

# 퀀트 투자 쿠북: R을 이용한 투자 포트폴리오 만들기

이현열

2019-07-07



---

## Contents

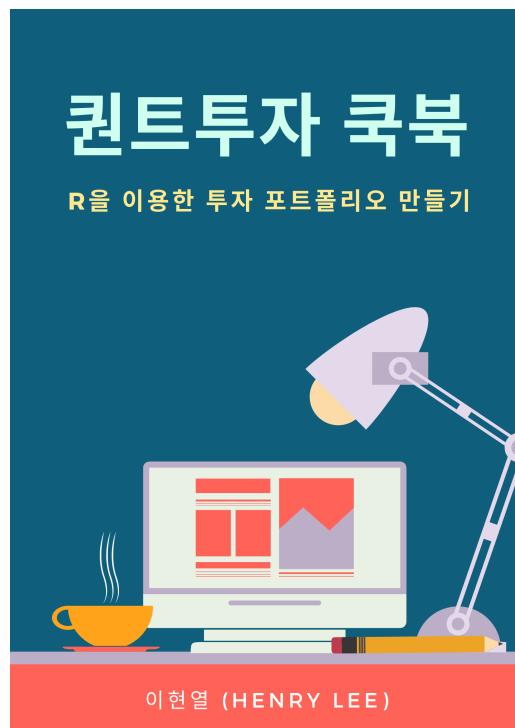
---



---

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.

PDF 편집이 완료되면 책으로 페블리쉬 할 예정입니다.

# 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 익숙한 단어가 되었습니다. 퀘트 투자에서 말하는 퀘트란 모형을 기반으로 금융상품의 가격을 산정하거나, 이를 바탕으로 투자를 하는 사람을 말합니다. 퀘트Quant라는 단어가 **계량적**을 의미하는 퀘티티티브Quantitative의 앞 글자를 따온 점을 생각하면 쉽게 이해가 될 것입니다.

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

이처럼 데이터의 수집과 가공, 이를 바탕으로 모델을 만든 후 실행하는 단계는 데이터 과학의 업무 흐름도와 매우 유사합니다. 해들리 위컴Hadley Wickham에 따르면<sup>1</sup>, 데이터 과학의 업무 과정은 그림 1.1과 같습니다.

데이터 과학자들은 프로그래밍을 통해 데이터를 불러온 후 이를 정리하고, 원하는 결과를 찾기 위해 데이터를 변형 혹은 시각화하고, 모델링 업무을 합니다. 그 후 이러한 결과를 바탕으로 타인과 소통하는 일련의 과정을 거칩니다.

퀀트 투자의 단계 역시 이와 매우 유사합니다. 투자에 필요한 주가, 재무제표 등의 데이터를 수집하여 정리한 후, 필요한 지표를 얻기 위해 가공을 합니다. 그 후 각종 모형을 이용해 투자 종목을 선택하거나 백테스트를 수행하며, 이를 바탕으로 실제 투자를 하고 성과를 평가합니다. 따라서 퀘트 투자는 데이터 과학이 금융에

---

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

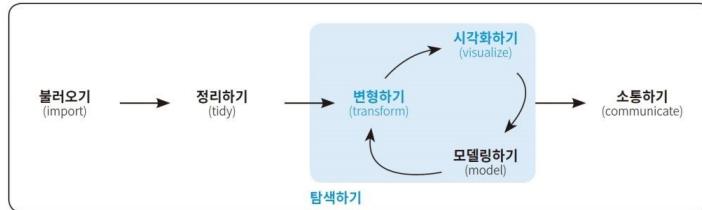


Figure 1.1: 데이터 과학 업무 과정

응용된 사례라고도 볼 수 있으며, 그 중심에는 역시 데이터와 프로그래밍이 있습니다.

본 책에서도 데이터 과학의 업단계와 동일하게 데이터를 불러오기, 각 데이터별로 정리하고 가공하기, 시각화를 통해 데이터의 특징 파악하기, 퀸트 모델을 이용하여 종목 선택하기, 백테스트를 실시한 후 성과 및 위험 평가하기에 대해 알아보겠습니다. 이에 앞서 본 장에서는 퀸트 투자의 심장이라고 할 수 있는 데이터를 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 제공업체들의 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 즉석에서 백테스트 및 성과 평가까지 가능합니다. Factset에서 판매하는 Alpha Testing 기능, 혹은 S&P Global에서 판매하는 ClariFI®(그림 1.2)를 사용한다면, 전세계 주식을 대상으로 원하는 전략의 백테스트 결과를 마우스 클릭과 몇 번의 동작만으로 수행할 수 있습니다.



Figure 1.2: ClariFI®의 백테스트 기능

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

해외데이터의 경우 Quandl<sup>2</sup>이나 tiingo<sup>3</sup>등의 업체가 제공하는 서비스를 통해 상대적으로 저렴한 가격으로 데이터를 구할 수 있습니다. 물론 대형 데이터 제공업체에 비해 데이터의 종류나 기간은 짧은 편이지만, 대부분의 일반 투자가 사용하기에는 충분한 데이터를 얻을 수 있습니다. tiingo에서는 전세계 64,386 개 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 소외되어 있습니다. 따라서 돈을 들이지 않고 국내 데이터를 얻기 위해서는 직접 발품을 파는 수밖에 없습니다. 야후 파이낸스<sup>4</sup> 혹은 국내 금융 사이트들에서 제공하는 정보를 크롤링하여 데이터를 수집할 수 있습니다.



Figure 1.3: NAVER 금융 제공 재무정보

이러한 정보를 잘만 활용한다면 장기간의 주가 및 재무정보를 무료로 수집할 수 있습니다. 물론 데이터 제공업체가 제공하는 깔끔한 형태의 데이터가 아니므로 클랜징 작업이 필요하다는 점, 그리고 상장폐지된 기업의 경우 데이터를 구하기 힘들다는 단점이 있습니다. 그러나 비용이 들지 않는다는 점, 그리고 현재 시점에서 투자 종목을 선택할 때는 상장폐지된 기업의 정보가 필요하지 않는다는 점을 고려하면, 이는 큰 문제가 되지 않습니다.

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

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

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

## 1.2 퀸트 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 클랜징 작업이라 합니다. 또한, 데이터가 정제된 이후 이를 활용한 투자 전략의 백테스트나 종목 선정을 위해서도 프로그래밍은 필수입니다. 물론 모든 퀸트 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 클랜징 작업을 하는 경우, 대상이 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 종목에 투자하는 모멘텀 전략의 백테스트를 있다고 가정합시다. 처음에는 엑셀을 통해 백테스트를 하는 것이 편하다고 생각할 수 있습니다. 그러나 만일 12개월이 아닌 6개월 누적 수익률로 백테스트를 하고자 한다면 어떨까요? 엑셀에서 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 하는 것만으로 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 저장을 할 것입니다. 그 후 B라는 전략을 새롭게 백테스트 하려면 해당 데이터를 새로운 엑셀 파일에 복사하여 작업한 후 다시 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 불과하므로, 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 퀸트 투자에 필요한 프로그래밍은 매우 한정적이고 반복적이기에, 몇 개의 단어와 구문만 익숙해지면 사용하는데 큰 어려움이 없습니다. 또한 전문 개발자들의

프로그래밍에 비하면 상당히 쉬운 수준이므로, 비교적 빠른 시간 내에 원하는 전략을 테스트하고 수행하는 정도의 능력을 갖출 수도 있습니다.

## 1.3 R 프로그램

인간이 사용하는 언어의 종류가 다양하듯이, 프로그램 언어의 종류 역시 매우 다양합니다. 대략 700여개 이상의 프로그램 언어 중<sup>5</sup> 사람들이 대중적으로 사용하는 언어는 그리 많지 않으므로, 대중성과 효율성을 위해 사용량이 많은 언어를 이용하는 것이 좋습니다.

그림 1.5는 프로그래밍 언어의 사용 통계 순위<sup>6</sup>입니다. 이 중 R과 Python은 매우 대중적인 언어입니다. 해당 언어가 많이 사용되는 가장 큰 이유는 무료인 점, 그리고 일반인들이 사용하기에도 매우 편한 형태로 언어가 구성되어 있다는 점입니다.

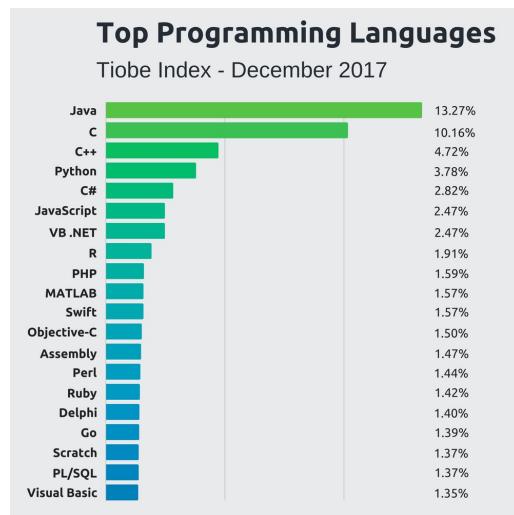


Figure 1.4: 2017년 기준 프로그래밍 언어 사용 통계 순위

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

<sup>5</sup>[https://en.wikipedia.org/wiki/List\\_of\\_programming\\_languages](https://en.wikipedia.org/wiki/List_of_programming_languages)

<sup>6</sup><https://www.tiobe.com/tiobe-index/>

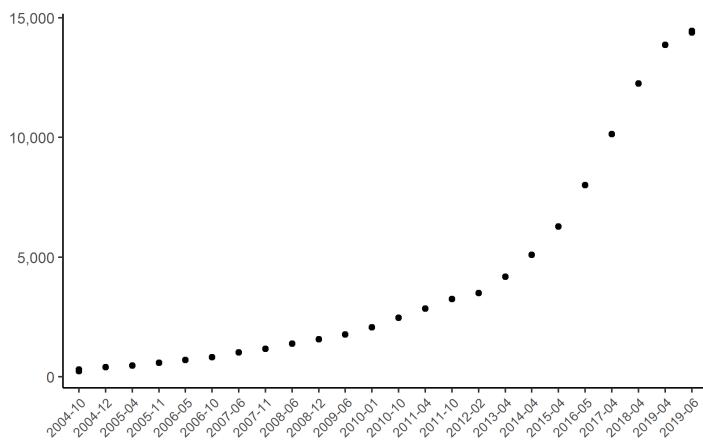


Figure 1.5: CRAN 등록 패키지 수

## 1.4 퀸트 투자에 유용한 R 패키지

여러 연구자 및 실무자들의 혼신적인 노력과 함께, R에는 금융 연구와 퀸트 투자를 위한 다양한 패키지들이 만들어져 있으며, 누구나 무료로 이용이 가능합니다. 그 중 해당 책의 내용에 사용되는 패키지 중 중요하다고 생각되는 것은 다음과 같으며, 각 패키지에 대한 자세한 설명은 구글에서 패키지 명으로 검색한 후 PDF 파일을 통해 확인할 수 있습니다.

- **quantmod**: 이름에서 알 수 있듯이 퀸트 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 매우 유용한 패키지입니다. 백테스트에 사용되는 `Return.portfolio()` 함수는 포트폴리오 백테스트에 필수적인 함수입니다. 또한 성과를 측정하는 상당히 많은 함수를 이용하여, 포트폴리오의 성과와 위험을 파악할 수 있습니다.
- **xts**: 기본적으로 금융 데이터는 시계열 형태이며, 해당 패키지는 여러 데이터들을 시계열 형태 Extensible Time Series로 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 유용한 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.
- **httr & rvest**: 데이터를 웹에서 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.

이 외에도 본 책에서는 다양한 패키지를 사용하였으며, 아래의 코드를 실행시 설치되지 않은 패키지를 설치할 수 있습니다.

```
pkg = c('magrittr', 'quantmod', 'rvest', 'httr', 'jsonlite',
       'readr', 'readxl', 'stringr', 'lubridate', 'dplyr',
       'tidyverse', 'ggplot2', 'corrplot', 'dygraphs',
       'highcharter', 'plotly', 'PerformanceAnalytics',
       'nloptr', 'quadprog', 'RiskPortfolios', 'cccp',
       'timetk', 'broom', 'stargazer')
```

```
new.pkg = pkg[!(pkg %in% installed.packages()[, "Package"])]
if (length(new.pkg)) {
  install.packages(new.pkg, dependencies = TRUE)}
```



# CHAPTER 2

---

## 크롤링을 위한 기본 지식

---

기준에 프로그래밍에 익숙한 분들도 크롤링은 생소한 경우가 많습니다. 기본적인 프로그래밍에 관한 책과 강의가 굉장히 많은 것과는 달리, 크롤링에 대한 지식을 접하기 힘들기 때문입니다. 물론 크롤링은 기계적인 단계가 많기 때문에 조금만 연습해도 매우 유용하게 사용할 수 있는 기술입니다. 그러나 복잡한 웹페이지나 데이터 내용을 수집하기 위해서는 인코딩, 통신구조에 대한 지식이 필요할 때가 있습니다.

본 장에서는 크롤링을 하기 위해 사전에 알고 있으면 도움이 되는 인코딩, 웹의 동작 방식, HTML과 CSS에 대해 알아보도록 하겠습니다. 그리고 실제 크롤링 시 유익하게 사용되는 파이프 오퍼레이터와 오류에 대한 예외처리에 대해서도 알아보도록 하겠습니다.

## 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 한글로 된 데이터를 크롤링하면 오류가 뜨거나 읽을 수 없는 문자로 나타나는 경우가 종종 있습니다. 이는 한글 인코딩 때문에 발생하는 문제이며, 이러한 현상을 흔히 인코딩이 깨졌다고 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 127 까지 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.

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

### 2.1.2 한글 인코딩의 종류

인코딩에 대한 전문적인 내용의 경우 본 책의 범위를 넘어가며, 크롤링을 위해서는 한글을 인코딩하는데 쓰이는 EUC-KR과 CP949, 그리고 UTF-8 정도만 이해해도 충분합니다. 만일 알이라는 단어를 인코딩하기 위해서는 어떠한 방법이 있을까요? 먼저 알이라는 문자 자체에 해당하는 코드를 부여하여 나타내는 방법이 있습니다. 아니면 이를 구성하는 모음과 자음을 나누어 ㅇ, ㅏ, 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 표현하기가 부족하여, 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 한글뿐만 아니라 다양한 언어에 적용할 수 있다는 장점으로 인해 전세계 웹페이지의 대부분이 UTF-8로 만들어지고 있습니다.

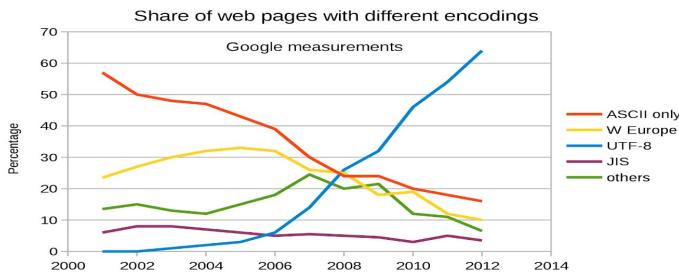


Figure 2.2: 웹페이지에서 사용되는 인코딩 비율

### 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 있어, 이러한 인코딩 방식의 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 그림 2.3와 같이 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 변경해주도록 합니다.

해당 방법으로도 해결되지 않을 경우 그림 2.4와 같이 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.

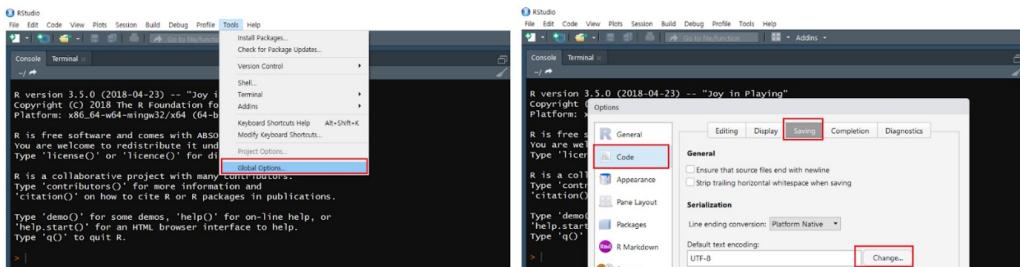


Figure 2.3: 인코딩 변경

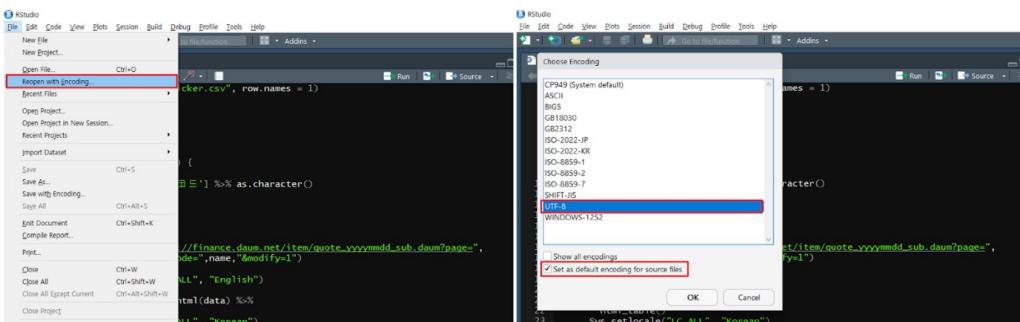


Figure 2.4: 인코딩 변경 후 재시작

## 2.2 웹의 동작 방식

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

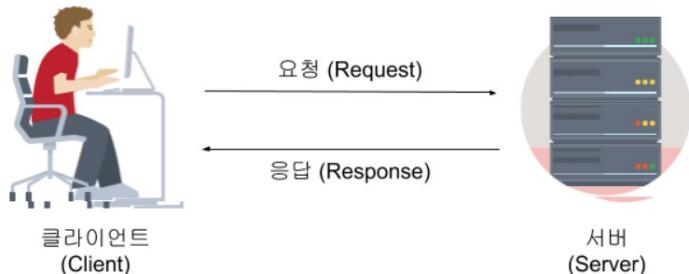


Figure 2.5: 웹 환경 구조

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 URL입니다.

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

### 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 표 2.1와 같이 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.

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

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 코드라 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.

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

## 2.3 HTML과 CSS 이해하기

클라이언트와 서버가 데이터를 주고 받을 때는 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)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.

메모장에서 HTML 코드를 입력한 후 파일명.html로 저장할 경우, 해당 코드가 웹페이지에서 어떻게 나타나는지 확인이 가능합니다.

### 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>
```

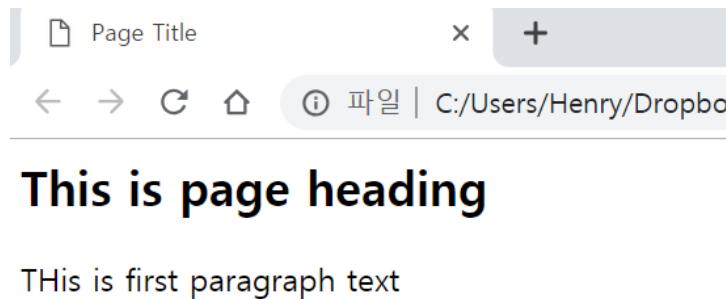


Figure 2.6: HTML 기본 구조

### 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, 뽑힌 자료에서 내용만을 찾는 방법등의 내용을 찾는 방법이 모두 다르기 때문에 이러한 요소에 대해 좀더 자세히 살펴보도록 하겠습니다.

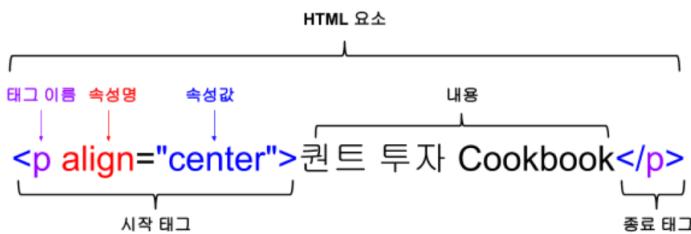


Figure 2.7: HTML 구성 요소 분석

꺽쇠 (<>)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>
```

# Page heading: size 1

## Page heading: size 2

### Page heading: size 3

Quant Cookbook

By Henry

Figure 2.8: h 태그와 p 태그 예제

### 2.3.4 리스트: ul과 ol 태그

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

```
<html>
<body>

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

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

</body>
</html>
```

## Unordered List

- Price
- Financial Statement
- Sentiment

## Ordered List

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

Figure 2.9: 리스트 관련 태그 예제

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>
  <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>
```

## 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

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.

```
<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 tag & href attribute

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

[Henry's Quantopia](#)

#### 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

a 태그 뒤 href 속성에 대한 속성값으로 연결하고자 하는 웹페이지의 주소를 입력한 후, 내용을 입력하면, 해당 텍스트에 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 찾으면 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.

```
<html>
<body>

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

<div style="background-color: yellow; color: red">
  <h5>Second Div</h5>
```

```

<p>Yellow backgrond, Red Color</p>
</div>

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

</body>
</html>

```



Figure 2.12: div 태그 예제

div 태그를 통해 총 3개의 레이아웃으로 나누어졌음이 확인됩니다. style 속성 중 background-color는 배경 색상을, color는 글자 색상을 의미하며, 각 레이아웃마다 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 셀렉터 Selector라 합니다.

```

<html>
<head>

```

```

<style>
body {background-color: powderblue;}
h4 {color: blue;}
</style>
</head>
<body>

<h4>This is a heading</h4>
<p>This is a first paragraph.</p>
<p>This is a second paragraph.</p>

</body>
</html>

```

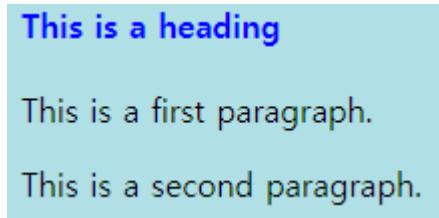


Figure 2.13: css 예제

head 태그 사이에 여러 태그에 대한 CSS 효과가 정의되었습니다. 먼저 body의 전체 배경색상을 파우더 블루로 설정하였으며, h4 태그의 글씨는 파란색으로 설정하였습니다. 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>

<div>
<h2 class="index">S&P 500</h2>
<p class="desc"> Market capitalizations of 500 large companies
having common stock listed on the NYSE, NASDAQ,
or the Cboe BZX Exchange</p>
</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 technology companies</p>
<div>
</html>

```



Figure 2.14: class 예제

셀렉터를 클래스에 적용할때는 클래스명 앞에 콤마(.)를 붙혀 표현합니다. 위의 예제에서 index 클래스는 배경 색상이 토마토, 글씨는 흰색, 여백은 10px로 정의

되었습니다. desc 클래스는 배경 색상이 모카신, 글씨는 검은색, 여백은 10px로 정의되었습니다. 본문의 첫번째(S&P 500)와 세번째(>NASDAQ Composite) 레이아웃의 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 --&gt;
&lt;h1 id="myHeader"&gt;My Header&lt;/h1&gt;

&lt;/body&gt;
&lt;/html&gt;</pre>

```



Figure 2.15: id 예제

셀렉터를 id에 적용할때는 클래스명 앞에 샵(#)를 붙혀 표현하며, 페이지에서 한번만 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.

HTML와 관련하여 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 필수적인 `rvest` 패키지를 설치할 경우 자동으로 `magrittr` 패키지가 설치되어 사용할 수 있습니다.

흔히 프로그래밍에서 `x`라는 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 결과값을 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 표 2.3와 같습니다.

Table 2.3: 파이프 오퍼레이터의 표현과 내용 비교

내용	표현. 방법
<code>F(x)</code>	<code>x %&gt;% F</code>
<code>G(F(x))</code>	<code>x %&gt;% F %&gt;% G</code>

파이프 오퍼레이터의 간단한 예제를 통해 사용법을 살펴보도록 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 팔호를 통해 감싸는 방법입니다. 앞선 방법에 비해 코드는 짧아졌지만, 계산 과정을 알아보기에는 매우 불편한 방법으로 코드가 짜여 있습니다.

```
library(magrittr)
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. `tryCatch()` 함수를 이용할 경우 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 발생 시 실행할 구문을 의미합니다. 이와 비슷하게, `error`와 `error-handler-code`는 각각 오류와 오류 발생 시 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 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.

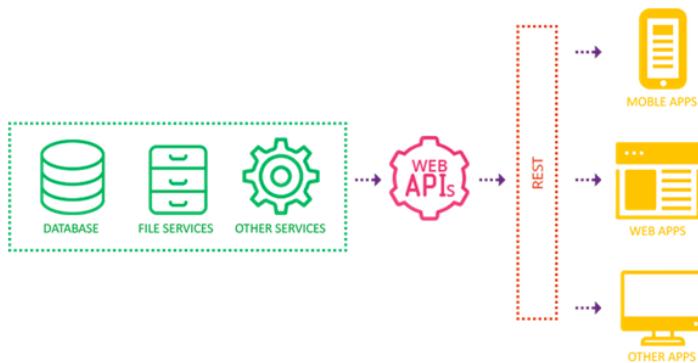


Figure 3.1: API 개념

API 주소만 가지고 있다면 데이터를 언제, 어디서, 누구나 쉽게 이용할 수 있다는 장점이 있습니다. 또한 대부분의 경우 사용자가 필요한 데이터만을 가지고 있으므로

로 접속 속도가 빠르며, 데이터를 가공하는 번거로움도 줄어듭니다. 해외의 경우 금융 데이터를 API의 형태로 제공하는 업체가 많으므로, 이를 잘만 활용한다면 매우 손쉽게 워크 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 예제로 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](https://www.quandl.com/api/v3/datasets/WIKI/AAPL/data.csv?api_key=xw3NU3xLUZ7vZgrz5QnG)

위 주소를 웹 브라우저 주소창에 직접 입력하면 csv 형식의 파일이 다운로드 되며, 이를 열어보면 애플의 주가에 해당하는 데이터가 있습니다.

	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

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
## 1 2018-03-27 173.7 175.2 166.9 168.3 38962839
## 2 2018-03-26 168.1 173.1 166.4 172.8 36272617
```

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

```

## 3 2018-03-23 168.4 169.9 164.9 164.9 40248954
## 4 2018-03-22 170.0 172.7 168.6 168.8 41051076
## 5 2018-03-21 175.0 175.1 171.3 171.3 35247358
## 6 2018-03-20 175.2 176.8 174.9 175.2 19314039
##   Ex.Dividend Split.Ratio Adj..Open Adj..High Adj..Low
## 1          0           1      173.7      175.2     166.9
## 2          0           1      168.1      173.1     166.4
## 3          0           1      168.4      169.9     164.9
## 4          0           1      170.0      172.7     168.6
## 5          0           1      175.0      175.1     171.3
## 6          0           1      175.2      176.8     174.9
##   Adj..Close Adj..Volume
## 1      168.3    38962839
## 2      172.8    36272617
## 3      164.9    40248954
## 4      168.8    41051076
## 5      171.3    35247358
## 6      175.2    19314039

```

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

## 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 것은 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 파이낸스 역시 주가 데이터를 무료로 제공하며, `quantmod` 패키지의 `getSymbols()` 함수는 해당 API에 접속하여 데이터를 다운로드 받습니다.

### 3.2.1 주가 다운로드

`getSymbols()` 함수의 기본적인 사용법은 매우 간단합니다. 괄호 안에 다운로드 받고자 하는 종목의 티커를 입력하면 됩니다.

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

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

```
head(AAPL)
```

	AAPL.Open	AAPL.High	AAPL.Low	AAPL.Close
## 2007-01-03	12.33	12.37	11.70	11.97
## 2007-01-04	12.01	12.28	11.97	12.24
## 2007-01-05	12.25	12.31	12.06	12.15
## 2007-01-08	12.28	12.36	12.18	12.21
## 2007-01-09	12.35	13.28	12.16	13.22
## 2007-01-10	13.54	13.97	13.35	13.86
	AAPL.Volume	AAPL.Adjusted		
## 2007-01-03	309579900	10.49		
## 2007-01-04	211815100	10.72		
## 2007-01-05	208685400	10.64		
## 2007-01-08	199276700	10.70		
## 2007-01-09	837324600	11.58		
## 2007-01-10	738220000	12.14		

먼저 `getSymbols()` 함수 내에 애플의 티커인 **AAPL**을 입력합니다. 티커와 동일한 변수인 AAPL이 생성되며, 주가 데이터가 다운로드 된 후 xts 형태로 입력됩니다.

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.

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



`Ad()` 함수를 통해 다운로드 받은 데이터에서 수정주가만을 선택한 후, `chart_Series()` 함수를 이용하여 시계열 그래프를 그려줄 수도 있습니다. 시계열 기간을 입력하지 않을 경우 2007년 1월부터 현재까지의 데이터가 다운로드 되며, 추가적인 입력 변수를 통해 원하는 기간의 데이터를 다운로드 받을 수도 있습니다.

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

##          AAPL.Open AAPL.High AAPL.Low AAPL.Close
## 2000-01-03     3.746     4.018    3.632     3.998
## 2000-01-04     3.866     3.951    3.614     3.661
## 2000-01-05     3.705     3.949    3.679     3.714
## 2000-01-06     3.790     3.821    3.393     3.393
## 2000-01-07     3.446     3.607    3.411     3.554
## 2000-01-10     3.643     3.652    3.384     3.491
##          AAPL.Volume AAPL.Adjusted
## 2000-01-03 133949200            3.502
## 2000-01-04 128094400            3.207
## 2000-01-05 194580400            3.254
## 2000-01-06 191993200            2.972
## 2000-01-07 115183600            3.113
## 2000-01-10 126266000            3.058
```

`from`에는 시작 시점을, `to`에는 종료 시점을 입력하여 주면, 해당 시점의 데이터가 다운로드 받아집니다.

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

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

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

```
head(FB)
```

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

```
head(NVDA)
```

```
##           NVDA.Open NVDA.High NVDA.Low NVDA.Close
## 2007-01-03     24.71     25.01    23.19     24.05
## 2007-01-04     23.97     24.05    23.35     23.94
## 2007-01-05     23.37     23.47    22.28     22.44
## 2007-01-08     22.52     23.04    22.13     22.61
## 2007-01-09     22.64     22.79    22.14     22.17
## 2007-01-10     21.93     23.47    21.60     23.26
##           NVDA.Volume NVDA.Adjusted
## 2007-01-03     28870500        22.19
## 2007-01-04     19932400        22.09
```

```
## 2007-01-05    31083600    20.70
## 2007-01-08    16431700    20.86
## 2007-01-09    19104100    20.45
## 2007-01-10    27718600    21.46
```

한번에 여러 종목의 주가를 다운로드 받을 수도 있습니다. 위 예제와 같이 페이스북과 엔비디아의 티커인 FB와 NVDA를 ticker 변수에 입력하여 주고, getSymbols() 함수에 티커들을 입력한 변수를 넣어주면 두 종목의 주가가 동시에 다운로드 됩니다.

### 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
## 2018-12-21      38293
## 2018-12-24      38442
## 2018-12-26      37996
## 2018-12-27      38250
## 2018-12-28      38700
```

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 양음부호를 붙여주어야 합니다.

국내 종목의 경우 종종 수정주가에 오류가 발생하는 경우가 많기에, 배당이 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 068670에 `.KQ`를 붙여 함수에 입력합니다. 역시나 데이터가 다운로드되어 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 Date이며, `getSymbols()` 함수를 통해 FRED 데이터를 다운로드 받을 수 있습니다. 먼저 미국 10년물 금리를 다운로드 받는 예제를 살펴보도록 하겠습니다.

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

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

```
chart_Series(DGS10)
```



미국채 10년물 금리에 해당하는 티커인 **DGS10**을 입력해주며, 데이터 출처에 해당하는 src에 **FRED**를 입력해줍니다. FRED에서 제공하는 API를 이용하여 데이터가 다운로드 되었으며, **chart\_Series()** 함수를 통해 금리 추이를 살펴볼 수 있습니다.

각 항목 별 티커를 찾는 방법은 매우 간단합니다. 먼저 FRED의 홈페이지<sup>2</sup>에 접속하여, 원하는 데이터를 검색합니다. 만일 원/달러 환율에 해당하는 티커를 찾고자 할 경우, 그럼 3.3와 같이 이에 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.

```
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
## 2019-06-21    1163
## 2019-06-24    1156
## 2019-06-25    1156
## 2019-06-26    1156
## 2019-06-27    1158
## 2019-06-28    1155
```

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

# CHAPTER 4

---

## 크롤링 이해하기

---

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

각종 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. 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 만들어진 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.

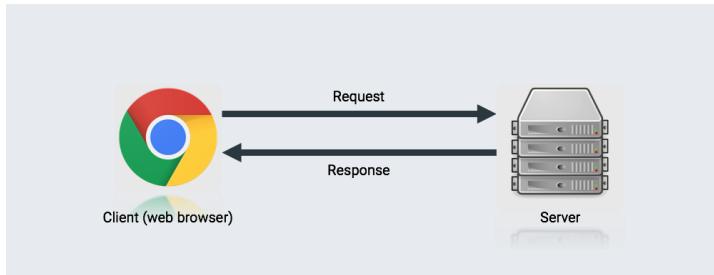


Figure 4.1: 클라이언트와 서버 간의 요청/응답 과정

### 4.1.1 GET 방식

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

한경컨센서스<sup>1</sup>에 접속한 후 상단의 탭에서 기업을 선택하면, 주소의 끝부분에 ?skinType=business가 추가되며 이에 해당하는 페이지의 내용을 보여줍니다. 즉, 해당 페이지는 GET 방식을 사용하고 있으며 입력종류는 skinType, 이에 해당하는 기업 탭의 입력값은 business임을 알 수 있습니다.

이번에는 파생 탭을 선택하여 봅니다. 역시나 홈페이지 주소가 변경되며 해당 주소에 맞는 내용이 나타납니다. 주소의 끝부분이 ?skinType=derivative로 변경되며, 입력 값이 변경됨에 따라 페이지의 내용이 이에 맞게 변하는 모습이 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 sdate와 edate, 그리고 now\_page 쿼리가 추가됩니다. sdate에 검색 기간의 시작시점, edate에 검색 기간의 종료시점, now\_page에 원하는 페이지를 수기로 입력해도 이에 해당하는

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

## 4.1. GET과 POST 방식 이해하기

47

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

The screenshot shows a search interface for corporate reports. The URL in the address bar is [hkconsensus.hankyung.com/apps.analysis/analysis.list?skinType=business](http://hkconsensus.hankyung.com/apps.analysis/analysis.list?skinType=business). The search filters are set to '2018-06-07' to '2018-07-07', '전체' (All), and '기업' (Corporate). The results table has columns for 작성일 (Date), 제목 (Title), 적정가격 (Fair Price), 투자의견 (Investment Recommendation), 작성자 (Author), 제공출처 (Source), and 기업정보 (Corporate Information). The results are listed for five companies: 삼성전자, 후성, 파리다이스, 동원산업, and GS건설, each with their respective details like price, recommendation, and author.

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 2차전지 주를 찾...	0	Not Rated	손세훈	NH투자증권			
2018-07-06	파리다이스(034230)6월 및 2분기 실적...	24,500	Buy	성준원, 강수연	신한금융투자			
2018-07-06	동원산업(006040)2분기도 순황 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

페이지의 데이터를 보여줍니다. 이처럼 GET 방식으로 데이터를 요청할 경우, 웹 페이지 주소를 수정하여 원하는 종류의 데이터를 받아올 수 있습니다.

② [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)

This screenshot shows the same search interface as Figure 4.2, but with a different URL. The URL is [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). The search filters remain the same. The results table shows a different set of five companies: BGF리테일, 휴비츠, 신세계, 삼성증권, and GS리테일. The page number 2 is highlighted with a red box at the bottom of the table.

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	삼성증권(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

### 4.1.2 POST 방식

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

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

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

구분	회사수	총목수	상장주식수(천주)	자본금(백만원)	시가총액(백만원)
주권	760	874	50,218,456	106,767,816	1,514,487,938
외국주권	1	1	47,870	-	179,271
투자회사	7	7	742,204	3,250,887	3,876,410
부동산투자회사	5	5	128,743	181,827	386,724

Figure 4.4: 상장공시시스템의 상장종목현황 메뉴

이번엔 조회일자를 2017-12-28로 선택한 후, 검색을 눌러보도록 합니다. 페이지의 내용은 선택일 기준으로 변경되었지만, 주소는 변경되지 않고 그대로 남아 있습니다. 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 개발자도구를 이용해야 하며, 크롬 브라우저에서 F12 키를 눌러 해당 화면을 열 수 있습니다. 개발자도구 화면을 연 상태에서 다시 한번 검색을 클릭해 봅니다. Network 탭을 클릭하면, 검색을 클릭함과 함께 브라우저와 서버간의 통신 과정을 살펴볼 수 있습니다. 이 중 `listedIssueStatus.do`라는 항목이 POST 형태임을 알 수 있습니다.

Name	Method	Status	Type	Request	Size	Time	Waterfall
listedIssueStatus.do	POST	200	xhr	listedIssueStatus.do?method=selDate&selDate=2017-12-28	29.3 KB	666 ms	

Figure 4.5: 크롬 개발자도구의 Network 화면

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

이처럼 POST 방식은 요청하는 데이터에 대한 쿼리가 GET 방식처럼 url을 통해

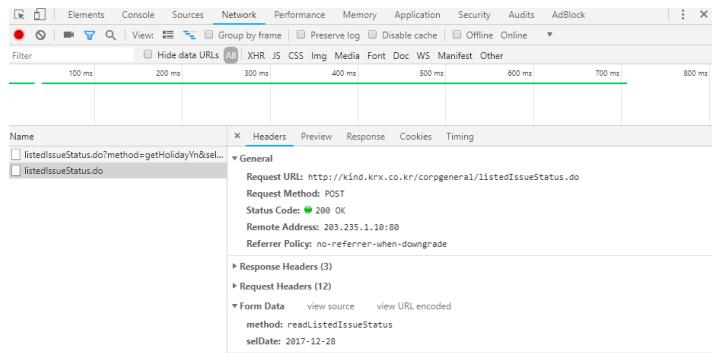


Figure 4.6: POST 방식의 서버 요청 내역

전송되는 것이 아닌 body를 통해 전송되므로, 이에 대한 정보는 웹브라우저를 통해 확인할 수 없습니다.

## 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 해당 장을 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)



Figure 4.7: 실시간 속보의 제목 부분 html

먼저 해당 페이지의 내용을 R로 불러오도록 하겠습니다.

```
library(rvest)
library(httr)

url = paste0('https://finance.naver.com/news/news_list.nhn?',
            'mode=LSS2D&section_id=101&section_id2=258')
data = GET(url)

print(data)
```

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, 인코딩(charset)은 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')
```

- 먼저 `read_html()` 함수를 이용하여 해당 페이지의 html 내용을 읽어오며, 인코딩은 **EUC-KR**로 셋팅해주도록 합니다.
- `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_attr()` 함수는 속성을 추출하는 함수로써, `title`에 해당하는 부분만을 추출합니다.

해당 과정을 거쳐 `data_title`에는 실시간 속보의 제목만이 저장됩니다. 이처럼 개발자도구 화면을 통해 내가 추출하고자 하는 데이터가 html 중 어디에 위치하고 있는지 먼저 확인을 하면, 어렵지 않게 해당 데이터를 읽어올 수 있습니다.

```
print(data_title)
```

```
## [1] "“스미모토 부동산, 도쿄 임대 업황 "
## [2] "\\"아직도 은행에서 환전하세요?\\"...외화"
## [3] "[부고] 양홍제(대신증권 컴플라이언스부"
## [4] "'상저' 지나 '하고' 기대하는 IT"
## [5] "韓증시는 부진했지만… 해외주식, 금 "
## [6] "올스웰, 中3위 수도강철과 기술협약"
## [7] "[주간증시전망] \"국내 주식시장 디커"
## [8] "미국 대형은행, 배당 매력 증가 기대"
## [9] "에듀윌, 국제무역사 및 무역영어 시험"
## [10] "7월증시 서머랠리는? "
## [11] "한화투자증권 \"화장품업종, 실적 차별"
## [12] "[영화로 경제 보기] BIFAN 폐막..."
## [13] "반일 감정 몰아..이번엔 `애국테마株"
## [14] "[주목! e해외주식] 테슬라, 2분기 사"
## [15] "[한주 증시 돌아보기] 매수 3주체 매"
## [16] "초등학교 여름방학 준비, 천재교육 밀"
## [17] "[주간추천주] 유안타증권"
## [18] "[주간추천주] KB증권"
## [19] "[주간추천주] SK증권"
## [20] "[주목! e스몰캡] 미스터블루, 웹툰 "
```

### 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



Figure 4.8: 오늘의공시 확인하기

쿼리가 body의 형태를 통해 전송되므로, 개발자도구 화면을 통해 해당 쿼리에 대한 내용을 확인해야 합니다.

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.

Figure 4.9: POST 방식의 데이터 요청

POST 방식으로 쿼리를 요청하는 방법을 코드로 나타내면 다음과 같습니다.

```
library(httr)
library(rvest)

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

url = 'http://kind.krx.co.kr/dDisclosure/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")

```

- 한글로 작성된 페이지를 크롤링 할 경우 오류가 발생하는 경우가 종종 있으므로, `Sys.setlocale()` 함수를 통해 로케일 언어를 영어로 설정 해줍니다.
- `POST()` 함수를 통해 해당 url에 원하는 쿼리를 요청해주며, 쿼리는 body 내에 list 형태로 입력해주도록 합니다. 해당 값은 개발자도구 화면의 Form Data와 동일하게 입력해주며, marketType과 같이 값이 존재하지 않는 항목은 입력하지 않아도 됩니다.
- `read_html()` 함수를 이용하여 해당 페이지의 html 내용을 읽어옵니다.
- `html_table()` 함수는 테이블 형태의 데이터를 읽어오는 함수입니다. 셀 간 병합이 된 열이 존재하므로 `fill=TRUE` 를 추가해주도록 합니다.
- `.[[1]]`를 통해 첫번째 리스트를 선택해 줍니다.
- 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
## 1 18:32      화신테크
## 2 18:26    에스제이케이
## 3 18:11      아이엠텍
## 4 18:10    시그넷이브이
## 5 18:09
## 6 18:09
##                                     NA
## 1                               최대주주변경
## 2 증권  발행결과(자율공시)(제3자배정 유상증자)

```

```

## 3 [정정] 유상증자결정(제3자배정)
## 4                               유형자산 양수 결정
## 5       자기주식매매신청내역(코스닥시장)
## 6       대량매매내역(코스닥시장)
##               NA
## 1   화신테크 공시차트\r\n\t\t\t\t\t\t\t\t\t주가차트
## 2   에스제이케이 공시차트\r\n\t\t\t\t\t\t\t\t\t주가차트
## 3   아이엠텍 공시차트\r\n\t\t\t\t\t\t\t\t\t\t주가차트
## 4   시그넷이브이 공시차트\r\n\t\t\t\t\t\t\t\t\t\t\t주가차트
## 5 코스닥시장본부
## 6 코스닥시장본부

```

POST 형식의 경우 body에 들어가는 퀴리 내용을 바꾸어 원하는 데이터를 받을 수 있습니다. 만일 2019년 1월 8일 공시를 확인하고자 할 경우, 위의 코드에서 selDate만 **2019-01-08**로 변경해주면 됩니다. 아래 코드의 출력 결과물을 2019년 1월 4일 공시와 확인하면 동일한 결과임을 확인할 수 있습니다.

```

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

url = 'http://kind.krx.co.kr/dDisclosure/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 = '2019-01-08'
            ))

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

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

```

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

```
##          NA          NA
## 1 18:58 해덕파워웨이
## 2 18:57 해덕파워웨이
## 3 18:57 퓨전데이타
## 4 18:56 해덕파워웨이
## 5 18:52 한국테크놀로지
## 6 18:52 한국테크놀로지
##
##                                     NA
## 1                                         소속부변경
## 2 주권매매거래정지기간변경(상장적격성 실질심사 대상 결정)
## 3                                         최대주주변경
## 4             관리종목지정(상장적격성 실질심사 대상 결정 )
## 5                                         전환사채권발행결정(제18회차)
## 6                                         전환사채권발행결정(제17회차)
##
##          NA          NA
## 1 코스닥시장본부 공시차트\r\n\t\t\t\t\t\t\t\t\t\t주가차트
## 2 코스닥시장본부 공시차트\r\n\t\t\t\t\t\t\t\t\t\t주가차트
## 3     퓨전데이타 공시차트\r\n\t\t\t\t\t\t\t\t\t\t주가차트
## 4 코스닥시장본부 공시차트\r\n\t\t\t\t\t\t\t\t\t\t\t주가차트
## 5 한국테크놀로지 공시차트\r\n\t\t\t\t\t\t\t\t\t\t\t주가차트
## 6 한국테크놀로지 공시차트\r\n\t\t\t\t\t\t\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 종목명을 클릭하여 이동하는 페이지의 url을 확인해보면, 끝 6자리가 각 종목의 거래소 티커임도 확인이 됩니다.

티커 정리를 위해 html에서 확인해야 할 부분은 총 2가지입니다. 먼저 하단의 페이지 네비게이션을 통해 코스피와 코스닥 시가총액에 해당하는 페이지가 각각

몇 번째 페이지까지 존재하는지를 알아야 합니다. 아래와 같은 항목 중 맨뒤에 해당하는 페이지가 가장 마지막 페이지입니다.

1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 다음 ► | 맨뒤 ►

Figure 4.10: 페이지 네비게이션

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



Figure 4.11: HTML 내 페이지 네비게이션 부분

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

Figure 4.12: 네이버 금융 시가총액 페이지

삼성전자에 마우스를 올려둔 후 우클릭 → 검사를 통해 개발자도구 화면을 살펴

보면, 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 url이, 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')
down_table = GET(url)
```

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.

가장 먼저 해야 할 작업은 가장 마지막 페이지가 몇 번째 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-KR로 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 알고 싶은 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 뒤에서 첫번째 데이터만 선택합니다.
4. `as.numeric()` 함수를 통해 해당 값을 숫자 형태로 바꾸어 주도록 합니다.

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

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

코스피 시가총액 페이지는 31번째 페이지까지 존재하며, for loop 구문을 이용할 경우 1 페이지부터 `navi.final`, 즉 31 페이지까지 모든 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")

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

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

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 추가 해줍니다.
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
## 1	NA						
## 2 1	삼성전자	45,650	350	-0.76%	100	2,725,206	
## 3 2	SK하이닉스	68,400	1,800	-2.56%	5,000	497,954	
## 4 3	삼성전자우	37,350	650	-1.71%	100	307,348	
## 5 4	현대차	141,500	2,500	+1.80%	5,000	302,340	
## 6 5	셀트리온	208,000	4,500	+2.21%	1,000	266,924	
## 7	상장주식수	외국인비율	거래량	PER	ROE	토론실	
## 8 1			NA	<NA>	<NA>	NA	
## 9 2	5,969,783		57.38	7,222,188	7.58	19.63	NA
## 10 3	728,002		50.70	2,037,255	3.20	38.53	NA
## 11 4	822,887		92.73	599,446	6.20	N/A	NA
## 12 5	213,668		44.37	338,120	26.44	2.20	NA
## 13 6	128,329		21.39	259,974	101.51	10.84	NA

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

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

	N	종목명	현재가	전일비	등락률	액면가	시가총액
##	2	1 삼성전자	45,650	350	-0.76%	100	2,725,206
##	3	2 SK하이닉스	68,400	1,800	-2.56%	5,000	497,954
##	4	3 삼성전자우	37,350	650	-1.71%	100	307,348
##	5	4 현대차	141,500	2,500	+1.80%	5,000	302,340
##	6	5 셀트리온	208,000	4,500	+2.21%	1,000	266,924
##	10	6 LG화학	354,500	4,000	-1.12%	5,000	250,250
##		상장주식수 외국인비율		거래량	PER	ROE	
##	2	5,969,783	57.38	7,222,188	7.58	19.63	
##	3	728,002	50.70	2,037,255	3.20	38.53	
##	4	822,887	92.73	599,446	6.20	N/A	
##	5	213,668	44.37	338,120	26.44	2.20	
##	6	128,329	21.39	259,974	101.51	10.84	
##	10	70,592	38.90	99,907	18.84	8.86	

이제 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"
## [2] "/item/board.nhn?code=005930"
## [3] "/item/main.nhn?code=000660"
## [4] "/item/board.nhn?code=000660"
## [5] "/item/main.nhn?code=005935"
## [6] "/item/board.nhn?code=005935"
## [7] "/item/main.nhn?code=005380"
## [8] "/item/board.nhn?code=005380"
```

```
## [9] "/item/main.nhn?code=068270"
## [10] "/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 `symbol`에는 `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=005935 /item/board.nhn?code=005935
##                               "005935"                      "005935"
## /item/main.nhn?code=005380 /item/board.nhn?code=005380
##                               "005380"                      "005380"
## /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 추출되지만, 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" "005935" "005380" "068270"
## [6] "051910" "012330" "005490" "017670" "055550"
```

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

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

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

1. 위에서 구한 티커를 N열에 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 j번째 리스트에 정리된 데이터를 입력해 줍니다.

위의 코드에서 i와 j값을 for loop를 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()

# 첫번째부터 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 주식티커를 크롤링하는 방법에 대해 살펴보았습니다. 그러나 해당 방법은 지나치게 복잡하고 시간이 오래 걸립니다. 반면 한국거래소에서 제공하는 산업별 현황과 개별종목 지표 데이터를 이용할 경우 훨씬 간단하게 주식티커 데이터를 수집할 수 있습니다.

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

해당 데이터들을 크롤링이 아닌 Excel 버튼을 눌러 엑셀로 받을 수도 있습니다. 그러나 매번 엑셀을 다운받고 이를 R로 불러오는 작업은 상당히 비효율적이며, 크롤링을 이용한다면 해당 데이터를 R로 직접 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** 두가지 항목이 존재합니다. 거래소에서 엑셀 데이터를 받는 과정은 다음과 같습니다.

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

먼저 1번 단계를 살펴보도록 하겠습니다.

시행구분	종목코드	종목명	산업분류	현재가(총가)	전일대비	시가총액
코스피	039040	동원수산	여업	8940	▲ 20	4
코스피	007160	시조산업	여업	54400	▲ 800	27
코스피	006040	동원신법	여업	246500	▲ 500	82
코스피	004970	신리고목	여업	14350	▲ 350	22
코스피	012230	장동인베스트	광업	40,800	▲ 1,350	9
코스피	005590	넥스토트리뷴	광업	5,200	▲ 260	12
코스피	017810	풀무원	음식료품	11,300	▲ 200	43
코스피	28390	롯데제과	음식료품	159,500	▼ 500	1,02
코스피	27190	오리온	음식료품	63,900	▲ 1,400	329
코스피	000090	시조소방	음식료품	8,220	▲ 30	7
코스피	26490	크라우제과우	음식료품	7,930	▲ 80	

Figure 5.1: OTP 생성 부분

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.

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

위 과정을 코드로 나타내면 다음과 같습니다.

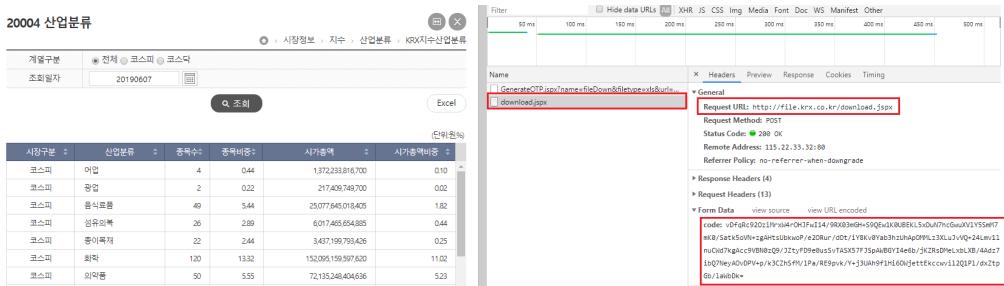


Figure 5.2: OTP 제출 부분

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

gen_otp_url =
  'http://marketdata.krx.co.kr/contents/COM/GenerateOTP.jsp'
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 나타는 쿼리 내용들을 리스트 형태로 입력합니다. 단, filetype은 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 값만을 추출하게 됩니다.

위의 과정을 거쳐 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. 거래소 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 거쳐온 과정을 흔적으로 남겨야 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 filetype을 csv로 지정했기에, 손쉽게 데이터를 읽어올 수 있습니다.

```
print(down_sector)
```

```
## # A tibble: 2,243 x 7
##   시장구분 종목코드 종목명 산업분류 `현재가(종가)` 
##   <chr>    <chr>    <chr>    <chr>      <dbl>
## 1 코스피    030720   동원수산~ 어업        8940
## 2 코스피    007160   사조산업~ 어업       54400
## 3 코스피    006040   동원산업~ 어업       246500
## 4 코스피    004970   신라교역~ 어업       14350
## 5 코스피    012320   경동인베스~ 광업      40300
## 6 코스피    003580   넥스트사이~ 광업       5200
## 7 코스피    017810   풀무원 음식료품     11300
## 8 코스피    280360   롯데제과~ 음식료품     159500
## 9 코스피    271560   오리온 음식료품     83300
## 10 코스피   006090   사조오양~ 음식료품    8220
## # ... with 2,233 more rows, and 2 more variables:
## #   전일대비 <dbl>, `시가총액(원)` <dbl>
```

위 과정을 통해 down\_sector 변수에는 산업별 현황 데이터가 저장되었습니다. 이를 csv 파일로 저장하겠습니다.

```
if else(dir.exists('data'), FALSE, dir.create('data'))
write.csv(down_sector, 'data/krx_sector.csv')
```

먼저 ifelse() 함수를 통해 data라는 이름의 폴더가 존재할 시에는 FALSE를 반환, 존재하지 않을 시 해당 이름으로 폴더를 생성하여 줍니다. 그 후, 위에서 다운로드 받은 데이터를 data 폴더 내에 krx\_sector.csv 이름으로 저장하여 줍니다. 해당 폴더를 확인해보면, 데이터가 csv 형태로 저장되어 있습니다.

### 5.1.2 개별종목 지표 크롤링

개별종목 데이터를 크롤링하는 방법은 위와 매우 유사하며, 요청하는 쿼리 값에만 차이가 있습니다. 개발자도구 화면을 연 상태에서 csv 버튼을 눌러주어 어떤 쿼리를 요청하는지 확인하도록 합니다.

종목명	관리부	종가	EPS	PER	EPS
스몰리아이	-	3,800	170	22.35	7974
민	-	16,050	1,609	9.98	8,878
슬시큐어	-	4,520	-	-	2,980
프원	-	13,450	1,091	12.33	8,686
N	-	1,450	52	27.88	1,604
삼천기공업	-	13,500	1,350	10	20,439
케이이씨...	관리종목	7,010	-	-	3,176
기온한역	-	8,360	104	80.38	5,568
진시스템	-	27,000	2,016	13.39	11,982
이브라우저	-	3,190	5	638	1,548
파이낸티	-	2,245	-	-	1,043
감읍픽스	-	14,700	1,246	11.8	3,595
노믹트리	-	29,100	-	-	695
책여섯제5...	-	5,200	-	-	-
	-	8,460	861	9.83	4,759

Figure 5.3: 개별지표 OTP 생성 부분

이 중 isu\_cdnm, isu\_cd, isu\_nm, isu\_srt\_cd, fromdate 항목은 종목구분의 개별탭에 해당하는 부분이므로 우리가 원하는 전체 데이터를 받을 때에는 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.jspx'
gen_otp_data = list(
  name = 'fileDown',
  filetype = 'csv',
  url = "MKD/13/1302/13020401/mkd13020401",
  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 종가 EPS    PER
##   <date>    <chr>   <chr>   <chr>   <dbl> <chr> <chr>
## 1 2019-06-07 060300 레드로버~ -       1190  -     -
## 2 2019-06-07 290650 엘앤씨바이~ -     22750 830  27.41
## 3 2019-06-07 239340 미래에셋제~ -     5200  -     -
## 4 2019-06-07 033430 디에스티~ -     1310  -     -
## 5 2019-06-07 038680 에스넷 ~ -     9600  300  32
## 6 2019-06-07 214680 디알텍 ~ -     2005  29   69.14
## 7 2019-06-07 242040 나무기술~ -     3345  -     -
## 8 2019-06-07 088800 에이스테크~ -     11500 97   118.~
## 9 2019-06-07 121890 에스디시스~ -     4980  -     -

```

```
## 10 2019-06-07 032500 케이엠더블~ - 41200 -
## # ... with 2,194 more rows, and 6 more variables:
## #   BPS <chr>, PBR <chr>, 주당배당금 <dbl>,
## #   배당수익률 <dbl>, `개시물 일련번호` <dbl>,
## #   총카운트 <dbl>
```

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

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

### 5.1.3 최근 영업일 기준 데이터 받기

위 예제의 쿼리 항목 중 `date`와 `schdate` 부분을 원하는 일자로 입력할 경우(예: 20190104), 해당일의 데이터를 다운로드 받을 수 있으며, 전 영업일 날짜를 입력할 경우 가장 최근의 데이터를 받을 수 있습니다. 그러나 매번 해당 항목을 입력하는 것은 번거로우므로, 자동으로 반영되게 할 필요가 있습니다.

네이버 금융의 국내증시 → 증시자금동향에는 이전 2영업일에 해당하는 날짜가 있으며, 자동으로 날짜가 업데이트 된다는 편리함이 있습니다. 따라서 해당 부분을 크롤링하여 쿼리 항목에 사용할 수 있습니다.



Figure 5.4: 최근 영업일 부분

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.



Figure 5.5: OTP 생성 부분

```
//*[@id="type_0"] /div/ul[2]/li/span
```

위에서 구한 날짜의 Xpath를 이용하여 해당 데이터를 크롤링하도록 하겠습니다.

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

url = 'https://finance.naver.com/sise/sise_deposit.nhn'

biz_day = GET(url) %>%
  read_html(encoding = 'EUC-KR') %>%
  html_nodes(xpath =
    '//*[@id="type_1"] /div/ul[2]/li/span') %>%
  html_text() %>%
  str_match(([0-9]+.[0-9]+.[0-9]+')) ) %>%
  str_replace_all('\\.', '')

print(biz_day)

## [1] "20190703"
```

1. 페이지의 url을 저장합니다.
2. GET() 함수를 통해 해당 페이지 내용을 받습니다.
3. read\_html() 함수를 이용하여 해당 페이지의 html 내용을 읽어오며, 인코딩은 EUC-KR로 셋팅해주도록 합니다.
4. html\_node() 함수 내에 위에서 구한 Xpath를 입력하여, 해당 지점의 데이터를 추출합니다.
5. html\_text() 함수를 통해 텍스트 데이터만을 추출합니다.

6. `str_match()` 함수 내에서 정규표현식<sup>1</sup>을 사용하여 숫자.숫자 형식의 데이터를 추출합니다.
7. `str_replace_all()` 함수를 이용하여 콤마(.)를 모두 없애주도록 합니다.

이처럼 Xpath를 이용할 경우 태그나 속성을 분해하지 않고도 원하는 지점의 데이터를 크롤링할 수 있습니다. 위 과정을 통해 yyyyymmdd 형태의 날짜만 남게 되었습니다. 이를 위의 date와 schdate에 입력하면 산업별 현황과 개별종목 지표를 최근일자 기준으로 내려받게 됩니다. 전체 코드는 다음과 같습니다.

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

# 최근 영업일 구하기
url = 'https://finance.naver.com/sise/sise_deposit.nhn'

biz_day = GET(url) %>%
  read_html(encoding = 'EUC-KR') %>%
  html_nodes(xpath =
    '//*[@id="type_1"]/div/ul[2]/li/span') %>%
  html_text() %>%
  str_match(([0-9]+.[0-9]+.[0-9]+')) ) %>%
  str_replace_all('\\.', '')

# 산업별 현황 OTP 발급
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 = biz_day, # 최근영업일로 변경
  lang = 'ko',
  pagePath = '/contents/MKD/03/0303/03030103/MKD03030103.jsp')
otp = POST(gen_otp_url, query = gen_otp_data) %>%
  read_html() %>%
  html_text()
```

<sup>1</sup>특정한 규칙을 가진 문자열의 집합을 표현하는데 사용하는 형식 언어

```
# 산업별 현황 데이터 다운로드
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()

ifelse(dir.exists('data'), FALSE, dir.create('data'))
write.csv(down_sector, 'data/krx_sector.csv')

# 개별종목 지표 OTP 발급
gen_otp_url =
  'http://marketdata.krx.co.kr/contents/COM/GenerateOTP.jspx'
gen_otp_data = list(
  name = 'fileDown',
  filetype = 'csv',
  url = "MKD/13/1302/13020401/mkd13020401",
  market_gubun = 'ALL',
  gubun = '1',
  schdate = biz_day, # 최근영업일로 변경
  pagePath = "/contents/MKD/13/1302/13020401/MKD13020401.jsp")

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

# 개별종목 지표 데이터 다운로드
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()

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

### 5.1.4 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/krx_sector.csv', row.names = 1,
                      stringsAsFactors = FALSE)
down_ind = read.csv('data/krx_ind.csv', row.names = 1,
                    stringsAsFactors = FALSE)
```

`read.csv()` 함수를 이용하여 csv 파일을 불러오며, `row.names = 1`를 통해 첫 번째 열을 행이름으로, `stringsAsFactors = FALSE`를 통해 문자열 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, 종목 코드와 종목명이 동일하게 위치합니다.

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

```
## [1] "엘브이엠씨홀딩스"      "한국패러랠"
## [3] "한국ANKOR유전"        "맵스리얼티1"
## [5] "맥쿼리인프라"          "하나니켈2호"
## [7] "하나니켈1호"            "베트남개발1"
## [9] "신한알파리츠"          "이리츠코크렙"
## [11] "모두투어리츠"           "하이골드12호"
## [13] "하이골드8호"            "바다로19호"
## [15] "하이골드3호"            "케이탑리츠"
## [17] "에이리츠"                "동북아13호"
## [19] "동북아12호"              "컬러레이"
## [21] "JTC"                     "뉴프라이드"
## [23] "윙입푸드"                 "글로벌에스엠"
## [25] "크리스탈신소재"          "씨케이에이치"
## [27] "차이나그레이트"          "골든센츄리"
## [29] "오가닉티코스메틱"        "GRT"
```

```

## [31] "로스웰"           "형성그룹"
## [33] "이스트아시아홀딩스" "에스앤씨엔진그룹"
## [35] "SNK"                 "SBI핀테크솔루션즈"
## [37] "잉글우드랩"         "코오롱티슈진"
## [39] "액세스바이오"

```

`setdiff()` 함수를 통해 두 데이터에 공통적으로 존재하지 않는 종목명, 즉 하나의 데이터에만 존재하는 종목을 살펴보면 위와 같습니다. 해당 종목들은 **선팩펀드**, **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 `by`를 기준으로 두 데이터를 하나로 합치며, 공통으로 존재하는 종목코드, 종목명을 기준으로 입력해줍니다. 또한 `all` 값을 `TRUE`로 설정할 경우는 합집합을, `FALSE`로 설정할 경우 교집합을 반환하며, 공통적으로 존재하는 항목을 원하므로 `FALSE`를 선택해 주도록 합니다.

```

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

```

	종목코드	종목명	시장구분	산업분류	현재가.종가
## 330	005930	삼성전자	코스피	전기전자	45400
## 45	000660	SK하이닉스	코스피	전기전자	69100
## 331	005935	삼성전자우	코스피	전기전자	37800
## 301	005380	현대차	코스피	운수장비	136000
## 1278	068270	셀트리온	코스피	의약품	206000
## 1082	051910	LG화학	코스피	화학	355500
##	전일대비	시가총액.원.	일자	관리여부	종가
## 330	-850	2.710e+14	2019-07-03	-	45400
## 45	-2300	5.030e+13	2019-07-03	-	69100
## 331	-350	3.111e+13	2019-07-03	-	37800
## 301	-1000	2.906e+13	2019-07-03	-	136000
## 1278	1000	2.644e+13	2019-07-03	-	206000
## 1082	7000	2.510e+13	2019-07-03	-	355500

```

##          EPS      PER      BPS      PBR 주당배당금 배당수익률
## 330    6,461   7.03   35,342   1.28     1416     3.12
## 45     22,255   3.1    64,348   1.07     1500     2.17
## 331      -       -       -       -     1417     3.75
## 301    5,632  24.15  245,447   0.55     4000     2.94
## 1278   2,063  99.85  19,766  10.42       0     0.00
## 1082   19,217  18.5   218,227   1.63    6000     1.69
##           게시물..일련번호 총카운트
## 330            2165      NA
## 45             1885      NA
## 331            2166      NA
## 301            2159      NA
## 1278           2049      NA
## 1082           2041      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[grep1('스팩', KOR_ticker[, '종목명']), '종목명']
```

```

## [1] "엔에이치스팩10호"      "케이비제10호스팩"
## [3] "엔에이치스팩14호"      "대신밸런스제5호스팩"
## [5] "케이비제18호스팩"      "삼성스팩2호"
## [7] "엔에이치스팩12호"      "한화에스비아이스팩"
## [9] "엔에이치스팩11호"      "신한제4호스팩"
## [11] "케이비17호스팩"        "IBKS제9호스팩"
## [13] "하나금융11호스팩"      "SK4호스팩"
## [15] "신한제5호스팩"        "한국제7호스팩"
## [17] "대신밸런스제6호스팩"    "미래에셋대우스팩1호"
## [19] "IBKS제6호스팩"         "대신밸런스제4호스팩"
## [21] "동부스팩5호"          "IBKS제5호스팩"
## [23] "DB금융스팩6호"        "삼성머스트스팩3호"
## [25] "상상인이안1호스팩"    "하나마스트제6호스팩"
## [27] "하나금융10호스팩"      "유안타제4호스팩"
## [29] "교보7호스팩"          "DB금융스팩7호"
## [31] "한국제6호스팩"         "IBKS제10호스팩"
## [33] "한화수성스팩"          "신영스팩4호"

```

```

## [35] "하이제4호스팩"      "하나금융9호스팩"
## [37] "유안타제3호스팩"    "한국제5호스팩"
## [39] "신영스팩5호"        "유진스팩4호"
## [41] "교보8호스팩"        "IBKS제7호스팩"
## [43] "신한제3호스팩"      "키움제5호스팩"
## [45] "엔에이치스팩13호"   "SK3호스팩"
## [47] "미래에셋대우스팩2호" "한국제8호스팩"
## [49] "한화에이스스팩4호"   "케이비제11호스팩"
## [51] "한화에이스스팩3호"

```

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

```
## character(0)
```

```

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

```

```

## [1] "삼성전자우"          "미래에셋대우"
## [3] "현대차우"            "LG생활건강우"
## [5] "LG화학우"            "아모레퍼시픽우"
## [7] "삼성화재우"          "LG전자우"
## [9] "신영증권우"          "두산우"
## [11] "연우"                 "한국금융지주우"
## [13] "대신증권우"          "S-Oil우"
## [15] "NH투자증권우"        "아모레G우"
## [17] "대림산업우"          "CJ제일제당 우"
## [19] "LG우"                 "SK이노베이션우"
## [21] "삼성SDI우"           "삼성전기우"
## [23] "CJ우"                 "금호석유우"
## [25] "SK우"                 "미래에셋대우우"
## [27] "GS우"                 "롯데칠성우"
## [29] "코오롱인더우"        "부국증권우"
## [31] "롯데지주우"          "유한양행우"
## [33] "유화증권우"          "호텔신라우"
## [35] "SK케미칼우"          "남양유업우"
## [37] "LG하우시스우"        "BYC우"
## [39] "유안타증권우"        "세방우"
## [41] "SK디스커버리우"      "대덕전자1우"

```

```

## [43] "태영건설우"           "한진칼우"
## [45] "대상우"                 "대한항공우"
## [47] "현대건설우"           "금호산업우"
## [49] "한화케미칼우"           "한화우"
## [51] "삼양홀딩스우"           "녹십자홀딩스2우"
## [53] "넥센우"                 "신풍제약우"
## [55] "삼양사우"               "SK증권우"
## [57] "코오롱우"               "NPC우"
## [59] "계양전기우"             "한화투자증권우"
## [61] "남선알미우"             "SK네트웍스우"
## [63] "태양금속우"             "쌍용양회우"
## [65] "서울식품우"             "대원전선우"
## [67] "일양약품우"             "유유제약1우"
## [69] "대한제당우"             "동원시스템즈우"
## [71] "크라운해태홀딩스우"   "성신양회우"
## [73] "코리아씨우"             "성문전자우"
## [75] "현대비앤지스틸우"       "삼성중공우"
## [77] "대호피앤씨우"           "CJ씨푸드1우"
## [79] "금강공업우"             "크라운제과우"
## [81] "깨끗한나라우"           "덕성우"
## [83] "대상홀딩스우"           "동부제철우"
## [85] "동양우"                  "노루페인트우"
## [87] "신원우"                  "코오롱글로벌우"
## [89] "하이트진로홀딩스우"   "DB하이텍1우"
## [91] "흥국화재우"              "JW중외제약우"
## [93] "한양증권우"              "동부건설우"
## [95] "노루홀딩스우"           "소프트센우"

```

```

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

```

```

## [1] "현대차2우B"           "미래에셋대우2우B"
## [3] "한화3우B"               "현대차3우B"
## [5] "삼성물산우B"           "대교우B"
## [7] "대신증권2우B"          "두산2우B"
## [9] "넥센타이어1우B"        "하이트진로2우B"
## [11] "진흥기업우B"           "진흥기업2우B"
## [13] "JW중외제약2우B"        "흥국화재2우B"
## [15] "코리아씨키트2우B"      "동양2우B"

```

```
## [17] "유유제약2우B"      "대한제당3우B"
## [19] "동양3우B"
```

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

```
## [1] "루트로닉3우C"
```

`grepl()` 함수를 통해 종목명에 ‘스팩’이 들어가는 종목, 스팩 종목인 ‘골든브릿지이안5호’, `substr()` 함수를 통해 종목명 끝이 ‘우’, ‘우B’, ‘우C’인 우선주 종목을 찾을 수 있습니다. 데이터 내에서 해당 데이터들을 제거<sup>2</sup>해 주도록 하겠습니다.

```
KOR_ticker = KOR_ticker[!grepl('스팩', KOR_ticker[, '종목명']), ]
KOR_ticker = KOR_ticker[KOR_ticker[, '종목명'] != '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.

---

<sup>2</sup> 해당 과정에서 미래에셋대우, 연우 등 의도치 않은 종목 역시 제거됩니다. 그러나 이러한 종목수가 그리 많지 않으므로 투자에 있어 중요하지는 않습니다.

```
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>3</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>4</sup>에서는 GICS와 비슷한 WICS 산업분류를 발표하고 있으므로, 이를 크롤링하여 필요한 정보를 수집해보도록 하겠습니다.

먼저, 웹페이지에 접속하여 **Index → WISE SECTOR INDEX → WICS → 에너지를 클릭합니다.** 그 후 Components 탭을 클릭하면, 해당 섹터의 구성 종목을 확인할 수 있습니다.

The screenshot shows the WiseIndex website interface. On the left, there's a sidebar with a tree view of index categories like WM500 INDEX, WMSE MARKET INDEX, WMSE STYLE INDEX, and WMSE SECTOR INDEX. Under WMSE SECTOR INDEX, the '에너지' (Energy) category is highlighted with a red box. The main content area has tabs for Overview, Performance, Components (which is highlighted with a red box), and Ratio. The Components tab shows a table with three rows: 지수시가총액 (Market Value), 거래대금 (Trading Volume), and 종목수 (Number of Stocks). To the right, there's a '구성종목' (Composition Stocks) table with several entries, each with a red box around it. The table columns are 종목명 (Stock Name) and 섹터명 (Sector Name). The data includes SK이노베이션 (Energy), S-Oil (Energy), GS (Energy), 에이치엘비생명과학 (Energy), SK디스커버리 (Energy), and 한국쉘석유 (Energy).

종목명	섹터명
SK이노베이션	에너지
S-Oil	에너지
GS	에너지
에이치엘비생명과학	에너지
SK디스커버리	에너지
한국쉘석유	에너지

Figure 5.6: WICS 기준 구성종목

<sup>3</sup>[https://en.wikipedia.org/wiki/Global\\_Industry\\_Classification\\_Standard](https://en.wikipedia.org/wiki/Global_Industry_Classification_Standard)

<sup>4</sup><http://www.wiseindex.com/>

개발자도구 화면(그림 5.7)을 통해 해당 페이지의 데이터 전송 과정을 살펴보도록 하겠습니다.

다음 사용어는 베너 사용미(설정화면)과 같은 버튼은 디렉터리에 포함된 앱의 기본 설정, 애플리케이션에 대한 정보 및 수신을 담당하는 입력 필드, 이외 템크 및 관련 소모 연료 생산업체나 배양장 등 [에너지] 버튼으로 구성된 지수입니다.

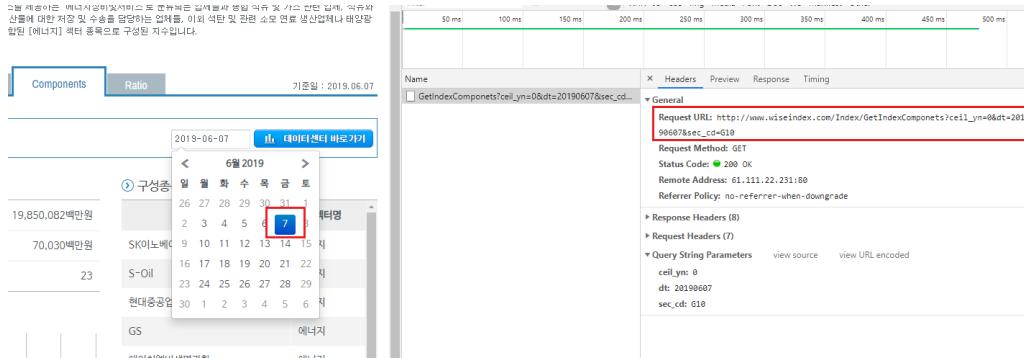


Figure 5.7: WICS 페이지 개발자도구 화면

일자를 선택하면 Network 탭의 **GetIndexComponents** 항목을 통해 데이터 전송 과정이 나타나며, Request URL의 주소를 살펴보면 다음과 같습니다.

1. `http://www.wiseindex.com/Index/GetIndexComponents`: 데이터를 요청하는 url입니다.
2. `ceil_yn = 0`: 실링여부를 나타내며, 0일 경우 비실링을 의미합니다.
3. `dt=20190607`: 조회일자를 나타냅니다.
4. `sec_cd=G10`: 섹터 코드를 나타냅니다.

이번엔 위 주소의 페이지를 열어보도록 하겠습니다.



Figure 5.8: WICS 데이터 페이지

글자들은 페이지에 출력된 내용이지만 매우 특이한 형태로 구성되어 있으며, 이는 JSON 형식의 데이터입니다. 기존에 우리가 살펴보았던 대부분의 웹페이지는 XML 형식으로 표현되었으며, 이는 문법이 복잡하고 엄격한 표현규칙으로 인해 데이터의 용량이 커진다는 단점이 있습니다. 반면 JSON 형식은 문법이 단순하여 데이터의 용량이 작아, 빠른 속도로 데이터를 교환할 수 있습니다. R에서는

jsonlite 패키지의 `fromJSON()` 함수를 사용하여 매우 손쉽게 해당 형태의 데이터를 크롤링 할 수 있습니다.

```
library(jsonlite)

url = paste0(
  'http://www.wiseindex.com/Index/GetIndexComponents',
  '?ceil_yn=0&dt=20190607&sec_cd=G10')
data = fromJSON(url)

lapply(data, print(head))

## function (x, ...)
## UseMethod("head")
## <bytecode: 0x0000000014d50da0>
## <environment: namespace:utils>

## $info
## $info$TRD_DT
## [1] "/Date(1559833200000)/"
##
## $info$MKT_VAL
## [1] 19850082
##
## $info$TRD_AMT
## [1] 70030
##
## $info$CNT
## [1] 23
##
## $list
##   IDX_CD   IDX_NM_KOR ALL_MKT_VAL CMP_CD
## 1     G10    WICS 에너지      19850082 096770
## 2     G10    WICS 에너지      19850082 010950
## 3     G10    WICS 에너지      19850082 267250
## 4     G10    WICS 에너지      19850082 078930
## 5     G10    WICS 에너지      19850082 067630
## 6     G10    WICS 에너지      19850082 006120
##                   CMP_KOR MKT_VAL     WGT S_WGT CAL_WGT SEC_CD
```

```

## 1 SK이노베이션 9052841 45.61 45.61      1   G10
## 2          S-Oil 3403265 17.14 62.75      1   G10
## 3 현대중공업지주 2873204 14.47 77.23      1   G10
## 4          GS 2491805 12.55 89.78      1   G10
## 5 에이치엘비생명과학 624986  3.15 92.93      1   G10
## 6 SK디스커버리 257059  1.30 94.22      1   G10
##   SEC_NM_KOR SEQ TOP60 APT_SHR_CNT
## 1   에너지    1    2   56403994
## 2   에너지    2    2   41655633
## 3   에너지    3    2   9283372
## 4   에너지    4    2   49245150
## 5   에너지    5    2   39307272
## 6   에너지    6    2  10470820
##
## $sector
##   SEC_CD      SEC_NM_KOR SEC_RATE IDX_RATE
## 1   G25       경기관련소비재   16.05      0
## 2   G35       건강관리     9.27      0
## 3   G50       커뮤니케이션서비스  2.26      0
## 4   G40       금융        10.31      0
## 5   G10       에너지      2.37    100
## 6   G20       산업재     12.68      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/GetIndexComponents' ,  
'?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 데이터를 csv 파일로 저장해주도록 합니다.

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



# CHAPTER 6

---

## 금융 데이터 수집하기 (심화)

---

지난 장에서는 수집한 주식티커를 바탕으로 이번 장에서는 퀀트 투자의 핵심 자료인 수정주가, 재무제표 및 가치지표를 크롤링 하는 법에 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 다운로드 되지만, 국내 중소형주의 경우 주가가 없는 경우가 있습니다.

또한 단순 주가를 구할 수 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.<sup>2</sup> 해당 차트는 주가 데이터를 받아 그래프를 그려주는 형태입니다. 따라서 해당 데이터가 어디에서 오는지 알기 위해 개발자도구 화면을 이용하도록 합니다.

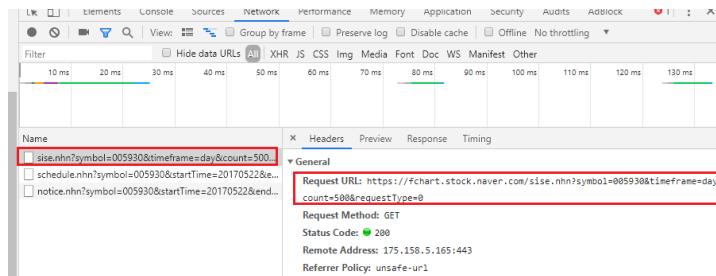


Figure 6.1: 네이버금융 차트의 통신기록

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.

```
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="KOSPI">
    <item data="20170522|45040|45380|44760|45100|352971"/>
    <item data="20170523|45400|45580|44900|44920|252141"/>
    <item data="20170524|44860|45300|44800|44880|173508"/>
    <item data="20170525|45160|45680|44800|45680|260995"/>
    <item data="20170526|45600|46460|45540|46080|272273"/>
    <item data="20170529|46220|46400|45380|45620|174791"/>
    <item data="20170530|45820|45660|44480|44640|248672"/>
    <item data="20170531|44580|45020|44400|44700|373382"/>
    <item data="20170601|44860|44900|44400|44680|195070"/>
    <item data="20170602|45060|45060|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|219066"/>
    <item data="20170613|45140|45820|45140|45400|172499"/>
    <item data="20170614|45800|46060|45240|45380|203334"/>
    <item data="20170615|45680|45920|45180|45680|193140"/>
```

Figure 6.2: 주가 데이터 페이지

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

url에서 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.

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

<sup>2</sup>플래쉬가 차단되어 화면이 나오지 않는 경우, 주소창의 왼쪽 상단에 위치한 자물쇠 버튼을 클릭한 다음, Flash를 허용으로 바꾼 후 새로고침을 누르면 차트가 나오게 됩니다.

```

library(stringr)

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

## [1] 5930

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

```

먼저 저장해두었던 csv 파일을 불러오도록 합니다. 종목코드를 살펴보면 005930이어야 할 삼성전자의 티커가 5930으로 입력되어 있습니다. 이는 파일을 불러오는 과정에서 0으로 시작하는 숫자들이 지워졌기 때문입니다. stringr 패키지의 str\_pad() 함수를 사용해 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]
## 2019-07-07    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. name에 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] "20170620|47240|48140|47220|48140|300900"
## [2] "20170621|47740|48120|47480|47480|199473"
## [3] "20170622|47960|48080|47720|47960|229116"
## [4] "20170623|47600|47780|47420|47620|190302"
## [5] "20170626|47520|48360|47520|48280|171056"
## [6] "20170627|48220|48400|47900|48300|192335"
```

1. paste0() 함수를 이용해 원하는 종목의 url을 생성해 줍니다. url 중 티커에 해당하는 6자리 부분만 위에서 입력한 name으로 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 데이터가 추출됩니다. 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
```

```
##   `20170620` `47240` `48140` `47220` `48140_1` `300900`
##   <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 20170621  47740    48120    47480    47480   199473
## 2 20170622  47960    48080    47720    47960   229116
## 3 20170623  47600    47780    47420    47620   190302
## 4 20170626  47520    48360    47520    48280   171056
## 5 20170627  48220    48400    47900    48300   192335
## 6 20170628  47600    48000    47560    47700   191450
```

`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 클랜징을 해주도록 합니다.

```
library(lubridate)
library(timetk)

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

print(tail(price))
```

```
##           Price
## 2019-06-28 47000
## 2019-07-01 46600
## 2019-07-02 46250
## 2019-07-03 45400
## 2019-07-04 46000
## 2019-07-05 45650
```

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 이용하면 yyyyymmdd 형태를 yyyy-mm-dd로 변경해주며, 데이터 형태 또한 Date 타입으로 변경됩니다.

5. timetk 패키지의 `tk_xts()` 함수를 이용해 시계열 형태로 변경해주며, 인덱스는 Date 열을 설정해줍니다. 형태 변경 후 해당 열은 자동으로 삭제됩니다.

데이터를 확인해보면 우리에게 필요한 형태로 정리가 되었습니다.

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

마지막으로 해당 데이터를 data 폴더의 KOR\_price 폴더 내에 `티커_price.csv` 이름으로 저장해주도록 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)
library(readr)

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. 마지막으로 무한크롤링을 방지하기 위해, 한번의 루프가 끝날 때마다 2초의 타임슬립을 적용하였습니다.

위의 코드가 모두 돌아가는데는 수시간이 걸리며, 작업이 끝난 후 data/KOR\_price 폴더를 확인해보면 전 종목 주가가 csv 형태로 저장되어 있습니다.

## 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>3</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/SVO2/ASP/SVD\\_Finance.asp?pGB=1&gicode=A005930](http://comp.fnguide.com/SVO2/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")
```

---

<sup>3</sup><http://comp.fnguide.com/>

```
url = paste0('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]]  
##   IFRS(연결) 2016/12 2017/12 2018/12 2019/03  
## 1   매출액 2,018,667 2,395,754 2,437,714 523,855  
## 2   매출원가 1,202,777 1,292,907 1,323,944 327,465  
## 3 매출총이익 815,890 1,102,847 1,113,770 196,391  
##   전년동기 전년동기(%)  
## 1 605,637      -13.5  
## 2 319,095       2.6  
## 3 286,542      -31.5  
##  
## [[2]]  
##   IFRS(연결) 2018/06 2018/09 2018/12 2019/03 전년동기  
## 1   매출액 584,827 654,600 592,651 523,855 605,637  
## 2   매출원가 312,746 351,944 340,160 327,465 319,095  
## 3 매출총이익 272,081 302,656 252,491 196,391 286,542  
##   전년동기(%)  
## 1      -13.5  
## 2       2.6  
## 3      -31.5  
##  
## [[3]]  
##                                     IFRS(연결) 2016/12 2017/12  
## 1                               자산 2,621,743 3,017,521  
## 2 유동자산계산에 참여한 계정 평치기 1,414,297 1,469,825  
## 3                               재고자산 183,535 249,834  
##   2018/12 2019/03
```

```

## 1 3,393,572 3,450,679
## 2 1,746,974 1,773,885
## 3 289,847 314,560
##
## [[4]]
## IFRS(연결) 2018/06 2018/09
## 1 자산 3,186,884 3,371,958
## 2 유동자산계산에 참여한 계정 평치기 1,569,768 1,762,820
## 3 재고자산 273,588 282,428
## 2018/12 2019/03
## 1 3,393,572 3,450,679
## 2 1,746,974 1,773,885
## 3 289,847 314,560
##
## [[5]]
## IFRS(연결) 2016/12 2017/12 2018/12
## 1 영업활동으로인한현금흐름 473,856 621,620 670,319
## 2 당기순손익 227,261 421,867 443,449
## 3 법인세비용차감전계속사업이익
## 2019/03
## 1 52,443
## 2 50,436
## 3
##
## [[6]]
## IFRS(연결) 2018/06 2018/09 2018/12
## 1 영업활동으로인한현금흐름 134,378 155,497 224,281
## 2 당기순손익 110,434 131,507 84,622
## 3 법인세비용차감전계속사업이익
## 2019/03
## 1 52,443
## 2 50,436
## 3

```

- 먼저 data 폴더 내에 KOR\_fs 폴더를 생성해줍니다.
- Sys.setlocale() 함수를 통해 로케일 언어를 영어로 설정 해줍니다.
- url을 입력한 후, GET() 함수를 통해 페이지 내용을 받아옵니다.
- read\_html() 함수를 통해 html 내용을 읽어오며, html\_table() 함수를 통해 테이블 내용만을 추출합니다.
- 로케일 언어를 다시 한글로 설정 해줍니다.

위의 과정을 거치면 `data` 변수에는 총 리스트 형태로 총 6개의 테이블이 들어오게 되며, 그 내용은 표 6.1와 같습니다.

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)

이 중 연간 기준 재무제표에 해당하는 첫번째, 세번째, 다섯번째 테이블을 선택합니다.

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

print(names(data_IS))

## [1] "IFRS(연결)"    "2016/12"      "2017/12"
## [4] "2018/12"        "2019/03"      "전년동기"
## [7] "전년동기(%)"

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

포괄손익계산서 테이블 (`data_IS`)에는 전년동기, 전년동기(%) 열이 존재하며, 통일성을 위해 이를 삭제해줍니다. 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']
```

- 먼저 `rbind()` 함수를 이용하여 세 테이블을 행으로 묶은 뒤 `data_fs`에 저장합니다.
- 첫번째 열인 계정명에는 계산에 참여한 계정 펼치기라는 글자가 들어간 항목이 존재합니다. 이는 페이지 내에서 펼치기 역할을 하는 (+) 항목에 해당하며, `gsub()` 함수를 이용해 해당 글자를 삭제해 줍니다.
- 중복되는 계정명이 다수 존재하며, 이는 대부분 불필요한 항목입니다. `!duplicated()` 함수를 사용해 중복되지 않는 계정명만을 선택해 줍니다.
- 행이름을 초기화 한 후, 첫번째 열의 계정명을 행이름으로 변경합니다. 그 후 첫번째 열은 삭제해주도록 합니다.
- 간혹 12월 결산법인이 아닌 종목, 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

```
sapply(data_fs, typeof)
```

	2016/12	2017/12	2018/12
##	"character"	"character"	"character"

데이터를 확인해보면 연간 기준 재무제표가 정리되었으며, 문자형 데이터이므로 숫자형으로 변경해주도록 합니다.

```
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

```
sapply(data_fs, typeof)
```

```
## X2016.12 X2017.12 X2018.12
## "double" "double" "double"
```

1. `sapply()` 함수를 이용해 각 열에 stringr 패키지의 `str_replace_all()` 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.

Table 6.2: 가치지표의 종류

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

## 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.

```
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
```

- 먼저 data 폴더 내에 KOR\_value 폴더를 생성해줍니다.
- 분모에 해당하는 항목을 저장한 후, `match()` 함수를 이용하여 해당 항목이 위치하는 지점을 찾으며, `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/SVO2/ASP/SVD\\_main.asp?pGB=1&gicode=A005930](http://comp.fnguide.com/SVO2/ASP/SVD_main.asp?pGB=1&gicode=A005930)

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

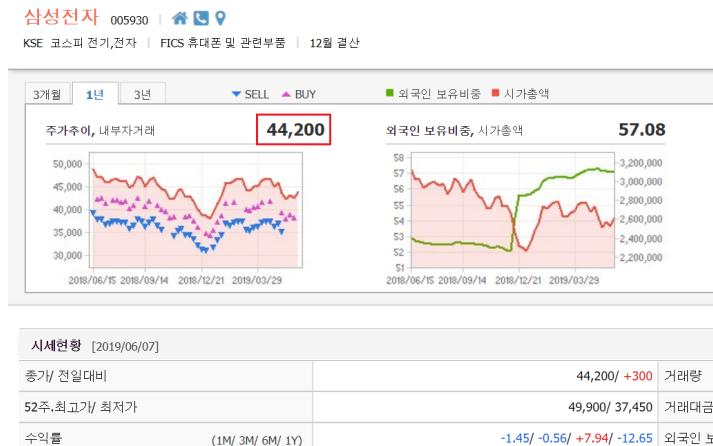


Figure 6.3: Company Guide 스냅샷 화면

주가추이 부분에 우리가 원하는 현재 주가가 있으며, 해당 데이터의 Xpath는 다음과 같습니다.

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

위에서 구한 주가의 Xpath를 이용하여 해당 데이터를 크롤링하도록 하겠습니다.

```
library(readr)

url =
  paste0('http://comp.fnguide.com/SVO2/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] 45650

```

- 먼저 url을 입력한 후, `GET()` 함수를 이용하여 데이터를 불러옵니다.
- `read_html()` 함수를 이용해 html 데이터를 불러온 후, `html_node()` 함수 내에 위에서 구한 Xpath를 입력하여, 해당 지점의 데이터를 추출합니다.
- `html_text()` 함수를 통해 텍스트 데이터만을 추출하며, `readr` 패키지의 `parse_number()` 함수를 적용합니다. 해당 함수는 문자형 데이터에서 콤마와 같은 불필요한 문자를 제거한 후, 숫자형 데이터로 변경해줍니다.

가치지표를 계산하기 위해서는 발행주식수 역시 필요합니다. 예를 들어 PER를 계산하는 방법은 다음과 같습니다.

$$PER = Price/EPS = \text{주가}/\text{주당순이익}$$

주당순이익의 경우 순이익을 전체 주식수로 나눈 값이므로, 해당 값의 계산을 위해 전체 주식수를 구해야 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 발행주식수가 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 저장이 됩니다.
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
## 6.209 1.100 4.066 1.118
```

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 변경해주도록 합니다.

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

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

data 폴더의 KOR\_value 폴더 내에 티커\_value.csv 이름으로 저장해주도록 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()
  name = KOR_ticker$'종목코드'[i]

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

---

<sup>4</sup> 분모에 사용되는 재무데이터의 구체적인 항목과 발행주식수를 계산하는 방법의 차이로 인해 여러 업체에서 제공하는 가치지표와 다소 차이가 발생할 수 있습니다.

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

# url 생성
url = paste0(
  'http://comp.fnguide.com/SVO2/ASP/',
  'SVD_Finance.asp?pGB=1&gicode=A',
  name)

# 이 후 과정은 위와 동일함

# 데이터 다운로드 후 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 클랜징
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 csv 형태로 저장되어 있게 됩니다.

## 6.3 야후 파이낸스 데이터 구하기

크롤링을 이용하여 데이터를 수집할 경우 주의해야 할 점은 웹페이지 구조가 변경되는 경우이며, 이런 경우에는 변경된 구조에 맞게 코드를 다시 짜야합니다. 그러나 최악의 경우는 웹사이트가 폐쇄되는 경우입니다. 실제로 투자자들이 많이 사용하던 구글 파이낸스가 2018년 서비스를 중단함에 따라 이를 이용하던 많은 사용자들이 혼란을 겪기도 했습니다.

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

### 6.3.1 재무제표 다운로드

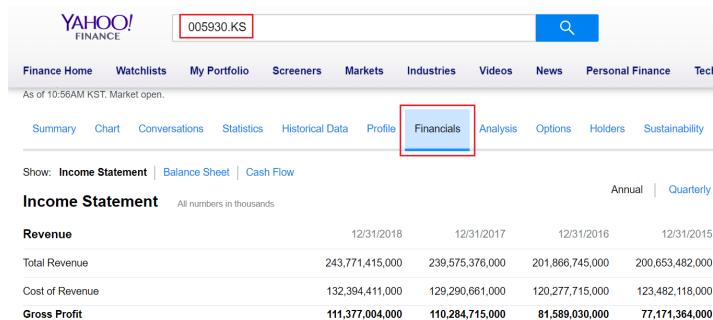


Figure 6.4: 야후 파이낸스 재무제표

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 표 6.3와 같습니다.

Table 6.3: 야후 파이낸스 재무제표 url

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

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 = paste0(
  'https://finance.yahoo.com/quote/005930.KS/financials?p=',
  '005930.KS')

url_BS = paste0(
  'https://finance.yahoo.com/quote/005930.KS/balance-sheet?p=',
  '005930.KS')

url_CF = paste0(
  '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
## 1	Revenue	12/31/2018
## 2	Total Revenue	243,771,415,000

```

## 3      Cost of Revenue    132,394,411,000
## 4          Gross Profit   111,377,004,000
## 5  Operating Expenses  Operating Expenses
## 6 Research Development   18,354,080,000
##                   X3           X4
## 1            12/31/2017     12/31/2016
## 2      239,575,376,000    201,866,745,000
## 3      129,290,661,000    120,277,715,000
## 4      110,284,715,000    81,589,030,000
## 5  Operating Expenses  Operating Expenses
## 6      16,355,612,000    14,111,381,000
##                   X5
## 1            12/31/2015
## 2      200,653,482,000
## 3      123,482,118,000
## 4      77,171,364,000
## 5  Operating Expenses
## 6      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 클랜징 작업을 해주도록 하며, 그 과정은 앞선 예시와 거의 동일합니다.

```

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
## Total Revenue	243771415000
## Cost of Revenue	132394411000
## Gross Profit	111377004000
## Operating Expenses	NA
## Research Development	18354080000
## Selling General and Administrative	32688565000
##	X12.31.2017
## Total Revenue	239575376000
## Cost of Revenue	129290661000
## Gross Profit	110284715000
## Operating Expenses	NA
## Research Development	16355612000
## Selling General and Administrative	38947445000
##	X12.31.2016
## Total Revenue	201866745000
## Cost of Revenue	120277715000
## Gross Profit	81589030000
## Operating Expenses	NA
## Research Development	14111381000
## Selling General and Administrative	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.
2. 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 12인 열, 즉 12월 결산 데이터만을 선택해 줍니다.
5. `sapply()` 함수를 이용해 각 열에 `stringr` 패키지의 `str_replace_all()` 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 동일합니다.

```
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]

print(value_index)

## [1] 43890877000 240068993000 67031863000 243771415000
```

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

다음으로 현재 주가 및 상장주식수는 Statistics 항목에서 구할 수 있습니다. 먼저 현재 주가를 크롤링 하는 방법입니다.

```
url = paste0(
  'https://finance.yahoo.com/quote/005930.KS/',
  'key-statistics?p=005930.KS')

data = GET(url)

price = read_html(data) %>%
```

```

html_node(
  xpath =
  '//*[@id="quote-header-info"]/div[3]/div/div/span[1]') %>%
html_text() %>%
parse_number()

print(price)

## [1] 45650

```

1. 해당 페이지 url을 저장 후, `GET()` 함수를 통해 페이지 정보를 받습니다.
2. `read_html()` 함수를 이용해 html 데이터를 받고, `html_node()` 함수와 Xpath를 이용해 현재 주가에 해당하는 부분을 추출합니다. 주가의 경우 페이지 상단에서 확인할 수 있습니다.
3. `html_text()` 함수를 이용해 텍스트 데이터만을 추출한 후, `parse_number()` 함수를 통해 콤마 삭제 및 숫자형태로 변경합니다.

이처럼 주가의 경우 상대적으로 쉽게 데이터를 구할 수 있습니다. 다음은 상장주식수 데이터를 크롤링하는 방법입니다.

```

share_xpath =
paste0('//*[ @id="Col1-0-KeyStatistics-Proxy"]/section',
'/div[2]/div[2]/div/div[2]/table/tbody/tr[3]/td[2]')

share_yahoo = read_html(data) %>%
html_node(xpath = share_xpath) %>%
html_text()

print(share_yahoo)

## [1] "5.97B"

```

상장주식수의 경우 **Shares Outstanding** 부분에서 찾을 수 있습니다. 해당 지점의 Xpath를 이용해 데이터를 찾으면 5.97B가 추출됩니다. 이 중 숫자 뒤 알파벳 부분은 단위에 해당하며, 각 문자 별 단위는 다음과 같습니다.

따라서 알파벳을 해당하는 숫자로 변경한 뒤, 이를 앞의 숫자에 곱해주어야 제대로 된 상장주식수가 계산됩니다.

Table 6.4: 발행주식수 단위

알파벳	단위	숫자
M	백만 (Million)	1,000,000
B	십억 (Billion)	1,000,000,000
T	일조 (Triliion)	1,000,000,000,000

```
library(stringr)

share_unit = str_match(share_yahoo, '[a-zA-Z]')
print(share_unit)
```

```
##      [,1]
## [1,] "B"

share_multiplier = switch(share_unit,
  'M' = { 1000000 },
  'B' = { 1000000000 },
  'T' = { 1000000000000 })
print(share_multiplier)
```

```
## [1] 1000000000
```

먼저 `str_match()` 함수 내에서 정규표현식을 사용하여 알파벳을 추출한 후, 이를 `share_unit`에 저장합니다. 그 후, `switch` 구문을 이용하여 알파벳에 해당하는 숫자를 `share_multiplier`에 저장해줍니다.

```
share_yahoo = share_yahoo %>%
  str_match('[0-9.0-9]*') %>% as.numeric()
share_yahoo = share_yahoo * share_multiplier

print(share_yahoo)
```

```
## [1] 5970000000
```

숫자 부분과 위에서 구한 단위 부분을 곱하여 최종 발행주식수를 구하도록 하겠습니다. 먼저 `str_match()` 함수 내에 정규표현식을 이용하여 숫자에 해당하는 부분만 추출한 후, `as.numeric()`을 통해 숫자 형태로 변경합니다. 그 후 단위에 해당하는 숫자를 곱해 최종값을 구하도록 합니다.

위에서 구한 재무데이터, 현재주가, 발행주식수를 이용하여 가치지표를 계산하도록 하겠습니다.

```
data_value_yahoo = price / (value_index * 1000 / share yahoo)
names(data_value_yahoo) = c('PER', 'PBR', 'PCR', 'PSR')
```

```
data_value_yahoo[data_value_yahoo < 0] = NA
print(data_value_yahoo)
```

```
##    PER    PBR    PCR    PSR
## 6.209 1.135 4.066 1.118
```

분자에는 주가를, 분모에는 재무 데이터를 보통주 발행주식수로 나눈 값을 입력합니다. 야후 파이낸스의 재무 데이터는 천원 단위이므로, 둘 간의 단위를 동일하게 맞춰주기 위해 분모에 천을 곱해 줍니다. 또한 가치지표가 음수인 경우는 NA로 변경해주도록 합니다.

결과를 확인해보면 4가지 가치지표가 잘 계산되었습니다. Company Guide에서 구한 값 6.21, 1.1, 4.07, 1.12 와는 재무데이터의 차이로 인해 미세한 차이가 있지만, 이는 무시해도 될 정도입니다.

해당 방법 또한 url의 티커 부분만 변경하면 전 종목의 재무제표와 가치지표 데이터를 다운로드 받을 수 있습니다. 그러나 주의해야 할 점은 코스피 종목은 끝이 .KS, 코스닥 종목은 끝이 .KQ가 되어야 합니다. 자세한 코드는 생략하도록 합니다.



# CHAPTER 7

---

## 데이터 정리하기

---

앞장에서는 API와 크롤링을 통하여 주가, 재무제표, 가치지표를 수집하는 방법에 대해 배웠습니다. 이번 장에서는 각각 csv 파일로 저장된 데이터들을 하나로 합친 후 저장하는 과정을 살펴보도록 하겠습니다.

### 7.1 주가 정리하기

주가의 경우 **data/KOR\_price** 폴더 내에 **티커\_price.csv** 파일로 저장되어 있습니다. 해당 파일들을 불러온 후 데이터를 묶는 작업을 통해 하나의 파일로 합치는 방법에 대해 알아보도록 하겠습니다.

```
library(stringr)
library(xts)
library(magrittr)

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')

price_list = list()
```

```

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

  name = KOR_ticker[i, '종목코드']
  price_list[[i]] =
    read.csv(paste0('data/KOR_price/', name,
                    '_price.csv'), row.names = 1) %>%
  as.xts()

}

price_list = do.call(cbind, price_list) %>% na.locf()
colnames(price_list) = KOR_ticker$'종목코드'

head(price_list[, 1:5])

```

	##	005930	000660	005380	068270	051910
##	2017-06-21	47480	64800	160500	112857	280000
##	2017-06-22	47960	65000	161500	110896	287000
##	2017-06-23	47620	65000	164000	111092	283500
##	2017-06-26	48280	67500	164000	111778	282500
##	2017-06-27	48300	69200	160500	112857	283000
##	2017-06-28	47700	67200	160000	111778	280500

- 먼저 티커가 저장된 csv 파일을 불러온 후, 티커를 6자리로 맞춰주도록 합니다.
- 빈 리스트인 price\_list를 생성합니다.
- for loop 구문을 이용해 종목별 가격 데이터를 불러온 후, as.xts()를 통해 시계열 형태로 데이터를 변경한 후 리스트에 저장합니다.
- do.call() 함수를 통해 리스트를 열의 형태로 묶습니다.
- 간혹 결측치가 발생할 수 있으므로, na.locf() 함수를 통해 결측치의 경우 전일 데이터를 사용하도록 합니다.
- 행 이름을 각 종목의 티커로 변경해 주도록 합니다.

해당 작업을 통해 개별 csv 파일로 흩어져 있던 가격 데이터가 하나의 데이터로 묶이게 됩니다.

```
write.csv(data.frame(price_list), 'data/KOR_price.csv')
```

마지막으로 해당 데이터를 data 폴더에 **KOR\_price.csv** 파일로 저장해주도록 합니다. 시계열 형태 그대로 저장할 경우 인덱스가 삭제되므로, 데이터프레임 형태로 변경한 후 저장하도록 합니다.

## 7.2 재무제표 정리하기

재무제표의 경우 **data/KOR\_fs** 폴더 내 **티커\_fs.csv** 파일로 저장되어 있습니다. 주가의 경우 하나의 열로 이루어져 있어 데이터를 정리하는 것이 간단하였지만, 재무제표의 경우 각 종목 별 재무 항목이 모두 다르다는 번거로움이 있습니다.

```
library(stringr)
library(magrittr)
library(dplyr)

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')

data_fs = list()

for (i in 1 : nrow(KOR_ticker)){
  name = KOR_ticker[i, '종목코드']
  data_fs[[i]] = read.csv(paste0('data/KOR_fs/', name,
                                 '_fs.csv'), row.names = 1)
}
```

위와 동일하게 티커 데이터를 읽어오며, 이를 바탕으로 종목별 재무제표 데이터를 읽어온 후 리스트에 저장합니다.

```
fs_item = data_fs[[1]] %>% rownames()
length(fs_item)
```

```
## [1] 236
```

```
print(head(fs_item))
```

```
## [1] "매출액"           "매출원가"
## [3] "매출총이익"       "판매비와관리비"
## [5] "인건비"           "유무형자산상각비"
```

다음으로 재무제표 항목의 기준을 정해줄 필요가 있습니다. 재무제표 작성 항목의 경우 각 업종별로 상이하므로, 이를 모두 고려할 경우 지나치게 데이터가 커지게

됩니다. 또한 퀸트 투자에는 일반적이고 공통적인 항목을 주로 사용하므로 대표적인 재무 항목을 정해 이를 기준으로 데이터를 정리하여도 충분합니다.

따라서 기준점으로 첫번째 리스트, 즉 삼성전자의 재무 항목을 선택하도록 하며, 총 236개 재무 항목이 존재합니다. 해당 기준을 바탕으로 재무제표 데이터를 정리하도록 하며, 전체 항목에 대한 정리 이전에 간단한 예시로 첫번째 항목인 **매출액** 기준 데이터 정리를 살펴보도록 하겠습니다.

```
select_fs = lapply(data_fs, function(x) {
  # 해당 항목이 있을시 데이터를 선택
  if ( '매출액' %in% rownames(x) ) {
    x[which(rownames(x) == '매출액'), ]
  }

  # 해당 항목이 존재하지 않을 시, NA로 된 데이터프레임 생성
} else {
  data.frame(NA)
}
})

select_fs = bind_rows(select_fs)

print(head(select_fs))
```

```
##   X2016.12 X2017.12 X2018.12 NA. X2015.12
## 1 2018667  2395754  2437714  NA     NA
## 2 171980   301094   404451  NA     NA
## 3 936490   963761   968126  NA     NA
## 4   6706    9491    9821  NA     NA
## 5 206593   256980   281830  NA     NA
## 6 382617   351446   351492  NA     NA
```

먼저 `lapply()` 함수를 이용하여 모든 재무데이터가 들어있는 `data_fs` 데이터를 대상으로 함수를 적용합니다. `%in%` 함수를 통해 만일 매출액이라는 항목이 행이름에 존재할 시, 해당 부분의 데이터를 `select_fs` 리스트에 저장하며, 그렇지 않을 경우, 즉 해당 항목이 존재하지 않을 경우 NA로 이루어진 데이터프레임을 저장합니다.

그 후, `dplyr` 패키지의 `bind_rows()` 함수를 이용하여 리스트 내 데이터들을 행으로 묶어주도록 합니다. `rbind()`의 경우 리스트 형태를 테이블로 묶기 위해서는 모든 데이터의 열갯수가 동일해야 하는 반면, `bind_rows()`의 경우 갯수가 다를 경우 나머지 부분을 NA로 처리해 합쳐주는 장점이 있습니다.

합쳐진 데이터를 살펴보면, 먼저 열이름이 . 혹은 NA.인 부분이 존재합니다. 이는 매출액 항목이 없는 종목의 경우 NA 데이터프레임을 저장하여 생긴 결과입니다. 또한 연도가 순서대로 저장되지 않은 경우가 있습니다. 이 두 가지를 고려하여 데이터를 클랜징해주도록 합니다.

```
select_fs = select_fs[!colnames(select_fs) %in%
                      c('. ', 'NA.')]
select_fs = select_fs[, order(names(select_fs))]
rownames(select_fs) = KOR_ticker[, '종목코드']

print(head(select_fs))
```

	X2015.12	X2016.12	X2017.12	X2018.12
## 005930	NA	2018667	2395754	2437714
## 000660	NA	171980	301094	404451
## 005380	NA	936490	963761	968126
## 068270	NA	6706	9491	9821
## 051910	NA	206593	256980	281830
## 012330	NA	382617	351446	351492

1. !와 %in% 함수를 이용하여, 열이름에 . 혹은 NA.가 들어가지 않은 열만을 선택해주세요도록 합니다.
2. order() 함수를 이용해 열이름의 연도별 순서를 구한 후, 이를 바탕으로 열을 다시 정리하도록 합니다.
3. 행이름을 티커들로 변경합니다.

해당 과정을 통해 전 종목의 매출액 데이터가 연도별로 정리되었습니다. for loop 구문을 이용하여 모든 재무항목에 대해 정리하는 법은 다음과 같습니다.

```
fs_list = list()

for (i in 1 : length(fs_item)) {
  select_fs = lapply(data_fs, function(x) {
    # 해당 항목이 있을시 데이터를 선택
    if ( fs_item[i] %in% rownames(x) ) {
      x[which(rownames(x) == fs_item[i]), ]
    }
    # 해당 항목이 존재하지 않을 시, NA로 된 데이터프레임 생성
  } else {
    data.frame(NA)
```

```

    }
})

# 리스트 데이터를 행으로 묶어줌
select_fs = bind_rows(select_fs)

# 열이름이 '.. 혹은 'NA.'인 지점은 삭제 (NA 데이터)
select_fs = select_fs[!colnames(select_fs) %in%
                      c('..', 'NA.')]

# 연도 순별로 정리
select_fs = select_fs[, order(names(select_fs))]

# 행이름을 티커로 변경
rownames(select_fs) = KOR_ticker[, '종목코드']

# 리스트에 최종 저장
fs_list[[i]] = select_fs

}

# 리스트 이름을 재무 항목으로 변경
names(fs_list) = fs_item

```

위의 과정을 거치면 fs\_list에 총 236개 리스트가 생성되며, 각 리스트에는 해당 재무 항목에 대한 전 종목의 연도별 데이터가 정리되어 있습니다.

```
saveRDS(fs_list, 'data/KOR_fs.Rds')
```

마지막으로 해당 데이터를 data 폴더 내에 저장해주도록 하며, 리스트 형태 그대로 저장하기 위해 saveRDS() 함수를 이용하여 **KOR\_fs.Rds**로 저장해주도록 합니다.

RDS 형식의 경우, 파일을 더블 클릭한 후 연결 프로그램을 RStudio로 설정해 파일을 불러올 수 있습니다. 혹은 readRDS() 함수를 이용하여 파일을 읽어올 수도 있습니다.

## 7.3 가치지표 정리하기

재무제표의 경우 **data/KOR\_value** 폴더 내 **티커\_value.csv** 파일로 저장되어 있으며, 재무제표를 정리하는 방법과 거의 동일합니다.

```
library(stringr)
library(magrittr)
library(dplyr)

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')

data_value = list()

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

  name = KOR_ticker[i, '종목코드']
  data_value[[i]] =
    read.csv(paste0('data/KOR_value/', name,
                    '_value.csv'), row.names = 1) %>%
    t() %>% data.frame()

}
```

먼저 티커에 해당하는 파일을 불러온 후, for loop 구문을 통해 가치지표 데이터를 **data\_value** 리스트에 저장해줍니다. 단, csv 내에 데이터가 테이블 7.1와 같이 행의 형태로 저장되어 있으므로, **t()** 함수를 이용해 열의 형태로 바꿔주도록 하며, 데이터프레임 형태로 저장해줍니다.

Table 7.1: 가치지표의 저장 예시

value	x
PER	Number 1
PBR	Number 2
PCR	Number 3
PSR	Number 4

```
data_value = bind_rows(data_value)
print(head(data_value))
```

```
##          PER      PBR      PCR      PSR X1
## 1    6.209  1.1000  4.066  1.1179 NA
## 2    3.204  1.0628  2.240  1.2312 NA
## 3   20.048  0.4091  8.032  0.3123 NA
## 4 101.957 10.1384 69.857 27.1789 NA
## 5  16.994  1.4447 11.776  0.8879 NA
## 6  11.782  0.7248 13.822  0.6331 NA
```

`bind_rows()` 함수를 이용하여 리스트 내 데이터들을 행으로 뷄어준 후 데이터를 확인해보면 PER, PBR, PCR, PSR 열 외에 불필요한 NA로 이루어진 열이 존재합니다. 해당 열을 삭제한 후 정리작업을 해주도록 하겠습니다.

```
data_value = data_value[colnames(data_value) %in%
                        c('PER', 'PBR', 'PCR', 'PSR')]
rownames(data_value) = KOR_ticker[, '종목코드']

print(head(data_value))
```

```
##          PER      PBR      PCR      PSR
## 005930  6.209  1.1000  4.066  1.1179
## 000660  3.204  1.0628  2.240  1.2312
## 005380 20.048  0.4091  8.032  0.3123
## 068270 101.957 10.1384 69.857 27.1789
## 051910  16.994  1.4447 11.776  0.8879
## 012330  11.782  0.7248 13.822  0.6331
```

```
data_value = data_value %>%
  mutate_all(list(~na_if(., Inf)))

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

- 열 이름이 가치지표에 해당하는 부분만을 선택한 후, 행이름을 티커들로 변경합니다.
- 일부 종목의 경우 재무 데이터가 0으로 표기되어 가치지표가 Inf으로 계산되는 경우가 있습니다. `mutate_all()` 내에 `na_if()` 함수를 이용하여 Inf 데이터를 NA로 변경해 주도록 합니다.
- data 폴더 내에 `KOR_value.csv` 파일로 저장해주세요.

# CHAPTER 8

---

## 데이터 분석 및 시각화하기

---

데이터 수집 및 정리가 끝났다면, 내가 가지고 있는 데이터가 어떠한 특성을 가지고 있는지에 대한 분석 및 시각화 과정, 즉 탐색적 데이터 분석 (**Exploratory Data Analysis**)을 할 필요가 있습니다. 해당 과정을 통해 데이터를 더 잘 이해할 수 있으며, 극단치나 결측치 등 데이터가 가지고 있는 잠재적인 문제를 발견하고 이를 어떻게 처리할지에 대해 고민할 수 있습니다.

해당 장에서는 `dplyr` 패키지를 이용한 데이터 분석과 `ggplot` 패키지를 이용한 데이터 시각화에 대해 알아보도록 하겠습니다.

### 8.1 종목정보 데이터 분석

먼저 거래소를 통해 수집한 산업별 현황과 개별지표를 정리한 파일, WICS 기준 섹터지표를 정리한 파일을 통해 국내 상장종목의 데이터를 분석해보도록 하겠습니다.

```
library(stringr)

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

```
KOR_ticker$'종목코드' =
  str_pad(KOR_ticker$'종목코드', 6, 'left', 0)
KOR_sector$'CMP_CD' =
  str_pad(KOR_sector$'CMP_CD', 6, 'left', 0)
```

먼저 각 파일을 불러온 후, 티커에 해당하는 종목코드와 CMP\_CD 열을 6자리 숫자로 만들어 주도록 합니다.

이제 `dplyr` 패키지의 여러 함수들을 이용하여 데이터를 분석해보도록 하겠습니다. 해당 패키지는 데이터 처리에 특화된 패키지이며, C++로 작성되어 매우 빠른 처리속도를 자랑합니다. 또한 문법이 SQL과 매우 비슷하여, 함수들의 내용을 직관적으로 이해할 수 있습니다.

### 8.1.1 \*\_join: 데이터 합치기

두 테이블을 하나로 합치기 위해 `*_join()` 함수를 이용하도록 합니다. 해당 함수는 기존에 살펴본 `merge()` 함수와 동일하며, 합치는 방법은 그림 8.1과 표 8.1과 같이 크게 4가지 종류가 있습니다.

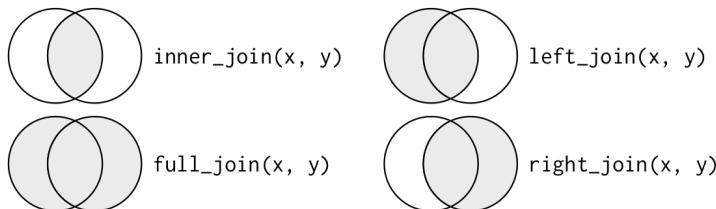


Figure 8.1: join 함수의 종류

Table 8.1: join 함수의 종류

함수	내용
inner_join()	교집합
full_join()	합집합
left_join()	좌측 기준
right_join()	우측 기준

이 중 거래소 티커 기준으로 데이터를 맞추기 위해 `left_join()` 함수를 사용하여 두 데이터를 합치도록 하겠습니다.

```
library(dplyr)

data_market = left_join(KOR_ticker, KOR_sector,
                        by = c('종목코드' = 'CMP_CD',
                               '종목명' = 'CMP_KOR'))
```

```
head(data_market)
```

	종목코드	종목명	시장구분	산업분류	현재가.종가.
## 1	005930	삼성전자	코스피	전기전자	45400
## 2	000660	SK하이닉스	코스피	전기전자	69100
## 3	005380	현대차	코스피	운수장비	136000
## 4	068270	셀트리온	코스피	의약품	206000
## 5	051910	LG화학	코스피	화학	355500
## 6	012330	현대모비스	코스피	운수장비	229500
##	전일대비	시가총액.원.		일자 관리여부	종가
## 1	-850	2.710e+14	2019-07-03		- 45400
## 2	-2300	5.030e+13	2019-07-03		- 69100
## 3	-1000	2.906e+13	2019-07-03		- 136000
## 4	1000	2.644e+13	2019-07-03		- 206000
## 5	7000	2.510e+13	2019-07-03		- 355500
## 6	-2500	2.187e+13	2019-07-03		- 229500
##	EPS	PER	BPS	PBR	주당배당금 배당수익률
## 1	6,461	7.03	35,342	1.28	1416 3.12
## 2	22,255	3.1	64,348	1.07	1500 2.17
## 3	5,632	24.15	245,447	0.55	4000 2.94
## 4	2,063	99.85	19,766	10.42	0 0.00
## 5	19,217	18.5	218,227	1.63	6000 1.69
## 6	19,944	11.51	314,650	0.73	4000 1.74
##	개시물..일련번호	총카운트	IDX_CD		IDX_NM_KOR
## 1		2165	NA	G45	WICS IT
## 2		1885	NA	G45	WICS IT
## 3		2159	NA	G25 WICS 경기관련소비재	
## 4		2049	NA	G35	WICS 건강관리
## 5		2041	NA	G15	WICS 소재
## 6		2169	NA	G25 WICS 경기관련소비재	
##	ALL_MKT_VAL	MKT_VAL	WGT	S_WGT	CAL_WGT SEC_CD
## 1	376270891	208452867	55.40	55.40	1 G45
## 2	376270891	35232402	9.36	64.76	1 G45
## 3	141374708	20042076	14.18	14.18	1 G25

```

## 4     81157991 16895164 20.82 20.82      1   G35
## 5     71815100 15165000 21.12 43.84      1   G15
## 6    141374708 14257881 10.09 24.26      1   G25
##           SEC_NM_KOR SEQ TOP60 APT_SHR_CNT
## 1             IT    1    2  4716128215
## 2             IT    2    2  538721750
## 3 경기관련소비재    1    12  143157685
## 4         건강관리    1    21  85980477
## 5         소재    2     6  45885023
## 6 경기관련소비재    2    12  64808552

```

`left_join()` 함수를 이용해 KOR\_ticker와 KOR\_sector 데이터를 합쳐주도록 합니다. `by` 인자는 데이터를 합치는 기준점을 의미하며, x 데이터(KOR\_ticker)의 **종목코드**와 y 데이터(KOR\_sector)의 **CMP\_CD**는 같음을, x 데이터의 **종목명**과 y 데이터의 **CMP\_KOR**는 같음을 정의합니다.

### 8.1.2 `glimpse()`: 데이터 구조 확인하기

```
glimpse(data_market)
```

```

## Observations: 2,041
## Variables: 30
## $ 종목코드          <chr> "005930", "000660", "005...
## $ 종목명            <chr> "삼성전자", "SK하이닉스", "현대차"...
## $ 시장구분          <chr> "코스피", "코스피", "코스피", "코...
## $ 산업분류          <chr> "전기전자", "전기전자", "운수장비", ...
## $ 현재가.종가.        <int> 45400, 69100, 136000, 206...
## $ 전일대비          <int> -850, -2300, -1000, 1000...
## $ 시가총액.원.        <dbl> 2.710e+14, 5.030e+13, 2.9...
## $ 일자              <chr> "2019-07-03", "2019-07...
## $ 관리여부          <chr> "-", "-", "-", "-", "-',...
## $ 종가              <int> 45400, 69100, 136000, ...
## $ EPS               <chr> "6,461", "22,255", "...
## $ PER               <chr> "7.03", "3.1", "24.1...
## $ BPS               <chr> "35,342", "64,348", ...
## $ PBR               <chr> "1.28", "1.07", "0.5...
## $ 주당배당금        <int> 1416, 1500, 4000, 0, 6000...
## $ 배당수익률        <dbl> 3.12, 2.17, 2.94, 0.00, 1...

```

```

## $ 게시물..일련번호 <int> 2165, 1885, 2159, 2049, 204...
## $ 총카운트          <int> NA, NA, NA, NA, NA, NA, ...
## $ IDX_CD            <chr> "G45", "G45", "G25", ...
## $ IDX_NM_KOR        <chr> "WICS IT", "WICS IT"...
## $ ALL_MKT_VAL       <int> 376270891, 376270891...
## $ MKT_VAL           <int> 208452867, 35232402, ...
## $ WGT               <dbl> 55.40, 9.36, 14.18, ...
## $ S_WGT              <dbl> 55.40, 64.76, 14.18, ...
## $ CAL_WGT            <int> 1, 1, 1, 1, 1, 1, 1, ...
## $ SEC_CD             <chr> "G45", "G45", "G25", ...
## $ SEC_NM_KOR         <chr> "IT", "IT", "경기관련소비재...
## $ SEQ                <int> 1, 2, 1, 1, 2, 2, 1, ...
## $ TOP60              <int> 2, 2, 12, 21, 6, 12, ...
## $ APT_SHR_CNT         <dbl> 4716128215, 53872175...

```

`glimpse()` 함수는 데이터 내용, 구조, 형식을 확인하는 함수입니다. 기본 함수인 `str()`과 그 역할은 비슷하지만, tidy 형태로 결과물이 훨씬 깔끔하게 출력됩니다. 총 관측값 및 열의 갯수, 각 열의 이름과 데이터 형식, 앞부분 데이터를 확인할 수 있습니다.

### 8.1.3 `rename()`: 열 이름 바꾸기

```
head(names(data_market), 10)
```

```

## [1] "종목코드"      "종목명"        "시장구분"
## [4] "산업분류"      "현재가.종가."   "전일대비"
## [7] "시가총액.원."  "일자"          "관리여부"
## [10] "종가"

```

```
data_market = data_market %>%
  rename(`시가총액` = `시가총액.원.`)
```

```
head(names(data_market), 10)
```

```

## [1] "종목코드"      "종목명"        "시장구분"
## [4] "산업분류"      "현재가.종가."   "전일대비"
## [7] "시가총액"      "일자"          "관리여부"
## [10] "종가"

```

`rename()` 함수는 열 이름을 바꾸는 함수로써, `rename(tbl, new_name, old_name)` 형태로 입력합니다. 위의 경우 시가총액.원. 열이름이 시가총액으로 변경되었습니다.

### 8.1.4 `distinct()`: 고유한 값 확인

```
data_market %>%
  distinct(SEC_NM_KOR) %>% c()
```

```
## $SEC_NM_KOR
## [1] "IT"                  "경기관련소비재"
## [3] "건강관리"             "소재"
## [5] "금융"                 "커뮤니케이션서비스"
## [7] "산업재"               "유틸리티"
## [9] "에너지"               "필수소비재"
## [11] NA
```

`distinct()` 함수는 고유한 값을 반환하며, 기본함수 중 `unique()`과 동일한 기능을 합니다. 데이터의 섹터 정보를 확인해보면, WICS 기준 10개 섹터 및 섹터 정보가 없는 종목인 NA 값이 있습니다.

### 8.1.5 `select()`: 원하는 열만 선택

```
data_market %>%
  select(`종목명`) %>% head()
```

```
##      종목명
## 1 삼성전자
## 2 SK하이닉스
## 3 현대차
## 4 셀트리온
## 5 LG화학
## 6 현대모비스
```

```
data_market %>%
  select(`종목명`, `PBR`, `SEC_NM_KOR`) %>% head()
```

```
##          종목명    PBR      SEC_NM_KOR
## 1 삼성전자  1.28           IT
## 2 SK하이닉스 1.07           IT
## 3 현대차   0.55 경기관련소비재
## 4 셀트리온 10.42       건강관리
## 5 LG화학    1.63       소재
## 6 현대모비스 0.73 경기관련소비재
```

`select()` 함수는 원하는 열을 선택해주는 함수이며, 원하는 열 이름을 입력하면 됩니다. 하나의 열 뿐만 아니라 다수의 열을 입력할 경우 해당 열들이 선택됩니다.

```
data_market %>%
  select(starts_with('시')) %>% head()
```

```
##      시장구분 시가총액
## 1     코스피 2.710e+14
## 2     코스피 5.030e+13
## 3     코스피 2.906e+13
## 4     코스피 2.644e+13
## 5     코스피 2.510e+13
## 6     코스피 2.187e+13
```

```
data_market %>%
  select(ends_with('R')) %>% head()
```

```
##      PER    PBR      IDX_NM_KOR      SEC_NM_KOR
## 1  7.03  1.28        WICS IT            IT
## 2  3.1   1.07        WICS IT            IT
## 3 24.15  0.55 WICS 경기관련소비재 경기관련소비재
## 4 99.85 10.42        WICS 건강관리       건강관리
## 5 18.5   1.63        WICS 소재           소재
## 6 11.51  0.73 WICS 경기관련소비재 경기관련소비재
```

```
data_market %>%
  select(contains('가')) %>% head()
```

```
##   현재가.종가 시가총액 종가
## 1      45400 2.710e+14 45400
## 2      69100 5.030e+13 69100
## 3     136000 2.906e+13 136000
## 4     206000 2.644e+13 206000
## 5    355500 2.510e+13 355500
## 6    229500 2.187e+13 229500
```

해당 함수는 다양한 응용기능도 제공합니다. `starts_with()`의 경우 특정 문자로 시작하는 열들을, `ends_with()`의 경우 특정 문자로 끝나는 열들을, `contains()`의 경우 특정 문자가 포함되는 열들을 선택해 줍니다.

### 8.1.6 `mutate()`: 열 생성 및 데이터 변형

```
data_market = data_market %>%
  mutate(`PBR` = as.numeric(PBR),
        `PER` = as.numeric(PER),
        `ROE` = PBR / PER,
        `ROE` = round(ROE, 4),
        `size` = ifelse(`시가총액` >=
                         median(`시가총액`, na.rm = TRUE),
                         'big', 'small')
  )
```

```
data_market %>%
  select(`종목명`, `ROE`, `size`) %>% head()
```

```
##   종목명 ROE size
## 1 삼성전자 0.1821 big
## 2 SK하이닉스 0.3452 big
## 3 현대차 0.0228 big
## 4 셀트리온 0.1044 big
## 5 LG화학 0.0881 big
## 6 현대모비스 0.0634 big
```

`mutate()` 함수는 원하는 형태로 열을 생성하거나 변형하는 함수입니다. 위의 예제에서는 먼저 PBR과 PER열을 `as.numeric()` 함수를 통해 숫자 형태로 변경한 후, PBR을 PER로 나눈 값을 ROE 열에 생성합니다. 그 후, `round()` 함수를 통해 ROE 값을 반올림 해주며, `ifelse()` 함수를 통해 시가총액의 중앙값보다 큰 기업의 경우 big, 아닐 경우 small 임을 size 열에 저장해줍니다.

이 외에도 `mutate_*`() 계열 함수에는 `mutate_all()`, `mutate_if()`, `mutate_at()`처럼 각 상황에 맞게 쓸수있는 다양한 함수들이 존재합니다.

### 8.1.7 filter(): 조건을 충족하는 행 선택

```
data_market %>%
  select(`종목명`, `PBR`) %>%
  filter(`PBR` < 1) %>% head()
```

```
##          종목명   PBR
## 1      현대차 0.55
## 2 현대모비스 0.73
## 3      POSCO 0.49
## 4 신한지주 0.59
## 5 SK텔레콤 0.93
## 6      KB금융 0.53
```

```
data_market %>%
  select(`종목명`, `PBR`, `PER`, `ROE`) %>%
  filter(PBR < 1 & PER < 20 & ROE > 0.1 ) %>% head()
```

```
##          종목명   PBR     PER     ROE
## 1      SK텔레콤 0.93 5.85 0.1590
## 2          SK 0.97 5.66 0.1714
## 3          LG 0.74 7.19 0.1029
## 4 롯데케미칼 0.72 5.83 0.1235
## 5          GS 0.56 5.28 0.1061
## 6 대림산업 0.80 6.69 0.1196
```

`filter()` 함수는 조건을 충족하는 부분의 데이터를 반환하는 함수입니다. 첫번째 예제와 같이 PBR이 1 미만인 단일 조건을 입력할 수도 있으며, 두번째 예제와 같이 PBR 1 미만, PER 20 미만, ROE 0.1 초과 등 복수 조건을 입력할 수도 있습니다.

### 8.1.8 summarize(): 요약 통계값 계산

```
data_market %>%
  summarize(PBR_max = max(PBR, na.rm = TRUE),
            PBR_min = min(PBR, na.rm = TRUE))

##     PBR_max PBR_min
## 1      87.2    0.19
```

summarize() 혹은 summarise() 함수는 원하는 요약 통계값을 계산해 줍니다. PBR\_max은 PBR 열에서 최대값을, PBR\_min은 최소값을 계산해줍니다.

### 8.1.9 arrange(): 데이터 정렬

```
data_market %>%
  select(PBR) %>%
  arrange(PBR) %>%
  head(5)
```

```
##     PBR
## 1  0.19
## 2  0.21
## 3  0.21
## 4  0.23
## 5  0.23
```

```
data_market %>%
  select(ROE) %>%
  arrange(desc(ROE)) %>%
  head(5)
```

```
##       ROE
## 1  1.3404
## 2  0.9589
## 3  0.7175
## 4  0.5761
## 5  0.5714
```

`arrange()` 함수는 선택한 열을 기준으로 데이터를 정렬해주며, 오름차순을 기준으로 정렬합니다. 내림차순으로 데이터를 정렬하고자 할 경우 `arrange()` 내에 `desc()` 함수를 추가로 입력해주면 됩니다.

### 8.1.10 `row_number()`: 순위 계산

```
data_market %>%
  mutate(PBR_rank = row_number(PBR)) %>%
  select(`종목명`, PBR, PBR_rank) %>%
  arrange(PBR) %>%
  head(5)
```

```
##           종목명   PBR PBR_rank
## 1 세아홀딩스 0.19      1
## 2 한국제지 0.21      2
## 3 세원정공 0.21      3
## 4 휴스틸 0.23      4
## 5 전방 0.23      5
```

```
data_market %>%
  mutate(PBR_rank = row_number(desc(ROE))) %>%
  select(`종목명`, ROE, PBR_rank) %>%
  arrange(desc(ROE)) %>%
  head(5)
```

```
##           종목명     ROE PBR_rank
## 1           효성 1.3404      1
## 2 바른손이앤에이 0.9589      2
## 3 THE E&M 0.7175      3
## 4       케이씨 0.5761      4
## 5 한일홀딩스 0.5714      5
```

`row_number()` 함수는 선택한 열의 순위를 구해줍니다. 기본적으로 오름차순 기준으로 순위를 구하며, 내림차순으로 구하고자 할 때는 `desc()` 함수를 추가해 주면 됩니다.

순위를 구하는 함수는 이 외에도 `min_rank()`, `dense_rank()`, `percent_rank()`가 있습니다.

### 8.1.11 ntile(): 분위수 계산

```
data_market %>%
  mutate(PBR_tile = ntile(PBR, n = 5)) %>%
  select(PBR, PBR_tile) %>%
  head()
```

```
##      PBR PBR_tile
## 1  1.28      3
## 2  1.07      3
## 3  0.55      1
## 4 10.42      5
## 5  1.63      4
## 6  0.73      2
```

`ntile()` 함수는 분위수를 계산해주며, `n` 인자를 통해 몇 분위로 나눌지를 선택할 수 있습니다. 해당 함수 역시 오름차순을 기준으로 분위수를 나눕니다.

### 8.1.12 group\_by(): 그룹별로 데이터를 뚫음

```
data_market %>%
  group_by(`SEC_NM_KOR`) %>%
  summarize(n())
```

```
## # A tibble: 11 x 2
##       SEC_NM_KOR     `n()`
##   <chr>           <int>
## 1 <NA>              77
## 2 IT                 574
## 3 건강관리          226
## 4 경기관련소비재    393
## 5 금융               78
## 6 산업재             332
## 7 소재               216
## 8 에너지             23
## 9 유틸리티          19
## 10 커뮤니케이션서비스 7
## 11 필수소비재        96
```

`group_by()` 함수는 선택한 열 중 동일한 데이터를 기준으로 데이터를 묶어줍니다. 위의 예제에서는 섹터를 나타내는 `SEC_NM_KOR` 기준으로 데이터를 묶었으며, `n()` 함수를 통해 해당 그룹 내 데이터의 갯수를 구할 수 있습니다.

```
data_market %>%
  group_by(`SEC_NM_KOR`) %>%
  summarize(PBR_median = median(PBR, na.rm = TRUE)) %>%
  arrange(PBR_median)
```

```
## # A tibble: 11 x 2
##   SEC_NM_KOR      PBR_median
##   <chr>            <dbl>
## 1 유틸리티        0.48
## 2 금융            0.72
## 3 소재            0.76
## 4 산업재          0.93
## 5 커뮤니케이션서비스 0.93
## 6 필수소비재      0.96
## 7 에너지          0.98
## 8 경기관련소비재 1.07
## 9 IT              1.53
## 10 <NA>           1.72
## 11 건강관리       2.59
```

위 예제는 섹터를 기준으로 데이터를 묶은 후, `summarize()`를 통해 각각의 섹터에 속하는 종목들의 PBR 중앙값을 구한 후, 정렬하였습니다.

```
data_market %>%
  group_by(`시장구분`, `SEC_NM_KOR`) %>%
  summarize(PBR_median = median(PBR, na.rm = TRUE)) %>%
  arrange(PBR_median)
```

```
## # A tibble: 22 x 3
## # Groups:   시장구분 [2]
##   시장구분 SEC_NM_KOR      PBR_median
##   <chr>     <chr>            <dbl>
## 1 코스피    유틸리티        0.46
## 2 코스피    금융            0.575
## 3 코스피    소재            0.69
```

```

## 4 코스피 경기관련소비재 0.79
## 5 코스피 에너지 0.8
## 6 코스피 필수소비재 0.81
## 7 코스피 산업재 0.84
## 8 코스피 커뮤니케이션서비스 0.92
## 9 코스닥 유틸리티 0.93
## 10 코스닥 소재 1.07
## # ... with 12 more rows

```

위 예제는 시장과 섹터를 기준으로 데이터를 그룹화한 후, 각 그룹별 PBR 중 앙값을 구하였습니다. 이처럼 그룹의 경우 하나만이 아닌 원하는 만큼 나눌 수 있습니다.

## 8.2 종목정보 시각화

R에서 기본적으로 제공하는 `plot()` 함수를 통해서도 시각화가 충분히 가능합니다. 그러나 데이터 과학자들에게 가장 많이 사랑받는 패키지 중 하나인 `ggplot2` 패키지의 `ggplot()` 함수를 사용할 경우 훨씬 아름답게 그림이 표현 가능하며, 다양한 기능들을 매우 쉽게 사용할 수도 있습니다.

`ggplot()` 함수는 플러스(+) 기호를 사용한다는 점과 문법이 다소 어색하다는 점 때문에 처음에 배우기가 쉽지는 않습니다. 그러나 해당 패키지의 근본이 되는 철학인 **The Grammar of Graphics**를 이해하고 조금만 연습해본다면, 충분히 손쉽게 사용이 가능합니다.

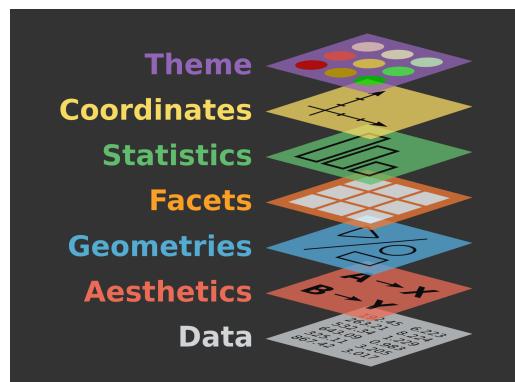
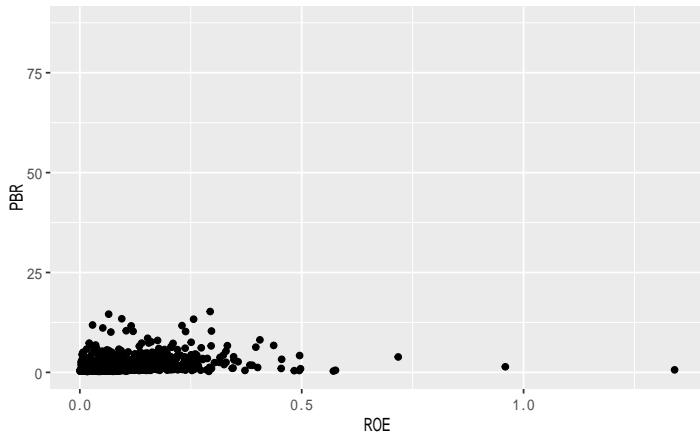


Figure 8.2: The Grammar of Graphics

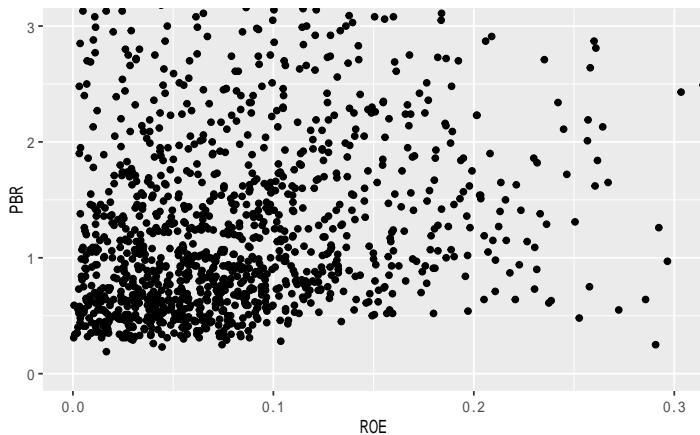
### 8.2.1 geom\_point(): 산점도 나타내기

```
library(ggplot2)  
  
ggplot(data_market, aes(x = ROE, y = PBR)) +  
  geom_point()
```



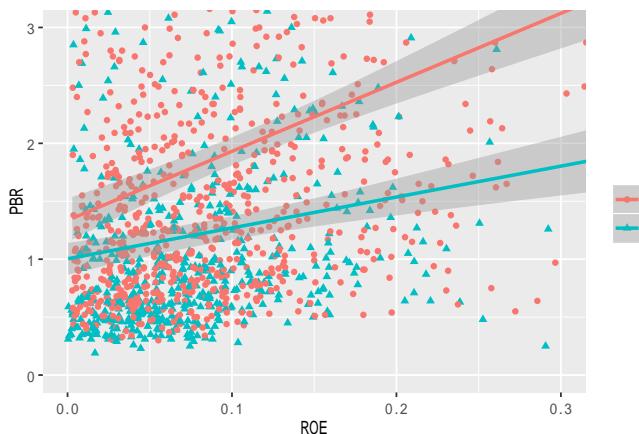
1. ggplot() 함수 내에 사용될 데이터인 data\_market을 입력하며, aes 인자 내부에 x축은 ROE, y축은 PBR 열을 사용하도록 정의합니다.
2. geom\_point() 함수를 통해 산점도 그래프를 그려주도록 합니다. 원하는 그림이 그려지기는 하였으나, ROE와 PBR에 극단치 데이터가 존재하여 둘간의 관계가 잘 보이지 않습니다.

```
ggplot(data_market, aes(x = ROE, y = PBR)) +  
  geom_point() +  
  coord_cartesian(xlim = c(0, 0.30), ylim = c(0, 3))
```



이번에는 극단치 효과를 제거하기 위해 `coord_cartesian()` 함수 내에 `xlim`과 `ylim`, 즉 x축과 y축의 범위를 직접 지정해주도록 합니다. 극단치가 제거되어 데이터를 한 눈에 확인할 수 있습니다.

```
ggplot(data_market, aes(x = ROE, y = PBR,
                        color = `시장구분`,
                        shape = `시장구분`)) +
  geom_point() +
  geom_smooth(method = 'lm') +
  coord_cartesian(xlim = c(0, 0.30), ylim = c(0, 3))
```

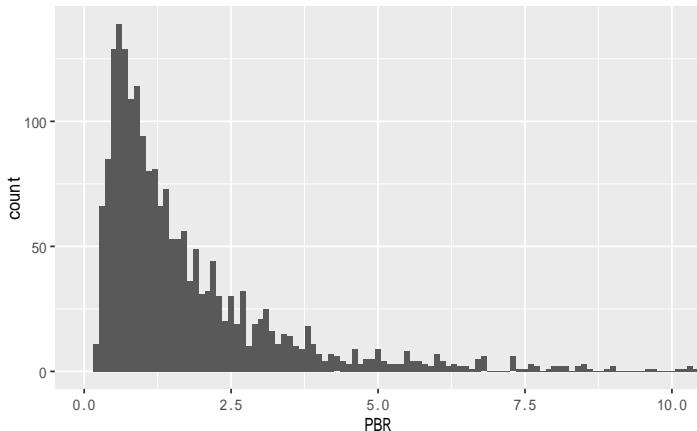


1. `ggplot()` 함수 내부 `aes` 인자에 `color`와 `shape`을 지정해주면, 해당 그룹 별로 모양과 색이 나타납니다. 코스피와 코스닥 종목들에 해당하는 데이터의 색과 점 모양이 다르게 표시할 수 있습니다.

2. `geom_smooth()` 함수를 통해 평활선을 추가해줄 수도 있으며, 방법으로 `lm`(linear model)을 지정해줄 경우 선형회귀선을 그려주게 됩니다. 이 외에도 `glm`, `gam`, `loess` 등의 다양한 회귀선을 그려줄 수 있습니다.

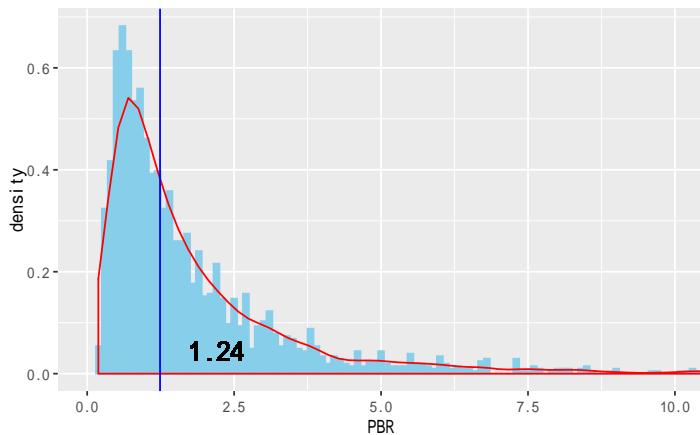
### 8.2.2 `geom_histogram()`: 히스토그램 나타내기

```
ggplot(data_market, aes(x = PBR)) +
  geom_histogram(binwidth = 0.1) +
  coord_cartesian(xlim = c(0, 10))
```



`geom_histogram()` 함수는 히스토그램을 나타내주며, `binwidth` 인자를 막대의 통해 너비를 선택해줄 수 있습니다. 국내 종목들의 PBR 데이터는 좌측에 쏠려 있고 오른쪽으로 꼬리가 긴 분포를 가지고 있습니다.

```
ggplot(data_market, aes(x = PBR)) +
  geom_histogram(aes(y = ..density..),
                 binwidth = 0.1,
                 color = 'sky blue', fill = 'sky blue') +
  coord_cartesian(xlim = c(0, 10)) +
  geom_density(color = 'red') +
  geom_vline(aes(xintercept = median(PBR, na.rm = TRUE)),
             color = 'blue') +
  geom_text(aes(label = median(PBR, na.rm = TRUE),
                x = median(PBR, na.rm = TRUE), y = 0.05),
            col = 'black', size = 6, hjust = -0.5)
```

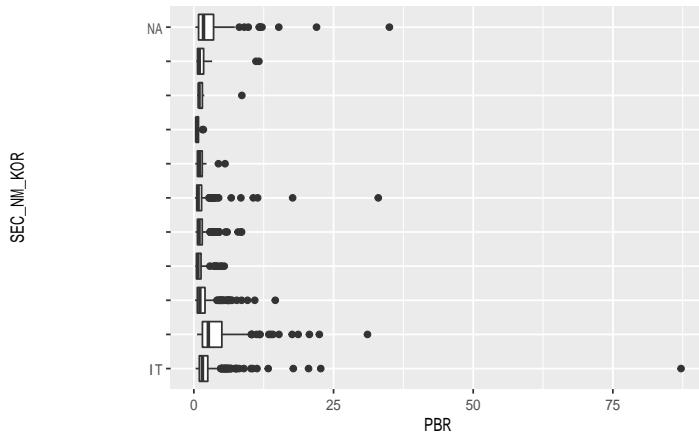


PBR 히스토그램을 좀 더 자세하게 나타내보도록 하겠습니다.

1. `geom_histogram()` 함수 내에 `aes(y = ..density..)`를 추가해주어 밀도함수로 바꾸어주도록 합니다.
2. `geom_density()` 함수를 추가해 밀도곡선을 그려줍니다.
3. `geom_vline()` 함수는 세로선을 그려주며, `xintercept` 즉 x축으로 PBR의 중앙값을 선택해 줍니다.
4. `geom_text()` 함수는 그림 내에 글자를 표현해주며, `label` 인자에 원하는 글자를 입력해준 후 글자가 표현될 x축, y축, 색상, 사이즈 등을 선택할 수 있습니다.

### 8.2.3 `geom_boxplot()`: 박스플롯 나타내기

```
ggplot(data_market, aes(x = SEC_NM_KOR, y = PBR)) +
  geom_boxplot() +
  coord_flip()
```



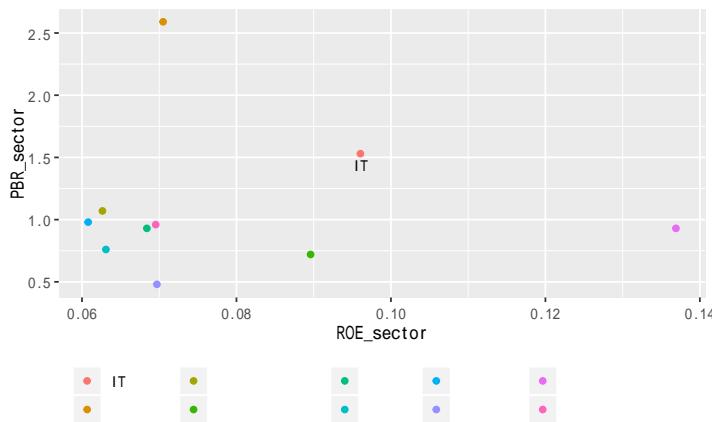
박스플롯 역시 데이터의 분포와 이상치를 확인하기 좋은 그림이며, `geom_boxplot()` 함수를 통해 나타낼 수 있습니다.

1. x축 데이터로는 섹터 정보, y축 데이터로는 PBR을 선택해 줍니다.
2. `geom_boxplot()`를 통해 박스플롯을 그려줍니다.
3. `coord_flip()` 함수는 x축과 y축을 뒤집어 표현해주며 x축에 PBR, y축에 섹터 정보가 나타나게 되었습니다.

결과를 살펴보면 유털리티나 금융 섹터의 경우 PBR이 잘 모여있는 반면, IT나 건강관리 섹터 등은 매우 극단적인 PBR을 가지고 있는 종목이 존재합니다.

#### 8.2.4 dplyr과 ggplot을 연결하여 사용하기

```
data_market %>%
  filter(!is.na(SEC_NM_KOR)) %>%
  group_by(SEC_NM_KOR) %>%
  summarize(ROE_sector = median(ROE, na.rm = TRUE),
            PBR_sector = median(PBR, na.rm = TRUE)) %>%
  ggplot(aes(x = ROE_sector, y = PBR_sector,
             color = SEC_NM_KOR, label = SEC_NM_KOR)) +
  geom_point() +
  geom_text(color = 'black', size = 3, vjust = 1.3) +
  theme(legend.position = 'bottom',
        legend.title = element_blank())
```



앞에서 배운 데이터 분석과 시각화를 동시에 연결하여 사용할 수도 있습니다.

- 데이터 분석의 단계로 `filter()`를 통해 섹터가 NA가 아닌 종목을 선택합니다.
- `group_by()`를 통해 섹터 별 그룹을 묶습니다.
- `summarize()`를 통해 ROE와 PBR의 중앙값을 계산해줍니다. 해당 과정을 거치면 다음의 결과가 계산됩니다.

```
## # A tibble: 10 x 3
##   SEC_NM_KOR      ROE_sector  PBR_sector
##   <chr>           <dbl>        <dbl>
## 1 IT              0.0960       1.53
## 2 건강관리       0.0705       2.59
## 3 경기관련소비재 0.0626       1.07
## 4 금융            0.0896       0.72
## 5 산업재          0.0684       0.93
## 6 소재            0.0631       0.76
## 7 에너지          0.0608       0.98
## 8 유tility         0.0697       0.48
## 9 커뮤니케이션서비스 0.1370       0.93
## 10 필수소비재     0.0696       0.96
```

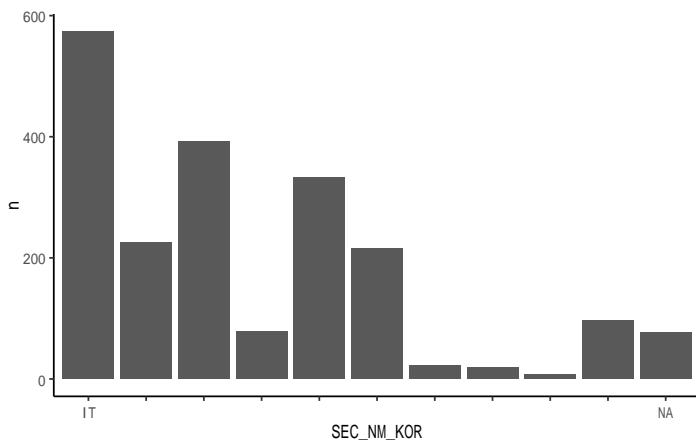
해당 결과를 파이프 오퍼레이터 (`%>%`)로 이용 경우 그대로 시각화가 가능하며, `ggplot()` 함수 내에 데이터를 입력하지 않아도 됩니다.

- x축과 y축을 설정해준 뒤, 색상과 라벨을 섹터로 지정해주면 각 섹터 별로 다른 색상의 산점도가 그려지게 됩니다.

5. `geom_text()` 함수를 통해 앞에서 라벨로 지정한 섹터 정보들을 출력해줍니다.
6. `theme()` 함수를 통해 다양한 테마를 지정해주도록 합니다. `legend.position` 인자를 통해 범례를 하단에 배치하였으며, `legend.title` 인자를 통해 범례의 제목을 삭제해 주었습니다.

### 8.2.5 `geom_bar()`: 막대그래프를 나타내기

```
data_market %>%
  group_by(SEC_NM_KOR) %>%
  summarize(n = n()) %>%
  ggplot(aes(x = SEC_NM_KOR, y = n)) +
  geom_bar(stat = 'identity') +
  theme_classic()
```

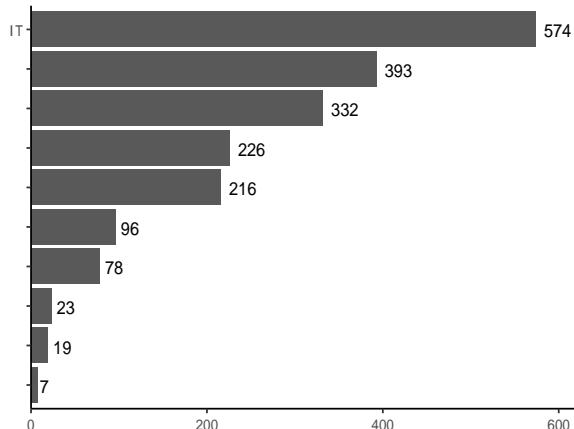


`geom_bar()`는 막대그래프를 그려주는 함수입니다.

1. 먼저 `group_by()`를 통해 섹터 별 그룹을 묶어줍니다.
2. `summarize()` 함수 내부에 `n()`을 통해, 각 그룹 별 데이터 갯수를 구합니다.
3. `ggplot()` 함수에서 x축에는 `SEC_NM_KOR`, y축에는 `n`을 지정해줍니다.
4. `geom_bar()`를 통해 막대그래프를 그려주도록 합니다. y축에 해당하는 `n` 데이터를 그대로 사용하기 위해서는 `stat`인자를 **identity**로 지정해주어야 합니다. `theme_*`() 함수를 통해 배경 테마를 바꿀 수도 있습니다.

한편 위 그래프는 데이터 갯수에 따라 순서대로 막대가 정렬되지 않아 보기 좋지 않은 형태입니다. 이를 반영하여 더욱 보기 좋은 그래프로 나타내주도록 하겠습니다.

```
data_market %>%
  filter(!is.na(SEC_NM_KOR)) %>%
  group_by(SEC_NM_KOR) %>%
  summarize(n = n()) %>%
  ggplot(aes(x = reorder(SEC_NM_KOR, n), y = n, label = n)) +
  geom_bar(stat = 'identity') +
  geom_text(color = 'black', size = 4, hjust = -0.3) +
  xlab(NULL) +
  ylab(NULL) +
  coord_flip() +
  scale_y_continuous(expand = c(0, 0, 0.1, 0)) +
  theme_classic()
```



1. `filter()` 함수를 통해 NA 종목은 삭제해준 후, 섹터 별 종목 갯수를 구해주세요 합니다.
2. `ggplot()`의 x축에 `reorder()` 함수를 적용하여 `SEC_NM_KOR` 변수를 n 순서대로 정렬해줍니다.
3. `geom_bar()`를 통해 막대그래프를 그려준 후, `geom_text()`를 통해 라벨에 해당하는 종목 갯수를 출력합니다.
4. `xlab()`과 `ylab()`에 `NULL`을 입력해 라벨을 삭제해 줍니다.
5. `coord_flip()` 함수를 통해 x축과 y축을 뒤집어 줍니다.
6. `scale_y_continuous()` 함수를 통해 그림의 간격을 약간 넓혀줍니다.
7. `theme_classic()`로 테마를 변경해줍니다.

결과를 보면, 종목수가 많은 섹터부터 순서대로 정렬되어 보기도 쉬우며, 종목수도 텍스트로 표현되어 한 눈에 확인할 수 있습니다.

이처럼 데이터 시각화를 통해 정보의 분포나 특성을 한눈에 확인할 수 있으며, `ggplot()`을 이용할 경우 복잡한 형태의 그림도 매우 간단하고 아름답게 표현할 수 있습니다.

## 8.3 주가 및 수익률 시각화

주가 혹은 수익률을 그리는 것 역시 매우 중요합니다. R 내의 기본 함수로도 이를 나타낼 수 있지만, 패키지를 사용한다면 더욱 보기좋은 그래프를 그릴수도 있습니다. 또한 최근에 나오는 여러 패키지들을 이용하면 매우 손쉽게 인터랙티브 그래프를 구현할 수도 있습니다.

### 8.3.1 주가 그래프 나타내기

```
library(quantmod)
```

```
getSymbols('SPY')
prices = Cl(SPY)
```

`getSymbols()` 함수를 이용하여 미국 S&P500 지수를 추종하는 ETF인 SPY의 데이터를 다운로드 받은 후, `Cl()` 함수를 이용하여 종가에 해당하는 데이터만 추출합니다. 이제 해당 가격 및 수익률을 바탕으로 그래프를 그려보도록 하겠습니다.

```
plot(prices, main = 'Price')
```



`getSymbols()` 함수는 데이터를 `xts` 형식으로 내려받으며, R에서는 데이터가 `xts` 형식일 경우 기본함수인 `plot()`으로 그래프를 그려도 x축에 시간을 나타내주며, 우측 상단에 기간을 표시해 줍니다. 그러나 완벽히 깔끔한 형태의 그래프를 보기는 어려운 면이 있습니다.

```
library(ggplot2)

SPY %>%
  ggplot(aes(x = Index, y = SPY.Close)) +
  geom_line()
```



`ggplot()`을 이용할 경우 기본 `plot()` 보다 한결 깔끔해졌으며, 패키지 내의 다양한 함수를 이용해 그래프를 꾸며 나갈 수도 있습니다.

### 8.3.2 인터랙티브 그래프 나타내기

```
library(dygraphs)

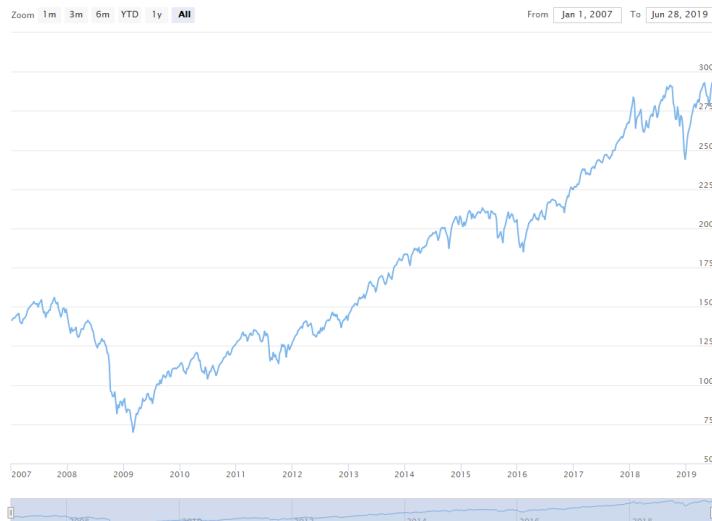
dygraph(prices) %>%
  dyRangeSelector()
```



`dygraphs` 패키지의 `dygraph()` 함수를 이용하면 사용자의 움직임에 따라 반응하는 그래프를 그릴 수 있습니다. 해당 패키지는 JavaScript를 이용하여 인터랙티브한 그래프를 구현하며, 그래프 위에 마우스를 올릴 경우 날짜 및 가격이 표시되기도 하며, 하단의 셀렉터를 이용해 원하는 기간의 수익률을 선택할 수도 있습니다.

```
library(highcharter)

highchart(type = 'stock') %>%
  hc_add_series(prices) %>%
  hc_scrollbar(enabled = FALSE)
```

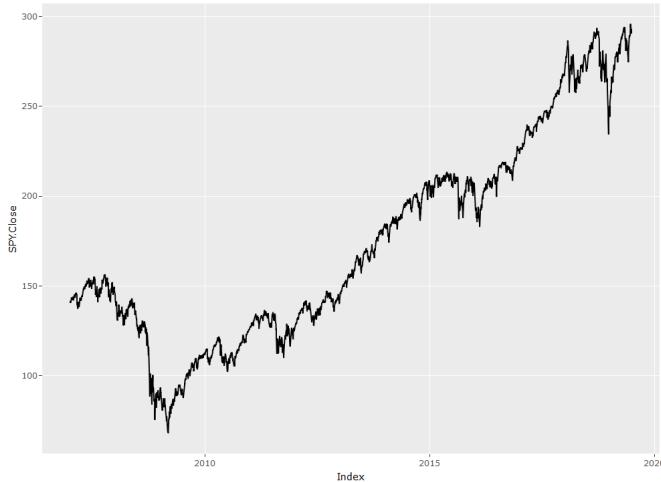


highcharter 패키지의 `highchart()` 함수 역시 이와 비슷한 기능을 하며, 인터랙티브 그래프를 생성해줍니다. 좌측 상단의 기간을 클릭할 경우 해당 기간의 수익률 만을 확인할 수도 있으며, 우측 상단에 기간을 직접 입력할 수도 있습니다.

```
library(plotly)

p = SPY %>%
  ggplot(aes(x = Index, y = SPY.Close)) +
  geom_line()

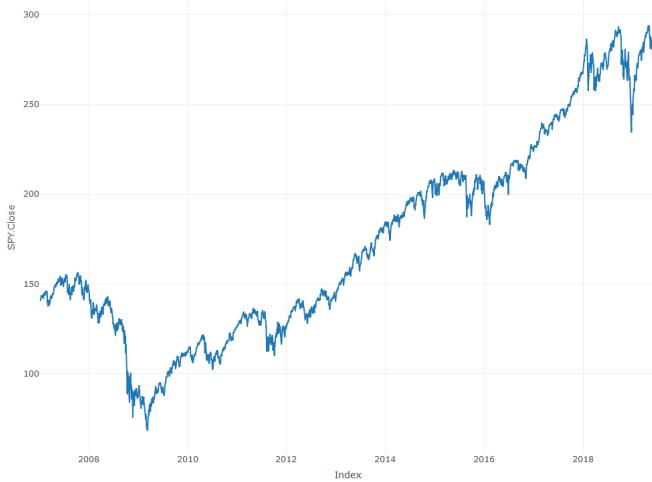
ggplotly(p)
```



plotly 패키지는 R 뿐만 아니라 Python, MATLAB, Julia 등 여러 프로그래밍 언어에 사용될 수 있는 그래픽 패키지로써 최근에 많은 사랑을 받고 있습니다. R 내에서는 단순히 `ggplot()`을 이용해 나타낸 그림에 `ggplotly()` 함수를 추가해주는 것 만으로 인터랙티브한 그래프를 만들어 줍니다.

또한 해당 패키지는 최근 샤이니에서도 많이 사용되고 있습니다. 따라서 샤이니를 이용한 웹페이지 제작을 생각하고 계신분이라면, 원래의 함수 실행법도 알아두면 좋습니다.

```
prices %>%
  fortify.zoo %>%
  plot_ly(x= ~Index, y = ~SPY.Close ) %>%
  add_lines()
```



`plot_ly()` 함수 내부에 x축과 y축을 설정해주며, 변수명 앞에 물결표(~)를 붙여줍니다. 그 후, `add_lines()` 함수를 추가해주면 선그래프를 표시해줍니다. `ggplot()` 함수의 경우 각 레이어의 연결을 플러스 기호(+)를 통해 해주었지만, `plot_ly()` 함수의 경우 파이프 오퍼레이터(%>%)를 통해 할 수 있다는 장점이 있습니다.

### 8.3.3 연도별 수익률 나타내기

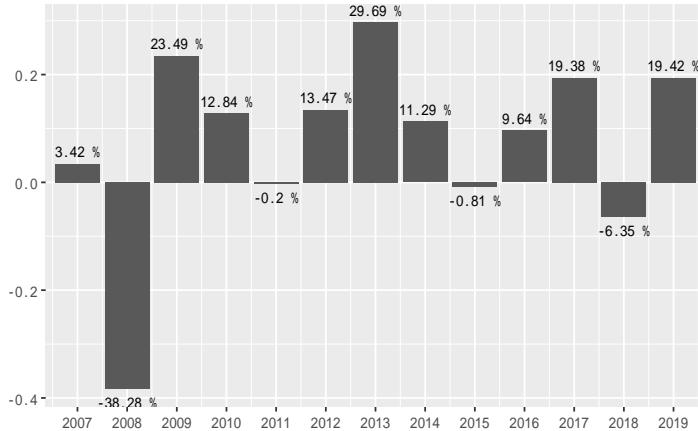
주가 그래프 외에 연도별 수익률을 그리는 것도 중요합니다. `ggplot()`을 통해 연도별 수익률을 막대그래프로 나타내는 방법을 살펴보도록 하겠습니다.

```
library(PerformanceAnalytics)

ret_yearly = prices %>%
  Return.calculate() %>%
  apply.yearly(., Return.cumulative) %>%
  round(4) %>%
  fortify.zoo() %>%
  mutate(Index = as.numeric(substr(Index, 1, 4)))

ggplot(ret_yearly, aes(x = Index, y = SPY.Close)) +
  geom_bar(stat = 'identity') +
  scale_x_continuous(breaks = ret_yearly$Index,
                     expand = c(0.01, 0.01)) +
```

```
geom_text(aes(label = paste(round(SPY.Close * 100, 2), "%"),
               vjust = ifelse(SPY.Close >= 0, -0.5, 1.5)),
           position = position_dodge(width = 1),
           size = 3) +
xlab(NULL) + ylab(NULL)
```



- 먼저 `apply.yearly()` 함수를 이용해 연도별 수익률을 계산해준 뒤 반올림을 해줍니다.
- `fortify.zoo()` 함수를 통해 인덱스에 있는 시간 데이터를 Index 열로 이동합니다.
- `mutate()` 함수 내에 `substring()` 을 통해 Index의 1번째부터 4번째 글자, 즉 연도에 해당하는 부분을 뽑아낸 후, 숫자 형태로 저장합니다.
- `ggplot()` 함수를 이용해 x축에는 연도가 저장된 Index, y축에는 수익률이 저장된 `SPY.Close`를 입력합니다.
- `geom_bar()` 함수를 통해 막대그래프를 그려줍니다.
- `scale_x_continuous()` 함수를 통해 x축에 모든 연도가 출력되도록 합니다.
- `geom_text()`를 통해 막대그래프에 연도별 수익률이 표시되도록 합니다. `vjust()` 내에 `ifelse()` 함수를 사용하여 수익률이 0보다 크면 위쪽에, 0보다 작으면 아래쪽에 표시되도록 합니다.

해당 과정을 거치면 막대 그래프와 텍스트를 통해 연도별 수익률을 한 눈에 확인할 수 있게 됩니다.

# CHAPTER 9

## 퀀트 전략을 이용한 종목선정 (기본)

투자에 필요한 주가, 재무제표, 가치지표 데이터가 준비되었다면 퀀트 전략을 활용하여 투자하고자 하는 종목을 선정해야 합니다.

퀀트 투자는 크게 포트폴리오 운용 전략과 트레이딩 전략으로 나눌 수 있습니다. 포트폴리오 운용 전략의 경우 과거 주식 시장을 분석하여 좋은 주식의 기준을 찾아낸 후 해당 기준에 만족하는 종목을 매수하거나, 이와 반대에 있는 나쁜 주식을 공매도하기도 합니다. 투자의 속도가 느리며, 다수의 종목을 하나의 포트폴리오로 구성하여 운용하는 특징이 있습니다. 반면 트레이딩 전략의 경우, 단기간에 발생되는 주식의 움직임을 연구한 후 예측하여, 매수 혹은 매도하는 전략입니다. 투자의 속도가 빠르며 소수의 종목을 대상으로 합니다.

Table 9.1: 퀀트 투자 종류의 비교

기준	포트폴리오 운용 전략	트레이딩 전략
투자철학	규칙에 기반한 투자	규칙에 기반한 투자
투자목적	좋은 주식을 매수	좋은 시점을 매수
학문적 기반	경제학, 통계학 등	통계학, 공학, 정보처리 등
투자의 속도	느림	빠름

이 중 본 책에서는 포트폴리오에 기반한 운용 전략에 대해 다루도록 합니다. 주식의 수익률에 영향을 미치는 요소를 팩터Factor라 합니다. 즉 팩터의 강도가 양인 종목들로 구성한 포트폴리오의 경우 향후 수익률이 높을 것으로 예상되며,

팩터의 강도가 음인 종목들로 구성한 포트폴리오의 경우 반대로 향후 수익률이 낮을 것으로 예상됩니다.

팩터에 대한 연구는 학자들에 의해 오랫동안 진행되어 왔지만, 일반 투자자들이 이러한 논문을 모두 찾아보고 연구하는 것은 사실상 불가능에 가깝습니다. 그러나 최근에는 **스마트베타**라는 이름으로 팩터 투자가 대중화되고 있습니다. 최근 유행하고 있는 스마트베타 ETF의 경우 팩터를 기준으로 포트폴리오를 구성한 상품으로써, 학계나 실무에서 검증된 팩터 전략을 기반으로 합니다.

해당 상품들의 홈페이지나 투자설명서에는 종목 선정 기준에 대해 자세히 나와 있으므로 이는 매우 훌륭한 투자 전략이기도 합니다. 따라서 스마트베타 ETF에 나와있는 투자 전략을 자세히 분석하는 것만으로도 훌륭한 퀸트 투자 전략을 만들 수 있습니다.

- 산출기관 : FnGuide
- 지수개요 : 유가증권시장에 상장된 시가총액 상위 300위 이내 종목 중, 아래 벨류 팩터 및 웰리티 팩터 총 8종류의 개별 값을 더하여 산출한 값 기준으로 상위 50종목으로 지수 구성
- 벨류 팩터 : 순자산/시가총액, 매출액/시가총액, 현금흐름/시가총액, 배당금/시가총액
- 웰리티 팩터 : 자기자본영업이익률 변동성, 베타, 레버리지 비율, 매출총이익/자산총액
- 산출방식: 동일가중방식
- 정기변경 : 연 2회(6월, 12월)
- 기준일 및 기준지수: 2001년 1월 2일 기준 1,000 Point

Figure 9.1: 스마트베타 ETF 전략 예시

본 장에서는 투자에 많이 활용되는 기본적인 팩터에 대해 알아보고, 우리가 구한 데이터를 바탕으로 각 팩터 별 투자 종목을 선택하는 방법에 대해 알아보도록 하겠습니다.

아울러 본 책에서 각종 모델을 통해 나온 종목들은, 데이터를 받은 시점에서의 종목이며 매우 추천은 아님을 밝힙니다.

## 9.1 베타 이해하기

투자자들이라면 누구나 한번은 들어봤을만한 용어가 베타<sup>Beta</sup>입니다. 기본적으로 개별 주식의 수익률에 가장 크게 영향을 주는 요소는 주식시장의 움직임일수 밖에 없습니다. 아무리 좋은 주식도 주식시장이 폭락한다면 같이 떨어지며, 아무리 나쁜 주식도 주식시장이 상승한다면 대부분 같이 오르기 마련입니다.

개별 주식이 전체 주식시장의 변동에 반응하는 정도를 나타낸 값이 베타입니다. 베타가 1이라는 뜻은 주식시장과 움직임이 정확히 같다는 뜻으로써, 시장 그 자체를 나타냅니다. 베타가 1.5라는 뜻은 주식시장이 수익률이 +1% 일 때 개별 주식의 수익률은 +1.5%, 반대로 주식시장의 수익률이 -1% 일 때 개별 주식의 수익률은 -1.5% 움직인다는 뜻입니다. 반면 베타가 0.5라면 주식시장 수익률의 절반 정도만이 움직이게 됩니다.

Table 9.2: 베타에 따른 개별 주식의 수익률 움직임

베타	주식시장이 +1% 일 경우	주식시장이 -1% 일 경우
0.5	+0.5%	-0.5%
1.0	+1.0%	-1.0%
1.5	+1.5%	-1.5%

이처럼 베타가 큰 주식은 주식시장보다 수익률의 움직임이 크며, 반대로 베타가 낮은 주식은 주식시장보다 수익률의 움직임이 작습니다. 따라서 일반적으로 상승장이 기대될 때는 베타가 큰 주식에, 하락장일이 기대될 때는 베타가 낮은 주식에 투자하는 것이 좋습니다.

주식시장에서의 베타는 통계학의 회귀분석모형에서 기울기를 나타내는 베타와 정확히 의미가 같습니다. 회귀분석모형은  $y = a + bx$  형태로 나타나며, x의 변화에 따른 y의 변화의 기울기가 회귀계수인 b입니다. 이를 주식에 적용한 모형이 자산가격결정모형(CAPM: Capital Asset Pricing Model)이며, 그 식은 다음과 같습니다.

$$\text{회귀분석모형} : y = a + bx$$

$$\text{자산가격결정모형} : R_i = R_f + \beta_i \times [R_m - R_f]$$

먼저 회귀분석모형의 상수항인 a에 해당하는 부분은 무위험 수익률을 나타내는  $R_f$ 입니다. 독립변수인 x에 해당하는 부분은 무위험 수익률 대비 주식 시장의 초과 수익률을 나타내는 시장위험 프리미엄인  $R_m - R_f$ 입니다. 종속변수인 y에 해당하는 부분은 개별주식의 수익률을 나타내는  $R_i$ 이며, 최종적으로 회귀계수인 b에 해당하는 부분은 개별 주식의 베타입니다.

Table 9.3: 회귀분석모형과 자산가격결정모형의 비교

구분	회귀분석모형	자산가격결정모형
상수항	a	$R_f$ (무위험 수익률)
독립변수	x	$R_m - R_f$ (시장위험 프리미엄)
종속변수	y	$R_i$ (개별주식의 수익률)
회귀계수	b	$\beta_i$ (개별주식의 베타)

통계학에서 회귀계수는  $\beta = \frac{\text{cov}(x,y)}{\sigma_x^2}$  형태로 구할 수 있으며, x와 y에 각각 시장수익률과 개별주식의 수익률을 대입할 경우 개별주식의 베타는  $\beta_i = \rho(i,m) \times \frac{\sigma_i}{\sigma_m}$  형태로 구할 수 있습니다. 그러나 이러한 수식을 모르더라도 R에서는 간단히 베타를 구할 수 있습니다.

### 9.1.1 베타 계산하기

베타를 구하는 방법을 알아보기 위해 주식시장에 대한 대용치로 KOSPI 200 ETF, 개별주식으로는 전통적 고베타주인 증권주를 이용하겠습니다.

```
library(quantmod)
library(PerformanceAnalytics)
library(magrittr)

symbols = c('102110.KS', '039490.KS')
getSymbols(symbols)

## [1] "102110.KS" "039490.KS"

prices = do.call(cbind,
                  lapply(symbols, function(x) Cl(get(x)))) 

ret = Return.calculate(prices)
ret = ret['2016-01::2018-12']
```

1. KOSPI 200 ETF인 TIGER 200(102110.KS), 증권주인 키움증권(039490.KS)의 티커를 입력합니다.
2. getSymbols() 함수를 이용하여 해당 티커들의 데이터를 다운로드 받습니다.
3. lapply() 함수 내에 Cl()과 get() 함수를 사용하여 종가에 해당하는 데이터만 추출하며, 리스트 형태의 데이터를 열의 형태로 묶어주기 위해 do.call() 함수와 cbind() 함수를 사용해 줍니다.
4. Return.calculate() 함수를 통해 수익률을 계산해 줍니다.
5. xts 형식의 데이터는 대괄호 속에 ['시작일자::종료일자']와 같은 형태로, 원하는 날짜를 편리하게 선택할 수 있으며, 위에서는 2016년 1월부터 2018년 12월 까지 데이터를 선택합니다.

```
rm = ret[, 1]
ri = ret[, 2]

reg = lm(ri ~ rm)
summary(reg)

##
```

```

## Call:
## lm(formula = ri ~ rm)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.06890 -0.01295 -0.00172  0.01082  0.09542
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.000373   0.000723   0.52    0.61
## rm          1.761433   0.090739  19.41   <2e-16 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0195 on 727 degrees of freedom
## (2 observations deleted due to missingness)
## Multiple R-squared:  0.341, Adjusted R-squared:  0.34
## F-statistic:  377 on 1 and 727 DF, p-value: <2e-16

```

증권주를 대상으로 베타를 구하기 위한 회귀분석을 실시합니다. 자산가격결정 모형의 수식인  $R_i = R_f + \beta_i \times [R_m - R_f]$ 에서 편의를 위해 무위험 수익률인  $R_f$ 를 0으로 가정하면,  $R_i = \beta_i \times R_m$ 의 형태로 나타낼 수 있습니다. 이 중  $R_m$ 는 독립변수인 주식시장의 수익률을,  $R_i$ 는 종속변수인 개별주식의 수익률을 의미합니다.

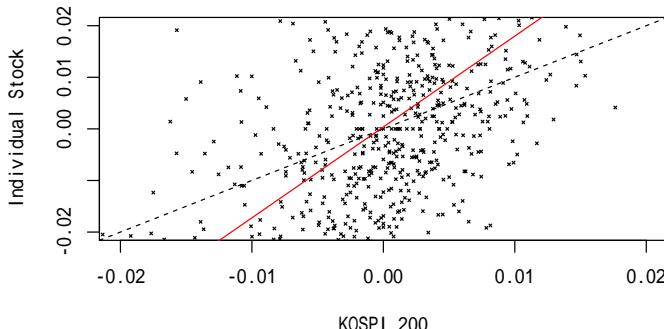
1. 독립변수는 첫번째 열인 KOSPI 200 ETF의 수익률을 선택하며, 종속변수는 두번째 열인 증권주의 수익률을 선택합니다.
2. `lm()` 함수를 통해 손쉽게 선형회귀분석을 실시할 수 있으며, 회귀분석의 결과를 `reg` 변수에 저장해줍니다.
3. `summary()` 함수는 데이터의 요약 정보를 나타내며, 해당 예시에서는 회귀분석결과에 대한 정보를 보여줍니다.

회귀분석의 결과 중 가장 중요한 부분은 계수를 나타내는 Coefficients 부분입니다. Intercept 부분은 회귀분석의 상수항에 해당하는 부분으로써, 값이 거의 0에 가깝고 t밸류 또한 매우 작아 유의하지 않음이 보입니다. 우리가 원하는 베타에 해당하는 부분은 x의 Estimate 부분으로써, 베타값이 1.76으로 증권주의 특성인 고베타주임이 확인되며, t밸류 또한 19.41로 매우 유의한 결과입니다. 조정된 결정계수(Adjusted R-square)는 0.34를 보입니다.

### 9.1.2 베타 시각화

다음으로 구해진 베타를 그림으로 표현해보도록 하겠습니다.

```
plot(as.numeric(rm), as.numeric(ri), pch = 4, cex = 0.3,
     xlab = "KOSPI 200", ylab = "Individual Stock",
     xlim = c(-0.02, 0.02), ylim = c(-0.02, 0.02))
abline(a = 0, b = 1, lty = 2)
abline(reg, col = 'red')
```



1. `plot()` 함수를 통해 그림을 그려주며, x축과 y축에 주식시장 수익률과 개별주식 수익률을 입력합니다. `pch`는 점들의 모양을, `cex`는 점들의 크기를 나타내며, `xlab`과 `ylab`은 각각 x축과 y축에 들어갈 문구를 나타냅니다. `xlim`과 `ylim`은 x축과 y축의 최소 및 최대 범위를 지정해줍니다.
2. 첫번째 `abline()`에서 `a`는 상수, `b`는 직선의 기울기, `lty`는 선의 유형을 나타냅니다. 이를 통해 기울기, 즉 베타가 1일 경우의 선을 점선으로 표현합니다.
3. 두번째 `abline()`에 회귀분석 결과를 입력해주면 자동적으로 회귀식을 그려줍니다.

검은색의 점선이 기울기가 1인 경우이며, 붉은색의 직선이 증권주의 회귀분석 결과를 나타냅니다. 기울기가 1보다 훨씬 가파름이 확인되며, 즉 베타가 1보다 크다는 사실을 알 수 있습니다.

## 9.2 저변동성 전략

금융 시장에서 변동성은 수익률이 움직이는 정도로 써, 일반적으로 표준편차가 사용됩니다. 표준편차는 자료가 평균을 중심으로 얼마나 퍼져 있는지를 나타내는

수치로써, 수식은 다음과 같습니다.

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}}$$

관측값의 개수가 작을 경우에는 수식에 대입하여 계산하는 것이 가능하지만, 관측값이 수백 혹은 수천개로 늘어날 경우 컴퓨터를 이용하지 않고 계산하는 것은 사실상 불가능합니다. R에서는 복잡한 계산과정 없이 `sd()` 함수를 이용하여 간단하게 표준편차를 계산할 수 있습니다.

```
example = c(85, 76, 73, 80, 72)
sd(example)
```

```
## [1] 5.357
```

개별 주식의 표준편차를 측정할 때는 주식의 가격이 아닌 수익률로 계산해야 합니다. 수익률의 표준편차가 크다는 의미는 수익률이 위 아래로 많이 움직여 위험한 종목으로 여겨집니다. 반면, 표준편차가 작다는 의미는 수익률의 움직임이 적어 상대적으로 안전한 종목으로 여겨집니다.

전통적 금융 이론에서는 수익률의 변동성이 클수록 위험이 크고, 이런 위험에 대한 보상으로 기대수익률이 높아야 한다고 보았습니다. 따라서 고변동성 종목의 기대수익률이 크고, 저변동성 종목의 기대수익률이 낮은 고위험 고수익이 당연한 믿음이었습니다. 그러나 현실에서는 오히려 변동성이 낮은 종목들의 수익률이 변동성이 높은 종목들의 수익률 보다 높은, 저변동성 효과가 발견되고 있습니다. 이러한 저변동성 효과가 발생하는 원인으로는 여러 가설이 있습니다.

1. 투자자들은 대체로 자신의 능력을 과신하는 경향이 있으며, 복권과 같이 큰 수익을 가져다 주는 고변동성 주식을 선호하는 경향이 있습니다. 이러한 결과로 고변동성 주식은 과대 평가되어 수익률이 낮은 반면, 과소 평가된 저변동성 주식들은 높은 수익률을 보이게 됩니다.
2. 대부분 기관투자가들이 레버리지 투자가 되지 않는 상황에서, 벤치마크 대비 높은 성과를 얻기 위해 고변동성 주식에 투자하는 경향이 있으며, 이 또한 고변동성 주식이 과대 평가되는 결과로 이어집니다.
3. 시장의 상승과 하락이 반복됨에 따라 고변동성 주식이 변동성 손실(Volatility Drag)로 인해 수익률이 하락하게 되는 이유도 있습니다.

주식의 위험은 변동성뿐만 아니라 베타 등 여러 지표로도 측정할 수 있습니다. 저변동성 효과와 비슷하게 고유변동성이 낮은 주식의 수익률이 높은 저고유변동성 효과, 베타가 낮은 주식의 수익률이 오히려 높은 저베타 효과도 발견되고 있으며, 이러한 효과들을 합쳐 저위험 효과로 부르기도 합니다.

### 9.2.1 저변동성 포트폴리오 구하기: 일간 기준

먼저 최근 1년 일간 수익률 기준 변동성이 낮은 30 종목을 선택하도록 하겠습니다.

```
library(stringr)
library(xts)
library(PerformanceAnalytics)
library(magrittr)
library(ggplot2)
library(dplyr)

KOR_price = read.csv('data/KOR_price.csv', row.names = 1,
                      stringsAsFactors = FALSE) %>% as.xts()
KOR_ticker = read.csv('data/KOR_ticker.csv', row.names = 1,
                      stringsAsFactors = FALSE)
KOR_ticker$'종목코드' =
  str_pad(KOR_ticker$'종목코드', 6, 'left', 0)

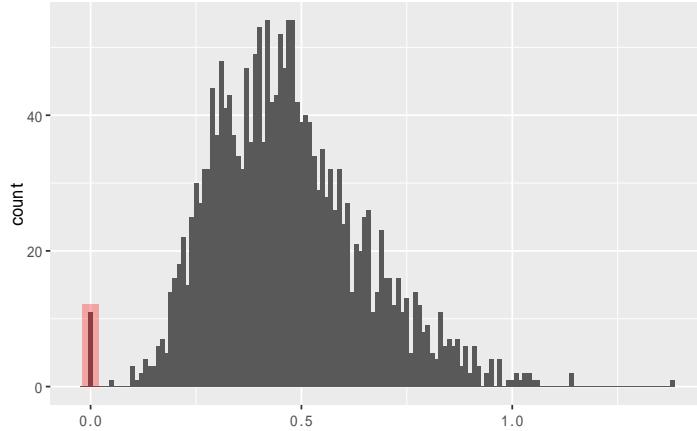
ret = Return.calculate(KOR_price)
std_12m_daily = xts::last(ret, 252) %>% apply(., 2, sd) %>%
  multiply_by(sqrt(252))
```

- 저장해둔 가격 정보와 티커 정보를 불러오도록 하며, 가격 정보의 경우 as.xts() 함수를 통해 xts 형태로 변경해주도록 합니다.
- Return.calculate() 함수를 통해 수익률을 구합니다.
- last() 함수는 마지막 n개 데이터를 선택해주는 함수이며, 1년 영업일 기준인 252개 데이터를 선택합니다. dplyr 패키지의 last() 함수와 이름이 같으므로, xts::last() 형식을 통해 xts 패키지의 함수임을 정의해줍니다.
- apply() 함수를 통해 sd 즉 변동성을 계산해주며, 연율화를 해주기 위해 multiply\_by() 함수를 통해  $\sqrt{252}$ 를 곱해주도록 합니다.

```
std_12m_daily %>%
  data.frame() %>%
  ggplot(aes(x = (`.`))) +
  geom_histogram(binwidth = 0.01) +
  annotate("rect", xmin = -0.02, xmax = 0.02,
           ymin = 0,
           ymax = sum(std_12m_daily == 0, na.rm = TRUE) * 1.1,
           alpha=0.3, fill="red") +
```

```
xlab(NULL)

std_12m_daily[std_12m_daily == 0] = NA
```

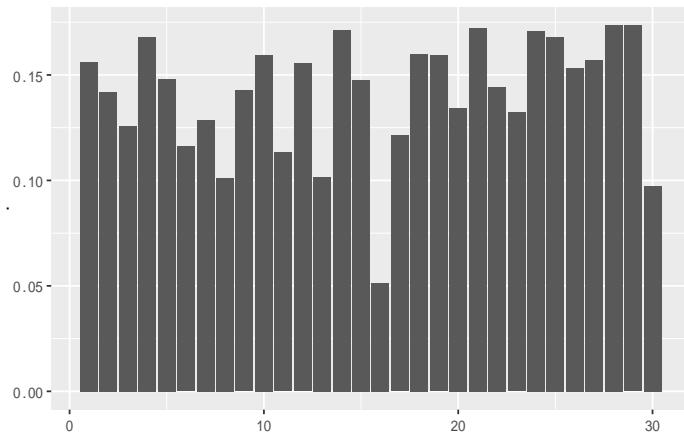


변동성을 히스토그램으로 나타내보면, 0에 위치하는 종목들이 다수 존재합니다. 해당 종목들은 최근 1년간 거래정지로 인해 가격이 변하지 않았고, 이로 인해 변동성이 없는 종목들입니다. 해당 종목들은 NA로 처리해주도록 합니다.

```
std_12m_daily[rank(std_12m_daily) <= 30]
```

```
## X030200 X001720 X015350 X017390 X034950 X015360 X092230
## 0.15627 0.14199 0.12591 0.16792 0.14813 0.11614 0.12852
## X018120 X092130 X001270 X117580 X006220 X003460 X003650
## 0.10121 0.14265 0.15961 0.11326 0.15551 0.10164 0.17117
## X007330 X034590 X040420 X023000 X000650 X003080 X107590
## 0.14762 0.05139 0.12132 0.15994 0.15922 0.13433 0.17215
## X004450 X001750 X006660 X014440 X066670 X115310 X049430
## 0.14424 0.13215 0.17100 0.16803 0.15343 0.15709 0.17370
## X025530 X066790
## 0.17350 0.09720
```

```
std_12m_daily[rank(std_12m_daily) <= 30] %>%
  data.frame() %>%
  ggplot(aes(x = rep(1:30), y = `.`)) +
  geom_col() +
  xlab(NULL)
```



`rank()` 함수를 통해 순위를 구할 수 있으며, R은 기본적으로 오름차순 즉 가장 낮은 값의 순위가 1이 됩니다. 따라서 변동성이 낮을수록 높은 순위가 되며, 30위 이하의 순위를 선택하면 변동성이 낮은 30 종목이 선택됩니다. 또한 `ggplot()` 함수를 이용해 해당 종목들의 변동성을 확인해볼 수도 있습니다.

이번에는 해당 종목들의 티커 및 종목명을 확인하도록 하겠습니다.

```
invest_lowvol = rank(std_12m_daily) <= 30
KOR_ticker[invest_lowvol, ] %>%
  select(`종목코드`, `종목명`) %>%
  mutate(`변동성` = round(std_12m_daily[invest_lowvol], 4))
```

##	종목코드	종목명	변동성
## 1	030200	KT	0.1563
## 2	001720	신영증권	0.1420
## 3	015350	부산가스	0.1259
## 4	017390	서울가스	0.1679
## 5	034950	한국기업평가	0.1481
## 6	015360	예스코홀딩스	0.1161
## 7	092230	KPX홀딩스	0.1285
## 8	018120	진로발효	0.1012
## 9	092130	이크레더블	0.1427
## 10	001270	부국증권	0.1596
## 11	117580	대성에너지	0.1133
## 12	006220	제주은행	0.1555
## 13	003460	유화증권	0.1016
## 14	003650	미창석유	0.1712
## 15	007330	푸른저축은행	0.1476

```

## 16 034590 인천도시가스 0.0514
## 17 040420 정상제이엘에스 0.1213
## 18 023000 삼원강재 0.1599
## 19 000650 천일고속 0.1592
## 20 003080 성보화학 0.1343
## 21 107590 미원홀딩스 0.1721
## 22 004450 삼화왕관 0.1442
## 23 001750 한양증권 0.1321
## 24 006660 삼성공조 0.1710
## 25 014440 영보화학 0.1680
## 26 066670 디스플레이텍 0.1534
## 27 115310 인포바인 0.1571
## 28 049430 코메론 0.1737
## 29 025530 SJM홀딩스 0.1735
## 30 066790 씨씨에스 0.0972

```

티커와 종목명, 그리고 연율화 변동성을 확인할 수 있습니다.

### 9.2.2 저변동성 포트폴리오 구하기: 주간 기준

이번에는 일간 변동성이 아닌 주간 변동성을 기준으로 저변동성 종목을 선택하도록 하겠습니다.

```

std_12m_weekly = xts::last(ret, 252) %>%
  apply.weekly(Return.cumulative) %>%
  apply(., 2, sd) %>% multiply_by(sqrt(52))

std_12m_weekly[std_12m_weekly == 0] = NA

```

먼저 최근 252일 수익률을 선택한 후, `apply.weekly()` 함수 내 `Return.cumulative`를 입력하여 주간 수익률을 계산하며, 연율화를 위해 연간 주수에 해당하는  $\sqrt{52}$ 를 곱해주도록 합니다. 이 외에도 `apply.monthly()`, `apply.yearly()` 함수 등으로 일간 수익률을 월간, 연간 수익률 등으로 변환할 수 있습니다. 그 후 과정은 위와 동일합니다.

```
std_12m_weekly[rank(std_12m_weekly) <= 30]
```

```

## X316140 X030200 X001720 X019680 X002960 X015350 X017390
## 0.15526 0.14535 0.11912 0.14722 0.13555 0.12668 0.15443

```

```

## X034950 X015360 X092230 X018120 X092130 X312610 X001270
## 0.13875 0.09979 0.11673 0.07575 0.14683 0.05745 0.16038
## X117580 X038390 X003460 X003650 X007330 X034590 X019440
## 0.12097 0.16150 0.08612 0.14894 0.15059 0.03623 0.16329
## X040420 X004450 X001750 X004080 X066670 X115310 X049430
## 0.11287 0.11352 0.12518 0.16374 0.14154 0.12913 0.13032
## X002070 X066790
## 0.14917 0.13632

```

```

invest_lowvol_weekly = rank(std_12m_weekly) <= 30
KOR_ticker[invest_lowvol_weekly, ] %>%
  select(`종목코드`, `종목명`) %>%
  mutate(`변동성` =
    round(std_12m_weekly[invest_lowvol_weekly], 4))

```

	종목코드	종목명	변동성
## 1	316140	우리금융지주	0.1553
## 2	030200	KT	0.1454
## 3	001720	신영증권	0.1191
## 4	019680	대교	0.1472
## 5	002960	한국쉘석유	0.1356
## 6	015350	부산가스	0.1267
## 7	017390	서울가스	0.1544
## 8	034950	한국기업평가	0.1387
## 9	015360	예스코홀딩스	0.0998
## 10	092230	KPX홀딩스	0.1167
## 11	018120	진로발효	0.0757
## 12	092130	이크레더블	0.1468
## 13	312610	에이에프더블류	0.0575
## 14	001270	부국증권	0.1604
## 15	117580	대성에너지	0.1210
## 16	038390	레드캡투어	0.1615
## 17	003460	유화증권	0.0861
## 18	003650	미창석유	0.1489
## 19	007330	푸른저축은행	0.1506
## 20	034590	인천도시가스	0.0362
## 21	019440	세아특수강	0.1633
## 22	040420	정상제이엘에스	0.1129
## 23	004450	삼화왕관	0.1135
## 24	001750	한양증권	0.1252

```
## 25 004080      신흥 0.1637
## 26 066670      디스플레이텍 0.1415
## 27 115310      인포바인 0.1291
## 28 049430      코메론 0.1303
## 29 002070      남영비비안 0.1492
## 30 066790      씨씨에스 0.1363
```

주간 수익률의 변동성이 낮은 30 종목을 선택하여 종목코드, 종목명, 연율화 변동성을 확인하도록 합니다.

```
intersect(KOR_ticker[invest_lowvol, '종목명'],
          KOR_ticker[invest_lowvol_weekly, '종목명'])
```

```
## [1] "KT"           "신영증권"      "부산가스"
## [4] "서울가스"     "한국기업평가"  "예스코홀딩스"
## [7] "KPX홀딩스"    "진로발효"     "이크레더블"
## [10] "부국증권"     "대성에너지"    "유화증권"
## [13] "미창석유"     "푸른저축은행" "인천도시가스"
## [16] "정상제이엘에스" "삼화왕관"    "한양증권"
## [19] "디스플레이텍"  "인포바인"    "코메론"
## [22] "씨씨에스"
```

`intersect()` 함수를 통해 일간 변동성 기준과 주간 변동성 기준 모두에 포함되는 종목을 찾을 수 있습니다.

## 9.3 모멘텀 전략

투자에서 모멘텀이란 주가 혹은 이익의 추세로써, 상승 추세의 주식은 지속적으로 상승하며 하락 추세의 주식은 지속적으로 하락하는 현상을 말합니다. 모멘텀 현상이 발생하는 원인 중 가장 큰 이유는 투자자들의 스스로에 대한 과잉 신뢰 때문입니다. 사람들은 자신의 판단을 지지하는 정보에 대해서는 과잉 반응하는, 자신의 판단을 부정하는 정보에 대해서는 과소 반응하는 경향이 있습니다. 이러한 투자자들의 비합리성으로 인해 모멘텀 현상이 생겨나게 됩니다.

모멘텀의 종류는 크게 기업의 이익에 대한 추세를 나타내는 이익 모멘텀과, 주가의 모멘텀에 대한 가격 모멘텀이 있습니다. 또한 가격 모멘텀도 1주일 혹은 1개월 이하를 의미하는 단기 모멘텀, 3개월에서 12개월을 의미하는 중기 모멘텀, 3년에서 5년을 의미하는 장기 모멘텀이 있으며, 이중에서도 3개월에서 12개월 가격 모멘텀을 흔히 모멘텀이라 합니다.

### 9.3.1 모멘텀 포트폴리오 구하기: 12개월 모멘텀

먼저 최근 1년 동안의 수익률이 높은 30 종목을 선택하도록 하겠습니다.

```
library(stringr)
library(xts)
library(PerformanceAnalytics)
library(magrittr)
library(dplyr)

KOR_price = read.csv('data/KOR_price.csv', row.names = 1,
                      stringsAsFactors = FALSE) %>% as.xts()
KOR_ticker = read.csv('data/KOR_ticker.csv', row.names = 1,
                      stringsAsFactors = FALSE)
KOR_ticker$'종목코드' =
  str_pad(KOR_ticker$'종목코드', 6, 'left', 0)

ret = Return.calculate(KOR_price) %>% xts::last(252)
ret_12m = ret %>% sapply(., function(x) {
  prod(1+x) - 1
})
```

- 가격 정보와 티커 정보를 불러온 후, `Return.calculate()` 함수를 통해 수익률을 계산합니다. 그 후, 최근 252일 수익률을 선택합니다.
- `sapply()` 함수 내부에 `prod()` 함수를 이용하여 각 종목의 누적수익률을 계산해줍니다.

```
ret_12m[rank(-ret_12m) <= 30]
```

```
## X032500 X214150 X078070 X048410 X230360 X239610 X061970
##   2.362   2.110   6.434   2.228   1.835   2.031   1.836
## X078130 X097520 X047310 X138080 X008350 X179900 X214870
##   2.004   1.439   1.524   4.521   1.836   1.651   2.827
## X143160 X176440 X263920 X009460 X263540 X190510 X037070
##   2.130   2.853   2.336   1.822   1.461   1.504   1.458
## X215090 X001140 X023770 X024060 X100030 X051160 X139670
##   1.420   2.816   3.609   1.449   2.141   3.089   2.930
## X090740 X051630
##   1.437   1.775
```

`rank()` 함수를 통해 순위를 구하도록 하며, 모멘텀의 경우 높을수록 좋은 내림차순으로 순위를 계산해야 하므로 수익률 앞에 마이너스(-)를 붙여주도록 합니다. 12개월 누적수익률이 높은 종목들이 선택됨이 확인됩니다.

```
invest_mom = rank(~ret_12m) <= 30
KOR_ticker[invest_mom, ] %>%
  select(`종목코드`, `종목명`) %>%
  mutate(`수익률` = round(ret_12m[invest_mom], 4))
```

##	종목코드	종목명	수익률
## 1	032500	케이엠더블유	2.362
## 2	214150	클래시스	2.110
## 3	078070	유비쿼스홀딩스	6.434
## 4	048410	현대바이오	2.228
## 5	230360	에코마케팅	1.835
## 6	239610	에이치엘사이언스	2.031
## 7	061970	엘비세미콘	1.836
## 8	078130	국일제지	2.004
## 9	097520	엠씨넥스	1.439
## 10	047310	파워로직스	1.524
## 11	138080	오이솔루션	4.521
## 12	008350	남선알미늄	1.836
## 13	179900	유티아이	1.651
## 14	214870	뉴지랩	2.827
## 15	143160	아이디스	2.130
## 16	176440	에이치엔티	2.853
## 17	263920	블러썸엠앤씨	2.336
## 18	009460	한창제지	1.822
## 19	263540	샘코	1.461
## 20	190510	나무가	1.504
## 21	037070	파세코	1.458
## 22	215090	리퓨어유니맥스	1.420
## 23	001140	국보	2.816
## 24	023770	플레이위드	3.609
## 25	024060	흥구석유	1.449
## 26	100030	모바일리더	2.141
## 27	051160	지어소프트	3.089
## 28	139670	키네마스터	2.930
## 29	090740	연이정보통신	1.437
## 30	051630	진양화학	1.775

티커와 종목명, 그리고 누적수익률을 확인할 수 있습니다.

### 9.3.2 모멘텀 포트폴리오 구하기: 위험조정 수익률

단순히 과거 수익률로만 모멘텀 종목을 선택할 경우, 각종 테마나 이벤트로 인한 급등으로 인해 변동성이 지나치게 높은 종목이 존재할 수도 있습니다. 누적수익률을 변동성으로 나누어 위험을 고려해줄 경우, 이러한 종목은 제외되며 상대적으로 안정적인 모멘텀 종목을 선택할 수 있습니다.

```
ret = Return.calculate(KOR_price) %>% xts::last(252)
ret_12m = ret %>% sapply(., function(x) {
  prod(1+x) - 1
})
std_12m = ret %>% apply(., 2, sd) %>% multiply_by(sqrt(252))
sharpe_12m = ret_12m / std_12m
```

1. 최근 1년에 해당하는 수익률을 선택합니다.
2. sapply()와 prod() 함수를 이용해 분자에 해당하는 누적수익률을 계산합니다.
3. apply()와 multiply\_by() 함수를 이용해 분모에 해당하는 연율화 변동성을 계산합니다.
4. 수익률을 변동성으로 나누어 위험조정 수익률을 계산해줍니다.

이를 통해 수익률이 높으면서 변동성이 낮은 종목을 선정할 수 있습니다.

```
invest_mom_sharpe = rank(-sharpe_12m) <= 30
KOR_ticker[invest_mom_sharpe, ] %>%
  select(`종목코드`, `종목명`) %>%
  mutate(`수익률` = round(ret_12m[invest_mom_sharpe], 2),
    `변동성` = round(std_12m[invest_mom_sharpe], 2),
    `위험조정 수익률` =
      round(sharpe_12m[invest_mom_sharpe], 2)) %>%
  as_tibble() %>%
  print(n = Inf)
```

		## # A tibble: 30 x 5				
		## 종목코드 종목명	수익률	변동성	위험조정 수익률	
		<chr>	<dbl>	<dbl>	<dbl>	
##	1	081660	휠라코리아	1.2	0.48	2.47

## 2 032500	케이엠더블유	2.36	0.6	3.94
## 3 091700	파트론	1.07	0.42	2.52
## 4 214150	클래시스	2.11	0.63	3.35
## 5 078070	유비쿼스홀딩스~	6.43	0.68	9.53
## 6 048410	현대바이오	2.23	1.03	2.16
## 7 029960	코엔텍	1.12	0.49	2.31
## 8 230360	에코마케팅	1.83	0.64	2.85
## 9 239610	에이치엘사이언스~	2.03	0.61	3.33
## 10 061970	엘비세미콘	1.84	0.87	2.11
## 11 078130	국일제지	2	0.91	2.2
## 12 097520	엠씨넥스	1.44	0.56	2.56
## 13 047310	파워로직스	1.52	0.61	2.49
## 14 138080	오이솔루션	4.52	0.63	7.19
## 15 008350	남선알미늄	1.84	0.77	2.4
## 16 123860	아나패스	1.06	0.47	2.25
## 17 214870	뉴지랩	2.83	0.66	4.29
## 18 143160	아이디스	2.13	0.85	2.5
## 19 176440	에이치엔티	2.85	0.84	3.38
## 20 263920	블러썸엠앤씨	2.34	0.84	2.79
## 21 009460	한창제지	1.82	0.78	2.33
## 22 190510	나무가	1.5	0.46	3.24
## 23 001140	국보	2.82	1.05	2.68
## 24 023770	플레이위드	3.61	0.8	4.51
## 25 100030	모바일리더	2.14	0.66	3.26
## 26 051160	지어소프트	3.09	0.85	3.63
## 27 139670	키네마스터	2.93	0.83	3.51
## 28 104460	동양피엔에프	1.09	0.43	2.52
## 29 051630	진양화학	1.77	0.83	2.13
## 30 050760	에스폴리텍	1.32	0.580	2.28

티커와 종목명, 누적수익률, 변동성, 위험조정 수익률을 확인할 수 있습니다.

```
intersect(KOR_ticker[invest_mom, '종목명'],
          KOR_ticker[invest_mom_sharpe, '종목명'])
```

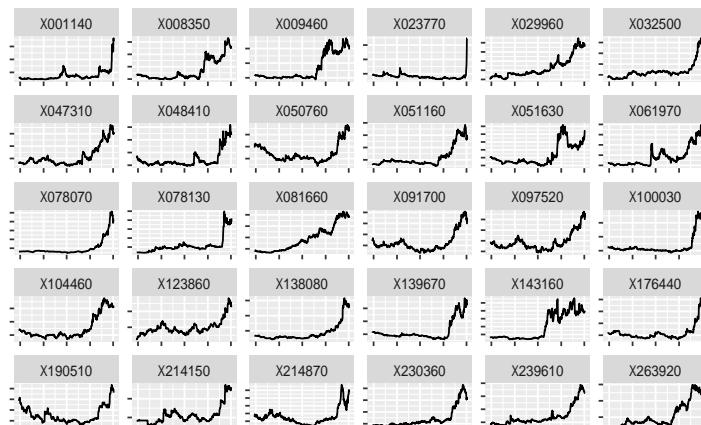
```
## [1] "케이엠더블유"      "클래시스"
## [3] "유비쿼스홀딩스"    "현대바이오"
## [5] "에코마케팅"        "에이치엘사이언스"
## [7] "엘비세미콘"         "국일제지"
## [9] "엠씨넥스"           "파워로직스"
```

```
## [11] "오이솔루션"      "남선알미늄"
## [13] "뉴지랩"           "아이디스"
## [15] "에이치엔티"     "블러썸엠앤씨"
## [17] "한창제지"       "나무가"
## [19] "국보"             "플레이위드"
## [21] "모바일리더"     "지어소프트"
## [23] "키네마스터"     "진양화학"
```

`intersect()` 함수를 통해 단순 수익률 및 위험조정 수익률 기준 모두에 포함되는 종목을 찾을 수 있습니다. 다음 그림은 위험조정 수익률 상위 30 종목의 가격 그래프입니다.

```
library(xts)
library(tidyr)
library(ggplot2)

KOR_price[, invest_mom_sharpe] %>%
  fortify.zoo() %>%
  gather(ticker, price, -Index) %>%
  ggplot(aes(x = Index, y = price)) +
  geom_line() +
  facet_wrap(. ~ ticker, scales = 'free') +
  xlab(NULL) +
  ylab(NULL) +
  theme(axis.text.x=element_blank(),
        axis.text.y=element_blank())
```



## 9.4 밸류 전략

가치주 효과란 내재 가치 대비 낮은 가격의 주식(저 PER, 저 PBR 등)이, 내재 가치 대비 비싼 주식보다 수익률이 높은 현상을 뜻합니다. 가치 효과가 발생하는 원인에 대한 이론은 다음과 같습니다.

1. 위험한 기업은 시장에서 상대적으로 낮은 가격에 거래되며, 이러한 위험을 감당하는 대가로 수익이 발생
2. 투자자들의 성장주에 대한 과잉 반응으로 인해 가치주는 시장에서 소외되며, 제자리를 찾아가는 과정에서 수익이 발생

기업의 가치를 나타내는 지표는 굉장히 많지만, 일반적으로 PER, PBR, PCR, PSR가 많이 사용됩니다.

### 9.4.1 밸류 포트폴리오 구하기: 저 PBR

먼저 기업의 가치 여부를 판단할 때 가장 많이 사용되는 지표인 PBR을 이용한 포트폴리오를 구성하도록 하겠습니다.

```
library(stringr)
library(ggplot2)
library(dplyr)

KOR_value = read.csv('data/KOR_value.csv', row.names = 1,
                      stringsAsFactors = FALSE)
KOR_ticker = read.csv('data/KOR_ticker.csv', row.names = 1,
                      stringsAsFactors = FALSE)
KOR_ticker$'종목코드' =
  str_pad(KOR_ticker$'종목코드', 6, 'left', 0)

invest_pbr = rank(KOR_value$PBR) <= 30
KOR_ticker[invest_pbr, ] %>%
  select(`종목코드`, `종목명`) %>%
  mutate(`PBR` = round(KOR_value[invest_pbr, 'PBR'], 4))
```

## 종목코드	종목명	PBR
## 1 088350	한화생명	0.2318
## 2 000880	한화	0.1212
## 3 034020	두산중공업	0.2028

```

## 4    006120      SK디스커버리 0.2315
## 5    058650      세아홀딩스 0.1228
## 6    032190      다우데이타 0.1375
## 7    005720      넥센 0.1834
## 8    003300      한일홀딩스 0.1856
## 9    001940      KISCO홀딩스 0.2044
## 10   002030      아세아 0.1773
## 11   036530      S&T홀딩스 0.1665
## 12   092230      KPX홀딩스 0.2199
## 13   003030      세아제강지주 0.1788
## 14   000140  하이트진로홀딩스 0.2080
## 15   033160      엠페이지전자 0.2107
## 16   035080      인터파크홀딩스 0.1895
## 17   009200      무림페이퍼 0.1951
## 18   007860      서연 0.1151
## 19   002300      한국제지 0.1950
## 20   005010      휴스틸 0.2328
## 21   040610      SG&G 0.1291
## 22   025530      SJM홀딩스 0.2030
## 23   031980  피에스케이홀딩스 0.1797
## 24   006200      한국전자홀딩스 0.1420
## 25   000950      전방 0.2298
## 26   037400      우리조명 0.1818
## 27   017680      데코앤이 0.0885
## 28   194510      파티게임즈 0.2083
## 29   192410      감마누 0.2165
## 30   149940      모다 0.0539

```

먼저 가치 지표들을 저장한 데이터와 티커 데이터를 불러오도록 하며, `rank()`를 통해 PBR이 낮은 30 종목을 선택해주도록 합니다. 그 후 종목코드와 종목명, PBR을 확인해보도록 합니다. 홀딩스 등 지주사가 그 특성상 저 PBR 포트폴리오에 많이 구성되어 있습니다.

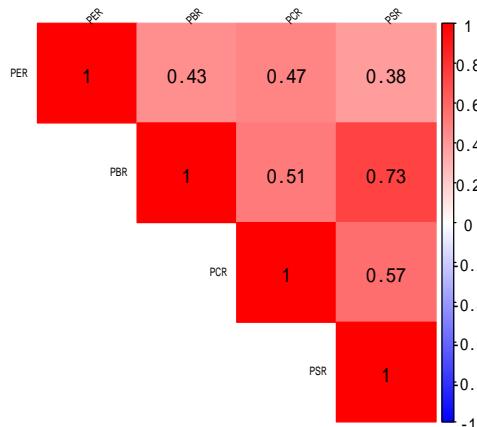
#### 9.4.2 각 지표를 결합하기

저 PBR 하나의 지표만으로도 우수한 성과를 거둘 수 있음은 오랜 기간동안 증명되고 있습니다. 그러나 저평가 주식이 계속해서 저평가에 머무르는 가치 함정에 빠지지 않기 위해서는, 여러 지표를 동시에 볼 필요도 있습니다.

```
library(corrplot)

rank_value = KOR_value %>%
  mutate_all(list(~min_rank(.)))

cor(rank_value, use = 'complete.obs') %>%
  round(., 2) %>%
  corrplot(method = 'color', type = 'upper',
           addCoef.col = 'black', number.cex = 1,
           tl.cex = 0.6, tl.srt=45, tl.col = 'black',
           col = colorRampPalette(
             c('blue', 'white', 'red'))(200),
           mar=c(0,0,0.5,0))
```



먼저 `mutate_all()` 함수를 이용해 모든 열에 함수를 적용해주며, `min_rank()`를 통해 순위를 구해주도록 합니다.

각 열에 해당하는 가치 지표 별 랭킹을 구해준 후 상관관계를 확인해보도록 하며, NA 종목은 삭제해주기 위해 `use = 'complete.obs'`를 입력해주도록 합니다.

`corrplot` 패키지의 `corrplot()` 함수를 이용해 상관관계를 그려보면, 같은 가치 지표임에도 불구하고 서로간의 상관관계가 꽤 낮은 지표도 존재하여, 지표를 통합적으로 고려시 분산효과를 기대할 수도 있습니다.

```
rank_sum = rank_value %>%
  rowSums()

invest_value = rank(rank_sum) <= 30
```

```
KOR_ticker[invest_value, ] %>%
  select(`종목코드`, `종목명`) %>%
  cbind(round(KOR_value[invest_value, ], 2))
```

##	종목코드	종목명	PER	PBR	PCR	PSR
## 19	034730	SK	7.23	0.32	2.07	0.16
## 87	001040	CJ	10.57	0.23	1.89	0.10
## 106	000880	한화	4.38	0.12	0.75	0.04
## 159	042670	두산인프라코어	5.15	0.33	1.52	0.16
## 288	006840	AK홀딩스	5.42	0.39	1.83	0.16
## 394	017940	E1	5.08	0.30	2.15	0.09
## 417	058650	세아홀딩스	11.26	0.12	2.62	0.07
## 444	005720	넥센	5.55	0.18	2.55	0.25
## 471	015750	성우하이텍	13.37	0.25	1.20	0.09
## 485	003300	한일홀딩스	0.63	0.19	2.53	0.27
## 552	084690	대상홀딩스	11.40	0.24	1.88	0.08
## 588	002030	아세아	4.90	0.18	1.03	0.15
## 589	036530	S&T홀딩스	9.13	0.17	1.94	0.18
## 596	013580	계룡건설	2.73	0.55	2.34	0.11
## 616	003030	세아제강지주	0.76	0.18	1.90	0.14
## 745	033160	엠케이전자	7.07	0.21	3.52	0.25
## 748	005990	매일홀딩스	7.08	0.35	2.48	0.12
## 887	267290	경동도시가스	4.38	0.47	1.24	0.09
## 963	009200	무림페이퍼	3.75	0.20	1.52	0.12
## 984	016710	대성홀딩스	6.58	0.24	2.28	0.14
## 1004	005710	대원산업	4.44	0.45	1.89	0.18
## 1085	036000	예림당	7.48	0.31	3.39	0.15
## 1204	003480	한진중공업홀딩스	11.02	0.27	1.73	0.10
## 1262	005010	휴스틸	5.49	0.23	0.83	0.16
## 1332	002200	수출포장	4.91	0.37	2.02	0.30
## 1402	037350	성도이엔지	5.07	0.40	1.57	0.15
## 1484	004140	동방	4.45	0.52	2.39	0.13
## 1717	002710	TCC스틸	5.52	0.48	2.51	0.13
## 1759	031980	피에스케이홀딩스	0.91	0.18	1.18	0.15
## 1890	012620	원일특강	6.09	0.38	4.34	0.16

`rowSums()` 함수를 이용해 종목 별 랭킹들의 합을 구해줍니다. 그 후 4개 지표 랭킹의 합 기준 랭킹이 낮은 30 종목을 선택해 줍니다. 즉 하나의 지표 보다 4개 지표가 골고루 낮은 종목을 선택하여 줍니다. 해당 종목들의 티커, 종목명과 가치 지표들을 확인할 수 있습니다.

```
intersect(KOR_ticker[invest_pbr, '종목명'],
          KOR_ticker[invest_value, '종목명'])
```

```
## [1] "한화"           "세아홀딩스"
## [3] "넥센"           "한일홀딩스"
## [5] "아세아"         "S&T홀딩스"
## [7] "세아제강지주"   "엠케이전자"
## [9] "무림페이퍼"     "휴스틸"
## [11] "피에스케이홀딩스"
```

단순 저 PBR 기준 선택된 종목과 비교해봤을 때, 겹치는 종목이 상당히 줄어들었습니다.

## 9.5 퀄리티 전략

기업의 우량성, 즉 퀄리티는 투자자들이 매우 중요하게 생각하는 요소입니다. 그러나 어떠한 지표가 기업의 퀄리티를 나타내는지 한마디로 정의하기에는 너무나 주관적이고 광범위하여 쉽지 않습니다. 학계 혹은 업계에서 사용되는 우량성 관련 지표는 다음과 같이 요약할 수 있습니다.

1. Profitability (수익성)
2. Earnings stability (수익의 안정성)
3. Capital structure (기업 구조)
4. Growth (수익의 성장성)
5. Accounting quality (회계적 우량성)
6. Payout/dilution (배당)
7. Investment (투자)

퀄리티 전략에는 재무제표 데이터가 주로 사용됩니다.

### 9.5.1 F-Score

F-Score 지표는 조셉 피오토로스키 교수가 발표한 지표입니다. 그는 논문에서, 저 PBR을 이용한 벤류 전략은 높은 성과를 기록하지만 재무 상태가 불량한 기업이 많으며, 저 PBR 종목 중 재무적으로 우량한 기업을 선정하여 투자한다면 성과를 훨씬 개선할 수 있다고 보았습니다.

Table 9.4: F-Score 요약

지표	항목	점수
Profitability	$ROA$	ROA가 양수면 1점
	$CFO$	CFO가 양수면 1점
	$\Delta ROA$	ROA가 증가했으면 1점
	$ACCRUAL$	$CFO > ROA$ 면 1점
Financial Performance	$\Delta LEVER$	레버리지가 감소했으면 1점
	$\Delta LIQUID$	유동성이 증가했으면 1점
	$EQ\_OFFER$	발행주식수가 감소했으면 1점
Operating Efficiency	$\Delta MARGIN$	매출총이익률이 증가했으면 1점
	$\Delta TURN$	회전율이 증가했으면 1점

F-Score에서는 재무적 우량 정도를 수익성(Profitability), 재무 성과(Financial Performance), 운영 효율성(Operating Efficiency)으로 구분하여 총 9개의 지표를 선정합니다. 표 9.4는 이를 요약한 테이블입니다.

각 지표가 우수할 경우 1점, 그렇지 않을 경우 0점을 매겨, 총 0점부터 9점까지의 포트폴리오를 구성합니다.

$$F\_SCORE = F\_ROA + F\_{\Delta}ROA + F\_CFO + F\_{\Delta}ACCRUAL + F\_{\Delta}MARGIN \\ + F\_{\Delta}TURN + F\_{\Delta}LEVER + F\_{\Delta}LIQUID + F\_EQ\_OFFER$$

```
library(stringr)
library(ggplot2)
library(dplyr)

KOR_fs = readRDS('data/KOR_fs.Rds')
KOR_ticker = read.csv('data/KOR_ticker.csv', row.names = 1,
                      stringsAsFactors = FALSE)
KOR_ticker$'종목코드' =
  str_pad(KOR_ticker$'종목코드', 6, 'left', 0)
```

먼저 재무제표와 티커 파일을 불러오도록 합니다. 재무제표 데이터의 경우 RdS 형태로 저장되어 있으며, `readRDS()` 함수를 이용해 list 형태 그래도 불러올 수 있습니다.

```

# 수익성
ROA = KOR_fs$'지배주주순이익' / KOR_fs$'자산'
CFO = KOR_fs$'영업활동으로인한현금흐름' / KOR_fs$'자산'
ACCURUAL = CFO - ROA

# 재무성과
LEV = KOR_fs$'장기차입금' / KOR_fs$'자산'
LIQ = KOR_fs$'유동자산' / KOR_fs$'유동부채'
OFFER = KOR_fs$'유상증자'

# 운영 효율성
MARGIN = KOR_fs$'매출총이익' / KOR_fs$'매출액'
TURN = KOR_fs$'매출액' / KOR_fs$'자산'

```

먼저 지표에 해당하는 내용을 계산해줍니다.

1. ROA는 지배주주순이익을 자산으로 나누어 계산합니다.
2. CFO는 영업활동현금흐름을 자산으로 나누어 계산합니다.
3. ACCURUAL은 CFO와 ROA의 차이를 이용해 계산합니다.
4. Leverage는 장기차입금을 자산으로 나누어 계산합니다.
5. Liquidity는 유동자산을 유동부채로 나누어 계산합니다.
6. 우리가 받은 데이터에서는 발행주식수 데이터를 구할수 없으므로, Offer에 대한 대용치로 유상증자 여부를 사용합니다.
7. Margin은 매출총이익을 매출액으로 나누어 계산합니다.
8. Turnover는 매출액을 자산으로 나누어 계산합니다.

다음으로 각 지표들이 조건을 충족하는지 여부를 판단하여, 지표 별로 1점 혹은 0점을 부여해줍니다.

```

num_col = ncol(KOR_fs[[1]])

F_1 = as.integer(ROA[, num_col] > 0)
F_2 = as.integer(CFO[, num_col] > 0)
F_3 = as.integer(ROA[, num_col] - ROA[, (num_col-1)] > 0)
F_4 = as.integer(ACCURUAL[, num_col] > 0)
F_5 = as.integer(LEV[, num_col] - LEV[, (num_col-1)] <= 0)
F_6 = as.integer(LIQ[, num_col] - LIQ[, (num_col-1)] > 0)
F_7 = as.integer(is.na(OFFER[, num_col]) |
                  OFFER[, num_col] <= 0)

```

```
F_8 = as.integer(MARGIN[, num_col] -
                  MARGIN[, (num_col-1)] > 0)
F_9 = as.integer(TURN[,num_col] - TURN[, (num_col-1)] > 0)
```

먼저 ncol() 함수를 이용해 열 갯수를 구해줍니다. 가장 최근 연도의 재무제표가 최우측에 위치하고 있으므로, 해당 변수를 통해 최근 연도 데이터만을 선택할 수 있습니다.

as.integer() 함수는 TRUE일 경우 1, FALSE 일 경우 0을 반환하는 함수로써, F-Score 지표의 점수를 매기는데 매우 유용합니다. 점수 기준은 다음과 같습니다.

1. ROA가 양수면 1점, 그렇지 않으면 0점
2. 영업활동현금흐름이 양수면 1점, 그렇지 않으면 0점
3. 최근 ROA가 전년 대비 증가했으면 1점, 그렇지 않으면 0점
4. Accrual(CFO - ROA)이 양수면 1점, 그렇지 않으면 0점
5. 레버리지가 전년 대비 감소했으면 1점, 그렇지 않으면 0점
6. 유동성이 전년 대비 증가해으면 1점, 그렇지 않으면 0점
7. 유상증자 항목이 없거나 0보다 작으면 1점, 그렇지 않으면 0점
8. 매출총이익률이 전년 대비 증가했으면 1점, 그렇지 않으면 0점
9. 회전율이 전년 대비 증가했으면 1점, 그렇지 않으면 0점

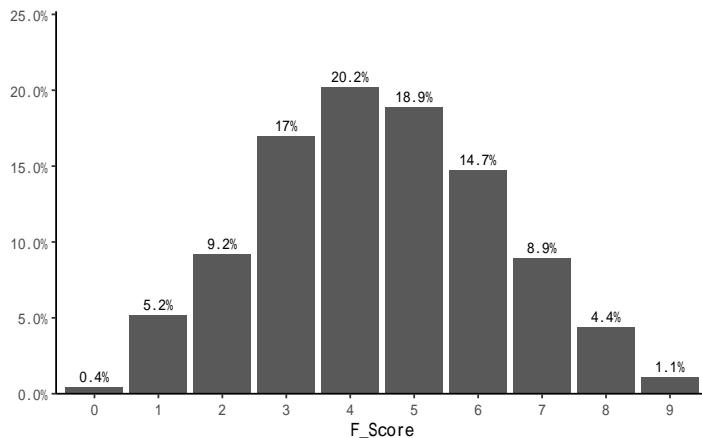
```
F_Table = cbind(F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8, F_9)
F_Score = F_Table %>%
  apply(., 1, sum, na.rm = TRUE) %>%
  setNames(KOR_ticker$`종목명`)
```

1. cbind()를 통해 열의 형태로 묶어줍니다.
2. apply() 함수를 통해 종목 별 지표의 합을 더해 F-Score를 계산해줍니다.
3. setNames() 함수를 통해 종목명을 입력해 줍니다.

```
(F_dist = prop.table(table(F_Score)) %>% round(3))
```

```
## F_Score
##      0      1      2      3      4      5      6      7      8
## 0.004 0.052 0.092 0.170 0.202 0.189 0.147 0.089 0.044
##      9
## 0.011
```

```
F_dist %>%
  data.frame() %>%
  ggplot(aes(x = F_Score, y = Freq,
             label = paste0(Freq * 100, '%'))) +
  geom_bar(stat = 'identity') +
  geom_text(color = 'black', size = 3, vjust = -0.4) +
  scale_y_continuous(expand = c(0, 0, 0, 0.05),
                     labels = scales::percent) +
  ylab(NULL) +
  theme_classic()
```



table() 함수를 통해 각 스코어 별 갯수를 구한 후, prop.table()을 통해 비중으로 변환합니다. 이를 통해 점수 별 비중을 살펴보면 3~6점에 상당히 많은 종목이 분포하고 있음이 확인됩니다.

```
invest_F_Score = F_Score %in% c(9)
KOR_ticker[invest_F_Score, ] %>%
  select(`종목코드`, `종목명`) %>%
  mutate(`F-Score` = F_Score[invest_F_Score])
```

	종목코드	종목명	F-Score
## 1	051900	LG생활건강	9
## 2	081660	휠라코리아	9
## 3	271560	오리온	9
## 4	031430	신세계인터내셔날	9
## 5	285130	SK케미칼	9
## 6	011280	태림포장	9

```

## 7    036540      SFA반도체      9
## 8    044340      위닉스        9
## 9    004690      삼천리        9
## 10   002310      아세아제지    9
## 11   232140      와이아이케이  9
## 12   023600      삼보판지    9
## 13   203650      드림시큐리티  9
## 14   089010      켐트로닉스    9
## 15   007980      태평양물산    9
## 16   009200      무림페이퍼    9
## 17   006580      대양제지    9
## 18   008250      이건산업    9
## 19   174880      장원테크    9
## 20   002200      수출포장    9
## 21   005670      푸드웰    9
## 22   091340      S&K폴리텍    9
## 23   080580      오킨스전자    9

```

F-Score가 9점인 종목의 티커와 종목명을 확인해봅니다. 재무적으로 우량하다고 판단되는 F-Score 9점인 종목은 총 23개가 존재하고 있습니다.

### 9.5.2 각 지표를 결합하기

이번에는 웰리티를 측정하는 요소 중 가장 널리 사용되는 수익성 지표를 결합한 포트폴리오를 만들어 보겠으며, 사용되는 지표는 자기자본이익률(ROE), 매출총이익(Gross Profit), 영업활동현금흐름(Cashflow From Operating)입니다.

```

library(stringr)
library(ggplot2)
library(dplyr)
library(tidyr)

KOR_fs = readRDS('data/KOR_fs.Rds')
KOR_ticker = read.csv('data/KOR_ticker.csv', row.names = 1,
                      stringsAsFactors = FALSE)
KOR_ticker$'종목코드' =
  str_pad(KOR_ticker$'종목코드', 6, 'left', 0)

num_col = ncol(KOR_fs[[1]])

```

```

quality_roe = (KOR_fs$'지배주주순이익' / KOR_fs$'자본')[num_col]
quality_gpa = (KOR_fs$'매출총이익' / KOR_fs$'자산')[num_col]
quality_cfo =
  (KOR_fs$'영업활동으로인한현금흐름' / KOR_fs$'자산')[num_col]

quality_profit =
  cbind(quality_roe, quality_gpa, quality_cfo) %>%
  setNames(., c('ROE', 'GPA', 'CFO'))

```

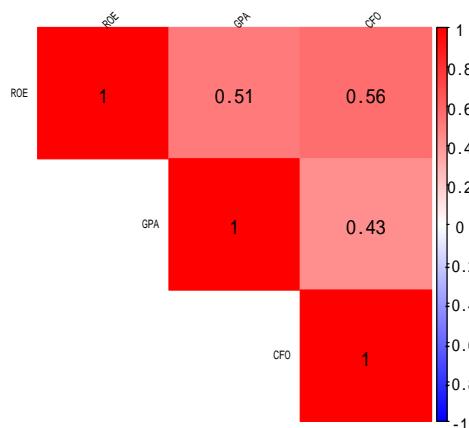
먼저 재무제표와 티커 파일을 불러온 후, 세가지 지표에 해당하는 값을 구한 뒤 최근년도 데이터만을 선택합니다. 그 후, cbind() 함수를 이용해 지표들을 하나로 뭉어줍니다.

```

rank_quality = quality_profit %>%
  mutate_all(list(~min_rank(desc(.)))))

cor(rank_quality, use = 'complete.obs') %>%
  round(., 2) %>%
  corrplot(method = 'color', type = 'upper',
            addCoef.col = 'black', number.cex = 1,
            tl.cex = 0.6, tl.srt = 45, tl.col = 'black',
            col =
              colorRampPalette(c('blue', 'white', 'red'))(200),
            mar=c(0,0,0.5,0))

```



mutate\_all() 함수와 min\_rank() 함수를 통해 지표 별 랭킹을 구해주도록 하며, 퀄리티 지표의 경우 높을수록 좋은 내림차순으로 계산하여야 하므로 desc() 을 추가해줍니다.

수익성 지표 역시 서로 간의 상관관계가 낮아, 지표를 통합적으로 고려시 분산효과를 기대할 수 있습니다.

```
rank_sum = rank_quality %>%
  rowSums()

invest_quality = rank(rank_sum) <= 30

KOR_ticker[invest_quality, ] %>%
  select(`종목코드`, `종목명`) %>%
  cbind(round(quality_profit[invest_quality, ], 4))
```

	종목코드	종목명	ROE	GPA	CFO
## 2	000660	SK하이닉스	0.3317	0.3969	0.3492
## 11	051900	LG생활건강	0.1900	0.7679	0.1549
## 46	021240	웅진코웨이	0.3220	0.7689	0.2266
## 72	282330	BGF리테일	0.2956	0.6877	0.2332
## 93	086900	메디톡스	0.2722	0.3828	0.1395
## 114	012510	더존비즈온	0.2311	0.4560	0.2228
## 175	192080	더블유게임즈	0.1694	0.4846	0.1568
## 193	030190	NICE평가정보	0.1930	1.4316	0.1843
## 232	214150	클래스	0.2922	0.4584	0.2114
## 245	067160	아프리카TV	0.2325	0.8038	0.2463
## 272	090460	비에이치	0.4343	0.3265	0.3429
## 283	069080	웹젠	0.1602	0.5517	0.2072
## 308	001820	삼화콘덴서	0.4851	0.4627	0.2768
## 369	042700	한미반도체	0.2287	0.3935	0.1854
## 387	192440	슈피겐코리아	0.1634	0.6277	0.1280
## 392	092730	네오팜	0.2597	0.6626	0.2222
## 398	215200	메가스터디교육	0.1894	0.5539	0.2500
## 461	119860	다나와	0.1822	0.9806	0.1531
## 497	034950	한국기업평가	0.1579	0.6442	0.1737
## 586	086390	유니테스트	0.3714	0.5698	0.3391
## 611	220630	해마로푸드서비스	0.2345	0.6656	0.1499
## 643	092130	이크레더블	0.2599	0.6572	0.2367
## 744	232140	와이아이케이	0.2460	0.2868	0.2306
## 797	036810	에프에스티	0.1770	0.3460	0.2114
## 913	225190	삼양옵틱스	0.3463	0.5934	0.3209
## 975	130580	나이스디앤비	0.2058	0.9171	0.2011
## 1087	241790	오션브릿지	0.2445	0.2841	0.3604

```
## 1295 285490 노바텍 0.2219 0.3133 0.1898
## 1369 308100 까스텔바작 0.1783 0.7352 0.1349
## 1603 063760 이엘피 0.2226 0.3052 0.2234
```

`rowSums()` 함수를 이용해 종목 별 랭킹들의 합을 구해주도록 합니다. 그 후 3개 지표 랭킹의 합 기준 랭킹이 낮은 30 종목을 선택해 줍니다. 즉 세가지 수익지표가 골고루 높은 종목을 선택합니다. 해당 종목들의 티커, 종목명, ROE, GPA, CFO을 출력하여 확인하도록 합니다.



# CHAPTER 10

---

## 퀀트 전략을 이용한 종목선정 (심화)

---

지난 장에서는 팩터를 이용한 투자 전략의 기본이 되는 로우볼, 모멘텀, 벨류, 웰리티 전략에 대해 알아보았습니다. 물론 이러한 단일 팩터를 이용한 투자도 장기적으로 우수한 성과를 보이지만, 여러 팩터를 결합하거나 정밀하게 전략을 만든다면 더욱 우수한 성과를 거둘 수 있습니다.

이번 장에서는 섹터 별 효과를 없앤 후 포트폴리오를 구성하는 방법, 이상치 데이터 제거 및 팩터 결합 방법, 그리고 멀티팩터 구성방법에 대해 알아보겠습니다.

### 10.1 섹터 중심 포트폴리오

팩터 전략의 단점 중 하나는 선택된 종목들이 특정 섹터로 쏠리는 경우가 있다는 점입니다. 특히 과거 수익률을 토대로 종목을 선정하는 모멘텀 전략의 경우, 특정 섹터의 호황기에 동일한 섹터의 모든 종목이 함께 움직이는 경향이 있어 이러한 쏠림이 심할 수 있습니다.

먼저 지난 장에서 배운 12개월 모멘텀을 이용한 포트폴리오 구성 방법을 다시 살펴보도록 하겠습니다.

```
library(stringr)
library(xts)
library(PerformanceAnalytics)
```

```

library(dplyr)
library(ggplot2)

KOR_price = read.csv('data/KOR_price.csv', row.names = 1,
                      stringsAsFactors = FALSE) %>% as.xts()
KOR_ticker = read.csv('data/KOR_ticker.csv', row.names = 1,
                      stringsAsFactors = FALSE)
KOR_ticker$'종목코드' =
  str_pad(KOR_ticker$'종목코드', 6, 'left', 0)

ret = Return.calculate(KOR_price) %>% xts::last(252)
ret_12m = ret %>% sapply(., function(x) {
  prod(1+x) - 1
})

invest_mom = rank(-ret_12m) <= 30

```

기준의 코드와 동일하게, 주식 가격 및 티커 데이터를 불러온 후, 최근 12개월 수익률을 구해 상위 30 종목을 선택합니다.

```

KOR_sector = read.csv('data/KOR_sector.csv', row.names = 1,
                      stringsAsFactors = FALSE)
KOR_sector$'CMP_CD' =
  str_pad(KOR_sector$'CMP_CD', 6, 'left', 0)
data_market = left_join(KOR_ticker, KOR_sector,
                        by = c('종목코드' = 'CMP_CD',
                               '종목명' = 'CMP_KOR'))

```

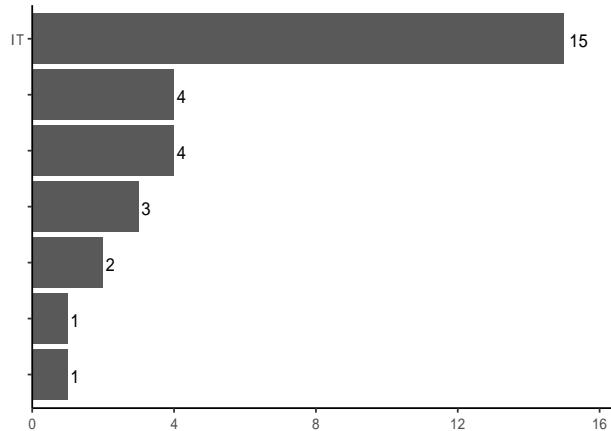
해당 종목들의 섹터 정보를 추가로 살펴보기 위해, 섹터 데이터를 불러온 후, `left_join()` 함수를 이용해 티커와 결합하여 `data_market`에 저장해줍니다.

```

data_market[invest_mom, ] %>%
  select(`SEC_NM_KOR`) %>%
  group_by(`SEC_NM_KOR`) %>%
  summarize(n = n()) %>%
  ggplot(aes(x = reorder(`SEC_NM_KOR`, `n`),
             y = `n`, label = n)) +
  geom_col() +
  geom_text(color = 'black', size = 4, hjust = -0.3) +
  xlab(NULL) +

```

```
ylab(NULL) +
coord_flip() +
scale_y_continuous(expand = c(0, 0, 0.1, 0)) +
theme_classic()
```



`group_by()` 함수를 이용하여 12개월 기준 모멘텀 포트폴리오 종목들의 섹터 별 종목수를 계산해준 후, `ggplot()` 함수를 이용하여 이를 그림으로 나타냅니다. 그림에서 알 수 있듯이, 특정 섹터에 대부분의 종목이 몰려있습니다.

따라서 여러 종목으로 포트폴리오를 구성하였지만, 이를 분해해보면 특정 섹터에 쏠림이 심하다는 것을 알 수 있습니다. 이러한 섹터 쏠림 현상을 제거한 섹터 중립 포트폴리오를 구성해 보도록 하겠습니다.

```
sector_neutral = data_market %>%
  select(`종목코드`, `SEC_NM_KOR`) %>%
  mutate(`ret` = ret_12m) %>%
  group_by(`SEC_NM_KOR`) %>%
  mutate(scale_per_sector = scale(`ret`),
        scale_per_sector = ifelse(is.na(`SEC_NM_KOR`),
                                   NA, scale_per_sector))
```

1. `data_market`에서 종목코드와 섹터정보를 선택합니다.
2. `mutate()` 함수를 통해 미리 계산한 12개월 수익률 정보를 새로운 열에 합쳐줍니다.
3. `group_by()` 함수를 통해 섹터 별 그룹을 만들어 줍니다.
4. `scale()` 함수를 이용해 그룹 별 정규화를 해줍니다. 정규화의 경우  $\frac{x-\mu}{\sigma}$ 로 계산됩니다.

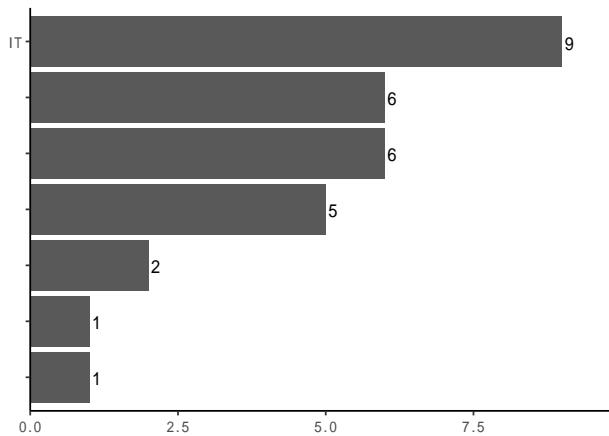
### 5. 섹터 정보가 없는 정보는 삭제해주도록 합니다.

위의 정규화 과정을 살펴보면, 전체 종목에서 12개월 수익률을 비교하는 것이 아닌 각 섹터별로 수익률의 강도를 비교하게 됩니다. 따라서 특정 종목의 과거 수익률이 전체 종목과 비교해서 높았어도 해당 섹터 내에서의 순위가 낮다면, 정규화된 값은 낮게됩니다.

따라서 섹터 별 정규화 과정을 거친 값으로 비교 분석을 한다면, 섹터 효과가 제거된 포트폴리오를 구성할 수 있습니다.

```
invest_mom_neutral =
  rank(~sector_neutral$scale_per_sector) <= 30

data_market[invest_mom_neutral, ] %>%
  select(`SEC_NM_KOR`) %>%
  group_by(`SEC_NM_KOR`) %>%
  summarize(n = n()) %>%
  ggplot(aes(x = reorder(`SEC_NM_KOR`, `n`),
             y = `n`, label = n)) +
  geom_col() +
  geom_text(color = 'black', size = 4, hjust = -0.3) +
  xlab(NULL) +
  ylab(NULL) +
  coord_flip() +
  scale_y_continuous(expand = c(0, 0, 0.1, 0)) +
  theme_classic()
```



정규화된 값의 랭킹이 높은 상위 30 종목을 선택하며, 내림차순을 위해 마이너스를 붙여줍니다. 해당 포트폴리오의 섹터 별 구성종목을 확인해보면, 단순하게 포트폴리오를 구성한 것 대비, 여러 섹터에 종목이 분산되어 있습니다.

이처럼 `group_by()` 함수를 통해 손쉽게 그룹별 중립화를 할 수 있으며, 글로벌 투자를 하는 경우에는 지역, 국가, 섹터 별로도 중립화된 포트폴리오를 구성하기도 합니다.

## 10.2 마법공식

하나의 팩터만을 보고 투자하는 것 보다, 둘 혹은 그 이상의 팩터를 결합하여 투자하는 것이 훨씬 좋은 포트폴리오를 구성할 수 있으며, 이러한 방법을 멀티팩터라 합니다. 그중에서도 밸류와 퀄리티의 조합은 전통적으로 많이 사용된 방법이며, 그 중 대표적인 예가 조엘 그린블라트의 마법공식입니다.

이번 장에서는 퀄리티와 밸류 간의 관계, 그리고 마법공식의 정의와 구성방법에 대해 알아보도록 하겠습니다.

### 10.2.1 퀄리티와 밸류 간의 관계

투자의 정석 중 하나는 **좋은 기업을 싸게 사는 것입니다**. 이를 팩터의 관점에서 이해하면 퀄리티 팩터와 밸류 팩터로 이해할 수도 있습니다.

여러 논문에 따르면흔히 밸류와 퀄리티 팩터는 반대의 관계에 있습니다. 먼저 가치주들은 위험이 크기 때문에 시장에서 소외를 받아 저평가가 이루어지는 것이며, 이러한 위험에 대한 댓가로 밸류 팩터의 수익률이 높게됩니다. 반대로 사람들은 우량주에 기꺼이 프리미엄을 지불하려 하기 때문에 퀄리티 팩터의 수익률이 높기도 합니다. 이는 마치 동전의 양면과 같지만, 장기적으로 가치주와 우량주 모두 우수한 성과를 기록합니다.

먼저 퀄리티의 지표인 매출총이익과 밸류 지표인 PBR을 통해 둘간의 관계를 확인해보도록 하겠습니다.

```
library(stringr)
library(dplyr)

KOR_value = read.csv('data/KOR_value.csv', row.names = 1,
                      stringsAsFactors = FALSE)
KOR_fs = readRDS('data/KOR_fs.Rds')
KOR_ticker = read.csv('data/KOR_ticker.csv', row.names = 1,
                      stringsAsFactors = FALSE)

data_pbr = KOR_value['PBR']
data_gpa =
```

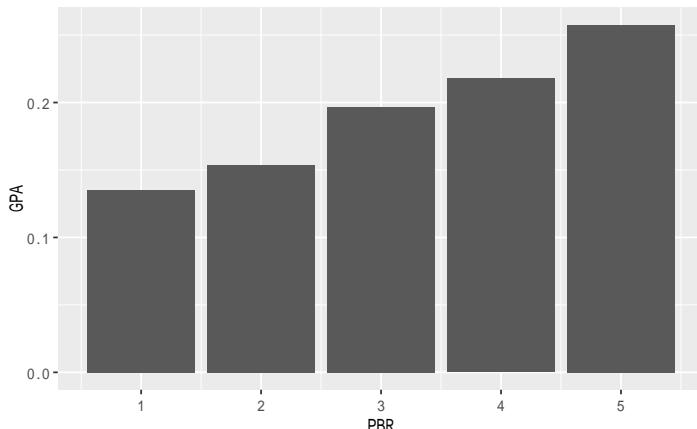
```
(KOR_fs$'매출총이익' / KOR_fs$'자산')[ncol(KOR_fs[[1]])] %>%
  setNames('GPA')

cbind(data_pbr, -data_gpa) %>%
  cor(method = 'spearman', use = 'complete.obs') %>% round(4)
```

```
##          PBR      GPA
## PBR  1.0000 -0.2078
## GPA -0.2078  1.0000
```

데이터를 불러온 후, PBR과 GPA(매출총이익 / 자산)를 구해주도록 합니다. 그 후 랭킹의 상관관계인 스피어만 상관관계를 구해보면, 서로 간 반대 관계가 있음이 확인됩니다. PBR의 경우 오름차순, GPA의 경우 내림차순 이므로 GPA 앞에 마이너스를 붙여주었습니다.

```
cbind(data_pbr, data_gpa) %>%
  mutate(quintile_pbr = ntile(data_pbr, 5)) %>%
  filter(!is.na(quintile_pbr)) %>%
  group_by(quintile_pbr) %>%
  summarise(mean_gpa = mean(GPA, na.rm = TRUE)) %>%
  ggplot(aes(x = quintile_pbr, y = mean_gpa)) +
  geom_col() +
  xlab('PBR') + ylab('GPA')
```



이번에는 PBR의 분위수 별 GPA 평균값을 구하도록 하겠습니다.

1. `ntile()` 함수를 이용해 PBR을 5분위수로 나누어 줍니다.

2. PBR이 없는 종목은 제외합니다.
3. `group_by()`를 통해 PBR의 분위수별 그룹을 끓어 줍니다.
4. 각 PBR 그룹 별 GPA의 평균값을 구해줍니다.
5. `ggplot()`을 이용해 시각화를 해줍니다.

그림에서 알 수 있듯이 PBR이 낮을수록 GPA도 낮으며, 즉 가치주일수록 우량성은 떨어집니다. 반면에 PBR이 높을수록 GPA도 높으며, 이는 주식의 가격이 비쌀수록 우량성도 높다는 것을 의미합니다.

이를 이용해 밸류 팩터와 퀄리티 팩터간의 관계를 나타내면 다음과 같습니다.

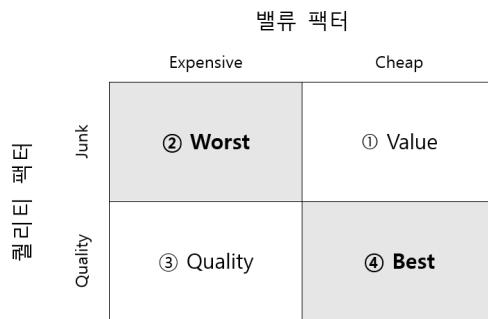


Figure 10.1: 밸류 팩터와 퀄리티 팩터간의 관계

주가가 쌀수록 기업의 우량성은 떨어지며(①번), 반대로 기업의 우량성이 좋으면 주식은 비싼 경향(③번)이 있습니다. 물론 우량성도 떨어지고 비싸기만한 주식(②번)을 사려는 사람들 아마 없을 겁니다. 결과적으로 우리가 원하는 최고의 주식은 우량성이 있으면서도 가격은싼 주식(④번)입니다.

## 10.2.2 마법공식 이해하기

마법공식이란 고담 캐피탈의 설립자이자 전설적인 투자자 조엘 그린블라트에 의해 알려진 투자방법입니다. 그는 본인의 책 **주식 시장을 이기는 작은 책**에서 투자를 하는데 있어 중요한 두가지 지표와, 이를 혼합할 경우 뛰어난 성과를 기록할 수 있다고 하였습니다.

첫번째 지표는 이율(Earnings Yield)로써 기업의 수익을 기업의 가치로 나눈 값입니다. 이는 PER의 역수와 비슷하며, 밸류 지표 중 하나입니다.

두번째 지표는 투자자본 수익률(Return on Capital)로써 기업의 수익을 투자한 자본으로 나눈 값입니다. 이는 ROE와도 비슷하며, 퀄리티 지표 중 하나입니다.

마법공식은 이 두가지 지표의 랭킹을 각각 구한 후, 랭킹의 합 기준 상위 30 개 종목을 1년간 보유한 후 매도하는 전략입니다.

해당 전략은 국내 투자자들에게도 많이 사랑받는 전략이지만 두 지표를 계산하기 위한 데이터를 수집하는데 어려움이 있어 많은 투자자들이 이율 대신 PER를, 투자자본 수익률 대신 ROE를 사용합니다. 그러나 우리가 수집한 데이터를 통해 충분히 원래의 마법공식을 구현할 수 있습니다.

Table 10.1: 마법공식의 구성 요소

팩터	Value	Quality
지표 계산	이율 (Earnings Yield) <small>이자 및 법인세 차감전이익 기업 가치</small>	투하자본 수익률 (Return On Capital) <small>이자 및 법인세 차감전이익 투하자본</small>

### 10.2.3 마법공식 구성하기

재무제표 항목을 통해 이율과 투하자본 수익률을 계산하고, 이를 통해 마법공식 포트폴리오를 구성하도록 하겠습니다.

먼저, 밸류지표에 해당하는 이익수익률을 계산해보도록 하겠습니다. 이익수익률은 이자 및 법인세 차감전이익(EBIT)을 기업가치(시가총액 + 순차입금)로 나눈 값입니다. 이를 분해하면 다음과 같습니다.

$$\begin{aligned}
 \text{이익수익률} &= \frac{\text{이자 및 법인세 차감전이익}}{\text{기업 가치}} \\
 &= \frac{\text{이자 및 법인세 차감전이익}}{\text{시가총액} + \text{순차입금}} \\
 &= \frac{\text{당기순이익} + \text{법인세} + \text{이자비용}}{\text{시가총액} + \text{총부채} - \text{여유자금}} \\
 &= \frac{\text{당기순이익} + \text{법인세} + \text{이자비용}}{\text{시가총액} + \text{총부채} - (\text{현금} - \max(0, \text{유동부채} - \text{유동자산} + \text{현금}))}
 \end{aligned}$$

```

library(stringr)
library(dplyr)

KOR_value = read.csv('data/KOR_value.csv', row.names = 1,
                      stringsAsFactors = FALSE)
KOR_fs = readRDS('data/KOR_fs.Rds')
KOR_ticker = read.csv('data/KOR_ticker.csv', row.names = 1,
                      stringsAsFactors = FALSE)
KOR_ticker$'종목코드' =

```

```

str_pad(KOR_ticker$'종목코드', 6, 'left', 0)

num_col = ncol(KOR_fs[[1]])

# 분자
magic_ebit = (KOR_fs$'지배주주순이익' + KOR_fs$'법인세비용' +
               KOR_fs$'이자비용')[num_col]

# 분모
magic_cap = KOR_value$PER * KOR_fs$'지배주주순이익'[num_col]
magic_debt = KOR_fs$'부채'[num_col]
magic_excess_cash_1 = KOR_fs$'유동부채' - KOR_fs$'유동자산' +
    KOR_fs$'현금및현금성자산'
magic_excess_cash_1[magic_excess_cash_1 < 0] = 0
magic_excess_cash_2 =
    (KOR_fs$'현금및현금성자산' - magic_excess_cash_1)[num_col]

magic_ev = magic_cap + magic_debt - magic_excess_cash_2

# 이익수익률
magic_ey = magic_ebit / magic_ev

```

먼저 가치지표, 재무제표, 티커 데이터를 불러온 후, 재무제표 열 갯수를 구해주세요. 그 후 분자와 분모 항목에 해당하는 부분을 하나씩 계산해 줍니다.

먼저 분자 부분인 **이자 및 법인세 차감전이익**은 **지배주주 순이익**에 **법인세비용**과 **이자비용**을 더해줍니다. 그 후, 최근년도 데이터를 선택해 줍니다.

분모 부분은 시가총액, 총부채, 여유자금 총 세가지로 구성되어 있습니다.

- 우리가 가지고 있는 벨류 데이터와 재무제표 데이터를 통해 시가총액을 역산할 수 있습니다. PER 값에 Earnings를 곱해주면 시가총액이 계산되게 됩니다. 이를 통해 계산된 시가총액을 HTS나 금융 사이트의 값과 비교하면 거의 비슷함이 확인됩니다.

$$\begin{aligned}
 PER \times Earnings &= \frac{Price}{Earnings/Shares} \times Earnings \\
 &= \frac{Price \times Shares}{Earnings} \times Earnings \\
 &= Price \times Earnings = Market Cap
 \end{aligned}$$

2. 총 부채는 부채 항목을 사용합니다.
3. 여유자금은 두 단계에 걸쳐 계산하도록 합니다. 먼저 **유동부채 - 유동자산 + 현금** 값을 구해준 후, 0보다 작은 값은 모두 0으로 바꾸줍니다. 이 값을 현금 및 현금성자산 항목에서 차감하여 최종적인 여유자금을 구하도록 합니다.

분자와 분모 부분을 나누어주면 이익수익률을 계산할 수 있습니다.

다음으로 퀄리티 지표에 해당하는 투하자본 수익률을 계산하도록 하겠습니다. 해당 값은 이자 및 법인세 차감전이익(EBIT)를 투하자본(IC)으로 나누어 계산되며, 이를 분해하면 다음과 같습니다.

$$\begin{aligned} \text{투하자본 수익률} &= \frac{\text{이자 및 법인세 차감전이익}}{\text{투하자본}} \\ &= \frac{\text{당기순이익} + \text{법인세} + \text{이자비용}}{(\text{유동자산} - \text{유동부채}) + (\text{비유동자산} - \text{감가상각비})} \end{aligned}$$

```
magic_ic = ((KOR_fs$'유동자산' - KOR_fs$'유동부채') +
            (KOR_fs$'비유동자산' - KOR_fs$'감가상각비')) [num_col]
magic_roc = magic_ebit / magic_ic
```

투하자본 수익률은 비교적 쉽게 계산할 수 있습니다. 분모에 해당하는 투자본의 경우 재무제표 항목을 그대로 사용하면 되며, 분자인 이자 및 법인세 차감전이익은 위에서 이미 구해둔 값을 사용하면 됩니다.

이제 두 지표를 활용하여 마법공식 포트폴리오를 구성하도록 하겠습니다.

```
invest_magic = rank(rank(-magic_ey) + rank(-magic_roc)) <= 30
KOR_ticker[invest_magic, ] %>%
  select(`종목코드`, `종목명`) %>%
  mutate(`이익수익률` = round(magic_ey[invest_magic, ], 4),
        `투하자본수익률` = round(magic_roc[invest_magic, ], 4))
```

	종목코드	종목명	이익수익률	투하자본수익률
## 1	005930	삼성전자	0.1839	0.2504
## 2	000660	SK하이닉스	0.3336	0.4793
## 3	004800	효성	0.5661	1.8508
## 4	010780	아이에스동서	0.1864	0.2914

## 5	012630	HDC	0.5149	0.3623
## 6	008060	대덕전자	0.2894	0.2936
## 7	001820	삼화콘덴서	0.1408	0.6383
## 8	095610	테스	0.1640	0.2526
## 9	003300	한일홀딩스	0.3921	0.2164
## 10	086390	유니테스트	0.3139	0.4580
## 11	003030	세아제강지주	0.2992	0.2303
## 12	121800	비덴트	0.3605	0.3265
## 13	045100	한양이엔지	0.2794	0.3235
## 14	004960	한신공영	0.1830	0.3066
## 15	036190	금화피에스시	0.2369	0.2316
## 16	029460	케이씨	0.9347	0.5832
## 17	040910	아이씨디	0.1751	0.2611
## 18	036200	유니셈	0.1594	0.2694
## 19	126700	하이비전시스템	0.2019	0.2266
## 20	035620	바른손이앤에이	0.7228	0.9668
## 21	006580	대양제지	0.1945	0.2779
## 22	083930	아바코	0.1673	0.2599
## 23	290740	액트로	0.1969	0.3194
## 24	001570	금양	0.1454	0.3548
## 25	042040	케이피엠테크	0.3173	0.2939
## 26	036010	아비코전자	0.5252	0.2416
## 27	127710	아시아경제	0.3459	0.2409
## 28	010280	쌍용정보통신	0.3154	0.3963
## 29	094970	제이엠티	0.2379	0.2700
## 30	194510	파티게임즈	0.2171	0.5069

이익수익률과 투하 자본 수익률의 랭킹을 각각 구해주며, 내림차순으로 값을 구하기 위해 마이너스를 붙여줍니다. 그 후 두 값의 합의 랭킹 기준 상위 30 종목을 선택한 후 종목코드와 종목명, 각 지표를 확인해주세요 합니다.

## 10.3 이상치 데이터 제거 및 팩터의 결합

모든 데이터 분석에서 중요한 문제 중 하나가 이상치(극단치, Outlier) 데이터를 어떻게 처리할 것인가입니다. 과거 12개월 수익률이 10배인 주식이 과연 모멘텀 관점에서 좋기만 한 주식인가, ROE가 100% 넘는 주식이 과연 퀄리티 관점에서 좋기만 한 주식인가 고민이 되기 마련입니다.

따라서 이러한 이상치를 제외하고 분석 할지, 포함해서 분석 할지를 판단해야 하며, 만일 포함한다면 그대로 사용할 것인지 보정하여 사용할 것인지도 판단해야

합니다.

우리가 가지고 있는 데이터에서 이러한 이상치 데이터를 탐색해보도록 하겠습니다.

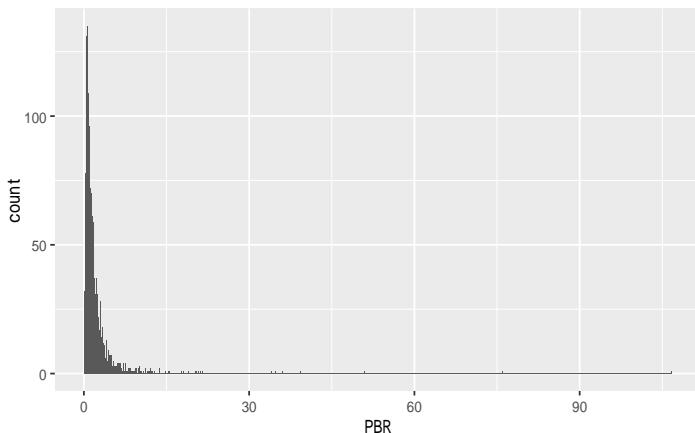
```
library(magrittr)
library(ggplot2)

KOR_value = read.csv('data/KOR_value.csv', row.names = 1,
                      stringsAsFactors = FALSE)

max(KOR_value$PBR, na.rm = TRUE)

## [1] 106.7

KOR_value %>%
  ggplot(aes(x = PBR)) +
  geom_histogram(binwidth = 0.1)
```



국내 종목들의 PBR 히스토그램을 그려보면 오른쪽으로 꼬리가 매우 긴 분포를 보이고 있습니다. 이는 PBR이 무려 106.71인 이상치 데이터가 존재하기 때문입니다. 이처럼 모든 팩터 지표에는 극단치 데이터가 존재하기 마련이며, 이를 처리하는 방법에 대해 알아보도록 하겠습니다.

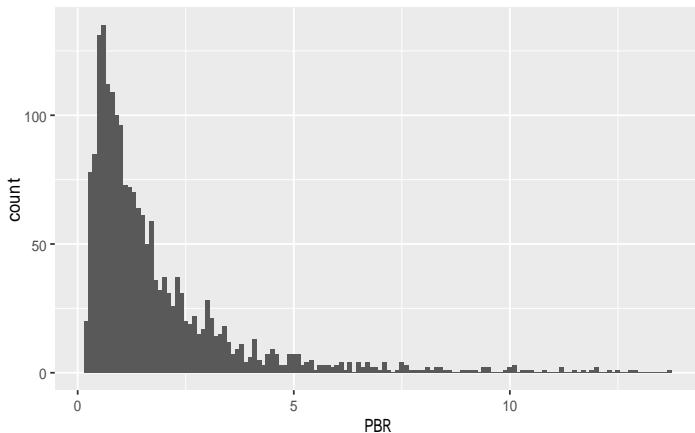
### 10.3.1 트림(Trim): 이상치 데이터 삭제

트림은 이상치 데이터를 삭제하는 방법입니다. 위의 예제에서 이상치에 해당하는 상하위 1% 데이터를 삭제해주도록 하겠습니다.

```
library(dplyr)

value_trim = KOR_value %>%
  select(PBR) %>%
  mutate(PBR = ifelse(percent_rank(PBR) > 0.99, NA, PBR),
        PBR = ifelse(percent_rank(PBR) < 0.01, NA, PBR))

value_trim %>%
  ggplot(aes(x = PBR)) +
  geom_histogram(binwidth = 0.1)
```



`percent_rank()` 함수를 통해 백분위를 구한 후 상하위 1%에 해당하는 데이터들은 NA로 변경하였습니다. 결과적으로 지나치게 PBR이 낮은 종목과 높은 종목은 제거가 되어 x축의 스케일이 많이 줄어든 모습입니다.

평균이나 분산과 같이 통계값을 구하는 과정에서는 이상치 데이터를 제거하는 것은 바람직 할 수 있습니다. 그러나 팩터를 이용해 포트폴리오를 구하는 과정에서 해당 방법은 잘 사용되지 않습니다. 데이터의 손실이 발생하게 되며, 제거된 종목 중 정말로 좋은 종목이 존재할 수도 있기 때문입니다.

### 10.3.2 원저라이징 (Winsorizing): 이상치 데이터 대체

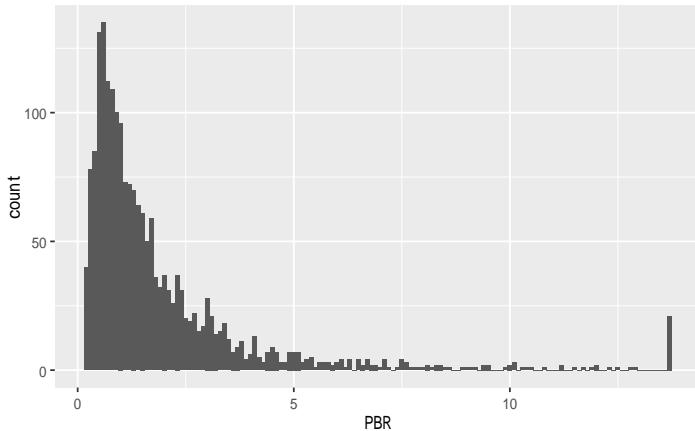
포트폴리오 구성에서는 일반적으로 이상치 데이터를 다른 데이터로 대체하는 원저라이징 방법이 사용됩니다. 예를 들어, 상위 99%를 초과하는 데이터는 99% 값으로 대체하며, 하위 1% 미만의 데이터는 1% 데이터로 대체합니다. 즉, 좌우로 울타리를 쳐놓고 해당 범위를 넘어가는 값을 강제로 울타리에 맞춰줍니다.

```

value_winsor = KOR_value %>%
  select(PBR) %>%
  mutate(PBR = ifelse(percent_rank(PBR) > 0.99,
                      quantile(., 0.99, na.rm = TRUE), PBR),
         PBR = ifelse(percent_rank(PBR) < 0.01,
                      quantile(., 0.01, na.rm = TRUE), PBR))

value_winsor %>%
  ggplot(aes(x = PBR)) +
  geom_histogram(binwidth = 0.1)

```



역시나 `percent_rank()` 함수를 통해 백분위를 구한 후, 해당 범위를 초과할 경우 각각 상하위 1% 데이터로 변형을 해줍니다. 그림을 살펴보면 x축 양 끝 부분의 막대가 길어진 것을 확인할 수 있습니다.

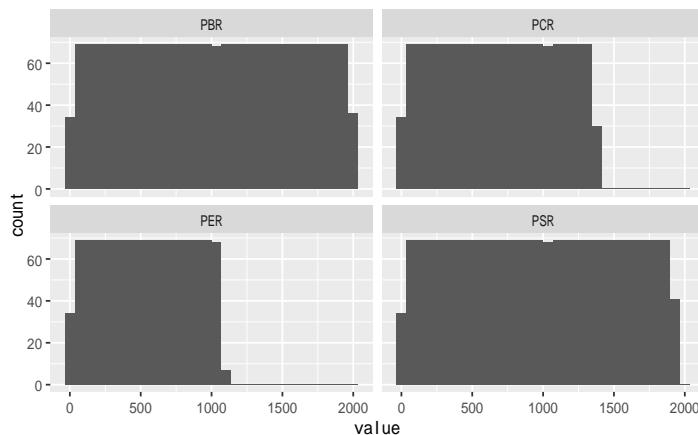
### 10.3.3 팩터의 결합 방법

밸류 지표의 결합, 켤리티 지표의 결합, 그리고 마법공식 포트폴리오를 구성할 때 단순히 랭킹을 더하는 방법을 사용하였습니다. 물론 투자 종목수가 얼마 되지 않거나, 개인투자자의 입장에서는 이러한 방법이 가장 심플하면서도 효과적일수 있습니다.

그러나 전문투자자 혹은 팩터를 분석하는 업무를 할 경우, 이처럼 단순히 랭킹을 더하는 방법은 여러가지 문제를 지니고 있습니다.

```
library(tidyverse)

KOR_value %>%
  mutate_all(list(~min_rank(.))) %>%
  gather() %>%
  ggplot(aes(x = value)) +
  geom_histogram() +
  facet_wrap(. ~ key)
```



위 그림은 각 밸류 지표의 랭킹을 구한 후 히스토그램으로 나타낸 것입니다. 랭킹을 구하는 것의 가장 큰 장점은 극단치로 인한 효과가 사라진다는 점과, 균등한 분포를 가진다는 점입니다.

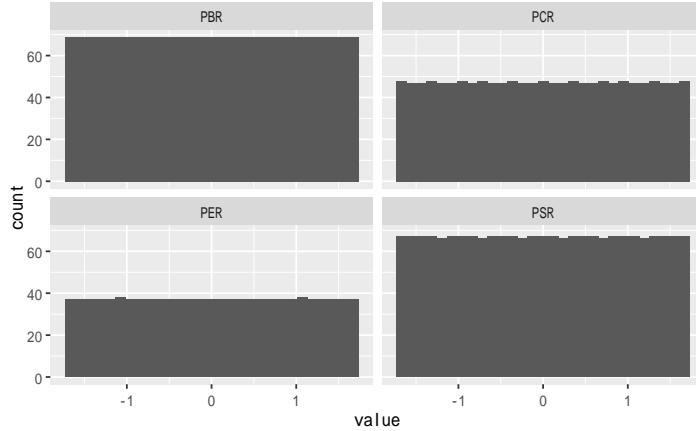
그러나 각 지표의 x축을 보면 최대값이 서로 다릅니다. 이는 지표 별 결측치로 인해 유효 데이터의 갯수가 달라 나타나는 현상이며, 서로 다른 범위의 분포를 단순히 합치는 것은 좋은 방법이 아닙니다.

예를 들어 A, B, C, D 팩터에 각각 비중을 40%, 30%, 20%, 10% 부여하여 포트폴리오를 구성하고자 할 경우를 생각해 봅시다. 각 랭킹은 분포의 범위가 다르므로, 랭킹과 비중의 가중평균을 통해 포트폴리오를 구성하는 방법은 왜곡된 결과를 발생시킵니다.

이러한 문제를 해결하는 가장 좋은 방법은 랭킹을 구한 후, 이를 Z-Score로 정규화하는 방법입니다.

```
KOR_value %>%
  mutate_all(list(~min_rank(.))) %>%
  mutate_all(list(~scale(.))) %>%
  gather()
```

```
ggplot(aes(x = value)) +
  geom_histogram() +
  facet_wrap(. ~ key)
```



`min_rank()`를 통해 랭킹을 구한 후, `scale()`을 통해 정규화를 해주었습니다. 기본적으로 랭킹의 분포가 가진 극단치 효과가 사라지는 점과 균등 분포의 장점을 유지하고 있으며, 분포의 범위 역시 거의 동일하게 바뀌었습니다.

이처럼 여러 팩터를 결합하여 포트폴리오를 구성하고자 하는 경우, 먼저 각 팩터(지표) 별 랭킹을 정규화를 한 뒤 더하는 것이, 왜곡 효과가 제거된 안정적인 방법입니다.

$$Z = Score(Rank(Factor A)) + Z = Score(Rank(Factor B)) + \dots + Z = Score(Rank(Factor N))$$

## 10.4 멀티팩터 포트폴리오

앞에서 배웠던 팩터 이론들과 결합 방법들을 응용하여 멀티팩터 포트폴리오를 구성해보도록 하겠습니다. 각 팩터에 사용되는 지표는 다음과 같습니다.

- 퀄리티: 자기자본이익률, 매출총이익, 영업활동현금흐름
- 밸류: PER, PBR, PSR, PCR
- 모멘텀: 3개월 수익률, 6개월 수익률, 12개월 수익률

```

library(xts)
library(stringr)

KOR_fs = readRDS('data/KOR_fs.Rds')
KOR_value = read.csv('data/KOR_value.csv', row.names = 1,
                      stringsAsFactors = FALSE)
KOR_price = read.csv('data/KOR_price.csv', row.names = 1,
                      stringsAsFactors = FALSE) %>% as.xts()

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

```

먼저 재무제표, 가치지표, 주가 데이터를 불러오도록 합니다.

```

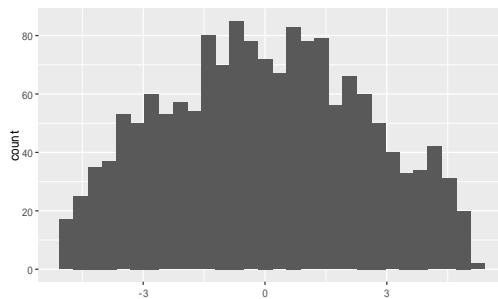
num_col = ncol(KOR_fs[[1]])
quality_roe = (KOR_fs$'지배주주순이익' / KOR_fs$'자본')[num_col]
quality_gpa = (KOR_fs$'매출총이익' / KOR_fs$'자산')[num_col]
quality_cfo =
  (KOR_fs$'영업활동으로인한현금흐름' / KOR_fs$'자산')[num_col]

quality_profit =
  cbind(quality_roe, quality_gpa, quality_cfo) %>%
  setNames(., c('ROE', 'GPA', 'CFO'))

factor_quality = quality_profit %>%
  mutate_all(list(~min_rank(desc(.)))) %>%
  mutate_all(list(~scale(.))) %>%
  rowSums()

factor_quality %>%
  data.frame() %>%
  ggplot(aes(x = `.`)) +
  geom_histogram()

```

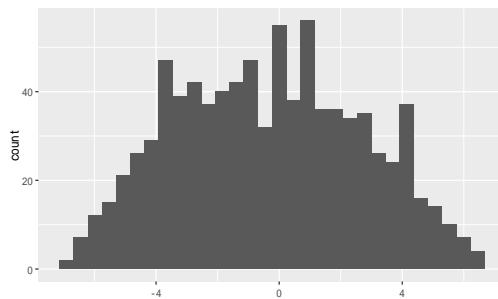


첫번째로 퀄리티 지표를 계산해줍니다. 코드는 앞에서 살펴본 것과 거의 비슷하며, 자기자본이익률, 매출총이익, 영업활동현금흐름을 계산해줍니다. 그 후 `mutate_all()` 함수를 통해 랭킹을 구한 후 다시 표준화를 해주도록 하며, 내림차순으로 정리하기 위해 랭킹 부분에 `desc()`를 붙여줍니다.

`rowSums()` 함수를 통해 계산된 Z-Score를 종목별로 합쳐주도록 합니다. Z-Score의 히스토그램을 살펴보면 이상치가 없이 중앙에 데이터가 많이 분포되어 있습니다.

```
factor_value = KOR_value %>%
  mutate_all(list(~min_rank(.))) %>%
  mutate_all(list(~scale(.))) %>%
  rowSums()

factor_value %>%
  data.frame() %>%
  ggplot(aes(x = `.`)) +
  geom_histogram()
```



두번째로 밸류 지표를 계산해줍니다. 밸류 지표의 경우 이미 테이블 형태로 들어와 있으며, 랭킹과 표준화를 거쳐 합을 구해주도록 합니다. 역시나 이상치가 없이 중앙에 데이터가 많이 분포되어 있습니다.

```

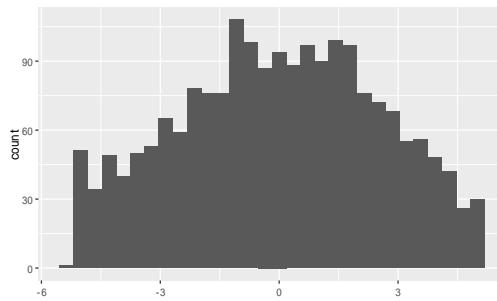
library(PerformanceAnalytics)
library(dplyr)

ret_3m = Return.calculate(KOR_price) %>% xts::last(60) %>%
  sapply(., function(x) {prod(1+x) - 1})
ret_6m = Return.calculate(KOR_price) %>% xts::last(120) %>%
  sapply(., function(x) {prod(1+x) - 1})
ret_12m = Return.calculate(KOR_price) %>% xts::last(252) %>%
  sapply(., function(x) {prod(1+x) - 1})
ret_bind = cbind(ret_3m, ret_6m, ret_12m) %>% data.frame()

factor_mom = ret_bind %>%
  mutate_all(list(~min_rank(desc(.)))) %>%
  mutate_all(list(~scale(.))) %>%
  rowSums()

factor_mom %>%
  data.frame() %>%
  ggplot(aes(x = ` `)) +
  geom_histogram()

```



마지막으로 모멘텀 지표를 계산해줍니다. 최근 60일, 120일, 252일 주가를 통해 3개월, 6개월, 12개월 수익률을 구해준 후 `cbind()`를 통해 열로 묶어주도록 합니다. 그 후 내림차순 기준 랭킹과 표준화를 거쳐 합을 구해주도록 합니다.

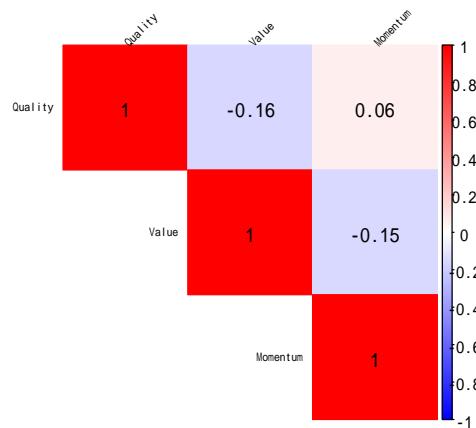
```

library(corrplot)

cbind(factor_quality, factor_value, factor_mom) %>%
  data.frame() %>%
  setNames(c('Quality', 'Value', 'Momentum')) %>%

```

```
cor(use = 'complete.obs') %>%
round(., 2) %>%
corrplot(method = 'color', type = 'upper',
         addCoef.col = 'black', number.cex = 1,
         tl.cex = 0.6, tl.srt = 45, tl.col = 'black',
         col =
           colorRampPalette(c('blue', 'white', 'red'))(200),
         mar=c(0,0,0.5,0))
```



퀄리티, 밸류, 모멘텀 팩터 간의 랭크의 서로 간 상관관계가 매우 낮으며, 여러 팩터를 동시에 고려함으로써 분산효과를 기대할 수 있습니다.

```
factor_qvm =
  cbind(factor_quality, factor_value, factor_mom) %>%
  data.frame() %>%
  mutate_all(list(~scale(.))) %>%
  mutate(factor_quality = factor_quality * 0.33,
        factor_value = factor_value * 0.33,
        factor_mom = factor_mom * 0.33) %>%
  rowSums()

invest_qvm = rank(factor_qvm) <= 30
```

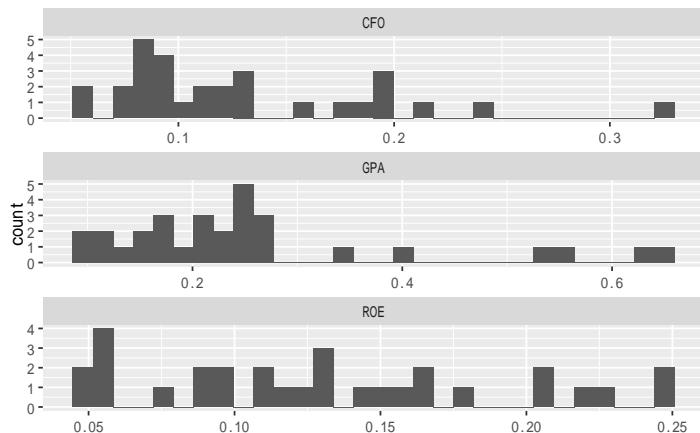
계산된 팩터들을 토대로 최종 포트폴리오를 구성해주도록 합니다. 각 팩터의 분포가 역시나 다르기 때문에 다시 한번 `scale()`을 통해 정규화를 해주도록 하며, 각 팩터에 동일한 비중인 0.33을 곱해준 후 이를 더해주도록 합니다.

물론 팩터 별 비중을 [0.2, 0.4, 0.4]와 같이 다르게 줄 수도 있으며, 이는 어떠한 팩터를 더욱 중요하게 생각하는지 혹은 더욱 좋게 보는지에 따라 조정이 가능합니다.

최종적으로 해당 값의 랭킹 기준 상위 30 종목을 선택해 줍니다.

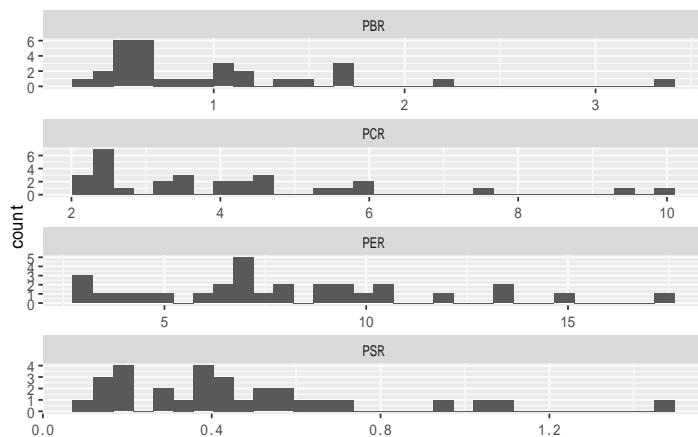
```
library(tidyr)

quality_profit[invest_qvm, ] %>%
  gather() %>%
  ggplot(aes(x = value)) +
  geom_histogram() +
  facet_wrap(. ~ key, scale = 'free', ncol = 1) +
  xlab(NULL)
```



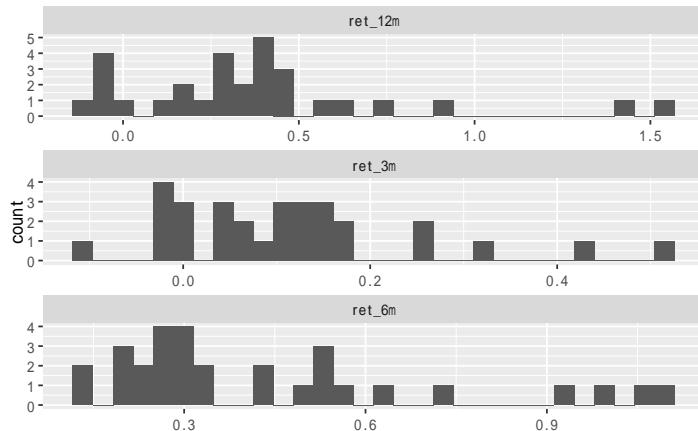
먼저 선택된 종목의 퀄리티 지표별 분포를 살펴보도록 하겠습니다. 대부분 종목의 수익성이 높음이 확인됩니다.

```
KOR_value[invest_qvm, ] %>%
  gather() %>%
  ggplot(aes(x = value)) +
  geom_histogram() +
  facet_wrap(. ~ key, scale = 'free', ncol = 1) +
  xlab(NULL)
```



이번에는 선택된 종목의 가치 지표별 분포입니다. 대부분 종목의 값이 낮아, 밸류 종목임이 확인됩니다.

```
ret_bind[invest_qvm, ] %>%
  gather() %>%
  ggplot(aes(x = value)) +
  geom_histogram() +
  facet_wrap(~ key, scale = 'free', ncol = 1) +
  xlab(NULL)
```



마지막으로 각 종목들의 기간 별 수익률 분포입니다. 역시나 대부분의 종목들이 높은 수익률을 보입니다.

```
KOR_ticker[invest_qvm, ] %>%
  select('종목코드', '종목명') %>%
  cbind(round(quality_roe[invest_qvm, ], 2)) %>%
  cbind(round(KOR_value$PBR[invest_qvm], 2)) %>%
  cbind(round(ret_12m[invest_qvm], 2)) %>%
  setNames(c('종목코드', '종목명', 'ROE', 'PBR', '12M'))
```

	종목코드	종목명	ROE	PBR	12M
## 65	000210	대림산업	0.11	0.65	0.41
## 167	010780	아이에스동서	0.22	1.05	0.34
## 220	034310	NICE	0.06	0.79	0.26
## 350	097520	엠씨넥스	0.23	3.39	1.44
## 385	047310	파워로직스	0.13	2.24	1.52
## 484	016590	신대양제지	0.16	0.63	0.17
## 494	102710	이엔에프테크놀로지	0.13	1.35	0.48
## 507	067990	도이치모터스	0.17	1.49	0.90
## 532	049070	인탑스	0.05	0.58	0.74
## 596	013580	계룡건설	0.20	0.55	0.12
## 605	036800	나이스정보통신	0.15	1.19	-0.06
## 629	005960	동부건설	0.20	0.65	0.15
## 706	002170	삼양통상	0.09	0.66	0.55
## 748	005990	매일홀딩스	0.05	0.35	-0.13
## 782	023600	삼보판지	0.16	0.46	0.24
## 797	036810	에프에스티	0.18	1.66	0.38
## 822	065130	탭엔지니어링	0.06	0.56	0.63
## 848	101330	모베이스	0.09	0.67	0.37
## 874	038390	레드캡투어	0.09	1.00	-0.03
## 923	069510	에스텍	0.14	1.00	0.27
## 980	039340	한국경제TV	0.13	1.15	0.41
## 1022	101160	월덱스	0.25	1.63	0.30
## 1038	007540	샘표	0.06	0.52	0.39
## 1080	006580	대양제지	0.25	0.91	0.44
## 1239	016740	두올	0.09	0.59	-0.07
## 1484	004140	동방	0.12	0.52	-0.05
## 1541	019180	티에이치엔	0.12	1.69	0.34
## 1551	037330	인지디스플레	0.06	0.44	0.31
## 1642	091340	S&K플리텍	0.11	0.81	0.44
## 1697	081580	성우전자	0.08	0.50	-0.03

포트폴리오 내 종목들을 대상으로 팩터 별 대표적인 지표인 ROE, PBR, 12개월 수익률을 나타내었습니다. 전반적으로 ROE는 높고 PBR은 낮으며, 12개월

수익률이 높은 모습을 보입니다. 물론 특정 팩터의 강도가 약하더라도 나머지 팩터의 강도가 충분히 강하다면, 포트폴리오에 편입되는 모습을 보이기도 합니다.

```
cbind(quality_profit, KOR_value, ret_bind)[invest_qvm, ] %>%
  apply(., 2, mean) %>% round(3) %>% t()
```

```
##          ROE      GPA     CFO      PER      PBR      PCR      PSR ret_3m
## [1,] 0.131 0.261 0.13 8.072 0.989 4.102 0.478 0.117
##          ret_6m ret_12m
## [1,] 0.442 0.374
```

마지막으로 포트폴리오 내 종목들의 지표 별 평균을 계산한 값입니다.

# CHAPTER 11

---

## 포트폴리오 구성

---

종목별로 비중을 어떻게 배분하느냐에 따라 성과가 달라지므로, 종목의 선택 못지 않게 중요한 것이 포트폴리오를 구성하는 방법입니다. 최적 포트폴리오의 구성은 수식을 기반으로 최적화된 해를 찾습니다. 물론 엑셀의 해찾기와 같은 기능을 사용하여 간단한 형태의 최적화 구현이 가능하지만, 데이터가 방대해질 경우에는 속도가 지나치게 느려지거나 계산을 할 수 없기도 합니다.

동일한 최적화 방법을 지속적으로 사용할 경우 프로그래밍을 통해 함수를 만들고, 입력 변수만 변경하는 것이 훨씬 효율적인 방법입니다. 또한 포트폴리오 최적화에 관한 좋은 패키지들이 이미 많이 나와 있으므로, 대략적인 내용만 이해하고 실제 구현은 패키지를 이용하는 것도 좋은 방법입니다.

본 장에서는 일반적으로 많이 사용되는 최소분산 포트폴리오, 최대분산효과 포트폴리오, 위험균형 포트폴리오를 구현해보도록 합니다. 먼저 포트폴리오 구성을 위해 글로벌 자산을 대표하는 ETF 데이터를 다운로드 받도록 하겠습니다.

```
library(quantmod)
library(PerformanceAnalytics)
library(magrittr)

symbols = c('SPY', # 미국 주식
           'IEV', # 유럽 주식
           'EWJ', # 일본 주식
           'EEM', # 이머징 주식
```

```

    'TLT', # 미국 장기채
    'IEF', # 미국 중기채
    'IYR', # 미국 리츠
    'RWX', # 글로벌 리츠
    'GLD', # 금
    'DBC' # 상품
)
getSymbols(symbols, src = 'yahoo')

```

```

## [1] "SPY" "IEV" "EWJ" "EEM" "TLT" "IEF" "IYR" "RWX"
## [9] "GLD" "DBC"

```

```

prices = do.call(cbind,
                  lapply(symbols, function(x) Ad(get(x)))) %>%
  setNames(symbols)

rets = Return.calculate(prices) %>% na.omit()

```

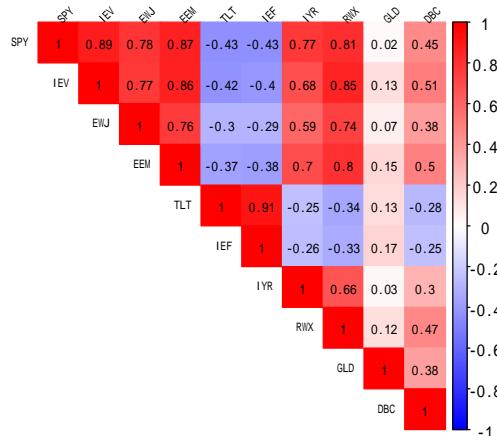
`getSymbols()` 함수를 통해 일반적으로 자산배분에서 많이 사용되는 주식과 채권, 대체자산에 해당하는 ETF 가격 데이터를 받은 후, `lapply()`와 `Ad()`, `get()` 함수의 조합을 통해 수정주가 만을 선택하여 열의 형태로 둉어주도록 합니다. 그 후 `Return.calculate()` 함수를 통해 수익률을 계산 해줍니다.

```

library(tidyr)
library(dplyr)
library(corrplot)

cor(rets) %>%
  corrplot(method = 'color', type = 'upper',
           addCoef.col = 'black', number.cex = 0.7,
           tl.cex = 0.6, tl.srt=45, tl.col = 'black',
           col =
             colorRampPalette(c('blue', 'white', 'red'))(200),
           mar = c(0,0,0.5,0))

```



각 ETF의 수익률 간 상관관계를 살펴보면 같은 자산군 내에서는 강한 상관관계를 보이며, 주식과 채권 간에는 매우 낮은 상관관계를 보입니다. 또한 주식과 리츠 간에도 꽤 높은 상관관계를 보입니다.

포트폴리오 최적화에는 **분산-공분산 행렬**이 대부분 사용되며, 이는 `cov()` 함수를 통해 손쉽게 계산할 수 있습니다.

```
covmat = cov(rets)
```

## 11.1 최소분산 포트폴리오

최소분산 포트폴리오 (Minimum Variance Portfolio)는 변동성이 최소인 포트폴리오입니다. 포트폴리오의 변동성은 일반적으로  $\sum_{i=1}^n \sum_{j=1}^n w_i w_j \sigma_{ij}$ 의 형태로 표현되지만, 최적화 작업을 위해서는 행렬의 형태인  $w' \Omega w$ 로 표현하는 것이 더욱 편리합니다. 이 중  $w$ 는 각 자산들의 비중을 행렬의 형태로 나타낸 것이며,  $\Omega$ 는 분산-공분산 행렬을 나타낸 것입니다. 분산-공분산 행렬은 사전에 고정되어 있는 값이므로, 각 자산들의 비중인  $w$ 를 변화시킴으로써 포트폴리오의 변동성이 최소인 지점을 찾을 수 있습니다.

최소분산 포트폴리오의 목적함수는 아래의 수식으로 표현할 수 있습니다. 이 중  $1/2$ 은 단지 미분했을 때 계산을 용이하게 하기 위한 장치일 뿐 결과에는 영향을 미치지 않습니다.

$$\text{최소분산 포트폴리오의 목적함수 : } \min \frac{1}{2} w' \Omega w$$

다만 단순히 위의 목적함수를 찾는 해를 구할 경우 결과 값이 음수가 나오기도 하며, 이는 공매도를 의미합니다. 일반적으로 공매도가 불가능하다는 점과, 투자

비중의 합이 100%가 되어야 한다는 점을 고려하면 아래와 같은 제약조건을 추가해야 합니다.

$$\text{최소분산 포트폴리오의 제약조건} : \sum_{i=1}^n w_i = 1, w_i \geq 0$$

물론 이 외에도 각 섹터의 투자비중 합에 대한 제약, 회전율에 대한 제약 등도 추가할 수 있습니다.

### 11.1.1 slsqp() 함수를 이용한 최적화

R에서 가장 손쉽게 최적화 작업을 수행하는 방법은 nloptr 패키지의 slsqp() 함수를 이용하는 법입니다. 해당 함수는 순차적 이차 계획(Sequential quadratic programming)을 이용하여 해를 찾으며, 목적함수와 제약조건은 아래와 같습니다.

Table 11.1: ‘slsqp()’ 함수 목적함수와 제약조건

목적함수	제약조건
$\min f(x)$	$b(x) \geq 0, c(x) = 0$

목적함수에서  $f(x)$  는 최소화 하고자 하는 값, 즉 포트폴리오의 변동성입니다. 제약조건은 크게 개별 자산의 투자 비중이 0 이상인 것과, 투자 비중의 합이 1이 되도록 하는 것입니다. 첫번째 제약조건은 자연스럽게 개별 자산의 투자 비중이 0 이상인 것을 의미합니다. 두 번째 제약조건은 약간의 변형을 통해 투자 비중의 합이 1이 되는 제약조건을 만들 수 있습니다.  $c(x)$ 를 투자 비중의 합 – 1로 변형할 경우, -1을 우변으로 넘기면 결국 투자 비중의 합 = 1의 형태로 나타낼 수 있습니다. slsqp() 함수의 구성을 다음과 같습니다.

```
slsqp(x0, fn, gr = NULL, lower = NULL, upper = NULL,
      hin = NULL, hinjac = NULL, heq = NULL, heqjac = NULL,
      nl.info = FALSE, control = list(), ...)
```

이 중 우리가 구체적으로 입력해야 할 값은 x0, fn, hin, heq 항목입니다.

1. x0는 초기값으로써, 일반적으로 모든 x에 대해 동일한 값을 입력합니다.
2. fn은 최소화하고자 하는 목적함수로, 포트폴리오 변동성에 해당합니다.
3. hin은 부등위 제약조건(inequality constraints)을 의미하며, 프로그래밍 내에서는  $hin \geq 0$ 로 인식하며, 각 자산의 비중이 0보다 크다는 제약조건과 연결됩니다.

4. heq는 등위 제약조건 (equality constraints)을 의미하며 프로그래밍 내에 서는  $heq == 0$ 을 의미합니다. 투자 비중의 합 - 1의 형태를 입력 할 경우, 투자 비중의 합이 1이 라는 제약조건과 연결됩니다.

표 11.2는 최소분산 포트폴리오를 구할 때 필요한 주요 변수에 대한 내용입니다.

Table 11.2: slsqp() 함수의 인자와 포트폴리오 내 변수

변수명	내용	포트폴리오 내 변수
x0	초기값	없음
fn	목적함수	포트폴리오 변동성
hin	부등위 제약조건	각 자산의 비중이 0 보다 큰 제약조건
heq	등위 제약조건	투자 비중의 합이 1인 제약조건

slsqp() 함수를 이용하여 최소분산 포트폴리오를 만족하는 자산의 투자 비중을 구하는 과정은 다음과 같습니다. 먼저 fn, hin, heq에 해당하는 함수들을 각각 만들어 준 후, 이를 slsqp() 함수와 결합하여 최적화된 결과값을 얻을 수 있습니다. 구체적인 과정은 아래와 같습니다.

```
objective = function(w) {
  obj = t(w) %*% covmat %*% w
  return(obj)
}
```

먼저 목적함수에 해당하는 부분입니다. covmat은 사전에 계산된 분산-공분산 행렬이며,  $w$ 는 각 자산의 투자 비중입니다. obj는 포트폴리오의 변동성인  $w' \Omega w$ 를 계산한 것입니다. 즉, 해당 함수는 계산된  $w$ 를 바탕으로 포트폴리오의 변동성을 반환하고, 우리의 목적은 해당 값이 최소가 되도록 하는 것입니다.

```
hin.objective = function(w) {
  return(w)
}
```

$w_i \geq 0$  제약조건에 해당하는 부등위 제약조건입니다. 패키지 내에서는  $hin >= 0$ 의 형태로 인식을 하므로, 계산된 비중인  $w$ 를 단순히 입력하기만 하면 됩니다.

```
heq.objective = function(w) {
  sum_w = sum(w)
  return( sum_w - 1 )
}
```

$\sum_{i=1}^n w_i = 1$  제약조건에 해당하는 등위 제약조건입니다. 먼저 계산된 비중인  $w$  들의 합계를 구한 후, 해당 값에서 1을 빼주는 값을 반환하도록 합니다. 프로그래밍 내에서는 `heq == 0` 의 형태로 인식을 하므로, 결국 (`sum_w - 1`) == 0, 즉 `sum_w == 1` 의 제약조건과 동일하게 됩니다.

```
library(nloptr)

result = slsqp( x0 = rep(0.1, 10),
                 fn = objective,
                 hin = hin.objective,
                 heq = heq.objective)

print(result$par)

## [1] 1.528e-01 2.777e-17 -2.349e-17 -5.037e-18
## [5] -1.683e-16 7.843e-01 1.041e-17 -1.480e-17
## [9] -3.463e-18 6.287e-02

print(result$value)

## [1] 0.000009694
```

위에서 만들어진 함수들을 바탕으로 최적화 작업을 실행합니다. 초기값인 `x0`에는 먼저 동일한 비중들을 입력합니다. 예제에서는 종목이 10개 이므로, `x0` 값에는 `rep(0.1, 10)` 인 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1 를 입력합니다. 최소화 하고자 하는 목적함수 `fn`에는 위에서 구성한 `objective` 함수를 입력합니다. 부등위 제약조건과 등위 제약조건에도 각각 위에서 구성한 `hin.objective` 함수와 `heq.objective`를 입력합니다.

즉, 해당 함수는 초기값을 시작점으로 하여 주어진 제약조건을 만족하는 해를 찾기 위해  $w$  값들을 조정하는 작업을 반복한 후, 목적함수가 최소가 되는 지점의  $w$ 를 반환합니다.

`result` 값 중 `$par`는 최적화된 지점의 해를 의미하며, 최소분산 포트폴리오를 구성하는 자산들의 투자 비중을 의미합니다. `$value`는 `$par`에서 산출된 값을 목적함수 `fn`에 입력하였을 때 나오는 결과값으로써, 포트폴리오의 분산을 의미합니다.

```
w_1 = result$par %>% round(., 4) %>%
  setNames(colnames(rets))

print(w_1)
```

```
##    SPY     IEV     EWJ     EEM     TLT     IEF     IYR     RWX
## 0.1528 0.0000 0.0000 0.0000 0.0000 0.7843 0.0000 0.0000
##    GLD     DBC
## 0.0000 0.0629
```

자산들의 투자비중은 `result$par` 를 통해 추출한 후, `round()` 함수를 이용하여 반올림을 합니다. 마지막으로 이름에 종목명을 입력해주도록 합니다. 계산된 비중으로 포트폴리오를 구성할 경우, 포트폴리오의 비중이 최소가 됩니다.

### 11.1.2 `solve.QP()` 함수를 이용한 최적화

다음으로는 `quadprog` 패키지 내의 `solve.QP()` 함수를 이용하여 포트폴리오 최적화를 하는 방법이 있습니다. 해당 함수는 쌍대기법(Dual Method)를 이용하여 제약조건 내에서 목적함수가 최소화 되는 해를 구합니다. 해당 함수의 목적함수와 제약조건은 표 11.3와 같습니다.

Table 11.3: `solve.QP` 함수 목적함수와 제약조건

목적함수	제약조건
$\min(-d^T b + \frac{1}{2} b^T D b)$	$A^T b \geq b_0$

최소분산 포트폴리오의 목적함수가  $\min \frac{1}{2} w' \Omega w$  로 표시된다는 점을 생각하면, 해당 함수는 매우 이해하기 쉽게 구성되어 있습니다.  $b$ 를 각 개별 자산의 투자비중인  $w$ ,  $D$ 를 분산-공분산 행렬인  $\Omega$ 라 생각하면, 목적함수 중  $\min \frac{1}{2} w' D w$ 는 최소분산 포트폴리오의 목적함수와 정확히 동일합니다.  $d$ 를 0으로 생각할 경우  $-d^T b$  또한 0이 되어 목적함수에 아무런 영향도 미치지 않습니다.

제약조건 역시  $A^T$  항목을 적절하게 수정하여 준다면, 개별 자산의 투자비중이 0 이상인 것과, 투자비중의 합이 1이 되도록 만들 수 있습니다. 이에 대해서는 뒤에서 구체적으로 다루도록 합니다. `solve.QP()` 함수의 사용법은 아래와 같습니다.

```
solve.QP(Dmat, dvec, Amat, bvec, meq = 0, factorized = FALSE)
```

1.  $Dmat$ 은 목적함수 중  $D$ 에 해당하는 행렬부분으로 써 분산-공분산 행렬과 일치합니다.
2.  $dvec$ 은 목적함수 중  $d$ 에 해당하는 벡터부분이며, 포트폴리오 최적화에서는 역할이 없습니다.

3. Amat은 제약조건 중  $A^T$ 에 해당하는 부분으로써, 제약조건 중 좌변에 위치하는 항목입니다. 주의할 점은 제약조건에서 보듯이 제약조건 행렬을 구한 후 이의 전치(Transpose) 행렬을 입력해야 합니다.
4. bvec은 제약조건 중  $b_0$ 에 해당하는 부분으로써, 제약조건 중 우변에 위치하는 항목입니다.
5. meq는 bvec의 몇번째 까지를 등위 제약조건으로 설정할지에 대한 부분입니다.

표 11.4는 위의 내용을 요약한 것이며, 각 변수를 입력한 후 함수를 실행할 경우 위의 목적함수와 제약조건을 만족하는 b값을 찾습니다.

Table 11.4: ‘solve.QP’ 함수의 인자와 포트폴리오 내 변수

변수명	내용	포트폴리오 내 변수
Dmat	목적함수 중 D	분산-공분산 행렬
dvec	목적함수 중 d	해당사항 없음
Amat	제약조건 (좌변)	$\sum_{i=1}^n w_i, w_i$
bvec	제약조건 (우변)	비중의 합이 1, 각 비중이 0보다 큼
meq	등위 제약조건 개수	1개 (비중의 합이 1)

`solve.QP()` 함수를 이용하여 최소분산 포트폴리오 비중을 구할 때는 Amat 항목을 제대로 입력하는 것이 가장 중요하며, 나머지 항목은 매우 손쉽게 입력이 가능합니다. 설명된 내용에 해당하는 행렬을 손으로 직접 써가며 계산해 보신다면 훨씬 이해하기가 쉬울 것입니다. 구체적인 과정은 아래와 같습니다.

```
Dmat = covmat
dvec = rep(0, 10)
Amat = t(rbind(rep(1, 10), diag(10), -diag(10)))
bvec = c(1, rep(0, 10), -rep(1, 10))
meq = 1
```

Dmat에는 분산-공분산 행렬을 입력하며, dvec은 최소분산 포트폴리오를 구하는데는 필요한 값이 아니므로 0벡터를 입력 합니다. 등위 제약조건과 부등위 제약조건 ( $A^T b \geq b_0$ )을 행렬의 형태로 표현하면 다음과 같습니다.

$$\begin{bmatrix} 1 & \dots & 1 \\ 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \\ -1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & -1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{10} \end{bmatrix} = \begin{bmatrix} w_1 + w_2 + \dots + w_{10} \\ w_1 \\ \vdots \\ w_{10} \\ -w_1 \\ \vdots \\ -w_{10} \end{bmatrix} \geq \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ -1 \\ \vdots \\ -1 \end{bmatrix}$$

이 중, 맨 좌측 행렬의 전치행렬이 제약조건의 좌변인  $A_{\text{mat}}$ 에 해당합니다.

$$A_{\text{mat}} = \begin{bmatrix} 1 & \dots & 1 \\ 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \\ -1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & -1 \end{bmatrix}^T$$

맨 우측 행렬이 제약조건의 우변인  $b_{\text{vec}}$ 에 해당합니다.

$$b_{\text{vec}} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ -1 \\ \vdots \\ -1 \end{bmatrix}$$

위의 제약조건은 크게 투자 비중의 합이 1인 제약조건, 최소 투자 비중이 0 이상인 제약조건, 최대 투자 비중이 1 이하인 제약조건, 총 3개 부분으로 나눌 수 있습니다.

$$(1) \sum_{i=1}^n w_i = 1 \Rightarrow [w_1 + w_2 + \dots + w_{10}] = [1]$$

$$(2) w_i \geq 0 \Rightarrow \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{10} \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$(3) -w_i \geq -1 \Rightarrow \begin{bmatrix} -w_1 \\ -w_2 \\ \vdots \\ -w_{10} \end{bmatrix} \geq \begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}$$

`solve.QP()` 함수의 제약조건은 항상 좌변이 큰 형태이므로, 최대 투자 비중에 대한 제약 조건의 경우 다음 행렬의 양변에 마이너스(-)를 곱하여 부등호를 맞춰주었습니다.

$$\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{10} \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

첫번째 제약조건은 부등호가 아닌 등호, 즉 투자 비중의 합이 1인 조건을 의미하므로 `meq = 1`을 통해 첫번째 제약조건은 등식 제약조건임을 선언할 수 있습니다.

제약조건의 좌변에 해당하는 `Amat`을 만드는 과정은 다음과 같습니다. 먼저 `rep(1, 10)`를 통해 최상단에 위치한 1로 이루어진 행렬을 만들어줍니다.

$$\begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}$$

하단의 1과 -1로 이루어진 대각행렬은 `diag()` 함수를 통해 쉽게 만들 수 있습니다.

$$diag(10) = \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix}$$

$$-diag(10) = \begin{bmatrix} -1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & -1 \end{bmatrix}$$

세 개의 행렬을 `rbind()` 함수를 통해 행으로 묶어주면 제약조건의 맨 좌측 행렬과 동일한 형태가 됩니다. 이를 `t()` 함수를 통해 전치행렬을 만들어 준 뒤 `Amat`에 입력합니다.

제약조건에 해당하는 `bvec`은 1, 0, -1로 이루어진 벡터를 통해 손쉽게 만들 수 있습니다.

```

library(quadprog)
result = solve.QP(Dmat, dvec, Amat, bvec, meq)

print(result$solution)

## [1] 1.528e-01 1.738e-18 2.061e-20 -4.724e-18
## [5] -2.858e-17 7.843e-01 -5.015e-18 -6.073e-19
## [9] 0.000e+00 6.287e-02

print(result$value)

## [1] 0.000004847

```

위에 입력된 내역들을 `solve.QP()` 함수에 넣어 최적화 값을 찾아줍니다. 결과 중 `$solution`은 최적화된 지점의 해, 즉 최소분산 포트폴리오를 구성하는 자산들의 투자 비중을 의미합니다. `$value`는 `$solution`에서 산출된 값을 목적함수에 입력하였을 때 나오는 결과값으로써, 포트폴리오의 분산을 의미합니다.

```

w_2 = result$solution %>% round(., 4) %>%
  setNames(colnames(rets))

print(w_2)

##      SPY      IEV      EWJ      EEM      TLT      IEF      IYR      RWX
## 0.1528 0.0000 0.0000 0.0000 0.0000 0.7843 0.0000 0.0000
##      GLD      DBC
## 0.0000 0.0629

```

자산들의 투자비중은 `result$solution`을 통해 추출한 후, `round()` 함수를 이용하여 반올림을 하여 줍니다. 마지막으로 이름에 종목명을 입력해주도록 합니다. 계산된 비중으로 포트폴리오를 구성할 경우, 포트폴리오의 비중이 최소화 됩니다.

### 11.1.3 `optimalPortfolio()` 함수를 이용한 최적화

`RiskPortfolios` 패키지의 `optimalPortfolio()` 함수를 이용하여 매우 간단하게 최적화 포트폴리오를 구현할 수도 있습니다. 해당 함수의 사용법은 아래와 같습니다.

```
optimalPortfolio(Sigma, mu = NULL, semiDev = NULL,
                  control = list())
```

Sigma는 분산-공분산 행렬입니다. mu와 semiDev는 각각 기대수익률과 세미 편차(semi deviation)로써, 입력하지 않아도 됩니다. control은 포트폴리오 종류 및 제약조건에 해당하는 부분이며, 자세한 내용은 표 11.5와 같습니다.

Table 11.5: optimalPortfolio() 포트폴리오 내 control 인자

종류	입력 값	내용
type	minvol	최소분산 포트폴리오
	invvol	역변동성 포트폴리오
	erc	위험 균형 포트폴리오
	maxdiv	최대 분산효과 포트폴리오
	riskeff	위험-효율적 포트폴리오
constraint	lo	최소 투자 비중이 0 보다 클 것
	user	최소(LB) 및 최대 투자 비중(UB) 설정

control 항목에서 원하는 포트폴리오 타입과 제약조건을 입력해주면, 매우 손쉽게 최적화 포트폴리오를 구현할 수 있습니다.

```
library(RiskPortfolios)

w_3 = optimalPortfolio(covmat,
                        control = list(type = 'minvol',
                                      constraint = 'lo')) %>%
  round(., 4) %>%
  setNames(colnames(rets))

print(w_3)
```

```
##      SPY      IEV      EWJ      EEM      TLT      IEF      IYR      RWX
## 0.1528 0.0000 0.0000 0.0000 0.0000 0.7843 0.0000 0.0000
##      GLD      DBC
## 0.0000 0.0629
```

optimalPortfolio() 함수 내부에 분산-공분산 행렬을 입력합니다. type 부분에 최소분산 포트폴리오에 해당하는 minvol을 입력하며, constraint에는 각 자산의 비중이 0보다 큰 제약조건인 lo(Long Only)를 입력합니다. 비중의 합이 1인 제약조건은 자동적으로 적용이 됩니다.

이처럼 패키지를 이용할 경우 훨씬 간단하게 원하는 값을 얻을 수 있습니다. github를 통해 해당 함수의 코드를 살펴보면 `solve.QP()`를 이용하여 작성한 방법과 거의 동일합니다. 따라서 위의 과정들을 대략적으로 이해한 후, 패키지를 사용하여 포트폴리오 최적화를 구현하는 것이 현명한 방법이 될 수도 있습니다.

### 11.1.4 결과값들의 비교

아래 표는 `slsqp()`, `solve.QP()`, `optimalPortfolio()`를 이용하여 구한 값들의 비교입니다.

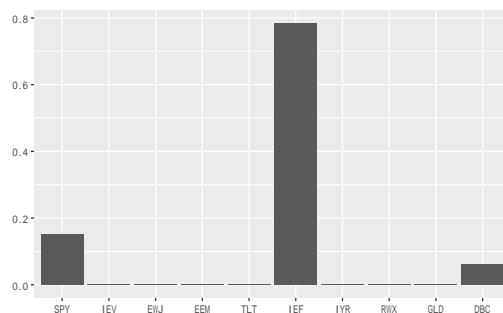
Table 11.6: 최적화 결과 비교

	SPY	IEV	EWJ	EEM	TLT	IEF	IYR	RWX	GLD	DBC
slsqp	0.1528	0	0	0	0	0.7843	0	0	0	0.0629
solve.QP	0.1528	0	0	0	0	0.7843	0	0	0	0.0629
optimalPortfolio	0.1528	0	0	0	0	0.7843	0	0	0	0.0629

3가지 방법 모두 결과가 동일합니다. 그러나 여기서 나온 결과를 이용하여 그대로 투자하기에는 문제가 있습니다. 일부 자산은 투자비중이 0%, 즉 전혀 투자를 하지 않는 반면, 특정 자산에 대부분의 비중인 78.43%를 투자를 하는 편중된 결과가 나옵니다.

```
library(ggplot2)

data.frame(w_1) %>%
  ggplot(aes(x = factor(rownames(.)), levels = rownames(.)),
         y = w_1)) +
  geom_col() +
  xlab(NULL) + ylab(NULL)
```



이처럼 변동성이 가장 낮은 종목에 대부분의 비중이 투자되는 구석해(Corner Solution) 문제를 해결하기 위해 각 자산의 최소 및 최대 투자 비중 제약조건을 추가해 줄 필요가 있습니다.

### 11.1.5 최소 및 최대 투자비중 제약조건

구석해 문제를 방지하고, 모든 자산에 골고루 투자하기 위해 개별 투자비중을 최소 5%, 최대 20%로 하는 제약조건을 추가해 주도록 하겠습니다. 먼저 `slsqp()` 함수에서 제약조건을 추가하는 방법은 다음과 같습니다.

```
result = slsqp( x0 = rep(0.1, 10),
                 fn = objective,
                 hin = hin.objective,
                 heq = heq.objective,
                 lower = rep(0.05, 10),
                 upper = rep(0.20, 10))

w_4 = result$par %>% round(., 4) %>%
  setNames(colnames(rets))

print(w_4)
```

```
##  SPY  IEV  EWJ  EEM  TLT  IEF  IYR  RWX  GLD  DBC
## 0.05 0.05 0.05 0.05 0.20 0.20 0.05 0.05 0.20 0.10
```

함수의 마지막에 `lower`와 `upper` 제약조건을 추가로 입력할 경우, 해당 값 사이에서 최적화를 만족하는 해를 찾게 되며, 해당 예에서는 5%와 20% 사이에서 해를 찾게 됩니다. 추가로 입력한 제약조건에 맞게, 최소 투자비중은 5%이며, 최대 투자비중은 20%임을 확인할 수 있습니다.

다음은 `solve.QP()` 함수 내에서 제약조건을 추가하는 방법입니다. 해당 함수 역시 다른 입력값은 모두 동일하며, 제약조건의 우변에 해당하는 `bvec` 항목만 수정하면 됩니다. 최소, 최대 투자비중 제약 조건을 기준  $[0, 1]$ 에서  $[0.05, 0.20]$ 로 변경하면, `bvec`에 해당하는 행렬은 다음과 같이 변경됩니다.

$$\text{기준 : } \begin{bmatrix} w_1 + w_2 + \cdots + w_{10} \\ w_1 \\ \vdots \\ w_{10} \\ -w_1 \\ \vdots \\ -w_{10} \end{bmatrix} \geq \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ -1 \\ \vdots \\ -1 \end{bmatrix}$$

$$\text{변경 : } \begin{bmatrix} w_1 + w_2 + \cdots + w_{10} \\ w_1 \\ \vdots \\ w_{10} \\ -w_1 \\ \vdots \\ -w_{10} \end{bmatrix} \geq \begin{bmatrix} 1 \\ 0.05 \\ \vdots \\ 0.05 \\ -0.20 \\ \vdots \\ -0.20 \end{bmatrix}$$

```
Dmat = covmat
dvec = rep(0, 10)
Amat = t(rbind(rep(1, 10), diag(10), -diag(10)))
bvec = c(1, rep(0.05, 10), -rep(0.20, 10))
meq = 1

result = solve.QP(Dmat, dvec, Amat, bvec, meq)

w_5 = result$solution %>% round(., 4) %>%
  setNames(colnames(rets))

print(w_5)
```

```
##   SPY   IEV   EWJ   EEM   TLT   IEF   IYR   RWX   GLD   DBC
## 0.05 0.05 0.05 0.05 0.20 0.20 0.05 0.05 0.20 0.10
```

bvec 항목을 제외한 모든 코드는 기존과 동일하며, 조건함수의 우변인 bvec만 각각 최소 투자비중과 최대 투자비중이 [0, 1]에서 [0.05, 0.20]으로 변경되었습니다. 해당 방법 역시 추가적인 투자비중 제약이 잘 적용되었음이 확인됩니다.

마지막으로 `optimalPortfolio()` 함수 내에서 최소 및 최대 투자비중을 추가하는 방법입니다. 입력변수의 control 항목 중 constraint 부분을 간단하게 수정하여 원하는 조건을 입력할 수 있습니다.

```
w_6 = optimalPortfolio(covmat,
                        control = list(type = 'minvol',
                                      constraint = 'user',
                                      LB = rep(0.05, 10),
                                      UB = rep(0.20, 10))) %>%
  round(., 4) %>%
  setNames(colnames(rets))

print(w_6)
```

```
## SPY IEV EWJ EEM TLT IEF IYR RWX GLD DBC
## 0.05 0.05 0.05 0.05 0.20 0.20 0.05 0.05 0.20 0.10
```

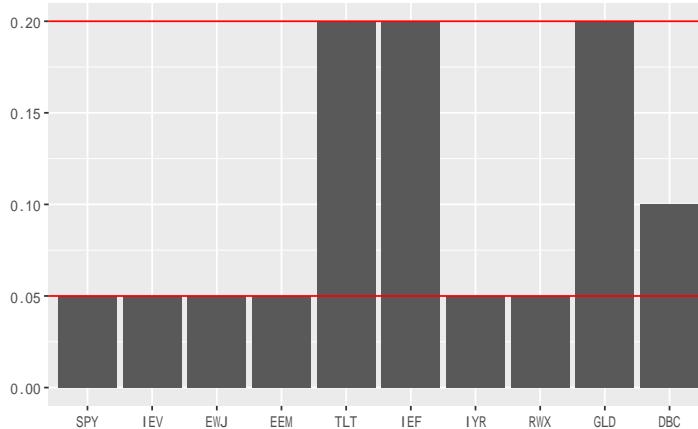
constraint 부분에 롱온리 제약조건에 해당하는 **lo** 대신 직접 제약값들을 입력할 수 있는 **user**를 입력하면, LB에는 최소 투자비중 벡터를, UB에는 최대 투자비중 벡터를 입력합니다. 결과적으로 원하는 제약조건 내에서 결과값이 계산됩니다.

Table 11.7: 최소 및 최대 비중제약 조건 후 결과 비교

	SPY	IEV	EWJ	EEM	TLT	IEF	IYR	RWX	GLD	DBC
slsqp	0.05	0.05	0.05	0.05	0.2	0.2	0.05	0.05	0.2	0.1
solve.QP	0.05	0.05	0.05	0.05	0.2	0.2	0.05	0.05	0.2	0.1
optimalPortfolio	0.05	0.05	0.05	0.05	0.2	0.2	0.05	0.05	0.2	0.1

최소 및 최대 제약 조건을 추가한 경우도, 3가지 방법 모두 동일한 결과가 나오게되며, 비중도 각각 5%와 20%로 제한되어 구석해 문제 또한 해결되었음이 확인됩니다.

```
data.frame(w_4) %>%
  ggplot(aes(x = factor(rownames(.), levels = rownames(.)),
             y = w_4)) +
  geom_col() +
  geom_hline(aes(yintercept = 0.05), color = 'red') +
  geom_hline(aes(yintercept = 0.20), color = 'red') +
  xlab(NULL) + ylab(NULL)
```



### 11.1.6 각 자산 별 제약조건의 추가

투자 규모가 크지 않을 경우에는 위에서 추가한 제약조건 만으로도 충분히 훌륭한 포트폴리오가 구성됩니다. 그러나 투자 규모가 커질 경우 추가적인 제약조건

들을 고려해야 할 경우가 생깁니다. 벤치마크 비중과의 괴리로 인한 추적오차 (Tracking Error)을 고려해야 할 수도 있고, 투자 대상 별 거래량을 고려한 제약 조건을 추가해야 할 경우도 있습니다.

기존 제약조건에는 자산 별로 동일한 최소 및 최대 투자비중 제약조건을 다루었지만, 자산 별로 상이한 제약조건이 필요할 경우도 있습니다. `slsqp()` 와 `optimalPortfolio()` 함수에서는 복잡한 제약조건을 다루기가 힘들지만, `solve.QP()` 함수의 경우 `bvec` 부분을 간단하게 수정하여 어렵지 않게 구현이 가능합니다.

먼저 표 11.8는 새롭게 설정하고자 하는 각 자산 별 최소 및 최대 제약조건입니다.

Table 11.8: 각 자산 별 최소 및 최대 제약조건

제약	1	2	3	4	5	6	7	8	9	10
최소	0.10	0.10	0.05	0.05	0.10	0.10	0.05	0.05	0.03	0.03
최대	0.25	0.25	0.20	0.20	0.20	0.20	0.10	0.10	0.08	0.08

이를 행렬의 형태로 나타내면 다음과 같습니다.

$$\begin{bmatrix} 1 & \dots & 1 \\ 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \\ -1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & -1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{10} \end{bmatrix} = \begin{bmatrix} w_1 + w_2 + \dots + w_{10} \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \\ w_{10} \\ -w_1 \\ -w_2 \\ -w_3 \\ -w_4 \\ -w_5 \\ -w_6 \\ -w_7 \\ -w_8 \\ -w_9 \\ -w_{10} \end{bmatrix} \geq \begin{bmatrix} 1 \\ 0.10 \\ 0.10 \\ 0.05 \\ 0.05 \\ 0.10 \\ 0.10 \\ 0.05 \\ 0.05 \\ 0.03 \\ 0.03 \\ -0.25 \\ -0.25 \\ -0.20 \\ -0.20 \\ -0.20 \\ -0.20 \\ -0.10 \\ -0.10 \\ -0.08 \\ -0.08 \end{bmatrix}$$

위의 행렬 중 우측 부분을 bvec에 그대로 입력해 주도록 합니다.

```
Dmat = covmat
dvec = rep(0, 10)
Amat = t(rbind(rep(1, 10), diag(10), -diag(10)))
bvec = c(1, c(0.10, 0.10, 0.05, 0.05, 0.10,
             0.10, 0.05, 0.05, 0.03, 0.03),
        -c(0.25, 0.25, 0.20, 0.20, 0.20,
           0.20, 0.10, 0.10, 0.08, 0.08))
meq = 1

result = solve.QP(Dmat, dvec, Amat, bvec, meq)

result$solution %>%
  round(., 4) %>%
  setNames(colnames(rets))
```

```
##  SPY  IEV  EWJ  EEM  TLT  IEF  IYR  RWX  GLD  DBC
## 0.14 0.10 0.05 0.05 0.20 0.20 0.05 0.05 0.08 0.08
```

결과값을 확인해보면, 각 자산 별 제약조건 내에 위치함을 확인할 수 있습니다.

## 11.2 최대분산효과 포트폴리오

앞서 설명했듯이 포트폴리오의 변동성은  $\sum_{i=1}^n \sum_{j=1}^n w_i w_j \sigma_{ij}$  형태로 나타나며, 이는 다음과 같이 표현할 수도 있습니다.

$$\sigma_p^2 = \sum_{i=1}^n \sum_{j=1}^n w_i w_j \sigma_{ij} = \sum_{i=1}^n w_i^2 \sigma_i^2 + \sum_{i=1}^n \sum_{i \neq j}^n w_i w_j \rho_{ij} \sigma_i \sigma_j$$

이 중  $\sum_{i=1}^n \sum_{i \neq j}^n w_i w_j \rho_{ij} \sigma_i \sigma_j$  부분에는 자산 간 상관관계( $\rho_{ij}$ )가 포함되어 있습니다. 상관관계는 -1과 1 사이에 위치하며 상관관계가 1, 즉 두 자산이 완벽하게 동일한 경우에는 포트폴리오의 변동성은 개별 자산 변동성의 가중합과 같습니다. 그러나 상관관계가 낮아질수록 포트폴리오의 변동성 또한 점차 낮아집니다. 이러한 효과를 투자에서는 **분산효과**라 합니다.

이러한 분산효과의 정도를 측정하는 지표가 **분산 비율**(DR: Diversification Ratio)입니다. 분산 비율의 분자는 개별 변동성의 가중합이며, 분모는 포트폴리오의 변동성입니다. 이를 수식으로 나타내면 다음과 같습니다.

$$\text{분산 비율} = \frac{\sum w_i \sigma_i}{\sigma_p} = \frac{w' \sigma}{\sqrt{w' \Omega w}}$$

모든 자산 간의 상관관계가 1일 경우, 위의 예시에서 살펴본 것과 같이 포트폴리오의 변동성은 개별 자산 변동성의 가중합과 같아지게 됩니다. 즉,  $\sum w_i \sigma_i = \sigma_p$ 가 되어, 분산 비율은 1이 됩니다. 그러나 대부분의 경우에서 자산 간의 상관관계는 1보다 낮으며, 이로 인해 포트폴리오의 분산은 단순 가중합 보다 작아지게 되고 ( $\sigma_p < \sum w_i \sigma_i$ ), 분산 비율은 1보다 커지게 됩니다.

자산 간 상관관계가 낮은 종목을 위주로 포트폴리오를 구성할 수록 분산효과로 인해 포트폴리오의 변동성은 낮아지게 되고, 분산 비율은 점점 커지게 됩니다. 최대분산 포트폴리오(Most Diversified Portfolio)는 분산효과가 최대가 되는, 즉 분산 비율이 최대가 되는 포트폴리오를 구성하는 방법입니다. 이에 대한 목적함수와 제약조건은 다음과 같습니다.

$$\text{목적함수} : \max DR = \max \frac{\sum w_i \sigma_i}{\sigma_p}$$

$$\text{제약조건} : \sum_{i=1}^n w_i = 1, w_i \geq 0$$

최대분산효과 포트폴리오의 목적함수는 분산비율을 최대화하는데 있는 반면, 대부분의 최적화 프로그래밍은 목적함수를 최소화 하는 형태로 이루어집니다. 따라서 목적함수인  $\max DR$ 을 최소화 형태로 바꿀 필요가 있으며 크게 세가지 방법이 있습니다.

1. Choueifaty Synthetic Asset Back-Transformation을 이용하는 방법 <sup>1</sup>
2. Duality를 이용하는 방법 <sup>2</sup>
3. Min (-)DR 방법

먼저 Choueifaty Synthetic Asset Back-Transformation 방법은 목적함수  $\min w_s' c w_s$  와 제약조건  $\sum_{i=1}^n w_i = 1, w_i \geq 0$ 을 만족하는 자산 별 비중을 구합니다. 그 후, 구해진 비중을 각각의 표준편차로 나누어 주며, 비중의 합이 1이 되도록 표준화를 해줍니다. 이 중 주의할 점은 목적함수의  $c$ 가 우리가 지금까지 사용하던 분산-공분산 행렬이 아닌, **상관관계 행렬**이라는 점입니다.

Duality 방법의 목적함수는 최소분산 포트폴리오와 동일한  $\min \frac{1}{2} w' \sigma w$ 이며, 제약조건만  $\sum_{i=1}^n w_i \sigma_i = 1, w_i \geq 0$ , 즉 개별 자산의 비중이 0보다 크고 개별

---

<sup>1</sup>Choueifaty, Y., & Coignard, Y. (2008). Toward maximum diversification. Journal of Portfolio Management, 35(1), 40.

<sup>2</sup>Choueifaty, Y., Froidure, T., & Reynier, J. (2011). Properties of the most diversified portfolio.

표준편차의 가중합이 1인 조건으로 바뀝니다. 그 후, 비중의 합이 1이 되도록 표준화를 해줍니다.

기존 두 방법이 수학적 증명에 의해  $\max DR$ 을 최소화의 형태로 풀어준 반면, 간단하게 목적함수를  $\min(-DR)$ 의 형태로 바꾸어 풀 수도 있습니다. 표 11.3은 3가지 방법을 요약한 내용입니다.

Table 11.9: MDP 방법 비교

방법	목적함수	제약조건	표준화
Transformation	$\min w'_s c w_s$	$\sum_{i=1}^n w_i = 1$ $w_i \geq 0$	비중을 각각의 표준편차로 나눈 후 비중의 합으로 표준화
Duality	$\min \frac{1}{2} w' \sigma w$	$\sum_{i=1}^n w_i \sigma_i = 1$ $w_i \geq 0$	비중의 합으로 표준화
-DR	$\min(-DR)$	$\sum_{i=1}^n w_i = 1$ $w_i \geq 0$	불필요

### 11.2.1 solve.QP() 함수를 이용한 최적화

먼저 `solve.QP()` 함수를 이용하여 Duality 방법을 통해 최대분산효과 포트폴리오를 만족하는 해를 찾도록 하겠습니다.

Duality 방법에서 목적함수는  $\min \frac{1}{2} w' \sigma w$ 로 최소분산 포트폴리오와 동일하며, 제약조건은  $\sum_{i=1}^n w_i \sigma_i = 1, w_i \geq 0$ 입니다. 제약조건 부분인 Amat과 bvec 부분을 입력할 때 이 부분을 고려하여 입력해야 합니다.

```
Dmat = covmat
dvec = rep(0, 10)
Amat = t(rbind(sqrt(diag(covmat)), diag(10)))
bvec = c(1, rep(0, 10))
meq = 1
```

제약조건에 해당하는 Amat 부분과 bvec 부분은 최소분산 포트폴리오와 다소 다릅니다. 표 11.10에는 둘 간에 코드가 어떻게 다른지 나타나 있습니다.

Table 11.10: Amat과 bvec 차이 비교

인자	최소분산 포트폴리오	최대분산효과 포트폴리오
Amat	<code>t(rbind(rep(1, 10), diag(10), -diag(10)))</code>	<code>t(rbind(sqrt(diag(covmat)), diag(10)))</code>
bvec	<code>c(1, rep(0, 10), -rep(1, 10))</code>	<code>c(1, rep(0, 10))</code>

이해를 위해 Duality 방법의 제약조건을 행렬의 형태로 표현하면 다음과 같습니다.

$$\begin{bmatrix} \sigma_1 & \dots & \sigma_{10} \\ 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{10} \end{bmatrix} = \begin{bmatrix} \sigma_1 w_1 + \sigma_2 w_2 + \dots + \sigma_{10} w_{10} \\ w_1 \\ \vdots \\ w_{10} \end{bmatrix} \geq \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

1행의  $\sigma_1 w_1 + \sigma_2 w_2 + \dots + \sigma_{10} w_{10}$  부분은  $\sum_{i=1}^n w_i \sigma_i = 1$  과 같으며, 해당 식은 등위제약조건으로써  $\sum_{i=1}^n w_i \sigma_i = \sigma_1 w_1 + \sigma_2 w_2 + \dots + \sigma_{10} w_{10} = 1$  을 의미합니다. 2행부터 마지막 행까지는 모두  $w_i \geq 0$  조건으로써, 개별 자산의 투자 비중이 0 보다 큰 조건을 의미합니다.

행렬의 맨좌측에 해당하는 Amat 부분은 각 자산의 표준편차로 이루어진 벡터 행렬과, 1로 이루어진 대각행렬로 구성되어 있습니다. 먼저 diag(covmat)을 통해 분산-공분산 부분에서 대각부분 즉 분산 부분만을 추출할 수 있습니다. 개별 자산의 분산인  $\sigma_{i,i}$ 의 경우  $\sigma_i \sigma_i \rho_{1,1}$  형태로 쓸 수 있으며,  $\rho_{1,1} = 1$  을 적용하면  $\sigma_i^2$  와 같습니다. 따라서 대각부분 값에 제곱근을 계산하는 sqrt() 함수를 적용하면 각각의 표준편차만 남게 됩니다. 이를 diag(10)을 통해 만든 대각행렬과 행으로 묶어준 후, 전치행렬을 입력해 줍니다.

bvec의 경우 행렬의 맨우측과 같이 등위제약조건에 해당하는 1과 부등위제약조건에 해당하는 0들로 구성되어 있습니다. 차후에 표준화 과정을 거쳐야 하므로 개별 자산의 투자 비중이 1보다 작은 조건은 Duality 방법에서는 입력하지 않아도 됩니다.

```
result = solve.QP(Dmat, dvec, Amat, bvec, meq)
```

```
w = result$solution %>%
  round(., 4) %>%
  setNames(colnames(rets))

print(w)
```

```
##    SPY     IEV     EWJ     EEM     TLT     IEF     IYR     RWX
## 17.404  2.308  3.787  0.000 27.782 36.317  3.016  0.000
##    GLD     DBC
##  8.405 11.688
```

입력된 목적함수와 제약조건들을 바탕으로 solve.QP()를 통해 최적화를 수행한 후, 최대분산효과를 만족하는 해를 구해보면, 비중의 합이 1을 초과하게 됩니다.

$w_i = \frac{w_i}{\sum_{i=1}^n w_i}$  를 통해 비중의 합이 1이 되도록 표준화를 해줍니다.

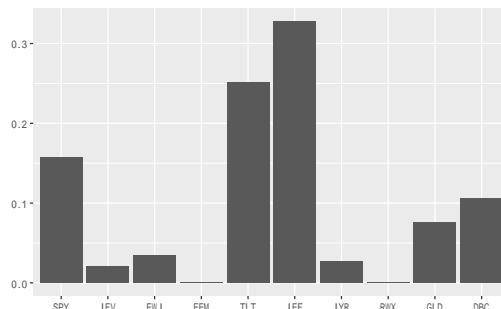
```
w = (w / sum(w)) %>%
  round(., 4)

print(w)

##      SPY      IEV      EWJ      EEM      TLT      IEF      IYR      RWX
## 0.1572 0.0208 0.0342 0.0000 0.2510 0.3280 0.0272 0.0000
##      GLD      DBC
## 0.0759 0.1056
```

표준화 과정을 통해 비중의 합이 1이 되었습니다.

```
data.frame(w) %>%
  ggplot(aes(x = factor(rownames(.), levels = rownames(.)),
             y = w)) +
  geom_col() +
  geom_col() +
  xlab(NULL) + ylab(NULL)
```



### 11.2.2 optimalPortfolio() 함수를 이용한 최적화

최소분산 포트폴리오와 동일하게 `optimalPortfolio()` 함수를 이용하여 매우 간단하게 최대분산효과 포트폴리오를 구현할 수 있습니다.

```
w = optimalPortfolio(covmat,
                      control = list(type = 'maxdiv',
                                     constraint = 'lo')) %>%
  round(., 4)

print(w)
```

```
## [1] 0.1572 0.0208 0.0342 0.0000 0.2510 0.3280 0.0272
## [8] 0.0000 0.0759 0.1056
```

control 항목의 type에 maximum diversification을 의미하는 ‘maxdiv’를 입력해주며, 제약조건에는 투자 비중이 0보다 큰 **lo**(long only) 조건을 입력해 줍니다. 폐기지를 활용하여 매우 간단하게 최대분산효과 포트폴리오를 구현할 수 있으며, 그 결과 또한 앞에서 계산한 것과 동일합니다. 해당 함수의 코드를 확인해보면, 최대분산효과 포트폴리오 계산시 -DR 방법 방법을 사용합니다.

### 11.2.3 최소 및 최대 투자비중 제약조건

최대분산효과 포트폴리오 역시 구석해 문제가 발생하며, 모든 자산에 골고루 투자하기 위해 개별 투자비중을 최소 5%, 최대 20%로 하는 제약조건을 추가해 주도록 하겠습니다.

Duality 방법에서는 목적함수인  $\min_{w} \frac{1}{2} w' \sigma w$ 과 제약조건인  $\sum_{i=1}^n w_i \sigma_i = 1, w_i \geq 0$ 에 맞게 해를 구한 후, 비중의 합이 1이 되도록 표준화하는 과정을 거쳤습니다. 따라서 비중의 최소 및 최대 제약조건은 단순히  $lb \leq w_i \leq ub$ 가 아닌 표준화 과정인  $w_i = \frac{w_i}{\sum_{i=1}^n w_i}$ 까지 고려하여 적용해 주어야 합니다. 표 11.11는 이를 수식으로 나타낸 것입니다.

Table 11.11: 최소 및 최대비중 제약조건

최소비중 제약조건	최대비중 제약조건
$\sum_{i=1}^n \frac{w_i}{w_i} \geq lb$	$\sum_{i=1}^n \frac{w_i}{w_i} \leq ub$
$\Rightarrow -lb + \sum_{i=1}^n \frac{w_i}{w_i} \geq 0$	$\Rightarrow ub - \sum_{i=1}^n \frac{w_i}{w_i} \geq 0$
$\Rightarrow -lb + \frac{w_i}{e^T w} \geq 0$	$\Rightarrow ub - \frac{w_i}{e^T w} \geq 0$
$\Rightarrow -lb \times e^T w + w \geq 0$	$\Rightarrow ub \times e^T w - w \geq 0$
$\Rightarrow (-lb \times e^T + I)w \geq 0$	$\Rightarrow (ub \times e^T - I)w \geq 0$

최소 비중 제약조건인  $-lb \times e^T + I$ 의 예를 행렬로 풀어보도록 하겠습니다.  $-lb \times e^T$ 의 경우 행렬로 표현할 경우 다음과 같으며,  $-lb$ 로 이루어진  $n \times n$  행렬입니다.

$$\begin{bmatrix} -lb \\ \vdots \\ -lb \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}^T = \begin{bmatrix} -lb \\ \vdots \\ -lb \end{bmatrix} \begin{bmatrix} 1 & \dots & 1 \end{bmatrix} = \begin{bmatrix} -lb & \dots & -lb \\ \vdots & \ddots & \vdots \\ -lb & \dots & -lb \end{bmatrix}$$

$I$ 는 대각선 부분이 1, 나머지가 0인 항등행렬을 의미합니다. 따라서  $(-lb \times e^T + I)$ 를 계산하면 다음과 같습니다.

$$\begin{bmatrix} -lb + 1 & \dots & -lb \\ \vdots & \ddots & \vdots \\ -lb & \dots & -lb + 1 \end{bmatrix}$$

최소분산 포트폴리오와는 다르게 Duality 방법으로 최대분산효과 포트폴리오를 구현할 경우 최소 및 최대 제약조건이 우변이 아닌 좌변에 들어가게 되며, 해당 제약조건을 고려하여 행렬로 표현하면 다음과 같습니다.

$$\begin{bmatrix} \sigma_1 & \dots & \sigma_{10} \\ -lb + 1 & \dots & -lb \\ \vdots & \ddots & \vdots \\ -lb & \dots & -lb + 1 \\ ub - 1 & \dots & ub \\ \vdots & \ddots & \vdots \\ ub & \dots & ub - 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{10} \end{bmatrix} = \begin{bmatrix} \sigma_1 w_1 + \sigma_2 w_2 + \dots + \sigma_{10} w_{10} \\ -lb(w_1 + w_2 + \dots + w_{10}) + w_1 \\ \vdots \\ -lb(w_1 + w_2 + \dots + w_{10}) + w_{10} \\ wb(w_1 + w_2 + \dots + w_{10}) - w_1 \\ \vdots \\ wb(w_1 + w_2 + \dots + w_{10}) - w_{10} \end{bmatrix} \geq \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

첫번째 행  $\sigma_1 w_1 + \sigma_2 w_2 + \dots + \sigma_{10} w_{10}$ 은 등위 제약조건인  $\sum_{i=1}^n w_i \sigma_i = 1$ 에 해당하며, 두번째 행부터는 부등위 제약조건에 해당합니다. 두번째 행을 정리하면  $\frac{w_1}{w_1 + w_2 + \dots + w_{10}} \geq lb$ , 즉 비중의 표준화가 고려된 최소비중 제약조건입니다. 마지막 행 역시 정리하면  $\frac{w_1}{w_1 + w_2 + \dots + w_{10}} \leq ub$ 가 되어 비중의 표준화가 고려된 최대비중 제약조건을 의미합니다. 위 행렬을 고려하여 Amat과 bvec을 수정한 코드는 다음과 같습니다.

```
Dmat = covmat
dvec = rep(0, 10)
Alb = -rep(0.05, 10) %*% matrix(1, 1, 10) + diag(10)
Aub = rep(0.20, 10) %*% matrix(1, 1, 10) - diag(10)

Amat = t(rbind(sqrt(diag(covmat)), Alb, Aub))
bvec = c(1, rep(0, 10), rep(0, 10))
meq = 1

result = solve.QP(Dmat, dvec, Amat, bvec, meq)

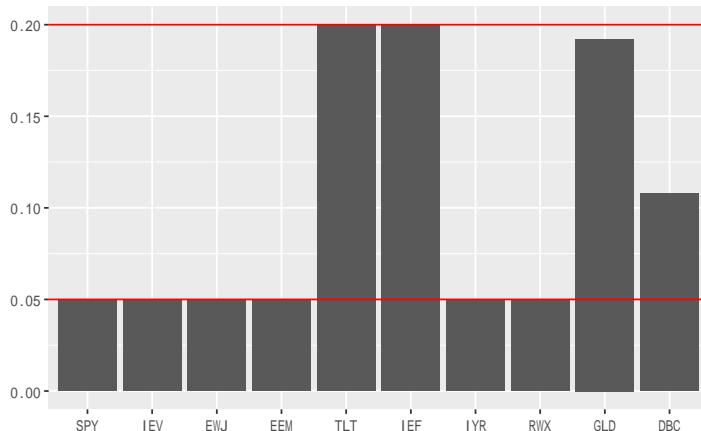
w = result$solution
w = (w / sum(w)) %>%
  round(., 4) %>%
  setNames(colnames(rets))
```

```
print(w)
```

```
##      SPY      IEV      EWJ      EEM      TLT      IEF      IYR      RWX
## 0.0500 0.0500 0.0500 0.0500 0.2000 0.2000 0.0500 0.0500
##      GLD      DBC
## 0.1922 0.1078
```

Alb의  $-\text{rep}(0.05, 10)$ 은  $-\text{lb}$  부분,  $\text{matrix}(1, 1, 10)$ 은  $e^T$  부분,  $\text{diag}(10)$ 부분은  $I$  부분을 의미하며, 이는 최소비중 제약조건의 좌변부분( $-\text{lb} \times e^T + I$ )과 같습니다. 동일하게 Aub는 최대비중 제약조건의 좌변부분( $ub \times e^T - I$ )과 같으며, 결과를 확인하면 최소 및 최대비중 제약조건인 [5%, 20%]가 제대로 반영되었습니다.

```
data.frame(w) %>%
  ggplot(aes(x = factor(rownames(.)), levels = rownames(.)),
         y = w)) +
  geom_col() +
  geom_hline(aes(yintercept = 0.05), color = 'red') +
  geom_hline(aes(yintercept = 0.20), color = 'red') +
  xlab(NULL) + ylab(NULL)
```



#### 11.2.4 각 자산 별 제약조건의 추가

최소분산 포트폴리오의 경우와 동일하게 자산 별로 다른 제약조건을 추가하여 포트폴리오를 구성하도록 하겠습니다. 표 11.12는 각 자산 별 최소 및 최대 투자비중 값이며, 변경된 제약조건을 행렬의 형태로 나타내었습니다. 주의해야 할

Table 11.12: 각 자산 별 최소 및 최대 제약조건

제약	1	2	3	4	5	6	7	8	9	10
최소	0.10	0.10	0.05	0.05	0.10	0.10	0.05	0.05	0.03	0.03
최대	0.25	0.25	0.20	0.20	0.20	0.20	0.10	0.10	0.08	0.08

점은 최소비중과 최대비중의 제약조건 추가 시,  $\frac{w_1}{w_1+w_2+\dots+w_{10}} \geq lb$  형태로 고려해 주어야 한다는 점입니다.

$$\begin{bmatrix} \sigma_1 & \sigma_2 & \dots & \sigma_{10} \\ -lb_1 + 1 & -lb_1 & \dots & -lb_1 \\ -lb_2 & -lb_2 + 1 & \dots & -lb_2 \\ \vdots & \ddots & \dots & \vdots \\ -lb_{10} & -lb_{10} & \dots & -lb_{10} + 1 \\ ub_1 - 1 & ub_1 & \dots & ub_1 \\ ub_2 & ub_2 - 1 & \dots & ub_2 \\ \vdots & \ddots & \dots & \vdots \\ ub_{10} & ub_{10} & \dots & ub_{10} - 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{10} \end{bmatrix} = \begin{bmatrix} \sigma_1 w_1 + \sigma_2 w_2 + \dots + \sigma_{10} w_{10} \\ -lb_1(w_1 + w_2 + \dots + w_{10}) + w_1 \\ -lb_2(w_1 + w_2 + \dots + w_{10}) + w_2 \\ \vdots \\ -lb_{10}(w_1 + w_2 + \dots + w_{10}) + w_{10} \\ wb_1(w_1 + w_2 + \dots + w_{10}) - w_1 \\ wb_2(w_1 + w_2 + \dots + w_{10}) - w_2 \\ \vdots \\ ub_{10}(w_1 + w_2 + \dots + w_{10}) - w_{10} \end{bmatrix} \geq \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

기준에 공통적으로 적용되던 최소 및 최대 투자비중이 자산 별로 다르게 구성되었습니다. 따라서  $-lb \times e^T + I$  와  $ub \times e^T - I$  부분만  $-lb_i \times e^T + I$   $ub_i \times e^T - I$ 로 수정하면 해당 제약조건 역시 손쉽게 구현이 가능합니다.

```
Dmat = covmat
dvec = rep(0, 10)
Alb = -c(0.10, 0.10, 0.05, 0.05, 0.10,
        0.10, 0.05, 0.05, 0.03, 0.03) %*%
  matrix(1, 1, 10) + diag(10)
Aub = c(0.25, 0.25, 0.20, 0.20, 0.20,
        0.20, 0.10, 0.10, 0.08, 0.08) %*%
  matrix(1, 1, 10) - diag(10)

Amat = t(rbind(sqrt(diag(covmat)), Alb, Aub))
bvec = c(1, rep(0, 10), rep(0, 10))
meq = 1

result = solve.QP(Dmat, dvec, Amat, bvec, meq)

w = result$solution
w = (w / sum(w)) %>%
  round(., 4) %>%
```

```

setNames(colnames(rets))

print(w)

## SPY IEV EWJ EEM TLT IEF IYR RWX GLD DBC
## 0.10 0.10 0.09 0.05 0.20 0.20 0.05 0.05 0.08 0.08

```

최소 및 최대투자비중 제약조건을 나타내는 Alb와 Aub 부분이 자산 별 각각의 제약비중으로 변경되었으며, 나머지 부분은 모두 동일합니다. 결과 값들이 모두 제약조건 내에 위치함을 확인할 수 있습니다.

## 11.3 위험균형 포트폴리오

포트폴리오를 구성하는 자산들과 전체 위험의 관계를 이해하기 위해서는, 먼저 한계 위험기여도(MRC: Marginal Risk Contribution)와 위험기여도(RC: Risk Contribution)에 대해 알아야 합니다. 한계 위험기여도는 특정 자산의 비중을 한 단위 증가시켰을 때 전체 포트폴리오의 위험의 증가를 나타내는 단위로써, 수학의 편미분과 같은 개념입니다.  $i$ 번째 자산의 한계 위험기여도는 아래와 같이 나타낼 수 있습니다.

$$MRC_i = \frac{\partial \sigma_p}{\partial w_i}$$

$\sqrt{f'(x)} = \frac{f'(x)}{2\sqrt{f(x)}}$  인 사실을 이용하면, 한계 위험기여도는 아래와 같이 풀 수 있습니다. 결과적으로 분자는 분산-공분산 행렬과 각 자산의 비중의 곱, 분모는 포트폴리오의 표준편차 형태로 나타납니다.

$$\begin{aligned}
\frac{\partial \sigma_p}{\partial w} &= \frac{\partial(\sqrt{w' \Omega w})}{\partial w} \\
&= \frac{\partial(w' \Omega w)}{\partial w} \times \frac{1}{2\sqrt{w' \Omega w}} \\
&= \frac{2\Omega w}{2\sqrt{w' \Omega w}} \\
&= \frac{\Omega w}{\sqrt{w' \Omega w}}
\end{aligned}$$

위험기여도는 특정 자산이 포트폴리오 내에서 차지하는 위험의 비중입니다. 한계 위험기여도가 큰 자산도 포트폴리오 내에서 비중이 작다면, 포트폴리오 내에서 차지하는 위험의 비중은 작을 것입니다. 반면에, 한계 위험기여도가 작은 자산일지라도 비중이 압도적으로 많다면, 포트폴리오 내에서 차지하는 위험의 비중은 클 것입니다.

결과적으로  $i$  번째 자산의 위험기여도는,  $i$  번째 자산의 한계 위험기여도와 포트폴리오 내 비중의 곱으로 이루어집니다.

$$RC_i = \frac{\partial \sigma_p}{\partial w_i} \times w_i$$

위험기여도를 코드로 나타내면 다음과 같습니다. 먼저 포트폴리오 비중인  $w$ 와 분산-공분산 행렬인  $covmat$ 을 이용하여 한계 위험기여도를 계산하여 줍니다. 그 후, 비중  $w$ 를 곱하여 위험기여도를 계산해 준 후, 합계가 1이 되도록 표준화를 해줍니다.

```
get_RC = function(w, covmat) {
  port_vol = t(w) %*% covmat %*% w
  port_std = sqrt(port_vol)

  MRC = (covmat %*% w) / as.numeric(port_std)
  RC = MRC * w
  RC = c(RC / sum(RC))

  return(RC)
}
```

### 11.3.1 주식 60%와 채권 40% 포트폴리오의 위험기여도

자산배분에서 가장 많이 사용되는 투자방법은 주식에 60%, 채권에 40% 가량의 비율로 투자하는 것입니다. 주식과 채권이 서로 상관관계가 낮아 분산효과가 있다는 점, 장기적으로 주식이 채권에 비해 장기적으로 수익률이 높다는 점을 감안하면 이는 꽤나 합리적인 방법으로 보입니다.

그러나 눈에 보이는 비중이 60대 40이라도, 포트폴리오 내에서 각 자산이 가지고 있는 위험기여도 60대 40의 비중이 아닌 전혀 다른 비중을 가지고 있습니다.

```
ret_stock_bond = rets[, c(1, 5)]
cov_stock_bond = cov(ret_stock_bond)
RC_stock_bond = get_RC(c(0.6, 0.4), cov_stock_bond)
```

```
RC_stock_bond = round(RC_stock_bond, 4)
print(RC_stock_bond)
```

```
## [1] 0.9692 0.0308
```

rets 데이터에서 첫번째 행은 미국 주식 수익률을, 다섯번째 행은 미국 장기채를 의미하므로, 해당 부분을 ret\_stock\_bond 변수에 지정합니다. 그 후 cov() 함수를 이용해 두 자산의 분산-공분산 행렬을 만들어 주며, 위에서 만든 get\_RC 함수를 통해 자산 별 위험기여도를 계산하도록 합니다.

주식과 채권이 가지는 위험기여도는 각각 96.92%, 3.08%로써 투자 비중인 60%, 40% 와는 전혀 다른 위험 비중을 보입니다. 즉, 주식이 포트폴리오 위험의 대부분을 차지하고 있습니다.

### 11.3.2 rp() 함수를 이용한 최적화

위 예제와 같이 특정 자산이 포트폴리오의 위험을 대부분 차지하는 문제를 막고, 모든 자산이 동일한 위험기여도를 가지는 포트폴리오가 위험균형 포트폴리오(Risk Parity Portfolio) 혹은 동일 위험기여도 포트폴리오(Equal Risk Contribution Portfolio)입니다. 이를 수식으로 쓰면 다음과 같습니다.

$$RC_1 = RC_2 = \dots = RC_n$$

$$\frac{\partial \sigma_p}{\partial w_1} \times w_1 = \frac{\partial \sigma_p}{\partial w_2} \times w_2 = \dots = \frac{\partial \sigma_p}{\partial w_n} \times w_n = \frac{1}{n}$$

위험균형 포트폴리오 역시 slsqp()나 optimalPortfolio() 함수를 이용하여 구현할 수 있으나, 간혹 최적화된 값을 찾지 못할 때도 있습니다. 반면 cccp 패키지의 rp() 함수를 사용하면 매우 정확하게 위험균형 포트폴리오를 구성하는 비중을 계산할 수 있습니다.

```
library(cccp)
opt = rp(x0 = rep(0.1, 10),
          P = covmat,
          mrc = rep(0.1, 10))
```

```
w = getx(opt) %>% drop()
w = (w / sum(w)) %>%
  round(., 4) %>%
  setNames(colnames(rets))

print(w)

##      SPY      IEV      EWJ      EEM      TLT      IEF      IYR      RWX
## 0.0622 0.0470 0.0555 0.0360 0.1809 0.3652 0.0396 0.0514
##      GLD      DBC
## 0.0867 0.0755
```

1.  $x_0$ 은 최적화를 위한 초기 입력값이며 동일비중인 10%씩을 입력해 줍니다.
2.  $P$ 는 분산-공분산 행렬을 입력해줍니다.
3.  $mrc$ 는 목표로 하는 각 자산 별 위험기여도 값<sup>3</sup>이며, 위험균형 포트폴리오의 경우 모든 자산의 위험기여도가 동일해야 하므로 10%씩을 입력해 줍니다.

`rp()` 함수는 위 입력변수를 바탕으로 최적해를 찾아줍니다. `getx()` 함수를 통해 해를 추출할 수 있으며, `drop()`을 통해 벡터 형태로 변환해줍니다. 마지막으로 비중의 합이 1이 되기 위해 비중들의 합으로 나눠주도록 합니다.

최종적으로 계산된 비중이 위험균형 포트폴리오를 만족하는 해가 됩니다.

```
get_RC(w, covmat)
```

```
## [1] 0.10004 0.10011 0.09991 0.09992 0.10001 0.10000
## [7] 0.10002 0.09996 0.10005 0.09999
```

`get_RC()` 함수를 통해 위험기여도를 확인해보면, 모든 자산이 거의 동일한 위험기여도를 가지는 것이 확인됩니다.

### 11.3.3 위험예산 포트폴리오

모든 자산의 위험기여도가 동일한 값이 아닌, 자산 별로 다른 위험기여도를 가지는 포트폴리오를 구성해야 할 경우도 있습니다. 이러한 포트폴리오를 위험예산 포트폴리오(Risk Budget Portfolio)라 합니다. 위험균형 포트폴리오 역시 각

---

<sup>3</sup>엄밀하게는  $mrc$ 가 아닌  $rc$ 가 맞는 용어입니다.

자산의 위험예산이  $\frac{1}{n}$ 로 동일한 특수 형태이며, rp() 함수를 이용하면 위험예산 포트폴리오 역시 손쉽게 구현할 수 있습니다.

먼저 각 자산 별 위험예산을 표 11.13와 같이 정합니다. 1~4번 자산은 각각 15% 씩, 5~6번 자산은 각각 10%씩, 7~10번 자산은 각각 5%씩 위험예산을 부여하고자 합니다.

Table 11.13: 위험예산 포트폴리오 예시

자산	1	2	3	4	5	6	7	8	9	10
예산	0.15	0.15	0.15	0.15	0.1	0.1	0.05	0.05	0.05	0.05

```
library(cccp)
```

```
opt = rp(x0 = rep(0.1, 10),
          P = covmat,
          mrc = c(0.15, 0.15, 0.15, 0.15, 0.10,
                 0.10, 0.05, 0.05, 0.05, 0.05))
```

```
w = getx(opt) %>% drop()
w = (w / sum(w)) %>%
  round(., 4) %>%
  setNames(colnames(rets))
```

```
print(w)
```

```
##      SPY      IEV      EWJ      EEM      TLT      IEF      IYR      RWX
## 0.0873 0.0671 0.0768 0.0515 0.1883 0.3856 0.0198 0.0254
##      GLD      DBC
## 0.0547 0.0435
```

mrc에 목표로 하는 각 자산 별 위험기여도를 입력하며, 나머지는 기존 위험균형 포트폴리오와 동일하게 입력합니다.

```
get_RC(w, covmat)
```

```
## [1] 0.14991 0.15007 0.14991 0.14991 0.10010 0.10009
## [7] 0.05005 0.05003 0.04998 0.04993
```

get\_RC() 함수를 통해 위험기여도를 확인해보면, 우리가 원하던 자산 별 위험예산과 거의 동일함이 확인됩니다.



# CHAPTER 12

---

## 포트폴리오 백테스트

---

백테스트란 현재 생각하는 전략을 과거부터 실행하였을 시 어떠한 성과가 발생하는지 테스트해보는 과정입니다. 과거의 데이터를 기반으로 전략을 실행하는 퀀트 투자에 있어서 이는 핵심 단계이기도 합니다. 백테스트 결과를 통해 해당 전략의 손익뿐만 아니라 각종 위험을 대략적으로 판단할 수 있으며, 어떤 구간에서 전략이 좋았는지 혹은 나빴는지에 대한 이해도 키울 수 있습니다. 이러한 이해를 바탕으로 퀀트 투자를 지속한다면 단기적으로 수익이 나쁜 구간에서도 그 이유에 대한 객관적인 안목을 키울 수 있으며, 확신을 가지고 전략을 지속할 수 있습니다.

그러나 백테스트를 아무리 보수적으로 혹은 엄밀하게 진행하더라도 이미 일어난 결과를 대상으로 한다는 점은 변하지 않습니다. 백테스트 수익률만을 보고 투자에 대한 판단을 하거나, 혹은 동일한 수익률이 미래에도 반복될 것이라 믿는다면 이는 백미러를 보고 운전을 하는 매우 위험한 결과를 초래할 수도 있습니다.

R에서 백테스트는 `PerformanceAnalytics` 패키지의 `Return.portfolio()` 함수를 사용하여 매우 간단하게 수행할 수 있습니다. 이번 장에서는 해당 함수에 대한 이해와 더불어, 구체적인 사용 방법에 대한 예시로써 전통적인 주식 60% & 채권 40% 포트폴리오, 시점 선택 전략, 동적 자산배분에 대한 백테스트를 실시합니다.

## 12.1 Return.portfolio() 함수

프로그래밍을 이용하여 백테스트를 할 경우, 전략이 단순하다면 단 몇 줄 만으로도 테스트가 가능합니다. 그러나 전략이 복잡해지거나 적용해야 할 요소가 많아질 경우, 패키지를 이용하는 것이 효율적인 방법입니다.

`PerformanceAnalytics` 패키지의 `Return.portfolio()` 함수는 백테스트를 수행하는데 가장 대중적으로 사용되는 함수입니다. 해당 함수의 가장 큰 장점은 각 자산의 수익률과 리밸런싱 비중만 있으면 백테스트 수익률, 회전율 등을 쉽게 계산할 수 있다는 점이며, 리밸런싱 시점과 수익률의 시점이 일치하지 않아도 된다는 점입니다. 즉, 수익률 데이터는 일간, 리밸런싱 시점은 분기 혹은 연간으로 된 경우에도 매우 쉽게 백테스트를 수행할 수 있습니다.

### 12.1.1 인자목록 살펴보기

먼저 `Return.portfolio()` 함수는 다음과 같은 형태로 구성되어 있으며, 표 12.1은 인자의 내용을 정리한 것입니다.

```
Return.portfolio(R, weights = NULL, wealth.index = FALSE,
  contribution = FALSE, geometric = TRUE,
  rebalance_on = c(NA, "years", "quarters",
    "months", "weeks", "days"),
  value = 1, verbose = FALSE, ...)
```

이 중 가장 중요한 인자는 개별 자산의 수익률인 R과 리밸런싱 시기의 자산 별 목표 비중인 `weights`입니다. 매 리밸런싱 시점마다 적용되는 자산 별 비중이 동일할 경우(예: 매월 말 60% 대 40% 비중으로 리밸런싱) 상수 형태로 입력하여도 되지만, 시점마다 자산 별 목표비중이 다를 경우 `weights`는 시계열 형태로 입력되어야 합니다.

목표 비중을 시계열 형태로 입력할 경우 주의해야 할 점은 다음과 같습니다.

1. 시계열 형태로 인식할 수 있도록, 행이름 혹은 인덱스가 날짜 형태로 입력되어야 합니다.
2. 수익률 데이터와 비중 데이터의 열 개수는 동일해야 하며, 각 열에 해당하는 자산은 동일해야 합니다. 즉, 수익률 데이터의 첫번째 열에 A주식 데이터가 있다면, 비중 데이터의 첫번째 열도 A주식의 목표 비중을 입력해야 합니다.
3. 각 시점의 비중의 합은 1이 되어야 합니다. 그렇지 않을 경우 제대로된 수익률이 계산되지 않습니다.

`weights`에 값을 입력하지 않을 경우 동일비중 포트폴리오를 구성하며, 포트폴리오 리밸런싱은 하지 않습니다.

Table 12.1: Return.portfolio() 함수 내 인자 설명

인자	내용
R	각 자산 수익률 데이터
weights	리밸런싱 시기의 자산 별 목표 비중. 미 입력시 동일비중 포트폴리오를 가정하여 백테스트가 이루어짐
wealth.index	포트폴리오 시작점이 1인 wealth index에 대한 생성여부이며, 디폴트는 FALSE로 설정
contribution	포트폴리오 내에서 자산 별 성과기여를 나타내는지에 대한 여부이며, 디폴트는 FALSE로 설정
geometric	포트폴리오 수익률 계산시 복리(기하)수익률 적용 여부이며, 디폴트는 TRUE로써 복리수익률을 계산
rebalance_onweight	값이 미입력 혹은 매번 같은 비중일 경우, 리밸런싱 주기를 선택할 수 있음
value	초기 포트폴리오 가치를 의미하며, 디폴트는 1
verbose	부가적인 결과를 표시할지에 대한 여부. 디폴트인 FALSE 를 입력할 경우 포트폴리오 수익률만이 시계열 형태로 계산되며, TRUE를 입력할 경우 수익률 외에 자산 별 성과기여, 비중, 성과 등이 리스트 형태로 계산됨

### 12.1.2 출력값 살펴보기

해당 함수는 verbose를 TRUE로 설정할 경우 다양한 결과값을 리스트 형태로 반환합니다.

Table 12.2: Return.portfolio() 함수 반환값

결과	내용
returns	포트폴리오 수익률
contribution	일자 별 개별 자산의 포트폴리오 수익률 기여도
BOP.Weight	일자 별 개별 자산의 포트폴리오 내 비중 (시작시점). 리밸런싱이 없을 시 직전 기간 EOP.Weight와 동일
EOP.Weight	일자 별 개별 자산의 포트폴리오 내 비중 (종료시점)
BOP.Value	일자 별 개별 자산의 가치 (시작시점). 리밸런싱이 없을 시 직전 기간 EOP.Value와 동일
EOP.Value	일자 별 개별 자산의 가치 (종료시점)

## 12.2 전통적인 60대 40 포트폴리오 백테스트

`Return.portfolio()` 함수의 가장 간단한 예제로써 전통적인 60대 40 포트폴리오를 백테스트 하도록 합니다. 해당 포트폴리오는 주식과 채권에 각각 60%와 40%를 투자하며, 특정 시점마다 해당 비중을 맞춰주기 위해 리밸런싱을 수행합니다. 매해 말 리밸런싱을 가정하는 예제를 살펴보도록 하겠습니다.

```
library(quantmod)
library(PerformanceAnalytics)
library(magrittr)

ticker = c('SPY', 'TLT')
getSymbols(ticker)

## [1] "SPY" "TLT"

prices = do.call(cbind,
                  lapply(ticker, function(x) Ad(get(x))))
rets = Return.calculate(prices) %>% na.omit()
```

글로벌 자산의 ETF 데이터 중 주식(S&P 500)과 채권(미국 장기채)에 해당하는 데이터를 다운로드 받은 후, 수익률을 계산하도록 합니다.

```
cor(rets)

##           SPY.Adjusted TLT.Adjusted
## SPY.Adjusted      1.0000     -0.4339
## TLT.Adjusted     -0.4339      1.0000
```

`cor()` 함수를 통해 두 자산간의 상관관계를 확인해보면 -0.43로써 매우 낮은 상관관계를 보이며, 강한 분산효과를 기대해볼 수 있습니다.

```
portfolio = Return.portfolio(R = rets,
                             weights = c(0.6, 0.4),
                             rebalance_on = 'years',
                             verbose = TRUE)
```

`Return.portfolio()` 함수를 이용하여 백테스트를 실행합니다.

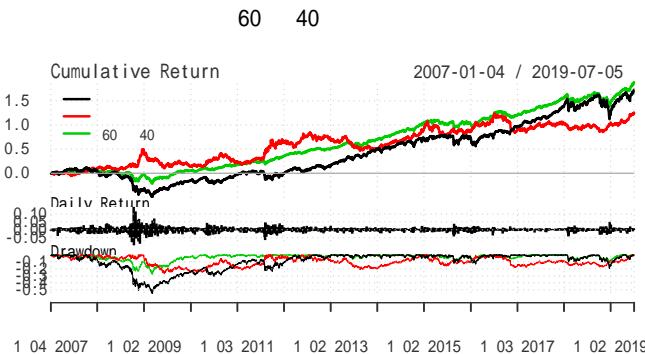
1. 자산의 수익률인 R에는 수익률 테이블인 rets를 입력합니다.
2. 리밸런싱 비중인 weights에는 60%와 40%를 의미하는 c(0.6, 0.4)를 입력합니다.
3. 리밸런싱 시기인 rebalance\_on에는 연간 리밸런싱에 해당하는 years를 입력합니다. 리밸런싱 주기는 이 외에도 quarters, months, weeks, days도 입력이 가능합니다.
4. 결과물을 리스트로 확인하기 위해 verbose를 TRUE로 설정합니다.

위 과정을 통해 주식과 채권 투자 비중을 매해 60%와 40%로 리밸런싱하는 포트폴리오의 백테스트가 실행됩니다. 표 12.3은 함수 내에서 포트폴리오의 수익률이 어떻게 계산되는지를 요약한 과정입니다.

Table 12.3: Return.portfolio() 계산 과정

	시작금액		시작합계		시작비중		수익률		종료금액		종료합계		종료비중		최종수익률
	1. 주식	2. 채권	3.1+2	4. 주식	5. 채권	6. 주식	7. 채권	8. 주식	9. 채권	10.8+9	11. 주식	12. 채권	13. 최종		
2017-12-26	1.603	0.940	2.543	0.630	0.370	-0.001	0.003	1.601	0.943	2.544	0.629	0.371	0.000		
2017-12-27	1.601	0.943	2.544	0.629	0.371	0.000	0.013	1.602	0.956	2.557	0.626	0.374	0.005		
2017-12-28	1.602	0.956	2.557	0.626	0.374	0.002	-0.001	1.605	0.955	2.560	0.627	0.373	0.001		
2017-12-29	1.605	0.955	2.560	0.627	0.373	-0.004	0.002	1.599	0.956	2.555	0.626	0.374	-0.002		
2018-01-02	1.533	1.022	2.555	0.600	0.400	0.007	-0.011	1.544	1.011	2.555	0.604	0.396	0.000		
2018-01-03	1.544	1.011	2.555	0.604	0.396	0.006	0.005	1.554	1.016	2.570	0.605	0.395	0.006		
2018-01-04	1.554	1.016	2.570	0.605	0.395	0.004	0.000	1.560	1.016	2.576	0.606	0.394	0.002		



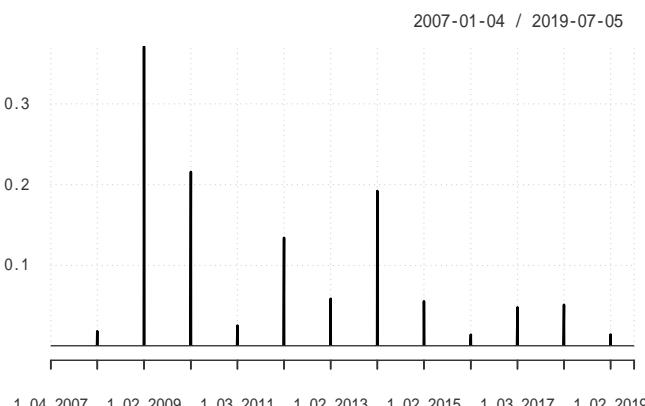


`PerformanceAnalytics` 패키지의 `charts.PerformanceSummary()` 함수는 기간별 수익률을 입력시 누적수익률, 일별수익률, 드로우다운 그래프를 자동으로 그려줍니다.

검은색 그래프는 주식 수익률(SPY), 붉은색 그래프는 채권 수익률(TLT), 초록색 그래프는 60대 40 포트폴리오 수익률을 의미합니다. 주식과 채권은 상반되는 움직임을 보이며 상승하며, 분산투자 포트폴리오는 각 개별 자산에 비해 훨씬 안정적인 수익률을 보입니다.

```
turnover = xts(
  rowSums(abs(portfolio$BOP.Weight -
    timeSeries::lag(portfolio$EOP.Weight)),
  na.rm = TRUE),
  order.by = index(portfolio$BOP.Weight))

chart.TimeSeries(turnover)
```



전일 종료시점의 비중인 EOP.Weight를 lag() 함수를 이용해 한 단계씩 내린 후, 시작시점의 비중인 BOP.Weight와의 차이의 절대값을 더해주면 해당 시점에서의 회전율이 계산됩니다. lag() 함수의 경우 dplyr 패키지에도 동일한 이름의 함수가 존재하므로, 충돌을 방지하기 위해 timeSeries 패키지의 함수임을 선언해줍니다. 이를 xts() 함수를 이용해 시계열 형태로 만들어준 뒤, chart.TimeSeries() 함수를 이용해 그래프로 나타내줍니다.

리밸런싱 시점에 해당하는 매매 첫 영업일에 회전율이 발생하며, 그렇지 않은 날은 매수 혹은 매도가 없으므로 회전율 역시 0을 기록합니다. 2008년에는 주식과 채권의 등락폭이 심하였으므로 이듬해엔 2009년 리밸런싱으로 인한 회전율이 심하지만, 이를 제외한 해는 회전율이 그리 심하지 않습니다.

## 12.3 시점 선택 전략 백테스트

이전 테스트가 리밸런싱 시점 별 비중이 60%와 40%로 고정되어 있었다면, 이번에는 시점 별 비중이 다른 형태의 예제를 살펴보도록 하겠습니다.

메브 파버 (Meb Faber)는 본인의 논문<sup>1</sup>을 통해, 시점 선택 (Market Timing) 전략을 사용할 경우 단순 매수 후 보유 대비 극심한 하락장에서 낙폭을 줄일 수 있으며, 이로 인해 위험 대비 수익률을 올릴 수 있다고 설명합니다. 논문에서 말하는 시점 선택의 투자 규칙은 다음과 같습니다.

주가 > 10개월 이동평균 → 매수

주가 < 10개월 이동평균 → 매도 및 현금 보유

해당 규칙을 미국 S&P 500에 적용하는 예제를 살펴보도록 하겠습니다. 현재 주식 가격이 과거 10개월 주식 가격의 단순 평균 대비 이상이면 매수, 그렇지 않으면 전량 매도 후 현금을 보유하는 전략이며, 리밸런싱은 매월 실행하도록 합니다.

```
library(quantmod)
library(PerformanceAnalytics)

symbols = c('SPY', 'SHY')
getSymbols(symbols, src = 'yahoo')

## [1] "SPY" "SHY"
```

---

<sup>1</sup>Faber, M. (2013). A quantitative approach to tactical asset allocation.

```
prices = do.call(cbind,
                  lapply(symbols, function(x) Ad(get(x))))
rets = na.omit(Return.calculate(prices))
```

먼저 주식과 현금에 해당하는 ETF 데이터를 다운로드 받습니다. 주식에 해당하는 ETF로는 S&P 500 수익률을 추종하는 SPY를 사용하며, 현금에 해당하는 ETF로는 미국 단기채 수익률을 추종하는 SHY를 사용합니다.

```
ep = endpoints(rets, on = 'months')
print(ep)
```

```
## [1] 0 19 38 60 80 102 123 144 167 186
## [11] 209 230 250 271 291 311 333 354 375 397
## [21] 418 439 462 481 503 523 542 564 585 605
## [31] 627 649 670 691 713 733 755 774 793 816
## [41] 837 857 879 900 922 943 964 985 1007 1027
## [51] 1046 1069 1089 1110 1132 1152 1175 1196 1217 1238
## [61] 1259 1279 1299 1321 1341 1363 1384 1405 1428 1447
## [71] 1468 1489 1509 1530 1549 1569 1591 1613 1633 1655
## [81] 1677 1697 1720 1740 1761 1782 1801 1822 1843 1864
## [91] 1885 1907 1928 1949 1972 1991 2013 2033 2052 2074
## [101] 2095 2115 2137 2159 2180 2201 2223 2243 2265 2284
## [111] 2304 2326 2347 2368 2390 2410 2433 2454 2475 2496
## [121] 2517 2537 2556 2579 2598 2620 2642 2662 2685 2705
## [131] 2727 2748 2768 2789 2808 2829 2850 2872 2893 2914
## [141] 2937 2956 2979 3000 3019 3040 3059 3080 3101 3123
## [151] 3143 3147
```

```
wts = list()
lookback = 10
```

먼저 xts 패키지의 `endpoints()` 함수를 이용해 매월 말일의 위치를 구합니다. 해당 함수는 `endpoints(x, on= 'months', k=1)`의 형태로 이루어지며 x는 시계열 데이터, on은 원하는 기간, k는 구간 길이를 의미합니다. 즉, 시계열 데이터에서 월말에 해당하는 부분의 위치를 반환하며, 매 월이 아닌 `weeks`, `quarters`, `years` 도 입력이 가능합니다.

결과적으로 ep에는 rets의 인덱스 중 매월 말일에 해당하는 부분의 위치가 구해집니다.

각 시점 별 비중이 입력될 wts를 공백의 리스트 형식으로 저장해주며, n개월 이동평균 값에 해당하는 lookback 변수는 10을 입력해 줍니다.

```
i = lookback + 1
sub_price = prices[ep[i-lookback] : ep[i] , 1]
```

```
head(sub_price, 3)
```

```
##           SPY.Adjusted
## 2007-01-03      109.5
## 2007-01-04      109.7
## 2007-01-05      108.9
```

```
tail(sub_price, 3)
```

```
##           SPY.Adjusted
## 2007-10-26      120.5
## 2007-10-29      120.9
## 2007-10-30      120.1
```

```
sma = mean(sub_price)
```

```
wt = rep(0, 2)
wt[1] = ifelse(last(sub_price) > sma, 1, 0)
wt[2] = 1 - wt[1]
```

```
wts[[i]] = xts(t(wt), order.by = index(rets[ep[i]]))
```

해당 전략은 for loop 구문을 통해, 매월 말 과거 10개월 이동평균을 구한 후 매수 혹은 매도를 선택한 후 비중을 계산합니다. 예시를 위해 첫번째 시점의 테스트 과정을 살펴보도록 하며, 과거 10개월에 해당하는 가격의 이동평균이 필요하므로 처음 시작은 i+1인 11부터 가능합니다.

1. 주가는 일별 데이터이며, 현재부터 과거 10개월에 해당하는 주가를 선택해야 합니다. 앞서 endpoints() 함수를 통해 주가에서 월말 기준점의 위치를 찾았으며, ep[i]는 현재시점 주가의 위치를, ep[i-lookback]는 현재부터 10개월 전 주가 위치를 의미합니다. 이를 통해 과거 10개월 간 주가를 찾은 후 sub\_price에 저장하도록 합니다.
2. mean()을 통해 10개월 주가의 평균을 계산합니다.

3. `rep(0, 2)`를 통해 비중이 들어갈 0 벡터를 생성해 줍니다.
4. `ifelse()` 구문을 통해 해당 전략의 조건에 맞는 비중을 계산합니다. `wt[1]`은 주식의 투자비중이며, 만일 현재 주가가 10개월 이동평균보다 클 경우 주식에 해당하는 비중은 1을, 그렇지 않을 경우 0을 부여합니다. `wt[2]`는 현금의 투자비중이며, 1에서 주식의 투자비중을 뺀 값을 입력합니다. 표 12.4는 해당 규칙이 요약되어 있습니다.
5. 위에서 만들어진 벡터를 `xts()`를 통해 시계열 형태로 바꾼 후, `wts`의 i번째 리스트에 저장해줍니다.

Table 12.4: 시점선택 조건 별 비중

자산	현재주가 > 10개월 이동평균	현재 주가 < 10개월 이동평균
주식비중	<code>wt[1] = 1</code>	<code>wt[1] = 0</code>
현금비중	<code>wt[2] = 0</code>	<code>wt[2] = 1</code>

위 과정을 for loop 구문을 통해 전체 기간에 적용한 백테스트는 다음과 같습니다.

```

ep = endpoints(rets, on = 'months')
wts = list()
lookback = 10

for (i in (lookback+1) : length(ep)) {
  sub_price = prices[ep[i-lookback] : ep[i] , 1]
  sma = mean(sub_price)
  wt = rep(0, 2)
  wt[1] = ifelse(last(sub_price) > sma, 1, 0)
  wt[2] = 1 - wt[1]

  wts[[i]] = xts(t(wt), order.by = index(rets[ep[i]]))
}

wts = do.call(rbind, wts)

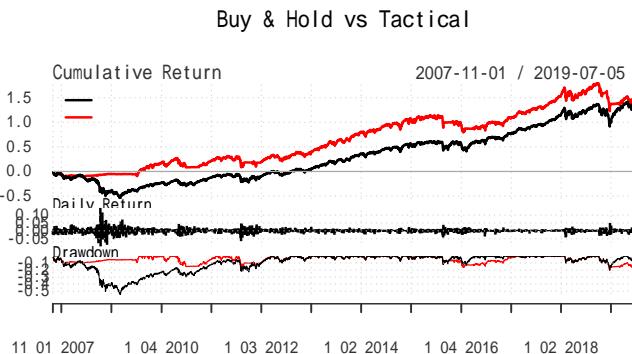
```

매월 말 과거 10개월 이동평균을 구한 후 현재 주가와 비교해 주식 혹은 현금 투자비중을 구한 후 `wts` 리스트에 저장합니다. 그 후, `do.call()` 함수를 통해 리스트를 테이블로 묶어주도록 합니다.

수익률 데이터와 비중 데이터가 구해졌으므로, `Return.portfolio()` 함수를 통해 포트폴리오의 수익률을 계산해주도록 합니다.

```
Tactical = Return.portfolio(rets, wts, verbose = TRUE)
portfolios = na.omit(cbind(rets[,1], Tactical$returns)) %>%
  setNames(c('매수 후 보유', '시점 선택 전략'))

charts.PerformanceSummary(portfolios,
                           main = "Buy & Hold vs Tactical")
```

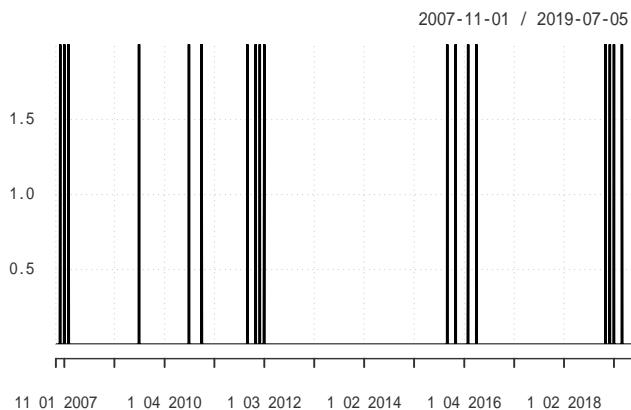


1. 수익률 데이터와 비중 데이터의 입력을 통해 백테스트를 실행합니다.
2. cbind() 함수를 통해 SPY 데이터와 포트폴리오 수익률을 합쳐주도록 합니다. 시점 선택 포트폴리오의 경우 lookback 기간인 초기 10개월에 대한 수익률이 없어 NA로 표시되므로, na.omit()을 통해 해당 부분을 제거합니다.
3. charts.PerformanceSummary() 함수를 통해 수익률을 그래프로 나타냅니다.

검은색 그래프가 S&P500에 매수 후 보유시 수익률, 붉은색 그래프는 시점 선택 전략을 적용한 수익률입니다. 2008년과 같은 하락장에서 낙폭이 훨씬 낮음이 확인됩니다.

```
turnover = xts(rowSums(abs(Tactical$BOP.Weight -
                            timeSeries::lag(Tactical$EOP.Weight)),
                        na.rm = TRUE),
               order.by = index(Tactical$BOP.Weight))

chart.TimeSeries(turnover)
```



해당 전략의 회전율을 확인해보면, 몇년 간 매매가 없는 경우도 존재합니다. 그러나 매매가 발생할 시 매수와 매도 포지션 양쪽의 매매로 인해 200%의 회전율이 발생하게 됩니다.

## 12.4 동적 자산배분 백테스트

백테스트의 마지막으로 기준에 배웠던 것들을 응용하여 동적 자산배분의 백테스트를 수행하도록 하겠습니다. 일반적인 자산배분이 주식과 채권, 대체자산에 투자비중을 사전에 정해놓고 약간의 비율만 수정하는 정적 자산배분인 반면, 동적 자산배분이란 투자비중에 대한 제한이 없이 동적으로 포트폴리오를 구성하는 방법입니다.

동적 자산배분을 이용한 포트폴리오는 다음과 같이 구성됩니다.

1. 글로벌 10개 자산 중, 과거 12개월 수익률이 높은 5개 자산 선택
2. 최소분산 포트폴리오를 구성하며, 개별 투자 비중은 최소 10%, 최대 30% 제약조건을 설정
3. 매월 리밸런싱 실시

```
library(quantmod)
library(PerformanceAnalytics)
library(RiskPortfolios)
library(tidyr)
library(dplyr)
library(ggplot2)

symbols = c('SPY', # 미국 주식
```

```

'IEV', # 유럽 주식
'EWJ', # 일본 주식
'EEM', # 이머징 주식
'TLT', # 미국 장기채
'IEF', # 미국 중기채
'IYR', # 미국 리츠
'RWX', # 글로벌 리츠
'GLD', # 금
'DBC' # 상품
)
getSymbols(symbols, src = 'yahoo')

## [1] "SPY" "IEV" "EWJ" "EEM" "TLT" "IEF" "IYR" "RWX"
## [9] "GLD" "DBC"

prices = do.call(cbind,
                  lapply(symbols, function(x) Ad(get(x)))) %>%
  setNames(symbols)

rets = Return.calculate(prices) %>% na.omit()

```

먼저 이전 장과 동일하게 글로벌 자산을 대표하는 ETF 데이터를 다운로드 받은 후, 수정주가의 수익률을 계산합니다.

```

ep = endpoints(rets, on = 'months')
wts = list()
lookback = 12
wt_zero = rep(0, 10) %>% setNames(colnames(rets))

```

백테스트에 사용되는 각종 값들을 사전에 정의합니다.

1. `endpoints()` 함수를 통해 매월 말일의 위치를 구합니다.
2. 매월의 투자 비중이 들어갈 빈 리스트를 `wts`에 설정합니다.
3. 수익률을 측정할 과거 n기간을 12개월로 설정합니다.
4. `rep()` 함수를 통해 비중이 들어갈 0으로 이루어진 벡터를 만들며, 이름을 설정합니다.

다음은 매월 말 투자 규칙에 따라 포트폴리오의 비중을 구하는 백테스트 과정입니다.

```

for (i in (lookback+1) : length(ep)) {
  sub_ret = rets[ep[i-lookback] : ep[i] , ]
  cum = Return.cumulative(sub_ret)

  K = rank(-cum) <= 5
  covmat = cov(sub_ret[, K])

  wt = wt_zero
  wt[K] = optimalPortfolio(covmat,
                            control = list(type = 'minvol',
                                           constraint = 'user',
                                           LB = rep(0.10, 5),
                                           UB = rep(0.30, 5)))

  wts[[i]] = xts(t(wt), order.by = index(rets[ep[i]]))
}

wts = do.call(rbind, wts)

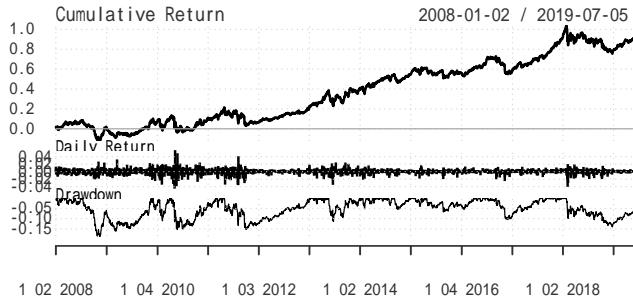
```

for loop 구문을 통해 매월 말 과거 12개월 수익률을 구한 후 비중을 계산하므로, 처음 시작은 i+1 인 13 부터 가능합니다.

1. ep[i]는 현재시점 수익률의 위치를, ep[i-lookback]는 현재부터 12개월 전 수익률의 위치를 의미합니다. 이를 통해 과거 12개월 간 수익률을 찾은 후 sub\_ret에 저장하도록 합니다.
2. Return.cumulative() 함수를 통해 해당 기간의 자산 별 누적수익률을 구합니다.
3. rank() 함수를 통해 수익률 상위 5개 자산을 선택하며, 내림차순으로 정렬해야 하므로 마이너스 부호를 붙여주도록 합니다.
4. cov() 함수를 통해 수익률 상위 5개 자산의 분산-공분산 행렬을 구하도록 합니다.
5. 임시로 비중이 저장될 wt 변수에 위에서 만든 0 벡터(wt\_zero)를 입력한 후, optimalPortfolio() 함수를 통해 최소분산 포트폴리오를 구성하는 해를 찾습니다. 개별 투자비중의 제한은 최소 10%, 최대 30%를 설정하며, 구해진 해를 wt의 K번째 값에 입력합니다.
6. 위에서 만들어진 벡터를 xts()를 통해 시계열 형태로 바꾼 후, wts의 i번째 리스트에 저장해줍니다.
7. for loop 구문이 끝난 후, do.call() 함수를 통해 투자 비중이 저장된 list 를 테이블 형태로 바꿔주도록 합니다.

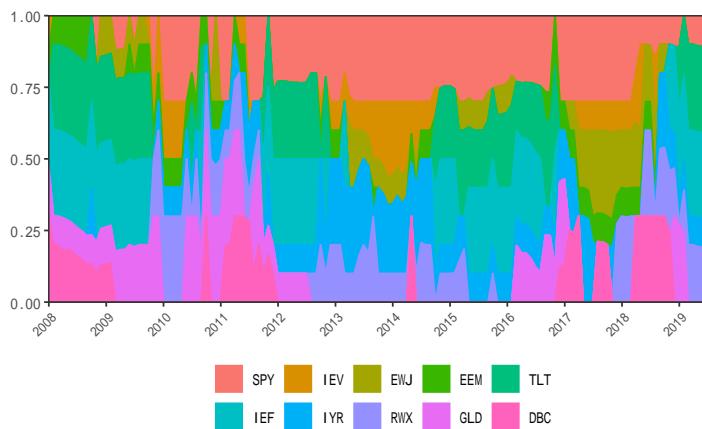
이를 통해 동적 자산배분의 투자 규칙에 맞는 매월 말 투자비중이 계산되었습니다.

```
GDAA = Return.portfolio(rets, wts, verbose = TRUE)
charts.PerformanceSummary(GDAA$returns, main = '동적자산배분')
```



수익률과 비중 데이터가 있으므로 `Return.portfolio()` 함수를 통해 백테스트 수익률을 계산할 수 있습니다. `charts.PerformanceSummary()` 함수를 통해 누적수익률을 확인하면, 해당 전략을 이용한 포트폴리오가 꾸준히 우상향 하는 모습을 보이게 됩니다.

```
wts %>% fortify.zoo() %>%
  gather(key, value, -Index) %>%
  mutate(Index = as.Date(Index)) %>%
  mutate(key = factor(key, levels = unique(key))) %>%
  ggplot(aes(x = Index, y = value)) +
  geom_area(aes(color = key, fill = key),
            position = 'stack') +
  xlab(NULL) + ylab(NULL) + theme_bw() +
  scale_x_date(date_breaks="years", date_labels="%Y",
               expand = c(0, 0)) +
  scale_y_continuous(expand = c(0, 0)) +
  theme(plot.title = element_text(hjust = 0.5,
                                  size = 12),
        legend.position = 'bottom',
        legend.title = element_blank(),
        axis.text.x = element_text(angle = 45,
                                   hjust = 1, size = 8),
        panel.grid.minor.x = element_blank()) +
  guides(color = guide_legend(byrow = TRUE))
```

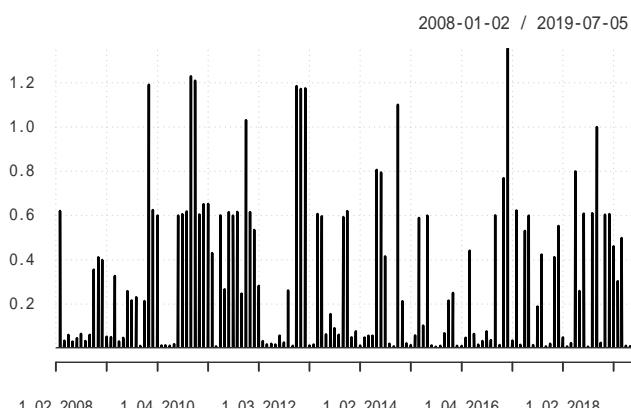


반면 자산 별 투자 비중의 변화가 많은것을 알 수 있습니다. 이는 수익률 상위 5 개에 해당하는 자산이 매월 말 바뀌는 원인과, 최소분산 포트폴리오를 구성하는 비중이 계속해서 바뀌는 원인 때문입니다.

회전율이 상대적으로 낮았던 기준 백테스트의 경우 매매비용, 세금, 기타비용 등을 고려하지 않아도 수익률에 크게 영향이 없지만, 회전율이 상대적으로 높은 전략의 경우 이러한 것들을 무시하지 않을 수 없습니다.

```
GDAA$turnover = xts(
  rowSums(abs(GDAA$BOP.Weight -
    timeSeries::lag(GDAA$EOP.Weight)),
  na.rm = TRUE),
  order.by = index(GDAA$BOP.Weight))

chart.TimeSeries(GDAA$turnover)
```

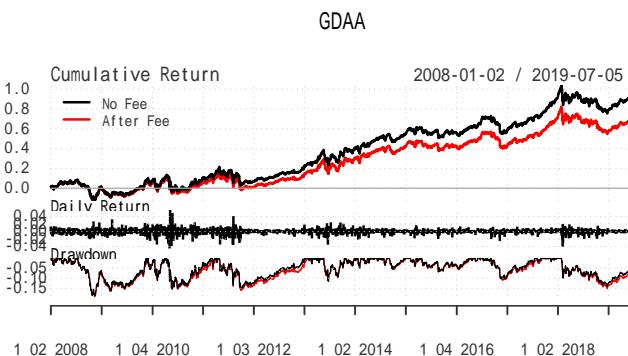


기준에 살펴본 방법으로 회전율을 계산할 경우, 매월 상당한 매매회전이 발생함이 확인됩니다.

```
fee = 0.0030
GDAA$net = GDAA$returns - GDAA$turnover*fee
```

매수 혹은 매도당 발생하는 세금, 수수료, 시장충격 등 총 비용을 0.3%로 가정합니다. 포트폴리오 수익률에서 회전율과 총비용의 곱을 빼면, 비용 후 포트폴리오의 순 수익률이 계산됩니다.

```
cbind(GDAA$returns, GDAA$net) %>%
  setNames(c('No Fee', 'After Fee')) %>%
  charts.PerformanceSummary(main = 'GDAA')
```



기존 비용을 고려하지 않은 포트폴리오(검은색)에 비해, 비용을 차감한 포트폴리오(붉은색)의 수익률이 시간이 지남에 따라 서서히 감소합니다. 이러한 차이는 비용이 클수록, 혹은 매매회전율이 높을수록 더욱 벌어지게 됩니다.



# CHAPTER 13

## 성과 및 위험 평가

백테스트를 통해 포트폴리오 수익률을 구했다면, 이를 바탕으로 각종 성과 및 위험을 평가해야 합니다. 아무리 성과가 좋은 전략이라도 위험이 너무 크다면 투자를 하기 부담스럽습니다. 또한 전략의 수익률이 지속적으로 감소하는 추세라면 경쟁이 치열해져 더이상 작동하지 않는 전략일 가능성도 있습니다.

본 장에서는 포트폴리오의 예시로 퀄리티 팩터를 종합적으로 고려한 QMJ(Quality Minus Junk) 팩터<sup>1</sup>의 수익률을 이용하겠습니다. QMJ 팩터란 우량성이 높은 종목들을 매수, 우량성이 낮은 종목들을 공매도한 전략을 지수의 형태로 나타낸 것입니다. 해당 팩터의 수익률을 통해 성과 및 위험을 평가해보고, 회귀분석을 통해 다른 팩터와의 관계도 살펴보도록 하겠습니다.

QMJ 팩터의 수익률은 저자들의 회사인 AQR Capital Management의 Datasets<sup>2</sup>에서 엑셀 파일을 다운로드 받은 후 가공할 수도 있습니다. 그러나 해당 작업을 매번 하는 것은 지나치게 번거로우므로, R 내에서 엑셀 파일을 다운로드 받은 후 가공하도록 하겠습니다.

```
library(dplyr)
library(readxl)
library(xts)
library(timetk)
```

<sup>1</sup>Asness, C. S., Frazzini, A., & Pedersen, L. H. (2019). Quality minus junk. Review of Accounting Studies, 24(1), 34-112.

<sup>2</sup><https://www.aqr.com/Insights/Datasets/Quality-Minus-Junk-Factors-Monthly>

```

url = paste0(
  'https://images.aqr.com/-/media/AQR/Documents/Insights/',
  'Data-Sets/Quality-Minus-Junk-Factors-Monthly.xlsx')

tf = tempfile(fileext = '.xlsx')
download.file(url, tf, mode = 'wb')

excel_sheets(tf)

## [1] "QMJ Factors"
## [2] "Definition"
## [3] "Data Sources"
## [4] "--> Additional Global Factors"
## [5] "MKT"
## [6] "SMB"
## [7] "HML FF"
## [8] "HML Devil"
## [9] "UMD"
## [10] "ME(t-1)"
## [11] "RF"
## [12] "Sources and Definitions"
## [13] "Disclosures"

```

1. 해당 데이터의 엑셀 url을 저장합니다.
2. tempfile() 함수 내 .xlsx 인자를 입력함으로써, 임시로 엑셀 파일을 만들도록 합니다.
3. download.file() 함수를 통해 url 파일을 tf 파일명에 저장하도록 하며, 엑셀의 경우 바이너리 파일이므로 wb 인자를 입력합니다.
4. readxl 패키지의 excel\_sheets() 함수를 통해 해당 엑셀의 시트명들을 확인합니다.

우리가 필요한 데이터는 수익률을 계산할 QMJ Factors, 회귀분석에 필요한 MKT, SMB, HML Devil, UMD, 무위험 이자율인 RF 시트의 데이터입니다.

```

df_QMJ = read_xlsx(tf, sheet = 'QMJ Factors', skip = 18) %>%
  select(DATE, Global)
df_MKT = read_xlsx(tf, sheet = 'MKT', skip = 18) %>%
  select(DATE, Global)
df_SMB = read_xlsx(tf, sheet = 'SMB', skip = 18) %>%

```

```

  select(DATE, Global)
df_HML_Devil = read_xlsx(tf, sheet = 'HML Devil',
                         skip = 18) %>%
  select(DATE, Global)
df_UMD = read_xlsx(tf, sheet = 'UMD', skip = 18) %>%
  select(DATE, Global)
df_RF = read_xlsx(tf, sheet = 'RF', skip = 18)

```

readxl 패키지의 `read_xlsx()` 함수를 통해 엑셀 데이터를 읽어올 수 있으며, 시트명을 정해줄 수도 있습니다. 또한 각 시트 내 18행 까지는 데이터를 설명하는 텍스트이므로, `skip` 인자를 통해 해당 부분은 읽어오지 않도록 합니다. 그 후 `select()` 함수를 통해 날짜에 해당하는 DATE와 수익률에 해당하는 Global 열만을 선택해 줍니다.

```

df = Reduce(function(x, y) inner_join(x, y, by = 'DATE'),
            list(df_QMJ, df_MKT, df_SMB,
                  df_HML_Devil, df_UMD, df_RF)) %>%
setNames(c('DATE', 'QMJ', 'MKT', 'SMB',
          'HML', 'UMD', 'RF')) %>%
na.omit() %>%
mutate(DATE = as.Date(DATE, "%m/%d/%Y"),
       R_excess = QMJ - RF,
       Mkt_excess = MKT - RF) %>%
tk_xts(date_var = DATE)

```

- `inner_join()` 함수를 통해 DATE를 기준으로 데이터를 묶어주어야 합니다. 해당 함수는 한 번에 두개 테이블만을 선택할 수 있으므로, `Reduce()` 함수를 통해 모든 데이터에 `inner_join()` 함수를 적용합니다.
- `setNames()` 함수를 통해 열이름을 입력합니다.
- 각 팩터별 시작시점이 다르므로 `na.omit()` 함수를 통해 NA 데이터를 삭제해줍니다.
- `mutate()` 함수를 통해 데이터를 변형해줍니다. DATE 열의 경우 mm/dd/yy의 문자열 형식이므로 이를 날짜 형식으로 변경해줍니다. QMJ 팩터 수익률에서 무위험 수익률을 차감하여 초과수익률을 구해주며, 시장 수익률에서 무위험 수익률을 차감하여 시장위험 프리미엄을 계산해줍니다.
- `tk_xts()` 함수를 이용해 티블 형태를 시계열 형태로 변경해주며, 인덱스는 DATE 열을 설정해줍니다. 형태 변경 후 해당 열은 자동으로 삭제됩니다.

위 과정을 통해 구한 데이터를 바탕으로 성과 및 위험을 평가하도록 하겠습니다.

## 13.1 결과 측정 지표

포트폴리오의 평가에서 가장 중요한 지표는 수익률과 위험입니다. 수익률은 누적수익률과 연율화 수익률, 연도별 수익률이 주요 지표이며, 위험의 경우 변동성과 낙폭이 주요 지표입니다.

이 외에도 승률, 롤링 윈도우 값 등 다양한 지표를 살펴보기도 합니다. 이러한 지표를 수식을 이용하여 직접 계산할 수도 있지만, `PerformanceAnalytics` 패키지에서 제공하는 다양한 함수들을 이용해 편하게 계산할 수 있습니다.

### 13.1.1 수익률 및 변동성

```
library(PerformanceAnalytics)
chart.CumReturns(df$QMJ)
```



먼저 `chart.CumReturns()` 함수를 이용해 QMJ 팩터의 누적수익률을 그래프로 나타내봅니다. 1989-07-31부터 2019-05-31까지 장기간동안 우상향 하는 모습을 보이고 있습니다.

```
prod((1+df$QMJ)) - 1 # 누적수익률
```

```
## [1] 4.082
```

```
mean(df$QMJ) * 12 # 연율화 수익률(산술)
```

```
## [1] 0.05709
```

```
(prod((1+df$QMJ)))^(12 / nrow(df$QMJ)) - 1 # 연율화 수익률(기하)
```

```
## [1] 0.05585
```

수익률 중 가장 많이보는 지표는 누적 수익률, 연율화 수익률(산술), 연율화 수익률(기하)입니다. 각 수익률을 구하는 법은 다음과 같습니다.

1. 누적 수익률:  $(1 + r_1) \times (1 + r_2) \times \dots \times (1 + r_n) = \{\prod_{i=1}^n (1 + r_i)\} - 1$ ,
2. 연율화 수익률(산술):  $\frac{(r_1+r_2+\dots+r_n)}{n} \times scale$
3. 연율화 수익률(기하):  $\{\prod_{i=1}^n (1 + r_i)\}^{scale/Days} - 1$

먼저 누적 수익률은 각 수익률에 1을 더한 값을 모두 곱한 후 1을 빼면 됩니다. 연율화 수익률(산술)의 경우 단순히 수익률의 평균을 구한 후, 연율화를 위한 조정값(scale)을 곱해주면 됩니다. 데이터가 일간일 경우 조정값은 252, 주간일 경우 52, 월간일 경우 12입니다. 현재 데이터는 월간 기준이므로 조정값은 12가 됩니다.

마지막으로 연율화 수익률(기하)의 경우 각 수익률에 1을 더한 값의 곱을 구한 후, 연율화를 위해 승수를 곱한 후 1을 빼주면 되며, Days는 시계열의 관측 기간입니다.

```
Return.cumulative(df$QMJ) # 누적수익률
```

```
## QMJ
## Cumulative Return 4.082
```

```
Return.annualized(df$QMJ, geometric = FALSE) # 연율화 수익률(산술)
```

```
## QMJ
## Annualized Return 0.05709
```

```
Return.annualized(df$QMJ) # 연율화 수익률(기하)
```

```
## QMJ
## Annualized Return 0.05585
```

수식에 맞게 값을 입력하여 계산할 수도 있지만, 함수를 이용할 경우 더욱 손쉽게 계산이 가능하며 실수를 할 가능성도 줄어들게 됩니다. 누적 수익률은 `Return.cumulative()` 함수를 통해, 연율화 수익률(산술)은 `Return.annualized()` 함수 내 `geometric` 인자를 `FALSE`로 선택해줌으로써, 연율화 수익률(기하)는 `Return.annualized()` 함수를 통해 계산이 가능합니다. 수식으로 계산한 값과 함수를 통해 계산한 값을 비교하면 동일함이 확인됩니다.

```
sd(df$QMJ) * sqrt(12) # 연율화 변동성
```

```
## [1] 0.07275
```

```
StdDev.annualized(df$QMJ) # 연율화 변동성
```

```
## QMJ
## Annualized Standard Deviation 0.07275
```

```
SharpeRatio.annualized(df$QMJ, Rf = df$RF, geometric = TRUE)
```

```
## QMJ
## Annualized Sharpe Ratio (Rf=2.8%) 0.366
```

위험으로 가장 많이 사용되는 지표는 변동성입니다. 연율화 변동성의 경우 `sd()` 함수를 통해 변동성을 계산한 후 조정값을 곱해 계산합니다. 그러나 `StdDev.annualized()` 함수를 사용하여 더욱 쉽게 계산할 수도 있습니다.

수익을 위험으로 나누어 위험 조정 수익률을 보는 지표가 샤프지수입니다. 해당 지수는  $\frac{R_i - R_f}{\sigma_i}$ 로 계산되며, 문자에는 포트폴리오 수익률에서 무위험 수익률을 차감한 값이, 분모에는 포트폴리오의 변동성이 오게 됩니다.

`SharpeRatio.annualized()` 함수를 이용할 경우 포트폴리오 수익률에서 무위험 수익률을 차감한 값을 연율화로 변경한 후, 연율화 변동성으로 나누어 샤프지수를 계산합니다. `geometric`을 `TRUE`로 설정할 경우 기하평균 기준 연율화 수익률을, `FALSE`로 설정할 경우 산술평균 기준 연율화 수익률을 계산합니다.

### 13.1.2 낙폭과 최대낙폭

먼저 낙폭(Drawdown)은 수익률이 하락한 후 반등하기 전까지 얼마나 하락하였는지를 나타냅니다. 최대낙폭(Maximum Drawdown)은 이러한 낙폭 중 가장 값이 큰 값으로써, 최고점에서 최저점까지 얼마나 손실을 보는지를 나타냅니다. 투자를 함에 있어 수익률이 하락하는 것은 어쩔 수 없지만, 최대낙폭이 지나치게 큰 전략에 투자하는 것은 매우 위험한 선택이 될 수 있습니다.

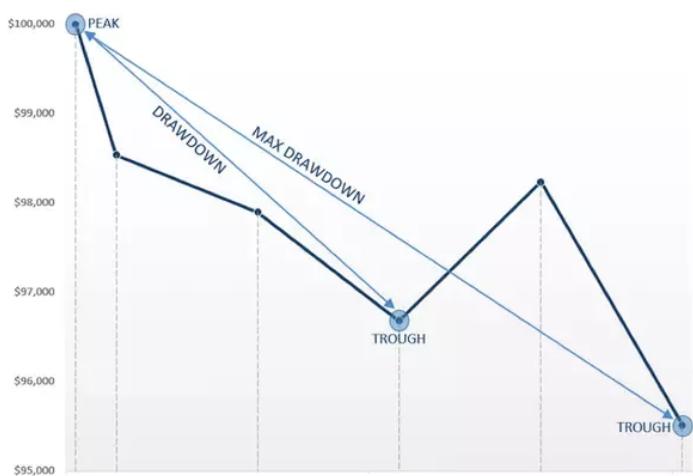


Figure 13.1: 낙폭과 최대낙폭

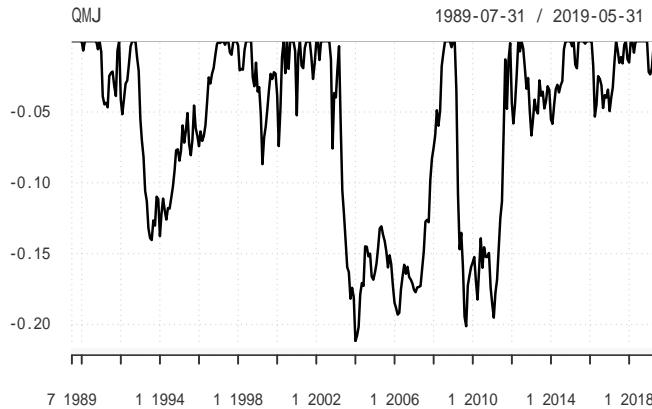
```
table.Drawdowns(df$QMJ)
```

##	From	Trough	To	Depth	Length
## 1	2002-10-31	2004-01-31	2008-08-31	-0.2118	71
## 2	2009-03-31	2009-09-30	2012-05-31	-0.2013	39
## 3	1992-11-30	1993-08-31	1997-01-31	-0.1404	51
## 4	1998-10-31	1999-04-30	2000-05-31	-0.0868	20
## 5	2012-08-31	2013-01-31	2014-10-31	-0.0666	27
## To Trough Recovery					
## 1	16	55			
## 2	7	32			
## 3	10	41			
## 4	7	13			
## 5	6	21			

```
maxDrawdown(df$QMJ)
```

```
## [1] 0.2118
```

```
chart.Drawdown(df$QMJ)
```



이러한 낙폭에 대한 지표들은 손으로 계산하기 번거롭지만, 패키지 내 함수를 사용한다면 매우 손쉽게 계산할 수 있습니다.

먼저 `table.Drawdowns()` 함수를 이용하면 역대 낙폭이 가장 심했던 순서대로 낙폭 정도, 하락 기간과 상승 기간, 원금 회복 기간 등을 테이블로 나타내줍니다. `maxDrawdown()` 함수는 포트폴리오의 최대 낙폭을 계산해주며, `chart.Drawdown()` 함수는 낙폭 만을 그래프로 그려줍니다.

```
CalmarRatio(df$QMJ)
```

```
## QMJ
## Calmar Ratio 0.2637
```

위험 조정 수익률 중 사용되는 지표 중 칼마 지수도 있습니다. 해당 지표는 연율화 수익률을 최대낙폭으로 나눈 값으로써, 특히나 안정적인 절대 수익률을 추구하는 헤지펀드에서 많이 참조하는 지표입니다.

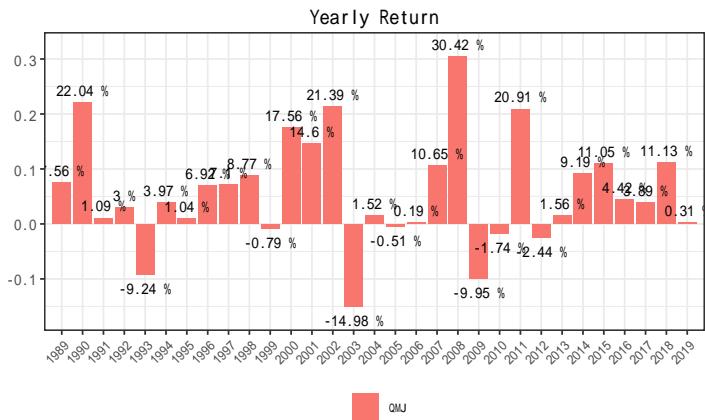
### 13.1.3 연도별 수익률



```

panel.grid.minor.x = element_blank() ) +
guides(fill = guide_legend(byrow = TRUE)) +
geom_text(aes(label = paste(round(value * 100, 2), "%")),
          vjust = ifelse(value >= 0, -0.5, 1.5)),
position = position_dodge(width = 1),
size = 3)

```



`apply.yearly()` 함수를 통해 계산한 연도별 수익률에 `ggplot()` 함수를 응용할 경우 막대그래프로 나타낼 수도 있으며, 시각화를 통해 포트폴리오의 수익률 추이가 더욱 쉽게 확인됩니다.

### 13.1.4 승률 및 롤링 윈도우 값

승률이란 포트폴리오가 벤치마크를 대비 높은 성과를 기록한 비율을 의미하며 다음과 같이 계산됩니다.

$$\frac{(\text{포트폴리오 수익률} > \text{벤치마크}) \text{ 일수}}{\text{전체 기간}}$$

벤치마크가 S&P 500 지수, KOSPI 200 지수처럼 구체적으로 존재하는 경우도 있지만, 절대수익을 추구하는 경우에는 이러한 벤치마크가 0 혹은 무위험 수익률이 되기도 합니다.

```
UpsideFrequency(df$QMJ, MAR = 0)
```

```
## [1] 0.5961
```

`UpsideFrequency()` 함수는 벤치마크 대비 승률을 계산해줍니다. MAR 인자는 0이 기본값으로 설정되어 있으며, 원하는 벤치마크가 있을 시 이를 입력해주면 됩니다. QMJ 팩터는 월간 기준 수익률이 플러스를 기록했던 비율이 59.61%입니다.

위에서 구한 각종 지표들은 투자자가 포트폴리오의 시작부터 현재까지 투자를 하였다는 전제하에 계산됩니다. 그러나 투자를 시작하는 시점은 사람마다 다르기에, 무작위 시점에 투자했을 시 향후 n개월 후 승률 혹은 연율화 수익률 등을 계산할 필요도 있습니다. 이러한 기법을 롤링 윈도우라 합니다.

```
roll_12 = df$QMJ %>% apply.monthly(., Return.cumulative) %>%
  rollapply(., 12, Return.annualized) %>% na.omit() %>%
  UpsideFrequency()

roll_24 = df$QMJ %>% apply.monthly(., Return.cumulative) %>%
  rollapply(., 24, Return.annualized) %>% na.omit() %>%
  UpsideFrequency()

roll_36 = df$QMJ %>% apply.monthly(., Return.cumulative) %>%
  rollapply(., 36, Return.annualized) %>% na.omit() %>%
  UpsideFrequency()

roll_win = cbind(roll_12, roll_24, roll_36)
print(roll_win)

##      roll_12 roll_24 roll_36
## [1,]  0.7644  0.7917  0.8673
```

롤링 윈도우 승률은 무작위 시점에 투자했을 시 미래 n기간 동안의 연율화 수익률을 구하고, 해당 값이 벤치마크 대비 수익이 높았던 비율을 계산합니다. 만일 12개월 롤링 윈도우 승률이 100%라면, 어떠한 시점에 투자하여도 12개월 후에는 언제나 벤치마크를 이겼음을 의미합니다. 반면 아무리 연율화 수익률이 높은 전략도 이러한 롤링 윈도우 승률이 지나치게 낮다면, 단순히 한번의 운으로 인해 수익률이 높아보이는 것처럼 보일 수 있습니다.

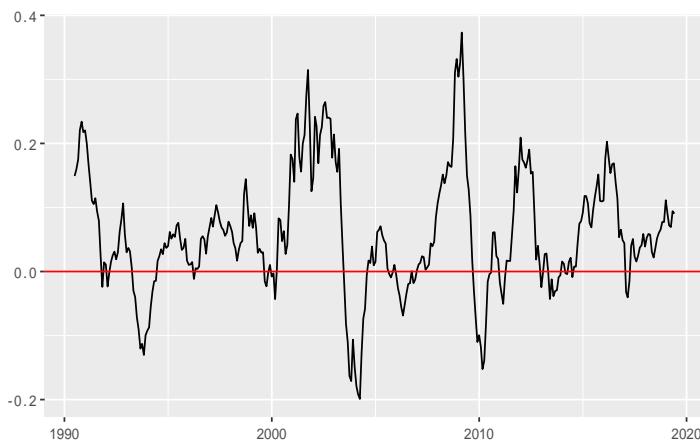
함수를 이용해 해당 값을 구하는 과정은 다음과 같습니다.

1. `apply.*()` 함수를 이용해 원하는 기간 수익률로 변경하며, 위 예제에서는 월간 수익률로 변경했습니다.
2. `rollapply()` 함수를 통해 원하는 기간의 롤링 윈도우 통계값을 구해줍니다. 각각 12개월, 24개월, 36개월 기간에 대해 연율화 수익률을 계산해줍니다.

3. 계산에 필요한 n기간 동안은 수익률이 없으므로 `na.omit()`을 통해 삭제 해줍니다.
4. `UpsideFrequency()` 함수를 통해 승률을 계산해 줍니다.

해당 과정을 통해 계산된 12개월, 24개월, 36개월 롤링 승률은 각각 76.44%, 79.17%, 86.73%이며, 투자 기간이 길어질수록 승률이 높아짐이 확인됩니다.

```
df$QMJ %>% apply.monthly(., Return.cumulative) %>%
  rollapply(., 12, Return.annualized) %>% na.omit() %>%
  fortify.zoo() %>%
  ggplot(aes(x = Index, y = QMJ)) +
  geom_line() +
  geom_hline(aes(yintercept = 0), color = 'red') +
  xlab(NULL) + ylab(NULL)
```



롤링 윈도우 연율화 수익률 역시 매우 중요한 지표입니다. 해당 값이 지속적으로 하락할 경우, 전략이 더 이상 동작하지 않는 것인지 혹은 가장 험난한 시기를 지났기에 인내심을 갖고 기다려야 할지 판단해야 합니다.

## 13.2 팩터 회귀분석 및 테이블로 나타내기

포트폴리오 수익률에 대한 성과 평가만큼 중요한 것이, 수익률이 어디에서 발생했는가에 대한 요인을 분석하는 것입니다. 베타를 통한 개별 주식과 주식시장과의 관계를 시작으로, 수익률을 설명하기 위한 여러 모형들이 개발되고 발표되었습니다. 그 중 일반적으로 많이 사용되는 모형은 기존의 CAPM에 사이즈

팩터(SMB), 밸류 팩터(HML)를 추가한 파마-프렌치의 쓰리팩터 모형, 그리고 쓰리팩터 모형에 모멘텀 팩터(UMD)를 추가한 카하트의 포팩터 모형입니다.

QMJ 팩터를 위 4개 팩터에 회귀분석한 결과를 토대로, 퀄리티 팩터의 수익률에 대한 요인 분석을 해보도록 하겠습니다.

```
reg = lm(R_excess ~ Mkt_excess + SMB + HML + UMD, data = df)
# summary(reg)
summary(reg)$coefficients
```

	Estimate	Std. Error	t value	Pr(> t )
## (Intercept)	0.002882	0.0007185	4.011	7.374e-05
## Mkt_excess	-0.275879	0.0167008	-16.519	7.565e-46
## SMB	-0.364831	0.0364337	-10.014	6.023e-21
## HML	-0.129366	0.0342693	-3.775	1.876e-04
## UMD	0.063467	0.0267821	2.370	1.834e-02

먼저 우리가 구한 데이터를 통해 다음과 같은 회귀분석을 실시합니다. 즉 QMJ 팩터의 초과수익률을 시장위험 프리미엄, 사이즈 팩터, 밸류 팩터, 모멘텀 팩터에 회귀분석 합니다.

$$QMJ - R_f = \beta_m \times [R_m - R_f] + \beta_{SMB} \times R_{SMB} + \beta_{HML} \times R_{HML} + \beta_{UMD} \times R_{UMD}$$

`lm()` 함수 내에서 `R_excess`는  $QMJ - R_f$ 와 동일하며, `Mkt_excess`는  $R_m - R_f$ 와 동일합니다. 베타의 절대값이 크다는 의미는 QMJ 팩터의 수익률이 해당 팩터와의 관계가 높다는 의미이며, 양수일 경우에는 양의 관계가, 음수일 경우에는 음의 관계가 높다는 의미입니다. 또한 t value 혹은 P value를 통해 관계가 얼마나 유의한지도 확인할 수 있습니다.

1. 시장 베타에 해당하는  $\beta_m$ 은 -0.276로 음수값을 보이며, 우량주의 경우 베타가 낮다고 볼 수 있습니다. 또한 t-value가 -16.519로 충분히 유의합니다.
2. 사이즈 베타에 해당하는  $\beta_{SMB}$ 는 -0.365로써 역시나 음수값을 보입니다. 즉 우량주는 주로 대형주, 비우량주는 주로 소형주일 가능성이 있습니다. t-value 역시 -10.014로 충분히 유의합니다.
3. 밸류 베타에 해당하는  $\beta_{HML}$ 은 -0.129로써 이 역시 음수값을 보입니다. 즉 퀄리티와 밸류 간의 관계에서 살펴본 것처럼, 우량주는 주로 밸류가 높은 경향이 있습니다. t-value 역시 -3.775로 유의합니다.
4. 모멘텀 베타에 해당하는  $\beta_{UMD}$ 는 0.063로 양의 관계가 있습니다. 즉 우량주는 주로 과거 수익률이 높았던 주식, 비우량주는 과거 수익률이 낮았던 주식인 경향이 있습니다. t-value는 2.37로 유의하다고 볼 수 있습니다.

5. 이러한 설명변수를 제외하고도 월간 초과수익률에 해당하는 계수값이 0.003이며, t-value는 4.011로 유의합니다. 즉, 퀘리티 팩터는 기존의 여러 팩터들로 설명되지 않는 새로운 팩터라고도 볼 수 있습니다.

```
library(broom)
tidy(reg)
```

```
## # A tibble: 5 x 5
##   term      estimate std.error statistic p.value
##   <chr>     <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept) 0.00288  0.000719    4.01 7.37e- 5
## 2 Mkt_excess  -0.276    0.0167   -16.5  7.56e-46
## 3 SMB         -0.365    0.0364   -10.0  6.02e-21
## 4 HML         -0.129    0.0343   -3.77  1.88e- 4
## 5 UMD         0.0635   0.0268    2.37  1.83e- 2
```

`broom()` 패키지의 `tidy()` 함수를 사용하면 분석 결과 중 계수에 해당하는 값만을 요약해서 볼 수 있습니다.

```
library(stargazer)
stargazer(reg, type = 'text', out = 'data/reg_table.html')
```

```
##
## =====
##                               Dependent variable:
##                               -----
##                               R_excess
## -----
## Mkt_excess                  -0.276***  

##                               (0.017)
## 
## SMB                         -0.365***  

##                               (0.036)
## 
## HML                         -0.129***  

##                               (0.034)
## 
## UMD                         0.063**  

##                               (0.027)
```

```
##  
## Constant 0.003***  
## (0.001)  
##  
## -----  
## Observations 359  
## R2 0.639  
## Adjusted R2 0.635  
## Residual Std. Error 0.013 (df = 354)  
## F Statistic 156.800*** (df = 4; 354)  
## ======  
## Note: *p<0.1; **p<0.05; ***p<0.01
```

**stargazer** 패키지를 사용하면, 회귀분석 결과를 논문에서 많이 사용되는 테이블 형식으로 손쉽게 출력과 저장을 할 수 있습니다. 테이블이 출력과 함께 data 폴더 내에 **reg\_table.html** 이름으로 html 파일 역시 저장이 됩니다.