

# SFERS DATA Seminar with Python: 1

Yitae Kwon

SFERS of SNU

2024-1

# Contents

## ① Web Crawling

- with Selenium
- with Requests

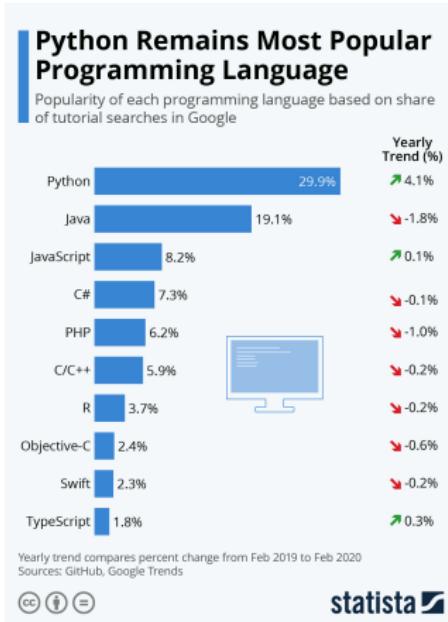
## ② Data Wrangling

- Introduction to Pandas
- Call data, Take subset, Handle missing data, Transpose, Change types
- Rename columns, Apply functions, Sort, Make new columns
- Group, Reshape, Combine, Lag&Difference, Save data

## ③ Regression

- What is regression?
- Simple linear regression
- How to interpret `smt.ols().fit().summary()`?

# Why Python?



파이썬은 현재 세계에서 가장 많이 쓰이는 프로그래밍 언어입니다. 가장 큰 장점 중 하나는 (근본 없지만) **간편한 문법**입니다. 또 하나의 장점은 **무료**입니다. 더불어 최근에는 대부분의 머신러닝 방법론들이 파이썬을 통해 개발되고 있어, 규모의 경제도 주요 원인 중 하나입니다. 단, Java/C/Julia에 비해서는 여전히 느리며, R과 STATA에서 제공하는 **통계프로그램적 요소**를 기대하시기는 어렵습니다.

# Web Crawling

## Web Crawling

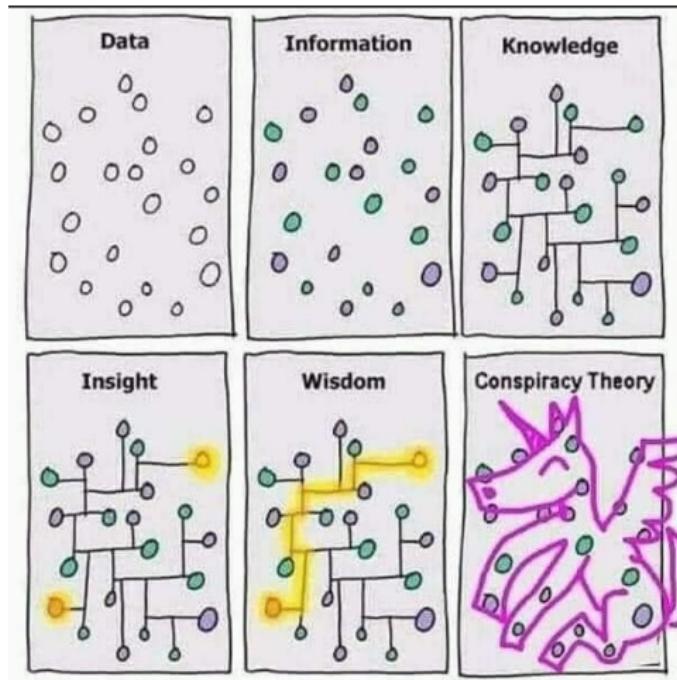
웹 크롤링이란

- 조직적이고
- 자동화된 방법으로

월드 와이드 웹을 탐색하는 행위이다.

검색엔진을 통해 특정 페이지에 접근할 수 있는 것도, 포털의 웹 크롤러가 이미 해당 페이지를 방문하고 해당 페이지의 정보를 보고하여 검색엔진에 추가하였기 때문입니다. 우리는 웹 크롤링 중에서도 데이터를 얻기 위한 **데이터 스크래핑(data scraping)**에 주목합니다.

# Data



우리는 정돈되지 않은 Data를 탐색해야 할 때가 많습니다.

# Web Crawlers



Scrapy



파이썬에서 데이터 스크래핑 툴로는 아래 네 개가 주로 사용됩니다.

- beautifulsoup
- selenium
- scrapy
- requests

우리는 그 중에서 selenium과 requests를 사용합니다.

selenium:

- 웹 브라우저를 통하기에 직관적임(동적 크롤링)
- 웹 브라우저 실행 과정에서 메모리/시간 문제
- 텍스트 형태의 자료를 다운로드받기 쉽다

requests:

- 쉽고, 간단하고, 빠르다(정적 크롤링)
- javascript 컨트롤이 불가능
- 파일 형태의 자료를 다운로드받기 쉽다

결론적으로 정리하면 이 두 개가 제일 쉽습니다.

# Be Careful!

크롤링은 법적/윤리적 문제를 고려하여 세심하게 수행하셔야 합니다.

- 크롤링을 하는 사이트의 서버에 많은 부하를 주지 말아 주세요. 짧은 시간에 많은 요청을 주는 것은 디도스 공격과 동일합니다. 적절한 시간 간격을 두어야 하고, 반복된 자료를 계속 요청하지 않는 게 좋습니다.
- 데이터 관련 규약이 있는 경우, 먼저 확인하고 이에 맞는 방식으로만 크롤링을 수행하셔야 합니다. 특히 학술적인 이용은 허용하지만 상업적으로는 이용을 허가하지 않는 경우도 많습니다.

# robots.txt

가장 편한 방법 중 하나는 robots.txt를 이용하는 것입니다. 원하는 사이트에 접속한 뒤, 링크 뒤에 'robots.txt'를 붙인 페이지에 접속해 보세요.

The screenshot shows a browser window with the URL 'econ.snu.ac.kr/robots.txt'. The page content is as follows:

```
User-agent: *
Allow: /
```

서울대학교 경제학부 홈페이지는 크롤링을 허용하고 있습니다. 따라서 경제학부 교수님들이 처음 내신 논문의 저널을 모두 확인한다거나 하는 작업은 웹 크롤링을 통해 가능하겠습니다.

- Allow: 크롤링을 허용하는 경로입니다. /만 덜렁 있다면, 이 사이트의 모든 하위 페이지에 대한 크롤링을 허용한다는 의미입니다. 혹은
- Disallow: 크롤링을 허용하지 않는 경로입니다. 마찬가지로 /만 있다면, 모든 페이지를 크롤링할 수 없습니다. 한편 공백이라면, 반대로 모든 페이지를 크롤링할 수 있습니다.

SFERS 홈페이지는 disallow: /로 제시하고 있습니다. 이는 우리가 SFERS를 크롤링할 경우 SFERS 측에서 이를 막겠다는 의미입니다(?). 이외에도 더 커스텀된 형식들이 있으나, 국내에서는 관련 논의가 미진하여 대개 일괄적으로 허용합니다.

# Required Packages

목표: 한국은행은 한국은행경제통계시스템 ECOS를 통하여 다양한 통계를 공시하고 있습니다. 어느날 박 교수님께서 국제수지를 확인해보라는 과제를 내 주셨습니다. 분석기간은 2014년 01월부터 2023년 12월까지입니다. 이 데이터를 어떻게 크롤링으로 다운로드받을 수 있을까요?

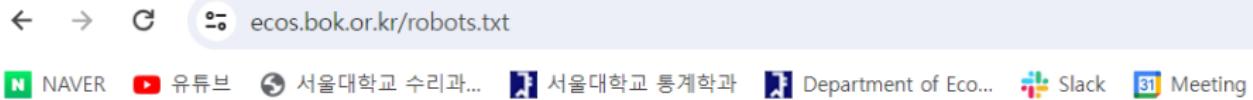
```
1 import selenium  
2 import requests  
3 import pandas as pd  
4 import numpy as np  
5
```

# ECOS에 접근하는 웹 드라이버 만들기

```
1  from selenium import webdriver
2  import chromedriver_binary
3  from selenium.webdriver.chrome.options import Options
4  chrome_options = Options()
5  chrome_options.add_argument("--disable-popup-blocking")
6  chrome_options.add_argument("--disable-extensions")
7  chrome_options.add_argument("--no-sandbox")
8  chrome_options.add_experimental_option("prefs", {
9      "download.default_directory": "./data"
10 })
11
```

크롬 웹드라이버는 크롬 버전마다 다른 chromedriver.exe 파일을 다운받을 수도 있고, chromedriver\_binary를 사용해도 되는데, 저는 편의상 후자를 이용했습니다.

# 크롤링 가능 여부 확인



```
# https://www.robotstxt.org/robotstxt.html
User-agent: *
Disallow:
```

확인해본 결과, Disallow: 0|네요.

# ECOS에 접근하는 웹 드라이버 만들기

```
1 parkdriver = webdriver.Chrome(options = chrome_options)
2 parkdriver.implicitly_wait(20)
3 parkdriver.get("https://ecos.bok.or.kr/")
4
```

ECOS 한국은행 경제통계시스템

HOME 로그인 ECOS 도움말 English

통계검색 | 테마별 통계 | 통계공표일정 | 통계정보 | 통계연구/간행물 | 고객지원

검색어를 입력해주세요.

주요검색어 #금리 #GDP #환율 #기준금리 #국고채

동화/금융 | 국민계정 | 한율/동관수출입 외환보유액 | 물가 | 기업경영분석 | 심리지수 | 지급결제 | 산업/가계/기타 | 해외/복한

위처럼 한국은행 경제통계시스템에 접근하셨다면 성공입니다.

# 검색창에 접근하기

```
1  from selenium.webdriver.common.by import By
2  from selenium.webdriver.common.keys import Keys
3  searchConsole = parkdriver.find_element(By.XPATH, '//*[@id="root"]/div[5]/div/div[2]/div[1]/div/div[1]/div/div[1]/div/input')
4  searchConsole.send_keys('권') # 참고로 한글 color가 좀 이상합니다ㅜ
5  searchConsole.send_keys(Keys.ENTER)
6
```

저 XPATH는 어떻게 찾으면 좋을까요?

# 객체의 XPATH에 접근하기

- ① F12 버튼을 누른다.



- ② '검사할 페이지 요소 선택' 클릭
- ③ 원하는 페이지 요소(우리는 검색창) 클릭
- ④ 오른쪽 음영 부분 우클릭 - 복사 - XPATH로 복사 클릭
- ⑤ 원하는 위치에 붙여넣기

The screenshot shows the browser's developer tools context menu open over an input element. The menu path is: 복사 > XPath 복사. Other options visible in the menu include 복사, 붙여넣기, 요소 숨기기, 강제 실행 상태, 중단 위치, outerHTML 복사, selector 복사, JS 경로 복사, 스타일 복사, XPath 복사, and 전체 XPath 복사. The 'XPath 복사' option is highlighted with a blue selection bar.

# 객체를 찾는 다양한 방법

기본 틀:

```
driver.find_element(By.???, ????)
```

```
driver.find_element(By.XPATH, '//button[text()="Some text"]')
driver.find_element(By.XPATH, '//button')
driver.find_element(By.ID, 'loginForm')
driver.find_element(By.LINK_TEXT, 'Continue')
driver.find_element(By.PARTIAL_LINK_TEXT, 'Conti')
driver.find_element(By.NAME, 'username')
driver.find_element(By.TAG_NAME, 'h1')
driver.find_element(By.CLASS_NAME, 'content')
driver.find_element(By.CSS_SELECTOR, 'p.content')
```

한편 복잡한 사이트에서는 거의 대부분 By.XPATH 혹은 By.CSS\_SELECTOR를 사용합니다.

# 자료에 접근하기

```
1 balance = parkdriver.find_element(By.XPATH, '//*[@@id="root"]/div[5]/div/div/div[4]/div[2]/div[2]/div/div/ul[1]/li/div[1]/div[2]/a/span[1]/mark/mark/mark/mark')
2 balance.click() # click 국제수지 button
3 rapid = parkdriver.find_element(By.XPATH, '//*[@@id="centerDiv"]/div/div/div[2]/div/div[2]/div/div[3]/div/button[1]')
4 rapid.click() # click rapid search button
5
```

이제 우리가 익숙한 직사각형 형태의 자료에 접근했습니다.

통계표	국제수지(BPM6)	단위	변환	2023/01	2023/02	2023/03	2023/04	2023/05	2023/06	2023/07	2023/08
2.5.1.1. 국제수지	경상수지	백만달러	원자료	-4,204.8	-1,325.9	-429.4	-1,372.7	2,300.8	6,177.8	4,113.9	5
	상품수지	백만달러	원자료	-7,349.7	-1,250.4	-1,181.0	612.5	1,883.6	3,913.3	4,427.5	5
	상품수출	백만달러	원자료	48,128.8	50,662.8	56,582.6	49,284.7	53,055.4	54,133.0	50,247.2	53
	상품수입(FOB)	백만달러	원자료	55,478.5	51,913.2	57,763.6	48,672.2	51,171.8	50,219.7	45,819.7	48
	일반상품수지	백만달러	원자료	-8,763.3	-2,755.3	-2,677.4	-715.7	380.7	2,658.5	3,212.1	3
	일반상품수출(FOB)	백만달러	원자료	46,632.5	49,116.7	55,034.8	47,906.9	51,496.1	52,820.3	48,948.0	52
	일반상품수입(FOB)	백만달러	원자료	55,395.8	51,872.0	57,712.2	48,622.6	51,115.4	50,161.8	45,735.9	48
	중계무역순수출	백만달러	원자료	1,429.3	1,428.8	1,296.3	1,047.2	1,298.5	1,221.8	1,231.2	1
	비화제여고스기	백만달러	인가율	-15.7	76.1	200.1	281.0	204.4	22.0	-15.9	

# 원하는 자료 형태로 커스텀하기(1): 클릭 기반

```
1 startmonth = parkdriver.find_element(By.XPATH, '//*[@@id="centerDiv"]/div/div[1]/div[2]/div[1]/div/div[1]/div[2]/div/div/div/div/div[1]/div/div')
2 parkdriver.execute_script("window.scrollTo(0, 0);")
3 startmonth.click() # click year-selection button
4 prevbutton = parkdriver.find_element(By.XPATH, '//*[@@id="centerDiv"]/div/div[1]/div[2]/div[1]/div/div[1]/div[2]/div/div/div/div/div[1]/span/div/div/div[1]/button[1]')
5 prevbutton.click() # click previous year button
6 button2014 = parkdriver.find_element(By.XPATH, "//*[contains(text(), '2014'))]")
7 button2014.click() # click 2014 button
8 buttonjan = parkdriver.find_element(By.CSS_SELECTOR, "[aria-label='2014년 1월 ']")
9 buttonjan.find_element(By.XPATH, "..").click()
10
```

기본적으로 단순 클릭은 객체 뒤에 .click()을 붙여 진행합니다. 만약 클릭하고자 하는 객체가 화면에 보이지 않는다면, execute\_script()를 통해 자바스크립트와 상호작용하여 스크롤할 수 있습니다.

## 원하는 자료 형태로 커스텀하기(2): 입력 기반

```
1  startyear = parkdriver.find_element(By.XPATH, '//*[@id="centerDiv"]/div/div[1]/div[2]/div[1]/div/div[1]/div[2]/div/div/div/div/div[2]/div/div/input[2]')
2  startyear.send_keys('2023')
3  startmonth_in = parkdriver.find_element(By.XPATH, '//*[@id="centerDiv"]/div/div[1]/div[2]/div[1]/div/div[1]/div[2]/div/div/div/div/div[2]/div/div/input[3]')
4  startmonth_in.send_keys('12')
5
6  searchbtn = parkdriver.find_element(By.XPATH, '//*[@id="centerDiv"]/div/div[1]/div[2]/div[1]/div/div[3]/div[3]/button')
7  searchbtn.click() # click search button
8
```

둘째 방법은 객체의 깊숙히까지 접근한 뒤 send\_keys를 이용하여 밸류를 바꿔주는 방식입니다.

# 다운로드 받기

```
1 parkdriver.execute_script("window.scrollTo(0, 10);")
2 download = parkdriver.find_element(By.XPATH, '//*[@@id="centerDiv"]/div/div[2]/div/div/div/div/div[2]/div/div[1]/div[1]/div/div/div/div[5]/button')
3 download.click() # attempt to download
4 csv = parkdriver.find_element(By.XPATH, '//*[@@id="centerDiv"]/div/div[2]/div/div/div/div/div[2]/div/div[2]/div/div/div[2]/div/div/div[3]/label')
5 csv.click() # file format to csv
6 downbutton = parkdriver.find_element(By.XPATH, '//*[@@id="centerDiv"]/div/div[2]/div/div/div/div/div[2]/div/div[2]/div/div/div[3]/button[2]')
7 downbutton.click() # click download button
8
```

한편, 앞에서 다운로드 경로를 현재 파일이 있는 경로에서 ./data라는 폴더에 저장하기로 하였으므로, 버튼 클릭 시 해당 폴더로 바로 저장됩니다.

# 파일 이름바꾸기

selenium은 아쉽게도 다운로드 과정 중에 파일 이름을 바꾸는 기능을 지원하지 않습니다. 따라서 shutil을 이용해 저희는 사후처리를 해주겠습니다.

```
1 import os, shutil  
2  
3 filepath = "./data"  
4 filename = max([filepath + '/' + f for f in os.listdir(filepath)  
]), key=os.path.getctime)  
5 shutil.move(os.path.join(filepath, filename), './data/  
국제수지_201401_202312.csv')  
6
```

이 메소드는 가장 최근에 폴더에 들어온 파일의 이름을 원하는 이름으로 바꿔 줍니다.

# 결과 확인

The screenshot shows a file explorer window with the following details:

- Path: SFERS > 2024-1 > 교육세션 > data
- File List:

이름	수정한 날짜	유형
AJ네트웍스_20140101_20231231.csv	2024-03-14 오후 12:18	Micro:
AK홀딩스_20140101_20231231.csv	2024-03-14 오후 12:18	Micro:
BGF_20140101_20231231.csv	2024-03-14 오후 12:19	Micro:
BGFE 태일_20140101_20231231.csv	2024-03-14 오후 12:18	Micro:
BNK금융지주_20140101_20231231.csv	2024-03-14 오후 12:19	Micro:
stock_list.csv	2024-02-12 오후 4:18	Micro:
week1.csv	2024-03-14 오후 7:17	Micro:
국제수지_201401_202312.csv	2024-03-13 오후 6:47	Micro:

원하는 이름으로 csv 파일이 잘 저장된 것을 볼 수 있네요.

# requests를 이용해 크롤링하기

목표: 어느날 안 교수님께서 KOSPI 시장에 상장된 953개 종목에 대하여 분석을 해보라는 과제를 내 주셨습니다. 이 반복적인 작업을 어떻게 더 쉽게 해볼 수 있을까요? 다행히도 목표가 되는 주식들의 리스트는 `stock_list.csv` 파일로 주셨습니다. 한국거래소 정보데이터시스템을 이용해보라는 힌트도 주셨습니다.

```
1  from pandas import DataFrame, Series
2  from io import BytesIO
3
4  stock_dataframe = pd.read_csv('./data/stock_list.csv', encoding
5      = 'euc-kr').iloc[0:5]
6  stock_name_list = stock_dataframe['한글 종목약명']
7  std_code_list = stock_dataframe['표준코드']
8  code_list = stock_dataframe['단축코드']
```

한편 거래소 홈페이지는 `robots.txt`를 제공하지 않는데, 이는 더욱 상세하고 자세한 데이터를 판매하고 있기 때문입니다. 다만 배포데이터라는 점에서 관행적으로... 많이들 크롤링합니다.

# requests를 이용해 크롤링하기

```
1 stock_dataframe  
2
```

	표준코드	단축코드	한글 종목명	한글 종목약명	영문 종목명	상장일	시장구분	증권구분	소속부	주식종류	액면가	상장주식수
0	KR7095570008	095570	AJ네트웍스보통주	AJ네트웍스	AJ Networks Co.,Ltd.	2015/08/21	KOSPI	주권	NaN	보통주	1000	45252759
1	KR7006840003	006840	AK홀딩스보통주	AK홀딩스	AK Holdings, Inc.	1999/08/11	KOSPI	주권	NaN	보통주	5000	13247561
2	KR7282330000	282330	BGF리테일보통주	BGF리테일	BGF Retail	2017/12/08	KOSPI	주권	NaN	보통주	1000	17283906
3	KR7027410000	027410	BGF보통주	BGF	BGF	2014/05/19	KOSPI	주권	NaN	보통주	1000	95716791
4	KR7138930003	138930	BNK금융지주보통주	BNK금융지주	BNK Financial Group Inc.	2011/03/30	KOSPI	주권	NaN	보통주	5000	322088438

# 특정 종목코드의 파일 불러오기 |

```
1 def foreign_data(stock_name, std_code, code, start_day, end_day
2 ) :
3     gen_otp_url = 'http://data.krx.co.kr/comm/fileDn/GenerateOTP/
4     generate.cmd'
5
6     gen_otp_data = {
7         'locale': 'ko_KR',
8         'searchType': 2,
9         'mktId': 'ALL',
10        'trdDd': '20240214',
11        'tboxisuCd_finder_stkisu0_3': code + '/' + stock_name,
12        'isuCd': std_code,
13        'isuCd2': '',
14        'codeNmisuCd_finder_stkisu0_1': stock_name,
15        'param1isuCd_finder_stkisu0_1': 'ALL',
16        'strtDd': start_day,
17        'endDd': end_day,
18        'share': '1',
19        'csvxls_isNo': 'false',
20        'name': 'fileDown',
21        'url': 'dbms/MDC/STAT/standard/MDCSTAT03702',
```

# 특정 종목코드의 파일 불러오기 //

```
20 }  
21  
22     headers = {'Referer' : 'http://data.krx.co.kr/contents/MDC/MDI/  
23     mdiLoader'}  
24     otp = requests.post(gen_otp_url, gen_otp_data, headers=headers)  
25     .text  
26  
27     down_url = 'http://data.krx.co.kr/comm/fileDn/download_csv/  
28     download.cmd'  
29     down_foreign = requests.post(down_url, {'code':otp}, headers=  
30     headers)  
31  
32     stock_foreign = pd.read_csv(BytesIO(down_foreign.content),  
33     encoding='cp949')  
34  
35     return stock_foreign
```

변수들은 대체 어떻게 결정해야 하는 걸까요?

# generate.cmd에 접근하기

- ① 한국거래소 정보데이터시스템에 접속해주세요.
- ② 상단 바에서 통계 - 기본 통계에 접속하고, 좌측 바에서 주식 - 세부안내 - 외국인보유량(개별종목)을 클릭해 접근해 주세요.
- ③ 조회구분을 전종목에서 개별종목으로 바꾸고, 조회 버튼을 클릭해 주세요.
- ④ F12버튼을 누르고, 상단 바에서 네트워크를 클릭한 다음 좌측 창을 킨 채로 다운로드 버튼을 클릭해 주세요.
- ⑤ 그러면 좌측 창에 파일 두 개가 나옵니다. 각각은 generate.cmd와 download.cmd입니다.
- ⑥ 그 중 generate.cmd를 클릭하고, 좌측 상단 탭에서 페이로드를 클릭하세요.

# generated.cmd에서 변수 얻기

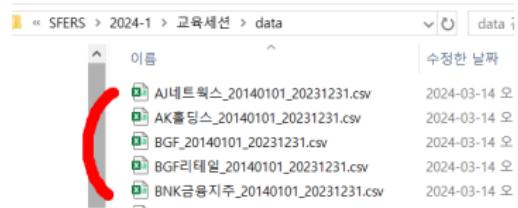
- gen\_otp\_url: generate.cmd 우클릭 - 복사 - URL 복사
- gen\_otp\_data: 페이지로드의 정보를 그대로 기입
- headers: Referer변수를 통하여 해당 웹 주소를 통해 접속한 것으로 취급해달라고 요청함
- down\_url: download.cmd 우클릭 - 복사 - URL 복사
- 나머지 변수나 함수는 그대로 복붙해서 사용하면 문제없음. 단, 사이트마다 다운로드받는 방식이 달라서 이것이 모든 사이트에 적용가능한 것은 아님. 해당 사이트에서 다운로드를 어떻게 받는지 확인하고 구글링해서 확인해봐야 함

# .csv파일로 저장하기

```
1 def foreign_csv(stock_name, std_code, code):
2     start_day = '20140101'
3     end_day = '20231231'
4
5     temp_data = foreign_data(stock_name, std_code, code, start_day,
6                               end_day)
6     temp_data.index.name = 'time'
7     temp_data.columns = ['time', 'finprice', 'contrast', 'return',
8                           'numOfStock', 'fore_vol', 'fore_ratio', 'fore_limit',
9                           'fore_limit_rate']
10
11    temp_data.to_csv("./data/" + stock_name + '_20140101_20231231.
12    csv', index = False)
13
14    import time
15
16    for i in range(5):
17        time.sleep(2)
18        foreign_csv(stock_name_list[i], std_code_list[i], code_list[i])
```

# 결과 확인

data 폴더에 5개 종목의 추가정보가 원하는 이름으로 모두 저장되어 있습니다.



이름	수정한 날짜
AJ네트웍스_20140101_20231231.csv	2024-03-14 오
AK홀딩스_20140101_20231231.csv	2024-03-14 오
BGF_20140101_20231231.csv	2024-03-14 오
BGF리테일_20140101_20231231.csv	2024-03-14 오
BNK금융지주_20140101_20231231.csv	2024-03-14 오

# 가장 추천하는 방법

가장 추천하는 방법은 바로 데이터 구매하기입니다. 여러분이 raw data를 직접 구하려는 게 아니라면, 이러한 데이터 정도는 이미 각종 데이터 전문 기업이나 국가기관이 적절한 비용에 판매하고 있습니다. 법적/저작권적 문제에서 벗어나 편하게 연구에 사용하려면, 그냥 그들을 통하는 것을 추천해 드립니다. 크롤링으로 데이터를 받아오는 것은 돈도 없고 지위도 없는 학부생들이나 하는 짓입니다...ㅠㅠ(반면 정말 raw data를 필요로 하는 연구를 해야한다면 - 인터넷 쇼핑몰 연구라든지 - 언젠가는 써먹지 않을까요?) 한편 이런 사이트들은 api로 데이터를 제공하는데, 이는 사이트들마다 방법이 달라서 제가 굳이 다루지는 않겠습니다. 앞으로는 예쁜 데이터만 가지고 분석하겠습니다.

- 한국거래소 데이터상품: <https://data.krx.co.kr/contents/MDC/DATA/datasale/index.cmd?viewNm=MDCDATA001>
- 한국은행 오픈 API 서비스: <https://ecos.bok.or.kr/api/#/>

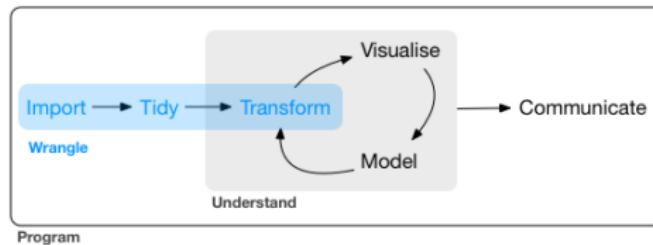
# Recommended Pages

- selenium 사용법과 명령어:  
<https://donghodazzi.tistory.com/306>
- selenium cheat sheet: <https://kihome15.tistory.com/10>
- requests cheat sheet: <https://proxiesapi.com/articles/python-requests-cheatsheet>
- requests 모듈 분석:  
<https://me2nuk.com/Python-requests-module-example/>
- 크롤링 방법 비교:  
<https://keyhyuk-kim.medium.com/python-%EC%9B%B9-%ED%81%AC%EB%A1%A4%EB%9F%AC-%EB%8F%84%EA%B5%AC-%EB%B9%84%EA%B5%90-%EB%B0%8F-%EC%82%AC%EC%9A%A9-%ED%9B%84%EA%B8%80-scrapy-vs-selenium-vs-requests-urllib-6483041ca1ba>

# Data Wrangling

# What is Data Wrangling?

data wrangling을 간단히 말하자면, 데이터를 시각화 및 모형화에 알맞은 형태로 변형하는 과정이라고 할 수 있습니다.



In a tidy data set:



Each **variable** is saved  
in its own **column**

&



Each **observation** is  
saved in its own **row**

# Why pandas?

pandas는 놀랍게도 **panel data**로부터 비롯되었습니다. 현재는 ML/DL 데이터 관리에 많이 쓰이는 면이 있긴 하지만, 그 장점 중 하나에 **시계열자료 저장의 용이함**이 있을 정도로 계량경제학에도 알맞는 라이브러리입니다.(위 키피디아에 따르면, 재무관리 분야에서 사용하기 위해 개발되었다고 하네요.) 더불어 비전문가들이 **쉽게 다룰 수 있고**, 간단한 분석에 필요한 **거의 모든 함수를 이미 제공한다는** 장점이 있습니다. 한편 **속도가 느리고**, 이 때문에 최근에는 torch.Tensor 같은 텐서 기반 연산을 제공하는 PyTorch 패키지가 더 인기를 끌고 있습니다.



# Why pandas?

pandas의 기본 데이터 형식은 Dataframe입니다.

The diagram illustrates the structure of a pandas DataFrame. It features a table with 7 rows and 10 columns. The columns are labeled: Name, Team, Number, Position, Age, Height, Weight, College, and Salary. The rows are indexed from 0 to 6. A purple box highlights the first row (index 0). Labels with arrows point to specific parts of the table:

- Column names**: Points to the top row of column headers.
- Columns axis=1**: Points to the vertical stack of column headers.
- Index label**: Points to the index value "0" in the first row.
- Index axis=0**: Points to the horizontal row of index values.
- Data**: Points to the numerical values in the table cells.
- Missing value**: Points to the cell containing "NaN" in the 3rd row, 5th column.

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	Nan	5000000.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0	6-8	235.0	LSU	1170960.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN	6-9	260.0	Ohio State	2569260.0
6	Evan Turner	Boston Celtics	11.0	SG	27.0	6-7	220.0	Ohio State	3425510.0

OG

# 파일 불러오기

```
1 balance = pd.read_csv('./data/국제수지_201401_202312.csv')
2 balance.shape # (289, 124) # 289 * 124 = 35836
3 balance.keys() # Index(['통계표', '국제수지(BPM6)', ...])
4 balance
5
```

통계표	국제수지(BPM6)	단위	변환																				
				2014/01	2014/02	2014/03	2014/04	2014/05	2014/06	...	2023/03	2023/04	2023/05	2023/06	2023/07	2023/08	2023/09	2023/10	2023/11	2023/12			
0	2.5.1.1. 국제수지	경상수지	백만 달러 원자료	1,837.7	4,812.5	6,182.8	6,875.5	9,148.3	7,768.8	...	-429.4	-1,372.7	2,300.8	6,177.8	4,113.9	5,412.7	6,072.7	7,437.8	3,890.7	7,414.6			
1	2.5.1.1. 국제수지	상품수지	백만 달러 원자료	3,674.4	5,930.7	7,450.3	9,833.7	8,560.9	6,088.3	...	-1,181.0	612.5	1,883.6	3,913.3	4,427.5	5,201.4	7,486.3	5,433.3	6,878.2	8,037.4			
2	2.5.1.1. 국제수지	상품수출	백만 달러 원자료	50,146.9	47,635.1	53,781.8	56,121.5	51,988.7	49,982.2	...	56,582.6	49,284.7	53,055.4	54,133.0	50,247.2	53,668.9	56,102.5	57,779.9	56,398.4	59,003.9			
3	2.5.1.1. 국제수지	상품수입(FOB)	백만 달러 원자료	46,472.5	41,704.4	46,331.5	46,287.8	43,427.8	43,893.9	...	57,763.6	48,672.2	51,171.8	50,219.7	45,819.7	48,467.5	48,616.2	52,346.6	49,520.2	50,966.5			
4	2.5.1.1. 국제수지	일반상품수지	백만 달러 원자료	2,242.9	4,672.4	5,950.6	8,113.3	6,940.1	4,694.7	...	-2,677.4	-715.7	380.7	2,658.5	3,212.1	3,772.5	6,360.4	3,818.3	5,323.7	6,240.8			
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	

# 필요없는 행/열을 잘라내기

사람의 숫자 기준 1, 3, 4열은 쓸모없는 자료입니다. 다르게 말하면, 2열과 5열 끝열까지만 필요합니다. 이때 우리는 함수 `balance.iloc[]`을 사용할 수 있습니다.

```
1     newbalance = balance.iloc[:, np.r_[1, 4:len(balance.keys())]]  
2  
3     newbalance
```

국제수 지 (BPM6)	2014/01	2014/02	2014/03	2014/04	2014/05	2014/06	2014/07	2014/08	2014/09	...	2023/03	2023/04	2023/05	2023/06	2023/07	2023/08	2023/09	2023/10	2023/11	2024/01	
0	감상수 지	1,837.7	4,812.5	6,182.8	6,875.5	9,148.3	7,768.8	6,198.8	6,793.9	7,418.7	—	-429.4	-1,372.7	2,300.8	6,177.8	4,113.9	5,412.7	6,072.7	7,437.8	3,890.7	
1	상품수 지	3,674.4	5,930.7	7,450.3	9,833.7	8,560.9	6,088.3	5,421.5	6,749.3	8,012.6	—	-1,181.0	612.5	1,883.6	3,913.3	4,427.5	5,201.4	7,486.3	5,433.3	6,878.2	
2	상품수 출	50,146.9	47,635.1	53,781.8	56,121.5	51,988.7	49,982.2	52,446.3	48,181.4	50,284.1	—	56,582.6	49,284.7	53,055.4	54,133.0	50,247.2	53,668.9	56,102.5	57,779.9	56,398.4	5
3	상품수 입(FOB)	46,472.5	41,704.4	46,331.5	46,287.8	43,427.8	43,893.9	47,024.8	41,432.1	42,271.5	—	57,763.6	48,672.2	51,171.8	50,219.7	45,819.7	48,467.5	48,616.2	52,346.6	49,520.2	5
4	일반상 품수지	2,242.9	4,672.4	5,950.6	8,113.3	6,940.1	4,694.7	4,824.0	6,180.0	7,508.5	—	-2,677.4	-715.7	380.7	2,658.5	3,212.1	3,772.5	6,360.4	3,818.3	5,323.7	
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	

한편 인덱싱에는 리스트를 다루는 데 최적화되어있는 라이브러리 NumPy를 사용할 수 있습니다.

# Why numpy?

NumPy, 혹은 numpy는 **Numerical Python**의 줄임말로, 과학계산을 위한 Python 라이브러리입니다. numpy가 제공하는 다차원 배열 객체인 ndarray는 파이썬에서 기초적으로 제공하는 list나 tuple 등보다 더 **효율적인 메모리 사용**, **빠른 연산 속도**, **다양한 계산함수 제공** 등의 장점을 가집니다. pandas에 비해서는 데이터 분석에 적합하지 않은 과학계산용 언어지만, 둘을 결합해서 사용할 경우 간단한 데이터 분석에 있어서는 단점이 없다고 보아도 무방할 정도의 범용성을 자랑합니다.



# 행/열 전환

일반적으로 자료는 개별적인 특성들이 column에, 각 copy(observations)가 row에 오는 것이 일반적입니다. 이 자료는 반대이기 때문에, .transpose()로 뒤집어 주겠습니다.

```
1 newbalance = newbalance.transpose()  
2 newbalance  
3
```

국제수지(BPMG)	0 경상수지	1 상품수지	2 상품수출	3 상품수입(FOB)	4 일반상품수지	5 일반상품수출(FOB)	6 일반상품수입(FOB)	7 증계무역수출	8 비화폐용금수지	9 비화폐용금수출(FOB)	... ...	279 중앙은행	280 일반정부	281 예금취급기관	282 기타부문
2014/01	1,837.7	3,674.4	50,146.9	46,472.5	2,242.9	48,610.8	46,367.9	1,482.2	-50.7	53.9	...	0.0	127.6	-103.6	-0.1
2014/02	4,812.5	5,930.7	47,635.1	41,704.4	4,672.4	46,276.0	41,603.6	1,264.3	-6.0	94.8	...	0.0	288.1	475.3	0.8
2014/03	6,182.8	7,450.3	53,781.8	46,331.5	5,950.6	52,164.2	46,213.6	1,510.0	-10.3	107.6	...	0.0	199.4	-113.6	0.1
2014/04	6,875.5	9,833.7	56,121.5	46,287.8	8,113.3	54,281.5	46,168.2	1,717.4	3.0	122.6	...	0.0	-118.1	978.2	-0.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

# 열 이름 바꾸기

현재 1행에 변수명이 있고, 열 이름은 숫자 인덱스로 되어 있습니다. 열의 이름을 바꾸어 보겠습니다. 열 이름에는 .columns로 접근합니다. 또, 숫자 대신 이름으로 행이나 열에 접근할 때에는 .loc[]을 이용합니다.

```
1 newbalance.keys() # RangeIndex(start=0, stop=289, step=1)
2
3 newbalance.columns = newbalance.loc['국제수지(BPM6)', :]
4 newbalance = newbalance.iloc[1:, :]
5 newbalance
```

국제수지(BPM6)	경상수지	상품수지	상품수출	상품수입(FOB)	일반상품수지	일반상품수출(FOB)	일반상품수입(FOB)	증계무역순수출	비화폐용금수지	비화폐용금수출(FOB)	...	중앙은행	일반정부	예금취급기관	기타부문
2014/01	1,837.7	3,674.4	50,146.9	46,472.5	2,242.9	48,610.8	46,367.9	1,482.2	-50.7	53.9	...	0.0	127.6	-103.6	-0.1
2014/02	4,812.5	5,930.7	47,635.1	41,704.4	4,672.4	46,276.0	41,603.6	1,264.3	-6.0	94.8	...	0.0	288.1	475.3	0.8
2014/03	6,182.8	7,450.3	53,781.8	46,331.5	5,950.6	52,164.2	46,213.6	1,510.0	-10.3	107.6	...	0.0	199.4	-113.6	0.1
2014/04	6,875.5	9,833.7	56,121.5	46,287.8	8,113.3	54,281.5	46,168.2	1,717.4	3.0	122.6	...	0.0	-118.1	978.2	-0.1
2014/05	9,148.3	8,560.9	51,988.7	43,427.8	6,940.1	50,271.9	43,331.8	1,615.7	5.1	101.1	...	0.0	47.0	-190.1	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

# Missing Data

우리 데이터에 결측치가 있는지 보겠습니다.

```
1 newbalance.isna().sum().sum() # 600
2 np.r_[0:newbalance.shape[1]][newbalance.isna().any(axis=0) ==
3 True] # array([182, 191, 199, 207, 210])
```

이 공백은 .csv 파일을 다운받으면서 데이터의 구조에 따라 나오는 자연스러운 공백입니다. 이처럼 자료에는 어떠한 이유로든 공백이 있을 수 있습니다. 이는 .isna() 함수로 캐치할 수 있습니다. 한편 어떠한 boolean list나 dataframe에 .any(axis = ?)을 붙이면, True가 하나라도 있으면 True를 리턴합니다. 이때 axis가 0이면 행, 1이면 열별입니다.

# Types of Missing Data

우리는 그냥 이 열을 다 삭제해 주면 되지만, 일반적인 경우에는 몇몇 원소만 없기 때문에 어떻게 해결해야 할지 고민이 됩니다. 결측치 발생의 매커니즘은 아래 세 가지로 나뉩니다.

- **MCAR(Missing Completely at Random)**: 결측치가 그 값이나 다른 변수들에 완전히 상관없이 무작위로 생성된 경우입니다. 실수, 전산오류 등에 의해 형성됩니다. ⇒ 일반적으로 그냥 삭제합니다.
- **MAR(Missing at Random)**: 변수의 누락 여부가 다른 변수와 관련되지만, 해당 변수의 참값이 그 다른 변수와는 관련이 없는 경우를 의미합니다. 예를 들어 MBTI가 I인 경우 인간관계에 대한 질문에 무응답하려는 경향성이 있지만, 실제로는 인간관계와 MBTI에 상관이 없을 수 있습니다. ⇒ 평균, 중앙값 등으로 impute합니다.
- **MNAR(Missing not at Random)**: 변수의 값과 누락 여부가 관련이 있는 경우입니다. 흡연자의 건강상태 정보가 사망으로 인해 결측되는 경우가 대표적입니다. ⇒ 다른 변수들로 적당히 estimate해 넣습니다.

# Missing Data

결측치가 있는 열을 다 드랍하려면, axis를 1로 하여 .dropna()를 쓰면 됩니다. MCAR과 같은 경우는 axis를 0으로 해서 드랍하면 되겠죠?

```
1 # 1. fill with specific value
2 # newbalance = newbalance.fillna(1)
3
4 # 2. interpolate
5 # newbalance = newbalance.interpolate()
6
7 # 3. drop whole column or row
8 newbalance = newbalance.dropna(axis = 1)
9
10 newbalance.drop_duplicates()
11 newbalance
12
```

한편 혹시 중복된 행이 있는 경우(보통 데이터를 만지는 사람의 실수일 가능성이 높음) drop\_duplicates()로 드랍해주면 됩니다.

# dtype 변환

데이터프레임을 이루는 각 열의 변수 타입은 dtype이라고 부릅니다. 대부분 우리는 수치형 데이터를 다룰 것이기 때문에, int32나 float64와 같은 numeric 타입이 기대됩니다.

```
1     newbalance.dtypes  
2
```

```
국제수지(BPM6)  
경상수지          object  
상품수지          object  
상품수출          object  
상품수입(FOB)    object  
일반상품수지    object  
...  
비금융기업등    object  
기타지분(부채)  object  
특별인출권      object  
준비자산          object  
오차및누락      object  
Length: 284, dtype: object
```

# dtype 변환

한편 데이터프레임의 열별로 어떠한 함수를 적용하고 싶다면, `.apply()` 메소드를 사용합니다. 그래서 우리가 자연스럽게 생각하는 것은 `numeric`으로 바꾸는 `pd.to_numeric()`을 `apply`하는 것인데, 그렇다면 에러가 납니다.

```
1 newbalance.apply(pd.to_numeric)
2 # Unable to parse string "1,837.7"
3
```

컴퓨터에서 파싱(parsing)이라는 것은 우리가 가진 객체와 타입 정보를 가지고 컴퓨터가 읽는 행위를 의미하는데, 현재 쉼표 때문에 이를 숫자로 인식하지 못하고 있는 상황입니다.

## dtype 변환

한 방법은 `.replace()`함수를 이용하여 자료에 있는 모든 콤마를 공백으로 변경하는 방법입니다. `regex`는 정규표현식(**regular expression**)의 준말인데, 우리는 몰라도 괜찮습니다.

```
1 newbalance = newbalance.replace(',', '', regex = True).apply(pd  
2 .to_numeric)  
newbalance
```

국제수 자 (BPM6)	경상 수지	상품 수지	상품수 출	상품수 입 (FOB)	일반 상품 수지	일반상 품수출 (FOB)	일반상 품수입 (FOB)	중계 무역 순수 지	비화 폐통 금수 지	비화 폐통 금수 출 (FOB)	... ...	중 양 은행	일반 정부	예금 취급 기관	기타부 문	기 용
2014/01	1837.7	3674.4	50146.9	46472.5	2242.9	48610.8	46367.9	1482.2	-50.7	53.9	..	0.0	127.6	-103.6	-0.1	
2014/02	4812.5	5930.7	47635.1	41704.4	4672.4	46276.0	41603.6	1264.3	-6.0	94.8	..	0.0	288.1	475.3	0.8	
2014/03	6182.8	7450.3	53781.8	46331.5	5950.6	52164.2	46213.6	1510.0	-10.3	107.6	..	0.0	199.4	-113.6	0.1	
2014/04	6875.5	9833.7	56121.5	46287.8	8113.3	54281.5	46168.2	1717.4	3.0	122.6	..	0.0	-118.1	978.2	-0.1	
2014/05	9148.3	8560.9	51988.7	43427.8	6940.1	50271.9	43331.8	1615.7	5.1	101.1	..	0.0	47.0	-190.1	0.0	

# 열 이름에서 공백 없애기

열 이름이 무언가 이상합니다.

```
1 newbalance.columns[1] # '상품수지'?
2
```

우리는 '상품수지'를 기대했는데, 지금 앞에 공백 두 개가 더 붙어 있네요. 이는 `.rename()` 메소드와 `lambda` 연산을 통해 해결할 수 있습니다. 마찬가지로 람다 오퍼레이터는 저희 수준에는 너무 어려워서, 그러려니 해주시면 되겠습니다.

```
1 newbalance = newbalance.rename(columns=lambda x: x.strip())
2 newbalance.columns[1] # '상품수지'
3
```

# 다양한 subset 방법들

- `.loc[]/.iloc[]`: 각각 열/행의 이름/인덱스번호를 통해 서브셋을 얻습니다.
- `.head()`, `.tail()`: 인덱스 순서대로 앞 혹은 뒤의 몇 개를 얻습니다.
- `.nlargest(n, columns = '???)`, `.nsmallest(n, columns = '???)`: '???' 이름의 열을 기준으로 큰/작은 순서대로 n개를 얻습니다.
- `.sample(n/frac)`: n을 넣으면 그 개수만큼, frac을 넣으면 그 비율로 데이터를 추출합니다.
- `.filter(regex = '?')`: '?'을 표현한 열만 보여줍니다.
- `.at[], .iat[]`: 이름 혹은 인덱스번호를 통해 특정 셀의 원소를 얻습니다.

# 컬럼 이름으로 subset 만들기

국제수지 데이터 중 큰 규모의 것들만 가져오겠습니다.

```
1 newbalance = newbalance.loc[:, ['경상수지', '상품수지', '서비스수지',
2                               '본월소득수지', '이전소득수지', '자본수지', '자본이전',
3                               '비생산비금융자산', '금융계정',
3                               '직접투자', '증권투자', '파생금융상품(순자산)', '기타투자',
3                               '준비자산', '오차및누락']]  
newbalance
```

국제수지 (BPMG)	경상수지	상품수지	서비스수지	본월소득수지	이전소득수지	자본수지	자본이전	비생산비금융자산	금융계정	직접투자	증권투자	파생금융상품(순자산)	기타투자	준비자산	오차및누락
2014/01	1837.7	3674.4	-1720.8	219.7	-335.6	-3.0	0.0	-3.0	3148.2	-1420.3	4393.6	-177.3	-2554.5	2906.7	1313.5
2014/02	4812.5	5930.7	-527.8	-426.3	-164.1	3.5	-0.7	4.2	6462.7	1594.5	6429.0	-264.6	-3621.2	2325.0	1646.7
2014/03	6182.8	7450.3	-603.6	-343.6	-320.3	-2.6	-0.5	-2.1	4701.5	2182.6	1791.8	-562.7	-502.2	1792.0	-1478.7
2014/04	6875.5	9833.7	-213.8	-1846.1	-898.3	2.6	0.4	2.2	5667.8	2370.2	-1793.6	-890.2	5315.6	665.8	-1210.3
2014/05	9148.3	8560.9	276.4	699.8	-388.8	-4.3	-1.4	-2.9	8648.6	3504.2	2027.7	-546.6	-2304.4	5967.7	-495.4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

# 다양한 sorting/filtering 방법들

- .query('??'): '??' 조건을 만족하는 관측만 내놓습니다.
- .sort\_values('??', ascending = Bool): 특정 열을 기준으로 상승/하락하는 순서대로 정렬합니다.

그 다음 조금의 수정을 더 가하겠습니다.

```
1 newbalance = newbalance.drop(columns = ['오차및누락'])
2 newbalance = newbalance.rename(columns = {'파생금융상품(순자산')': '파생금융상품'})
3 newbalance.columns.name = '국제수지(오차무시)'
4
```

# 밸류 바꾸기

현재 금융계정 부분이 전부 순투자금액으로 저장되어 있어서, 국제수지 합이 0이 아닌 상황입니다. 저희는 자금 흐름에 조금 더 관심이 있기 때문에, 합이 0에 가까워질 수 있도록 금융계정 부분의 부호를 바꿔 주겠습니다.

```
1 newbalance.loc[:, ['금융계정', '직접투자', '증권투자', '파생금융상품', '기타투자',  
'준비자산']] = -newbalance.loc[:, ['금융계정', '직접투자', '증권투자',  
'파생금융상품', '기타투자', '준비자산']]
```

2

국제수지 (BPMG)	경상수지	상품수지	서비스수지	본월소득수지	이전소득수지	자본수지	자본이전	비생산비금융자산	금융계정	직접투자	증권투자	파생금융상품(순자산)	기타투자	준비자산	오차및누락
2014/01	1837.7	3674.4	-1720.8	219.7	-335.6	-3.0	0.0	-3.0	3148.2	-1420.3	4393.6	-177.3	-2554.5	2906.7	1313.5
2014/02	4812.5	5930.7	-527.8	-426.3	-164.1	3.5	-0.7	4.2	6462.7	1594.5	6429.0	-264.6	-3621.2	2325.0	1646.7
2014/03	6182.8	7450.3	-603.6	-343.6	-320.3	-2.6	-0.5	-2.1	4701.5	2182.6	1791.8	-562.7	-502.2	1792.0	-1478.7
2014/04	6875.5	9833.7	-213.8	-1846.1	-898.3	2.6	0.4	2.2	5667.8	2370.2	-1793.6	-890.2	5315.6	665.8	-1210.3
2014/05	9148.3	8560.9	276.4	699.8	-388.8	-4.3	-1.4	-2.9	8648.6	3504.2	2027.7	-546.6	-2304.4	5967.7	-495.4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

# 새로운 column 만들기

이번에는 앞서 저장한 BGF리테일 추가정보를 가져오겠습니다.

```
1 bgfretail = pd.read_csv('./data/BGF  
리테일_20140101_20231231.csv')  
2 bgfretail  
3
```

	time	finprice	contrast	return	numOfStock	fore_vol	fore_ratio	fore_limit	fore_limit_rate
0	2023/12/28	131300	2600	2.02	17283906	5314527	30.75	17283906	30.75
1	2023/12/27	128700	-6100	-4.53	17283906	5365880	31.05	17283906	31.05
2	2023/12/26	134800	2900	2.20	17283906	5352057	30.97	17283906	30.97
3	2023/12/22	131900	-600	-0.45	17283906	5368849	31.06	17283906	31.06
4	2023/12/21	132500	-1300	-0.97	17283906	5380859	31.13	17283906	31.13
...	...	...	...	...	...	...	...	...	...
1486	2017/12/14	234000	-1000	-0.43	17283906	6402211	37.04	17283906	37.04
1487	2017/12/13	235000	0	0.00	17283906	6424750	37.17	17283906	37.17
1488	2017/12/12	235000	-13500	-5.43	17283906	6415802	37.12	17283906	37.12
1489	2017/12/11	248500	54500	28.09	17283906	6297276	36.43	17283906	36.43
1490	2017/12/08	194000	44500	29.77	17283906	6293321	36.41	17283906	36.41

# 날짜 정렬 순서를 바꾸기

먼저 날짜가 날짜 타입인지 확인해보겠습니다.

```
1     bgfretail.dtypes  
2
```

```
time          object  
finprice      int64  
contrast       int64  
return        float64  
numOfStock    int64  
fore_vol      int64  
fore_ratio    float64  
fore_limit    int64  
fore_limit_rate float64  
dtype: object
```

나머지는 다 수치형으로 원하는대로 나오는데, 시간은 시간 타입이 아니라 object이네요.

# 날짜 정렬 순서를 바꾸기

.to\_datetime과 .sort\_values()를 이용해 성공적으로 소팅했습니다.

```
1     bgfretail['time'] = bgfretail['time'].apply(pd.to_datetime)
2     bgfretail = bgfretail.sort_values('time')
3     bgfretail
4
```

	time	finprice	contrast	return	numOfStock	fore_vol	fore_ratio	fore_limit	fore_limit_rate
1490	2017-12-08	194000	44500	29.77	17283906	6293321	36.41	17283906	36.41
1489	2017-12-11	248500	54500	28.09	17283906	6297276	36.43	17283906	36.43
1488	2017-12-12	235000	-13500	-5.43	17283906	6415802	37.12	17283906	37.12
1487	2017-12-13	235000	0	0.00	17283906	6424750	37.17	17283906	37.17
1486	2017-12-14	234000	-1000	-0.43	17283906	6402211	37.04	17283906	37.04
...	...	...	...	...	...	...	...	...	...

# 년-월을 만들기

데이터타입을 날짜로 바꾸면 좋은 점 중 하나는 일, 월, 년에 더욱 접근하기 쉬워진다는 점입니다. 한편 NumPy는 np.where()를 제공합니다.

```
1     bgfretail['month'] = bgfretail['time'].dt.month
2     bgfretail['year'] = bgfretail['time'].dt.year
3     bgfretail['yearmonth'] = bgfretail['year'].astype(str) + '/' +
4         np.where(bgfretail['month'] <= 9, '0', '') + bgfretail['month'].astype(str)
5     bgfretail
```

	time	finprice	contrast	return	numOfStock	fore_vol	fore_ratio	fore_limit	fore_limit_rate	month	year	yearmonth
1490	2017-12-08	194000	44500	29.77	17283906	6293321	36.41	17283906	36.41	12	2017	2017/12
1489	2017-12-11	248500	54500	28.09	17283906	6297276	36.43	17283906	36.43	12	2017	2017/12
1488	2017-12-12	235000	-13500	-5.43	17283906	6415802	37.12	17283906	37.12	12	2017	2017/12
1487	2017-12-13	235000	0	0.00	17283906	6424750	37.17	17283906	37.17	12	2017	2017/12
1486	2017-12-14	234000	-1000	-0.43	17283906	6402211	37.04	17283906	37.04	12	2017	2017/12
...	...	...	...	...	...	...	...	...	...	...	...	...

# 그룹별로 aggregation

.groupby()를 이용하여 특정 원소를 기반으로 그룹화한 뒤 .agg()로써 특정 컬럼에 속하는 모든 원소를 하나의 함수로써 서머라이즈하여 쓸 수 있게 됩니다.

```
1 newbgf = bgfretail.groupby('yearmonth').agg({'finprice': 'mean'  
2 , 'return': 'mean', 'fore_ratio': 'mean'})  
3 newbgf
```

yearmonth	finprice	return	fore_ratio
2017/12	225357.142857	2.955000	36.950714
2018/01	211545.454545	0.240000	37.307727
2018/02	194777.777778	-1.267222	37.368333
2018/03	162714.285714	-0.090000	33.677619
2018/04	178642.857143	0.636190	33.433810
..	..	..	..
2023/08	165031.818182	-0.300909	34.683182
2023/09	149836.842105	-0.535263	34.934737
2023/10	136473.684211	-0.061579	34.772632
2023/11	139359.090909	-0.047727	34.041818
2023/12	134515.789474	-0.154211	31.399474

# 데이터 합치기

먼저 BGF리테일의 시간 스케일에 맞추기 위하여, 2017년 12월부터 2023년 12월까지의 newbalance만 잘라서 이용하겠습니다.

```
1 newbgf = bgfretail.groupby('yearmonth').agg({'finprice': 'mean',
2 , 'return': 'mean', 'fore_ratio': 'mean'})
3 newbgf
```

국제수지 (오차무 시)	경상 수지	상품 수지	서비스 수지	본원소 득수지	이전소 득수지	자본 수지	자본 이전	비생 산비 금융 자산	금융계 정	직접투 자	증권투 자	파생금 융상품	기타투 자	준비자 산
2017/12	4486.5	7915.5	-3707.4	1123.7	-845.3	-31.1	-14.9	-16.2	-6524.9	82.8	-12072.0	1451.9	4776.6	-764.2
2018/01	2556.9	7338.0	-4654.4	1543.5	-1670.2	-6.1	2.3	-8.4	-4640.5	-885.9	-3516.5	1310.9	-104.9	-1444.1
2018/02	3226.7	4793.3	-2723.5	1553.1	-396.2	-13.3	-6.6	-6.7	-3635.3	-253.1	-9867.3	888.1	5756.7	-159.7
2018/03	5210.4	9216.5	-2314.0	-1008.4	-683.7	1.7	5.7	-4.0	-5341.4	-1923.4	-399.2	1497.1	-3328.1	-1187.8
2018/04	1490.4	9402.2	-1970.1	-5269.2	-672.5	47.7	-0.5	48.2	-37.2	-2189.0	-4070.4	749.3	8591.9	-3119.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2023/08	5412.7	5201.4	-1549.5	1879.0	-118.2	-26.5	3.0	-29.5	-6286.7	-2112.9	-4246.9	328.8	-1847.5	1591.8
2023/09	6072.7	7486.3	-3209.9	2180.4	-384.1	23.7	25.6	-1.9	-4370.2	-2071.5	-5131.1	-665.3	2262.2	1235.5
2023/10	7437.8	5433.3	-1279.8	3358.5	-74.2	44.0	45.9	-1.9	-8773.7	-171.3	-4400.9	-430.0	-3812.2	40.7
2023/11	3890.7	6878.2	-2210.9	-116.6	-660.0	-11.5	-7.2	-4.3	-1128.1	-2888.1	2203.0	229.6	-54.9	-617.7
2023/12	7414.6	8037.4	-2535.4	2459.5	-546.9	3.5	2.4	1.1	-5676.2	-4417.1	-210.2	-1267.0	1613.8	-1395.7

# 데이터 합치기

동일한 행 개수를 가지고 있는 두 데이터를 합치려면, `.merge()` 함수를 사용합니다. 이러한 과정을 SQL이나 R 등에서는 `join`이라고 합니다.

```
1 tmp = pd.merge(newbalance, newbgf, left_index= True,  
2 right_index= True)  
3 tmp
```

경상수 지	성률수 지	서비스수 지	본원소득수 지	이전소득수 지	자본수 지	자본이 전	비생산비금을 자산	금융계 정	직접투 자	증권투 자	파생금융상 품	기타투 자	준비자 산	finprice	return	fore_ratio	
2017/12	4486.5	7915.5	-3707.4	1123.7	-845.3	-31.1	-14.9	-16.2	-6524.9	82.8	-12072.0	1451.9	4776.6	-764.2	225357.142857	2.955000	36.950714
2018/01	2556.9	7338.0	-4654.4	1543.5	-1670.2	-6.1	2.3	-8.4	-4640.5	-885.9	-3516.5	1310.9	-104.9	-1444.1	211545.454545	0.240000	37.307727
2018/02	3226.7	4793.3	-2723.5	1553.1	-396.2	-13.3	-6.6	-6.7	-3635.3	-253.1	-9867.3	888.1	5756.7	-159.7	194777.777778	-1.267222	37.368333
2018/03	5210.4	9216.5	-2314.0	-1008.4	-683.7	1.7	5.7	-4.0	-5341.4	-1923.4	-399.2	1497.1	-3328.1	-1187.8	162714.285714	-0.090000	33.677619
2018/04	1490.4	9402.2	-1970.1	-5269.2	-672.5	47.7	-0.5	48.2	-37.2	-2189.0	-4070.4	749.3	8591.9	-3119.0	178642.857143	0.636190	33.433810
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
2023/08	5412.7	5201.4	-1549.5	1879.0	-1182	-26.5	3.0	-29.5	-6286.7	-2112.9	-4246.9	328.8	-1847.5	1591.8	165031.818182	-0.300090	34.683182
2023/09	6072.7	7486.3	-3209.9	2180.4	-384.1	23.7	25.6	-1.9	-4370.2	-2071.5	-5131.1	-665.3	2262.2	1235.5	149836.842105	-0.535263	34.934737
2023/10	7437.8	5433.3	-1279.8	3358.5	-74.2	44.0	45.9	-1.9	-8773.7	-171.3	-4400.9	-430.0	-3812.2	40.7	136473.684211	-0.061579	34.772632
2023/11	3890.7	6878.2	-2210.9	-116.6	-660.0	-11.5	-7.2	-4.3	-1128.1	-2888.1	2203.0	229.6	-54.9	-617.7	139359.090909	-0.047727	34.041818
2023/12	7414.6	8037.4	2535.4	2459.5	-546.9	3.5	2.4	1.1	-5676.2	-4417.1	-210.2	-1267.0	1613.8	-1395.7	134515.789474	-0.154211	31.399474

# 데이터 변형하기

우리의 연구과제 중 하나는, 국제수지 발표 결과가 외국인에게 어떠한 사인을 줌으로써, 주식시장에서 주식보유비율을 어떻게 바꿀지(즉, 한국 증권시장을 얼마나 더 매력적으로 보게 되는지) 확인하는 것입니다. 그렇다면 우리 데이터에서, 국제수지 발표 결과는 전월 정보를 써야 합니다. 즉 2017/12 정보를 삭제하고, 2018/01~2023/12에는 (당월 `fore_ratio` - 전월 `fore_ratio`)를 보유량 변화(반응변수)로, 전월 국제수지 정보를 설명변수로 쓰기로 합니다.

```
1     result = tmp.copy()
2     result.loc[:, '경상수지':'준비자산'] = result.loc[:, '경상수지':''
3     준비자산'].shift(1)
4     result.loc[:, 'fore_ratio'] = tmp.loc[:, 'fore_ratio'].diff()
```

# 데이터 변형하기

	경상수지 _lag	상품수지 _lag	서비스수지 _lag	본원소득수 지_lag	이전소득수 지_lag	자본수지 _lag	자본이전 _lag	비생산비금융 자산_lag	금융계정 _lag	직접투자 _lag	증권투자 _lag	파생금융상 품_lag	기타투자 _lag	준비자산 _lag	외국인보유비율 변화
2018/01	4486.5	7915.5	-3707.4	1123.7	-845.3	-31.1	-14.9	-16.2	-6524.9	82.8	-12072.0	1451.9	4776.6	-764.2	0.357013
2018/02	2556.9	7338.0	-4654.4	1543.5	-1670.2	-6.1	2.3	-8.4	-4640.5	-885.9	-3516.5	1310.9	-104.9	-1444.1	0.060606
2018/03	3226.7	4793.3	-2723.5	1553.1	-396.2	-13.3	-6.6	-6.7	-3635.3	-253.1	-9867.3	888.1	5756.7	-159.7	-3.690714
2018/04	5210.4	9216.5	-2314.0	-1008.4	-683.7	1.7	5.7	-4.0	-5341.4	-1923.4	-399.2	1497.1	-3328.1	-1187.8	-0.243810
2018/05	1490.4	9402.2	-1970.1	-5269.2	-672.5	47.7	-0.5	48.2	-37.2	-2189.0	-4070.4	749.3	8591.9	-3119.0	0.329690
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2023/08	4113.9	4427.5	-2572.1	3356.3	-1097.8	-32.1	-0.2	-31.9	-3677.0	-1036.9	-4274.2	293.6	-125.7	1466.2	0.638420
2023/09	5412.7	5201.4	-1549.5	1879.0	-118.2	-26.5	3.0	-29.5	-6286.7	-2112.9	-4246.9	328.8	-1847.5	1591.8	0.251555
2023/10	6072.7	7486.3	-3209.9	2180.4	-384.1	23.7	25.6	-1.9	-4370.2	-2071.5	-5131.1	-665.3	2262.2	1235.5	-0.162105
2023/11	7437.8	5433.3	-1279.8	3358.5	-74.2	44.0	45.9	-1.9	-8773.7	-171.3	-4400.9	-430.0	-3812.2	40.7	-0.730813
2023/12	3890.7	6878.2	-2210.9	-116.6	-660.0	-11.5	-7.2	-4.3	-1128.1	-2888.1	2203.0	229.6	-54.9	-617.7	-2.642344

```
1 # result.to_csv('./data/week1.csv') # 인코딩 문제
2 result.to_csv('./data/week1.csv', encoding = "utf-8-sig")
3
```



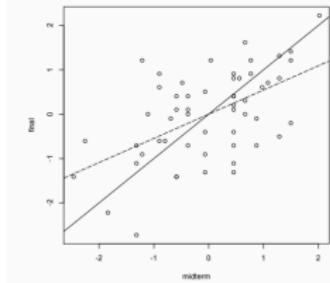
# 도움이 되는 링크들

- Pandas Cheat Sheet:  
[https://pandas.pydata.org/Pandas\\_Cheat\\_Sheet.pdf](https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf)
- NumPy Cheat Sheet: <https://www.datacamp.com/cheat-sheet/numpy-cheat-sheet-data-analysis-in-python>
- Pandas 동영상 강의:  
[https://www.dolearn.ai/sub/lecture/lecture\\_detail?idx=88](https://www.dolearn.ai/sub/lecture/lecture_detail?idx=88)
- NumPy 동영상 강의: <https://www.inflearn.com/course/%EB%8D% B0%EC%9D%B4%ED%84%B0-%EA%B3%BC%ED%95%99-%EB%84%98%ED%8C% 8C%EC%9D%B4-%EA%B8%B0%EB%B3%B8#curriculum>
- 류경석 교수님 Python Tutorial:  
[https://ernestryu.com/courses/deep\\_learning.html](https://ernestryu.com/courses/deep_learning.html)

# Regression

# What is Regression?

- Regression analysis is a statistical technique for investigating and modeling the **relation** between variables of interest. It is one of the most widely used statistical techniques.
- The word ‘regression’: Coined by Francis Galton who found that the height of the sons of tall fathers regressed towards the mean height of the population through successive generations.



# What is Regression?

Regression Problem은 아래와 같은 형식으로 구성됩니다.

$$(\text{Outcome}) = (\text{Explained by other variables}) + (\text{Noise})$$

이를 더욱 수식적으로 쓰면

$$Y = f(X) + \epsilon$$

처럼 쓸 수 있습니다.  $Y$ 를 **반응변수**(response variable),  $X$ 를 **설명변수**(explanatory variable) 혹은 **예측변수**(predictor)로 부릅니다.  $f$ 는 **regression function**입니다.

# What is Regression?

각 unit  $i$ 에 대해서는, 우리가

$$(X_i, Y_i)_{i=1,2,\dots,n}$$

을 얻게 됩니다. 우리는 일반적으로  $X_i$ 들은 각 unit에 대해 고정적이고 외생적으로 주어진다고 생각합니다. 따라서 noise, 혹은 error  $\epsilon$ 과 설명변수  $X$ 는 아래의 관계를 갖습니다.

$$X_i \perp\!\!\!\perp \epsilon_i$$

그렇다면 error가  $\mathbb{E}[\epsilon_i|X_i] = 0$ 이게 되므로,

$$\mathbb{E}[Y_i|X_i] = \mathbb{E}[f(X_i) + \epsilon_i|X_i] = f(X_i) + \mathbb{E}[\epsilon_i|X_i] = f(X_i)$$

입니다. 즉 우리는 주어진  $X_i$ 에 대하여  $Y_i$ 를 예측하는 좋은 방법 중 하나가  $f(X_i)$ 임을 알게 됩니다.

# Simple Linear Regression

단순선형회귀(simple linear regression)은 아래처럼 regression function  $f$ 를 설정합니다.

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

이때  $X_i$ 는 외생적으로 이미 관측된 값이라는 측면에서(즉,  $X_i$ 가 미리 관측된 후에  $\epsilon_i$ 를 통해  $Y_i$ 가 결정된다는 의미)  $x_i$ 로 씁니다. 혹은

$$\mathbb{E}[Y|x] = \beta_0 + \beta_1 x$$

처럼 쓸 수도 있습니다. 더하여, 우리는  $\epsilon_i$ 가 i.i.d 확률변수이며

$$\mathbb{E}[\epsilon] = 0, \quad \text{Var}(\epsilon) = \sigma^2$$

라고 가정합니다.

**Note:**  $Y_i$ 는 1차원 확률변수,  $x_i$ 는 1차원 외생변수,  $\epsilon_i$ 는 1차원 확률변수.

# Least Square Estimation

그 추정 중에는 잔차(예측오차)의 제곱합을 최소화하는 **최소제곱법**(Least Square Estimation; LSE)를 사용합니다.

$$(\hat{\beta}_0, \hat{\beta}_1) = \operatorname{argmin}_{\beta_0, \beta_1} \sum_{i=1}^n (Y_i - \beta_0 - \beta_1 x_i)^2$$

그 해는 이를  $\beta_0, \beta_1$ 으로 미분하여 얻은 **normal equation**

$$\begin{cases} \sum_{i=1}^n (Y_i - \beta_0 - \beta_1 x_i) = 0 \\ \sum_{i=1}^n x_i (Y_i - \beta_0 - \beta_1 x_i) \end{cases}$$

을 풀어서 사용합니다. 이는  $\beta_0, \beta_1$ 에 대한 연립 2원1차방정식이므로, 해를 쉽게 구할 수 있습니다.

# Notation

- $\bar{Y}$ :  $Y_i$ 들의 표본평균

$$\bar{Y} = \frac{1}{n} \sum_{i=1}^n Y_i = \mathbb{E}_n[Y]$$

- $\bar{x}$ :  $x_i$ 들의 표본평균

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

# Notation

- $S_{YY}$ :  $Y_i$ 들의 표본분산

$$S_{YY} = \frac{1}{n-1} \sum_{i=1}^n (Y_i - \bar{Y})^2 = \text{Var}_n(Y_i)$$

- $S_{xx}$ :  $x_i$ 들의 표본분산

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

- $S_{xY}$ :  $x_i, Y_i$ 의 표본공분산

$$S_{xY} = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(Y_i - \bar{Y})$$

# Estimators

- LSE 추정량  $\hat{\beta}_1$ :

$$\hat{\beta}_1 = \frac{S_{xy}}{S_{xx}}$$

- LSE 추정량  $\hat{\beta}_0$ :

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{x}$$

- 예측값  $\hat{Y}_i$ :

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

- 추정오차  $\hat{e}_i$ :

$$\hat{e}_i = Y_i - \hat{Y}_i = Y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i$$

# Property of Estimators

## Unbiasedness:

- $\mathbb{E}[\hat{\beta}_1] = \beta_1$ ,  $\mathbb{E}[\hat{\beta}_0] = \beta_0$ ,  $\mathbb{E}[\hat{Y}_i|x_i] = \beta_0 + \beta_1 x_i$ ,  $\mathbb{E}[\hat{e}_i|x_i] = 0$

## Variance of the Estimators:

- $\text{Var}(\hat{\beta}_1) = \frac{\sigma^2}{S_{xx}}$
- $\text{Var}(\hat{\beta}_0) = \sigma^2 \left( \frac{1}{n} + \frac{\bar{x}^2}{S_{xx}} \right)$
- $\text{Var}(\hat{Y}_i) = \sigma^2 \left( \frac{1}{n} + \frac{(x_i - \bar{x})^2}{S_{xx}} \right)$
- $\text{Var}(\hat{e}_i) = \sigma^2 \left( 1 + \frac{1}{n} + \frac{(x_i - \bar{x})^2}{S_{xx}} \right)$

Note: 이 성질은  $\sigma^2$ 의 분포가 무엇이거나와는 상관없다.

# Estimation of Variance $\sigma^2$

- 분산의 추정량  $\hat{\sigma}^2$ :

$$\hat{\sigma}^2 = \frac{1}{n-2} \sum_{i=1}^n \hat{e}_i^2 = \frac{1}{n-2} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

- Unbiasedness:

$$\mathbb{E}[\hat{\sigma}^2] = \sigma^2$$

# Simple Linear Regression in Python

```
1 import pandas as pd
2 import numpy as np
3 import statsmodels.formula.api as sm
4
5 data = pd.read_csv('./data/week1.csv', index_col=0)
6 regmodel =
7 sm.ols("외국인보유비율변화 ~ 경상수지_lag", data = data).fit()
8 print(regmodel.summary())
9
```

우리는 라이브러리 `statsmodels`를 주로 이용하고자 합니다. 기성 통계 프로그램 사용자들에게 가장 최적화된 통계 라이브러리 중 하나입니다.

- `formula`: "종속변수(반응변수) ~ 독립변수(설명변수)" 순
- `data`: 적합하고자 하는 `data`를 넣습니다.
- `.fit()`: `sm.ols()`를 통해 만들어진 모형을 적합시킵니다.
- `.summary()` 결과를 요약해 정리해줍니다.

# Interpretation of Regression Results

OLS Regression Results						
Dep. Variable:	외국인보유비율변화	R-squared:	0.000			
Model:	OLS	Adj. R-squared:	-0.014			
Method:	Least Squares	F-statistic:	0.0003212			
Date:	Sat, 16 Mar 2024	Prob (F-statistic):	0.986			
Time:	12:43:09	Log-Likelihood:	-70.805			
No. Observations:	72	AIC:	145.6			
Df Residuals:	70	BIC:	150.2			
Df Model:	1					
Covariance Type:	nonrobust					
coef	std err	t	P> t	[0.025	0.975]	
Intercept	-0.0753	0.128	-0.590	0.557	-0.330	0.179
경상수지_lag	-3.669e-07	2.05e-05	-0.018	0.986	-4.12e-05	4.05e-05
Omnibus:	76.239	Durbin-Watson:	1.372			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	741.546			
Skew:	-3.098	Prob(JB):	9.45e-162			
Kurtosis:	17.450	Cond. No.	1.03e+04			
Notes:						
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.						
[2] The condition number is large, 1.03e+04. This might indicate that there are strong multicollinearity or other numerical problems.						

# Estimators

	coef	std err	t	P> t	[0.025	0.975]
Intercept	-0.0753	0.128	-0.590	0.557	-0.330	0.179
경상수지_lag	-3.669e-07	2.05e-05	-0.018	0.986	-4.12e-05	4.05e-05

가장 중요한 결과는 둘째 박스에 모두 정리되어 있습니다. 우리는 이로부터

$$(\hat{\beta}_1, \hat{\beta}_0) = (-3.669 \times 10^{-7}, -0.0753)$$

임을 알 수 있습니다. 다르게 말하면, 우리가 추정한 모형은

$$\hat{Y}_i = -0.0753 - (3.699 \times 10^{-7})x_i$$

혹은

$$\widehat{\text{(외국인보유비율변화)}} = -0.0753 - (3.699 \times 10^{-7})(\text{전월 경상수지})$$

# Statistical Inference on Regression Coefficients

	coef	std err	t	P> t	[0.025	0.975]
Intercept	-0.0753	0.128	-0.590	0.557	-0.330	0.179
경상수지_lag	-3.669e-07	2.05e-05	-0.018	0.986	-4.12e-05	4.05e-05

std.err는 standard error의 줄임말로, 각 계수의 표준편차의 추정량

$$\widehat{\text{sd}}(\hat{\beta}_i) = \sqrt{\widehat{\text{Var}}(\hat{\beta}_i)}$$

을 의미합니다. ( $i = 1, 0$ )

$$\widehat{\text{sd}}(\hat{\beta}_1) = \sqrt{\frac{\hat{\sigma}^2}{S_{xx}}}, \quad \widehat{\text{sd}}(\hat{\beta}_0) = \sqrt{\hat{\sigma}^2 \left( \frac{1}{n} + \frac{\bar{x}^2}{S_{xx}} \right)}$$

# Statistical Inference on Regression Coefficients

어떠한 변수와 다른 변수가 상관관계가 존재하는지 확인하려면, 아래의 가설

$$H_0 : \beta_1 = 0 \quad \text{vs.} \quad H_1 : \beta_1 \neq 0$$

을 검정해야 합니다. 만약  $\epsilon_i \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$ 이라면,

$$\hat{\beta}_1 \sim \mathcal{N}(\beta_1, \sigma^2 / S_{xx})$$

$$\frac{(n-2)\hat{\sigma}^2}{\sigma^2} \sim \chi^2(n-2)$$

$$\hat{\beta}_1 \perp \hat{\sigma}^2$$

이므로, 귀무가설 하에서

$$T = \frac{\hat{\beta}_1 - 0}{\text{sd}(\hat{\beta}_1)} = \frac{\hat{\beta}_1 - 0}{\sqrt{\hat{\sigma}^2 / S_{xx}}} = \frac{\frac{\hat{\beta}_1 - \beta_1}{\sqrt{\sigma^2 / S_{xx}}}}{\sqrt{\frac{(n-2)\hat{\sigma}^2}{\sigma^2}} / (n-2)} \sim t(n-2)$$

# Statistical Inference on Regression Coefficients

	coef	std err	t	P> t	[0.025	0.975]
Intercept	-0.0753	0.128	-0.590	0.557	-0.330	0.179
경상수지_lag	-3.669e-07	2.05e-05	-0.018	0.986	-4.12e-05	4.05e-05

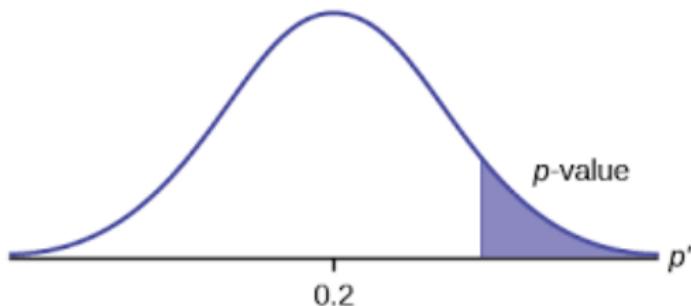
여기에서 t가 바로 이  $T$ 를 의미합니다. 한편  $\hat{\beta}_0$ 에 대해서는 상세한 식은 다르긴 하지만 어찌하였든

$$T = \frac{\hat{\beta}_0 - 0}{\hat{sd}(\hat{\beta}_0)} \sim t(n - 2)$$

가 성립합니다.

한편  $P>|t|$ 는 자유도  $n - 2$ 인  $t$ 분포 하에서 검정통계량  $T$ 보다 극단적인 값이 나올 확률을 의미합니다.

# Statistical Inference on Regression Coefficients



$t_\alpha(n - 2)$ 는 자유도가  $n - 2$ 인  $t$ 분포에서 오른쪽 꼬리의 확률을  $\alpha$ 로 만드는 점입니다. 따라서  $H_0$ 을 기각하는 방법에는 크게 두 가지가 있습니다.

- $t_{P>|t|}(n - 2) = T_{\text{obs}}$  이므로,  $P>|t| < \alpha$ 라면 기각합니다.
- 만약  $T_{\text{obs}} > t_\alpha(n - 2)$ 라면 기각합니다.

# Statistical Inference on Regression Coefficients

	coef	std err	t	P> t	[0.025	0.975]
Intercept	-0.0753	0.128	-0.590	0.557	-0.330	0.179
경상수지_lag	-3.669e-07	2.05e-05	-0.018	0.986	-4.12e-05	4.05e-05

[0.025, 0.975]는  $\beta_1$  혹은  $\beta_0$ 를 95%의 확률로 포함한다고 생각되는 양측 신뢰구간을 표시합니다.

$$CI_{1-\alpha} = \left( \hat{\beta}_i - t_{\alpha/2}(n-2) \cdot \hat{sd}(\hat{\beta}_i), \hat{\beta}_i + t_{\alpha/2}(n-2) \cdot \hat{sd}(\hat{\beta}_i) \right)$$

우리는 여기에서  $\alpha = 0.05$ 인 상황의 값을 보고받습니다.

**Note:** 신뢰구간을 뒤집는 방식으로 검정을 수행할 수도 있습니다. 만약  $H_0 : \beta_i = \beta$ 에서  $\beta$ 가 신뢰구간 안의 값이라면 귀무가설을 기각할 수 없고, 그 밖 이라면 귀무가설을 기각할 수 있게 됩니다. 우리의 경우에는 모든 신뢰구간이 0을 포함하므로, 절편과 기울기에 대한 검정 모두를 기각할 수 없습니다.

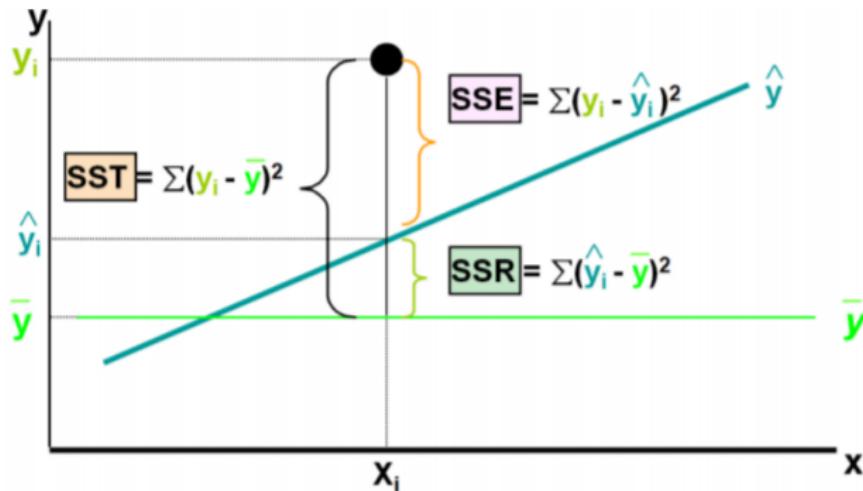
# Basic Results

OLS Regression Results				
Dep. Variable:	외국인보유비율변화	R-squared:	0.000	
Model:	OLS	Adj. R-squared:	-0.014	
Method:	Least Squares	F-statistic:	0.0003212	
Date:	Sat, 16 Mar 2024	Prob (F-statistic):	0.986	
Time:	12:43:09	Log-Likelihood:	-70.805	
No. Observations:	72	AIC:	145.6	
Df Residuals:	70	BIC:	150.2	
Df Model:	1			
Covariance Type:	nonrobust			

- Model, Method: 모형과 모형의 적합 방법
- No. Observations, Df Residuals, Df Model: 관측값수( $n$ ),  $t$ 분포 자유도( $n - 2$ ), 모형 자유도(1)
- 기타: 추후 설명

# SST, SSR, and SSE

- **SST**(Sum of Squares Total):  $\sum_{i=1}^n (Y_i - \bar{Y})^2$ , 전체 변동
- **SSR**(Sum of Squares Regression):  $\sum_{i=1}^n (\hat{Y}_i - \bar{Y})^2$ , 회귀로 설명되는 변동
- **SSE**(Sum of Squares Error):  $\sum_{i=1}^n (Y_i - \hat{Y}_i)^2$ , 회귀로 설명되지 않는 변동



# SST, SSR, and SSE

## Some facts

- $\text{SST} = S_{YY}$
- $\text{SST} = \text{SSR} + \text{SSE}$
- $\text{SSR} = \hat{\beta}_1 S_{xY}$
- $\hat{\sigma}^2 = \text{SSE}/(n - 2) = \text{MSE}$
- $\epsilon_i \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$ 이라면,

$$\frac{\text{SST}}{\sigma^2} \sim \chi^2(n - 1), \quad \frac{\text{SSR}}{\sigma^2} \sim \chi^2(1), \quad \frac{\text{SSE}}{\sigma^2} \sim \chi^2(n - 2)$$

- $\text{SSR} \perp\!\!\!\perp \text{SSE}$

결정계수  $R^2$ , 혹은 R-squared는 아래와 같이 정의됩니다.

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST} = \frac{S_{xY} S_{xY}}{S_{xx} S_{YY}} = r_{xY}^2$$

따라서 이는 전체 변동 중 회귀분석으로써 설명할 수 있는 변동의 비율을 의미합니다. 일반적으로  $R^2$ 이 클수록 회귀모형의 적합이 잘 이루어진 것으로 평가합니다. 그러나 그 절대적인 기준은 없습니다. 단, 귀무가설 하에서

$$R^2 \sim \text{Beta} \left( \frac{1}{2}, \frac{n-2}{2} \right)$$

이며  $\mathbb{E}[R^2] = \frac{1}{n-1}$  이므로, 이보다 작게 나오는 경우 사실상 거의 피팅이 안되었다고 보는 편이 옳습니다.

## $F$ statistics

피팅이 잘 되었는지 평가하는 방법에는  $F$ 통계량도 있습니다.

$$F = \frac{SSR/1}{SSE/(n-2)} \sim F(1, n-2)$$

이며 이 값이 클수록 회귀모형이 잘 적합되었다고 취급할 수 있습니다. 만약

$$F_{\text{obs}} > F_{\alpha}(1, n-2)$$

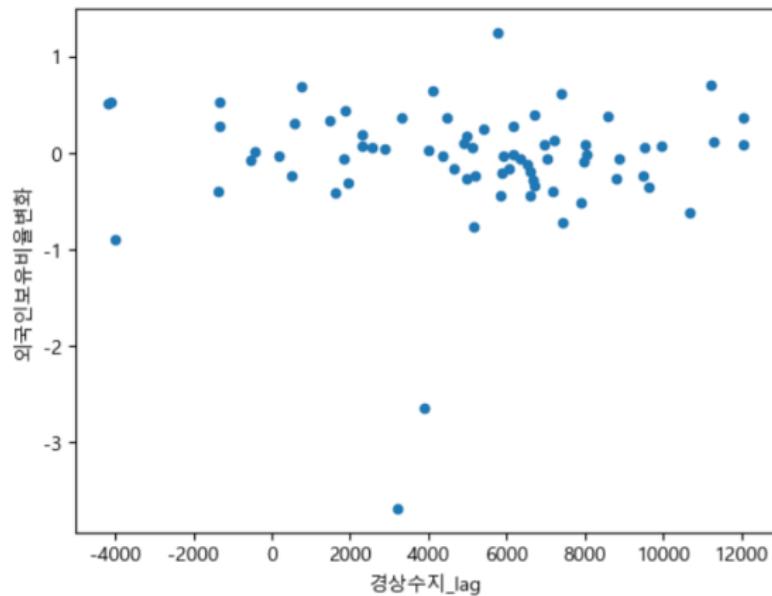
라면,  $H_0 : \beta_1 = 0$ 을 기각하게 됩니다.

**Note:** 단순회귀분석에서는  $t$ 분포를 이용한 검정과  $F$ 분포를 이용한 검정의 결과가 동일합니다.

**Note:** Prob (F-statistic)은  $P(F > F_{\text{obs}})$ 을 보고합니다.

# Visualization

결국 우리의 결과는 둘 사이에 큰 상관관계가 존재하지 않는다고 암시합니다. 이를 한 번 시각적으로 확인해 볼까요?



# 다른 값들에 접근하기(추정)

```
1 regmodel.params # estimates
2 regmodel.bse # standard errors
3 regmodel.predict() # predicted mean responses \hat{\beta}_0 + \
4 regmodel.fittedvalues # predict()와 유사하나 index가 더 잘 정렬되어 있음
5 regmodel.get_prediction().summary_frame() # info about mean
6 responses
7 regmodel.resid # residuals
```

	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
2018/01	-0.076929	0.077911	-0.232318	0.078460	-1.394670	1.240812
2018/02	-0.076221	0.091579	-0.258869	0.106426	-1.397454	1.245012
2018/03	-0.076467	0.085022	-0.246039	0.093105	-1.395956	1.243022
2018/04	-0.077195	0.077500	-0.231765	0.077375	-1.394840	1.240450
2018/05	-0.075830	0.104911	-0.285069	0.133409	-1.401000	1.249341
...	...	...	...	...	...	...
2023/08	-0.076792	0.079210	-0.234772	0.081188	-1.394842	1.241257
2023/09	-0.077269	0.077891	-0.232618	0.078080	-1.395005	1.240467
2023/10	-0.077511	0.080645	-0.238353	0.083330	-1.395906	1.240884
2023/11	-0.078012	0.092550	-0.262597	0.106573	-1.399514	1.243490
2023/12	-0.076711	0.080326	-0.236916	0.083495	-1.395028	1.241607

# 다른 값들에 접근하기(검정)

```
1 regmodel.f_test("경상수지_lag = 0") # test H_0 by F statistics
2 regmodel.f_pvalue # 0.9857530006659048
3 regmodel.fvalue # 0.00032115585063297324
4 regmodel.t_test("경상수지_lag = 0") # test H_0 by t statistics
5 regmodel.pvalues # p values
6 regmodel.tvalues # Intercept -0.590299
7 경상수지_lag -0.017921
```

# 다른 값들에 접근하기(모형 적합)

```
1 regmodel.nobs # 72.0
2 regmodel.centered_tss # 30.13271262071226, our SST
3 regmodel.mse_total # 0.42440440310862343, our SST/(nobs-1)
4 regmodel.ess # 0.00013824646507032412, our SSR(This can be
confusing. In ess, e is 'explained')
5 regmodel.mse_model # 0.00013824646507032412, our SSR/(1)
6 regmodel.ssr # 30.13257437424719, our SSE(This can be confusing
. In ssr, r is 'residual')
7 regmodel.mse_resid # 0.43046534820353133, our SSE/(nobs-2)
8 regmodel.rsquared # 4.587919674170493e-06, R^2
9
```

**Note:** 가끔 SSR과 SSE를 반대로 쓰는 책들이 있는데, 이 라이브러리도 마찬가지입니다... 잘 알아보고 사용하세요.

# ANOVA Table로 분석하기

```
1 import statsmodels.stats.api as sm_api  
2 sm_api.anova_lm(regmodel, typ = 1)  
3
```

	df	sum_sq	mean_sq	F	PR(>F)
경상수지_lag	1.0	0.000138	0.000138	0.000321	0.985753
Residual	70.0	30.132574	0.430465	NaN	NaN

- df: 해당하는 제곱합이 따르는 카이제곱분포의 자유도
- sum\_sq: SSR 혹은 SSE (SST도 제공하는 경우 있음)
- mean\_sq: MSR 혹은 MSE (MST도 제공하는 경우 있음)
- F, PR>F: 변수에 해당하는 F통계량, p값

# Statistical Inference on Variance $\sigma^2$

만약  $\epsilon_i$ 가  $\mathcal{N}(0, \sigma^2)$ 을 따르는 IID 확률변수라면,

$$\frac{(n-2)\hat{\sigma}^2}{\sigma^2} \sim \chi^2(n-2)$$

이므로,  $\sigma^2$ 의  $1 - \alpha$  신뢰구간은

$$\frac{(n-2)\hat{\sigma}^2}{\chi_{\alpha/2}^2(n-2)} \leq \sigma^2 \leq \frac{(n-2)\hat{\sigma}^2}{\chi_{1-\alpha/2}^2(n-2)}$$

입니다.

# 도움이 되는 링크들

- 박병욱 교수님 강의노트(regression):

<https://sites.google.com/view/theostat/lecture-notes/regression-analysis-and-lab-2020-spring?authuser=0>

- 회귀분석 기초(R):

[https://freshrimpsushi.github.io/ko/categories/%ED%86%  
B5%EA%B3%84%EC%A0%81%EB%B6%84%EC%84%9D/](https://freshrimpsushi.github.io/ko/categories/%ED%86%B5%EA%B3%84%EC%A0%81%EB%B6%84%EC%84%9D/)

- 회귀분석과 Python:

<https://www.statsmodels.org/stable/examples/index.html>

- scikit-learn을 이용한 회귀분석: [https://scikit-learn.org/stable/auto\\_examples/linear\\_model/plot\\_ols.html#sphx-glr-auto-examples-linear-model-plot-ols-py](https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html#sphx-glr-auto-examples-linear-model-plot-ols-py)

# Take-Home Messages

- 오늘 수업 템포가 빨랐던 것 알고 있습니다.
- 다음주에는 조금 더 느리면 느렸지 빠르지는 않을 예정입니다.
- 기타 질문은 구글링해보신 뒤 안 보이면 저에게 주세요.
- 오늘 느끼셨겠지만 2시간 내에 이걸 모두 이해하는 것은 불가능합니다. 과제를 해보시면서 직접 이해하려 노력해 보세요.
- 오타/오개념 발견하시면 언제나 말씀해 주세요.
- 이 페이지 참고하시면 항상 과제하시는 데 도움이 되실 것 같습니다.

https:

//www.utc.fr/~jlaforet/Suppl/python-cheatsheets.pdf

# Next Week

- Introduction to  $\text{\LaTeX}$ 
  - Basic Concepts and Overleaf
  - Article
  - Beamer
- Data Visualization
  - matplotlib
  - seaborn
- Regression(2)
  - Multiple Linear Regression
  - Diagnosis
  - Extensions
  - Generalized Linear Model

- ❶ 크롤링은 혹시 모를 문제를 피하기 위하여 과제에서 수행하지 않습니다. 관심 있으신 경우 구글링해보세요.
- ❷ HW1의 목표는 적절한 연산을 통해
  - 생산자물가지수와 소비자물가지수의 차
  - 수출물가지수와 수입물가지수의 차의 상관관계를 선형회귀분석으로 확인해보는 것입니다.
- ❸ 자료들을 적절한 방식으로 요약하고, 정리하고, 새로운 자료를 만들고, 그 분포를 간단히 확인해보는 방법을 체험해볼 수 있습니다. 수업시간에 배운 내용에서 조금 더 나아가야 합니다.
- ❹ 제가 만들고 제가 안해봐서(ㅋㅋㅋ) 오류가 있을 수 있습니다. 오류 있으면 말씀해 주세요!