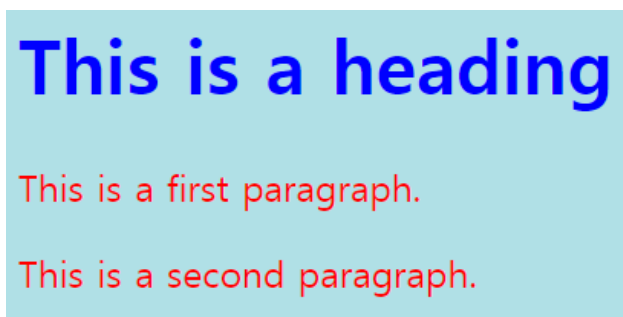


Figure 2.13: css 예제

```
<html>
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a first paragraph.</p>
<p>This is a second paragraph.</p>

</body>
</html>
```

head 태그 사이에 여러 태그에 대한 CSS 효과가 정의되었습니다. 먼저 body의 전체 배경색상을 파우더 블루로 주었으며, h1 태그의 글씨는 파란색으로, p 태그의 글씨는 붉은색으로 주었습니다. body 태그 내에서 style을 태그를 주지 않더라도, CSS 효과가 모두 적용되었음이 확인됩니다.

### 2.3.9 class와 id

CSS를 이용할 경우 본문의 모든 태그에 효과가 적용되므로, 특정한 요소<sup>Element</sup>에만 동일한 효과를 적용할 수 없습니다. 클래스 속성을 이용할 경우 동일한 이름을 가진 클래스에는 동일한 효과가 적용됩니다.

```
<html>
<style>
.index {
  background-color: tomato;
  color: white;
  padding: 10px;
}
.desc {
  background-color: moccasin;
  color: black;
  padding: 10px;
}
</style>
```

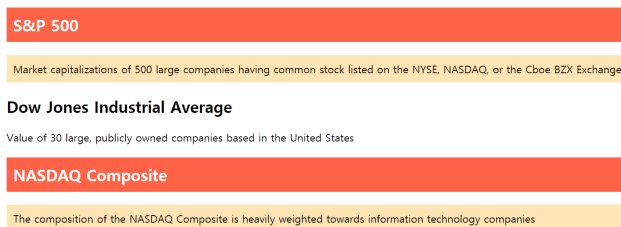


Figure 2.14: class 예제

```
<div>
<h2 class="index">S&P 500</h2>
<p class="desc"> Market capitalizations of 500 large companies having common stock listed on the NYSE, I
</div>

<div>
<h2>Dow Jones Industrial Average</h2>
<p>Value of 30 large, publicly owned companies based in the United States</p>
</div>

<div>
<h2 class="index">NASDAQ Composite</h2>
<p class="desc">The composition of the NASDAQ Composite is heavily weighted towards information technol
</div>
</html>
```

셀렉터를 클래스에 적용할때는 클래스명 앞에 콤마(,)를 붙혀 표현합니다. 위의 예제에서 index 클래스는 배경 색상이 토마토, 글씨는 흰색, 여백은 10px로 정의되었습니다. desc 클래스는 배경 색상이 모카신, 글씨는 검은색, 여백은 10px로 정의되었습니다. 본문의 첫번째와 세번째 레이아웃에만 h2 태그 뒤에는 'index' 클래스를, p 태그 뒤에는 'desc' 클래스를 속성으로 입력하였습니다. 따라서 해당 레이아웃에만 CSS 효과가 적용되며, 클래스 값이 없는 두번째 레이아웃에는 효과가 적용되지 않습니다.


id 또한 이와 비슷한 역할을 하며, HTML 내에서 여러 개의 class가 정의될 수 있는 반면, id는 단 하나만 사용하기를 권장합니다.

```
<html>
<head>
<style>

/* Style the element with the id "myHeader" */
#myHeader {
  background-color: lightblue;
  color: black;
  padding: 15px;
  text-align: center;
}
</style>
</head>
<body>

<!-- A unique element -->
<h1 id="myHeader">My Header</h1>
```





My Header

Figure 2.15: id 예제

Table 2.3: 파이프 오퍼레이터의 표현과 내용 비교

내용	주소
F(x)	x %>% F
G(F(x))	x %>% F %>% G

```
</body>
</html>
```

셀렉터를 id에 적용할때는 클래스명 앞에 샵(#)를 붙혀 표현하며, 페이지에서 한 번만 사용된다는 점을 제외하면 클래스와 사용방법이 거의 동일합니다. 클래스나 id 값을 통해 원하는 내용을 크롤링 하는 경우도 많으므로, 각각의 이름 앞에 콤마(.)와 샵(#)을 붙여야 한다는 점을 꼭 기억하시기 바랍니다.

크롤링에 필요한 HTML 관련 지식은 위에서 언급한 정도로도 충분하다고 생각합니다. 추가적인 정보가 필요하거나 내용이 궁금하신 분들은 아래 사이트를 참고하기 바랍니다.

- w3schools: <https://www.w3schools.in/html-tutorial/>
- 웨버 스터디: <http://webberstudy.com/>

## 2.4 파이프 오퍼레이터 (%>%)

R 내에서 동일한 데이터를 대상으로 연속적으로 작업하게 해주는 오퍼레이터(연산자)가 바로 파이프 오퍼레이터입니다. 크롤링에 꼭 필요한 **rvest** 패키지를 설치할 경우 자동으로 **magrittr** 패키지가 설치되어 사용할 수 있습니다.

흔히 프로그래밍에서 x라는 데이터를 F()라는 함수에 넣어 결과값을 확인하고 싶을 경우, F(x)의 방법을 사용합니다. 예를 들어 3과 5라는 데이터 중 큰 값을 찾고 싶을 때는 **max(3,5)**를 통해 확인합니다. 이를 통해 나온 결과 값을 또 다시 G()라는 함수에 넣어 결과값을 확인하고자 할 경우, 비슷한 과정을 거칩니다. **max(3,5)**를 통해 나온 값의 제곱근을 구하고자 할 경우 **result = max(3,5)**를 통해 첫 번째 결과값을 저장하고, **sqrt(result)**를 통해 두 번째 결과값을 계산합니다. 물론 **sqrt(max(3,5))**와 같은 표현법으로 한번에 표현할 수 있습니다.

이러한 표현의 단점은, 계산하는 함수가 많아질수록 저장하는 변수가 늘어나거나 혹은 괄호가 지나치게 길어집니다. 그러나 파이프 오퍼레이터인 %>%를 사용할 경우, 함수 간의 관계를 매우 직관적으로 표현하고 이해할 수 있습니다. 이를 정리하면 아래 표와 같습니다.

다음으로 파이프 오퍼레이터의 간단한 예제 를 통해 사용법을 살펴해보도록 하겠습니다. 먼저 다음과 같은 10개의 숫자가 있다고 가정합니다.

```
x = c(0.3078, 0.2577, 0.5523, 0.0564, 0.4685,
      0.4838, 0.8124, 0.3703, 0.5466, 0.1703)
```

우리가 원하는 과정은

1. 각 값들의 로그값을 구할 것
2. 로그값들의 계차를 구할 것
3. 구해진 계차의 지수값을 구할 것
4. 소수 둘째 자리까지 반올림할 것

입니다. 즉 `log()`, `diff()`, `exp()`, `round()`에 대한 값을 순차적으로 구하고자 합니다.

```
x1 = log(x)
x2 = diff(x1)
x3 = exp(x2)
round(x3, 2)
```

```
## [1] 0.84 2.14 0.10 8.31 1.03 1.68 0.46 1.48 0.31
```

첫 번째 방법은, 단계 별 함수의 결과값을 변수에 저장하고, 저장된 변수를 다시 불러와 함수에 넣고 계산하는 방법입니다. 전반적인 계산 과정을 확인하기에는 좋지만, 매번 변수에 저장하고 불러오고 하는 과정은 매우 비효율적이며, 코드 또한 불필요하게 길어집니다.

```
round(exp(diff(log(x))), 2)
```

```
## [1] 0.84 2.14 0.10 8.31 1.03 1.68 0.46 1.48 0.31
```

두 번째는 괄호를 통해 감싸는 방법입니다. 앞선 방법에 비해 코드는 짧아졌지만, 계산 과정을 알아보기에는 매우 불편한 방법으로 코드가 짜여 있습니다.

```
library(magrittr)
```

```
## Warning: 패키지 'magrittr'는 R 버전 3.5.3에서 작성되었습니다
```

```
x %>% log() %>% diff() %>% exp() %>% round(., 2)
```

```
## [1] 0.84 2.14 0.10 8.31 1.03 1.68 0.46 1.48 0.31
```

마지막으로 파이프 오퍼레이터를 사용하는 방법입니다. 코드도 짧으며, 계산 과정을 한눈에 파악하기도 좋습니다. 맨 왼쪽에는 원하는 변수를 입력하며, `%>%` 뒤에는 차례대로 계산하고자 하는 함수를 입력합니다. 변수의 입력값을 ()로 비워둘 경우, 오퍼레이터의 좌측에 있는 값이 입력변수가 됩니다. 반면 `round()`와 같이 입력값이 2개 이상 필요할 경우, 콤마(.)가 오퍼레이터의 좌측 값으로 입력됩니다.

파이프 오퍼레이터는 크롤링 뿐만이 아닌, 모든 코드에 사용할 수 있습니다. 이를 통해 훨씬 깔끔하면서도 데이터 처리과정을 직관적으로 이해할 수 있습니다.

## 2.5 오류에 대한 예외처리

크롤링을 이용하여 데이터를 수집할 경우, 일반적으로 `for loop` 구문을 통해 수천 종목에 해당하는 웹페이지에 접속하여 해당 데이터를 읽어옵니다. 그러나 특정 종목에 해당하는 페이지가 존재하지 않거나, 혹은 단기적으로 접속이 불안정할 경우 오류가 발생하여 루프를 처음부터 다시 실행해야 하는 번거로움이 있습니다. 이러한 경우 R에서는 `tryCatch()` 함수를 이용하여 예외처리, 즉 오류가 발생할 경우 이를 무시하고 넘어갈 수 있습니다.

`tryCatch()` 함수의 구조는 다음과 같습니다.

```
result = tryCatch({
  expr
}, warning = function(w) {
  warning-handler-code
}, error = function(e) {
```

```

    error-handler-code
}, finally = {
    cleanup-code
})

```

먼저 `expr`는 실행하고자 하는 코드를 의미합니다. `warning`은 경고를 나타내며, `warning-handler-code`는 경고 발생시 실행할 구문을 의미합니다. 이와 비슷하게, `error`와 `error-handler-code`는 각각 오류와 오류 발생시 실행할 구문을 의미합니다. `finally`는 오류의 여부와 관계없이 무조건 수행할 구문을 의미하며, 이는 생략이 가능합니다.

```

number = data.frame(1,2,3,"4",5, stringsAsFactors = FALSE)
str(number)

```

```

## 'data.frame':    1 obs. of  5 variables:
## $ X1 : num 1
## $ X2 : num 2
## $ X3 : num 3
## $ X.4.: chr "4"
## $ X5 : num 5

```

먼저 `number` 변수에는 1에서 5까지 값이 입력되어 있으며, 다른 값들은 형태가 숫자인 반면, 4는 문자 형태입니다.

```

for (i in number) {
  print(i^2)
}

```

```

## [1] 1
## [1] 4
## [1] 9

```

```

## Error in i^2: 이항연산자에 수치가 아닌 인수입니다

```

`for loop` 구문을 통해 순서대로 값들의 제곱을 출력하는 명령어를 실행할 경우, 문자 4는 제곱을 할 수 없어 오류가 발생하게 됩니다. `tryCatch()` 함수를 사용할 경우, 이처럼 오류가 발생하는 루프를 무시하고 다음 루프로 넘어갈 수 있게 됩니다.

```

for (i in number) {
  tryCatch({
    print(i^2)
  }, error = function(e) {
    print(paste('Error:', i))
  })
}

```

```

## [1] 1
## [1] 4
## [1] 9
## [1] "Error: 4"
## [1] 25

```

`expr`부분은 `print(i^2)`이며, `error-handler-code` 부분은 `print(paste('Error:', i))`, 즉 오류가 발생한 `i`를 출력하는 것입니다. 해당 코드를 실행할 경우 문자 4에서 오류가 발생함을 알려준 후, 루프가 멈추지 않고 다음으로 진행됩니다.



## Chapter 3

# API를 이용한 데이터 수집

본 장과 다음 장에서는 본격적으로 데이터를 수집하는 방법에 대해 배우도록 하겠습니다. 그 중 해당 장에서는 API를 이용하여 데이터를 수집하는 방법에 대해 살펴보도록 합니다.

API 제공자는 본인이 가진 데이터베이스를 다른 누군가가 쉽게 사용할 수 있는 형태로 가지고 있으며, 해당 데이터베이스에 접근할 수 있는 열쇠인 API 주소를 가진 사람은 이를 언제든지 사용할 수 있습니다.

API 주소만 가지고 있다면 데이터를 언제, 어디서, 누구나 쉽게 이용할 수 있다는 장점이 있습니다. 또한 대부분의 경우 사용자가 필요한 데이터만을 가지고 있으므로 접속 속도가 빠르며, 데이터를 가공하는 번거로움도 줄어듭니다. 해외의 경우 금융 데이터를 API의 형태로 제공하는 업체가 많으므로, 이를 잘만 활용한다면 매우 손쉽게 퀀트 투자에 필요한 데이터를 수집할 수 있습니다.

### 3.1 API를 이용한 Quandl 데이터 다운로드

데이터 제공업체 Quandl은 일부 데이터를 무료로 제공하며, API를 통해서 이를 다운로드 받을 수 있습니다.<sup>1</sup> 해당 책에서는 예제로써 애플(AAPL)의 주가를 다운로드 받도록 하겠습니다. csv 형식의 API 주소는 다음과 같습니다.

```
https://www.quandl.com/api/v3/datasets/WIKI/AAPL/data.csv?api_key=xw3NU3xLUZ7vZgrz5QnG
```

위 주소를 웹 브라우저 주소 창에 직접 입력하면 csv 형식의 파일이 다운로드 되며, 이를 열어보면 애플의 주가에 해당하는 정보들이 다운로드 됨이 확인됩니다.

그러나 웹 브라우저에 해당 주소를 입력하여 csv 파일을 다운로드 받고, 이를 다시 R에서 불러오는 작업은 무척이나 비효율적입니다. R 내에서 API 주소를 이용하여 직접 데이터를 다운로드 받아올 수 있습니다.

```
url.aapl = "https://www.quandl.com/api/v3/datasets/WIKI/AAPL/
data.csv?api_key=xw3NU3xLUZ7vZgrz5QnG"
data.aapl = read.csv(url.aapl)

head(data.aapl)
```

##	Date	Open	High	Low	Close	Volume	Ex.Dividend	Split.Ratio
## 1	2018-03-27	173.68	175.15	166.92	168.340	38962839	0	1
## 2	2018-03-26	168.07	173.10	166.44	172.770	36272617	0	1
## 3	2018-03-23	168.39	169.92	164.94	164.940	40248954	0	1
## 4	2018-03-22	170.00	172.68	168.60	168.845	41051076	0	1

<sup>1</sup>자세한 내용은 <https://docs.quandl.com/> 에서 확인할 수 있습니다.

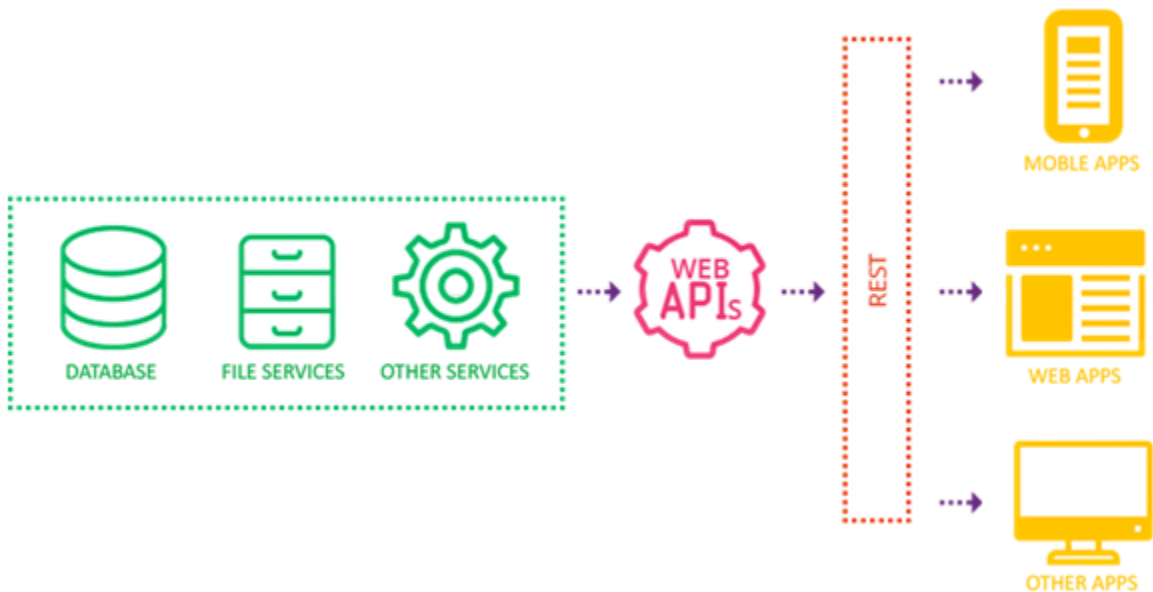


Figure 3.1: API 개념

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Date	Open	High	Low	Close	Volume	Ex-Divider	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
2	2018-03-27	173.68	175.15	166.92	168.34	38962839	0	1	173.68	175.15	166.92	168.34	38962839
3	2018-03-26	168.07	173.1	166.44	172.77	36272617	0	1	168.07	173.1	166.44	172.77	36272617
4	2018-03-23	168.39	169.92	164.94	164.94	40248954	0	1	168.39	169.92	164.94	164.94	40248954
5	2018-03-22	170	172.68	168.6	168.845	41051076	0	1	170	172.68	168.6	168.845	41051076
6	2018-03-21	175.04	175.09	171.26	171.27	35247358	0	1	175.04	175.09	171.26	171.27	35247358
7	2018-03-20	175.24	176.8	174.94	175.24	19314039	0	1	175.24	176.8	174.94	175.24	19314039
8	2018-03-19	177.32	177.47	173.66	175.3	32804695	0	1	177.32	177.47	173.66	175.3	32804695
9	2018-03-16	178.65	179.12	177.62	178.02	36836456	0	1	178.65	179.12	177.62	178.02	36836456
10	2018-03-15	178.5	180.24	178.0701	178.65	22584565	0	1	178.5	180.24	178.0701	178.65	22584565

Figure 3.2: API 주소를 이용한 데이터 다운로드

```
## 5 2018-03-21 175.04 175.09 171.26 171.270 35247358      0      1
## 6 2018-03-20 175.24 176.80 174.94 175.240 19314039      0      1
##   Adj..Open Adj..High Adj..Low Adj..Close Adj..Volume
## 1    173.68    175.15    166.92    168.340    38962839
## 2    168.07    173.10    166.44    172.770    36272617
## 3    168.39    169.92    164.94    164.940    40248954
## 4    170.00    172.68    168.60    168.845    41051076
## 5    175.04    175.09    171.26    171.270    35247358
## 6    175.24    176.80    174.94    175.240    19314039
```

url1에 해당 주소를 입력한 후, `read.csv()` 함수를 이용하여 간단하게 csv 파일을 불러올 수 있습니다.

API 주소를 조금만 수정하면, 다른 종목의 데이터도 다운로드 받을 수 있습니다. 위의 주소에서 애플의 티커에 해당하는 AAPL를 넷플렉스의 티커에 해당하는 NFLX로 변경하여 데이터를 다운로드 받아보도록 하겠습니다.

```
url.nflx = 'https://www.quandl.com/api/v3/datasets/WIKI/NFLX/
data.csv?api_key=xw3NU3xLUZ7vZgrz5QnG'
data.nflx = read.csv(url.nflx)

head(data.nflx)
```

```
##      Date   Open   High    Low  Close  Volume Ex.Dividend Split.Ratio
## 1 2018-03-27 322.49 322.90 297.000 300.69 11890994      0      1
## 2 2018-03-26 309.36 321.03 302.000 320.35 11906279      0      1
## 3 2018-03-23 307.41 310.73 300.360 300.94  9226978      0      1
## 4 2018-03-22 313.07 314.12 305.660 306.70  7920524      0      1
## 5 2018-03-21 316.35 319.40 314.511 316.48  5016980      0      1
## 6 2018-03-20 313.26 319.50 312.800 317.50  5922296      0      1
##   Adj..Open Adj..High Adj..Low Adj..Close Adj..Volume
## 1    322.49    322.90    297.000    300.69    11890994
## 2    309.36    321.03    302.000    320.35    11906279
## 3    307.41    310.73    300.360    300.94    9226978
## 4    313.07    314.12    305.660    306.70    7920524
## 5    316.35    319.40    314.511    316.48    5016980
## 6    313.26    319.50    312.800    317.50    5922296
```

넷플렉스의 주가 역시 손쉽게 다운로드 받을 수 있음이 확인됩니다.

## 3.2 getSymbols() 함수를 이용한 API 다운로드

앞선 예에서 API 주소를 이용할 경우 매우 간단하게 데이터를 수집할 있음을 살펴 보았습니다. 그러나 해당 방법에는 여러 단점 또한 존재합니다. 먼저, 원하는 항목에 대한 API를 일일이 얻는 것이 힘든 일입니다. 또한 Quandl의 경우 무료로 얻을 수 있는 정보에 제한이 있으며, 다운로드 양에 대한 제한도 있습니다. 한 두 종목의 경우 해당 방법으로 데이터를 수집할 수 있지만, 전 종목의 데이터를 해당 방법으로 구하는 것은 사실상 불가능 합니다.

다행히 야후 파이낸스에서 주가 데이터를 무료로 제공하며, `quantmod` 패키지의 `getSymbols()` 함수는 해당 API에 접속하여 데이터를 다운로드 받아옵니다.

### 3.2.1 주가 다운로드

`getSymbols()` 함수의 기본적인 사용법은 매우 간단합니다. 괄호 안에 다운로드 받고자 하는 종목의 티커를 입력하면 됩니다. 아래의 코드는 애플의 티커인 “AAPL”을 입력한 경우입니다. 티커와 동일한 변수인 AAPL이 생성되며, 주가 데이터가 다운로드 된 후 `xts` 형태로 입력됩니다.

```
library(quantmod)
getSymbols('AAPL')
```

```
## [1] "AAPL"
```

```
head(AAPL)
```

```
##          AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume
## 2007-01-03  12.32714  12.36857  11.70000   11.97143   309579900
## 2007-01-04  12.00714  12.27857  11.97429   12.23714   211815100
## 2007-01-05  12.25286  12.31428  12.05714   12.15000   208685400
## 2007-01-08  12.28000  12.36143  12.18286   12.21000   199276700
## 2007-01-09  12.35000  13.28286  12.16429   13.22429   837324600
## 2007-01-10  13.53571  13.97143  13.35000   13.85714   738220000
##          AAPL.Adjusted
## 2007-01-03      7.951960
## 2007-01-04      8.128461
## 2007-01-05      8.070577
## 2007-01-08      8.110430
## 2007-01-09      8.784165
## 2007-01-10      9.204537
```

다운로드 결과로써 총 6개의 열이 생성됩니다. Open은 시가, High는 고가, Low는 저가, Close는 종가를 의미합니다. 또한, Volume은 거래량을 의미하며, Adjusted는 배당이 반영된 수정주가를 의미합니다. 이 중 가장 많이 사용되는 데이터는 Adjusted, 즉 배당이 반영된 수정주가입니다. `Ad()` 함수는 `getSymbols()` 함수를 통해 다운로드 받은 데이터에서 수정주가만을 선택하여 줍니다. 시계열 그래프를 그려주는 `chart_Series()`와 함께 수정주가를 그리면 다음과 같습니다.

```
chart_Series(Ad(AAPL))
```





시계열 기간을 입력하지 않을 경우 2007년 1월부터 현재까지의 데이터가 다운로드 되며, 추가적인 입력 변수를 통해 원하는 기간의 데이터를 다운로드 받을 수도 있습니다. from에는 시작 시점을, to에는 종료 시점을 입력하여 주면, 해당 시점의 데이터가 다운로드 받아짐이 확인됩니다.

getSymbols() 함수를 통해 다운로드 받은 데이터는 자동으로 티커와 동일한 변수명에 저장됩니다. 만일 티커명이 아닌 원하는 변수명에 데이터를 저장하고 싶을 경우, auto.assign 인자를 FALSE로 설정해주면 다운로드 받은 데이터가 원하는 변수에 저장됩니다.

```
data = getSymbols('AAPL', from = '2000-01-01', to = '2018-12-31', auto.assign = FALSE)
head(data)
```

```
##          AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume
## 2000-01-03  3.745536  4.017857  3.631696   3.997768  133949200
## 2000-01-04  3.866071  3.950893  3.613839   3.660714  128094400
## 2000-01-05  3.705357  3.948661  3.678571   3.714286  194580400
## 2000-01-06  3.790179  3.821429  3.392857   3.392857  191993200
## 2000-01-07  3.446429  3.607143  3.410714   3.553571  115183600
## 2000-01-10  3.642857  3.651786  3.383929   3.491071  126266000
##          AAPL.Adjusted
## 2000-01-03      3.502161
## 2000-01-04      3.206892
## 2000-01-05      3.253822
## 2000-01-06      2.972241
## 2000-01-07      3.113032
## 2000-01-10      3.058280
```

한번에 여러 종목의 주가를 다운로드 받을 수도 있습니다. 아래 예제와 같이 페이스북과 엔비디아의 티커인 FB와

NVDA를 ticker 변수에 입력하여 주고, `getSymbols()` 함수에 티커들을 입력한 변수를 넣어주면 두 종목의 주가가 동시에 다운로드 됩니다.

```
ticker = c('FB', 'NVDA')
getSymbols(ticker)
```

```
## [1] "FB"    "NVDA"
```

```
head(FB)
```

```
##           FB.Open FB.High FB.Low FB.Close FB.Volume FB.Adjusted
## 2012-05-18   42.05   45.00  38.00   38.23 573576400      38.23
## 2012-05-21   36.53   36.66  33.00   34.03 168192700      34.03
## 2012-05-22   32.61   33.59  30.94   31.00 101786600      31.00
## 2012-05-23   31.37   32.50  31.36   32.00  73600000      32.00
## 2012-05-24   32.95   33.21  31.77   33.03  50237200      33.03
## 2012-05-25   32.90   32.95  31.11   31.91  37149800      31.91
```

```
head(NVDA)
```

```
##           NVDA.Open NVDA.High NVDA.Low NVDA.Close NVDA.Volume
## 2007-01-03   24.71333   25.01333  23.19333   24.05333   28870500
## 2007-01-04   23.96667   24.05333  23.35333   23.94000   19932400
## 2007-01-05   23.37333   23.46667  22.28000   22.44000   31083600
## 2007-01-08   22.52000   23.04000  22.13333   22.60667   16431700
## 2007-01-09   22.64000   22.79333  22.14000   22.16667   19104100
## 2007-01-10   21.93333   23.46667  21.60000   23.26000   27718600
##           NVDA.Adjusted
## 2007-01-03      22.19124
## 2007-01-04      22.08669
## 2007-01-05      20.70281
## 2007-01-08      20.85657
## 2007-01-09      20.45063
## 2007-01-10      21.45933
```

### 3.2.2 국내 종목 주가 다운로드

`getSymbols()` 함수를 이용하면 미국뿐 아니라 국내 종목의 주가를 다운로드 받을 수도 있습니다. 국내 종목의 티커는 총 6자리로 구성되어 있으며, 해당 함수에 입력되는 티커는 코스피 상장 종목의 경우 “티커.KS”, 코스닥 상장 종목의 경우 “티커.KQ”의 형태로 입력해 주어야 합니다.

먼저 코스피 상장종목인 삼성전자 데이터의 다운로드 예시입니다.

```
getSymbols('005930.KS', from = '2000-01-01', to = '2018-12-31')
```

```
## [1] "005930.KS"
```

```
tail(Ad(`005930.KS`))
```

```
##          005930.KS.Adjusted
## 2018-12-20          38293.23
## 2018-12-21          38293.23
## 2018-12-24          38441.84
## 2018-12-26          37996.00
## 2018-12-27          38250.00
## 2018-12-28          38700.00
```

삼성전자의 티커인 005930에 .KS를 붙여 함수에 입력할 경우, 티커명에 해당하는 005930.KS 변수명에 데이터가 저장됩니다. 변수명에 콤마(.)가 있는 관계로, Ad 함수를 통해 수정주가를 확인하고자 할 때는 변수명의 앞뒤에 역음부호(`)를 붙여주어야 합니다.

국내 종목의 경우 가끔 수정주가에 오류가 발생하는 경우가 많습니다. 해당이 반영된 값 보다는 단순 종가(Close) 데이터를 사용하기를 권장합니다.

```
tail(Cl(`005930.KS`))
```

```
##          005930.KS.Close
## 2018-12-20          38650
## 2018-12-21          38650
## 2018-12-24          38800
## 2018-12-26          38350
## 2018-12-27          38250
## 2018-12-28          38700
```

Cl() 함수는 Close, 즉 종가만을 선택하여 주며, 사용 방법은 기존 Ad() 함수와 동일합니다. 비록 배당을 고려할 수는 없지만, 전반적으로 오류가 없는 데이터를 사용할 수 있습니다.

다음은 코스닥 상장종목인 셀트리온제약의 예시이며, 티커인 068760에 .KQ를 붙여 함수에 입력합니다. 역시나 데이터가 다운로드 되어 티커명의 변수에 저장됩니다.

```
getSymbols("068760.KQ", from = '2000-01-01', to = '2018-12-31')
```

```
## [1] "068760.KQ"
```

```
tail(Cl(`068760.KQ`))
```

```
##          068760.KQ.Close
## 2018-01-24          95100
## 2018-01-25          97300
## 2018-01-26          97400
## 2018-01-29          99900
## 2018-01-30          99500
## 2018-01-31          97500
```

### 3.2.3 FRED 데이터 다운로드

미국 및 각국의 중요 경제지표 데이터를 살펴볼 때, 가장 많이 참조되는 곳 중 하나가 미 연방 준비 은행에서 관리하는 Fred Economic Data 입니다. getSymbols() 함수를 통해 FRED 데이터를 다운로드 받을 수 있습니다. 먼저 미국채 10년물 금리를 다운로드 받는 예제를 살펴보도록 하겠습니다.

```
getSymbols('DGS10', src='FRED')
```

```
## [1] "DGS10"
```

```
chart_Series(DGS10)
```



먼저 미 국채 10년물 금리에 해당하는 티커인 “DGS10”을 입력해 줍니다. 그 후, 데이터 소스에 해당하는 src에 “FRED”를 입력해 주면, 해당 데이터가 다운로드 됩니다. `chart_Series`를 통해 해당 데이터를 그래프로 나타내면 다음과 같습니다.

각 항목 별 티커를 찾는 방법은 매우 간단합니다. 먼저 FRED의 홈페이지<sup>2</sup>에 접속하여, 원하는 데이터를 검색합니다. 예시로써 원/달러 환율에 해당하는 South Korea / U.S. Foreign Exchange Rate 를 검색하여 원하는 페이지에 접속합니다. 이 중 페이지 주소에서 /series/ 다음에 위치하는 DEXKOUS 가 해당 항목의 티커입니다.

해당 티커를 입력하면, 홈페이지와 동일한 데이터가 다운로드 됨이 확인됩니다. 이 외에도 509,000 여개의 방대한 FRED 데이터를 해당 함수를 통해 손쉽게 R에서 다운로드 받을 수 있습니다.

```
getSymbols('DEXKOUS', src='FRED')
```

```
## [1] "DEXKOUS"
```

```
tail(DEXKOUS)
```

<sup>2</sup><https://fred.stlouisfed.org/>



Figure 3.3: FRED 사이트 내 원/달러 환율의 티커 확인

```
##          DEXKOUS
## 2019-05-31 1190.50
## 2019-06-03 1180.97
## 2019-06-04 1180.58
## 2019-06-05 1178.24
## 2019-06-06 1179.51
## 2019-06-07 1181.27
```

```
chart_Series(DEXKOUS)
```



## Chapter 4

# 크롤링 이해하기

앞선 장에서 볼 수 있듯이 API를 이용할 경우 데이터를 매우 쉽게 수집할 수 있지만, 국내 주식 데이터를 다운로드 받기에는 한계가 있으며, 우리가 원하는 데이터가 API의 형태로 제공된다는 보장도 없습니다. 따라서 우리는 필요한 데이터를 얻기 위해 직접 찾아나서야 합니다.

다행히도 금융 사이트들에는 주가, 재무정보 등 우리가 원하는 대부분의 주식 정보가 제공되고 있으며, API를 활용할 수 없는 경우에도 크롤링을 통해 이러한 데이터를 수집할 수 있습니다.

크롤링 혹은 스크래핑이란 웹사이트에서 원하는 정보를 수집하는 기술입니다. 대부분의 금융 사이트들이 간단한 형태로 작성되어 있어, 몇 가지 기술만 익히면 어렵지 않게 데이터를 크롤링 할 수 있습니다. 그러나 크롤링에 대한 대부분의 강의나 설명이 파이썬을 이용한 방법으로써, 초보자가 R을 이용한 크롤링을 배우는 데는 어려움이 있습니다.

해당 장에서는 크롤링에 대한 간단한 설명과 예제를 살펴보도록 하겠습니다.

크롤링을 할 때는 주의해야 할 점이 있습니다. 특정 사이트의 페이지를 쉬지 않고 크롤링을 하는 행위를 무한 크롤링이라 합니다. 이러한 경우 해당 사이트의 자원을 독점하게 되어 타인의 사용을 막게 되며, 사이트에 부하를 주게 됩니다. 일부 사이트에서는 동일한 IP로 쉬지 않고 크롤링을 할 경우 접속을 막아버리는 경우도 있습니다. 따라서 하나의 페이지를 크롤링 한 후, 1~2초 가량 정지한 후 다시 다음 페이지를 크롤링 할 필요가 있습니다.

## 4.1 GET과 POST 방식 이해하기

우리가 인터넷에 접속하여 서버에 파일을 요청하면, 서버는 이에 해당하는 파일을 우리에게 보내줍니다. 이러한 과정을 사람이 수행하기 편하고 시각적으로 보기 편하도록 만들어 진 것이 크롬과 같은 웹브라우저이며, 서버의 주소를 기억하기 쉽게하기 위해 만든 것이 인터넷 주소입니다. 우리가 서버에 데이터를 요청하는 형태는 다양하지만 크롤링에서는 주로 GET과 POST 방식을 사용합니다.

### 4.1.1 GET 방식

GET 방식은 인터넷 주소를 기준으로, 이에 해당하는 데이터나 파일을 요청하는 것입니다. 주로 클라이언트가 요청하는 쿼리를 앰퍼샌드(&) 혹은 물음표(?) 형식으로 결합하여 서버에 전달됩니다.

한경컨센서스<sup>1</sup>에 접속한 후 전체 REPORT를 선택하면, 홈페이지의 주소 뒤에 `/apps.analysis/analysis.list`가 붙으며 이에 해당하는 페이지의 내용을 보여줍니다. 상단의 탭에서 기업을 선택하면, 주소의 끝부분에 `?skinType=business`가 추가되며 이에 해당하는 페이지의 내용을 보여줍니다. 즉, 해당 페이지는 GET 방식을 사용하고 있으며 입력종류는 skinType, 이에 해당하는 기업 탭의 입력값은 business 임을 알 수 있습니다.

이번에는 과생 탭을 선택하여 봅니다. 역시나 홈페이지 주소가 변경되며 해당 주소에 맞는 내용이 나타납니다. 주소의 끝부분이 `?skinType=derivative`로 변경되며, 입력 값이 변경됨에 따라 페이지의 내용이 이에 맞게 변하는 모습이

<sup>1</sup><http://hkconsensus.hankyung.com/>

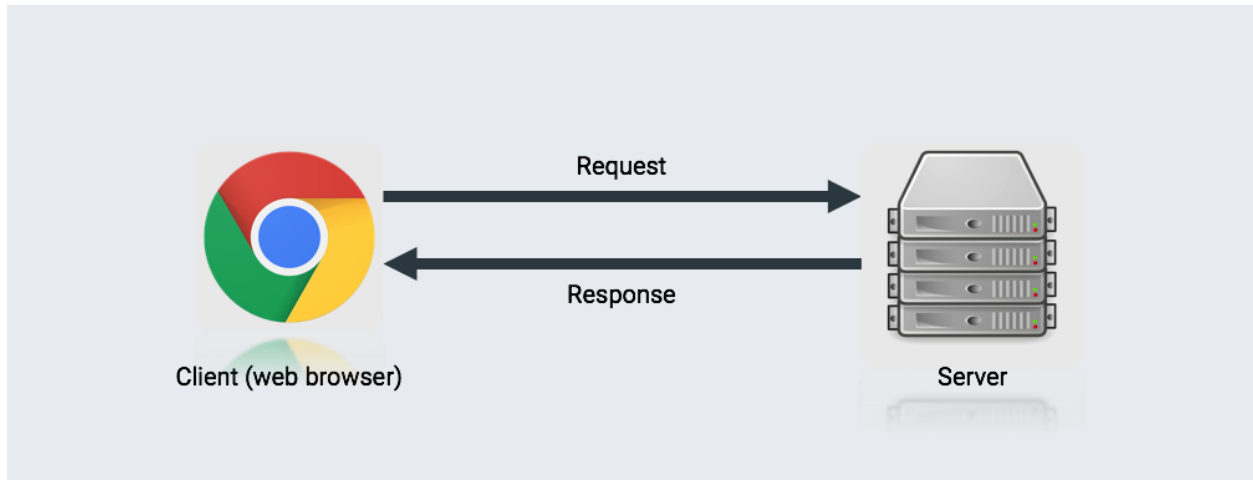


Figure 4.1: 클라이언트와 서버 간의 요청/응답 과정

① [hkconsensus.hankyung.com/apps.analysis/analysis.list?skinType=business](http://hkconsensus.hankyung.com/apps.analysis/analysis.list?skinType=business)

한경컨센서스

2018-06-07

 ~ 

2018-07-07

전체

기업

검색

Q 종목

전체

기업

산업

시장

파생

경제

상향

하향

기업정보

LIST

20

50

80










작성일 ▼	제목 ▼	적정가격 ▼	투자의견 ▼	작성자 ▼	제공출처 ▼	기업정보	차트	첨부파일
2018-07-06	삼성전자(005930)일회일비 하지 말자	63,000	Buy	김경민, 박강호	대신증권			
2018-07-06	후성(093370)안 오른 2차전지주를 찾...	0	Not Rated	손세훈	NH투자증권			
2018-07-06	파라다이스(034230)6월 및 2분기 실적...	24,500	Buy	성준원, 강수연	신한금융투자			
2018-07-06	동원산업(006040)2분기도 순항 중	400,000	Buy	구현지, 홍세종	신한금융투자			
2018-07-06	GS건설(006360)또다시 기대되는 호실...	60,000	Buy	오경석	신한금융투자			

Figure 4.2: 한경 컨센서스 기업 REPORT 페이지



① [hkconsensus.hankyung.com/apps.analysis/analysis.list?skinType=business&sdate=2018-06-07&edate=2018-07-07&order\\_type=&now\\_page=2](http://hkconsensus.hankyung.com/apps.analysis/analysis.list?skinType=business&sdate=2018-06-07&edate=2018-07-07&order_type=&now_page=2)

작성일	제목	적정가격	투자의견	작성자	제공출처	기업정보	차트	첨부파일
2018-07-06	BGF리테일(282330)점포당 매출 회복에...	240,000	Buy	허나래	한국투자증권			
2018-07-06	휴비츠(065510)안광학 의료기기 전문...	0	nr	김한경	이베스트증권			
2018-07-06	신세계(004170)면세점으로 한 단계 도...	460,000	Buy	허나래	한국투자증권			
2018-07-06	삼성중공업(010140)하반기 개선점 찾...	8,000	Hold	이상우	유진투자증권			
2018-07-06	GS리테일(007070)오피스와 편의점의 ...	57,000	Buy	허나래	한국투자증권			

Figure 4.3: 쿼리 추가로 인한 url의 변경

확인됩니다. 여러 다른 탭들을 눌러보면 **?skinType=** 뒷부분의 입력값이 변함에 따라 이에 해당하는 페이지로 내용이 변경됨이 확인됩니다.

다시 기업 탭을 선택한 후, 다음 페이지를 확인하기 위해 하단의 2를 클릭합니다. 기존 주소인 **?skinType=business** 뒤에 추가로 **sdate**와 **edate**, 그리고 **now\_page** 쿼리가 추가됨이 확인됩니다. **sdate**에 검색 기간의 시작시점, **edate**에 검색 기간의 종료시점, **now\_page**에 원하는 페이지를 수기로 입력해도 이에 해당하는 페이지의 데이터를 보여줍니다. 이처럼 GET 방식으로 데이터를 요청할 경우, 웹 페이지 주소를 수정하여 원하는 종류의 데이터를 받아올 수 있습니다.

#### 4.1.2 POST 방식

POST 방식은 사용자가 필요한 값을 추가해서 요청하는 방법입니다. GET 방식과의 차이는 클라이언트가 요청하는 쿼리를 body에 넣어서 전송하므로, 요청 내역을 직접적으로 볼 수 없습니다.

한국거래소 상장공시시스템<sup>2</sup>에 접속하여 전체메뉴보기를 누른 후, 상장법인상세정보 중 상장종목현황을 선택합니다. 웹 페이지 주소가 바뀌며, 상장종목현황이 보여집니다.

이번엔 조회일자를 2017-12-28로 선택한 후, 검색을 눌러보도록 합니다. 페이지의 내용은 선택일 기준으로 변경되었지만, 주소는 변경되지 않고 그대로 남아있습니다. GET 방식에서는 선택항목에 따라 웹 페이지 주소가 변경되었지만, POST 방식을 사용하여 서버에 데이터를 요청하는 해당 사이트는 그렇지 않음이 확인됩니다.

POST 방식의 데이터 요청과정을 살펴보기 위해서는 개발자도구 이용해야 하며, 크롬 브라우저에서 F12 키를 눌러 해당 화면을 열 수 있습니다. 개발자도구 화면을 연 상태에서 다시 한번 '검색'을 클릭해 봅니다. Network 탭을 클릭하면, '검색'을 클릭함과 함께 브라우저와 서버간의 통신 과정을 살펴볼 수 있습니다. 이 중 **listedIssueStatus.do**라는 항목이 POST 형태임을 알 수 있습니다.

해당 메뉴를 클릭하면 통신 과정을 좀 더 자세히 알 수 있습니다. 가장 하단의 Form Data에 서버에 데이터를 요청하는 내역이 있습니다. method에는 **readListIssueStatus**, selDate에는 2017-12-28라는 값이 있습니다.

이처럼 POST 방식은 요청하는 데이터에 대한 쿼리가 GET 방식처럼 url을 통해 전송되는 것이 아닌 body를 통해 전송되므로, 이에 대한 정보는 웹 브라우저를 통해 확인할 수는 없습니다.

<sup>2</sup><http://kind.krx.co.kr/>



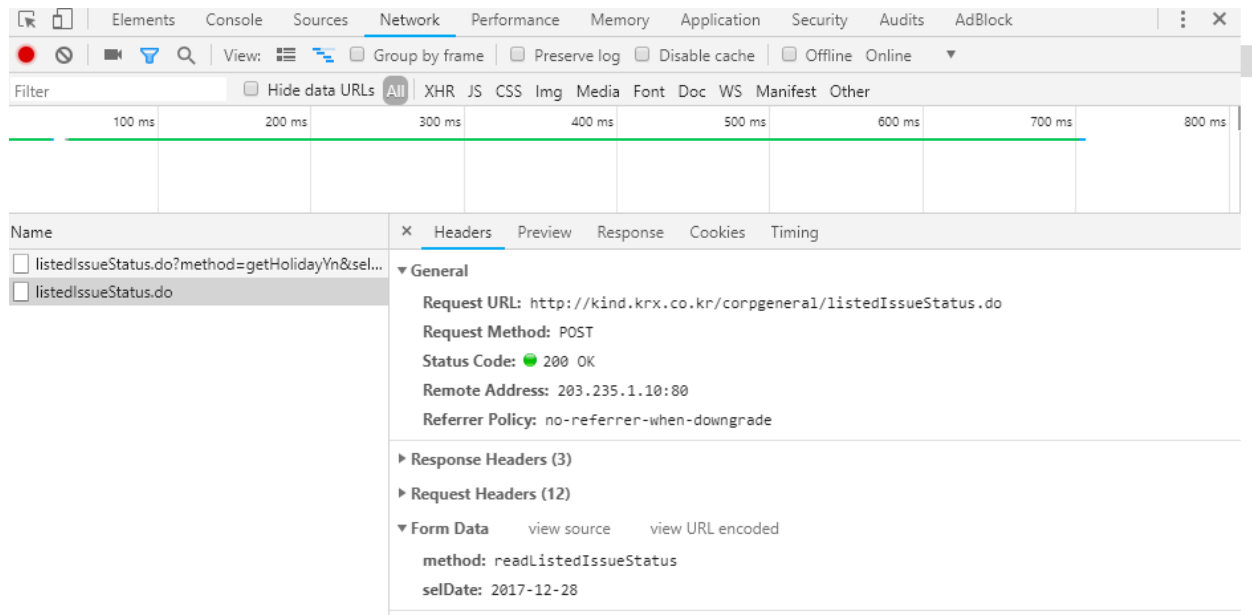


Figure 4.6: POST 방식의 서버 요청 내역



Figure 4.7: 실시간 속보의 제목 부분 html

## 4.2 크롤링 예제

크롤링의 일반적인 과정은 `httr` 패키지의 `GET()` 혹은 `POST()` 함수를 이용하여 데이터를 다운로드 받은 후, `rvest` 패키지의 함수들을 이용하여 원하는 데이터를 찾아내는 과정으로 이루어집니다. 해당 장에서는 GET 방식의 예제로 금융 실시간 속보의 제목을 추출하는 방법을, POST 방식의 예제로 기업공시채널에서 오늘의 공시를 추출하는 방법을, 마지막으로 태그와 속성, 페이지 네비게이션 값을 결합하여 국내 상장 주식의 종목명 및 티커를 추출하는 방법에 대해 알아보도록 하겠습니다.

### 4.2.1 금융 속보 크롤링

크롤링의 간단한 예제로 금융 속보의 제목을 추출해 보도록 하겠습니다. 먼저 네이버 금융에 접속한 후 뉴스 → 실시간 속보<sup>3</sup>를 선택해 줍니다. 이 중 뉴스의 제목에 해당하는 텍스트만 추출하고자 합니다.

뉴스 제목 부분에 마우스를 올려둔 후 우클릭 → 검사를 선택할 경우 개발자도구 화면이 열리며, 해당 글자가 html 내에서 어떤 부분에 위치하는지 확인할 수 있습니다. 해당 제목은 `dl` 태그 → `dd` 태그의 `articleSubject` 클래스 → `a` 태그 중 `title` 속성에 위치하고 있습니다. 태그와 속성의 차이가 이해되지 않으시는 분은 태그와 속성 부분을 다시 살펴보시기 바랍니다.

<sup>3</sup>[https://finance.naver.com/news/news\\_list.nhn?mode=LSS2D&section\\_id=101&section\\_id2=258](https://finance.naver.com/news/news_list.nhn?mode=LSS2D&section_id=101&section_id2=258)

먼저 해당 페이지의 내용을 R로 불러오도록 하겠습니다.

```
library(rvest)
library(httr)

url = 'https://finance.naver.com/news/news_list.nhn?mode=LSS2D&section_id=101&section_id2=258'
data = GET(url)

print(data)

## Response [https://finance.naver.com/news/news_list.nhn?mode=LSS2D&section_id=101&section_id2=258]
##   Date: 2019-06-13 04:45
##   Status: 200
##   Content-Type: text/html; charset=EUC-KR
##   Size: 59.9 kB
##
##
##
##
##
##
##
## <!-- global include -->
##
## ...
```

먼저 url 변수에 해당 주소를 입력한 후, GET() 함수를 이용하여 해당 페이지의 내용을 받아 data 변수에 저장합니다. data 변수를 확인해보면 Status가 200, 즉 데이터가 이상없이 받아졌으며, 인코딩은 EUC-KR 타입으로 되어 있습니다.

우리는 개발자도구 화면을 통해 제목에 해당하는 부분이 dl 태그 → dd 태그의 articleSubject 클래스 → a 태그 중 title 속성에 위치하고 있음을 살펴보았습니다. 이를 활용해 제목 부분만을 추출하는 방법은 다음과 같습니다.

```
data_title = data %>%
  read_html(encoding = 'EUC-KR') %>%
  html_nodes('dl') %>%
  html_nodes('.articleSubject') %>%
  html_nodes('a') %>%
  html_attr('title')
```

1. 먼저 read\_html() 함수를 이용하여 해당 페이지의 html 내용을 읽어오며, 인코딩은 'EUC-KR'로 셋팅해 주도록 합니다.
2. html\_nodes() 함수는 해당 태그를 추출하는 함수로써, dl 태그에 해당하는 부분을 추출합니다.
3. html\_nodes() 함수를 이용하여 articleSubject 클래스에 해당하는 부분을 추출할 수 있으며, 클래스 속성의 경우 이름 앞에 콤마(.)를 붙여주어야 합니다.
4. html\_nodes() 함수를 이용하여 a 태그를 추출합니다.
5. html\_attr은 속성을 추출하는 함수로써, title에 해당하는 부분만을 추출합니다.

해당 과정을 거쳐 data\_title에는 실시간 속보의 제목만이 저장되게 됩니다. 이처럼 개발자도구 화면을 통해 내가 추출하고자 하는 데이터가 html 중 어디에 위치하고 있는지 먼저 확인을 하면, 어렵지 않게 원하는 데이터를 읽어올 수 있습니다.

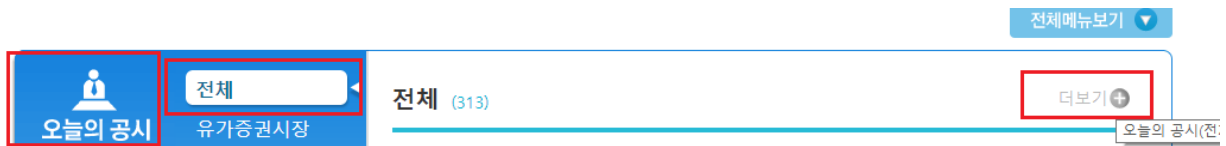


Figure 4.8: 오늘의공시 확인하기

```
print(data_title)
```

```
## [1] "'금호에이치티' 5% 이상 상승, 주가 상승 흐름, 단기 이평선 정배열, 중기 이평선 역배열"
## [2] "세종, 전일 대비 약 7% 상승한 3,960원"
## [3] "우진아이엔에스 (주)태영건설과 201억원 계약체결"
## [4] "유니트론텍, 52주 신고가 경신... 전일 대비 8% 상승"
## [5] "[ET투자뉴스] 덕산네오룩스_기관/개인 순매수, 외국인은 순매도(한달누적)"
## [6] "[fnRASSI] 유니트론텍, 52주 신고가...7.36% ↑"
## [7] "[ET투자뉴스] 카카오_상장주식수 대비 거래량은 0.98%로 적정수준"
## [8] "[fnRASSI] 도이치모터스, 52주 신고가...2.89% ↑"
## [9] "KB증권, 해외 파생결합증권 발행 시장 진출"
## [10] "대양금속, 전일 대비 약 4% 하락한 7,900원"
## [11] "'유니트론텍' 52주 신고가 경신, 단기·중기 이평선 정배열로 상승세"
## [12] "동부건설, 453억원 규모 주택공사 수주"
## [13] "[오후시황] 외국인 기관 매도에...코스피 2,100선 아래로"
## [14] "[코스닥 수급] 13시 30분 개인(235억), 기관(-346억)"
## [15] "'고려산업' 5% 이상 상승, 단기·중기 이평선 정배열로 상승세"
## [16] "오후 1:30 현재 코스피는 52:48으로 매도우위, 매수강세 업종은 전기가스업(1.12%↓)"
## [17] "오후 1:30 현재 코스닥은 47:53으로 매수우위, 매도강세 업종은 건설업(0.75%↑)"
## [18] "금호에이치티, 전일 대비 약 4% 상승한 5,740원"
## [19] "[코스피 수급] 13시 30분 외인(-1,832억), 개인(2,414억)"
## [20] "게임주, 신작 효과+정부 챙기기에 기대감 'UP'"
```

#### 4.2.2 기업공시채널에서 오늘의 공시 불러오기

한국거래소 상장공시시스템에 접속한 후 오늘의 공시 → 전체 → 더보기를 선택하여 전체 공시내용을 확인할 수 있습니다.

해당 페이지에서 날짜를 변경할 경우, 페이지의 내용은 해당일의 공시로 변경되지만 url은 변경되지 않습니다. 이처럼 POST 방식의 경우 요청하는 데이터에 대한 쿼리가 body의 형태를 통해 전송되므로, 개발자도구 화면을 통해 해당 쿼리에 대한 내용을 확인할 수 있습니다.

개발자도구 화면을 연 상태에서 조회일자를 2018-12-28로 선택한 후 Network 탭의 **todaydisclosure.do** 항목을 살펴보면 Form Data를 통해 서버에 데이터를 요청하는 내역을 확인할 수 있습니다. 여러 항목 중 selDate 부분이 우리가 선택한 일자로 설정되어 있습니다.

POST 방식으로 쿼리를 요청하는 방법을 코드로 나타내면 다음과 같습니다.

```
library(httr)
library(rvest)
```

```
Sys.setlocale("LC_ALL", "English")
```

```
## [1] "LC_COLLATE=English_United States.1252;LC_CTYPE=English_United States.1252;LC_MONETARY=English_U"
```

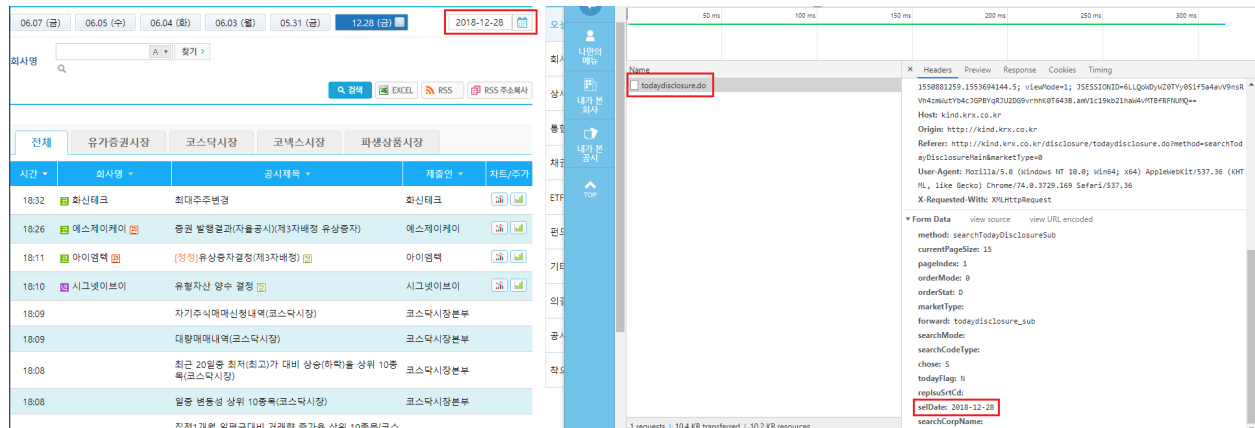


Figure 4.9: POST 방식의 데이터 요청

```
url = 'http://kind.krx.co.kr/disclosure/todaydisclosure.do'
data = POST(url, body =
  list(
    method = 'searchTodayDisclosureSub',
    currentPageSize = '15',
    pageIndex = '1',
    orderMode = '0',
    orderStat = 'D',
    forward = 'todaydisclosure_sub',
    chose = 'S',
    todayFlag = 'Y',
    selDate = '2018-12-28'
  ))

data = read_html(data) %>%
  html_table(fill = TRUE) %>%
  .[[1]]

Sys.setlocale("LC_ALL", "Korean")
```

```
## [1] "LC_COLLATE=Korean_Korea.949;LC_CTYPE=Korean_Korea.949;LC_MONETARY=Korean_Korea.949;LC_NUMERIC=C"
```

1. 한글로 작성된 페이지를 크롤링 할 경우 오류가 발생하는 경우가 종종 있으므로, `Sys.setlocale()` 함수를 통해 로케일 언어를 영어로 설정 해줍니다.
2. `POST()` 함수를 통해 해당 url에 원하는 쿼리를 요청해주며, 쿼리는 body 내에 list 형태로 입력해주시도록 합니다. 해당 값은 개발자도구 화면의 Form Data와 동일하게 입력해주며, marketType과 같이 값이 존재하지 않는 항목은 입력하지 않아도 됩니다.
3. `read_html()` 함수를 이용하여 해당 페이지의 html 내용을 읽어옵니다.
4. `html_table()` 함수는 테이블 형태의 데이터를 읽어오는 함수입니다. 셀 간 병합이 된 열이 존재하므로 `fill=TRUE` 를 추가해주시도록 합니다.
5. `.[[1]]`를 통해 첫번째 리스트를 선택해 줍니다.
6. 한글을 읽기 위해 `Sys.setlocale()` 함수를 통해 로케일 언어를 다시 Korean으로 변경해 줍니다.

저장된 데이터를 확인하면 화면과 동일한 내용이 출력됩니다.

```
print(head(data))
```

```
##      NA      NA      NA
## 1 18:32   화신테크   최대주주변경
## 2 18:26 에스제이케이 증권 발행결과(자율공시)(제3자배정 유상증자)
## 3 18:11   아이엠텍   [정정]유상증자결정(제3자배정)
## 4 18:10 시그넷이브이   유형자산 양수 결정
## 5 18:09   자기주식매매신청내역(코스닥시장)
## 6 18:09   대량매매내역(코스닥시장)
##      NA      NA
## 1   화신테크   공시차트\r\n\t\t\t\t\t주가차트
## 2   에스제이케이   공시차트\r\n\t\t\t\t\t주가차트
## 3   아이엠텍   공시차트\r\n\t\t\t\t\t주가차트
## 4   시그넷이브이   공시차트\r\n\t\t\t\t\t주가차트
## 5 코스닥시장본부
## 6 코스닥시장본부
```

POST 형식의 경우 body에 들어가는 쿼리 내용을 바꾸어 원하는 데이터를 받을 수 있습니다. 만일 2019년 1월 4일 공시를 확인하고자 할 경우, 위의 코드에서 selDate만 '2019-01-04'로 변경해주면 됩니다. 아래 코드의 출력 결과물을 2019년 1월 4일 공시와 확인하면 동일한 결과임을 확인할 수 있습니다.

```
Sys.setlocale("LC_ALL", "English")
```

```
## [1] "LC_COLLATE=English_United States.1252;LC_CTYPE=English_United States.1252;LC_MONETARY=English_U
```

```
url = 'http://kind.krx.co.kr/disclosure/todaydisclosure.do'
data = POST(url, body =
  list(
    method = 'searchTodayDisclosureSub',
    currentPageSize = '15',
    pageIndex = '1',
    orderMode = '0',
    orderStat = 'D',
    forward = 'todaydisclosure_sub',
    chose = 'S',
    todayFlag = 'Y',
    selDate = '2019-01-04'
  ))
```

```
data = read_html(data) %>%
  html_table(fill = TRUE) %>%
  .[[1]]
```

```
Sys.setlocale("LC_ALL", "Korean")
```

```
## [1] "LC_COLLATE=Korean_Korea.949;LC_CTYPE=Korean_Korea.949;LC_MONETARY=Korean_Korea.949;LC_NUMERIC=C
```

```
print(head(data))
```

```
##      NA      NA
```



Figure 4.10: 페이지 네비게이션

```
## 1 18:18          휴백셀
## 2 18:15 케이엠더블유
## 3 18:15 스튜디오썸머
## 4 18:14 스튜디오썸머
## 5 18:10   헬릭스미스
## 6 18:10     KJ프리텍
##
##                                     NA
## 1                                     [정정] 최대주주 변경
## 2                                     불성실공시법인지정예고(공시변경)
## 3 [정정] 최대주주 변경을 수반하는 주식 담보제공 계약 해제·취소 등
## 4 [정정] 최대주주 변경을 수반하는 주식 담보제공 계약 체결
## 5                                     공매도 과열종목 지정(공매도 거래 금지 적용)
## 6                                     공매도 과열종목 지정(공매도 거래 금지 적용)
##
##          NA                                     NA
## 1          휴백셀 공시차트\r\n\t\t\t\t\t주가차트
## 2 코스닥시장본부 공시차트\r\n\t\t\t\t\t주가차트
## 3   스튜디오 썸머 공시차트\r\n\t\t\t\t\t주가차트
## 4   스튜디오 썸머 공시차트\r\n\t\t\t\t\t주가차트
## 5 코스닥시장본부 공시차트\r\n\t\t\t\t\t주가차트
## 6 코스닥시장본부 공시차트\r\n\t\t\t\t\t주가차트
```

### 4.2.3 네이버 금융에서 주식티커 크롤링

태그와 속성, 페이지 네비게이션 값을 결합하여 국내 상장 주식의 종목명 및 티커를 추출하는 방법에 대해 알아보도록 하겠습니다. 네이버 금융에서 국내증시 → 시가총액 페이지에는 코스피와 코스닥의 시가총액별 정보가 나타나 있습니다.

- 코스피: [https://finance.naver.com/sise/sise\\_market\\_sum.nhn?sosok=0&page=1](https://finance.naver.com/sise/sise_market_sum.nhn?sosok=0&page=1)
- 코스닥: [https://finance.naver.com/sise/sise\\_market\\_sum.nhn?sosok=1&page=1](https://finance.naver.com/sise/sise_market_sum.nhn?sosok=1&page=1)

또한 각 종목명을 클릭하여 이동하는 페이지의 url을 확인해보면, 끝 6자리가 각 종목의 거래소 티커임도 확인이 됩니다.

티커 정리를 위해 우리가 html에서 확인해야 할 부분은 총 2가지 입니다. 먼저 하단의 페이지 네비게이션을 통해 코스피와 코스닥 시가총액에 해당하는 페이지가 각각 몇번째 페이지까지 존재하는지를 알아야 합니다. 아래와 같은 항목 중 **맨뒤**에 해당하는 페이지가 가장 마지막 페이지에 해당합니다.

맨뒤 글자에 마우스를 올려둔 후 우클릭 → 검사를 선택할 경우 개발자도구 화면이 열리며, 해당 글자가 html 내에서 어떤 부분에 위치하는지 확인할 수 있습니다. ‘맨뒤’에 해당하는 링크는 pgRR 클래스 → a 태그 중 href 속성에 위치하며, page= 뒷부분의 숫자에 위치하는 페이지로 링크가 걸려있습니다.

종목명 링크에 해당하는 주소 중 끝 6자리는 티커에 해당합니다. 따라서 각 링크들의 주소를 알아야 할 필요도 있습니다.

삼성전자에 마우스를 올려둔 후 우클릭 → 검사를 통해 개발자도구 화면을 살펴보면, 해당 링크가 tbody → td → a 태그에서 href 속성에 위치하고 있음을 알 수 있습니다.

위의 정보들을 이용하여 데이터를 다운로드 받도록 하겠습니다. 아래 코드에서  $i = 0$  일 경우 코스피에 해당하는 url 이,  $i = 1$  일 경우 코스닥에 해당하는 url이 생성되며, 먼저 코스피에 해당하는 데이터를 다운로드 받도록 하겠습니다.





```
library(httr)
library(rvest)

i = 0
ticker = list()
url = paste0('https://finance.naver.com/sise/sise_market_sum.nhn?sosok=', i, '&page=1')
print(url)
```

```
## [1] "https://finance.naver.com/sise/sise_market_sum.nhn?sosok=0&page=1"
```

```
down_table = GET(url)
print(down_table)
```

```
## Response [https://finance.naver.com/sise/sise_market_sum.nhn?sosok=0&page=1]
##   Date: 2019-06-13 04:45
##   Status: 200
##   Content-Type: text/html; charset=EUC-KR
##   Size: 95.6 kB
##
##
##
##
##
##
##
## <!-- global include -->
##
##
## ...
```

1. 빈 리스트인 ticker 변수를 만들어 줍니다.
2. paste0() 함수를 이용하여 코스피 시가총액 페이지의 url을 만듭니다.
3. GET() 함수를 통해 해당 페이지 내용을 받아 down\_table 변수에 저장합니다.

down\_table 변수를 확인해보면 Status가 200, 즉 데이터가 이상없이 받아졌으며, 인코딩은 EUC-KR 타입으로 되어 있습니다. 가장 먼저 해야할 작업은 가장 마지막 페이지가 몇번째 페이지인지 찾아내는 작업입니다. 우리는 이미 개발자도구 화면을 통해 해당 정보가 pgRR 클래스의 a태그 중 href 속성에 위치하고 있음을 알고 있습니다.

```
navi.final = read_html(down_table, encoding = 'EUC-KR') %>%
  html_nodes(., '.pgRR') %>%
  html_nodes(., 'a') %>%
  html_attr(., 'href')
```

1. read\_html() 함수를 이용하여 해당 페이지의 html 내용을 읽어오며, 인코딩은 'EUC-KR'로 셋팅해주도록 합니다.
2. html\_nodes() 함수를 이용하여 pgRR 클래스 정보만을 불러오도록 하며, 클래스 속성이므로 앞에 콤마(.)를 붙여 주도록 합니다.
3. html\_nodes() 함수를 통해 a 태그 정보만을 불러오도록 합니다.
4. html\_attr() 함수를 통해 href 속성을 불러오도록 합니다.

이를 통해 navi.final에는 해당 부분에 해당하는 내용이 저장됩니다.

```
print(navi.final)
```

```
## [1] "/sise/sise_market_sum.nhn?sosok=0&page=31"
```

이 중 우리가 알고싶은 내용은 page= 뒤에 존재하는 숫자입니다. 해당 내용을 추출하는 코드는 다음과 같습니다.

```
navi.final = navi.final %>%
  strsplit(., '=') %>%
  unlist() %>%
  tail(., 1) %>%
  as.numeric()
```

1. `strsplit()` 함수는 전체 문장을 특정 글자 기준으로 나누는 것입니다. `page=` 뒷부분만의 데이터가 필요하므로 '='를 기준으로 문장을 나눠주도록 합니다.
2. `unlist()` 함수를 통해 결과를 벡터 형태로 변환합니다.
3. `tail()` 함수를 통해 마지막 첫번째 데이터만 선택합니다.
4. `as.numeric()` 함수를 통해 해당 값을 숫자 형태로 바꾸어 주도록 합니다.

```
print(navi.final)
```

```
## [1] 31
```

for loop 구문을 이용할 경우 1 페이지 부터 `navi.final`, 즉 마지막 페이지까지 모든 페이지의 내용을 읽어올 수 있습니다. 먼저 코스피의 첫번째 페이지에서 우리가 원하는 데이터를 추출하는 방법을 살펴보도록 하겠습니다.

```
i = 0 # 코스피
j = 1 # 첫번째 페이지
url = paste0("https://finance.naver.com/sise/sise_market_sum.nhn?sosok=", i, "&page=", j)
down_table = GET(url)
```

1. `i`와 `j`에 각각 0과 1을 입력하여 코스피 첫번째 페이지에 해당하는 url을 생성해 줍니다.
2. `GET()` 함수를 이용하여 해당 페이지의 데이터를 다운로드 받습니다.

```
Sys.setlocale("LC_ALL", "English")
```

```
## [1] "LC_COLLATE=English_United States.1252;LC_CTYPE=English_United States.1252;LC_MONETARY=English_U"
```

```
table = read_html(down_table, encoding = "EUC-KR") %>% html_table(fill = TRUE)
table = table[[2]]
```

```
Sys.setlocale("LC_ALL", "Korean")
```

```
## [1] "LC_COLLATE=Korean_Korea.949;LC_CTYPE=Korean_Korea.949;LC_MONETARY=Korean_Korea.949;LC_NUMERIC=C"
```

1. `Sys.setlocale()` 함수를 통해 로케일 언어를 영어로 설정 해줍니다.
2. `read_html()` 함수를 통해 html 정보를 읽어옵니다.
3. `html_table()` 함수를 통해 테이블 정보를 읽어오며, `fill=TRUE` 를 추가해줍니다.

4. table 변수에는 리스트 형태로 총 3가지 테이블이 저장되어 있습니다. 첫번째 리스트에는 거래량, 시가, 고가 등 적용 항목, 세번째 리스트에는 페이지 네비게이션 테이블이 저장되어 있으므로, 우리에게 필요한 두번째 리스트만을 table 변수에 다시 저장하도록 합니다.
5. 한글을 읽기 위해 `Sys.setlocale()` 함수를 통해 로케일 언어를 다시 Korean으로 변경해 줍니다.

저장된 table 내용을 확인하면 다음과 같습니다.

```
print(head(table))
```

```
##      N      종목명   현재가 전일비   등락률 액면가   시가총액   상장주식수
## 1 NA
## 2  1   삼성전자  43,650    950 -2.13%    100 2,605,810   5,969,783
## 3  2   SK하이닉스 63,300  2,400 -3.65%   5,000  460,825   728,002
## 4  3     현대차  141,500    500 +0.35%   5,000  302,340   213,668
## 5  4   삼성전자우  36,050    700 -1.90%    100  296,651   822,887
## 6  5     셀트리온 205,500  1,500 +0.74%   1,000  263,716   128,329
##   외국인비율   거래량    PER   ROE   토론실
## 1           NA           <NA> <NA>    NA
## 2      57.10 7,727,537    7.25 19.63    NA
## 3      50.23 3,832,167    2.97 38.53    NA
## 4      44.59  523,410   26.44  2.20    NA
## 5      92.52 1,164,474    5.98  N/A    NA
## 6      20.46  259,540  100.29 10.84    NA
```

이 중 마지막 열인 토론실은 필요가 없는 열이며, 첫번째 행과 같이 아무런 정보가 없는 행이 존재하기도 합니다. 이를 정리하면 다음과 같습니다.

```
table[, ncol(table)] = NULL
table = na.omit(table)
print(head(table))
```

```
##      N      종목명   현재가 전일비   등락률 액면가   시가총액   상장주식수
## 2  1   삼성전자  43,650    950 -2.13%    100 2,605,810   5,969,783
## 3  2   SK하이닉스 63,300  2,400 -3.65%   5,000  460,825   728,002
## 4  3     현대차  141,500    500 +0.35%   5,000  302,340   213,668
## 5  4   삼성전자우  36,050    700 -1.90%    100  296,651   822,887
## 6  5     셀트리온 205,500  1,500 +0.74%   1,000  263,716   128,329
## 10 6      LG화학 342,500  5,500 +1.63%   5,000  241,779    70,592
##   외국인비율   거래량    PER   ROE
## 2      57.10 7,727,537    7.25 19.63
## 3      50.23 3,832,167    2.97 38.53
## 4      44.59  523,410   26.44  2.20
## 5      92.52 1,164,474    5.98  N/A
## 6      20.46  259,540  100.29 10.84
## 10     38.51  144,786   18.21  8.86
```

이제 우리가 필요한 정보는 6자리 티커입니다. 티커 역시 개발자도구 화면을 통해 `tbody` → `td` → `a` 태그에서 `href` 속성에 위치하고 있음을 알고 있으며, 이를 추출하는 코드는 다음과 같습니다.

```
symbol = read_html(down_table, encoding = 'EUC-KR') %>%
  html_nodes(., 'tbody') %>%
  html_nodes(., 'td') %>%
```

```
html_nodes(., 'a') %>%
  html_attr(., 'href')
print(head(symbol, 10))
```

```
## [1] "/item/main.nhn?code=005930" "/item/board.nhn?code=005930"
## [3] "/item/main.nhn?code=000660" "/item/board.nhn?code=000660"
## [5] "/item/main.nhn?code=005380" "/item/board.nhn?code=005380"
## [7] "/item/main.nhn?code=005935" "/item/board.nhn?code=005935"
## [9] "/item/main.nhn?code=068270" "/item/board.nhn?code=068270"
```

1. read\_html() 함수를 통해 html 정보를 읽어오며, 인코딩은 'EUC-KR'로 설정합니다.
2. html\_nodes() 함수를 통해 'tbody' 태그 정보를 불러옵니다.
3. 다시 html\_nodes() 함수를 통해 'td'와 'a' 태그 정보를 불러옵니다.
4. html\_attr() 함수를 이용하여 'href' 속성을 불러옵니다.

이를 통해 symbol에는 href 속성에 해당하는 링크 주소들이 저장되게 됩니다. 이 중 마지막 6자리 글자만 추출하는 코드는 다음과 같습니다.

```
symbol = sapply(symbol, function(x) {
  substr(x, nchar(x) - 5, nchar(x))
})
print(head(symbol, 10))
```

```
## /item/main.nhn?code=005930 /item/board.nhn?code=005930
## "005930" "005930"
## /item/main.nhn?code=000660 /item/board.nhn?code=000660
## "000660" "000660"
## /item/main.nhn?code=005380 /item/board.nhn?code=005380
## "005380" "005380"
## /item/main.nhn?code=005935 /item/board.nhn?code=005935
## "005935" "005935"
## /item/main.nhn?code=068270 /item/board.nhn?code=068270
## "068270" "068270"
```

sapply() 함수를 통해 symbol 변수의 내용들에 function()을 적용하며, substr() 함수 내에 nchar() 함수를 적용하여 마지막 6자리 글자만을 추출하도록 합니다.

결과를 살펴보면 티커에 해당하는 마지막 6글자만 추출된 것이 확인됩니다. 그러나 결과를 살펴보면 동일한 내용이 두번 연속하여 추출됩니다. 이는 main.nhn?code= 에 해당하는 부분은 종목명에 설정된 링크, board.nhn?code= 에 해당하는 부분은 토론실에 설정된 링크이기 때문입니다.

```
symbol = unique(symbol)
print(head(symbol, 10))
```

```
## [1] "005930" "000660" "005380" "005935" "068270" "051910" "055550"
## [8] "005490" "207940" "012330"
```

unique() 함수를 이용하여 중복되는 티커를 제거하면 우리가 원하는 티커 부분만 깔끔하게 정리가 됩니다. 해당 내용을 위에서 구한 table에 입력한 후 데이터를 다듬는 과정은 다음과 같습니다.

```
table$N = symbol
colnames(table)[1] = '종목코드'

rownames(table) = NULL
ticker[[j]] = table
```

1. 'N'열에 위에서 구한 티커를 입력해 줍니다.
2. 해당 열 이름을 '종목코드'로 변경합니다.
3. `na.omit()`을 통해 특정 행을 삭제하였으므로, 행 이름을 초기화 해주도록 합니다.
4. `ticker`의 `j`번째 리스트에 정리된 데이터를 입력해 줍니다.

위의 코드에서 `i`와 `j`값을 `for loop`를 이용하면 코스피와 코스닥 전 종목의 티커가 정리된 테이블을 만들 수 있습니다. 이를 전체 코드로 나타내면 다음과 같습니다.

```
data = list()

# i = 0 은 코스피, i = 1 은 코스닥 종목
for (i in 0:1) {

  ticker = list()
  url = paste0("https://finance.naver.com/sise/sise_market_sum.nhn?sosok=", i, "&page=1")

  down_table = GET(url)

  # 최종 페이지 번호 찾아주기
  navi.final = read_html(down_table, encoding = "EUC-KR") %>%
    html_nodes(., ".pgRR") %>%
    html_nodes(., "a") %>%
    html_attr(., "href") %>%
    strsplit(., "=") %>%
    unlist() %>%
    tail(., 1) %>%
    as.numeric()

  # 첫번째 부터 마지막 페이지까지 for loop를 이용하여 테이블 추출하기
  for (j in 1:navi.final) {

    # 각 페이지에 해당하는 url 생성
    url = paste0("https://finance.naver.com/sise/sise_market_sum.nhn?sosok=", i, "&page=", j)
    down_table = GET(url)

    Sys.setlocale("LC_ALL", "English") # 한글 오류 방지를 위해 영어로 로케일 언어 변경

    table = read_html(down_table, encoding = "EUC-KR") %>% html_table(fill = TRUE)
    table = table[[2]] # 원하는 테이블 추출

    Sys.setlocale("LC_ALL", "Korean") # 한글을 읽기위해 로케일 언어 재변경

    table[, ncol(table)] = NULL # 토론식 부분 삭제
    table = na.omit(table) # 빈 행 삭제

    # 6자리 티커만 추출
    symbol = read_html(down_table, encoding = "EUC-KR") %>%
```

```
html_nodes(., "tbody") %>%
html_nodes(., "td") %>%
html_nodes(., "a") %>%
html_attr(., "href")

symbol = sapply(symbol, function(x) {
  substr(x, nchar(x) - 5, nchar(x))
}) %>% unique()

# 테이블에 티커 넣어준 후, 테이블 정리
table$N = symbol
colnames(table)[1] = "종목코드"

rownames(table) = NULL
ticker[[j]] = table

Sys.sleep(0.5) # 페이지 당 0.5초의 슬립 적용
}

# do.call을 통해 리스트를 데이터 프레임으로 묶기
ticker = do.call(rbind, ticker)
data[[i + 1]] = ticker
}

# 코스피와 코스닥 테이블 묶기
data = do.call(rbind, data)
```





## Chapter 5

# 금융 데이터 수집하기 (기본)

API와 크롤링을 이용한다면 비용을 지불하지 않고 얼마든지 금융 데이터를 수집할 수 있습니다. 본 장에서는 금융 데이터를 받기 위해 필요한 주식티커를 구하는 법, 그리고 섹터별 구성종목을 크롤링하는 법에 대해 알아보도록 하겠습니다.

### 5.1 한국거래소의 산업별 현황 및 개별지표 크롤링

앞 장의 예제를 통해 네이버 금융에서 주식티커 크롤링을 하는 방법에 대해 살펴보았습니다. 그러나 해당 방법은 지나치게 복잡하기도 하며 시간이 오래 걸리기도 합니다. 반면 한국거래소에서 제공하는 산업별 현황과 개별종목 지표 데이터를 이용할 경우 훨씬 간단하게 해당 데이터를 수집할 수 있습니다.

- 산업별 현황: <http://marketdata.krx.co.kr/mdi#document=03030103>
- 개별지표: <http://marketdata.krx.co.kr/mdi#document=13020401>

물론 해당 데이터들을 크롤링이 아닌 Excel 버튼을 눌러 엑셀로 받을수도 있습니다. 그러나 매번 엑셀을 다운받고 이를 R로 불러오는 작업은 상당히 비효율적이며, 크롤링을 이용한다면 해당 데이터를 효율적으로 불러올 수 있습니다.

#### 5.1.1 산업별 현황 크롤링

먼저 산업별 현황에 해당하는 페이지에 접속한 후, 개발자도구 화면을 연 상태에서 Excel 버튼을 눌러줍니다. Network 탭에는 **GenerateOTP.jspx**와 **download.jspx** 총 두가지 항목이 존재합니다. 거래소에서 엑셀 데이터를 받는 과정은 다음과 같습니다.

1. **GenerateOTP.jspx**: <http://marketdata.krx.co.kr/contents/COM/GenerateOTP.jspx>에 원하는 항목을 쿼리로 발송하면 해당 쿼리에 해당하는 OTP를 받게 됩니다.
2. **download.jspx**: 부여받은 OTP를 <http://file.krx.co.kr/download.jspx>에 제출하면 이에 해당하는 데이터를 다운로드 받게 됩니다.

먼저 1번 단계를 살펴보도록 하겠습니다.

General 항목의 Request URL의 앞부분이 원하는 항목을 제출할 주소이며, Query String Parameters에는 우리가 원하는 항목들이 적혀있습니다. 이를 통해 POST 방식으로 데이터를 요청함을 알 수 있습니다.

다음으로 2번 단계를 살펴보도록 하겠습니다.

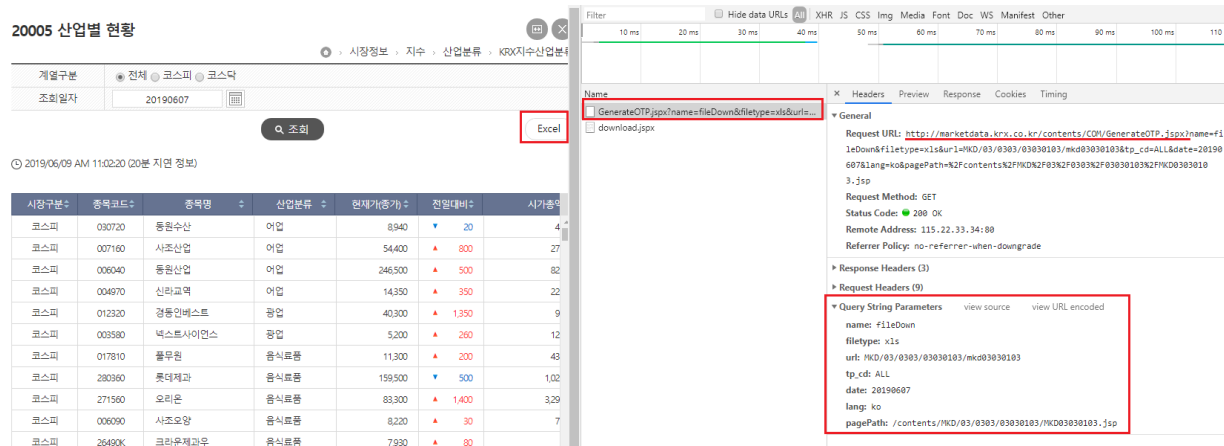


Figure 5.1: OTP 생성 부분

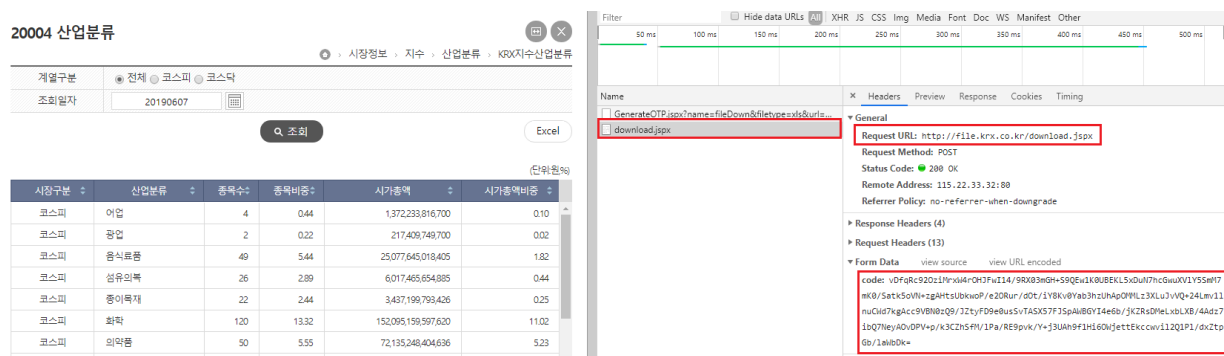


Figure 5.2: OTP 제출 부분

General 항목의 Request URL은 OTP를 제출할 주소이며, Form Data의 OTP는 1번 단계에서 부여받은 OTP에 해당합니다. 이 역시 POST 방식으로 데이터를 요청합니다.

이러한 과정을 코드로 나타내면 다음과 같습니다.

```
library(httr)
library(rvest)
library(readr)

gen_otp_url = 'http://marketdata.krx.co.kr/contents/COM/GenerateOTP.jspx'
gen_otp_data = list(name = 'fileDown',
                    filetype = 'csv',
                    url = 'MKD/03/0303/03030103/mkd03030103',
                    tp_cd = 'ALL',
                    date = '20190607',
                    lang = 'ko',
                    pagePath = '/contents/MKD/03/0303/03030103/MKD03030103.jsp')
otp = POST(gen_otp_url, query = gen_otp_data) %>%
  read_html() %>%
  html_text()
```

1. gen\_otp\_url에 원하는 항목을 제출할 url을 입력합니다.
2. 개발자도구 화면에 나타나는 쿼리 내용들을 리스트 형태로 입력합니다. 단, filetype은 기존 xls이 아닌 csv로 변경하여 주며, 이는 csv 형태로 다운로드 받을 경우 데이터를 처리하기 훨씬 쉽기 때문입니다.
3. POST() 함수를 통해 해당 url에 쿼리를 전송하면 이에 해당하는 데이터를 받게 됩니다.
4. read\_html() 함수를 통해 html 내용을 읽어옵니다.
5. html\_text() 함수는 html 내에서 텍스트에 해당하는 부분만을 추출하며, 이를 통해 otp 값을 추출하게 됩니다.

위의 과정을 거쳐 생성된 OTP는 다음과 같습니다.

```
print(otp)
```

```
## [1] "vDfqRc920ziMrxW4r0HJfXTZsN8e8XYP94rhdwsrOUL8U8N9GChJE3fa0cogdFyBfvHsDLoyf7fa0c2hWue9mj9IDtcym7S"
```

이제 생성된 OTP를 제출하면, 우리가 원하는 데이터를 다운로드 받을 수 있습니다.

```
down_url = 'http://file.krx.co.kr/download.jspx'
down_sector = POST(down_url, query = list(code = otp),
                  add_headers(referer = gen_otp_url)) %>%
  read_html() %>%
  html_text() %>%
  read_csv()
```

1. OTP를 제출할 url을 down\_url에 입력합니다.
2. POST() 함수를 통해 해당 url에 위에서 부여받은 OTP 코드를 제출합니다.
3. add\_headers() 구문을 통해 referer를 추가해 주어야 합니다. 리퍼러란 링크를 통해서 각각의 사이트로 방문시 남는 흔적입니다. 데이터를 다운로드 받는 과정을 살펴보면 첫번째 url에서 OTP를 부여 받고, 이를 다시 두번째 url에 제출하였습니다. 그런데 이러한 과정의 흔적이 없이 OTP를 바로 두번째 url에 제출하면 서버는 이를 로봇으로 인식하여 데이터를 반환하지 않습니다. 따라서 add\_headers()를 통해 우리가 거처온 과정을 흔적으로 남겨야 사람이 데이터를 다운로드 받는 과정과 동일하게 인식하여 데이터를 반환하게 되며, 첫번째 url을 리퍼러로 지정해 줍니다.

4. `read_html()`과 `html_text()` 함수를 통해 텍스트 데이터만 추출합니다.
5. `read_csv` 함수는 csv 형태의 데이터를 불러옵니다. 위의 요청 쿼리에서 `filetype`을 csv로 지정했기에, 손쉽게 데이터를 읽어올 수 있습니다.

```
print(down_sector)
```

```
## # A tibble: 2,243 x 7
##   시장구분 종목코드 종목명 산업분류 `현재가(종가)` 전일대비 `시가총액(원)`
##   <chr>      <chr>      <chr> <chr>          <dbl>      <dbl>          <dbl>
## 1 코스피    030720    동원수산~ 어업          8940        -20    41605016700
## 2 코스피    007160    사조산업~ 어업         54400         800   272000000000
## 3 코스피    006040    동원산업~ 어업        246500         500   829028800000
## 4 코스피    004970    신라교역~ 어업         14350         350   229600000000
## 5 코스피    012320    경동인베스~ 광업        40300        1350   95310426900
## 6 코스피    003580    넥스트사이~ 광업         5200         260   122099322800
## 7 코스피    017810    풀무원    음식료품        11300         200   430427735000
## 8 코스피    280360    롯데제과~ 음식료품       159500        -500  1023466361500
## 9 코스피    271560    오리온    음식료품        83300        1400  3293359795600
## 10 코스피   006090    사조오양~ 음식료품         8220         30    77454914580
## # ... with 2,233 more rows
```

위 과정을 통해 `down_sector` 변수에는 산업별 현황 데이터가 저장되었습니다. 이를 csv 파일로 다운로드 받도록 하겠습니다.

```
ifelse(dir.exists('data'), FALSE, dir.create('data'))
```

```
## [1] FALSE
```

```
write.csv(down_sector, 'data/KOR_sector.csv')
```

먼저 `ifelse()` 함수를 통해 `data`라는 이름의 폴더가 존재할 시에는 `FALSE`를 반환, 존재하지 않을 시 해당 이름으로 폴더를 생성하여 줍니다. 그 후, 위에서 다운로드 받은 데이터를 'KOR\_sector.csv' 이름으로 저장하여 줍니다. 해당 폴더를 확인해보면, 데이터가 csv 형태로 저장되어 있음이 확인할 수 있습니다.

### 5.1.2 개별종목 지표 크롤링

개별종목 데이터를 크롤링 하는 방법은 위와 거의 동일하며, 요청하는 쿼리 값에만 차이가 있습니다. 위와 동일하게 개발자도구 화면을 연 상태에서 csv 버튼을 눌러주어 어떠한 쿼리를 요청하는지 확인하도록 합니다.

이 중 `isu_cdnm`, `isu_cd`, `isu_nm`, `isu_srt_cd`, `fromdate` 항목은 종목구분의 개별탭에 해당하는 부분이므로 우리가 원하는 전체 데이터를 받을때에는 필요하지 않은 요청값입니다. 이를 제외한 요청값을 산업별 현황 예제에 적용하면 해당 데이터 역시 손쉽게 다운로드 받을 수 있습니다.

```
library(httr)
library(rvest)
library(readr)

gen_otp_url = 'http://marketdata.krx.co.kr/contents/COM/GenerateOTP.aspx'
gen_otp_data = list(name = 'fileDown',
                    filetype = 'csv',
                    url = "MKD/13/1302/13020401/mkd13020401",
```

Figure 5.3 shows the process of generating an OTP for individual indicators. On the left, the KOSDAQ website interface is visible, with the 'CSV' button highlighted. On the right, the browser's developer tools show the 'Headers' tab, displaying the request parameters for the OTP generation endpoint. The parameters include:

- name: fileDown
- filetype: csv
- url: MKD/13/1302/13020401/mkd13020401
- market\_gubun: ALL
- gubun: 1
- isu\_cdmn: A005930/삼성전자
- isu\_cd: K7005930003
- isu\_nm: 삼성전자
- isu\_srt\_cd: A005930
- schdate: 20190607
- fromdate: 20190531
- todate: 20190607
- pagePath: /contents/MKD/13/1302/13020401/MKD13020401.jsp

Figure 5.3: 개별지표 OTP 생성 부분

```
market_gubun = 'ALL',
gubun = '1',
schdate = '20190607',
pagePath = "/contents/MKD/13/1302/13020401/MKD13020401.jsp")

otp = POST(gen_otp_url, query = gen_otp_data) %>%
  read_html() %>%
  html_text()

down_url = 'http://file.krx.co.kr/download.jspx'
down_ind = POST(down_url, query = list(code = otp),
  add_headers(referer = gen_otp_url)) %>%
  read_html() %>%
  html_text() %>%
  read_csv()

print(down_ind)
```

```
## # A tibble: 2,204 x 13
##   일자      종목코드 종목명  관리여부  증가 EPS  PER  BPS  PBR
##   <date>    <chr>    <chr>  <chr>    <dbl> <chr> <chr> <chr> <chr>
## 1 2019-06-07 060300 레드로버~ -      1190 -    -    2,128 0.56
## 2 2019-06-07 290650 엘앤씨바이~ -      22750 830  27.41 7,252 3.14
## 3 2019-06-07 239340 미래에셋제~ -      5200 -    -    -    -
## 4 2019-06-07 086060 진바이오텍~ -      6940 312  22.24 5,570 1.25
## 5 2019-06-07 033430 디에스티~ -      1310 -    -    338  3.88
## 6 2019-06-07 038680 에스넷 -      9600 300  32    4,302 2.23
## 7 2019-06-07 214680 디알텍 -      2005 29   69.14 709  2.83
## 8 2019-06-07 242040 나무기술~ -      3345 -    -    462  7.24
```

```
## 9 2019-06-07 121890 에스디시스~ - 4980 - - 1,912 2.6
## 10 2019-06-07 088800 에이스테크~ - 11500 97 118.~ 2,291 5.02
## # ... with 2,194 more rows, and 4 more variables: 주당배당금 <dbl>,
## # 배당수익률 <dbl>, `개시물 일련번호` <dbl>, 총카운트 <dbl>
```

위 과정을 통해 `down_ind` 변수에는 개별종목 지표 데이터가 저장되었습니다. 해당 데이터 역시 csv 파일로 다운로드 받도록 하겠습니다.

```
write.csv(down_ind, 'data/KOR_ind.csv')
```

위 예제의 쿼리 항목 중 `date`와 `schdate` 부분만 원하는 일자로 입력할 경우(예: 20190104), 해당일의 데이터를 다운로드 받을 수 있습니다. 단, 휴장일을 입력할 경우 데이터를 받을 수 없습니다.

### 5.1.3 데이터 정리하기

위에서 다운받은 데이터는 중복된 열이 있으며, 불필요한 데이터 역시 존재합니다. 따라서 하나의 테이블로 합쳐준 후 정리를 할 필요가 있습니다. 먼저 다운로드 받은 csv 파일을 읽어오도록 합니다.

```
down_sector = read.csv('data/KOR_sector.csv', row.names = 1, stringsAsFactors = FALSE)
down_ind = read.csv('data/KOR_ind.csv', row.names = 1, stringsAsFactors = FALSE)
```

`read.csv()` 함수를 이용하여 데이터를 읽어오며, `row.names = 1`를 통해 첫번째 열을 행이름으로, `stringsAsFactors = FALSE`를 통해 캐릭터 데이터가 팩터 형태로 변형되지 않게 합니다.

```
intersect(names(down_sector), names(down_ind))
```

```
## [1] "종목코드" "종목명"
```

먼저 `intersect()` 함수를 통해 두 데이터간 중복되는 열이름을 살펴보면, 종목코드와 종목명이 동일하게 위치합니다.

```
setdiff(down_sector[, '종목명'], down_ind[, '종목명'])
```

```
## [1] "엘브이엠씨홀딩스" "한국패러렐" "한국ANKOR유전"
## [4] "맵스리얼티1" "맥쿼리인프라" "하나니켈2호"
## [7] "하나니켈1호" "베트남개발1" "신한알파리츠"
## [10] "이리츠코크렙" "모두투어리츠" "하이골드12호"
## [13] "하이골드8호" "바다로19호" "하이골드3호"
## [16] "케이탑리츠" "에이리츠" "동북아13호"
## [19] "동북아12호" "컬러레이" "JTC"
## [22] "뉴프라이드" "윙임푸드" "글로벌에스엠"
## [25] "크리스탈신소재" "씨케이에이치" "차이나그레이트"
## [28] "골든센츄리" "오가닉티코스메틱" "GRT"
## [31] "로스웰" "형성그룹" "이스트아시아홀딩스"
## [34] "에스앤씨엔진그룹" "SNK" "SBI핀테크솔루션즈"
## [37] "잉글우드랩" "코오롱티슈진" "엑세스바이오"
```

`setdiff()` 함수를 통해 두 데이터에 공통적으로 존재하지 않는 종목명, 즉 하나의 데이터에만 존재하는 종목을 살펴보면 위와 같습니다. 해당 종목들은 **선박펀드**, **광물펀드**, **해외종목** 등 일반적이지 않은 종목들이므로, 제외해주는 것이 좋습니다. 따라서 둘간에 공통적으로 존재하는 종목을 기준으로 데이터를 합쳐주도록 하겠습니다.

```
KOR_ticker = merge(down_sector, down_ind,
                    by = intersect(names(down_sector), names(down_ind)),
                    all = FALSE
                    )
```

merge() 함수는 by를 기준으로 하여 두 데이터를 하나로 합치게 되며, 그 기준은 공통으로 존재하는 열이름으로 합니다. 또한 all 값을 TRUE로 설정할 경우는 합집합을, FALSE로 설정할 경우 교집합을 반환하며, 공통적으로 존재하는 항목을 원하므로 FALSE를 선택해 주도록 합니다.

```
KOR_ticker = KOR_ticker[order(-KOR_ticker['시가총액.원.']), ]
print(head(KOR_ticker))
```

```
##      종목코드      종목명 시장구분 산업분류 현재가.증가. 전일대비
## 330      005930      삼성전자 코스피 전기전자      44200      300
## 45       000660      SK하이닉스 코스피 전기전자      65400      300
## 301      005380      현대차      코스피 운수장비      140000     -1000
## 331      005935      삼성전자우 코스피 전기전자      36250      200
## 1278     068270      셀트리온 코스피 의약품      196500      500
## 1082     051910      LG화학      코스피 화학      330500     -1000
##      시가총액.원.      일자 관리여부      증가      EPS      PER      BPS      PBR
## 330  2.638644e+14  2019-06-07      -      44200  6,461  6.84  35,342  1.25
## 45   4.761135e+13  2019-06-07      -      65400  22,255  2.94  64,348  1.02
## 301  2.991355e+13  2019-06-07      -      140000  5,632  24.86  245,447  0.57
## 331  2.982964e+13  2019-06-07      -      36250      -      -      -      -
## 1278 2.521666e+13  2019-06-07      -      196500  2,063  95.25  19,766  9.94
## 1082 2.333077e+13  2019-06-07      -      330500  19,217  17.2  218,227  1.51
##      주당배당금 배당수익률 게시물..일련번호 총카운트
## 330      1416      3.20      1456      NA
## 45       1500      2.29      1408      NA
## 301      4000      2.86      1450      NA
## 331      1417      3.91      2085      NA
## 1278      0      0.00      1595      NA
## 1082     6000      1.82      1536      NA
```

데이터를 시가총액 순으로 내림차순 해줄 필요도 있습니다. order() 함수를 통해 상대적인 순서를 구할 수 있으며, R은 기본적으로 오름차순으로 순서를 구하므로 앞에 마이너스(-)를 붙여 내림차순 형태로 바꾸어 주도록 합니다. 결과적으로 시가총액 기준 내림차순으로 해당 데이터가 정렬됩니다.

마지막으로 스팩, 우선주 종목 역시 제외해 주어야 합니다.

```
KOR_ticker[grep('스팩', KOR_ticker[, '종목명']), '종목명'] # 스팩
```

```
## [1] "미래에셋제5호스팩" "한화에스비아이스팩" "엔에이치스팩14호"
## [4] "엔에이치스팩10호" "엔에이치스팩12호" "삼성스팩2호"
## [7] "대신밸런스제5호스팩" "케이비제10호스팩" "엔에이치스팩11호"
## [10] "유진스팩4호" "신한제4호스팩" "하나금융11호스팩"
## [13] "케이비17호스팩" "DB금융스팩7호" "SK4호스팩"
## [16] "한국제7호스팩" "대신밸런스제6호스팩" "미래에셋대우스팩1호"
## [19] "대신밸런스제4호스팩" "IBKS제5호스팩" "동부스팩5호"
## [22] "DB금융스팩6호" "하나머스트제6호스팩" "하나금융10호스팩"
## [25] "삼성머스트스팩3호" "유안타제4호스팩" "한국제6호스팩"
## [28] "IBKS제6호스팩" "신영스팩4호" "한화수성스팩"
```

```
## [31] "IBKS제10호스팩"      "하이제4호스팩"      "하나금융9호스팩"
## [34] "유안타제3호스팩"      "교보7호스팩"        "한국제5호스팩"
## [37] "IBKS제9호스팩"        "교보8호스팩"        "IBKS제7호스팩"
## [40] "신한제3호스팩"        "키움제5호스팩"      "SK3호스팩"
## [43] "한국제8호스팩"        "한화에이스스팩4호"  "엔에이치스팩13호"
## [46] "미래에셋대우스팩2호"  "한화에이스스팩3호"  "케이비제11호스팩"
```

```
KOR_ticker[KOR_ticker[, '종목명'] == '골든브릿지이안5호', '종목명'] # 골든브릿지이안5호
```

```
## [1] "골든브릿지이안5호"
```

```
KOR_ticker[substr(KOR_ticker[, '종목명'],
                  nchar(KOR_ticker[, '종목명']), nchar(KOR_ticker[, '종목명'])) == '우', '종목명']
```

```
## [1] "삼성전자우"          "미래에셋대우"        "현대차우"
## [4] "LG생활건강우"        "LG화학우"            "아모레퍼시픽우"
## [7] "삼성화재우"          "LG전자우"            "신영증권우"
## [10] "연우"                 "두산우"              "한국금융지주우"
## [13] "대신증권우"          "S-Oil우"             "대림산업우"
## [16] "NH투자증권우"        "아모레G우"           "CJ제일제당우"
## [19] "LG우"                 "SK이노베이션우"      "삼성SDI우"
## [22] "삼성전기우"          "CJ우"                "금호석유우"
## [25] "SK우"                 "GS우"                "미래에셋대우우"
## [28] "롯데칠성우"          "부국증권우"          "코오롱인더우"
## [31] "롯데지주우"          "유한양행우"          "유화증권우"
## [34] "호텔신라우"          "SK케미칼우"          "남양유업우"
## [37] "한진칼우"            "LG하우시스우"        "BYC우"
## [40] "유안타증권우"        "대한항공우"          "세방우"
## [43] "SK디스커버리우"      "대덕전자1우"         "태영건설우"
## [46] "대상우"              "금호산업우"          "한화우"
## [49] "현대건설우"          "한화케미칼우"        "삼양홀딩스우"
## [52] "녹십자홀딩스2우"     "신풍제약우"          "삼양사우"
## [55] "넥센우"              "SK증권우"            "남산알미우"
## [58] "코오롱우"            "계양전기우"          "NPC우"
## [61] "한화투자증권우"      "SK네트웍스우"        "태양금속우"
## [64] "쌍용양회우"          "서울식품우"          "성문전자우"
## [67] "대원전선우"          "일양약품우"          "성신양회우"
## [70] "대한제당우"          "유유제약1우"         "코리아씨우"
## [73] "크라운해태홀딩스우"  "동원시스템즈우"      "CJ씨푸드1우"
## [76] "현대비앤지스틸우"    "삼성중공우"          "금강공업우"
## [79] "대호피앤씨우"        "크라운제과우"        "동부제철우"
## [82] "깨끗한나라우"        "노루페인트우"       "대상홀딩스우"
## [85] "덕성우"              "코오롱글로벌우"      "동양우"
## [88] "신원우"              "DB하이텍1우"         "하이트진로홀딩스우"
## [91] "흥국화재우"          "JW중외제약우"        "한양증권우"
## [94] "동부건설우"          "노루홀딩스우"        "소프트센우"
```

```
KOR_ticker[substr(KOR_ticker[, '종목명'],
                  nchar(KOR_ticker[, '종목명']) - 1, nchar(KOR_ticker[, '종목명'])) == '우B', '종목명']
```

```
## [1] "현대차2우B"          "미래에셋대우2우B"    "한화3우B"
## [4] "현대차3우B"          "삼성물산우B"         "대교우B"
```



```
## [7] "대신증권2우B"      "두산2우B"          "넥센타이어1우B"
## [10] "하이트진로2우B"    "진흥기업2우B"      "진흥기업우B"
## [13] "JW중외제약2우B"    "흥국화재2우B"      "코리아씨키트2우B"
## [16] "유유제약2우B"      "동양2우B"          "대한제당3우B"
## [19] "동양3우B"
```

```
KOR_ticker[substr(KOR_ticker[, '종목명'],
                  nchar(KOR_ticker[, '종목명']) - 1, nchar(KOR_ticker[, '종목명'])) == '우C', '종목명']
```

```
## [1] "루트로닉3우C"
```

grep1() 함수를 통해 종목명에 '스팩'이 들어가는 종목, 스�팩 종목인 '골든브릿지이안5호', substr() 함수를 통해 종목명 끝이 '우', '우B', '우C'인 우선주 종목을 찾을 수 있습니다. 데이터 내에서 해당 데이터들을 제거<sup>1</sup>해 주도록 하겠습니다.

```
KOR_ticker = KOR_ticker[!grep1('스팩', KOR_ticker[, '종목명']), ] # 스�팩
KOR_ticker = KOR_ticker[KOR_ticker[, '종목명'] != '골든브릿지이안5호', ] # 골든브릿지이안5호
KOR_ticker = KOR_ticker[substr(KOR_ticker[, '종목명'],
                              nchar(KOR_ticker[, '종목명']), nchar(KOR_ticker[, '종목명'])) != '우', ]
KOR_ticker = KOR_ticker[substr(KOR_ticker[, '종목명'],
                              nchar(KOR_ticker[, '종목명']) - 1, nchar(KOR_ticker[, '종목명'])) != '우B', ]
KOR_ticker = KOR_ticker[substr(KOR_ticker[, '종목명'],
                              nchar(KOR_ticker[, '종목명']) - 1, nchar(KOR_ticker[, '종목명'])) != '우C', ]
```

마지막으로 행이름을 초기화 한 후, 정리된 데이터를 csv 파일로 저장해주도록 합니다.

```
rownames(KOR_ticker) = NULL
write.csv(KOR_ticker, 'data/KOR_ticker.csv')
```

## 5.2 WICS 기준 섹터정보 크롤링

일반적으로 주식의 섹터를 나누는 기준은 MSCI와 S&P가 개발한 GICS<sup>2</sup>를 가장 많이 사용합니다. 국내 종목의 GICS 기준 정보 역시 한국거래소에서 제공하고 있으나, 이는 독점적 지적재산으로 명시했기에 사용하는데 무리가 있습니다.

그러나 지수제공업체인 와이즈인덱스<sup>3</sup>에서는 GICS와 비슷한 WICS 산업분류를 발표하고 있으므로, 이를 크롤링하여 필요한 정보를 수집해보도록 하겠습니다.

먼저, 웹페이지에 접속하여 **Index → WISE SECTOR INDEX → WICS → 에너지**를 클릭합니다. 그 후 **Components** 탭을 클릭하면, 해당 섹터의 구성종목을 확인할 수 있습니다.

개발자도구 화면을 통해 해당 페이지의 데이터전송 과정을 살펴보도록 하겠습니다.

일자를 선택하면 Network 탭의 **GetIndexComponets** 항목을 통해 데이터 전송과정이 나타나며, Request URL의 주소를 살펴보면 다음과 같습니다.

[http://www.wiseindex.com/Index/GetIndexComponets?ceil\\_yn=0&dt=20190607&sec\\_cd=G10](http://www.wiseindex.com/Index/GetIndexComponets?ceil_yn=0&dt=20190607&sec_cd=G10)

1. <http://www.wiseindex.com/Index/GetIndexComponets>: 데이터를 요청하는 url 입니다.

<sup>1</sup> 해당 과정에서 미래에셋대우, 연우 등 의도치 않은 종목 역시 제거됩니다. 그러나 이러한 종목수가 그리 많지 않으므로 투자에 있어 중요하지는 않습니다.

<sup>2</sup> [https://en.wikipedia.org/wiki/Global\\_Industry\\_Classification\\_Standard](https://en.wikipedia.org/wiki/Global_Industry_Classification_Standard)

<sup>3</sup> <http://www.wiseindex.com/>

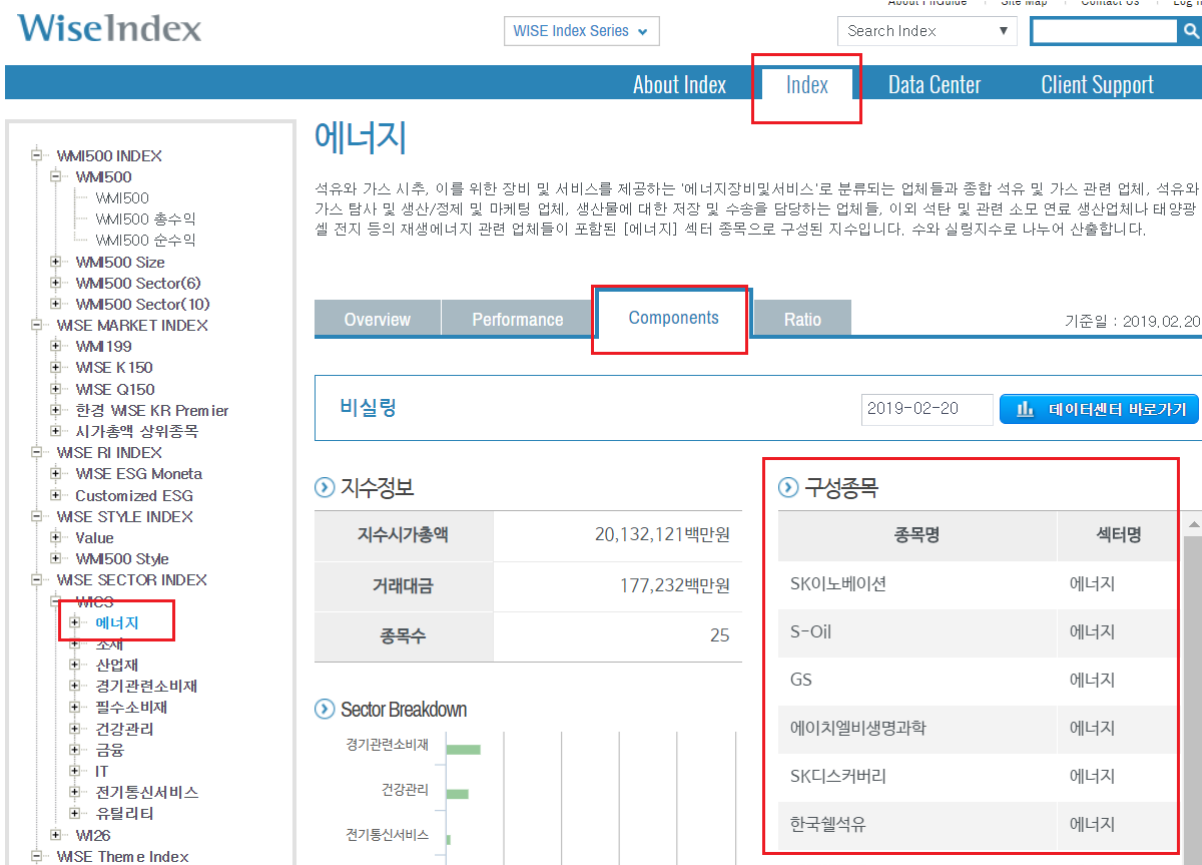


Figure 5.4: WICS 기준 구성종목

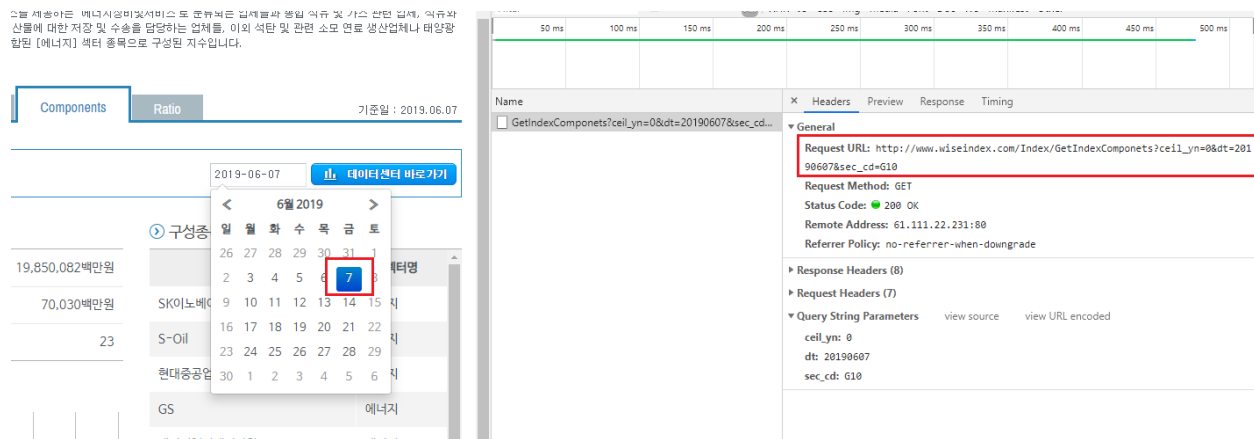


Figure 5.5: WICS 페이지 개발자도구 화면



## 6	G10 WICS 에너지	19850082 006120	SK디스커버리	257059	1.30
## 7	G10 WICS 에너지	19850082 002960	한국셀석유	198881	1.00
## 8	G10 WICS 에너지	19850082 091090	세원셀론텍	146439	0.74
## 9	G10 WICS 에너지	19850082 011930	신성이엔지	118388	0.60
## 10	G10 WICS 에너지	19850082 128820	대성산업	79162	0.40
## 11	G10 WICS 에너지	19850082 043200	파루	75903	0.38
## 12	G10 WICS 에너지	19850082 003650	미창석유	71918	0.36
## 13	G10 WICS 에너지	19850082 014530	극동유화	62828	0.32
## 14	G10 WICS 에너지	19850082 099220	SDN	55967	0.28
## 15	G10 WICS 에너지	19850082 024060	흥구석유	53460	0.27
## 16	G10 WICS 에너지	19850082 041590	에너지전트	50901	0.26
## 17	G10 WICS 에너지	19850082 095910	에스에너지	46873	0.24
## 18	G10 WICS 에너지	19850082 060900	퍼시픽바이오	43739	0.22
## 19	G10 WICS 에너지	19850082 012320	경동인베스트	40030	0.20
## 20	G10 WICS 에너지	19850082 093230	이아이디	32266	0.16
## 21	G10 WICS 에너지	19850082 038620	위즈코프	30094	0.15
## 22	G10 WICS 에너지	19850082 137950	제이씨케미칼	29363	0.15
## 23	G10 WICS 에너지	19850082 000440	중앙에너지비스	10711	0.05

##	S_WGT	CAL_WGT	SEC_CD	SEC_NM_KOR	SEQ	TOP60	APT_SHR_CNT
## 1	45.61	1	G10	에너지	1	2	56403994
## 2	62.75	1	G10	에너지	2	2	41655633
## 3	77.23	1	G10	에너지	3	2	9283372
## 4	89.78	1	G10	에너지	4	2	49245150
## 5	92.93	1	G10	에너지	5	2	39307272
## 6	94.22	1	G10	에너지	6	2	10470820
## 7	95.22	1	G10	에너지	7	2	611000
## 8	95.96	1	G10	에너지	8	2	42693554
## 9	96.56	1	G10	에너지	9	2	100756131
## 10	96.96	1	G10	에너지	10	2	15832417
## 11	97.34	1	G10	에너지	11	2	29707590
## 12	97.70	1	G10	에너지	12	2	922026
## 13	98.02	1	G10	에너지	13	2	18132098
## 14	98.30	1	G10	에너지	14	2	38598075
## 15	98.57	1	G10	에너지	15	2	9000000
## 16	98.83	1	G10	에너지	16	2	31132353
## 17	99.06	1	G10	에너지	17	2	10290527
## 18	99.28	1	G10	에너지	18	2	53017111
## 19	99.48	1	G10	에너지	19	2	993310
## 20	99.65	1	G10	에너지	20	2	140285009
## 21	99.80	1	G10	에너지	21	2	24075079
## 22	99.95	1	G10	에너지	22	2	8353752
## 23	100.00	1	G10	에너지	23	2	1556783

##

## \$sector

##	SEC_CD	SEC_NM_KOR	SEC_RATE	IDX_RATE
## 1	G25	경기관련소비재	16.05	0
## 2	G35	건강관리	9.27	0
## 3	G50	전기통신서비스	2.26	0
## 4	G40	금융	10.31	0
## 5	G10	에너지	2.37	100
## 6	G20	산업재	12.68	0
## 7	G55	유틸리티	1.70	0
## 8	G30	필수소비재	3.75	0
## 9	G15	소재	7.89	0

```
## 10      G45          IT    33.73      0
##
## $size
##   SEC_CD   SEC_NM_KOR SEC_RATE IDX_RATE
## 1 WMI510 WMI500 대형주    69.40    89.78
## 2 WMI520 WMI500 중형주    13.56     4.44
## 3 WMI530 WMI500 소형주    17.04     5.78
```

\$list 항목에는 해당 섹터의 구성종목 정보가 있으며, \$sector 항목을 통해 다른 섹터들의 코드도 확인할 수 있습니다. for loop 구문을 이용해 url의 sec\_cd=에 해당하는 부분만 변경해주면, 모든 섹터의 구성종목을 매우 쉽게 얻을 수 있습니다.

```
sector_code = c('G25', 'G35', 'G50', 'G40', 'G10', 'G20', 'G55', 'G30', 'G15', 'G45')
data_sector = list()

for (i in sector_code) {

  url = paste0(
    'http://www.wiseindex.com/Index/GetIndexComponets?ceil_yn=0&dt=20190607&sec_cd=', i)
  data = fromJSON(url)
  data = data$list

  data_sector[[i]] = data

  Sys.sleep(1)
}

data_sector = do.call(rbind, data_sector)
```

해당 데이터 역시 csv 파일로 저장해주도록 합니다.

```
write.csv(data_sector, 'data/KOR_sector.csv')
```



## Chapter 6

# 금융 데이터 수집하기 (심화)

지난 장에서는 주식티커를 구하였습니다. 이를 바탕으로 이번 장에서는 퀀트 투자의 핵심 자료인 수정주가, 재무제표 및 가치지표를 크롤링 하는법에 대해 알아보도록 하겠습니다.

### 6.1 수정주가 크롤링

주가 데이터는 투자를 함에 있어 반드시 필요한 데이터이며, 인터넷에서 주가를 수집할 수 있는 방법은 매우 많습니다. 먼저, API를 이용한 데이터 수집에서 살펴본 것과 같이, `getSymbols()` 함수를 이용하여 데이터를 받을 수 있습니다. 그러나 야후 파이낸스에서 제공하는 데이터의 경우 미국 주가는 이상없이 다운로드가 되지만, 국내 중소형주의 경우 주가가 없는 경우가 있습니다.

또한 단순 주가를 구할수 있는 방법은 많지만, 투자에 필요한 수정주가를 구할 수 있는 방법은 찾기 힘듭니다. 다행히 네이버 금융에서 제공하는 정보를 통해 모든 종목의 수정주가를 매우 손쉽게 구할 수 있습니다.

#### 6.1.1 개별 종목 주가 크롤링

먼저 네이버 금융에서 특정종목(예: 삼성전자)의 차트 탭<sup>1</sup>을 선택합니다.

위와 같이 플래쉬가 차단되어 화면이 나오지 않는 경우, 주소창의 왼쪽 상단에 위치한 자물쇠 버튼을 클릭한 다음, Flash를 허용으로 바꾼 후 새로고침을 누르면 차트가 나오게 됩니다.

해당 차트는 주가 데이터를 받아 그림을 그려주는 형태입니다. 따라서 해당 데이터가 어디에서 오는지 알기 위해 개발자도구 화면을 이용하도록 합니다.

화면을 연 상태에서 일봉 탭을 선택하면 **sise.nhn**, **schedule.nhn**, **notice.nhn** 총 3가지 항목이 생성됩니다. 이 중 sise.nhn 항목의 Request URL이 주가 데이터를 요청하는 주소입니다. 해당 url에 접속해 보도록 하겠습니다.

각 날짜별로 시가, 고가, 저가, 종가, 거래량이 있으며, 주가의 경우 모두 수정주가 기준입니다. 또한 해당 데이터가 item 태그 내 data 속성에 위치하고 있습니다.

위 주소에서 symbol= 뒤에 위치하는 6자리 티커만 변경하면 해당 종목의 주가 데이터가 위치한 페이지로 이동할 수 있으며, 우리가 원하는 모든 종목들의 주가 데이터를 크롤링 할 수 있습니다. 이러한 티커의 경우 우리는 이미 거래소에서 받은 데이터를 통해 가지고 있습니다.

```
library(stringr)

KOR_ticker = read.csv('data/KOR_ticker.csv', row.names = 1)
print(KOR_ticker$'종목코드'[1])
```

<sup>1</sup><https://finance.naver.com/item/fchart.nhn?code=005930>



이 사이트는 보안 연결(HTTPS)이 사용되었습니다.

비밀번호나 신용카드 번호 등의 정보는 비공개 상태로 이 사이트에 전송됩니다. 자세히 알아보기

Flash

요청(기본값)

허용

차단

인증서: (유효)

쿠키 (18개 사용 중)개

사이트 설정

NAVER 금융

종목명·펀드명·환율명·원자재명 입력

통합검색

금융 홈 국내증시 해외증시 시장지표 펀드 투자전략 뉴스 MY 추천종목

· (중요) MY 매매내역 및 MY펀드서비스가 종료되었습니다. · 해외 증시휴장일 정보 제공 종료 · [한국거래소] 공매도

삼성전자 005930 코스피 2019.06.07 기준(장마감) 실시간 기업개요

44,200

전일대비 ▲300 +0.68%

전일 43,900 고가 44,350 (상한가 57,000) 거래량 11,620,372

시가 43,600 저가 43,450 (하한가 30,750) 거래대금 511,238 백만

종합정보 시세 차트 투자자별 매매동향 뉴스·공시 종목분석 종목토론실 전자공시 공매도현황

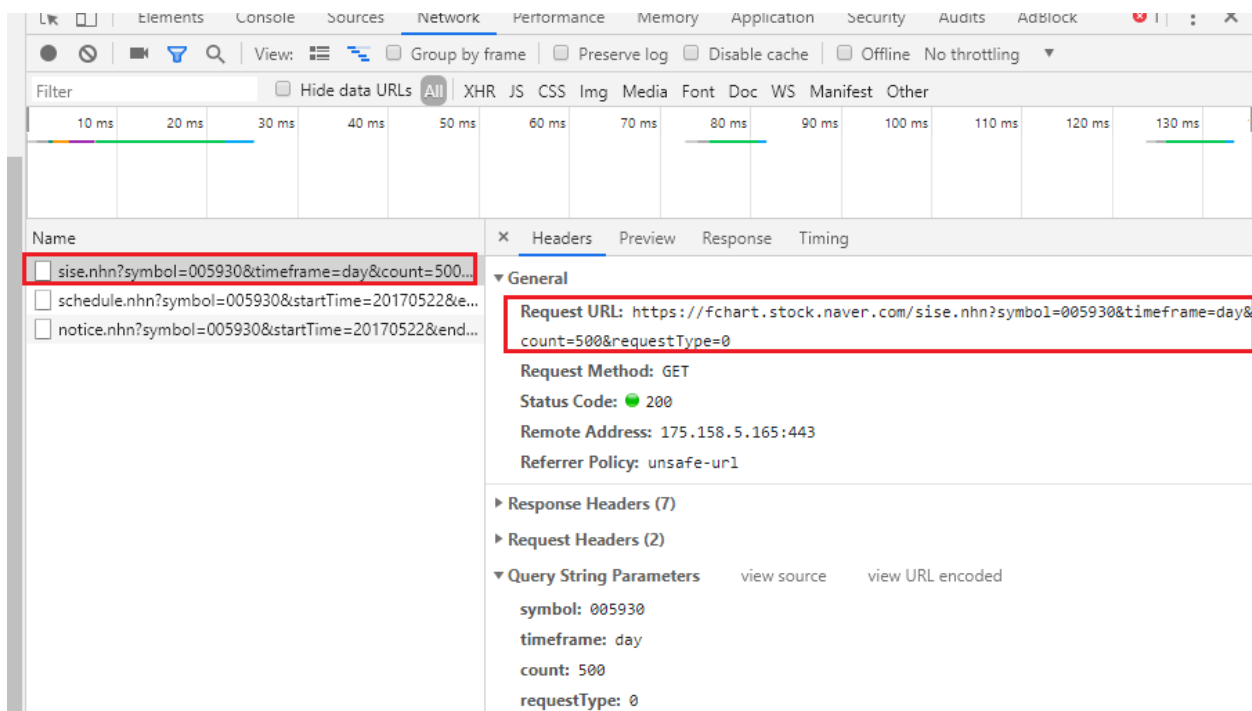
안내 드립니다.

고객님께 불편을 드려 대단히 죄송합니다.

플래시(Flash)를 지원하지 않아 차트 정보를 이용하실 수 없습니다.

플래시를 지원하는 PC 또는 모바일 기기를 이용해 주세요.

Figure 6.1: 네이버금융 개별종목 차트 탭



Network

Filter

Hide data URLs

XHR JS CSS Img Media Font Doc WS Manifest Other

Name

sise.nhn?symbol=005930&timeframe=day&count=500...

schedule.nhn?symbol=005930&startTime=20170522&e...

notice.nhn?symbol=005930&startTime=20170522&end...

Headers

Preview

Response

Timing

General

Request URL: https://fchart.stock.naver.com/sise.nhn?symbol=005930&timeframe=day&count=500&requestType=0

Request Method: GET

Status Code: 200

Remote Address: 175.158.5.165:443

Referrer Policy: unsafe-url

Response Headers (7)

Request Headers (2)

Query String Parameters

view source

view URL encoded

symbol: 005930

timeframe: day

count: 500

requestType: 0

Figure 6.2: 네이버금융 차트의 통신기록



This XML file does not appear to have any style information associated with it. The

```
▼ <protocol>
  ▼ <chartdata symbol="005930" name="삼성전자" count="500" timeframe="day" precision="0" origin
    <item data="20170522|45040|45380|44760|45100|352871"/>
    <item data="20170523|45400|45580|44900|44920|252141"/>
    <item data="20170524|44860|45300|44800|44880|173508"/>
    <item data="20170525|45160|45680|44800|45680|260896"/>
    <item data="20170526|45600|46460|45540|46080|272273"/>
    <item data="20170529|46220|46400|45380|45620|174791"/>
    <item data="20170530|45520|45660|44480|44640|248672"/>
    <item data="20170531|44580|45020|44400|44700|373382"/>
    <item data="20170601|44860|44900|44400|44680|195070"/>
    <item data="20170602|45060|45960|45000|45960|249775"/>
    <item data="20170605|46040|46360|45720|45940|151988"/>
    <item data="20170607|46500|46500|45240|45300|274588"/>
    <item data="20170608|45000|45580|45000|45160|279575"/>
    <item data="20170609|45680|46440|45600|46100|234657"/>
    <item data="20170612|45420|45600|45140|45380|219086"/>
    <item data="20170613|45140|45620|45140|45400|172498"/>
    <item data="20170614|45800|46060|45240|45360|203334"/>
    <item data="20170615|45680|45920|45180|45680|193140"/>
```

Figure 6.3: 주가 데이터 페이지

```
## [1] 5930
```

```
KOR_ticker$'종목코드' = str_pad(KOR_ticker$'종목코드', 6, side = c('left'), pad = '0')
```

먼저 저장해두었던 csv 파일을 불러오도록 합니다. 종목코드를 살펴보면 005930 이어야 할 삼성전자의 티커가 5930으로 입력되어 있습니다. 이는 파일을 불러오는 과정에서 0으로 시작하는 숫자들이 지워져버렸기 때문입니다. 이를 6자리로 다시 맞춰주기 위해, `stringr` 패키지의 `str_pad()` 함수를 사용해 줍니다. 6자리가 되지 않는 문자는, 왼쪽에 0을 추가하여 강제로 6자리로 만들어 주도록 합니다.

다음은 첫번째 종목인 삼성전자의 주가를 크롤링한 후 가공하는 방법입니다.

```
library(xts)

ifelse(dir.exists('data/KOR_price'), FALSE, dir.create('data/KOR_price'))
```

```
## [1] FALSE
```

```
i = 1
name = KOR_ticker$'종목코드'[i]

price = xts(NA, order.by = Sys.Date())
print(price)
```

```
##           [,1]
## 2019-06-13    NA
```

1. 먼저 data 폴더 내에 KOR\_price 폴더를 생성해줍니다.

2. `i = 1` 을 입력해 줍니다. 향후 for loop 구문을 통해 `i` 값만 변경하면 모든 종목의 주가를 다운로드 받을 수 있습니다.
3. `name`에 해당 티커를 입력해줍니다.
4. `xts()` 함수를 이용해 빈 시계열 데이터를 생성해주며, 인덱스는 `Sys.Date()`를 통해 현재 날짜를 입력합니다.

```
library(httr)
library(rvest)

url = paste0('https://fchart.stock.naver.com/sise.nhn?symbol=',
             name, '&timeframe=day&count=500&requestType=0')
data = GET(url)
data_html = read_html(data, encoding = 'EUC-KR') %>%
  html_nodes('item') %>%
  html_attr('data')

print(head(data_html))
```

```
## [1] "20170526|45600|46460|45540|46080|272273"
## [2] "20170529|46220|46400|45380|45620|174791"
## [3] "20170530|45520|45660|44480|44640|248672"
## [4] "20170531|44580|45020|44400|44700|373382"
## [5] "20170601|44860|44900|44400|44680|195070"
## [6] "20170602|45060|45960|45000|45960|249775"
```

1. `paste0()` 함수를 이용해 원하는 종목의 url을 생성해 줍니다. url중 티커에 해당하는 6자리 부분만 위에서 입력한 `name`으로 설정해주면 됩니다.
2. `GET()` 함수를 통해 페이지의 데이터를 불러옵니다.
3. `read_html()` 함수를 통해 html 정보를 읽어옵니다.
4. `html_nodes()`와 `html_attr()` 함수를 통해 item 태그 및 data 속성의 데이터를 추출합니다.

결과적으로 날짜 및 주가, 거래량 데이터만 추출되어 집니다. 해당 데이터는 '|'으로 구분되어 있으며, 이를 테이블 형태로 바꿀 필요가 있습니다.

```
library(readr)

price = read_delim(data_html, delim = '|')
print(head(price))
```

```
## # A tibble: 6 x 6
##   `20170526` `45600` `46460` `45540` `46080` `272273`
##   <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1  20170529  46220    46400    45380    45620    174791
## 2  20170530  45520    45660    44480    44640    248672
## 3  20170531  44580    45020    44400    44700    373382
## 4  20170601  44860    44900    44400    44680    195070
## 5  20170602  45060    45960    45000    45960    249775
## 6  20170605  46040    46360    45720    45940    151988
```

`readr` 패키지의 `read_delim()` 함수를 쓸 경우 구분자로 이루어진 데이터를 테이블로 쉽게 변경할 수 있습니다. 데이터를 확인해보면 테이블 형태로 변경되었으며 각 열은 날짜, 시가, 고가, 저가, 종가, 거래량을 의미합니다. 이 중 우리가 필요한 날짜와 종가를 선택한 후, 데이터 클렌징을 해주도록 합니다.

```
library(lubridate)

price = price[c(1, 5)]
price = data.frame(price)
colnames(price) = c('Date', 'Price')
price[, 1] = ymd(price[, 1])

rownames(price) = price[, 1]
price[, 1] = NULL

print(tail(price))
```

```
##           Price
## 2019-06-05 43900
## 2019-06-07 44200
## 2019-06-10 44800
## 2019-06-11 44850
## 2019-06-12 44600
## 2019-06-13 43700
```

1. 날짜에 해당하는 첫번째 열과, 종가에 해당하는 다섯번째 열만을 선택해 저장해 줍니다.
2. 티블 형태의 데이터를 데이터프레임 형태로 변경해 줍니다.
3. 열이름을 Date와 Price로 변경해 줍니다.
4. lubridate 패키지의 ymd() 함수를 이용하여 yyyy-mm-dd 형태를 yyyy-mm-dd로 변경해주며, 데이터 형태 또한 Date 타입으로 변경됩니다.
5. 행이름을 첫번째 열인 날짜로 변경해준 후, 해당 열은 NULL을 통해 삭제해 줍니다.

데이터를 확인해보면 우리에게 필요한 형태로 정리가 되었습니다. 마지막으로 해당 데이터를 csv 파일로 저장해두도록 합니다.

```
write.csv(price, paste0('data/KOR_price/', name, '_price.csv'))
```

data 폴더의 KOR\_price 폴더 내에 티커\_price.csv 이름으로 저장해두도록 합니다.

### 6.1.2 전 종목 주가 크롤링

위의 코드에서 for loop 구문을 이용하여 i 값만 변경해주면 모든 종목의 주가를 다운로드 받을 수 있습니다. 전 종목 주가를 다운로드 받는 전체 코드는 다음과 같습니다.

```
library(httr)
library(rvest)
library(stringr)
library(xts)
library(lubridate)

KOR_ticker = read.csv('data/KOR_ticker.csv', row.names = 1)
print(KOR_ticker$'종목코드'[1])
KOR_ticker$'종목코드' = str_pad(KOR_ticker$'종목코드', 6, side = c('left'), pad = '0')

ifelse(dir.exists('data/KOR_price'), FALSE, dir.create('data/KOR_price'))
```

```

for(i in 1 : nrow(KOR_ticker) ) {

  price = xts(NA, order.by = Sys.Date()) # 빈 시계열 데이터 생성
  name = KOR_ticker$종목코드[i] # 티커 부분 선택

  # 오류 발생 시 이를 무시하고 다음 루프로 진행
  tryCatch({
    # url 생성
    url = paste0('https://fchart.stock.naver.com/sise.nhn?symbol='
                  ,name,'&timeframe=day&count=500&requestType=0')

    # 이 후 과정은 위와 동일함
    # 데이터 다운로드
    data = GET(url)
    data_html = read_html(data, encoding = 'EUC-KR') %>%
      html_nodes("item") %>%
      html_attr("data")

    # 데이터 나누기
    price = read_delim(data_html, delim = '|')

    # 필요한 열만 선택 후 클렌징
    price = price[c(1, 5)]
    price = data.frame(price)
    colnames(price) = c('Date', 'Price')
    price[, 1] = ymd(price[, 1])

    rownames(price) = price[, 1]
    price[, 1] = NULL

  }, error = function(e) {

    # 오류 발생시 해당 종목명을 출력하고 다음 루프로 이동
    warning(paste0("Error in Ticker: ", name))
  })

  # 다운로드 받은 파일을 생성한 폴더 내 csv 파일로 저장
  write.csv(price, paste0('data/KOR_price/', name, '_price.csv'))

  # 타임슬립 적용
  Sys.sleep(2)
}

```

위의 코드에서 추가된 점은 다음과 같습니다. 페이지 오류, 통신 오류 등 오류가 발생할 경우 for loop 구문은 멈추어 버리며, 전체 데이터를 처음부터 다시 받는 일은 매우 귀찮은 작업입니다. 따라서 tryCatch() 함수를 이용해 오류가 발생 시 해당 티커를 출력한 후 다음 loop로 넘어가게 합니다.

또한 오류가 발생했을 시에는 xts() 함수를 통해 만들어 둔 빈 데이터를 저장하게 됩니다.

위의 코드가 모두 돌아가는 데는 수시간이 걸리며, 작업이 끝난 후 data/KOR\_price 폴더를 확인해보면, 전 종목 주가가 csv 형태로 저장되어 있게 됩니다.

## 6.2 재무제표 및 가치지표 크롤링

주가와 더불어 재무제표와 가치지표 역시 투자에 있어 핵심이 되는 데이터입니다. 해당 데이터 역시 구할 수 있는 여러 사이트가 있지만, 국내의 데이터 제공업체인 FnGuide에서 운영하는 Company Guide 홈페이지<sup>2</sup>에서 손쉽게 구할 수 있습니다.

### 6.2.1 재무제표 다운로드

먼저 개별종목의 재무제표를 탭을 선택하면 **포괄손익계산서**, **재무상태표**, **현금흐름표** 항목이 보이게 되며, 티커에 해당하는 A005930 뒤의 주소는 불필요한 내용이므로, 이를 제거한 주소로 접속하도록 합니다. A 뒤의 6자리 티커만 변경할 경우, 해당 종목의 재무제표 페이지로 이동하게 됩니다.

`http://comp.fnguide.com/SV02/ASP/SVD_Finance.asp?pGB=1&gicode=A005930`

우리가 원하는 재무제표 항목들은 모두 테이블 형태로 제공되고 있으므로, `html_table()` 함수를 이용하여 손쉽게 추출할 수 있습니다.

```
library(httr)
library(rvest)

ifelse(dir.exists('data/KOR_fs'), FALSE,
       dir.create('data/KOR_fs'))

Sys.setlocale("LC_ALL", "English")

url = 'http://comp.fnguide.com/SV02/ASP/SVD_Finance.asp?pGB=1&gicode=A005930'
data = GET(url)
data = data %>%
  read_html() %>%
  html_table()

Sys.setlocale("LC_ALL", "Korean")

lapply(data, function(x) {head(x, 3)})
```

1. 먼저 data 폴더 내에 KOR\_fs 폴더를 생성해줍니다.
2. `Sys.setlocale()` 함수를 통해 로케일 언어를 영어로 설정 해줍니다.
3. url을 입력한 후, `GET()` 함수를 통해 페이지 내용을 받아옵니다.
4. `read_html()` 함수를 통해 html 내용을 읽어오며, `html_table()` 함수를 통해 테이블 내용만을 추출합니다.
5. 로케일 언어를 다시 한글로 설정 해줍니다.

위의 과정을 거치면 data 변수에는 총 리스트 형태로 총 6개의 테이블이 들어오게 되며, 그 내용은 다음과 같습니다.

이 중 연간 기준 재무제표에 해당하는 첫번째, 세번째, 다섯번째 테이블을 선택한 후, 클랜징 작업을 해주도록 하겠습니다.

```
data_IS = data[[1]]
data_BS = data[[3]]
data_CF = data[[5]]

print(names(data_IS))
```

<sup>2</sup><http://comp.fnguide.com/>

Table 6.1: 재무제표 테이블 내역

순서	내용
1	포괄손익계산서 (연간)
2	포괄손익계산서 (분기)
3	재무상태표 (연간)
4	재무상태표 (분기)
5	현금흐름표 (연간)
6	현금흐름표 (분기)

```
## [1] "IFRS(연결)" "2016/12" "2017/12" "2018/12" "2019/03"
## [6] "전년동기" "전년동기(%)"
```

```
data_IS = data_IS[, 1:(ncol(data_IS)-2)]
```

포괄손익계산서 테이블(data\_IS)에는 전년동기, 전년동기(%) 열이 존재하며, 통일성을 위해 이를 삭제해주었습니다. 이제 테이블을 묶은 후 클랜징을 해주도록 하겠습니다.

```
data_fs = rbind(data_IS, data_BS, data_CF)
data_fs[, 1] = gsub('계산에 참여한 계정 펼치기',
                  '', data_fs[, 1])
data_fs = data_fs[!duplicated(data_fs[, 1]), ]

rownames(data_fs) = NULL
rownames(data_fs) = data_fs[, 1]
data_fs[, 1] = NULL

data_fs = data_fs[, substr(colnames(data_fs), 6,7) == "12"]
```

1. 먼저 `rbind()` 함수를 이용하여 세 테이블을 행으로 묶은 뒤 저장합니다.
2. 첫번째 열인 계정명에는 ‘계산에 참여한 계정 펼치기’ 라는 글자가 들어간 항목이 존재합니다. 이는 페이지 내에서 펼치기 역할을 하는 (+) 항목에 해당하며, `gsub()` 함수를 이용해 해당 글자를 삭제해 줍니다.
3. 중복되는 계정명이 다수 존재하며, 이는 대부분 불필요한 항목입니다. `!duplicated()` 함수를 사용해 중복되지 않는 계정명만을 선택해 줍니다.
4. 행이름을 초기화 한 후, 첫번째 열의 계정명을 행이름으로 변경합니다. 그 후 첫번째 열은 삭제해주도록 합니다.
5. 간혹 12월 결산법인이 아닌 중목, 혹은 연간 재무제표임에도 불구하고 분기 재무제표가 들어와 있는 경우가 있습니다. 비교의 통일성을 위해 `substr()` 함수를 이용하여 끝 글자가 12인 열, 즉 12월 결산 데이터만을 선택해 줍니다.

```
print(head(data_fs))
```

```
##           2016/12  2017/12  2018/12
## 매출액      2,018,667 2,395,754 2,437,714
## 매출원가    1,202,777 1,292,907 1,323,944
## 매출총이익      815,890 1,102,847 1,113,770
## 판매비와관리비    523,484   566,397   524,903
## 인건비        59,763    67,972    64,514
## 유무형자산상각비   10,018    13,366    14,477
```

```
str(data_fs)
```

```
## 'data.frame':   236 obs. of  3 variables:
## $ 2016/12: chr  "2,018,667" "1,202,777" "815,890" "523,484" ...
## $ 2017/12: chr  "2,395,754" "1,292,907" "1,102,847" "566,397" ...
## $ 2018/12: chr  "2,437,714" "1,323,944" "1,113,770" "524,903" ...
```

데이터를 확인해보면 연간 기준 재무제표가 정리되었습니다. 그러나 데이터에 콤마가 문자형이므로, 이를 숫자형으로 변경해주어야 합니다.

```
library(readr)
library(stringr)

data_fs = sapply(data_fs, function(x) {
  str_replace_all(x, ',', '') %>%
  as.numeric()
}) %>%
  data.frame(., row.names = rownames(data_fs))

print(head(data_fs))
```

```
##                X2016.12 X2017.12 X2018.12
## 매출액          2018667  2395754  2437714
## 매출원가        1202777  1292907  1323944
## 매출총이익       815890  1102847  1113770
## 판매비와관리비   523484   566397   524903
## 인건비           59763    67972    64514
## 유무형자산상각비 10018    13366    14477
```

```
str(data_fs)
```

```
## 'data.frame':   236 obs. of  3 variables:
## $ X2016.12: num  2018667 1202777 815890 523484 59763 ...
## $ X2017.12: num  2395754 1292907 1102847 566397 67972 ...
## $ X2018.12: num  2437714 1323944 1113770 524903 64514 ...
```

1. `sapply()` 함수를 이용해 각 열에 `stringr` 패키지의 `str_replace_allr()` 함수를 적용하여 콤마(,)를 제거한 후, `as.numeric()` 함수를 통해 숫자형 데이터로 변경합니다.
2. `data.frame()` 함수를 이용해 데이터프레임 형태로 만들어주며, 행이름은 기존 내용을 그대로 유지해줍니다.

정리된 데이터를 출력해보면 문자형이던 데이터가 숫자형으로 변경되었습니다.

```
write.csv(data_fs, 'data/KOR_fs/005930_fs.csv')
```

data 폴더의 KOR\_fs 폴더 내에 티커\_fs.csv 이름으로 저장해주도록 합니다.

## 6.2.2 가치지표 계산하기

위에서 구한 재무제표 데이터를 이용해 가치지표를 계산할 수 있습니다. 흔히 사용되는 가치지표는 **PER**, **PBR**, **PCR**, **PSR**이며 분자는 주가, 분모는 재무제표 데이터가 사용됩니다.

위에서 구한 재무제표 항목에서 분모 부분에 해당하는 데이터만 선택하도록 하겠습니다.

Table 6.2: 가치지표의 종류

순서	분모
PER	Earnings (순이익)
PBR	Book Value (순자산)
PCR	Cashflow (영업활동현금흐름)
PSR	Sales (매출액)

```
ifelse(dir.exists('data/KOR_value'), FALSE,
      dir.create('data/KOR_value'))
```

```
## [1] FALSE
```

```
value_type = c('지배주주순이익',
               '자본',
               '영업활동으로인한현금흐름',
               '매출액')

value_index = data_fs[match(value_type, rownames(data_fs)),
                      ncol(data_fs)]
print(value_index)
```

```
## [1] 438909 2477532 670319 2437714
```

1. 먼저 data 폴더 내에 KOR\_value 폴더를 생성해줍니다.
2. 분모에 해당하는 항목을 저장한 후, match() 함수를 이용하여 해당 항목이 위치하는 지점을 찾아 데이터를 선택해줍니다.ncol() 함수를 이용하여 가장 우측, 즉 최근년도 재무제표 데이터를 선택해줍니다.

다음으로 분자 부분에 해당하는 현재 주가를 수집해야 합니다. 이 역시 Company Guide 접속화면에서 구할 수 있으며, 불필요한 부분을 제거한 url은 다음과 같습니다.

```
http://comp.fnguide.com/SV02/ASP/SVD_main.asp?pGB=1&giccode=A005930
```

위의 주소 역시 A 뒤의 6자리 티커만 변경할 경우, 해당 종목의 스냅샷 페이지로 이동하게 됩니다.

주가추이 부분에 우리가 원하는 현재 주가가 있으므로, 해당 데이터를 크롤링 하면 됩니다. 이처럼 크롤링하고자 하는 데이터가 하나 혹은 소수 일때는 html 구조를 모두 분해한 후 데이터를 추출하는 것 보다 Xpath를 이용하는 것이 훨씬 효율적입니다.

Xpath란 XML 중 특정 값의 태그나 속성을 찾기 쉽게 만든 주소라 생각하면 됩니다. 예를 들어 R 프로그램이 저장된 곳을 윈도우 탐색기를 이용하여 이용할 경우 C:\Program Files\R\R-3.4.1 형태의 주소를 보이며, 이는 윈도우의 path 문법입니다. XML 역시 이와 동일한 개념의 XPath가 존재합니다. 웹페이지에서 Xpath를 찾는 법은 다음과 같습니다.

먼저 크롤링하고자 하는 내용에 마우스를 올린 채 우클릭 → 검사를 누르면, 개발자도구 화면이 열리며 해당 지점의 html 부분이 선택됩니다. 그 후 html 화면에서 우클릭 → Copy → Copy Xpath를 선택하면, 해당 지점의 xpath가 복사됩니다.

```
//*[@id="svdMainChartTxt11"]
```

위에서 구한 xpath를 이용하여 주가 데이터를 크롤링하도록 하겠습니다.



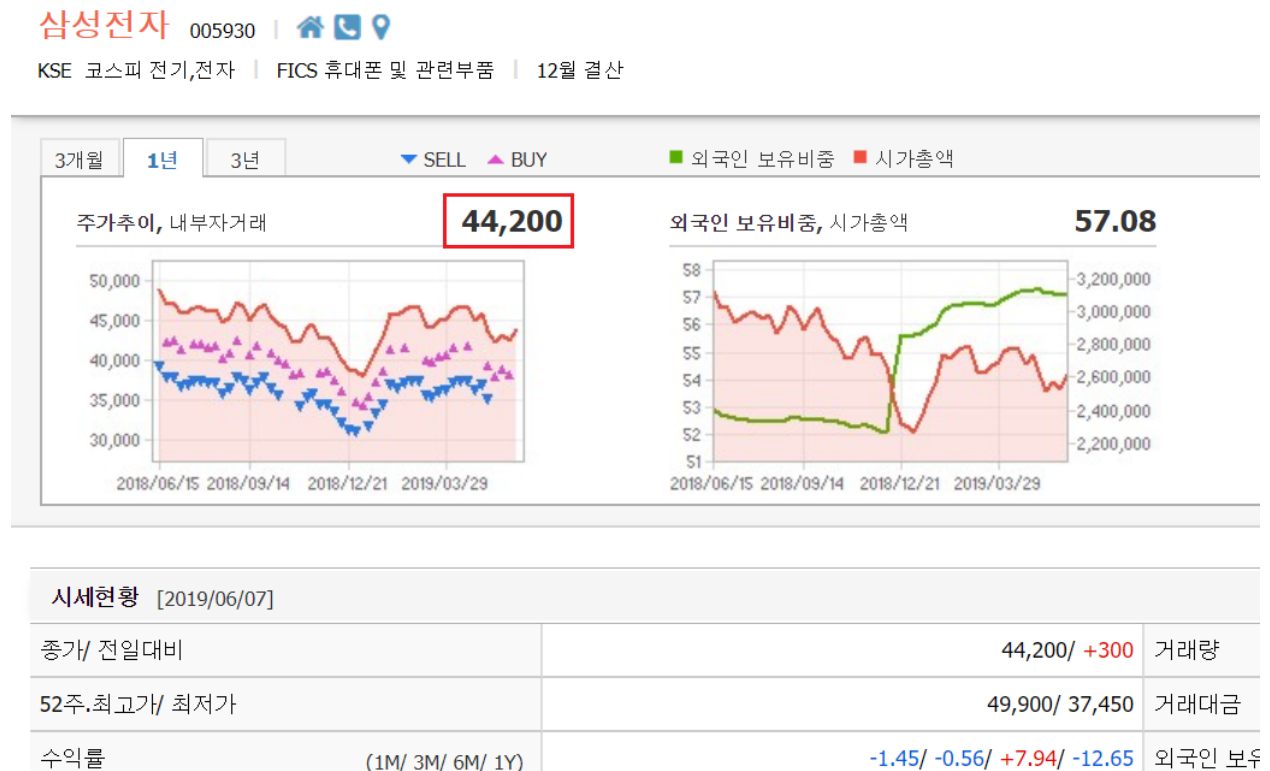


Figure 6.4: Company Guide 스냅샷 화면

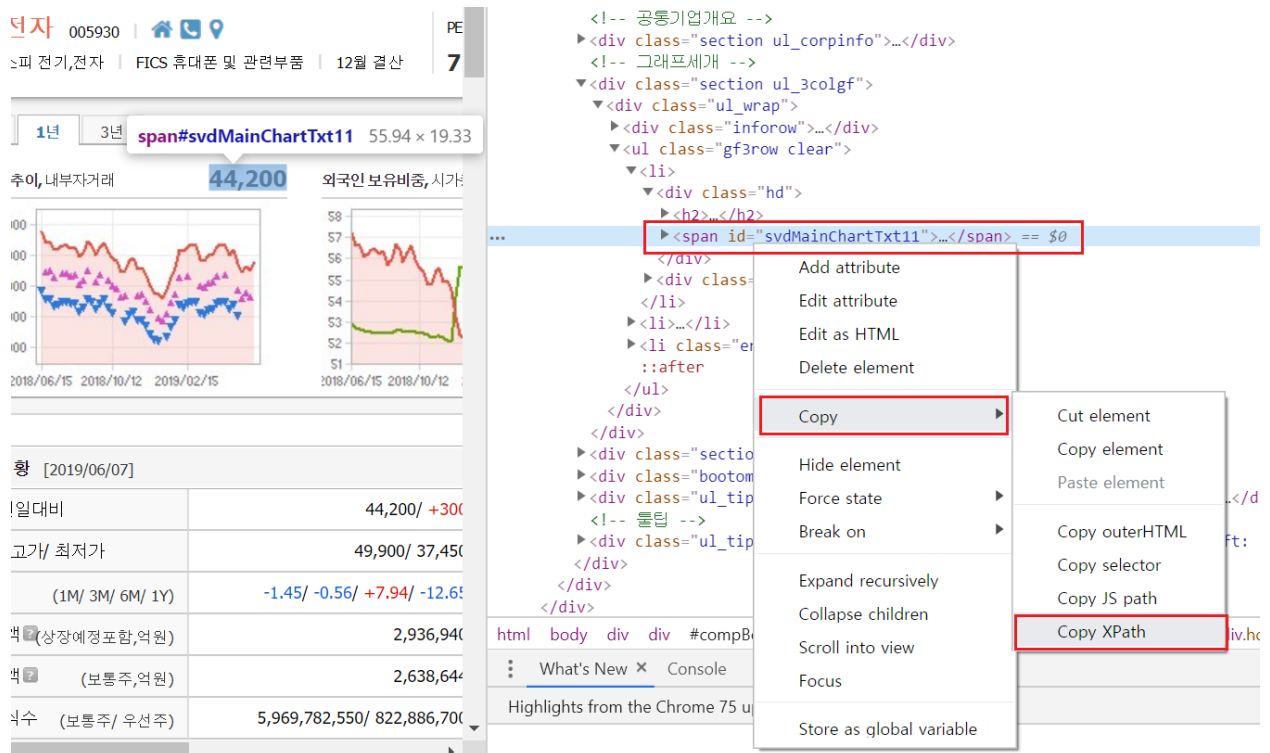


Figure 6.5: Company Guide 스냅샷 화면

```
url = 'http://comp.fnguide.com/SV02/ASP/SVD_main.asp?pGB=1&gicode=A005930'
data = GET(url)

price = read_html(data) %>%
  html_node(xpath = '//*[@id="svdMainChartTxt11"]') %>%
  html_text() %>%
  parse_number()

print(price)
```

```
## [1] 44600
```

1. 먼저 url을 입력한 후, GET() 함수를 이용하여 데이터를 불러옵니다.
2. read\_html() 함수를 이용해 html 데이터를 불러온 후, html\_node() 함수 내에 위에서 구한 xpath를 입력하여, 해당 지점의 데이터를 추출합니다.
3. html\_text() 함수를 통해 텍스트 데이터만을 추출하며, parse\_number() 함수를 적용합니다. 해당 함수는 문자형 데이터에서 콤마와 같은 불필요한 문자를 제거한 후, 숫자형 데이터로 변경해줍니다.

이처럼 xpath를 이용할 경우 태그나 속성을 분해하지 않고도 원하는 지점의 데이터를 크롤링할 수 있습니다. 가치지표를 계산하기 위해서는 발행주식수 역시 필요합니다. 예를 들어 PER를 계산하는 방법은 다음과 같습니다.

$$PER = Price/EPS = \text{주가/주당순이익}$$

주당순이익의 경우 순이익을 전체 주식수로 나눈값이므로, 해당 값의 계산을 위해 전체 주식수 역시 크롤링해야 합니다. 해당 데이터 역시 웹페이지에 존재하므로 위에서 주가를 크롤링한 방법과 동일하게 구할 수 있으며, xpath는 다음과 같습니다.

```
//*[@id="svdMainGrid1"]/table/tbody/tr[7]/td[1]
```

이를 이용해 발행주식수 중 보통주를 선택하는 방법은 다음과 같습니다.

```
share = read_html(data) %>%
  html_node(xpath = '//*[@id="svdMainGrid1"]/table/tbody/tr[7]/td[1]') %>%
  html_text()

print(share)
```

```
## [1] "5,969,782,550/ 822,886,700"
```

read\_html() 함수와 html\_node() 함수를 이용해, html 내에서 xpath에 해당하는 데이터를 추출합니다. 그 후, html\_text() 함수를 통해 텍스트 부분만 추출하도록 합니다. 해당 과정을 거치면 보통주/우선주의 형태로 발행주식주가 저장되어 있습니다. 이 중 우리가 원하는 데이터는 '/' 앞에 위치한 보통주 발행주식수입니다.

```
share = share %>%
  strsplit('/') %>%
  unlist() %>%
  .[1] %>%
  parse_number()

print(share)
```

```
## [1] 5969782550
```

1. `strsplit()` 함수를 통해 '/'를 기준으로 데이터를 나누어주며, 해당 결과는 리스트 형태로 저장됩니다.
2. `unlist()` 함수를 통해 리스트를 벡터 형태로 변환합니다.
3. `.[1]`을 통해 보통주 발행주식수인 첫번째 데이터를 선택합니다.
4. `parse_number()` 함수를 통해 문자형 데이터를 숫자형으로 변환해 줍니다.

재무데이터, 현재 주가, 발행주식수를 이용하여 가치지표를 계산해보도록 하겠습니다.

```
data_value = price / (value_index * 100000000 / share)
names(data_value) = c('PER', 'PBR', 'PCR', 'PSR')
data_value[data_value < 0] = NA

print(data_value)
```

```
##          PER          PBR          PCR          PSR
## 6.066230 1.074667 3.972024 1.092221
```

분자에는 현재 주가를 입력하며, 분모에는 재무 데이터를 보통주 발행주식수로 나눈 값을 입력합니다. 단, 주가는 원 단위, 재무 데이터는 억 단위이므로, 둘 간의 단위를 동일하게 맞춰주기 위해 분모에 억을 곱해 줍니다. 또한 가치지표가 음수인 경우는 NA로 변경해두도록 합니다.

결과를 확인해보면 4가지 가치지표가 잘 계산되었습니다.<sup>3</sup>

```
write.csv(data_value, 'data/KOR_value/005930_value.csv')
```

data 폴더의 KOR\_value 폴더 내에 티커\_value.csv 이름으로 저장해두도록 합니다.

### 6.2.3 전 종목 재무제표 및 가치지표 다운로드

위의 코드에서 for loop 구문을 이용하여 url 중 6자리 티커에 해당하는 값만 변경해주면 모든 종목의 재무제표를 다운로드 받고, 이를 바탕으로 가치지표를 계산할 수 있습니다. 해당 코드는 다음과 같습니다.

```
library(stringr)
library(httr)
library(rvest)
library(stringr)
library(readr)

KOR_ticker = read.csv('data/KOR_ticker.csv', row.names = 1)
KOR_ticker$'종목코드' = str_pad(KOR_ticker$'종목코드', 6, side = c('left'), pad = '0')

ifelse(dir.exists('data/KOR_fs'), FALSE, dir.create('data/KOR_fs'))
ifelse(dir.exists('data/KOR_value'), FALSE, dir.create('data/KOR_value'))

for(i in 1 : nrow(KOR_ticker) ) {

  data_fs = c()
  data_value = c()
```

<sup>3</sup> 분모에 사용되는 재무데이터의 구체적인 항목과 발행주식수를 계산하는 방법의 차이로 인해 여러 업체에서 제공하는 가치지표와 다소 차이가 발생할 수 있습니다.

```

name = KOR_ticker$'종목코드'[i]

# 오류 발생 시 이를 무시하고 다음 루프로 진행
tryCatch({

  # url 생성
  url = paste0("http://comp.fnguide.com/SV02/ASP/SVD_Finance.asp?pGB=1&gicode=A",
               name)

  # 이 후 과정은 위와 동일함
  # 데이터 다운로드
  Sys.setlocale("LC_ALL", "English")

  # 테이블 추출
  data = GET(url) %>%
    read_html() %>%
    html_table()

  Sys.setlocale("LC_ALL", "Korean")

  # 3개 재무제표를 하나로 합치기
  data_IS = data[[1]]
  data_BS = data[[3]]
  data_CF = data[[5]]

  data_IS = data_IS[, 1:(ncol(data_IS)-2)]
  data_fs = rbind(data_IS, data_BS, data_CF)

  # 데이터 클렌징
  data_fs[, 1] = gsub('계산에 참여한 계정 펼치기',
                    '', data_fs[, 1])
  data_fs = data_fs[!duplicated(data_fs[, 1]), ]

  rownames(data_fs) = NULL
  rownames(data_fs) = data_fs[, 1]
  data_fs[, 1] = NULL

  # 12월 재무제표만 선택
  data_fs = data_fs[, substr(colnames(data_fs), 6,7) == "12"]

  data_fs = sapply(data_fs, function(x) {
    str_replace_all(x, ',', '') %>%
    as.numeric()
  }) %>%
  data.frame(., row.names = rownames(data_fs))

  # 가치지표 분모부분
  value_type = c('지배주주순이익',
                 '자본',
                 '영업활동으로인한현금흐름',
                 '매출액')

```

```

# 해당 재무데이터만 선택
value_index = data_fs[match(value_type, rownames(data_fs)),
                        ncol(data_fs)]

# Snapshot 페이지 불러오기
url = paste0("http://comp.fnguide.com/SV02/ASP/SVD_Main.asp?pGB=1&gicode=A",
             name)
data = GET(url)

# 현재 주가 크롤링
price = read_html(data) %>%
  html_node(xpath = '//*[@id="svdMainChartTxt11"]') %>%
  html_text() %>%
  parse_number()

# 보통주 발행장주식수 크롤링
share = read_html(data) %>%
  html_node(xpath = '//*[@id="svdMainGrid1"]/table/tbody/tr[7]/td[1]') %>%
  html_text() %>%
  strsplit('/') %>%
  unlist() %>%
  .[1] %>%
  parse_number()

# 가치지표 계산
data_value = price / (value_index * 100000000 / share)
names(data_value) = c('PER', 'PBR', 'PCR', 'PSR')
data_value[data_value < 0] = NA

}, error = function(e) {

  # 오류 발생시 해당 종목명을 출력하고 다음 루프로 이동
  data_fs <- NA
  data_value <- NA
  warning(paste0("Error in Ticker: ", name))
})

# 다운로드 받은 파일을 생성한 각각의 폴더 내 csv 파일로 저장

# 재무제표 저장
write.csv(data_fs, paste0('data/KOR_fs/', name, '_fs.csv'))

# 가치지표 저장
write.csv(data_value, paste0('data/KOR_value/', name, '_value.csv'))

# 2초간 타임슬립 적용
Sys.sleep(2)
}

```

전종목 주가 데이터를 받는 과정과 동일하게, KOR\_ticker.csv 파일을 불러온 후 for loop을 통해 i값이 변함에 따라 티커를 변경해가며 모든 종목의 재무제표 및 가치지표를 다운로드 받습니다. tryCatch() 함수를 이용해 오류가 발생 시 NA로 이루어진 빈 데이터를 저장한 후, 다음 루프로 넘어가게 됩니다.

data/KOR\_fs 폴더에는 전 종목의 재무제표 데이터가, data/KOR\_value 폴더에는 전 종목의 가치지표 데이터가

YAHOO! FINANCE

005930.KS

Finance Home Watchlists My Portfolio Screeners Markets Industries Videos News Personal Finance Tech

As of 10:56AM KST. Market open.

Summary Chart Conversations Statistics Historical Data Profile **Financials** Analysis Options Holders Sustainability

Show: **Income Statement** | Balance Sheet | Cash Flow

Annual | Quarterly

**Income Statement** All numbers in thousands

	12/31/2018	12/31/2017	12/31/2016	12/31/2015
<b>Revenue</b>				
Total Revenue	243,771,415,000	239,575,376,000	201,866,745,000	200,653,482,000
Cost of Revenue	132,394,411,000	129,290,661,000	120,277,715,000	123,482,118,000
<b>Gross Profit</b>	<b>111,377,004,000</b>	<b>110,284,715,000</b>	<b>81,589,030,000</b>	<b>77,171,364,000</b>

Figure 6.6: 야후 파이낸스 재무제표

csv 형태로 저장되어 있게 됩니다.

## 6.3 야후 파이낸스 데이터 구하기

크롤링을 이용하여 데이터를 수집할 경우 주의해야 할 점은, 웹페이지 구조가 변경되는 경우입니다. 웹페이지의 형태가 변경될 경우 이에 맞춰 코드를 다시 짜야합니다. 그러나 최악의 경우는 웹사이트가 폐쇄되는 경우입니다. 실제로 투자자들이 많이 사용하던 구글 파이낸스가 2018년 서비스를 중단함에 따라 이를 이용하던 사이트 및 패키지에서 오류가 발생하기도 했습니다.

이러한 상황에 대비하여 데이터를 구할수 있는 예비 사이트를 알아두어 테스트 코드를 작성해둘 필요가 있으며, 이를 위해 야후 파이낸스에서 데이터 구하는 법을 살펴보도록 하겠습니다. 주가 데이터의 경우 `getSymbols()` 함수를 통해 야후 파이낸스에서 제공하는 API를 사용하여 주가를 다운로드 받는 방법을 이미 살펴보았으며, 웹페이지에서 재무제표를 크롤링하는법 및 가치지표를 계산하는 법에 대해 알아보도록 하겠습니다.

### 6.3.1 재무제표 다운로드

먼저 야후 파이낸스에 접속하여 삼성전자 티커에 해당하는 005930.KS를 입력합니다. 그 후, 재무제표 데이터에 해당하는 Financials 항목을 선택합니다. 손익계산서 (Income Statement), 재무상태표 (Balance Sheet), 현금흐름표 (Cash Flow) 총 3개 지표가 있으며, 각각의 url은 다음과 같습니다.

- Income Statement: <https://finance.yahoo.com/quote/005930.KS/financials?p=005930.KS>
- Balance Sheet: <https://finance.yahoo.com/quote/005930.KS/balance-sheet?p=005930.KS>
- Cash Flow: <https://finance.yahoo.com/quote/005930.KS/cash-flow?p=005930.KS>

각 페이지에서 Xpath를 이용하여 재무제표에 해당하는 테이블 부분만을 선택하여 추출할 수 있으며, 3개 페이지의 해당 Xpath는 모두 아래와 같이 동일합니다.

```
//*[@id="Col1-1-Financials-Proxy"]/section/div[3]/table
```

위의 정보를 이용하여 재무제표를 다운로드 받는 과정은 다음과 같습니다.

```

library(httr)
library(rvest)

url_IS = 'https://finance.yahoo.com/quote/005930.KS/financials?p=005930.KS'
url_BS = 'https://finance.yahoo.com/quote/005930.KS/balance-sheet?p=005930.KS'
url_CF = 'https://finance.yahoo.com/quote/005930.KS/cash-flow?p=005930.KS'

yahoo_finance_xpath = '//*[@id="Col1-1-Financials-Proxy"]/section/div[3]/table'

data_IS = GET(url_IS) %>%
  read_html() %>%
  html_node(xpath = yahoo_finance_xpath) %>%
  html_table()

data_BS = GET(url_BS) %>%
  read_html() %>%
  html_node(xpath = yahoo_finance_xpath) %>%
  html_table()

data_CF = GET(url_CF) %>%
  read_html() %>%
  html_node(xpath = yahoo_finance_xpath) %>%
  html_table()

data_fs = rbind(data_IS, data_BS, data_CF)

print(head(data_fs))

```

```

##           X1           X2           X3
## 1      Revenue      12/31/2018      12/31/2017
## 2    Total Revenue    243,771,415,000    239,575,376,000
## 3    Cost of Revenue    132,394,411,000    129,290,661,000
## 4      Gross Profit    111,377,004,000    110,284,715,000
## 5 Operating Expenses Operating Expenses Operating Expenses
## 6 Research Development    18,354,080,000    16,355,612,000
##           X4           X5
## 1      12/31/2016      12/31/2015
## 2    201,866,745,000    200,653,482,000
## 3    120,277,715,000    123,482,118,000
## 4     81,589,030,000     77,171,364,000
## 5 Operating Expenses Operating Expenses
## 6     14,111,381,000     13,705,695,000

```

1. 위에서 구한 url을 저장해줍니다.
2. GET() 함수를 통해 페이지 정보를 받아온 후, read\_html() 함수를 통해 html 정보를 받아옵니다.
3. html\_node() 함수 내에서 위에서 구한 xpath를 이용해 테이블 부분의 html을 선택한 후, html\_table()을 통해 테이블 형태만 추출합니다.
4. 3개 페이지에 위의 내용을 동일하게 적용한 후, rbind()를 이용해 행으로 묶어줍니다.

다운로드 받은 데이터를 클렌징처리 해주도록 하며, 그 과정은 앞선 예시와 거의 비슷합니다.

```
library(stringr)

data_fs = data_fs[!duplicated(data_fs[, 1]), ]
rownames(data_fs) = NULL
rownames(data_fs) = data_fs[, 1]

colnames(data_fs) = data_fs[1, ]
data_fs = data_fs[-1, ]

data_fs = data_fs[, substr(colnames(data_fs), 1,2) == "12"]

data_fs = sapply(data_fs, function(x) {
  str_replace_all(x, ',', '') %>%
  as.numeric()
}) %>%
data.frame(., row.names = rownames(data_fs))

print(head(data_fs))
```

```
##                X12.31.2018  X12.31.2017  X12.31.2016
## Total Revenue      243771415000 239575376000 201866745000
## Cost of Revenue    132394411000 129290661000 120277715000
## Gross Profit       111377004000 110284715000  81589030000
## Operating Expenses                NA                NA                NA
## Research Development    18354080000 16355612000 14111381000
## Selling General and Administrative 32688565000 38947445000 37235161000
##                X12.31.2015
## Total Revenue      200653482000
## Cost of Revenue    123482118000
## Gross Profit       77171364000
## Operating Expenses                NA
## Research Development    13705695000
## Selling General and Administrative 36081636000
```

1. `!duplicated()` 함수를 사용해 중복되지 않는 계정명만을 선택해 줍니다. 4. 행이름을 초기화 한 후, 첫번째 열의 계정명을 행이름으로 변경합니다. 그 후 첫번째 열은 삭제해주도록 합니다. 3. 열이름으로 첫번째 행으로 변경한 후, 해당 행은 삭제해주도록 합니다. 4. `substr()` 함수를 이용하여 처음 두 글자가 '12'인 열, 즉 12월 결산 데이터만을 선택해 줍니다. 5. `sapply()` 함수를 이용해 각 열에 `stringr` 패키지의 `str_replace_allr()` 함수를 적용하여 콤마(,)를 제거한 후, `as.numeric()` 함수를 통해 숫자형 데이터로 변경합니다. 6. `data.frame()` 함수를 이용해 데이터프레임 형태로 만들어주며, 행이름은 기존 내용을 그대로 유지해줍니다.

### 6.3.2 가치지표 계산하기

가치지표를 계산하는 과정도 앞선 예시와 거의 비슷합니다.

```
value_type =
  c('Net Income Applicable To Common Shares', # Earnings
    'Total Stockholder Equity', # Book Value
    'Total Cash Flow From Operating Activities', # Cash Flow
    'Total Revenue') # Sales

value_index = data_fs[match(value_type, rownames(data_fs)), 1]
```



먼저 분모에 해당하는 항목을 저장한 후, `match()` 함수를 이용하여 해당 항목이 위치하는 지점을 찾아 데이터를 선택해줍니다. 기존 예제와 다른점은, 야후 파이낸스의 경우 최근년도 데이터가 가장 좌측에 위치하므로, 첫번째 열을 선택해 줍니다.