KAIST MFE, 2024 Spring

Kim Hyeonghwan

2024-04-29

Table of contents

Welcome!

안녕하세요, KAIST MFE 24년 봄학기에 이수한 과목의 과제 등을 정리해두었습니다.

Part I

Python('24봄)

1 Python Basics

1.1 Values and Variables

1.1.1 - Values

```
type("Hello, World")
```

str

```
type(())
```

tuple

```
type([])
```

list

```
type(set())
```

set

```
type({})
```

dict

```
# 파이썬에서는 복소수 i를 j로 표현함
type(2+3j)
```

complex

1.1.2 - Variables

```
message="Hello, world!"
n=42
e=2.71
print(n)
print(message)
```

```
42
Hello, world!
```

```
length=4.2
height=3.5
area=length*height
print(area)
```

```
14.700000000000001
```

```
# 변수는 방문앞에 이름만 바꾼것... 새로운 변수에 기존 변수를 할당하고, 기존 변수를 변경하면 새로운 변수도 영향
a=[1,2,3]
b=a
a[0]=10
print(b)
```

```
[10, 2, 3]
```

1.1.3 - Variable names

- 숫자로 시작하면 안됨

```
x=1.0
X=1.0
X1=1.0
x1=1.0
dell=1.0
# not allowed
# x:, 1X, X1-1, for
```

```
x,y,z=1,3.14,'a'
print(x,y,z)
```

```
1 3.14 a
```

1.2 Core Data Types

1.2.1 - Floating Point(float)

```
x=1
y=1.0
z=float(1)
xx=1.234e-5
print(x,y,z,xx)
print(type(z),type(float(True)))
```

```
1 1.0 1.0 1.234e-05
<class 'float'> <class 'float'>
```

```
# float 타입은 근사치를 사용하기때문에 사용에 유의
1-0.9==0.1
```

```
False
```

```
float('inf')
```

```
inf
```

1.2.2 - Complex

```
x=1j
y=2+3j
z=complex(1)
print(x,y,z,y.real,y.imag)
print(type(y.imag))
```

```
1j (2+3j) (1+0j) 2.0 3.0
<class 'float'>
```

1.2.3 - Integer

```
# Integer는 float와 달리 정확하다
x=2**127+2**65
x
```

```
170141183460469231768580791863303208960
```

1.2.4 - Boolean(bool)

```
print(bool(1),bool(1.2),bool(-1),bool(0.1))
```

```
True True True True
```

```
print(bool(0),bool(0.0),bool(0.0000),bool(None),bool(""),bool(()))
```

```
False False False False False False
```

```
3>4
```

```
False
```

1.3 Operators

1.3.1 - Arithmetic Operators

```
a=10
b=20
```

```
# 나누기 연산은 항상 float로 반환됨을 유의
print(a+b,a-b,a*b,b/a)
```

```
30 -10 200 2.0
```

```
# 나머지 연산자
b%a
```

0

```
# 몫 연산자
b//a
```

2

```
# 거듭제곱 연산자, 우선순위가 가장 높음
a**b
```

10000000000000000000

1.3.2 - Relational Operators

```
print(a==b,a!=b,a>b,a<b,a>=b,a<=b)
```

False True False True False True

1.3.3 - Assignment Operators

```
t=0
t+=1
t
```

1

```
t/=2
t
```

0.5

```
t*=2
t
```

1.0

1.3.4 - Logical Operators

```
print(True and False, True or False, not True)
```

```
False True False
```

```
# 논리연산자는 container가 bool이 아닌 경우 용법이 다름
# or은 가장 먼저나오는 True값을 반환, and는 가장 먼저나오는 false값을 반환
print('' or 'abc',0 or 1, 'abc' or '','abc' or 'ab')
print(1 and 0, 'abc' and 0, None and '' and 0)
```

```
abc 1 abc abc
0 0 None
```

1.4 Python Operators Precedence

1.4.1 - Python Operators Precedence

```
# 거듭제곱 > 덧샘뺄샘 > 나눗셈, 몫, 나머지 > 부등호 > 논리
1+3/2
```

```
2.5
```

```
2400//500*500+2000%500
```

```
2000
```

```
(2400//500)*500+(2000%500)
```

```
2000
```

1.5 Mathmatical Function

```
# built-in
print(abs(-1),round(1.111,1))
```

```
1 1.1
```

```
# functional in math module
import math

# factorial, floor, isfinite, isinf, isnan, trunc
# exp, log, log10, sqrt, cos, sin, tan, degrees, radians, pi, e

print(
math.ceil(1.11),
math.log(10),
math.log(10,10),
math.exp(1),
math.e,
math.pi,
math.trunc(math.pi),
math.degrees(1),
math.cos(math.pi)
)
```

```
2 2.302585092994046 1.0 2.718281828459045 2.718281828459045 3.141592653589793 3 57.29577951308232
```

1.6 Python I/O

1.6.1 - Python Output

```
# print(object1, object2, ... , sep=' ', end='\n',...)
print(1,2,3,4)
print(1,2,3,4,sep='*')
print(1,2,3,4,end='END')
print(1,2,3,4,sep='')
```

```
1 2 3 4
1*2*3*4
1 2 3 4END1234
```

```
# formatting
x=5; y=10
print('The value of x is {} and y is {}'.format(x,y))
print('I love {2} and {1}, but hate {0}'.format('cucumber','butter','bread'))
```

```
The value of x is 5 and y is 10
I love bread and butter, but hate cucumber
```

1.6.2 - Python Input

```
# Input value is a string
num=input('Enter a number:')
print(num)
print(int(num))
```

```
10
10
```

2 Containers

- 기초적인 내용, String / Integer / Dictionary 많이 사용함 ## Strings

Strings : An immutable sequence of unicode characters

```
"You can put single quotes ' in a string with double quotes on the outside"
```

```
"You can put single quotes ' in a string with double quotes on the outside"
```

```
'and visa versa (")'
```

```
'and visa versa (")'
```

```
print(
"""muti-line strings
with three quotes""")
```

```
muti-line strings
with three quotes
```

2.0.1 - Accessing Values in Strings

```
# 잘 알아둘것, 많이 사용
word='KAIST MFE Program'
# indexing
print(word[0],word[5],word[6],sep="/")
# slicing
# str[a:b:c] = start a, end b, jump c
# a is inclusive, b is exclusive, c can be minus
# default : a =0, b=n+1, c=1 but when c is minus then default : a=n, b=0(inclusive)
```

```
print(word[0:5],word[6:9],word[10:])
print(word[:2],word[-2:])
print(word[0:10:2])
print(word[::-1])
```

```
K/ /M
KAIST MFE Program
KA am
KITME
margorP EFM TSIAK
```

```
text='Python strings are sliceable.'
text[10]
```

```
'i'
```

```
length=len(text)
text[length]
```

```
IndexError: string index out of range
```

```
text[length-1]
```

```
'.'
```

2.0.2 - String operations

- +, -, in, not in

```
a='Hello'; b='Python'
print(a+b, a*2+b*3, 'H' in a, 'm' not in a)
```

```
HelloPython HelloHelloPythonPythonPython True True
```

```
'123'+1
```

```
TypeError: can only concatenate str (not "int") to str
```

```
'123'+str(1)
```

```
'1231'
```

```
int('123')+1
```

```
124
```

2.0.3 - Built-in string function and methods

```
# len, .find, .count, .lower, .upper, .split, .replace, .starswith, .endswith
# .isalpha, .isnumeric, .isalnum
# len is function, others are methods
print(
    word,
    len(word),
    word.find('I'),
    word.count(' '),
    word.lower(),
    word.upper(),
    word.split(' '),
    word.replace(' ','-'),
    sep='\n'
)
```

```
KAIST MFE Program
17
2
2
kaist mfe program
KAIST MFE PROGRAM
['KAIST', 'MFE', 'Program']
KAIST-MFE-Program
True
False
```

```
False
True
False
True
False
```

```
print(
    word.startswith('K'),
    word.endswith('M'),
    word.isalpha(),
    word.replace(' ','').isalpha(),
    word.isnumeric(),
    '123'.isnumeric(),
    word.isalnum(),
    word.replace(' ','1').isalnum(),
    sep='\n'
)
```

```
True
False
False
True
False
True
False
True
```

2.0.4 - Immutable String

```
# string = immutable, not possible to modify
# method or operations on string != modify but make new string
sequence="ACGT"
sequence.replace("A","G")
```

```
'GCGT'
```

```
sequence
```

```
'ACGT'
```

```
id(sequence)
```

```
2053186997616
```

```
sequence=sequence.replace("A","G")
did(sequence)
```

```
2053183781744
```

2.1 Lists

```
# a mutable sequence of objects, important!!
# To create List object, use [] or list()
x1=[1,2,3,4]
x2=[1,1.0,1+0j,'one',None,True]
x3=[[1,2,3,4],[5,6,7],[8,9]] # nested list
# 리스트 안에는 무엇이든 올 수 있음
print(x1,x2,x3,sep='\n')
```

```
[1, 2, 3, 4]
[1, 1.0, (1+0j), 'one', None, True]
[[1, 2, 3, 4], [5, 6, 7], [8, 9]]
```

```
e1=list()
type(e1)
```

```
list
```

```
len(e1)
```

```
NameError: name 'e1' is not defined
```

```
list('cat')
```

```
['c', 'a', 't']
```

```
a_tuple=('abc')
list(a_tuple)
```

```
['a', 'b', 'c']
```

2.1.1 - Accessing Items in lists

```
# indexing -> original object, slicing -> make sub-list
myList=[5,2.3,'hello']
print(
    myList[0],
    myList[-1],
    myList[-3]
)
```

```
5 hello 5
```

```
many_types=[1,55.5,"Am I in a list?",True,"the end"]
print(
    many_types[2:4],
    many_types[2:],
    many_types[:3],
    many_types[-4:-2],
    many_types[-2:-5:-1],
    many_types[:-3],
    many_types[:3:2],
    many_types[:-2:2],
    sep='\n'
)
```

```
['Am I in a list?', True]
['Am I in a list?', True, 'the end']
[1, 55.5, 'Am I in a list?']
[55.5, 'Am I in a list?']
[True, 'Am I in a list?', 55.5]
[1, 55.5]
[1, 'Am I in a list?']
[1, 'Am I in a list?']
```

```
# Multidimensional lists can alse be indexed and sliced.
x=[[1,2,3],[4,5,6],[7,8,9,0]]
x[0]
```

```
[1, 2, 3]
```

```
x[0][1]
```

```
2
```

```
x[0][0:2]
```

```
[1, 2]
```

2.1.2 - Updating Lists

```
# List object is an immutable!!
colorList=['red','blue','green','black','white']
print(id(colorList),colorList)
```

```
2665431628288 ['red', 'blue', 'green', 'black', 'white']
```

```
colorList[2]='yellow'
print(id(colorList),colorList)
```

```
2665431628288 ['red', 'blue', 'yellow', 'black', 'white']
```

```
colorList[2:4]=['gray','purple']
colorList
```

```
['red', 'blue', 'gray', 'purple', 'white']
```

```
colorList[1:4]='r e d'
colorList
```

```
['red', 'r', ' ', 'e', ' ', 'd', 'white']
```

```
colorList[1:]=['orange']
colorList
```

```
['red', 'orange']
```

2.1.3 - Basic List Operations

```
L1=[1,2,3]; L2=[4,5,6]; L3=[7,8,9,0]
print(
    L1+L2+L3,
    [L1],
    [L1]+[L2]+[L3],
    sep='\n')
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
[[1, 2, 3]]
[[1, 2, 3], [4, 5, 6], [7, 8, 9, 0]]
```

```
print(
    L1*3,
    3*L1,
    [L1]*3,
    [L1+L2]*3,
    [1,2,3] in [L1]+[L2],
    sep='\n'
```

```
)
```

```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
[1, 2, 3, 1, 2, 3, 1, 2, 3]
[[1, 2, 3], [1, 2, 3], [1, 2, 3]]
[[1, 2, 3, 4, 5, 6], [1, 2, 3, 4, 5, 6], [1, 2, 3, 4, 5, 6]]
True
```

2.1.4 - Built-in List Functions and Mothods

```
# len, min, max, del x[slice]
# x.append(value), x.extend(list), x.remove(value), x.count(value),
# x.insert(index, value), x.index(value), x.sort(), x.reverse()
x=[0,1,2,3,4,5,6,7,8,9]
del x[0] # 많이 사용
x
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
del x[:3]
x
```

```
[4, 5, 6, 7, 8, 9]
```

```
del x[:]
x
```

```
[]
```

```
x=[0,1,2,3,4,5,6,7,8,9]
len(x)
```

```
10
```

```
# in place
x.reverse()
x
```

```
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

```
# not in place
sorted(x)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
x
```

```
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

```
# in place
x.sort()
x
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

2.1.5 - Coping List

```
pockets=[5,3,6,7]
pockets_copy=pockets
pockets_copy.append(1)
pockets_copy
```

```
[5, 3, 6, 7, 1]
```

```
# pockets_copy is just labeling of pockets
pockets
```

```
[5, 3, 6, 7, 1]
```

```
# [:] makes copy of original list object
# At list [:] is copy, but at Array(numpy) [:] is view
pockets_realcopy=pockets[:]
pockets_realcopy.append(1)
pockets_realcopy
```

```
[5, 3, 6, 7, 1, 1]
```

```
pockets
```

```
[5, 3, 6, 7, 1]
```

```
pockets_method=pockets.copy()
pockets_method.append(1)
pockets_method
```

```
[5, 3, 6, 7, 1, 1]
```

```
pockets
```

```
[5, 3, 6, 7, 1]
```

2.2 Tuples

2.2.0.1 : tuples are basically immutable lists. To create a tuple, () or tuple()

```
t=(12345,54321,'hello')
t
```

```
(12345, 54321, 'hello')
```

```
# if it contains a single variables, should include comma!
x=(2)
type(x)
```

```
int
```

```
x=(2,)
type(x)
```

```
tuple
```

```
a=1,2,3,'hello'
a
```

```
(1, 2, 3, 'hello')
```

```
w,x,y,z=a
print(w,x,y,z)
```

```
1 2 3 hello
```

```
x,_,y,_=a
print(x,y)
```

```
1 3
```

```
x,*y=a
print(x,y)
```

```
1 [2, 3, 'hello']
```

2.2.1 - Accesing values in tuples

```
# can be indexed or sliced.
x=(1,2,3,4,5,6,7,8,9)
print(
    x[5],
    x[-3],
    x[3:7],
    x[3:7:2],
    sep='\n'
```

```
)
```

```
6
7
(4, 5, 6, 7)
(4, 6)
```

2.2.2 - Updating Tuples

```
# Tuples are immutable : impossible to add/remove/replace elements in a tuple
tup1=(12,34.56)
tup1[0]=100
```

```
TypeError: 'tuple' object does not support item assignment
```

```
tup2=([1,2],[3,4])
tup2[0][0]=100
tup2
```

```
([100, 2], [3, 4])
```

2.2.3 - Basic tuples operations

```
tup1=(1,2,3,4)
tup2='a','b','c'
tup1+tup2
```

```
(1, 2, 3, 4, 'a', 'b', 'c')
```

```
tup1*3
```

```
(1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4)
```

```
'a' in tup2
```

```
True
```

```
tup1[0:1]+tup2[1:]
```

```
(1, 'b', 'c')
```

```
tup1[0:1]+tup2[1]
```

```
TypeError: can only concatenate tuple (not "str") to tuple
```

```
# concatenation
tup_even=tup1[1::2]+(6,)
tup_even
```

```
(2, 4, 6)
```

2.2.4 - Built-in tuple functions and methods

```
# len, max, min, x.index, x.count
animals=('lama','sheep','lama',48)
len(animals)
```

```
4
```

```
print(
    animals.index('lama'),
    animals.count('sheep'),
    sep='\n')
```

```
0
1
```

```
# to fix tuple, tuple -> list -> tuple
x_list=list(x)
x_list.reverse()
x=tuple(x_list)
x
```

```
(9, 8, 7, 6, 5, 4, 3, 2, 1)
```

```
# tuple is faster than list!
%timeit x=[1,2,3,4,5]
%timeit x=(1,2,3,4,5)
```

```
47.3 ns ± 0.743 ns per loop (mean ± std. dev. of 7 runs, 10,000,000 loops each)
10.2 ns ± 0.0818 ns per loop (mean ± std. dev. of 7 runs, 100,000,000 loops each)
```

2.3 Dictionaries

2.3.0.1 : a mutable, unordered collection of key-value pairs

2.3.0.1.1 : unordered collection 최근 업데이트로 입력순으로 order가 생김. 그러나 sequence는 여전히 아님

2.3.0.2 : To create a dict, {key1:value1, key2:value2} initiate with {}, dict()

2.3.0.3 : Keys must be unique and immutable data! (If tuples, it can only contain immutable data.)

```
type({})
```

```
dict
```

```
# two-value sequenced can convert to dict easy!
lot=[(1,2),(3,4),(5,6)]
dict(lot)
```

```
{1: 2, 3: 4, 5: 6}
```

```
los=['a1','b2','c3']
dict(los)
```

```
{'a': '1', 'b': '2', 'c': '3'}
```

```
dict(a=1,b=2,c=3)
```

```
{'a': 1, 'b': 2, 'c': 3}
```

```
a="abc"
dict(a=1,b=2,c=3)
```

```
{'a': 1, 'b': 2, 'c': 3}
```

2.3.1 - Accesing values in dictionary

```
dict1=dict(Name='Zara', Age=7, Class='First')
dict1
```

```
{'Name': 'Zara', 'Age': 7, 'Class': 'First'}
```

```
dict1['Name']
```

```
'Zara'
```

2.3.2 - Updating dict

```
dict1['Age']=8
dict1['School']='ABC School'
dict1
```

```
{'Name': 'Zara', 'Age': 8, 'Class': 'First', 'School': 'ABC School'}
```

```
a='Town'
dict1[a]='Downtown'
dict1
```

```
{'Name': 'Zara',
 'Age': 8,
 'Class': 'First',
 'School': 'ABC School',
 'Town': 'Downtown'}
```

2.3.3 - Basic dict operations

```
# Check only key value!!
print(
    'Name' in dict1,
    'Age' in dict1,
    'Country' in dict1,
    'Country' not in dict1,
    'Zara' in dict1,
    sep='\n'
)
```

```
True
True
False
True
False
```

2.3.4 - Bulit-in dict functions and methods

```
# len, del x[key]
# x.clear(), x.items(), x.keys(), x.values(), x.update(dict2)
print(dict1, len(dict1), sep='\n')
```

```
{'Name': 'Zara', 'Age': 8, 'Class': 'First', 'School': 'ABC School', 'Town': 'Downtown'}
5
```

```
del dict1['Town']
dict1
```

```
{'Name': 'Zara', 'Age': 8, 'Class': 'First', 'School': 'ABC School'}
```

```
dict1.clear()
dict1
```

```
{}
```

```
del dict1
dict1
```

```
NameError: name 'dict1' is not defined
```

```
dict1=dict(a=1,b=2,c=3)
dict1.keys()
```

```
dict_keys(['a', 'b', 'c'])
```

```
list(dict1.keys())
```

```
['a', 'b', 'c']
```

```
dict1.values()
```

```
dict_values([1, 2, 3])
```

```
list(dict1.values())
```

```
[1, 2, 3]
```

```
dict1.items()
```

```
dict_items([('a', 1), ('b', 2), ('c', 3)])
```

```
d2=dict(apples=1,oranges=2,pears=2)
ud=dict(pears=4,grapes=5,lemons=6)
d2.update(ud)
d2
```

```
{'apples': 1, 'oranges': 2, 'pears': 4, 'grapes': 5, 'lemons': 6}
```

2.3.5 - Coping Dictionary

```
dict1
```

```
{'a': 1, 'b': 2, 'c': 3}
```

```
dict2=dict1.copy()
dict2.clear()
dict1
```

```
{'a': 1, 'b': 2, 'c': 3}
```

```
dict2=dict1
dict2.clear()
dict1
```

```
{}
```

2.4 Sets

2.4.0.1 : unordered collection with immutable and unique elements -> Just dict.key !

2.4.0.2 : initiate should be set(), {} is dict!!

```
# elements of set should be immutable!
list1=[[1,2],[3,4]]
set(list1)
```

```
TypeError: unhashable type: 'list'
```

```
list2=[(1,2),(3,4)]
set(list2)
```

```
{(1, 2), (3, 4)}
```

```
# if elements are not unique, automatically remove duplicates
list3=[1,2,2,3,4]
```

```
set(list3)
```

```
{1, 2, 3, 4}
```

```
dict1=dict(a=1,b=2,c=3)
set(dict1)
```

```
{'a', 'b', 'c'}
```

```
set(dict1.keys())
```

```
{'a', 'b', 'c'}
```

```
set(dict1.values())
```

```
{1, 2, 3}
```

```
set(dict1.items())
```

```
{('a', 1), ('b', 2), ('c', 3)}
```

2.4.1 - Accesing or Updating items in a Set

2.4.1.1 indexing and slicing not supported!!

2.4.1.2 set is mutable but set elements are immutable

2.4.2 - Basic sets operation

2.4.3 Warning!

2.4.3.1 Shallow copy vs. Deep copy

```
import copy as cp
```

```
A=dict(a=1,b=2,c=[1,2,3])
B=A.copy()
C=cp.deepcopy(A)

B['a']=9
B['c'][0]=9
print(A,B,C,sep='\n')
```

```
{'a': 1, 'b': 2, 'c': [9, 2, 3]}
{'a': 9, 'b': 2, 'c': [9, 2, 3]}
{'a': 1, 'b': 2, 'c': [1, 2, 3]}
```

3 Control Flows

3.0.0.1 - Control statements in Python

- Conditional Constructs : if
- Loops : for, while

3.1 Statements & Comments

3.2 if Conditionals

```
# if condition:
#    statement
# ---optional!---
# elif condition:
#     statement
# elif condition:
#     statement
# else:
#     statement
exam1=90; exam2=85
if(exam1>=90) and (exam2>=90):
    print('Excellent!')
```

```
if(exam1>=90) | (exam2>=90):
    print('Good')
```

```
Good
```

```
a=10
if a>10:
    print("a>10")
else:
    print("a=10")
```

```
a=10
```

```
# Exercise 1
a=[1,2,3,4,6,7,9]
b=a.copy()
b.sort()

if a==b:
    print("a is sorted")
else:
    print("a is not sorted")
```

```
a is sorted
```

```
# Exercise2
x=[1,2,3,4,5,6,7,8]

x_sort=x.copy()
x.sort()

len_x=len(x_sort)
mid_even=(x_sort[int(len(x_sort)/2-1)]+x_sort[int(len(x_sort)/2)])/2
mid_odd=x_sort[int(len(x_sort)//2)]

if len(x_sort)%2==0:
    print("n={}이고, 중간값은 {}입니다.".format(len_x,mid_even))
elif len(x_sort)%2==1:
    print("n={}이고, 중간값은 {}입니다.".format(len_x,mid_odd))
else :
    print("중간값을 산출할 수 없습니다.")
```

n=8이고, 중간값은 4.5입니다.

3.3 Loops - for loop

```
# for interating_var in iterable:
#    statements
```

```
# exercise 3

a=input("When you enter...")
a_int=int(a)
print("Input a number: "+a,
          "The output will be...",
          sep="\n")
for i in range(1,10):
    print(a+" x "+str(i)+" = "+str(a_int*i))
```

```
Input a number: 99
The output will be...
99 x 1 = 99
99 x 2 = 198
99 x 3 = 297
99 x 4 = 396
99 x 5 = 495
99 x 6 = 594
99 x 7 = 693
99 x 8 = 792
99 x 9 = 891
```

```
# exercise 4
sum_even=0
sum_odd=0
for i in range(10001):
    if i%2==0:
        sum_even+=i
```

```
    else:sum_odd+=i

print(sum_even,
      sum_odd,
      sum_even+sum_odd,
      sep='\n')
```

```
25005000
25000000
50005000
```

```
# exercise 5
fibonacci=[1,1]
for i in range(10):
    fibonacci.append(fibonacci[i]+fibonacci[i+1])
print(
    fibonacci,
    sum(fibonacci),
    sep='\n'
)
```

```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144]
376
```

```
# exercise 6
a=input("When you enter alpha-num string...")
letter_index=list()
digits_index=list()
non_index=list()
print(
    "Sample Data : "+a,
    "The output will be...",
    sep='\n'
)
for i in range(0,len(a)):
    if a[i].isalpha():
```

```
        letter_index.append(i)
    elif a[i].isnumeric():
        digits_index.append(i)
    else: non_index.append(i)

print(
    "Letters {} at {}".format(len(letter_index),letter_index),
    "Digits {} at {}".format(len(digits_index),digits_index),
    "And there are {} at {} which are NOT alphanumeric".format(len(non_index),non_index),
    sep='\n'
)
```

```
Sample Data : sample asdfl11231,2,1,3.2  21
The output will be...
Letters 11 at [0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11]
Digits 11 at [12, 13, 14, 15, 16, 18, 20, 22, 24, 27, 28]
And there are 7 at [6, 17, 19, 21, 23, 25, 26] which are NOT alphanumeric
```

4 Functions

4.0.0.1 Sqrt 사용자 정의 함수 실습

```
def mySqrt (n):
    old=0
    new=n/2
    while abs(old-new)>1e-10 :
        old=new
        new=1/2*(old+n/old)
    return new
```

```
mySqrt(3)
```

```
1.7320508075688772
```

```
list_int=[1,2,3,4,5,6,7,8]
list_str=["1","2","3","4","5","6","7","8"]
```

```
list(filter(lambda x:x,list_str))
```

```
['1', '2', '3', '4', '5', '6', '7', '8']
```

```
list(filter(lambda x:x+"1",list_str))
```

```
['1', '2', '3', '4', '5', '6', '7', '8']
```

```
bool("2")
```

```
True
```

```
list(filter(lambda x:x,list_int))
```

```
[]
```

```
list(filter(lambda x:bool(x-1),list_x))
```

```
[2, 3, 4]
```

```
filter(lambda x:x-1,list_int)
```

```
<filter at 0x1f5d235af10>
```

5 Numpy

```
import numpy as np
```

```
a=np.array([np.arange(6)]*6)+np.arange(0,51,10).reshape(6,1)
a
```

```
array([[ 0,  1,  2,  3,  4,  5],
       [10, 11, 12, 13, 14, 15],
       [20, 21, 22, 23, 24, 25],
       [30, 31, 32, 33, 34, 35],
       [40, 41, 42, 43, 44, 45],
       [50, 51, 52, 53, 54, 55]])
```

```
a[0,3:5]
```

```
array([3, 4])
```

```
a[4:,4:]
```

```
array([[44, 45],
       [54, 55]])
```

```
a[0:,2]
```

```
array([ 2, 12, 22, 32, 42, 52])
```

```
a[0:,2:3]
```

```
array([[ 2],
       [12],
       [22],
       [32],
       [42],
       [52]])
```

```
a[0:,2]=[1,2,3,4,5,6]
a
```

```
array([[ 0,  1,  1,  3,  4,  5],
       [10, 11,  2, 13, 14, 15],
       [20, 21,  3, 23, 24, 25],
       [30, 31,  4, 33, 34, 35],
       [40, 41,  5, 43, 44, 45],
       [50, 51,  6, 53, 54, 55]])
```

```
a[2::2,0:6:2]
```

```
array([[20,  3, 24],
       [40,  5, 44]])
```

```
# Operations on Arrays
# Numpy aggregates

# Exercise 4
# F=9/5C+32

c=np.arange(20,31)
c*9/5+32
```

```
array([68. , 69.8, 71.6, 73.4, 75.2, 77. , 78.8, 80.6, 82.4, 84.2, 86. ])
```

```
# Exercise 5
np.random.seed(1)
Z=np.random.rand(5)
```

```
print(Z,(Z-Z.min())/(Z.max()-Z.min()),sep="\n")
```

```
[4.17022005e-01 7.20324493e-01 1.14374817e-04 3.02332573e-01
 1.46755891e-01]
[0.57886944 1.         0.         0.41962504 0.20360935]
```

```
# Exercise 6
Z=np.random.uniform(0,10,10)
np.floor(Z)
```

```
array([6., 4., 5., 1., 1., 8., 9., 3., 6., 8.])
```

```
np.ceil(Z)-1
```

```
array([6., 4., 5., 1., 1., 8., 9., 3., 6., 8.])
```

```
Z-Z%1
```

```
array([6., 4., 5., 1., 1., 8., 9., 3., 6., 8.])
```

```
# Exercise 7
x=np.zeros((6,6))
y=np.arange(0,6)
```

```
x+y
```

```
array([[0., 1., 2., 3., 4., 5.],
       [0., 1., 2., 3., 4., 5.],
       [0., 1., 2., 3., 4., 5.],
       [0., 1., 2., 3., 4., 5.],
       [0., 1., 2., 3., 4., 5.],
       [0., 1., 2., 3., 4., 5.]])
```

```
x+y[:,np.newaxis]
```

```
array([[0., 0., 0., 0., 0., 0.],
       [1., 1., 1., 1., 1., 1.],
       [2., 2., 2., 2., 2., 2.],
       [3., 3., 3., 3., 3., 3.],
       [4., 4., 4., 4., 4., 4.],
       [5., 5., 5., 5., 5., 5.]])
```

```
x+y.reshape(6,1)
```

```
array([[0., 0., 0., 0., 0., 0.],
       [1., 1., 1., 1., 1., 1.],
       [2., 2., 2., 2., 2., 2.],
       [3., 3., 3., 3., 3., 3.],
       [4., 4., 4., 4., 4., 4.],
       [5., 5., 5., 5., 5., 5.]])
```

```
# Exercise 9
np.random.seed(123)
mat = np.random.randn(6, 4); mat
```

```
array([[-1.0856306 ,  0.99734545,  0.2829785 , -1.50629471],
       [-0.57860025,  1.65143654, -2.42667924, -0.42891263],
       [ 1.26593626, -0.8667404 , -0.67888615, -0.09470897],
       [ 1.49138963, -0.638902  , -0.44398196, -0.43435128],
       [ 2.20593008,  2.18678609,  1.0040539 ,  0.3861864 ],
       [ 0.73736858,  1.49073203, -0.93583387,  1.17582904]])
```

```
mat=np.concatenate([mat,np.exp(mat[:,0])[:,np.newaxis]],axis=1)
mat
```

```
array([[-1.0856306 ,  0.99734545,  0.2829785 , -1.50629471,  0.33768877],
       [-0.57860025,  1.65143654, -2.42667924, -0.42891263,  0.56068263],
       [ 1.26593626, -0.8667404 , -0.67888615, -0.09470897,  3.54641154],
       [ 1.49138963, -0.638902  , -0.44398196, -0.43435128,  4.44326571],
       [ 2.20593008,  2.18678609,  1.0040539 ,  0.3861864 ,  9.07869158],
       [ 0.73736858,  1.49073203, -0.93583387,  1.17582904,  2.09042747]])
```

```
mat=mat+[1,-1,1,-1,1]; mat
```

```
array([[-8.56306033e-02, -2.65455342e-03,  1.28297850e+00,
        -2.50629471e+00,  1.33768877e+00],
       [ 4.21399748e-01,  6.51436537e-01, -1.42667924e+00,
        -1.42891263e+00,  1.56068263e+00],
       [ 2.26593626e+00, -1.86674040e+00,  3.21113848e-01,
        -1.09470897e+00,  4.54641154e+00],
       [ 2.49138963e+00, -1.63890200e+00,  5.56018040e-01,
        -1.43435128e+00,  5.44326571e+00],
       [ 3.20593008e+00,  1.18678609e+00,  2.00405390e+00,
        -6.13813601e-01,  1.00786916e+01],
       [ 1.73736858e+00,  4.90732028e-01,  6.41661316e-02,
         1.75829045e-01,  3.09042747e+00]])
```

```
mat=mat*np.array([1/10,1/10,1/10,10,10,10]).reshape(6,1)
mat
```

```
array([[-8.56306033e-03, -2.65455342e-04,  1.28297850e-01,
        -2.50629471e-01,  1.33768877e-01],
       [ 4.21399748e-02,  6.51436537e-02, -1.42667924e-01,
        -1.42891263e-01,  1.56068263e-01],
       [ 2.26593626e-01, -1.86674040e-01,  3.21113848e-02,
        -1.09470897e-01,  4.54641154e-01],
       [ 2.49138963e+01, -1.63890200e+01,  5.56018040e+00,
        -1.43435128e+01,  5.44326571e+01],
       [ 3.20593008e+01,  1.18678609e+01,  2.00405390e+01,
        -6.13813601e+00,  1.00786916e+02],
       [ 1.73736858e+01,  4.90732028e+00,  6.41661316e-01,
         1.75829045e+00,  3.09042747e+01]])
```

```
np.median(mat,axis=0)
```

```
array([ 8.80013969,  0.0324391 ,  0.38497958, -0.19676037, 15.67945792])
```

```
np.percentile(mat,70,axis=0)
```

```
array([21.14379101,  2.48623197,  3.10092086, -0.12618108, 42.66846589])
```

```
np.sort(mat[2,:])[::-1]
```

```
array([ 0.45464115,  0.22659363,  0.03211138, -0.1094709 , -0.18667404])
```

```
np.argsort(mat[2,:])[::-1]
```

```
array([4, 0, 2, 3, 1])
```

```
np.sum(np.where(np.exp(mat)>3,1,0),axis=1)
```

```
array([0, 0, 0, 3, 4, 4])
```

```
# or
np.sum(np.exp(mat)>3,axis=1)
```

```
array([0, 0, 0, 3, 4, 4])
```

```
np.nonzero(mat>np.mean(mat,axis=0))
```

```
(array([0, 1, 1, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5]),
 array([3, 1, 3, 3, 0, 2, 4, 0, 1, 2, 4, 0, 1, 3]))
```

6 Pandas

```
import pandas as pd
import numpy as np
```

```
Xdf=pd.DataFrame(np.arange(16).reshape(4,4))
Xdf
```

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

```
Xdf[0]
```

```
0     0
1     4
2     8
3    12
Name: 0, dtype: int64
```

```
Xdf[0:3]
```

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11

```
Xdf[Xdf[1]<10]
```

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11

```
Xdf[Xdf<10]
```

	0	1	2	3
0	0.0	1.0	2.0	3.0
1	4.0	5.0	6.0	7.0
2	8.0	9.0	NaN	NaN
3	NaN	NaN	NaN	NaN

```
Xdf[[0,3,2]]
```

	0	3	2
0	0	3	2
1	4	7	6
2	8	11	10
3	12	15	14

```
Xdf.loc[0:1]
```

	0	1	2	3
0	0	1	2	3
1	4	5	6	7

```
Xdf.iloc[0:1]
```

	0	1	2	3
0	0	1	2	3

```
Xdf[0:1]
```

	0	1	2	3
0	0	1	2	3

```
# Exercise 2
DF=pd.DataFrame(dict(X1=np.random.uniform(0,1,15),
                     X2=np.random.standard_normal(15),
                     X3=1))
DF
```

	X1	X2	X3
0	0.002925	-0.693743	1
1	0.033349	0.774359	1
2	0.879925	-0.317643	1
3	0.810953	0.087859	1
4	0.655664	0.336143	1
5	0.748835	-0.021620	1
6	0.971919	-0.463584	1
7	0.933795	0.763103	1
8	0.898300	-0.311474	1
9	0.382806	-0.697514	1
10	0.621569	1.838894	1
11	0.693038	0.949865	1
12	0.629064	-0.204435	1
13	0.961133	2.522102	1
14	0.370875	1.190071	1

```
DF[(DF['X1']>0.5)&(DF['X1']<=0.8)].index
```

```
Index([4, 5, 10, 11, 12], dtype='int64')
```

```
DF.loc[(DF['X1']>0.5)&(DF['X1']<=0.8)].index
```

```
Index([4, 5, 10, 11, 12], dtype='int64')
```

```
DF.loc[(DF['X1']>0.5)&(DF['X1']<=0.8),:].index
```

```
Index([4, 5, 10, 11, 12], dtype='int64')
```

```
DF.iloc[np.array((DF['X1']>0.5)&(DF['X1']<=0.8))].index
```

```
Index([4, 5, 10, 11, 12], dtype='int64')
```

```
DF[['X3','X2','X1']]
```

	X3	X2	X1
0	1	-0.693743	0.002925
1	1	0.774359	0.033349
2	1	-0.317643	0.879925
3	1	0.087859	0.810953
4	1	0.336143	0.655664
5	1	-0.021620	0.748835
6	1	-0.463584	0.971919
7	1	0.763103	0.933795
8	1	-0.311474	0.898300
9	1	-0.697514	0.382806
10	1	1.838894	0.621569
11	1	0.949865	0.693038
12	1	-0.204435	0.629064
13	1	2.522102	0.961133
14	1	1.190071	0.370875

```
DF.loc[DF['X2']<0,['X1']]
```

	X1
0	0.002925
2	0.879925
5	0.748835
6	0.971919
8	0.898300
9	0.382806
12	0.629064

6.0.0.1 Input / Output functions in Pandas

```
# read_csv, read_excel, read_table, read_fwf ...
import os
os.getcwd()
```

```
'/Users/hwan/Desktop/Homepage/study_24spring'
```

```
pd.read_csv('pandas.csv')
```

	S.No	Name	Age	City	Salary
0	1	Tom	28	Toronto	20000
1	2	Lee	32	HongKong	3000
2	3	Steven	43	Bay Area	8300
3	4	Ram	38	Hyderabad	3900

```
pd.read_csv('pandas.csv',sep=',')
```

	S.No	Name	Age	City	Salary
0	1	Tom	28	Toronto	20000
1	2	Lee	32	HongKong	3000
2	3	Steven	43	Bay Area	8300
3	4	Ram	38	Hyderabad	3900

```
pd.read_csv('pandas.csv',index_col=0)
```

S.No	Name	Age	City	Salary
1	Tom	28	Toronto	20000
2	Lee	32	HongKong	3000
3	Steven	43	Bay Area	8300
4	Ram	38	Hyderabad	3900

```
pd.read_csv('pandas.csv',index_col='Name')
```

Name	S.No	Age	City	Salary
Tom	1	28	Toronto	20000
Lee	2	32	HongKong	3000
Steven	3	43	Bay Area	8300
Ram	4	38	Hyderabad	3900

```
pd.read_csv('pandas.csv',header=None)
```

	0	1	2	3	4
0	S.No	Name	Age	City	Salary
1	1	Tom	28	Toronto	20000
2	2	Lee	32	HongKong	3000
3	3	Steven	43	Bay Area	8300
4	4	Ram	38	Hyderabad	3900

```
pd.read_csv('pandas.csv',names=[0,1,2,3,4])
```

	0	1	2	3	4
0	S.No	Name	Age	City	Salary
1	1	Tom	28	Toronto	20000

	0	1	2	3	4
2	2	Lee	32	HongKong	3000
3	3	Steven	43	Bay Area	8300
4	4	Ram	38	Hyderabad	3900

```
pd.read_csv('pandas.csv',names=[1,2,3,4])
```

	1	2	3	4
S.No	Name	Age	City	Salary
1	Tom	28	Toronto	20000
2	Lee	32	HongKong	3000
3	Steven	43	Bay Area	8300
4	Ram	38	Hyderabad	3900

```
pd.read_csv('pandas.csv',dtype={'Salary':np.float64})
```

	S.No	Name	Age	City	Salary
0	1	Tom	28	Toronto	20000.0
1	2	Lee	32	HongKong	3000.0
2	3	Steven	43	Bay Area	8300.0
3	4	Ram	38	Hyderabad	3900.0

```
pd.read_csv('pandas.csv',names=['a','b','c','d','e'])
```

	a	b	c	d	e
0	S.No	Name	Age	City	Salary
1	1	Tom	28	Toronto	20000
2	2	Lee	32	HongKong	3000
3	3	Steven	43	Bay Area	8300
4	4	Ram	38	Hyderabad	3900

```
pd.read_csv('pandas.csv',names=['a','b','c','d'],header=0,index_col=0)
```

	a	b	c	d
1	Tom	28	Toronto	20000
2	Lee	32	HongKong	3000
3	Steven	43	Bay Area	8300
4	Ram	38	Hyderabad	3900

```
pd.read_csv('pandas.csv',names=['a','b','c','d'],skiprows=1,index_col=0)
```

	a	b	c	d
1	Tom	28	Toronto	20000
2	Lee	32	HongKong	3000
3	Steven	43	Bay Area	8300
4	Ram	38	Hyderabad	3900

```
pd.read_csv('pandas.csv',na_values=['Tom'])
```

	S.No	Name	Age	City	Salary
0	1	NaN	28	Toronto	20000
1	2	Lee	32	HongKong	3000
2	3	Steven	43	Bay Area	8300
3	4	Ram	38	Hyderabad	3900

6.0.0.2 Binary Operations

```
A = pd.DataFrame(np.random.randint(0, 20, (2, 2)), columns=list('AB'))
B = pd.DataFrame(np.random.randint(0, 10, (3, 3)), columns=list('BAC'))
print( A, B, sep="\n\n")
```

```
    A  B
0   0  1
1  18  1

   B  A  C
```

```
0  1  9  9
1  4  6  2
2  4  1  0
```

```
A+B
```

	A	B	C
0	9.0	2.0	NaN
1	24.0	5.0	NaN
2	NaN	NaN	NaN

```
A.add(B,fill_value=0)
```

	A	B	C
0	9.0	2.0	9.0
1	24.0	5.0	2.0
2	1.0	4.0	0.0

```
B-B.loc[0,:]
```

	B	A	C
0	0	0	0
1	3	-3	-7
2	3	-8	-9

```
B-B.loc[0:0,:]
```

	B	A	C
0	0.0	0.0	0.0
1	NaN	NaN	NaN
2	NaN	NaN	NaN

```
B.sub(B.loc[0,:])
```

	B	A	C
0	0	0	0
1	3	-3	-7
2	3	-8	-9

```
B.sub(B['A'],axis=0)
```

	B	A	C
0	-8	0	0
1	-2	0	-4
2	3	0	-1

6.0.0.3 Pandas DataFrame Manipulation

```
import numpy as np
import pandas as pd
df = pd.DataFrame(np.random.rand(7, 5))
df
```

	0	1	2	3	4
0	0.254541	0.022793	0.292160	0.256052	0.742068
1	0.959650	0.210256	0.937732	0.523141	0.660096
2	0.694372	0.674311	0.502167	0.164087	0.527422
3	0.591313	0.952835	0.800817	0.357809	0.029770
4	0.004058	0.930224	0.908414	0.104988	0.372187
5	0.062482	0.480021	0.220693	0.820707	0.578013
6	0.678025	0.481827	0.727163	0.119174	0.252283

```
df[[0,1]]
```

	0	1
0	0.254541	0.022793
1	0.959650	0.210256

	0	1
2	0.694372	0.674311
3	0.591313	0.952835
4	0.004058	0.930224
5	0.062482	0.480021
6	0.678025	0.481827

7 7. Pandas Practice

```
import numpy as np
import pandas as pd
import FinanceDataReader as fdr
```

7.1 Practice 1 : Magic formula investing

```
krx = fdr.StockListing('KRX')
krx.iloc[:, :4]
```

	Code	ISU_CD	Name	Market
0	005930	KR7005930003	삼성전자	KOSPI
1	000660	KR7000660001	SK하이닉스	KOSPI
2	373220	KR7373220003	LG에너지솔루션	KOSPI
3	005380	KR7005380001	현대차	KOSPI
4	207940	KR7207940008	삼성바이오로직스	KOSPI
...	...	...	...	...
2805	002995	KR7002991008	금호건설우	KOSPI
2806	266170	KR7266170000	뿌리깊은나무들	KONEX
2807	217320	KR7217320001	썬테크	KONEX
2808	245450	KR7245450002	씨앤에스링크	KONEX
2809	308700	KR7308700004	테크엔	KONEX

```
df = pd.read_csv('investment_hw/PER_ROA.csv')
df.iloc[:5, [0, 3, 5, 6, 7, 8, 9]]
```

	종목명	등락률	거래량	시가총액	영업이익	PER	ROA
0	AJ네트웍스	-1.48%	27984	2177	-213.0	4.14	2.18
1	AJ렌터카	-0.47%	50886	2325	218.0	61.40	0.28
2	AK홀딩스	-1.99%	15535	4564	2697.0	4.27	6.52
3	ARIRANG 200	-0.10%	234206	7572	NaN	NaN	NaN
4	ARIRANG 200동일가중	-0.19%	31	46	NaN	NaN	NaN

```
df = df.loc [ df.isnull().sum(axis=1) == 0, : ]
```

```
per = per_val = df['PER']
per_val[ per < 0 ] = np.nan
per_rank = per_val.rank( ascending=True, na_option='bottom')
per_rank.head()
```

```
0      35.0
1     519.0
2      38.0
50    222.0
51    408.0
Name: PER, dtype: float64
```

```
roa= roa_val =  df['ROA']
roa_val[roa < 0] = np.nan
roa_rank = roa_val.rank( ascending=False, na_option='bottom')
roa_rank.head()
```

```
0     355.0
1     556.0
2     110.0
50    239.5
51     37.0
Name: ROA, dtype: float64
```

```
result = per_rank + roa_rank
result_rank = result.rank( ascending=True, na_option='bottom')
result_rank[ result_rank > 10 ] = 0
result_rank [ result_rank > 0 ] = 1
result_rank
```

```
0       0.0
1       0.0
2       0.0
50      0.0
51      0.0
       ...
1522    0.0
1523    0.0
1524    0.0
1526    0.0
1529    0.0
Length: 758, dtype: float64
```

```
result_rank.sum()
```

```
10.0
```

```
mf_df = df.loc[ result_rank > 0 , ['종목명', '시가총액']].copy()
mf_df
```

	종목명	시가총액
96	HDC	7468
440	SIMPAC	1835
459	SK하이닉스	541634
749	대한유화	7508
1040	세아제강지주	2044
1057	신대양제지	2756
1157	에쓰씨엔지니어링	354
1298	케이씨	1694

	종목명	시가총액
1443	한일홀딩스	2904
1513	효성	18353

```
mf_stock_list = df.loc[ result_rank > 0, '종목명'].values
mf_stock_list
```

```
array(['HDC', 'SIMPAC', 'SK하이닉스', '대한유화', '세아제강지주', '신대양제지', '에쓰씨엔지니어링',
       '케이씨', '한일홀딩스', '효성'], dtype=object)
```

```
krx.iloc[:,:4].head()
```

	Code	ISU_CD	Name	Market
0	005930	KR7005930003	삼성전자	KOSPI
1	000660	KR7000660001	SK하이닉스	KOSPI
2	373220	KR7373220003	LG에너지솔루션	KOSPI
3	005380	KR7005380001	현대차	KOSPI
4	207940	KR7207940008	삼성바이오로직스	KOSPI

```
mf_df['종목코드']=''
```

```
for stock in mf_stock_list:
    mf_df.loc[ mf_df['종목명']==stock, '종목코드']=krx[krx['Name']==stock]['Code'].values
mf_df
```

	종목명	시가총액	종목코드
96	HDC	7468	012630
440	SIMPAC	1835	009160
459	SK하이닉스	541634	000660
749	대한유화	7508	006650
1040	세아제강지주	2044	003030
1057	신대양제지	2756	016590
1157	에쓰씨엔지니어링	354	023960

	종목명	시가총액	종목코드
1298	케이씨	1694	029460
1443	한일홀딩스	2904	003300
1513	효성	18353	004800

```
for x in mf_df['종목코드'].values :
    df = fdr.DataReader( x, '2019-01-01', '2019-12-31' )

    cum_ret = df.loc[df.index[-1], 'Close'] / df.loc[df.index[0], 'Close']-1
    mf_df.loc[mf_df['종목코드']==x, '수익률'] = cum_ret

    historical_max = df['Close'].cummax()
    daily_drawdown = df['Close']/historical_max - 1.

    MDD = daily_drawdown.min()
    mf_df.loc[mf_df['종목코드']==x, '최대낙폭'] = MDD

    df['daily_rtn'] = df['Close'].pct_change( periods = 1 )
    VOL = df['daily_rtn'].std()*np.sqrt(252)
    mf_df.loc[mf_df['종목코드']==x, '변동성'] = VOL

    df = None
mf_df.sort_values('시가총액', ascending=False)
```

	종목명	시가총액	종목코드	수익률	최대낙폭	변동성
459	SK하이닉스	541634	000660	0.552805	-0.228606	0.353011
1513	효성	18353	004800	0.649635	-0.115124	0.300178
749	대한유화	7508	006650	-0.203390	-0.407609	0.346504
96	HDC	7468	012630	-0.332326	-0.504808	0.330864
1443	한일홀딩스	2904	003300	-0.197424	-0.363892	0.324106
1057	신대양제지	2756	016590	0.034783	-0.388466	0.364506
1040	세아제강지주	2044	003030	-0.008368	-0.228353	0.240910
440	SIMPAC	1835	009160	0.206967	-0.299566	0.407815
1298	케이씨	1694	029460	0.594017	-0.369942	0.442983

	종목명	시가총액	종목코드	수익률	최대낙폭	변동성
1157	에쓰씨엔지니어링	354	023960	-0.062092	-0.479592	0.565059

7.2 Practice 2 : Mean reversion strategy

```
df = pd.read_csv('investment_hw/SPY.csv')
df.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	1993-01-29	43.96875	43.96875	43.75000	43.93750	26.706757	1003200
1	1993-02-01	43.96875	44.25000	43.96875	44.25000	26.896694	480500
2	1993-02-02	44.21875	44.37500	44.12500	44.34375	26.953669	201300
3	1993-02-03	44.40625	44.84375	44.37500	44.81250	27.238594	529400
4	1993-02-04	44.96875	45.09375	44.46875	45.00000	27.352570	531500

```
df.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	6648.000000	6648.000000	6648.000000	6648.000000	6648.000000	6.648000e+03
mean	133.762935	134.541071	132.893598	133.759854	110.399391	8.440122e+07
std	59.488006	59.671285	59.277882	59.492056	64.113369	9.837713e+07
min	43.343750	43.531250	42.812500	43.406250	26.383823	5.200000e+03
25%	96.780937	97.735000	95.726562	96.921875	71.256485	6.966775e+06
50%	124.433750	125.335938	123.500000	124.312500	93.641503	5.709990e+07
75%	151.702503	152.514999	150.810624	151.791714	125.251474	1.229908e+08
max	296.040009	296.309998	293.760010	295.859985	294.427979	8.710263e+08

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6648 entries, 0 to 6647
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
```

```
---  ------     --------------  -----
 0   Date       6648 non-null   object
 1   Open       6648 non-null   float64
 2   High       6648 non-null   float64
 3   Low        6648 non-null   float64
 4   Close      6648 non-null   float64
 5   Adj Close  6648 non-null   float64
 6   Volume     6648 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 363.7+ KB
```

```
df['Date']=pd.to_datetime(df.Date)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6648 entries, 0 to 6647
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       6648 non-null   datetime64[ns]
 1   Open       6648 non-null   float64
 2   High       6648 non-null   float64
 3   Low        6648 non-null   float64
 4   Close      6648 non-null   float64
 5   Adj Close  6648 non-null   float64
 6   Volume     6648 non-null   int64
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 363.7 KB
```

```
price_df = df.loc[:,['Date', 'Adj Close']].copy()
price_df.head()
```

	Date	Adj Close
0	1993-01-29	26.706757
1	1993-02-01	26.896694

	Date	Adj Close
2	1993-02-02	26.953669
3	1993-02-03	27.238594
4	1993-02-04	27.352570

```
price_df.set_index(['Date'], inplace=True)
price_df.head()
```

	Adj Close
Date	
1993-01-29	26.706757
1993-02-01	26.896694
1993-02-02	26.953669
1993-02-03	27.238594
1993-02-04	27.352570

```
# 볼린저 밴드
# - 상단 밴드 : 중간밴드 + 2*20일 이동표준편차
# - 중간 밴드 : 20일 이동표준편차
# - 하단 밴드 : 중간밴드 - 2*20일 이동표준편차
n = 20
sigma = 2
def bollinger_band(price_df, n, sigma):
    bb = price_df.copy()
    bb['center']= price_df['Adj Close'].rolling(n).mean()
    bb['ub']= bb['center'] + sigma * price_df['Adj Close'].rolling(window=n).std()
    bb['lb']= bb['center'] - sigma * price_df['Adj Close'].rolling(window=n).std()
    return bb
```

```
bollinger = bollinger_band(price_df, n, sigma)
bollinger.tail()
```

Date	Adj Close	center	ub	lb
2019-06-18	290.984741	282.981668	293.213256	272.750080
2019-06-19	291.641541	283.307582	294.219075	272.396089
2019-06-20	294.427979	283.816605	295.809022	271.824187
2019-06-21	294.000000	284.477884	297.200319	271.755449
2019-06-24	293.640015	285.089319	298.355028	271.823610

```
base_date = '2009-01-02'
sample = bollinger.loc[base_date:]
sample.head()
```

Date	Adj Close	center	ub	lb
2009-01-02	75.099487	71.378963	74.538229	68.219698
2009-01-05	75.010582	71.711677	74.931608	68.491746
2009-01-06	75.511505	71.964058	75.543401	68.384716
2009-01-07	73.249435	71.980327	75.580937	68.379718
2009-01-08	73.548378	72.071645	75.736733	68.406557

```
def create_trade_book(sample):
    book = sample[['Adj Close']].copy()
    book['trade'] = ''
    return book
```

```
def tradings ( sample, book ):
    for i in sample.index:
        if sample.loc[i, 'Adj Close'] > sample.loc[i, 'ub']:
            book.loc[i, 'trade']=''
        elif sample.loc[i, 'lb'] > sample.loc[i, 'Adj Close']:
            if book.shift(1).loc[i, 'trade']=='buy':
                book.loc[i, 'trade']='buy'
            else :
                book.loc[i, 'trade']='buy'
```

```
        elif (sample.loc[i, 'ub']>= sample.loc[i, 'Adj Close'] and
              sample.loc[i, 'Adj Close']>=sample.loc[i, 'lb']):
            if book.shift(1).loc[i, 'trade'] == 'buy':
                book.loc[i, 'trade']='buy'
            else :
                book.loc[i, 'trade']=''
    return book
```

```
book = create_trade_book(sample)
book = tradings(sample, book)
book.tail(10)
```

Date	Adj Close	trade
2019-06-11	287.501678	buy
2019-06-12	286.994171	buy
2019-06-13	288.178375	buy
2019-06-14	287.859955	buy
2019-06-17	287.969391	buy
2019-06-18	290.984741	buy
2019-06-19	291.641541	buy
2019-06-20	294.427979	buy
2019-06-21	294.000000	buy
2019-06-24	293.640015	buy

```
def returns(book):
    rtn = 1.0
    book['return']=1
    buy = 0.0
    sell = 0.0
    for i in book.index:
        #long 진입
        if book.loc[i, 'trade']=='buy' and book.shift(1).loc[i, 'trade']=='':
            buy = book.loc[i, 'Adj Close']
            print('진입일 :', i, '\n\t\t long 진입가격:', buy)
```

```
        #long 청산
        elif book.loc[i, 'trade']=='' and book.shift(1).loc[i, 'trade']=='buy':
            sell = book.loc[i, 'Adj Close']
            rtn = (sell-buy)/buy + 1 # 손익계산
            book.loc[i, 'return']= rtn
            print('청산일 :', i, '\n\t\t long 진입가격:', buy,
                  'long 청산가격:', sell, '\n\t\t| return :', round(rtn, 4))

        if book.loc[i, 'trade']=='': # 제로 포지션
            buy = sell = 0.0

    book['acc return'] = book['return'].cumprod()
    acc_rtn = book['acc return'][-1]
    print( 'Accumulated return :', round( acc_rtn, 4))
    return( round(acc_rtn, 4) )
```

```
returns(book)
```

```
진입일 : 2009-01-20 00:00:00
         long 진입가격: 65.089966
청산일 : 2009-03-23 00:00:00
         long 진입가격: 65.089966 long 청산가격: 66.898392
        | return : 1.0278
진입일 : 2010-01-22 00:00:00
         long 진입가격: 90.269791
청산일 : 2010-04-14 00:00:00
         long 진입가격: 90.269791 long 청산가격: 100.584618
        | return : 1.1143
진입일 : 2010-05-04 00:00:00
         long 진입가격: 97.538597
청산일 : 2010-10-13 00:00:00
         long 진입가격: 97.538597 long 청산가격: 98.862717
        | return : 1.0136
진입일 : 2011-03-10 00:00:00
         long 진입가격: 109.513054
청산일 : 2011-04-26 00:00:00
```

long 진입가격: 109.513054 long 청산가격: 114.094101
| return : 1.0418
진입일 : 2011-05-23 00:00:00
long 진입가격: 111.783257
청산일 : 2011-06-30 00:00:00
long 진입가격: 111.783257 long 청산가격: 112.26088
| return : 1.0043
진입일 : 2011-08-02 00:00:00
long 진입가격: 106.748672
청산일 : 2012-02-03 00:00:00
long 진입가격: 106.748672 long 청산가격: 115.768776
| return : 1.0845
진입일 : 2012-04-10 00:00:00
long 진입가격: 117.451515
청산일 : 2012-07-03 00:00:00
long 진입가격: 117.451515 long 청산가격: 119.371857
| return : 1.0164
진입일 : 2012-10-23 00:00:00
long 진입가격: 123.511292
청산일 : 2012-12-18 00:00:00
long 진입가격: 123.511292 long 청산가격: 126.961044
| return : 1.0279
진입일 : 2013-06-05 00:00:00
long 진입가격: 142.477417
청산일 : 2013-07-11 00:00:00
long 진입가격: 142.477417 long 청산가격: 148.711197
| return : 1.0438
진입일 : 2013-08-15 00:00:00
long 진입가격: 147.769791
청산일 : 2013-09-11 00:00:00
long 진입가격: 147.769791 long 청산가격: 150.45195
| return : 1.0182
진입일 : 2013-10-08 00:00:00
long 진입가격: 147.686783
청산일 : 2013-10-17 00:00:00
long 진입가격: 147.686783 long 청산가격: 154.594528

| return : 1.0468
진입일 : 2014-01-24 00:00:00
long 진입가격: 160.521667
청산일 : 2014-04-02 00:00:00
long 진입가격: 160.521667 long 청산가격: 170.233917
| return : 1.0605
진입일 : 2014-04-11 00:00:00
long 진입가격: 163.591492
청산일 : 2014-05-27 00:00:00
long 진입가격: 163.591492 long 청산가격: 172.613312
| return : 1.0551
진입일 : 2014-07-31 00:00:00
long 진입가격: 174.862244
청산일 : 2014-09-18 00:00:00
long 진입가격: 174.862244 long 청산가격: 182.768143
| return : 1.0452
진입일 : 2014-09-25 00:00:00
long 진입가격: 178.636536
청산일 : 2016-04-13 00:00:00
long 진입가격: 178.636536 long 청산가격: 195.218124
| return : 1.0928
진입일 : 2016-06-24 00:00:00
long 진입가격: 191.742584
청산일 : 2016-08-11 00:00:00
long 진입가격: 191.742584 long 청산가격: 206.280853
| return : 1.0758
진입일 : 2016-09-09 00:00:00
long 진입가격: 201.21463
청산일 : 2016-12-07 00:00:00
long 진입가격: 201.21463 long 청산가격: 212.964615
| return : 1.0584
진입일 : 2017-03-21 00:00:00
long 진입가격: 223.89772
청산일 : 2017-04-24 00:00:00
long 진입가격: 223.89772 long 청산가격: 227.193008
| return : 1.0147

```
진입일 : 2017-07-06 00:00:00
         long 진입가격: 231.55455
청산일 : 2017-07-14 00:00:00
         long 진입가격: 231.55455 long 청산가격: 236.377182
        | return : 1.0208
진입일 : 2017-08-10 00:00:00
         long 진입가격: 234.644501
청산일 : 2017-09-11 00:00:00
         long 진입가격: 234.644501 long 청산가격: 239.890701
        | return : 1.0224
진입일 : 2018-02-05 00:00:00
         long 진입가격: 256.626129
청산일 : 2018-05-10 00:00:00
         long 진입가격: 256.626129 long 청산가격: 265.551544
        | return : 1.0348
진입일 : 2018-06-27 00:00:00
         long 진입가격: 264.125763
청산일 : 2018-08-07 00:00:00
         long 진입가격: 264.125763 long 청산가격: 280.040985
        | return : 1.0603
진입일 : 2018-10-10 00:00:00
         long 진입가격: 274.137573
청산일 : 2019-03-18 00:00:00
         long 진입가격: 274.137573 long 청산가격: 280.96347
        | return : 1.0249
진입일 : 2019-05-13 00:00:00
         long 진입가격: 279.50058
Accumulated return : 2.6528

/var/folders/n2/jbh_0_091bx8qgz7j87t2qwc0000gp/T/ipykernel_19470/1068455384.py:15: FutureWarning:
  book.loc[i, 'return']= rtn
/var/folders/n2/jbh_0_091bx8qgz7j87t2qwc0000gp/T/ipykernel_19470/1068455384.py:23: FutureWarning:
  acc_rtn = book['acc return'][-1]

2.6528
```

```
book.tail()
```

Date	Adj Close	trade	return	acc return
2019-06-18	290.984741	buy	1.0	2.652793
2019-06-19	291.641541	buy	1.0	2.652793
2019-06-20	294.427979	buy	1.0	2.652793
2019-06-21	294.000000	buy	1.0	2.652793
2019-06-24	293.640015	buy	1.0	2.652793

```
result = book['return'].resample('A').aggregate(
                [np.prod, lambda x: (x-1).std() * np.sqrt(252) ])
result.columns = ['Return', 'Vol']
result
```

```
/var/folders/n2/jbh_0_091bx8qgz7j87t2qwc0000gp/T/ipykernel_19470/2886312715.py:1: FutureWarning:
  result = book['return'].resample('A').aggregate(
/var/folders/n2/jbh_0_091bx8qgz7j87t2qwc0000gp/T/ipykernel_19470/2886312715.py:1: FutureWarning:
  result = book['return'].resample('A').aggregate(
```

Date	Return	Vol
2009-12-31	1.027783	0.027783
2010-12-31	1.129393	0.115017
2011-12-31	1.046283	0.042032
2012-12-31	1.133016	0.090658
2013-12-31	1.112403	0.066348
2014-12-31	1.169581	0.093156
2015-12-31	1.000000	0.000000
2016-12-31	1.244337	0.132818
2017-12-31	1.059011	0.033853
2018-12-31	1.097132	0.069591
2019-12-31	1.024900	0.036083

```
book[(book['trade']=='buy') &
     (book['trade'].shift(1)=='')].index.year.value_counts().sort_index()
```

```
Date
2009    1
2010    2
2011    3
2012    2
2013    3
2014    4
2016    2
2017    3
2018    3
2019    1
Name: count, dtype: int64
```

```
sample[(book['trade']=='') & (book['trade'].shift(1)=='buy')].head()
```

Date	Adj Close	center	ub	lb
2009-03-23	66.898392	60.247113	66.842427	53.651799
2010-04-14	100.584618	97.853165	100.219620	95.486710
2010-10-13	98.862717	96.240554	98.860432	93.620676
2011-04-26	114.094101	112.277610	114.046010	110.509210
2011-06-30	112.260880	109.184084	111.624576	106.743592

금융공학프로그래밍3, Quiz1

1. Calculate the following

```
import math
```

(1) 근의 공식 코드 작성

```
a=2.0;b=-1.0;c=-15.0;
print(
    (-b+math.sqrt(b**2-4*a*c))/(2*a),
    (-b-math.sqrt(b**2-4*a*c))/(2*a)
)
```

```
3.0 -2.5
```

(2) 정규분포 확률밀도함수 코드 작성

```
mean=2; std=math.sqrt(3); x=1;
print(
    (1/(math.sqrt(2*math.pi)*std))*(math.e**(-((x-mean)**2)/(2*(std)**2)))
)
```

```
0.19496965572274114
```

2. Why the error occurs?

(1) Input error

```
a=input("enter a number:")


a+3
```

input 값은 string타입으로 받게되므로, integer타입인 3과 더하기 연산을 하면 오류 발생, input 값을 받아 integer로 변환하여 integer간 연산을 하거나, 3을 string으로 변환하여 문자열간 연산을 하거나 해야함.

(2) Range error

```
tmp='My String'
tmp[10]
```

tmp는 공백포함 9자리 문자열이므로 index는 0~8까지만 할 수 있고 9부터는 범위 밖이므로 오류 발생.

(3) method error

```
ex1='sample string'
ex2=ex1.upper
ex2[:4]
```

method를 사용할 때, 뒤에 ()를 붙여야 작동함. 붙이지 않을경우 단순히 함수를 호출하는 것.

3. String

```
grade='ABCDF'
```

(1) '+'를 이용해서 'ABCDFFFDCBA' 만들기

```
grade_str=grade+grade[-1]+grade[::-1]
grade_str
```

```
'ABCDFFFDCBA'
```

(2) A의 개수

```
grade_str.count('A')
```

```
2
```

(3) 'FFF'를 반환하는 4가지 다른 표현식

```
print(
    grade_str[4:7],
    grade_str[6:3:-1],
    grade_str[-7:-4],
    grade_str[-5:-8:-1]
)
```

```
FFF FFF FFF FFF
```

(4) 'ABCDAAADCBA'로 수정하기

```
grade_str=grade_str.replace('F','A')
grade_str
```

```
'ABCDAAADCBA'
```

(5) 소문자로 모두 바꾸기

```
grade_str=grade_str.lower()
grade_str
```

```
'abcdaaadcba'
```

4. Why the error occurs?

(1) slicing 반환 형식

```
L=[[1,3,5,7,9],[2,4,6,8,10]]
L[0][1:2]=30
```

list를 원소로 가지는 list에서 slicing하는 경우, output은 list로 나타나게 되는데, list에 integer 30을 할당하려고 하므로 오류가 발생함.

30을 list로 만들어 할당하거나, slicing 대신 indexing을 통해 output을 integer로 만드는 방법을 사용해야 함.

(2) 단일 원소 tuple 표현

```
T=(10,20,30)
T[:2]+(40)
```

tuple에 더하기를 사용하는 경우인데, (40)은 tuple이 아니라 integer이므로 오류가 발생함. 단일 원소를 가지는 tuple을 표현하려면 (40,)으로 써야함.

(3) 변수 할당 labeling vs. copy

```
D=dict(A=10,B=20,C=30)
# Copy method를 써야 복사본이 할당됨
# D2=D.copy()
D2=D
del D2['A']
D['A']
```

D2=D는 D2에 D.copy가 할당되는게 아니라 단순히 D에 D2라는 label만 새로 달아준것임. 따라서, D2에 'A' key를 제거하면 D의 'A' key도 사라지게 되므로 호출시 오류 발생.

(4) Key값은 Immutable 해야함

```
D3={['Park','Male']:30,}
```

Dictionary 타입의 key값은 immutable한 값만 허용되므로 mutable한 list를 사용하면 오류 발생. tuple을 쓰던지 해야함.

(5) slicing&indexing for Dict

```
dict_y={(1,):10,(2,):20,(3,):30,(4,):40}
dict_y[(1,)]
```

Dictionary은 순서가 없음. slicing이 불가하며 indexing도 key값으로만 접근 가능.

5. List

```
days=['Mon','Tues','Wed','Thur','Fri',['Sat','Sun']]
```

(1) indexing, slicing

```
print(
    [days[5]],
    days[-1::-2],
    days[5][0],
    sep='\n'
)
```

```
[['Sat', 'Sun']]
[['Sat', 'Sun'], 'Thur', 'Tues']
Sat
```

(2) '+' 활용

```
days2=[days[0:5]]+days[5][0:1]+days[5][1:]
days2
```

```
[['Mon', 'Tues', 'Wed', 'Thur', 'Fri'], 'Sat', 'Sun']
```

(3) remove method

```
days2[0].remove('Wed')
days2[0].remove('Fri')
days2
```

```
[['Mon', 'Tues', 'Thur'], 'Sat', 'Sun']
```

(4) insert method

```
days2[0].insert(2,'W')
days2
```

```
[['Mon', 'Tues', 'W', 'Thur'], 'Sat', 'Sun']
```

6. list2

```
Nums=[1,5,2,7,3,6,4]
```

(1) append method

```
Nums.append(7)
Nums
```

```
[1, 5, 2, 7, 3, 6, 4, 7]
```

(2) sorting list

```
Nums.sort()
Nums.reverse()
Nums
```

```
[7, 7, 6, 5, 4, 3, 2, 1]
```

(3) slicing and replacing

```
Nums[::2]=['a','a','a','a']
Nums
```

```
['a', 7, 'a', 5, 'a', 3, 'a', 1]
```

7. tuple

```
price=(180,130,110,160,140,170)
```

(1) sorting tuple via translate to list

```
price_list=list(price)
price_list.sort()
price=tuple(price_list)
price
```

```
(110, 130, 140, 160, 170, 180)
```

(2) basic operation

```
170 in price
```

```
True
```

(3) concatenating tuple

```
price=price[0:4]+(0,)*3+price[len(price)-1:]
price
```

```
(110, 130, 140, 160, 0, 0, 0, 180)
```

금융공학프로그래밍3, Quiz2

1. Dictionary

```
icecream=dict(cheery=[12,100],cookies=[18,200],greentea=[15,140],mango=[10,200])
```

(1)

Change the price of 'cookies' as 13

```
icecream['cookies'][0]=13
icecream['cookies']
```

```
[13, 200]
```

(2)

Add the following information to the dictionary 'icecream'. - 'Strawberry' is priced at 12 and inventory is 170. - 'pistachio' is priced at 15 and inventory is 100.

```
update_dict=dict(Strawberry=[12,170],pistachio=[15,100])
icecream.update(update_dict)
icecream
```

```
{'cheery': [12, 100],
 'cookies': [13, 200],
 'greentea': [15, 140],
 'mango': [10, 200],
 'Strawberry': [12, 170],
 'pistachio': [15, 100]}
```

2. Dictionary2

```
d1=dict(key1=1,key2=3,key3=2,key10=7)
d2=dict(key3=1,key2=2,key7=2,key5=4,key1=7,key9=8,key0=7)
```

(1)

Select the largest two values of the dictionary 'd2'.

```
d2_list=list(d2.values())
d2_list.sort()
d2_list.reverse()
d2_list[0:2]
```

```
[8, 7]
```

(2)

Create a set of values which are the unique values of 'd2'.

```
set(d2.values())
```

```
{1, 2, 4, 7, 8}
```

(3)

Write code that returns True if keys of 'd1' are all contained in the key of 'd2' and False otherwise.

```
set(d1.keys()) <= set(d2.keys())
```

```
False
```

(4)

Create a set of values which are common in both 'd1' and 'd2'

```
set(d1.values())&set(d2.values())
```

```
{1, 2, 7}
```

3. if statement

For any arbitrary string object A, if the first character of A is '#', we want it to be replaced with '*'. Otherwise, we want to add '*' as the first character of A. Write appropriate code using an if statement for this

```
A="#abcde"

if A[0]=="#":
    A="*"+A[1:]
else:
    A="*"+A

A
```

```
'*abcde'
```

4. for loop

Calculate the following by using 'for' loop : $\sum_{i=1}^{10} i^4$

```
result=int()

for i in range(1,11):
    result+=i**4

result
```

```
25333
```

금융공학프로그래밍3, Quiz3

1. Calculate the following by using 'while' loop.

$$\sum_{i\geq 1, i^4<1000}^{10} i^4$$

```
i=1
sum_i4=0
while i>=1 and i**4<1000 and i<=10:
    sum_i4+=i**4
    i+=1
sum_i4
```

979

2. List comprehension

```
x=[1,2,3,4]
y=[5,6,7,8]
```

(1) Cartesian product

```
z=list([i,j] for i in x for j in y)
z
```

```
[[1, 5],
 [1, 6],
 [1, 7],
```

```
 [1, 8],
 [2, 5],
 [2, 6],
 [2, 7],
 [2, 8],
 [3, 5],
 [3, 6],
 [3, 7],
 [3, 8],
 [4, 5],
 [4, 6],
 [4, 7],
 [4, 8]]
```

(2) Cartesian product with condition

```
z=list([i,j] for i in x for j in y if i+j>=8)
z
```

```
[[1, 7],
 [1, 8],
 [2, 6],
 [2, 7],
 [2, 8],
 [3, 5],
 [3, 6],
 [3, 7],
 [3, 8],
 [4, 5],
 [4, 6],
 [4, 7],
 [4, 8]]
```

3. Function

```
def grading (score):
    if type(score)!=int and type(score)!=float:
        print("Score must be integer or float type.")
    elif score>100 or score<0:
        print("Score must be a number between 0 and 100!!")
    elif score>90:
        print("Grade is A !")
    elif score>80:
        print("Grade is B !")
    elif score>70:
        print("Grade is C !")
    elif score>60:
        print("Grade is D !")
    else:
        print("Grade is F !")
```

```
grading(score=75)
```

```
Grade is C !
```

```
grading(score=-5)
```

```
Score must be a number between 0 and 100!!
```

4. Explain why the error occurs

(1) positional argument와 keyword argument를 혼용하는 경우, positional argument가 먼저 와야합니다. keyword argument를 먼저 사용하는 경우 오류가 발생합니다.

```
def infoprint( name, age, gender):
    print(name, 'is', age, 'years old', gender, '.')

infoprint (name='Kim', 30, 'male')
```

(2) 'Kim'은 positional argument로 name에 할당, 'male'은 keyword로 gender에 할당되고 age는 할당값이 없어서 오류가 발생하였습니다. 함수 정의시 age는 default값을 정의하지 않았으므로 반드시 값을 할당해야 합니다.

```
def infoprint( name, age, gender):
    print(name, 'is', age, 'years old', gender, '.')

infoprint ( 'Kim', gender='male' )
```

(3) 함수 myfactorial 정의시 변수 fac을 사용하는데, 해당 변수는 함수 안에서 정의된바 없으므로 함수 밖에서 정의된 global variable인 fac을 가져와서 사용하게 됩니다. 그러나, 이런 경우 global variable의 읽기만 가능하고 쓰기는 불가능하므로 for문 내에 fac을 수정하는 것은 허용되지 않습니다.

```
fac = 1
def myfactorial(n):
    for i in range(n):
        fac *= i+1
    return fac

myfactorial(n=5)
```

금융공학프로그래밍3, Quiz4

1. Find out which one of following is wrong?

```
game/
__init__.py
sound/
    __init__.py
    echo.py
    wav.py
graphic/
    __init__.py
    screen.py
    render.py
play/
    __init__.py
    run.py
    test.py
```

① >>> from game.sound import echo

```
>>> echo.echo_test()
```

Right

② >>> from game.sound.echo import echo_test

```
>>> echo_test()
```

Right

③ >>> import game

```
>>> game.sound.echo.echo_test()
```

FALSE!! : ___init___.py가 empty이므로 game package import시 하단의 하부구조가 자동으로 import 되지 않습니다. 따라서, eco.py 내의 함수 eco_test()를 호출하기 위해서는 eco.py 또는 eco_test()를 직접 호출하여야 합니다.

④ 〉〉〉 import game.sound.echo

```
>>> game.sound.echo.echo_test()
```

Right

2.

np.arrange[5]를 이용하여 array구조를 만들면, dtype이 integer인 array가 생성됩니다.

이중 하나의 object를 float타입으로 수정하더라도, dtype은 integer로 고정되므로 입력한 float타입의 데이터가 integer로 수정되어 할당됩니다.

3.

indexing방식으로 호출하였으므로 1-dimensional이 됩니다. 2-dimensional을 유지하기 위해서는 slicing방식으로 호출하여야 합니다.

(ex: rmat[0:1])

4.

np.hstack 함수는 array를 병합하는 함수로, np.concatenate에 axis=1 옵션과 동일한 함수입니다.

병합하는 두 array간 dimension이 동일해야하는데, a=2, b=1이므로 오류가 발생합니다.

다른 dimension간 병합을 수행하려면 np.colmun_stack함수를 사용하여야 합니다.

5.

a는 2-dim. 3X5 array, b는 0-dim. 3 array입니다.

b[np.newaxis,:]는 0-dim인 b에 x축을 추가하여 1X3 array로 만들어주게 됩니다.

3X5와 1X3간에 덧셈이 성립하지 않으므로 오류가 발생합니다.

한편, b[:,np.newaxis]를 사용하면 3X1 array가 되므로, 덧셈이 가능합니다.

금융공학프로그래밍3, Quiz5

Problem 1

1. Create a 2-dimensional ndarray object 'Arr' with the code below and answer the following.

```
import numpy as np
np.random.seed(123)
Arr=np.random.randn(8,10)
```

(1) Write an expression to calculate column sum of Arr.

```
np.sum(Arr,axis=0)
```

```
array([-3.68648899, -3.20797062,  1.89628963, -4.31697028, -1.68652947,
        3.31014718,  3.61439765,  2.82609217,  1.68894737,  0.84241575])
```

(2) Write an expression that finds the position (row and column index) of an element greater than 2 in Arr

```
np.where(Arr>2)
```

```
(array([1, 1, 4, 4]), array([6, 7, 6, 9]))
```

Problem 2

2. Write 3 different expressions to create a pandas Series object 'S' displayed as follows.

   〉〉〉 S

   a 0

   b 1

c 2

d 3

dtype: int64

```
import pandas as pd
S=pd.Series([0,1,2,3],index=['a','b','c','d'])
S
```

```
a    0
b    1
c    2
d    3
dtype: int64
```

```
S=pd.Series(dict(a=0,b=1,c=2,d=3))
S
```

```
a    0
b    1
c    2
d    3
dtype: int64
```

```
S=pd.Series(np.arange(4),index=['a','b','c','d'])
S
```

```
a    0
b    1
c    2
d    3
dtype: int64
```

Problem 3

3. Write 4 different expressions to select the 2nd and the 3rd elements of Snew.

```
Snew = pd.Series({'a': 1, 'b':4, 'c':2, 'd':3} )
```

```
Snew[1:3]
```

```
b    4
c    2
dtype: int64
```

```
Snew.iloc[[1,2]]
```

```
b    4
c    2
dtype: int64
```

```
Snew['b':'c']
```

```
b    4
c    2
dtype: int64
```

```
Snew[['b','c']]
```

```
b    4
c    2
dtype: int64
```

Problem 4~5

```
np.random.seed(123)
DF = pd.DataFrame(np.random.randn(6,7),
                  columns=list('abcdefg'),
                  index=[3,2,4,5,1,0])
DF
```

	a	b	c	d	e	f	g
3	-1.085631	0.997345	0.282978	-1.506295	-0.578600	1.651437	-2.426679
2	-0.428913	1.265936	-0.866740	-0.678886	-0.094709	1.491390	-0.638902
4	-0.443982	-0.434351	2.205930	2.186786	1.004054	0.386186	0.737369
5	1.490732	-0.935834	1.175829	-1.253881	-0.637752	0.907105	-1.428681
1	-0.140069	-0.861755	-0.255619	-2.798589	-1.771533	-0.699877	0.927462
0	-0.173636	0.002846	0.688223	-0.879536	0.283627	-0.805367	-1.727669

4. Fill in the appropriate expression in square brackets so you can select columns 'c' and 'e'.

```
DF.iloc[:,2:5:2]
```

	c	e
3	0.282978	-0.578600
2	-0.866740	-0.094709
4	2.205930	1.004054
5	1.175829	-0.637752
1	-0.255619	-1.771533
0	0.688223	0.283627

```
DF[['c','e']]
```

	c	e
3	0.282978	-0.578600
2	-0.866740	-0.094709
4	2.205930	1.004054
5	1.175829	-0.637752
1	-0.255619	-1.771533
0	0.688223	0.283627

5. Fill in the appropriate expression in square brackets so that you can select rows with negative values for column 'c'.

```
DF.loc[DF.c<0,:]
```

	a	b	c	d	e	f	g
2	-0.428913	1.265936	-0.866740	-0.678886	-0.094709	1.491390	-0.638902
1	-0.140069	-0.861755	-0.255619	-2.798589	-1.771533	-0.699877	0.927462

```
DF[DF.c<0]
```

	a	b	c	d	e	f	g
2	-0.428913	1.265936	-0.866740	-0.678886	-0.094709	1.491390	-0.638902
1	-0.140069	-0.861755	-0.255619	-2.798589	-1.771533	-0.699877	0.927462

Python Midterm

20249132 김형환

Problem 1.

1. 채권의 가격과 듀레이션

- 각 시점 $t = 1,\dots,n$ 에서 C_t의 현금흐름이 발생함.
- 연간 쿠폰 지급 횟수가 f 번이며 ($f = 1,2,4$인 경우만 고려), 쿠폰 이자율은 연 $r\%$ 임.
- 쿠폰은 매 시점마다 지급되며, 액면가격은 PV 원은 만기 시점에 쿠폰과 함께 지급됨.
- 채권의 만기는 M (년)임.
- 시장의 만기수익률이 연 $y\%$임.
- 이 경우 해당 채권 가격과 듀레이션은 다음 식으로 계산될 수 있음

  채권 가격 : $P = \sum_{t=1}^{n} \frac{C_t}{(1+(y/100)/f)^{t'}}$ 듀레이션 : $D = \sum_{t=1}^{n} t \cdot \frac{\frac{C_t}{(1+(y/100)/f)^t}}{P}$

(1)

5개의 인자 facevalue= , couprate= , yield= , maturity= 으로 입력하고, frequency= 에 맞게 “annual”, “semi-annual”, “quarterly” 중 하나의 문자 값으로 입력하면, 위 식에 따라 계산된 채권가격과 듀레이션을 아래와 같은 형태의 튜플로 반환하는 함수 bondftn을 작성하여라.

```
def bondftn(pv,r,y,m,f="annual",rnd=3):
    if type(pv)!=float and type(pv)!=int : return print("error: pv should be number!!!")
    if type(r)!=float and type(r)!=int: return print("error: r should be number!!!")
    if type(y)!=float and type(y)!=int: return print("error: y should be number!!!")
```

```
    if type(m)!=int: return print("error: M should be integer!!!")
    if f=="annual": fre=1
    elif f=="semi-annual": fre=2
    elif f=="quarterly": fre=4
    else: return print("error: f can be 'annual', 'semi-annual', 'quarterly'")
    n=fre*m
    coupon=pv*r/100/fre
    price=0.0
    value_duration=0.0
    for i in range(n):
        if i==n-1: cf_i=pv+coupon
        else: cf_i=coupon
        pv_cf_i=cf_i/((1+(y/100/fre))**(i+1))
        tw_pv_cf_i=((i+1)/fre)*cf_i/(1+(y/100/fre))**(i+1)
        price+=pv_cf_i
        value_duration+=tw_pv_cf_i
    duration=value_duration/price
    result=(round(price,rnd),round(duration,rnd))
    return result
```

```
bondftn(pv=100,r=5,y=4.5,m=2,f="quarterly")
```

```
(100.951, 1.916)
```

(2)

만기 M이 5, 4, 3, 2, 1인 각 경우에 대해, 쿠폰이자율이 5%, 4%, 3%, 2%, 1% 인 각각에 대하여 만기수익률이 10%에 11%로 상승함에 따라 채권 가격의 변화율이 몇 %인지를 모두 계산한 뒤, 아래와 같은 중첩된 딕셔너리 result_dict으로 저장하여라.

단, 액면가는 100, 쿠폰 지급 횟수는 연간 1회라고 하자.

```
result_dict=dict()
for i in range(5):
    m_i=5-i
    result_dict['M='+str(m_i)]=dict()
```

```
    for j in range(5):
        r_j=5-j
        price_before=bondftn(pv=100,r=r_j,y=10,m=m_i,f="annual")[0]
        price_after=bondftn(pv=100,r=r_j,y=11,m=m_i,f="annual")[0]
        price_return=(price_after-price_before)/price_before
        result_dict['M='+str(m_i)][str(r_j)+'%']=round(price_return,5)
result_dict
```

```
{'M=5': {'5%': -0.03974,
  '4%': -0.04046,
  '3%': -0.04126,
  '2%': -0.04215,
  '1%': -0.04314},
 'M=4': {'5%': -0.03287,
  '4%': -0.03332,
  '3%': -0.03381,
  '2%': -0.03434,
  '1%': -0.03491},
 'M=3': {'5%': -0.02544,
  '4%': -0.02568,
  '3%': -0.02593,
  '2%': -0.02619,
  '1%': -0.02648},
 'M=2': {'5%': -0.01749,
  '4%': -0.01758,
  '3%': -0.01765,
  '2%': -0.01776,
  '1%': -0.01784},
 'M=1': {'5%': -0.00901,
  '4%': -0.009,
  '3%': -0.009,
  '2%': -0.009,
  '1%': -0.00901}}
```

```
result_dict['M=5']['5%']
```

```
-0.03974
```

(3)

만기 M이 5, 4, 3, 2, 1인 각 경우에 대해, 쿠폰이자율이 5%, 4%, 3%, 2%, 1% 인 각각에 대하여 만기수익률이 10%로 주어졌을 때의 듀레이션을 (1)에서의 bondftn을 활용하여 모두 계산하여라.

또한 이를 (2)에서와 동일한 구조를 가지는 (중첩된) 딕셔너리 result_dict_dur에 저장하여라.

즉 아래와 같은 방식으로 값을 추출할 수 있어야 한다. 단, 여기서도 액면가는 100, 쿠폰 지급 횟수는 연간 1회라고 하자.

```
result_dict_dur=dict()
for i in range(5):
    m_i=5-i
    result_dict_dur['M='+str(m_i)]=dict()
    for j in range(5):
        r_j=5-j
        result_dict_dur['M='+str(m_i)][str(r_j)+'%']=bondftn(pv=100,r=r_j,
                                                             y=10,m=m_i,
                                                             f="annual",rnd=5)[1]
result_dict_dur
```

```
{'M=5': {'5%': 4.48786,
  '4%': 4.57019,
  '3%': 4.66101,
  '2%': 4.76171,
  '1%': 4.874},
 'M=4': {'5%': 3.6951,
  '4%': 3.74653,
  '3%': 3.80216,
  '2%': 3.8625,
  '1%': 3.9282},
 'M=3': {'5%': 2.84899,
  '4%': 2.87566,
  '3%': 2.90394,
  '2%': 2.93397,
  '1%': 2.96593},
```

```
 'M=2': {'5%': 1.95023,
  '4%': 1.95941,
  '3%': 1.96896,
  '2%': 1.97889,
  '1%': 1.98923},
 'M=1': {'5%': 1.0, '4%': 1.0, '3%': 1.0, '2%': 1.0, '1%': 1.0}}
```

```
result_dict_dur['M=5']['4%']
```

```
4.57019
```

Problem 2.

2. 자동차 보험회사에 관한 몬테카를로 시뮬레이션

어느 자동차 보험회사의 보험료 수입과 보험금 지급은 다음의 프로세스로 설명된다고 하자.

-
- 보험금의 연간 청구 건수는 평균이 100인 포아송 분포를 가짐.
- 각 청구 건수 별 청구금액은 모양이 2, 척도가 1/2 (즉, 평균은 1이고 분산은 1/2)인 감마분포로부터 나오며, 청구가 이루어지자 마자 보험금은 지급된다고 가정.
- 각 청구 건수 별 청구 발생시점은 0에서 1(년) 사이의 균일분포를 따른다고 가정.
- 보험회사의 보험료 수입은 연간 150의 비율로 1년 균등히 보험료를 받게 됨. 즉 어떤 해에 t시점 $(0 \le t \le 1)$까지 받게 되는 전체 보험료는 150t임.

(1)

어떤 1년 동안 발생하는 모든 청구들의 시간과 금액을 모의실험해 보자.

0에서 시작하여1년 동안 보험회사의 잔고를 계산한 뒤 balance라는 리스트로 저장하여라.

단, balance의 첫번째 값은 0이며, 보험 청구가 발생하는 시점( )에만, 해당 시점의 balance (그 시점까지 받은 보험료 수익(150)을 더하고, 그 시점에서 청구로 지급되는 보험금을 빼 주는 방식)를 계산하여 순서대로 저장할 것.

Answer

먼저, 보험금의 연간 청구건수가 평균이 100인 포아송분포를 따르므로 $pmf,\ P(X=x)=\frac{\lambda^x e^{-\lambda}}{x!},\ \lambda=100$입니다.

다음으로, 보험금의 청구금액은 평균은 1, 분산은 1/2인 감마분포를 따르므로 $pdf,\ f(x)=x^{k-1}\frac{e^{-x/\theta}}{\theta^k\Gamma(k)},\ k\theta=1,\ k\theta^2=1/2$입니다. 즉, $k=2,\ \theta=1/2$입니다.

각 청구금액은 즉시 지급되며, 발생시점은 0~1 사이의 uniform distribution을 따르므로 $pdx.\ f(x)=\frac{1}{1-0}=1$입니다.

이제, 1년간 발생하는 보험손익의 모의실험에 대한 청구건수(포아송분포)를 N번이라고 하고, 청구금액(감마분포) 및 청구시점(균등분포)를 각각 g_i와 t_i for $1,...,N$라고 하면, $t=t_i$ 시점의 balance는 아래와 같이 계산됩니다.

$$balance_{t_i}=150t_i-\sum_{k=1}^{i}g_k$$

또한, 기말시점(t=1)의 최종 balance는 $150-\sum_{all\ k}g_k$가 됩니다.

```
# 포아송, 감마, 균등분포에 다한 모의실험은 numpy의 random module 활용
import numpy as np
from numpy import random as rd



rd.seed(0) # seed를 설정하여 통제
N=rd.poisson(lam=100,size=1)
N=N[0]
t=rd.uniform(low=0.0,high=1.0,size=N)
t=np.sort(t)
g=rd.gamma(shape=2.,scale=0.5,size=N)
balance=[0] # balance의 초기값은 0
for i in range(N):
    balance.append(round(t[i]*150-sum(g[:i+1]),4))
balance.append(round(150-sum(g),4)) # balance의 기말값
```

```
print(balance[:5],balance[N-5:],sep="\n")
```

```
[0, -1.9755, -2.1182, -2.7092, -3.9799]
[44.5214, 47.2399, 48.4395, 48.2655, 47.6871, 48.74, 50.4839]
```

(2)

위 모의실험을 10000회 반복하여 다음에 대한 확률을 추정해 보자.

- 보험회사가 최종적으로 가지게 되는 balance의 기대값(모의실험을 10000회 반복하였을 때 각 모의실험마다 계산된 balance의 최종값들의 평균으로 추정)은 얼마인가?
- 보험회사의 balance가 1년 중 한번 이상 -5 이하로 떨어질 확률(10000회의 모의실험 중에서 balance가 -5이하로 떨어진 적이 있었던 경우의 비율로 추정)은 얼마인가?

Answer

평균은 약 49.86, -5이하로 떨어질 확률은 약 7.05%입니다.

```
last_balance=list()
minus5_balance=0
for k in range(10000):
    rd.seed(k)
    N=rd.poisson(lam=100,size=1)
    N=N[0]
    t=rd.uniform(low=0.0,high=1.0,size=N)
    t=np.sort(t)
    g=rd.gamma(shape=2.,scale=0.5,size=N)
    balance=[0]
    for i in range(N):
        balance.append(round(t[i]*150-sum(g[:i+1]),4))
    last_balance.append(round(150-sum(g),4))
    balance.sort()
    if balance[0]<=-5:minus5_balance+=1
print(sum(last_balance)/10000,
      minus5_balance/10000,sep="\n")
```

```
49.864134890000095
0.0705
```

Python Final Exam

Problem 1.

1. Markowitz의 평균 분산 모델에 따르면 k개의 자산으로 구성된 포트폴리오에서 주어진 기대수익률에 대해 분산이 가장 작아지게 하는 각 자산 별 투자비중은 아래와 같이 도출할 수 있다.

   ① $X_1, \dots, X_k$은 각 자산 별 수익률을 나타내는 유한한 분산 값을 가지는 확률변수로, $X_1, \dots, X_k$의 공분산행렬을 $Q_{k\times k}$, 기대값 벡터는 $r_{k\times 1} = (r_1, \dots, r_k) = (E[X_1], \dots, E[X_k])$라고 하자.

   ② 각 자산 별 투자비중을 $\mathrm{w}_{k\times 1} = (w_1, \dots, w_k)$ (단, $\vec{1}^T w = \sum w_i = 1$, $\vec{1}_{k\times 1} = (1,1,\dots,1)$)로 하는 포트폴리오 P를 가정할 때, 포트폴리오 P 의 기대수익률은 $\mu_p = r^T w = \sum w_i r_i$, 분산은 $\sigma_p^2 = w^T Q w$가 된다.

   ③ 이 경우 다음을 만족하는 w를 찾는 최적화 문제를 고려하자.

$$\min\{w^T Q w \mid w \in \mathcal{R}^k, \quad \vec{1}^T w = 1, \quad r^T w = \mu_p\}$$

   즉 포트폴리오의 기대수익률 μ_p이 원하는 수준에서 주어진 상태에서 포트폴리오의 분산 $w^T Q w$를 최소로 하기 위해 필요한 각 자산 별 가중치 $w = (w_1, \dots, w_k)$를 찾는 문제이다.

   ④ 이는 선형제약식을 가진 2차 함수에 대한 최적화 문제로, 라그랑지 이론에 따르면 아래와 같은 선형방정식계를 이용하여 풀 수 있다.

$$\begin{bmatrix} Q & \vec{1} & r \\ \vec{1}^T & 0 & 0 \\ r^T & 0 & 0 \end{bmatrix} \begin{bmatrix} w \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ \mu_p \end{bmatrix}$$

   (1) 이상의 내용을 수행하기 위한 파이썬 함수 MVportfolio를 작성하여라. 함수의 인자는 k개의 자산 별 가격을 m일 동안 일별로 수집한 $\mathrm{m}\times\mathrm{k}$의 데이터프레임 asset과, 원하는 포트폴리오 기대수익률 (μ_p)인 mu_p이고, 함수를 실행한 결과 k개의 자산별 가중치 $w = (w_1, \dots, w_k)$와 포트폴리오 분산 σ_p^2 값이 반환되어야 한다. 단, 각 자산 별 일별 수익률은 log(해당일가격/전일가격)으로 구하여라.

   (2) 5개의 자산 가격을 일별로 수집한 'it.csv'의 자료에 (1)에서 작성한 MVportfolio 함수를 적용하여, μ_p가 -0.001에서 0.0010사이의 값을 가질 때 최적의 투자비중으로 계산된 분산 σ_p^2를 도출한 뒤, 이를 이용하여 효율적인 투자선 (가로축은 σ_p^2, 세로축은 μ_p인 선그림)을 도출하여라.

Answer 1.

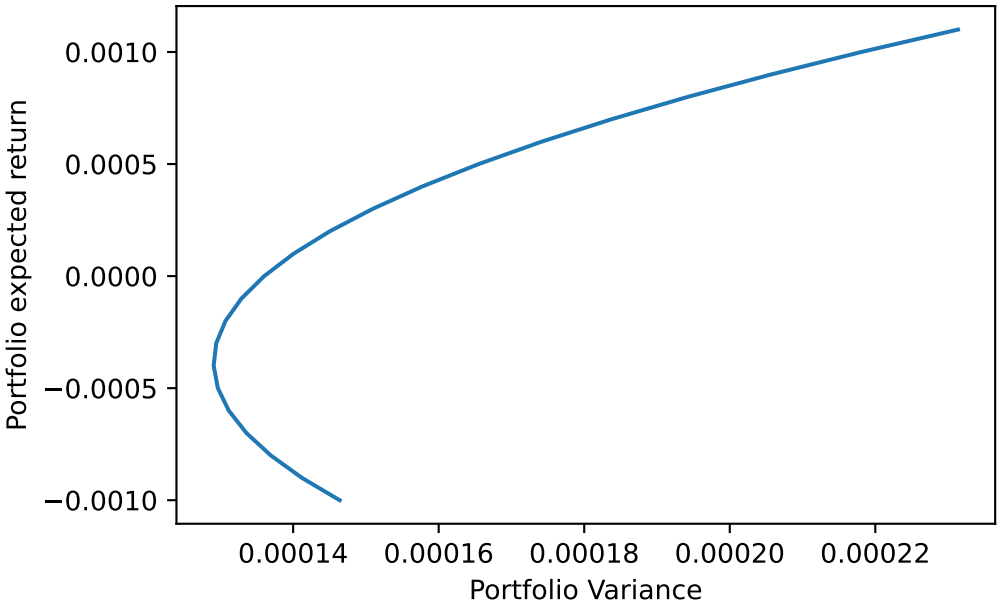
```
import pandas as pd
import numpy as np

def MVportfolio(asset, mu_p):
    # Calculate log returns (first column is date.)
    return_asset = np.log(asset.iloc[:, 1:]).diff()

    # Calculate expected log returns
    expect_asset = return_asset.mean()
    expect_asset_np = np.array(expect_asset)

    # Calculate covariance matrix
    cov_asset = return_asset.cov()
    cov_asset_np = np.array(cov_asset)

    # Use lagrange multipliers method for optimazation
    num_assets = return_asset.shape[1]
    u = np.ones(num_assets)

    a = np.block([
        [cov_asset_np, u[:, np.newaxis], expect_asset_np[:, np.newaxis]],
        [u[np.newaxis, :], np.zeros(1), np.zeros(1)],
        [expect_asset_np[np.newaxis, :], np.zeros(1), np.zeros(1)]
    ])

    b = np.concatenate([np.zeros(num_assets), [1], [mu_p]])

    lagrange = np.linalg.solve(a, b)

    # Optimal weights
    result = lagrange[:num_assets]

    # Portfolio variance
```

```
    Var_p = np.dot(result, np.dot(cov_asset, result))

    return result, Var_p

# Test
it = pd.read_csv('it.csv')
MVportfolio(asset = it, mu_p = -0.001)
```

```
(array([ 0.16186582,  0.25684189, -0.35114871,  0.44970549,  0.48273551]),
 0.00014638968239383282)
```

```
import matplotlib.pyplot as plot

# Range of mu_p
mu_p_values = np.arange(-0.001, 0.0011, 0.0001)

# Looping
portfolio_variances = []
for mu_p in mu_p_values:
    tmp = MVportfolio(asset=it, mu_p=mu_p)
    Var_p = tmp[1]
    portfolio_variances.append(Var_p)

# Plotting
plot.plot(portfolio_variances, mu_p_values)
plot.xlabel('Portfolio Variance')
plot.ylabel('Portfolio expected return')
```

```
Text(0, 0.5, 'Portfolio expected return')
```


0.0010
0.0005
0.0000
−0.0005
−0.0010
Portfolio expected return
0.00014
0.00016
0.00018
0.00020
0.00022
Portfolio Variance

Problem 2.

2. 다음은 12개월 모멘텀을 이용하여 10개의 종목을 선택하는 문제이다.

   (1) 'price.csv' 파일을 불러온 뒤, 'date' 열이 DatetimeIndex 자료형이 되도록 변경한 뒤, 이를 Index로 하는 pd.DataFrame 객체 price를 생성하여라.

   (2) price에서 2019년도에 해당하는 자료만 선택하여 price_sub 로 저장하여라.

   (3) price_sub에서 각 종목 별(열 별)로 일별 수익률(=1+변화율)에 대한 누적곱으로 누적수익률을 구한 뒤 cum_ret라는 pd.Series 객체로 저장하여라.

   (4) cum_ret에서 누적수익률이 높은 순서로 10개 종목을 출력한 뒤, 가장 누적수익률이 높은 종목에 대한 2019년 price의 시계열 그림을 출력하여라.

   (5) price_sub에서 각 종목 별(열 별)로 일별 변화율에 대한 표준편차에 252의 제곱근을 곱하여, 각 종목 별 연율화 변동성을 구한 뒤 이를 std라는 pd.Series 객체로 저장하여라.

   (6) std에서 std가 0인 경우와 NaN인 경우를 제외하여라.

   (7) 종목 별 cum_ret와 std의 비율로 정의되는 위험조정 수익률 노게 (= cum_ret / std)를 pd.Series 객체로 저장하여라.

   (8) 위험조정 수익률 shrp에 NaN이 포함된 경우는 이를 shrp의 최소값으로 대체하여라.

   (9) 위험조정 수익률 shrp의 값이 큰 순서대로 10개 종목을 선택하여라.

   (10) 선택된 10개 종목을 index로 하고, 10개 종목에 대한 누적수익률(cum_ret 값), 변동성(std 값), 위험조정수익률(shrp 값)을 column으로 하는 pd.DataFrame 객체 final_result를 생성한 뒤 출력하여라.

Answer 2.

```
# 1. 'price.csv'의 'date'를 DatetimeIndex로 하는 pd.DataFrame price 생성
price = pd.read_csv('price.csv')
price['date'] = pd.to_datetime(price['date'])
price = price.set_index('date')
price
```

date	X005930	X000660	X051910	X207940	X035420	X035720	X005380	X006400	X068270	X0
2018-05-23	51800	95300.0	343500.0	399500.0	136994	22580	144500.0	204500	237286	336
2018-05-24	51400	94600.0	345000.0	418000.0	136193	22681	140000.0	204500	243737	327
2018-05-25	52700	95200.0	352500.0	430500.0	137194	22480	139000.0	207000	253413	322
2018-05-28	52300	94500.0	352000.0	429000.0	136193	21677	138500.0	209000	249726	323
2018-05-29	51300	94800.0	343500.0	433000.0	133789	21677	140000.0	211500	249266	316
...	...	...	...	...	...	...	...	...	...	...
2020-08-31	54000	75100.0	740000.0	778000.0	322500	81691	176500.0	452500	291656	424
2020-09-01	54200	75200.0	743000.0	781000.0	324500	80587	180000.0	454500	292638	431
2020-09-02	54400	75500.0	742000.0	770000.0	332500	82695	176000.0	452000	295093	422
2020-09-03	56400	78700.0	768000.0	779000.0	339000	82293	176500.0	452500	294111	427
2020-09-04	55600	78700.0	743000.0	773000.0	328500	80688	172500.0	439000	288710	421

```
# 2. price에서 2019년만 선택해서 price_sub로 저장
price_sub=price.loc['2019']
price_sub
```

date	X005930	X000660	X051910	X207940	X035420	X035720	X005380	X006400	X068270	X0
2019-01-02	38750	60600.0	337000.0	374000.0	118000	20473	114000.0	210500	201192	328
2019-01-03	37600	57700.0	328000.0	376500.0	122000	20573	116500.0	203000	197440	328
2019-01-04	37450	58300.0	330500.0	381000.0	125500	20774	119500.0	201000	206350	348
2019-01-07	38750	58700.0	343000.0	388000.0	131500	20673	120500.0	217000	202599	342
2019-01-08	38100	59200.0	349500.0	386000.0	130000	20051	119500.0	218500	200254	338
...	...	...	...	...	...	...	...	...	...	...
2019-12-23	55500	94600.0	313000.0	422500.0	184000	29806	122500.0	228000	174929	447
2019-12-24	55000	93800.0	312500.0	417000.0	182000	29405	122000.0	225000	173521	447
2019-12-26	55400	94800.0	308000.0	413500.0	180500	29706	122500.0	222500	177743	451
2019-12-27	56500	96000.0	310500.0	429500.0	183500	30810	120500.0	233000	180689	443
2019-12-30	55800	94100.0	317500.0	433000.0	186500	30810	120500.0	236000	177743	443

```
# 3. 열 별로 직전 대비 변동률을 구하고, 누적 곱하여 cum_ret 생성
returns = price_sub.pct_change()
cum_ret = (1 + returns).prod(axis=0) - 1
# 기간누적수익률 산출을 위해 (-1) 추가. 안하는 경우 (-) 수익률인 주식의 샤프비율 고평가됨
cum_ret
```

```
/var/folders/n2/jbh_0_091bx8qgz7j87t2qwc0000gp/T/ipykernel_2239/3098040869.py:2: FutureWarning:

The default fill_method='pad' in DataFrame.pct_change is deprecated and will be removed in a futu

X005930    0.440000
X000660    0.552805
X051910   -0.057864
X207940    0.157754
X035420    0.580508
             ...
X121890    0.273713
X114570   -0.818388
X347140    0.000000
X080440   -0.644615
X158310   -0.090741
Length: 2203, dtype: float64
```

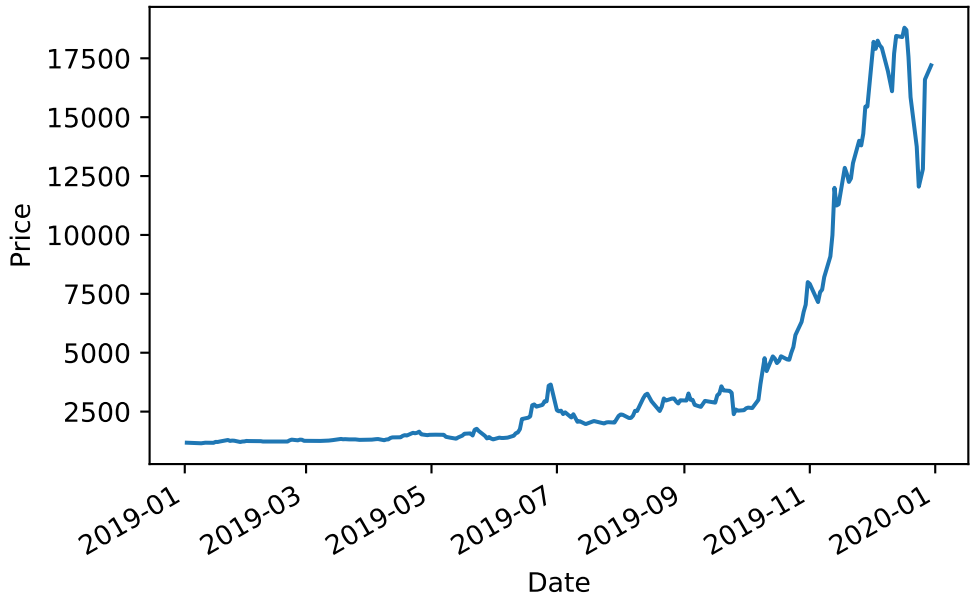
```
# 4. cum_ret에서 높은 순서로 10개 출력 및 가장 높은 종목의 시계열 그림 출력
top10_cum_ret = cum_ret.nlargest(10)
top10_cum_ret
```

```
X088290    13.576271
X101360     7.400000
X078130     4.155172
X139670     4.049401
X032500     3.443478
X009190     2.992042
X138080     2.719443
X336370     2.693285
```

```
X033250     2.536524
X214150     2.498146
dtype: float64
```

```
top_index = top10_cum_ret.index[0]
price_sub[top_index].plot()
plot.xlabel('Date')
plot.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```




```
# 5. 열 별로 직전 대비 변동률의 표준편차를 계산하여 std 생성
std = returns.std(axis=0) * np.sqrt(252)
std
```

```
X005930    0.230925
X000660    0.353011
X051910    0.248067
X207940    0.367346
X035420    0.328026
              ...
X121890    0.941212
```

```
X114570    0.709787
X347140         NaN
X080440    0.995452
X158310    1.000060
Length: 2203, dtype: float64
```

```
# 6. std에서 0이거나 NaN인 경우 제외
std = std[(std != 0) & (~std.isna())]
std
```

```
X005930    0.230925
X000660    0.353011
X051910    0.248067
X207940    0.367346
X035420    0.328026
              ...
X058420    0.680404
X121890    0.941212
X114570    0.709787
X080440    0.995452
X158310    1.000060
Length: 2097, dtype: float64
```

```
# 7. 종목 별 Sharpe ratio=cum_ret/std를 구하여 shrp 생성
shrp = cum_ret / std
shrp
```

```
X000020   -0.412952
X000040   -1.106285
X000050   -0.795542
X000060   -0.651532
X000070   -0.593173
              ...
X361390         NaN
X361610         NaN
X363280         NaN
```

```
X375500         NaN
X378850         NaN
Length: 2203, dtype: float64
```

```python
# 8. shrp에 NaN이 포함된 경우 shar의 최소값으로 대체
shrp.fillna(shrp.min(), inplace=True)
shrp
```

```
X000020   -0.412952
X000040   -1.106285
X000050   -0.795542
X000060   -0.651532
X000070   -0.593173
             ...
X361390   -2.164228
X361610   -2.164228
X363280   -2.164228
X375500   -2.164228
X378850   -2.164228
Length: 2203, dtype: float64
```

```python
# 9. shrp를 값이 큰 순서로 10개 종목 출력
top10_shrp = shrp.nlargest(10)
top10_shrp
```

```
X088290    12.317314
X032500     5.582321
X101360     5.364799
X139670     4.570944
X138080     4.033614
X078130     3.760220
X214150     3.711147
X036540     3.707145
X097520     3.624222
X000990     3.552103
dtype: float64
```

```
# 10. 위 10개 종목에 대한 cum_ret, std, shrp 값을 각 열로 하는 pd.DataFrame  final_result 생성
final_result = pd.DataFrame({
    'cum_ret': cum_ret[top10_shrp.index],
    'std': std[top10_shrp.index],
    'shrp': shrp[top10_shrp.index]
})
final_result
```

	cum_ret	std	shrp
X088290	13.576271	1.102210	12.317314
X032500	3.443478	0.616854	5.582321
X101360	7.400000	1.379362	5.364799
X139670	4.049401	0.885900	4.570944
X138080	2.719443	0.674195	4.033614
X078130	4.155172	1.105034	3.760220
X214150	2.498146	0.673147	3.711147
X036540	2.142336	0.577894	3.707145
X097520	1.903846	0.525312	3.624222
X000990	1.628571	0.458481	3.552103

Problem 3.

3. 유러피안 콜 옵션 가격 몬테카를로 시뮬레이션 다음 ECallSimul_1은 유러피안 콜 옵션 가격을 몬테카를로 시뮬레이션으로 도출하기 위한 함수를 작성한 것이다. (산출식, 변수명 등은 ch3 강의에서 다룬 내용과 동일함)

```
def ECallSimul_1(S0, K, T, r, sigma, M, I=250000):
        import math
        import random
        S = []
        dt = T/M
        for i in range(I):
                path = []
                for t in range (M+1):
                    if t == 0 :
                        path.append(S0)
                    else:
                        z=random.gauss(0., 1.)
                        St = path[t-1] * math.exp( (r-0.5*sigma**2)*dt + sigma * math.sqrt(dt)*z )
                        path.append(St)
                S.append(path)
        sum_val = 0.0
         for path in S:
                sum_val += max(path[-1]-K, 0)
        C0 = math.exp(-r*T)*sum_val/I
        return ( round(C0, 3) )
```

(1) ECallSimul_1 함수의 body에서 가능한 모든 부분을 Numpy의 기능을 활용하는 것으로 수 정한 뒤 ECallSimul_2라는 함수로 저장하여라.

(2) S0=100., K=105., T=1., r=0.05, sigma=0.2, M=50, I=250000인 경우에 대하여 두 함수 ECallSimaul_1과 ECallSimaul_2의 결과가 유사한 지 확인해 보고, 두 함수의 연산시간을 비교하여라. (※ Jupyter Notebook에서 %time이라는 매직 명령어 뒤에 실행하고자 하는 코 드를 작성하면 그 코드를 실행하는데 소요된 CPU time을 출력할 수 있음.)

Answer 3.

```
def ECallSimul_2(S0, K, T, r, sigma, M, I=250000):
    import numpy as np
    S = np.zeros((I,M+1))
    S[:,0] = S0
    dt = T/M
    z = np.random.standard_normal(size=(I,M))
```

```
    for t in range(1,M+1):
        S[:,t] = S[:,t-1]*np.exp((r-0.5*sigma**2)*dt + sigma*np.sqrt(dt)*z[:,t-1])
    sum_val = np.maximum(S[:,-1]-K, 0)
    C0 = np.exp(-r*T)*np.mean(sum_val)
    return ( round(C0, 3) )
```

```
%time ECallSimul_1(S0=100.,K=105.,T=1.,r=0.05,sigma=0.2,M=50,I=250000)
```

```
CPU times: user 11.1 s, sys: 97.4 ms, total: 11.2 s
Wall time: 10.8 s

8.019
```

```
%time ECallSimul_2(S0=100.,K=105.,T=1.,r=0.05,sigma=0.2,M=50,I=250000)
```

```
CPU times: user 699 ms, sys: 39.3 ms, total: 739 ms
Wall time: 739 ms

8.025
```

Part II

채권분석&기간구조('24봄)

채권분석 과제1

1. CB

(***Question***)

전환사채를 액면 F 만큼 매입할 때 전환가가 P라면, 전환주식수는 F/P 로 주어진다. 특정 전환사채를 액면 1억원 만큼 시장에서 매입하였다. 매입가는 1억 1천만원이다. 전환가격은 5만원으로 주어져 있다.

해당 주식의 시장가 가 6만원일 때 전환하여 매도하였다면 투자자의 수익률은무엇인가? % 로 쓰되, 소수점 둘째 자리까지만 쓰시오.

(***Answer***) 9.09%

$$= \frac{1}{5} = 2,000$$

$$= 2000 \times 60,000 \ = 1 \ 2$$

$$= 1 \ 1 \quad , \quad = \frac{1.2}{1.1} - 1 \approx 9.09\%$$

2. FRN

(***Question***)

정부가 2년 만기 변동금리채권을 발행한다고 가정하자. 기준금리는 만기 1년 현물 이자율 (spot rate 또는 1년 만기 무이표채 국채의 이자율) 이고, 해당 변동금리채권의 이자는 1년마다 계산하여 지급한다, 즉 1년 말에 지급하고 2년 만기시 원금과 함께 지급한다.

현재 1년 spot rate 을 $r_{1,0}$, 1년 후 spot rate 을 $r_{1,1}$ 이라 하면 받게 되는 총 금액은 액면 100 당 $100r_{1,0}$ (1년 시점) 과 $100(1 + r_{1,1})$ (2년 시점)이다. 물론 $r_{1,1}$ 은 현재 시점에서는 알 수 없는 확률변수이다.

액면이 100 인 위의 2년 만기 변동금리채권의 현재 시점의 공정한 가격을 구하시오. 투자자는 시장에서 1년 만기 무이표채 국채를 현재 시점이던 1년 후 시점이던 자유롭게 거래할 수 있다.

(***Answer***) 100

상기 FRN은 1년 만기 무이표 국채의 이자율만큼의 쿠폰을 지급하고 있다. 이러한 액면 100인 FRN의 현금흐름은 현재 1년 만기 무이표국채에 $100r_{1,0}$만큼 투자하고, 1년 뒤 만기시점에 다시 1년 만기 무이표국채에 $100 + r_{1,1}$투자함으로써 복제할 수 있다.

즉, 해당 FRN은 무위험이자율 만큼의 연수익률을 보장하는 채권으로서 쿠폰=할인율이므로 **공정가격은 100(par-value)**에 형성되어야 한다.

> Foward rate를 이용한 풀이
>
> $r_{1,1}$은 확률변수이므로, FRN의 2년뒤 현금흐름 $100(1+r_{1,1})$을 현재가치로 환산한다면 그 공정가치는 $\frac{100(1+E[r_{1,1}])}{(1+r_{2,0})^2}$으로 표현할 수 있다.
> 한편, 1년뒤 1년 spot-rate의 기댓값은 foward rate으로 볼 수 있고, 현재시점의 2년 spot-rate는 $(1+r_{1,0})(1+f_{1,1})$로 표현할 수 있다. 따라서, FRN의 공정가격은 다음과 같다.
>
> $$FRN\ price = \frac{100r_{1.0}}{1+r_{1,0}} + \frac{100(1+E[r_{1,1}])}{1+r_{2,0}} = \frac{100r_{1.0}}{1+r_{1,0}} + \frac{100(1+f_{1,1})}{(1+r_{1,0})(1+f_{1,1})} = \frac{100(1+r_{1,0})}{1+r_{1,0}} = 100$$

3. YTM

(***Question***)

잔존만기가 3년이고 연 8% 이자를 6개월마다 제공하는 채권을 가정하자. A 투자자는 이 채권에 관심이 있고 거래하는 딜러는 100달러 액면당 92.5067 달러를 요구한다.

이때 이 채권의 수익률은 얼마인가? % 로 표시하되 소수점은 반올림하여 정수로 나타내시오.

(***Answer***) 11%

채권의 수익률 y와 가격에 관한 산식은 다음과 같다.

$$92.5067 = \sum_{k=1}^{6} \frac{8/2}{(1+y/2)^k} + \frac{100}{(1+y/2)^6}$$

이를 엑셀 해찾기 기능으로 근사한 y값은 약 11%이다.

4. Continuous compounding

(***Question***) 10.71%

위의 문제는 compounding frequency 를 연 1회로 가정하고 있다. 이것은 이산복리법이다. 이 frequency가 무한이 되는 연속복리법에 따르면 1달러 투자가 1년 후에 e^y 가 된다. 여기서 y 는 연속복리이자율이다.

위 현금흐름 수익률을 연속복리이자율로 표현하시오. %로 나타내되 소수점 둘째 자리까지 쓰시오.

(***Answer***)

(3)의 현금흐름을 연속복리로 표현하면 다음과 같다.

$$92.5067 = \sum_{k=1}^{6} \frac{8}{2} e^{-0.5ky} + 100e^{-3y}$$

이를 엑셀 해찾기 기능으로 근사한 y값은 10.71%이다.

> ℹ 수익률 관계를 이용한 풀이
>
> (3)의 수식과 비교하여 단순히 $e^{0.5y_{continuous}} = (1 + y_{annual}/2) = 1.055$로 표현할 수 있고 값은 10.71%로 오차범위 내에서 동일하다.

5. Arbitrage

(***Question***)

아래와 같은 세 개의 채권이 시장에서 거래되고 있다. 트레이더 B 는 이를 보고 곧 차익거래의 기회가 있음을 파악하고 전략적으로 거래하여 이익을 얻었다. 어떤 전략이 가능할지 본인의 생각을 기술하시오.

- 채권 a: 만기 1년 무이표채, 액면 100 당 가격 90
- 채권 b: 만기 2년 무이표채, 액면 100 당 가격 80
- 채권 c: 만기 2년 이표채, coupon rate 10%, 매년 말 이자지급, 액면 100 당 가격 100

(***Answer***) 채권a:채권b=1:11로 투자하여 채권c를 만들고 시장에 매도

채권a와 채권b의 수익률은 zero-rates이며, 이를 y_a, y_b라고 할 때, 다음과 같이 쓸 수 있다.

$$\frac{100}{1+y_a} = 90,\ \frac{100}{(1+y_b)^2} = 80 \Rightarrow \frac{1}{1+y_a} = 0.9,\ \frac{1}{(1+y_b)^2} = 0.8$$

이를 이용하여 채권c의 현금흐름을 평가하면, $\frac{10}{1+y_a} + \frac{110}{(1+y_b)^2} = 9 + 88 = 97$

즉, **액면 100인 채권c의 공정가격은 97이나, 현재 100에 거래되고 있으므로 3만큼 고평가**되어있는 상태이다. 따라서 트레이더B가 **채권a:채권b=1:11 비중으로 매수한다면 액면이 10인 채권c의 현금흐름을 정확하게 복제**할 수 있는데, 이를 이용하면 무위험 차익거래를 할 수 있다.

예를 들어,

(1) 액면 100억원 채권c를 시장에 매도(공매도 or 발행)하여 100억원의 자금을 조달

(2) 액면 10억원 채권a를 9억원에, 액면 110억원 채권b를 88억원에 매수

(3) 1년뒤 채권a의 원금 10억을 받아 채권c의 이표 10억원을 갚고, 2년뒤 채권b의 원금 110억을 받아 채권c의 원금+이표 110억원을 갚을 수 있으므로 현재 시점에서 미래현금흐름은 0임(리스크=0)

(4) 한편, 현재 100억원을 빌려 97억원을 사용하였으므로 3억원의 무위험차익 실현 (또는, 남은 3억원을 채권b에 투자하여 현재시점의 투입자본없이 2년뒤 3.75억의 수익 실현 가능)

채권분석 과제2

1. Callable bond

(***Question***)

만기 2년, 액면이자율 10% (연 1회 지급), 액면가 100 인 수의상환채권 callable bond 를 생각하자. 이 채권의 상환액 call price 는 100, 상환기간은 1년부터 (이자지급 이후) 만기까지 이다. 즉, 채권발행자 (기채자, 채무자) 는 상환기간 동안 원하는 때 100으로 채권을 상환할 수 있다. 1년이 지난 시점에서 만기수익률을 10.05% 라 가정하자.

채권의 가격 P_0 를 구하시오.

시중금리변화에 따라 이 만기수익률이 즉각적으로 10bp 상승할 때의 가격 P_+ 를 구하시오.

그리고 만기수익률이 즉각적으로 10bp 하락할 때의 가격 P_- 를 구하시오.

채권발행자는 채권의 가치가 100을 넘어서면 상환할 인센티브가 있음을 고려하라. 이들 세 값을 바탕으로 우리는 수의상환채권에 대하여 유효한 듀레이션을 $-\frac{1}{P_0} \times \frac{P_+ - P_-}{20bp}$ 로 구할 수 있다. 소수점 네째자리까지 리포트하시오.

(***Answer***)

1년이 지난 시점에서 수의상환채권의 잔여현금흐름은 만기 일시납 110이며, YTM은 10.05%이므로 가격 P_0는 약 99.9546입니다.

```
P0=110/(1+0.1005)
P0
```

```
[1] 99.95457
```

만기수익률이 10bp씩 즉각적으로 움직일 때, 수의상환부를 고려하지 않은 채권가격은 아래와 같습니다.

```
P0_up=110/(1+0.1005+0.001)
P0_down=110/(1+0.1005-0.001)
c(P0_up,P0_down)
```

```
[1]  99.86382 100.04548
```

그러나, 수의상환채권의 가격은 call price를 초과할 수 없으므로 $P_+ = 99.8638,\ P_- = 100$입니다.

위 가격으로부터 산출한 잔존만기 1년의 수의상환채권의 **듀레이션은 약 0.6812**입니다.

$$D_{callable} - \frac{1}{P_0} \times \frac{P_+ - P_-}{20bp} = 0.6812$$

```
duration=(-1/P0)*(P0_up-100)/0.002
round(duration,4)
```

```
[1] 0.6812
```

2. Spot rate

(***Question***)

hw2.xlsx 에는 Treasury par yield (% 값) 가 일부 주어져 있다. 이것은 2024년 3월 21일 기준 US Treasury Par Yield Curve 이다.

선형보간법을 이용하여 빈 곳을 채우고, 1년에 2회 이자를 지급하는 기준으로 하여 spot rate 을 구하시오. 단, 0.5년 과 1년 채권은 무이표 채권이다. 제출할 때에는 % 로 소수점 둘째 자리까지 보이시오.

(***Answer***)

먼저, 선형보간법을 통해 기간별 Par-yield를 구하도록 하겠습니다.

```
library(tidyverse)
par_yield <- tibble("year"=seq(0.5,10,0.5),
                    "period"=c(1:20),
                    "paryield"=c(5.36,5.01,NA,4.62,NA,
                                 4.42,NA,NA,NA,4.26,
                                NA,NA,NA,4.28,NA,
                                NA,NA,NA,NA,4.27))
```

```
par_yield <- par_yield %>%
  mutate(interpolation=if_else(paryield %>% is.na(),
                               (lag(paryield)+lead(paryield))/2,
```

```
                                paryield)) %>%
  mutate(interpolation=if_else(interpolation %>% is.na(),
                                (lag(lag(interpolation))+lead(lead(interpolation)))/2,
                                interpolation)) %>%
  mutate(interpolation=if_else(interpolation %>% is.na(),
                                (lag(interpolation)+lead(interpolation))/2,
                                interpolation))
par_yield <- par_yield %>%
  mutate(interpolation=if_else(interpolation %>% is.na(),
                                ((20-period)*par_yield$interpolation[14]
                                +(period-14)*par_yield$interpolation[20])/6,
                                interpolation))

par_yield %>% as.data.frame()
```

	year	period	paryield	interpolation
1	0.5	1	5.36	5.360000
2	1.0	2	5.01	5.010000
3	1.5	3	NA	4.815000
4	2.0	4	4.62	4.620000
5	2.5	5	NA	4.520000
6	3.0	6	4.42	4.420000
7	3.5	7	NA	4.380000
8	4.0	8	NA	4.340000
9	4.5	9	NA	4.300000
10	5.0	10	4.26	4.260000
11	5.5	11	NA	4.265000
12	6.0	12	NA	4.270000
13	6.5	13	NA	4.275000
14	7.0	14	4.28	4.280000
15	7.5	15	NA	4.278333
16	8.0	16	NA	4.276667
17	8.5	17	NA	4.275000
18	9.0	18	NA	4.273333
19	9.5	19	NA	4.271667

```
20 10.0      20      4.27       4.270000
```

다음으로, Par-yield와 6개월/1년 spot rate를 통해 기간별 spot rate를 boot-strapping 방식으로 산출하도록 하겠습니다.

Par-yield는 채권의 현재가격을 액면가격으로 만들어주는 coupon-rate입니다. 즉, 액면가 100의 채권이 연간 par-yield 만큼 쿠폰을 semi-annually 지급한다면 다음과 같이 쓸 수 있습니다.

$$100 = \sum_{k=1}^{n-1} \frac{Par/2}{(1+Spot_k/2)^k} + \frac{100+Par/2}{(1+Spot_n/2)^n}$$

$$\Rightarrow \frac{1}{(1+Spot_n/2)^n} = \frac{1}{100+Par/2}(100 - \sum_{k=1}^{n-1} \frac{Par/2}{(1+Spot_k/2)^k})$$

$$\Rightarrow Spot_n = ((\frac{100+Par/2}{100-Sum\ of\ Coupon\ PV})^{\frac{1}{n}} - 1) \times 2$$

위 수식을 R을 이용해서 산출하고, 소수점 둘째자리까지 반올림한 결과는 아래와 같습니다.

```r
spot <- par_yield %>%
  select(year,period,interpolation) %>%
  mutate(coupon=interpolation/2,
         spot=if_else(period<=2,interpolation,NA)) %>%
  mutate(dc=1/(1+spot/100/2)^period) %>%
  mutate(cum_dc=cumsum(dc)) %>%
  mutate(sum_pvc=coupon*(cum_dc-dc))

for (i in 3:20){
spot$sum_pvc[i] <- spot$coupon[i]*spot$cum_dc[i-1]
spot$dc[i] <- (100-spot$sum_pvc[i])/(100+spot$coupon[i])
spot$spot[i] <- {{(1/spot$dc[i])^(1/spot$period[i])-1}*200}
spot$cum_dc[i] <- spot$cum_dc[i-1]+spot$dc[i]
}
spot <- spot %>% mutate(spot2=spot %>% round(2))
spot %>% select(year,period,interpolation,spot2)
```

```
# A tibble: 20 x 4
    year period interpolation spot2
```

	<dbl>	<int>	<dbl>	<dbl>
1	0.5	1	5.36	5.36
2	1	2	5.01	5.01
3	1.5	3	4.81	4.81
4	2	4	4.62	4.61
5	2.5	5	4.52	4.51
6	3	6	4.42	4.4
7	3.5	7	4.38	4.36
8	4	8	4.34	4.32
9	4.5	9	4.3	4.28
10	5	10	4.26	4.24
11	5.5	11	4.26	4.25
12	6	12	4.27	4.25
13	6.5	13	4.28	4.26
14	7	14	4.28	4.27
15	7.5	15	4.28	4.27
16	8	16	4.28	4.26
17	8.5	17	4.27	4.26
18	9	18	4.27	4.26
19	9.5	19	4.27	4.26
20	10	20	4.27	4.26

3. Forward rate

(***Question***)

위의 문제에서 구한 현물이자율을 이용하여 선도이자율을 구하여라. 이 때 선도이자율은 6개월 구간마다 주어진다. 물론 0 부터 6개월까지의 선도이자율은 6개월 현물이자율과 같다. 선도이자율도 % 로 소수점 둘째 자리까지 구하시오.

(***Answer***)

semi-annually 현물이자율에서 6개월 선도이자율에 대한 수식은 다음과 같습니다.

$$(1 + Spot_{n-1})^{n-1}(1 + Forward_{n-1,n}) = (1 + Spot_n)^n$$

$$\Rightarrow Forward_{n-1,n} = \frac{(1 + Spot_n)^n}{(1 + Spot_{n-1})^{n-1}} - 1$$

위 수식과 (2)의 **소수점 둘째자리까지 반올림한 Spot rate를 이용**하여 산출한 선도이자율은 아래와 같습니다.

```
forward <- spot %>%
  select(year,period,spot,spot2) %>%
  mutate(spot_yield=(1+spot/100/2)^period,
         spot_yield2=(1+spot2/100/2)^period) %>%
  mutate(forward={{spot_yield/lag(spot_yield)-1}*200} %>% round(2),
         forward2={{spot_yield2/lag(spot_yield2)-1}*200} %>% round(2))
forward %>% select(year,period,spot,forward2)
```

```
# A tibble: 20 x 4
    year period  spot forward2
   <dbl>  <int> <dbl>    <dbl>
 1   0.5      1  5.36    NA
 2   1        2  5.01     4.66
 3   1.5      3  4.81     4.41
 4   2        4  4.61     4.01
 5   2.5      5  4.51     4.11
 6   3        6  4.40     3.85
 7   3.5      7  4.36     4.12
 8   4        8  4.32     4.04
 9   4.5      9  4.28     3.96
10   5       10  4.24     3.88
11   5.5     11  4.25     4.35
12   6       12  4.25     4.25
13   6.5     13  4.26     4.38
14   7       14  4.27     4.4
15   7.5     15  4.27     4.27
16   8       16  4.26     4.11
17   8.5     17  4.26     4.26
18   9       18  4.26     4.26
19   9.5     19  4.26     4.26
20  10       20  4.26     4.26
```

Spot rate 소수점 자리 문제

Forward rate를 산출할 때, Spot rate를 소수점 두번째 자리까지 반올림하지 않는다면, Spot rate의 오차는 미미하더라도 Forward rate 산출에 영향을 주게 되며, 오차는 기간이 길어질수록 커집니다.

```
forward %>% select(year,period,forward2, forward) %>% filter(year>=8)
```

```
# A tibble: 5 x 4
   year period forward2 forward
  <dbl>  <int>    <dbl>   <dbl>
1   8        16     4.11    4.25
2   8.5      17     4.26    4.24
3   9        18     4.26    4.24
4   9.5      19     4.26    4.23
5  10        20     4.26    4.23
```

4. Continuous compounding - Forward rate

(***Question***)

챕터2 에서 우리는 선도이자율이 구간별로 주어질 수 있음을 배웠다.

Semi-annual 베이스로 현물이자율을 정의했을 때, r_1 은 1년 현물이자율, $r_{1.5}$ 는 1년 6개월 현물이자율을 나타낸다. 그리고 6개월부터 1년까지 기간에 대한 선도이자율은 $(1+r_1/2)^2 = (1+r_{0.5}/2)(1+f_{[0.5,1]}/2)$ 를 만족한다.

현물이자율을 연속복리법으로 정의했을 때, 1년 현물이자율을 r_1 이라 놓으면 시작점에서 1 의 투자는 e_1^r 으로 성장한다. 마찬가지로 시간 t 에 대한 현물이자율을 r_t 라 놓으면 시작점에서의 1의 투자는 e^{tr_t} 로 성장한다. 두 시점 a 〈 b 에 대하여 연속복리법에 의한 현물이자율 r_a, r_b 가 주어져 있다. 투자자는 (a,b) 구간에 적용되는 선도거래를 체결하려고 한다.

이 때 중요한 것은 선도이자율인데, 무차익원리에 의한 선도이자율 $f(a,b)$ 를 a, b, r_a, r_b 를 이용하여 나타내시오. 물론 연속복리법을 적용한다.

(***Answer***)

연속복리법 하에서, $f(a,b), a, b, r_a, r_b$ 간에는 아래와 같은 식이 성립해야 합니다.

$$e^{a\times r_a} \times e^{(b-a)\times f(a,b)} = e^{b\times r_b}$$

이를 통해 $f(a,b)$를 표현하면,

$$\Rightarrow e^{ar_a+(b-a)f(a,b)} = e^{br_b}$$

$$ar_a + (b-a)f(a,b) = br_b \;\Rightarrow\; f(a,b) = \frac{br_b - ar_a}{b-a}$$

5. Pricing convention - full price

(***Question***)

2024년 3월 15일 호가가 액면 100 당 103-22+ 로 주어진 미 국채가 있다. 이 채권의 만기는 2029년 10월 15일이고 표면금리는 6.125% 이다. 이자는 매 4월 15일, 10월 15일 지급될 때, full price를 구하시오. 소수점 여섯째자리까지 반올림하여 나타내시오.

(***Answer***)

미국 국채의 가격표시방법에 따라, 103-22+의 가격표시는 아래와 같습니다.

$$103 - 22+ = 103 + \frac{22}{32} + \frac{1}{64} = 103.703125$$

```
options(digits=10)
clean_price <- 103+22/32+1/64
clean_price
```

```
[1] 103.703125
```

해당 가격은 호가가격이므로, clean price=103.703125입니다.

경과이자(accrued interest)를 가산해야 Full price를 산출할 수 있습니다.

경과이자는 $\frac{\times}{2} \times$ 을 통해 구할 수 있습니다.

경과일은 152일, 이자지급기간은 183일이고, 산출된 경과이자는 약 2.543716입니다.

따라서, full price = clean(quoted) price + accrued interest = **106.246841**입니다.

```
library(lubridate)


accrued_date <- ymd(20240315)-ymd(20231015)
interest_period <- ymd(20240415)-ymd(20231015)
```

```r
accrued_interest={(100*0.06125)/2}*(as.integer(accrued_date)/as.integer(interest_period))
full_price=round(clean_price+accrued_interest,6)

paste(accrued_date,interest_period,accrued_interest,full_price,sep=" / ")
```

```
[1] "152 / 183 / 2.54371584699454 / 106.246841"
```

채권분석 과제3

Question 1

투자자 A 는 표면금리 7% 미 국채를 액면 1,000 usd 구입하였다. 이 국채의 호가는 101-25+ 이다. 결제일은 2024년 2월 5일이고, 쿠폰 지급일은 매년 4월 15일, 10월 15일이다. 지불한 invoice price(full price)를 센트까지만 구하시오.

Answer

먼저, 액면 100당 Quote price(clean price)=101-25+=101+25/32+1/64=101.7969입니다.

다음으로 경과이자(accrued interest)는 "(지난이자지급일로부터 경과일)/이자지급주기*이자"로 계산할 수 있습니다. (real/real 기반)

경과일은 113일(2023.10.15~2024.02.05), 이자지급주기는 183일(2023-10-15~2024.04.15)이므로, 경과이자는 약 21.6120입니다.

따라서, **Full price는 약 1039.58**입니다.

```
library(tidyverse)
```

```
clean <- (101+25/32+1/64)*10
day_interest <- as.integer(ymd(20240415)-ymd(20231015))
day_accrued <- as.integer(ymd(20240205)-ymd(20231015))
accrued <- 35*day_accrued/day_interest
full <- clean+accrued
paste(clean,accrued %>% round(6),full %>% round(6),sep=" / ")
```

```
[1] "1017.96875 / 21.612022 / 1039.580772"
```

Question 2

시장에서 현물이자율이 1년 4%, 2년 5% 라 가정하자. 그리고 1년 현물이자율의 연 변동(log rate)의 표준편차는 0.005 라 하자. 1년 후 1년 현물이자율의 시나리오를 r_u, r_d 라 놓으면 시장 정보를반영하는 이 두 값을 구하시오.

Answer

1년 후 현물이자율이 상승하였을 때를 r_u, 하락하였을 때를 r_d로 나눌 수 있으며 현물이자율의 로그수익률의 연표준편차가 0.005이므로, 아래와 같이 표현할 수 있습니다.

$$r_u = r_d e^{2\sigma} = r_d e^{0.01}$$

한편, 2년 현물이자율에 투자한 zero-coupon bond와 1년 zero-coupon 및 1년 후에 다시 zero-coupon(forward)에 투자한 포트폴리오의 가치는 같아야 하므로,

$$\frac{100}{1.05^2} = \frac{100}{(1.04)(1+E(r_1))} = 0.5 \times \frac{100}{1.04(1+r_u)} + 0.5 \times \frac{100}{1.04(1+r_d)}$$

이분법을 통해 값을 추정하면, $r_u \approx 6.04\%$. $r_d \approx 5.98\%$

```
initial_d <- 0.01; initial_u <- 0.10
r_d <- (initial_u+initial_d)/2; r_u <- r_d*exp(0.01)

while(abs(0.5*100/(1.04*(1+r_u))+0.5*100/(1.04*(1+r_d))-100/1.05^2)>0.001){
  if(0.5*100/(1.04*(1+r_u))+0.5*100/(1.04*(1+r_d))-100/1.05^2>0){initial_d <- r_d}
  if(0.5*100/(1.04*(1+r_u))+0.5*100/(1.04*(1+r_d))-100/1.05^2<=0){initial_u <- r_d}
  r_d <- (initial_u+initial_d)/2
  r_u <- r_d*exp(0.01)}
paste(r_u %>% round(4),r_d %>% round(4))
```

```
[1] "0.0604 0.0598"
```

Question 3

2년 만기이고 표면금리가 8% 인 회사채를 가정하자. 표면금리의 지급은 연 1회이다. 그리고 이 회사채는 1년 시점에서 액면가 100 으로 call 할 수 있는 수의상환채이다. 신용위험 때문에 시장 에서는 이 회사채를 위에서 구한 금리모형에 스프레드를 더하여 그 가치를 계산한다.

시장가가 101.9 라 할 때, static spread 와 option-adjusted spread 를 구하여라

Answer

먼저, 수의상환부회사채의 시장가격 101.9와 문제 2의 zero-rate를 통해 **Static spread를 산출하면 아래와 같습니다.**

$$101.9 = \frac{8}{(1.04+ss)} + \frac{108}{(1.05+ss)^2} \Rightarrow ss \approx 1.988\%$$

```
market_price <- 101.9; initial_d <- 0.001; initial_u <- 0.1
ss <- (initial_u+initial_d)/2

while(abs(8/(1.04+ss)+108/(1.05+ss)^2-market_price)>0.00001){
  if(8/(1.04+ss)+108/(1.05+ss)^2-market_price>0){initial_d <- ss}
  if(8/(1.04+ss)+108/(1.05+ss)^2-market_price<=0){initial_u <- ss}
  ss <- (initial_u+initial_d)/2}
ss
```

```
[1] 0.01988313
```

다음으로, 문제 2의 금리모형을 이용하여 OAS를 구해보겠습니다.

먼저, OAS를 적용한 할인율은 zero rate + OAS로 표현할 수 있습니다.

문제 2에 따라 1년 zero-rate는 4%, 상승시 $r_u = 0.0604$, 하락시 $r_d = 0.0598$이므로,

$$r_1^* = 0.04 + OAS,\ r_u^* = 0.0604 + OAS,\ r_d^* = 0.0598 + OAS$$

수의상환회사채는 1년 후 가격이 액면가보다 높으면 100에 상환되므로, $r_u^*,\ r_d^*$가 8%보다 높은 경우 100에 상환됩니다.

따라서, 수의상환회사채의 금리 상승/하락에 따른 가치평가는 아래와 같습니다.

$$101.9 = \frac{1}{1+r_1^*}[8 + \frac{1}{2}(min\{\frac{108}{1+r_u^*}, 100\} + min\{\frac{108}{1+r_d^*}, 100\})]$$

$$= \frac{1}{1.04+OAS}[8 + \frac{1}{2}(min\{\frac{108}{1.0604+OAS}, 100\} + min\{\frac{108}{1.0598+OAS}, 100\})]$$

$$\Rightarrow OAS \approx 1.978\%$$

```r
initial_d <- 0.001; initial_u <- 0.1
OAS <- (initial_u+initial_d)/2
while(abs((1/(1.04+OAS))*(8+0.5*(min(108/(1+r_u+OAS),100)+min(108/(1+r_d+OAS),100)))-market_pr
  if((1/(1.04+OAS))*(8+0.5*(min(108/(1+r_u+OAS),100)+min(108/(1+r_d+OAS),100)))-market_price>0
  if((1/(1.04+OAS))*(8+0.5*(min(108/(1+r_u+OAS),100)+min(108/(1+r_d+OAS),100)))-market_price<=
  OAS <- (initial_u+initial_d)/2
}
OAS
```

```
[1] 0.01978343
```

! Difference in each spreads

Nominal spread, Static spread, Options adjusted spread

1. Nominal spread : 단순히 국채 vs. 회사채의 YTM을 비교
2. Static spread : zero rate + α를 계산, 기간구조를 반영하고있으나 향후 금리변동은 반영하지 않고 단순 현금흐름만 비교하므로, Zero-volatility spread라고도 함
3. Options adjusted spread : 수의상환채의 옵션에 따른 향후 현금흐름 변동가능성을 고려하여 산출. 미래 금리변동으로 인해 옵션이 권리행사되는 경우를 감안하여 산출

Question 4

PSA 방법에 따르면 PSA 스피드에 따라 월별 CPR 이 결정되고, 이는 다시 SMM 을 결정한다. 강의노트의 PSA 방법론을 참조하여 다음의 SMM 표를 완성하시오.

월	100 PSA	60 PSA	230 PSA
1			
20			
200			

Answer

PSA란 연조기상환율(CPR:Conditional Prepayment Rate)이 매달 0.2% 증가하며, 6%까지 증가한 후 만기까지 지속되는 형태의 조기상환모형입니다.

100 PSA가 기준이며, 50 PSA인 경우 매달 0.1%씩 증가 등 비율이 달라지게됩니다.

다음으로, 월별조기상환율(SMM:Single Monthly Mortality rate)는 CPR을 월환산한 것으로, $1 - SMM = (1 - CPR)^{\frac{1}{12}}$ 입니다.

이에 따라 산출한 (1-SMM) 표는 아래와 같습니다.

월	100 PSA	60 PSA	230 PSA
1	$(1 - 0.2\%)^{\frac{1}{12}}$	$(1 - 0.2\% \times 0.6)^{\frac{1}{12}}$	$(1 - 0.2\% \times 2.3)^{\frac{1}{12}}$
20	$(1 - 4\%)^{\frac{1}{12}}$	$(1 - 4\% \times 0.6)^{\frac{1}{12}}$	$(1 - 4\% \times 2.3)^{\frac{1}{12}}$
200	$(1 - 6\%)^{\frac{1}{12}}$	$(1 - 6\% \times 0.6)^{\frac{1}{12}}$	$(1 - 6\% \times 2.3)^{\frac{1}{12}}$

이를 정리한 SMM 표는 아래와 같습니다.

월	100 PSA	60 PSA	230 PSA
1	0.000167	0.0001	0.000384
20	0.003396	0.002022	0.008010
200	0.005143	0.003051	0.012299

Question 5

다음의 CMO 는 9% 금리를 갖는 모기지 풀을 담보로 가지고 있다.

tranche	par(m USD)	coupon rate(%)
A	300	7
B	200	6.75
C	200	7.25
D	250	7.75

투자자가 금리 9% 의 notional IO 투자를 원한다고 하자. 해당 상품의 액면가를 계산하시오.

Answer

먼저, CMO의 담보에서는 9%의 현금흐름이 발생하는 반면 각 tranche의 쿠폰은 9% 미만이므로 초과이자가 $17.125m 만큼 발생합니다.

이 초과이자분으로 금리 9%의 원금 없이 이자만 지급하는 구조화채권(Nominal IO)을 설계한다면, 액면가(명목금액)은 9%의 이자로 CMO의 초과수익인 $17.125m를 지급해야합니다.

따라서, **액면가(명목금액)는 $17.125m/9%=$190.2778m**입니다.

```
total_inflow=(300+200+200+250)*0.09
total_outflow=300*0.07+200*0.0675+200*0.0725+250*0.0775
excess_interest=total_inflow-total_outflow
notional=excess_interest/0.09
paste(excess_interest, notional, sep=" / ")
```

```
[1] "17.125 / 190.277777777778"
```

이자율기간구조 과제1

Homework1

Problem1

1. 아래의 표와 같이 spot rate curve 가 주어져 있다고 가정하자. 여기에서 각 금리는 연속복리법이 적용되었다. 아래 세 가지 채권의 가격을 구하시오.

   (a) 3-year zero coupon bond

   (b) 1-year coupon bond paying 4% quarterly

   (c) 3-year floating rate bond with a 35 basis point spread, paid semiannually.

   참고: for the floating rate bond with a semi-annual schedule, if the payment date is t_i, then it pays $\frac{\mathsf{R}_{t_{i-1}}(t_i)+\text{spread}}{2}$. 여기에서 $\mathsf{R}_{t_{i-1}}(t_i)$는 t_{i-1}에 결정되는 spot rate을 의미한다. 물론 만기시에는 액면가도 지급한다.

maturity	yield(%)	maturity	yield(%)
.25	6.33	1.75	6.87
.5	6.49	2	6.88
.75	6.62	2.25	6.89
1	6.71	2.5	6.88
1.25	6.79	2.75	6.86
1.5	6.84	3	6.83

Answer

```
rm(list=ls())
library(tidyverse)
spot <- tibble(t=seq(0.25,3,0.25),
               y=c(0.0633,0.0649,0.0662,0.0671,0.0679,0.0684,
                   0.0687,0.0688,0.0689,0.0688,0.0686,0.0683)) %>%
```

```
  mutate(d=exp(-y*t))
```

(a) 81.4729

```
# (a) 3-year zero coupon bond
a=100*spot$d[which(spot$t==3)]
a
```

```
[1] 81.47288
```

(b) 97.3492

```
b=spot %>%
  filter(t<=1) %>%
  mutate(cf=c(1,1,1,101)) %>%
  mutate(pv_cf=cf*d) %>%
  select(pv_cf) %>%
  sum()
b
```

```
[1] 97.3492
```

(c) 100.9334

FRN의 spread가 0이라면 가격은 액면가인 100에 거래될 것 입니다.

문제의 FRN은 35bp의 프리미엄을 지급하므로, FRN의 가격은 액면가+35bp쿠폰의 현재가치입니다.

```
# (c) 3-year floating rate bond with a 35bp spread, semiannually
premium=spot %>%
  filter(t%%0.5==0) %>%
  mutate(cf=rep(35/10000/2*100,6)) %>%
  mutate(pv_cf=cf*d) %>%
  select(pv_cf) %>%
  sum()
c=100+premium
c
```

```
[1] 100.9334
```

Problem2

2. sofr 기준 거래에 있어서 compound in arrears 형태로 된 이자를 계산해보자. 10 million usd 를 명목 액수로 하고 이자의 계산은 2019년 9월 30일 부터 2019년 10월 31일까지라고 가정하자. 즉, 9월 30일에 해당하는 sofr 부터 10월 30일에 해당하는 sofr 가 주어지고, 복리법으로 계산된 이자는 10월 31일에 최종결정된다. hw1.xlsx 의 sofr 탭에는 뉴욕 Fed 에서 발표한 실제 sofr 데이터가 주어져 있다. 최종 이자를 계산하라. (1년은 360일로 하고, 휴일의 경우 가장 가까운 영업일의 sofr 가 적용된다)

Answer : **16,187.27$**

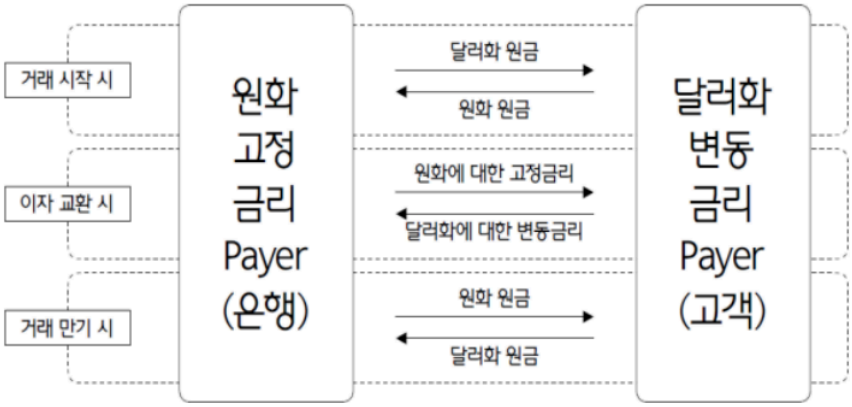
publish	apply	sofr	days	daily sofr	start	end
2019.9.30	2019.10.1	2.35	1	1.000065278	10,000,000	10,000,652.78
2019.10.1	2019.10.2	1.88	1	1.000052222	10,000,652.78	10,001,175.03
2019.10.2	2019.10.3	1.85	1	1.000051389	10,001,175.03	10,001,688.98
2019.10.3	2019.10.4	1.84	1	1.000051111	10,001,688.98	10,002,200.18
2019.10.4	2019.10.5	1.82	3	1.000151667	10,002,200.18	10,003,717.18
2019.10.5	2019.10.6			1	10,003,717.18	10,003,717.18
2019.10.6	2019.10.7			1	10,003,717.18	10,003,717.18
2019.10.7	2019.10.8	1.83	1	1.000050833	10,003,717.18	10,004,225.70
2019.10.8	2019.10.9	1.85	1	1.000051389	10,004,225.70	10,004,739.81
2019.10.9	2019.10.10	1.85	1	1.000051389	10,004,739.81	10,005,253.94
2019.10.10	2019.10.11	1.85	1	1.000051389	10,005,253.94	10,005,768.10
2019.10.11	2019.10.12	1.85	4	1.000205556	10,005,768.10	10,007,824.84
2019.10.12	2019.10.13			1	10,007,824.84	10,007,824.84
2019.10.13	2019.10.14			1	10,007,824.84	10,007,824.84
2019.10.14	2019.10.15			1	10,007,824.84	10,007,824.84
2019.10.15	2019.10.16	2	1	1.000055556	10,007,824.84	10,008,380.83
2019.10.16	2019.10.17	2.05	1	1.000056944	10,008,380.83	10,008,950.75
2019.10.17	2019.10.18	1.95	1	1.000054167	10,008,950.75	10,009,492.91
2019.10.18	2019.10.19	1.88	3	1.000156667	10,009,492.91	10,011,061.06
2019.10.19	2019.10.20			1	10,011,061.06	10,011,061.06
2019.10.20	2019.10.21			1	10,011,061.06	10,011,061.06
2019.10.21	2019.10.22	1.86	1	1.000051667	10,011,061.06	10,011,578.30
2019.10.22	2019.10.23	1.87	1	1.000051944	10,011,578.30	10,012,098.34
2019.10.23	2019.10.24	1.87	1	1.000051944	10,012,098.34	10,012,618.42
2019.10.24	2019.10.25	1.86	1	1.000051667	10,012,618.42	10,013,135.73
2019.10.25	2019.10.26	1.84	3	1.000153333	10,013,135.73	10,014,671.08
2019.10.26	2019.10.27			1	10,014,671.08	10,014,671.08
2019.10.27	2019.10.28			1	10,014,671.08	10,014,671.08
2019.10.28	2019.10.29	1.82	1	1.000050556	10,014,671.08	10,015,177.38
2019.10.29	2019.10.30	1.81	1	1.000050278	10,015,177.38	10,015,680.92
2019.10.30	2019.10.31	1.82	1	1.000050556	10,015,680.92	10,016,187.27
Total Interest						**16,187.27**

Problem3

3. 아래 그림은 은행-고객 사이의 통화스왑상품을 설명하고 있다. 여기에서 고객이 지급하는 외화금리는 compound SOFR 이다. 아래 설명의 빈 칸을 다음의 예시들에서 골라 순서대로 쓰시오.

증가하므로 / 감소하므로 / 상승하므로 / 하락하므로 / 거래이익이 / 거래손실이

■ 통화 스왑의 Cashflow



■ 손익구조 및 성격

본 통화스왑은 외화변동금리 및 환율의 움직임에 따라 이익 또는 손실이 발생할 수 있는 거래입니다.

1) 이자교환일 손익구조

① 환율의 변동은 없고, **외화자산 변동금리**만 변동하는 경우

이자교환 시점에 적용되는 외화자산 변동금리가 원화고정금리 수준 이하인 경우, 고객이 지급하는 외화변동금리 기준 이자금액이 [ ] 통화스왑 거래에서 고객에게 [ ] 발생하게 됩니다. 반대로 외화자산 변동금리가 원화고정금리 수준 이상인 경우, 고객이 지급하는 외화변동금리 기준 이자금액이 [ ] 통화스왑 거래에서 고객에게 [ ] 발생하게 됩니다. 이러한 거래손익의 크기는 외화자산 변동금리의 상승과 하락 정도에 따라 비례하여 증가하게 됩니다.

② **환율**이 변동하고, 외화자산 변동금리는 변동하지 않는 경우

이자교환 시점에 적용되는 환율이 거래시점보다 하락하는 경우, 지급하는 외화자산 변동금리 기준 이자금액의 원화 환산가치가 [ ] 통화스왑 거래에서 고객에게 [ ] 발생하게 됩니다. 반대로 환율이 거래시점보다 상승하는 경우, 지급하는 외화자산 변동금리 기준 이자금액의 원화 환산가치가 [ ] 통화스왑 거래에서 고객에게 [ ] 발생하게 됩니다. 이러한 거래손익의 크기는 환율의

Answer

이자교환 시점에 적용되는 외화자산 변동금리가 원화고정금리 수준 이하인 경우, 고객이 지급하는 외화변동금리 기준 이자금액이 **감소하므로** 통화스왑 거래에서 고객에게 **거래이익이** 발생하게 됩니다. 반대로 외화자산 변동금리가 원화고정금리 수준 이상인 경우, 고객이 지급하는 외화변동금리 기준 이자금액이 **증가하므로** 통화스왑 거래에서 고객에게 **거래손실이** 발생하게 됩니다.

이자교환 시점에 적용되는 환율이 거래시점보다 하락하는 경우, 지급하는 외화자산 변동금리 기준 이자금액의 원화 환산가치가 **하락하므로** 통화스왑 거래에서 고객에게 **거래이익이** 발생하게 됩니다. 반대로 환율이 거래시점보다 상승하는 경우, 지급하는 외화자산 변동금리 기준 이자금액의 원화 환산가치가 **상승하므로** 통화스왑 거래에서 고객에게 **거래손실이** 발생하게 됩니다.

Problem4

4. 수업에서는 금리선물만 학습하였으나, 국채선물도 활발히 거래된다. 금리와 달리 x년 국채선물은 해당 만기를 지닌 국채를 실물인도한다. 시장에는 다양한 표면금리를 지닌 국채들이 존재하므로 x년 국채선물 가격은 1개 있으나, 인도할 수 있는 자격을 갖춘 국채들은 여럿이 있다. 실물인도자(seller)는 자격요건을 갖춘 국채들 중 가장 낮은 가격의 채권을 인도할 것이며, 이 때 전환계수 conversion factor 를 곱하여 실물인수자(buyer)로부터 금액을 받게 된다. 보다 정확히, 1계약당 (선물정산가격 settlement price $\times$ 전환계수 $+$ 경과이자)$\times$ 1,000 usd 이다.

   실물인도자가 건네기로 결정한 채권이 표면금리 5.25% 국채라 하고 해당되는 전환계수는 0.9014 이다. 선물정산가격이 120-08 이고 (국채 호가방식을 상기하시오) 경과이자는 0.38 일때, 실물인수자가 지불하는 청구금액을 구하시오.

Answer : **108,773.3$**

```
settle=120+8/32
cf=0.9014
accrued=0.38

value=(settle*cf+accrued)*1000
value
```

```
[1] 108773.3
```

Problem5

5. 금리선물의 가격 Fut_0 는 미래 settlement price 로부터 오는 금리 Fut_T 의 위험중립기대치로 결정이 된다. 선도금리가격과 이 금리선물의 차이가 존재하는데(convexity adjustment), 이 작은 차이를 무시한다면 공정한 선도금리가격으로 금리선물의 이론가를 생각할 수 있다. 예를 들어, 현재 1개월 Euribor 가 1%, 4개월 Euribor 가 $\mathsf{R}(4)$ 라고 할 때, 1개월 후 만기인 3month Euribor 선물의 이론가는 1개월-4개월 구간의 선도금리이다. act/360 을 사용하고, 현재부터 1개월 후까지는 30일, 4개월까지는 122일이라 하자.

$$\left(1+\mathsf{R}(1)\frac{30}{360}\right)\left(1+\mathsf{Fut}_0\frac{92}{360}\right)=\left(1+\mathsf{R}(4)\frac{122}{360}\right).$$

위의 식으로부터 Fut_0 이 도출되고 선물이론가는 $100-\mathsf{Fut}_0\times 100$ 이다. 현재 이 선물가격이 97.32 이라 하자. $\mathsf{R}(4)$ 과 함께 1개월, 4개월 할인팩터 $\mathsf{d}(1),\mathsf{d}(4)$ 를 구하시오.

Answer

$R(4)=2.2686\%,\ \ d(1)=0.9992,\ \ d(4)=0.9924$

```
r1=0.01
theo_prc=97.32
futures=(100-theo_prc)/100
r4=((1+r1*30/360)*(1+futures*92/360)-1)*360/122
d1=1/(1+r1*30/360)
d4=1/(1+r4*122/360)
paste(round(r4,6),round(d1,6),round(d4,6),sep=" / ")
```

```
[1] "0.022686 / 0.999167 / 0.992371"
```

이자율기간구조 과제2

Problem 1

한 제조업체의 재무담당자가 usd 100 million 에 달하는 운전자본 working capital 을 3개월 정도 repo 시장에 투자하여 이익을 취하고자 한다.

기간은 2020년 1월 15일(현재, 포함)에 시작하여 2020년 4월 15일(불포함)이 환매일이다. 즉 재무담당자는 91일 동안 daily SOFR 를 얻게 된다.

그러나 최근 SOFR 의 하락세가 고민이 되어 투자금의 절반인 50 million 은 2020년 4월 만기인 3개월 SOFR 선물을 매수하여 수익 금리를 고정하는 효과를 가지려고 한다.

기간 내내 금리가 1bp 하락한다면, act/360 기준으로 회사는 얼마의 손해가 발생하는가?

이 절반에 대하여 SOFR 선물로 헤지를 한다면 몇 개의 선물 계약을 해야할지 본인의 계산을 쓰세요.

Answer

전체 노출액 중 절반은 정확히 헷지하였으므로, 기간 중 노출액은 50mUSD입니다.

노출기간은 91일이므로, **회사의 손해**는 $50,000,000 \times 0.0001 \times \frac{91}{360} = 1263.889USD$

SOFR선물은 (100-r)로 호가하며, 1bp에 대해 25$만큼 변동합니다.

따라서, 손실액만큼 SOFR선물에서 이익이 발생하도록 하려면 $\frac{1263.889}{25} = 50.56$, **약 51계약 매수**해야합니다.

Problem 2

다음의 표는 2020년 6월 5일 ICE에서 가져온 usd스왑데이터이다. 편의상 각 스왑금리는 연 1회 지급한다고 하자. 우리는 bootstrapping 기법을 이용하여 이 스왑금리들을 할인율곡선 또는 현물이자율 곡선으로 변환할 수 있다.

구 분	1	2	3	4	5	6
Swap rate	0.307%	0.299%	0.346%	0.424%	0.517%	0.614%

이 문제에서는 그와 달리 매해 말 지급부담을 지니고 있는 은행 A를 고려한다. 이 은행은 현 시점에서 floating rate notes 변동금리부채권과 스왑을 매수하여 이 지급부담을 맞추려고 한다. 이때 변동금리부채권과 스왑의 액면값들을 구하는 것을 reverse bootstrapping이라 한다. 보다 정확히 다음의 구조를 고려하자.

- 각 FRN은 $N_1, ..., N_6$의 액면을 가지고 있으며 이들은 i번째 해의 말까지 매년 N_i 에 해당하는 변동금리를 지급하고 i번째 해의 말에는 액면 N_i 도 지급한다.
- 각 스왑의 액면 역시 N_i이며 A는 floating rate payer이다. 즉 i번째 해의 말까지 매년 N_i에 해당하는 변동금리를 거래상대방에게 지급한다. 역으로 거래상대방은 매해 말 $N_i s_i$ 을 지불한다. 여기에서 s_i는 i 만기 스왑금리이다.

$N_1, ..., N_6$를 구하고 소수점 네째자리까지 리포트하시오.

Answer

은행이 매년 1m$의 채무를 커버하기위해 1~6년 만기의 FRN을 $N_1 \sim N_6$만큼 매수하고, 여기에서 발생하는 변동금리이자분을 고정금리로 전환하기위해 Fixed-payer로 IRS를 동일한 명목금액 $N_1 \sim N_6$ 만큼 체결한다면, 매년 은행의 pay-off는 아래와 같습니다.

Year	FRN	IRS	Net
1	$\sum_{i=1}^{6} N_i r_i + N_1$	$\sum_{i=1}^{6}(N_i s_i - N_i r_i)$	$N_1 + \sum_{i=1}^{6} N_i s_i$
2	$\sum_{i=2}^{6} N_i r_i + N_2$	$\sum_{i=2}^{6}(N_i s_i - N_i r_i)$	$N_2 + \sum_{i=2}^{6} N_i s_i$
3	$\sum_{i=3}^{6} N_i r_i + N_3$	$\sum_{i=3}^{6}(N_i s_i - N_i r_i)$	$N_3 + \sum_{i=3}^{6} N_i s_i$
4	$\sum_{i=4}^{6} N_i r_i + N_4$	$\sum_{i=4}^{6}(N_i s_i - N_i r_i)$	$N_4 + \sum_{i=4}^{6} N_i s_i$
5	$\sum_{i=5}^{6} N_i r_i + N_5$	$\sum_{i=5}^{6}(N_i s_i - N_i r_i)$	$N_5 + \sum_{i=5}^{6} N_i s_i$
6	$N_6 r_6 + N_6$	$N_6 s_6 - N_6 r_6$	$N_6 + N_6 s_6$

해당 pay-off에 따라 6년의 액면금액 N_6부터 reverse-bootstrapping방식으로 N_i를 모두 산출한 결과는 아래와 같습니다.

N_1	N_2	N_3	N_4	N_5	N_6
0.9753m	0.9783m	0.9812m	0.9846m	0.9888m	0.9939m

```
swap <- swap %>% mutate(par=c(NA,NA,NA,NA,NA,c(1000000/(swap$rate[6]+1))))
for(i in 5:1){
  swap$par[i]=(1000000-sum(swap$rate[(i+1):6]*swap$par[(i+1):6]))/(1+swap$rate[i])
}
swap
```

```
# A tibble: 6 x 3
  tenor    rate     par
  <int>   <dbl>   <dbl>
1     1 0.00307 975296.
2     2 0.00299 978291.
3     3 0.00346 981216.
4     4 0.00424 984611.
5     5 0.00517 988785.
6     6 0.00614 993897.
```

Problem 3

미 재무부에서 발표하는 일별 par yield 데이터를 찾되, 2024년 1월부터 5월까지 매월 첫번째 일자의 커브를 구하시오. 그리고 이 5개의 금리 커브에 대하여 Nelson-Siegel-Svensson 모형을 적용하여 본인이 구한 최선의 fitted curve를 리포트하시오. 1개월 금리부터 30년 금리까지 데이터 전체를 활용하시오.

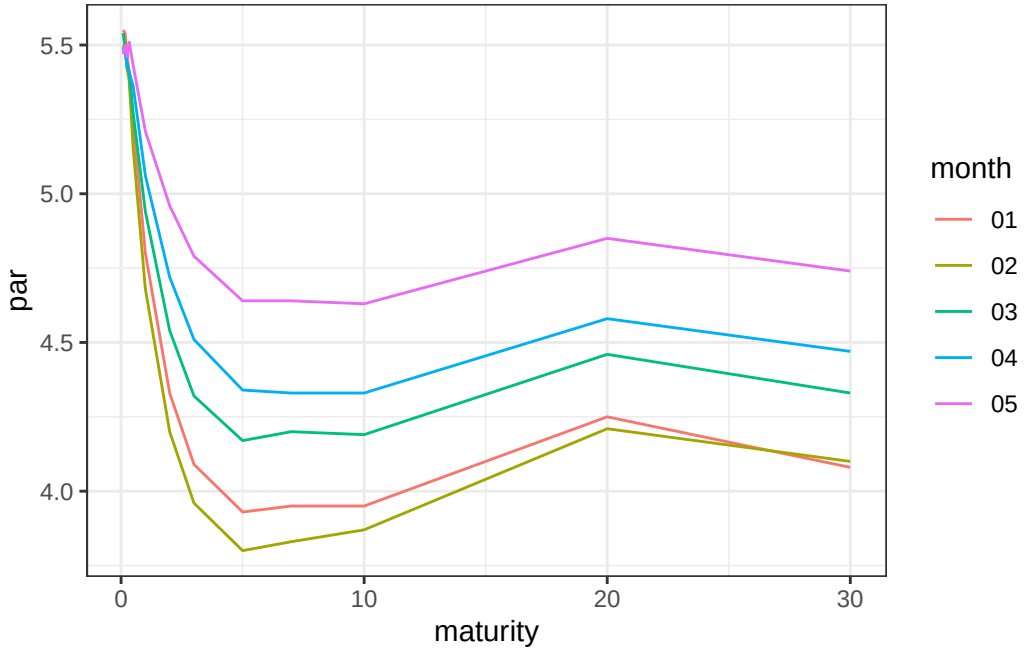
여기서 정답은 없으니, 본인이 적절한 초기값 및 제약식을 넣어 문제를 푸세요.

Answer

먼저, **미 재무부의 daily par yield** 데이터를 이용하여 산출한 매월 첫번째 일자의 커브는 아래와 같습니다.

```
uspar <- tibble()
uspar <- read_csv("investment_hw/usparyield.csv") %>%
  arrange(Date) %>%
  mutate(year=substr(Date,7,10),
         month=substr(Date,1,2)) %>%
  group_by(year,month) %>%
  slice(1) %>%
  pivot_longer(cols = contains(" "),values_to = "par", names_to = "time") %>%
  mutate(maturity=c(1,2,3,4,6,12,24,36,60,84,120,240,360)/12) %>%
  ungroup()
```

```
ggplot(uspar,aes(x=maturity,y=par,colour=month))+
  geom_line()+
  theme_bw()
```



다음으로, **NSS모형을 이용한 최적 fitted curve**를 구해보겠습니다.

> Nelson-Siegel-Svensson Model
>
> NSS모형은 instantaneous forword rate를 추정하는 대표적인 모형으로, 3개의 factor와 6개의 parameter($\beta_0 \sim \beta_3, \tau_1 \sim \tau_2$)를 이용합니다.
>
> $$f(t) = \beta_0 + \beta_1 e^{-\frac{t}{\tau_1}} + \beta_2 \frac{t}{\tau_1} e^{-\frac{t}{\tau_1}} + \beta_3 \frac{t}{\tau_2} e^{-\frac{t}{\tau_2}}$$
>
> 이를 통해 continuous-compounding spot rate curve를 추정할 수 있습니다.
>
> $$r(t) = \beta_0 + \beta_1 \frac{1 - e^{-\frac{t}{\tau_1}}}{\frac{t}{\tau_1}} + \beta_2 [\frac{1 - e^{-\frac{t}{\tau_1}}}{\frac{t}{\tau_1}} - e^{-\frac{t}{\tau_1}}] + \beta_3 [\frac{1 - e^{-\frac{t}{\tau_2}}}{\frac{t}{\tau_2}} - e^{-\frac{t}{\tau_2}}]$$

저는 **최적 fitted curve를 시뮬레이션을 통해 산출**할 예정입니다.

먼저, 초기값을 $\beta_0 = 30y \;\; par \;\; rate, \;\; \beta_1 = 1m \;\; par \;\; rate - \beta_0$ 및 $\beta_{2,3}$, $\tau_{1,2}$는 $(-5,\ 5)$ 범위의 Uniform dist. random number로 설정하겠습니다.

해당 초기값에 따라 constraint condition을 기준금리의 변동폭을 고려하여 25bp, $|\beta_0 + \beta_1 - 1m \;\; par \;\; rate| < 0.25$로 설정하고 finite difference method(optim fn. in R)를 통해 최적해를 산출합니다. (tau와 beta는 별도의 제약식 없이 진행하였습니다.)

이러한 방식을 1,000번 반복하는 Monte-carlo simulation을 통해 1,000개의 최적해를 산출하고, 1,000개의 NSS rate curve에서 MSE가 가장 낮은 curve를 최적 fitted curve로 선정하겠습니다.

이를 구현하기 위한 **R코드 및 24년 1~5월의 fitted curve는 아래와 같습니다.**

1. NSS Model 함수

```
# (nelson_siegel) return NSS model rates for given parameters
# maturity : vector of year maturity
# params : vector of NSS medel 6 parameters, beta0 to tau2

nelson_siegel <- function(maturity, params){

  # parameters
  beta0 = params[1]
  beta1 = params[2]
  beta2 = params[3]
  beta3 = params[4]
  tau1  = params[5]
  tau2  = params[6]

  # function
  beta0 +
    beta1 * (1 - exp(-maturity/tau1)) / (maturity/tau1) +
    beta2 * ((1 - exp(-maturity/tau1)) / (maturity/tau1) - exp(-maturity/tau1)) +
    beta3 * ((1 - exp(-maturity/tau2)) / (maturity/tau2) - exp(-maturity/tau2))

}
```

2. NSS 최적해 산출 함수(finite difference method)

```
# (optim_nelson_siegel) return optimal tibble for given term-structure,
#                       using finite difference method
# maturity : vector of year maturity
# term_sturucture : vector of real spot rate, order to maturity, omit %

optim_nelson_siegel <- function(term_structure = NULL, maturity = NULL, params = rep(0.1, 6),
                                contraint_threshold = 0.1, tau_threshold = 10, beta_threshold

  loss_function <- function(params){
```

```
  # parameters
  beta0 = params[1]
  beta1 = params[2]
  beta2 = params[3]
  beta3 = params[4]
  tau1  = params[5]
  tau2  = params[6]

  # contraints
  contraint = abs(beta0 + beta1 - term_structure[1])

  if(contraint > contraint_threshold ){
    return(NA)
  }

  # if(tau1 > tau_threshold | tau2 > tau_threshold){
  #   return(NA)
  # }
  #
  # if(beta2 > beta_threshold | beta3 > beta_threshold ){
  #   return(NA)
  # }

  sum(
    (term_structure - beta0 -
       beta1 *  (1 - exp(-maturity/tau1)) / (maturity/tau1) -
       beta2 * ((1 - exp(-maturity/tau1)) / (maturity/tau1) - exp(-maturity/tau1)) -
       beta3 * ((1 - exp(-maturity/tau2)) / (maturity/tau2) - exp(-maturity/tau2))
    )^2)

}

# set initial values for to beta0 and beta1 that respect the contraint
# beta0 is long-term rate & beta1 is short-term rate minus beta0
```

```
  params = c(term_structure[length(term_structure)],
             term_structure[1]-term_structure[length(term_structure)],
             params[3:6] )

  # beta0 & beta1 is equal to (short-term rate)/2
  # params = c(term_structure[1]/2,term_structure[1]/2,params[3:6] )

  # minimization of the loss function
  optimization = optim(params, loss_function)

  # optimal parameters
  optim_params = c(beta0 = optimization$par[1],
                   beta1 = optimization$par[2],
                   beta2 = optimization$par[3],
                   beta3 = optimization$par[4],
                   tau1  = optimization$par[5],
                   tau2  = optimization$par[6])

  # fitted value from Nelson-Siegel base function
  fitted_values = nelson_siegel(maturity, optim_params)

  # compute the mean square error
  mse_fit = sd(fitted_values - term_structure, na.rm = TRUE)

  # output
  dplyr::tibble(
    maturity = list(maturity),
    term_structure = list(term_structure),
    start_params = list(params),
    optim_params = list(optim_params),
    fit = list(fitted_values),
    mse = mse_fit
  )
}
```

3. Uniform dist. random number 생성

```r
# (random_params) return random number follows an uniform distribution
# n.params: number of parameter to generate
# params.min: for random generation, minimum parameter
# params.max: for random generation, maximum parameter
# seed: to control randomness

random_params <- function(n.params = 1, params.min = 0, params.max = 5, seed = 1){

  set.seed(seed)

  runif(n.params, params.min, params.max)

}
```

4. Monte-carlo simulation (Calibration)

```r
# (calibrate_nelson_siegel) return n-times "optim_nelson_siegel" outputs
#                           using Monte-carlo simulation for random 3 parameters(beta2 to tau2
# n: number of simulations
# params.min: for random generation, minimum parameter
# params.max: for random generation, maximum parameter
# verbose: disply progress in the importation.

calibrate_nelson_siegel <- function(object, n = 100, params.min = -5, params.max = 5,
                                    contraint_threshold = 0.01, verbose = TRUE ){

  # initialize the parameters
  term_structure = object$term_structure[[1]]
  maturity = object$maturity[[1]]

  # list containing all the simulations
  simulations = list()
```

```
  # safe version to avoid errors if we made many simulations
  safe_optim = purrr::safely(optim_nelson_siegel)

  for(i in 1:n){

    # generate a random seed
    random_seed = mean(random_params(10, 0, 100000, seed = i))

    # generate random parameters
    random_params = random_params(6, params.min, params.max, seed = random_seed)

    simulations[[i]] = safe_optim(term_structure  = term_structure, maturity = maturity, param

    if( verbose &(i%%50 == 0)){message("Simulations: ", i, "/", n)}

  }

  # unique dataset for all the simulation
  simulations_df = dplyr::bind_rows(simulations)

  # add the initial object
  simulations_df = dplyr::bind_rows(object, simulations)

  # index for the simulations
  simulations_df = dplyr::mutate(simulations_df, n = 1:nrow(simulations_df))

  return(simulations_df)

}
```

5. Optimal fitted curve 산출 및 시각화

```
# (optimal_params_ns) summarize of all above, return optimal result minimize MSE from n-times
# term_structure = NULL
# maturity = NULL
# n = 1000
# params.init = rep(0.1, 6)
# params.min = -5
# params.max = 5
# contraint_threshold = 0.1
# label = NULL (label for the plot)
# verbose = TRUE

optimal_params_ns <- function(term_structure = NULL, maturity = NULL,
                              n = 1000, params.init = rep(0.1, 6), params.min = -5, params.max
                              contraint_threshold = 0.1, label = NULL, verbose = TRUE){

  # first fit
  first_fit_ns = optim_nelson_siegel(term_structure = term_structure, maturity = maturity, par

  # simulations
  sim_fit_ns = calibrate_nelson_siegel(first_fit_ns, n = n, params.min = params.min, params.ma

  # best parameters
  df_optim_params = sim_fit_ns[which(sim_fit_ns$mse == min(sim_fit_ns$mse, na.rm = TRUE)),]



  # setting the title of the plot
  if(!is.null(label) & is.character(label)){

    plot_title = paste0("Fitted Nelson-Siegel-Svensonn vs Real Value ", "(", label, ")" )

  } else {

    plot_title = "Fitted Nelson-Siegel-Svensonn vs Real Value"
```

```
}


# plot of fitted vs real values
plot_df =  dplyr::inner_join(
  dplyr::tibble(
    t = maturity,
    pred = df_optim_params$fit[[1]]
  ),
  dplyr::tibble(
    t = maturity,
    real = term_structure
  ),
  by = "t"
)

# Plot Real value vs Fitted Values
plot_ns = plot_df %>%
  mutate(label = paste0("T = ", round(t, 3)))%>%
  ggplot()+
  geom_point(aes(t, pred), color = "red", size = 2, alpha = 0.8) +
  geom_point(aes(t, real), color = "black", alpha = 0.5)+
  geom_line(aes(t, pred), color = "red", size = 1) +
  geom_line(aes(t, real), color = "black", linetype = "dashed") +
  geom_label(aes(t+1, real-0.001, label = label), size = 1.5)+
  theme(axis.text.x = element_text(angle = 25, face = "bold", size = 7),
        axis.text.y = element_text(face = "bold"),
        axis.title  = element_text(face = "bold"),
        plot.title  = element_text(face = "bold"),
        plot.subtitle = element_text(face = "italic"),
        plot.caption = element_text(face = "italic"),
        panel.grid.major.x = element_line(colour="grey60", linetype="dotted"),
        panel.grid.minor.x = element_blank(),
        panel.grid.major.y = element_line(colour="grey60", linetype="dotted"),
        legend.text = element_text(face = "italic", size = 5),
```

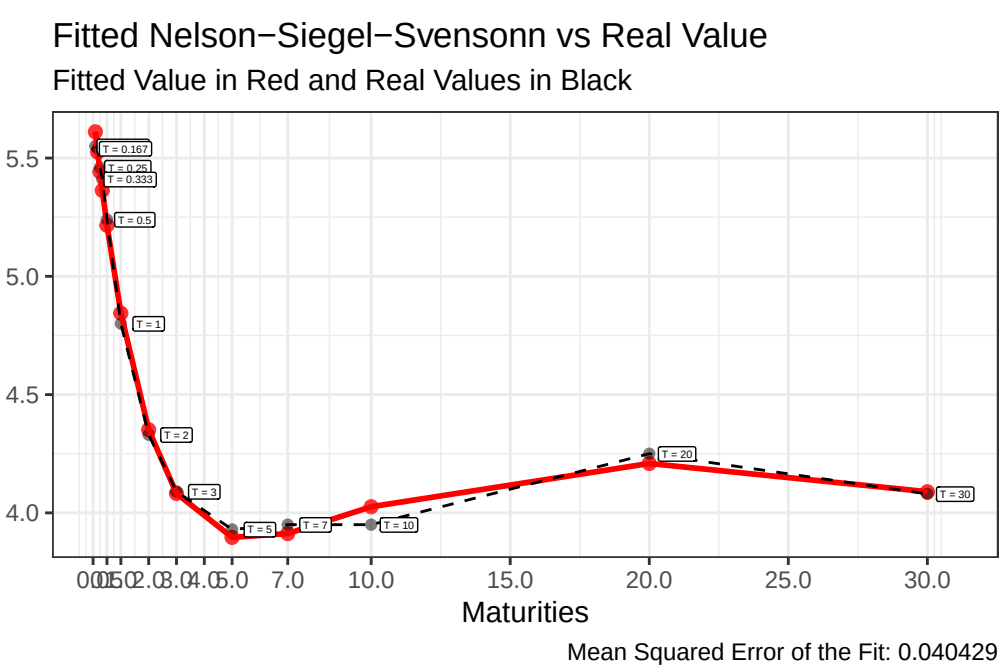
```
          legend.title = element_text(face = "bold"),
          legend.position = "top" ) +
    scale_x_continuous(breaks=c(0, 0.5, 1,2,3,4,5,7,10,15, 20, 25, 30))+
    ggtitle(plot_title, subtitle = "Fitted Value in Red and Real Values in Black" )+
    xlab("Maturities")+
    ylab("") +
    labs(caption = paste0("Mean Squared Error of the Fit: ", round(df_optim_params$mse, 6)))+
    theme_bw()



  # output
  structure(
    list(
      optim_params = tibble( beta0=df_optim_params$optim_params[[1]][1],
                             beta1=df_optim_params$optim_params[[1]][2],
                             beta2=df_optim_params$optim_params[[1]][3],
                             beta3=df_optim_params$optim_params[[1]][4],
                             tau1=df_optim_params$optim_params[[1]][5],
                             tau2=df_optim_params$optim_params[[1]][6]),
      plot_ns = plot_ns,
      simulations = sim_fit_ns,
      df_optim = df_optim_params

    )
  )

}
```

Result

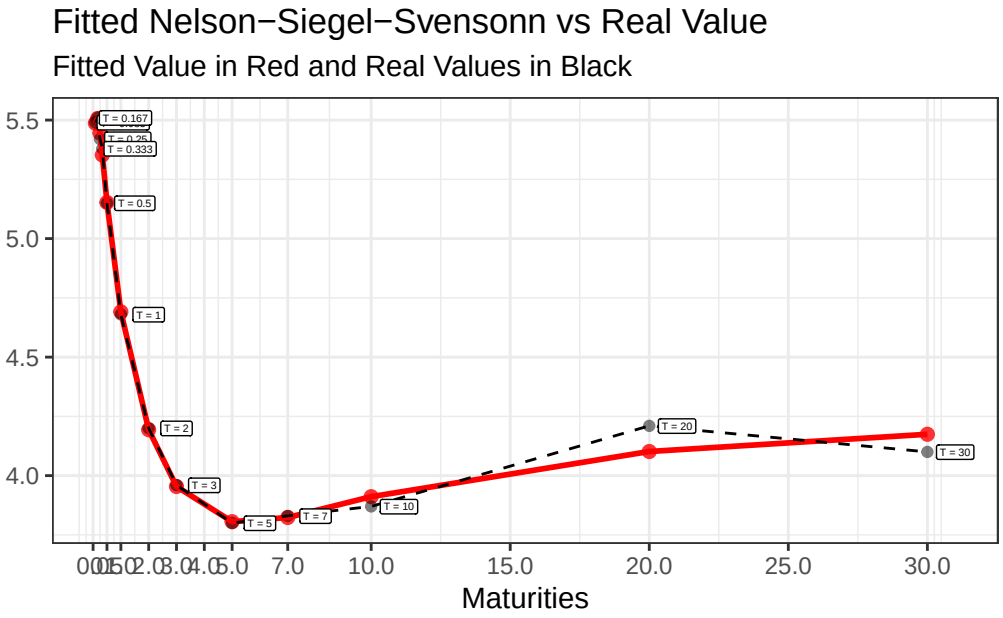
앞서 도식화한 **24년 1~5월의 US par-yield curve**에 위 방법을 적용하여 **각각 모델링한 fitted curve**는 아래와 같습니다.

```
uspar_jan <- uspar %>% filter(month=="01")
uspar_feb <- uspar %>% filter(month=="02")
uspar_mar <- uspar %>% filter(month=="03")
uspar_apr <- uspar %>% filter(month=="04")
uspar_may <- uspar %>% filter(month=="05")

Jan <- optimal_params_ns(uspar_jan$par,uspar_jan$maturity,n=1000,contraint_threshold = 0.25)
Feb <- optimal_params_ns(uspar_feb$par,uspar_feb$maturity,n=1000,contraint_threshold = 0.25)
Mar <- optimal_params_ns(uspar_mar$par,uspar_mar$maturity,n=1000,contraint_threshold = 0.25)
Apr <- optimal_params_ns(uspar_apr$par,uspar_apr$maturity,n=1000,contraint_threshold = 0.25)
May <- optimal_params_ns(uspar_may$par,uspar_may$maturity,n=1000,contraint_threshold = 0.25)
```

```
Jan$plot_ns
```

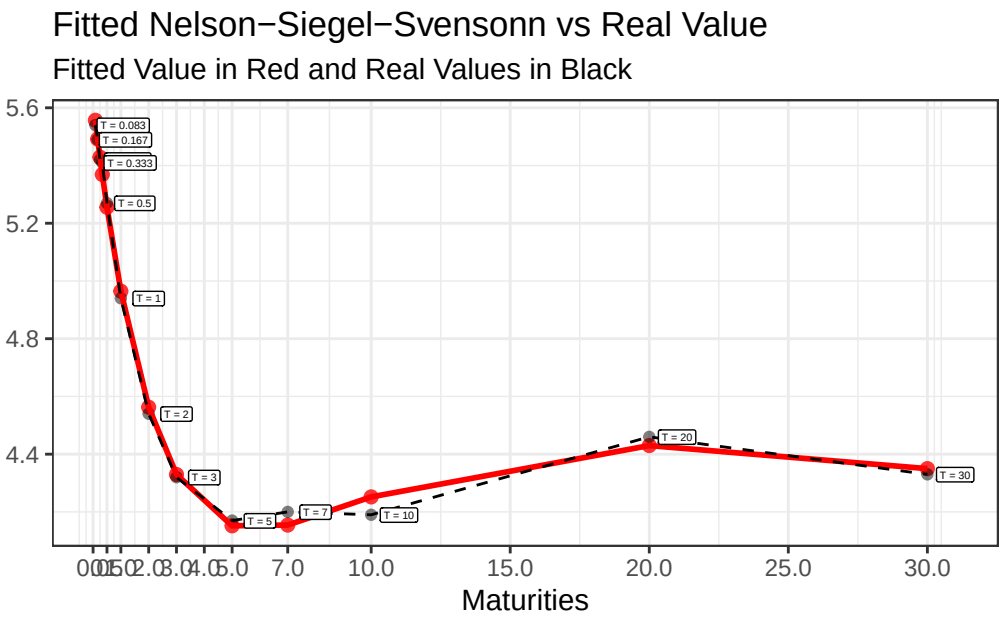



```
Feb$plot_ns
```



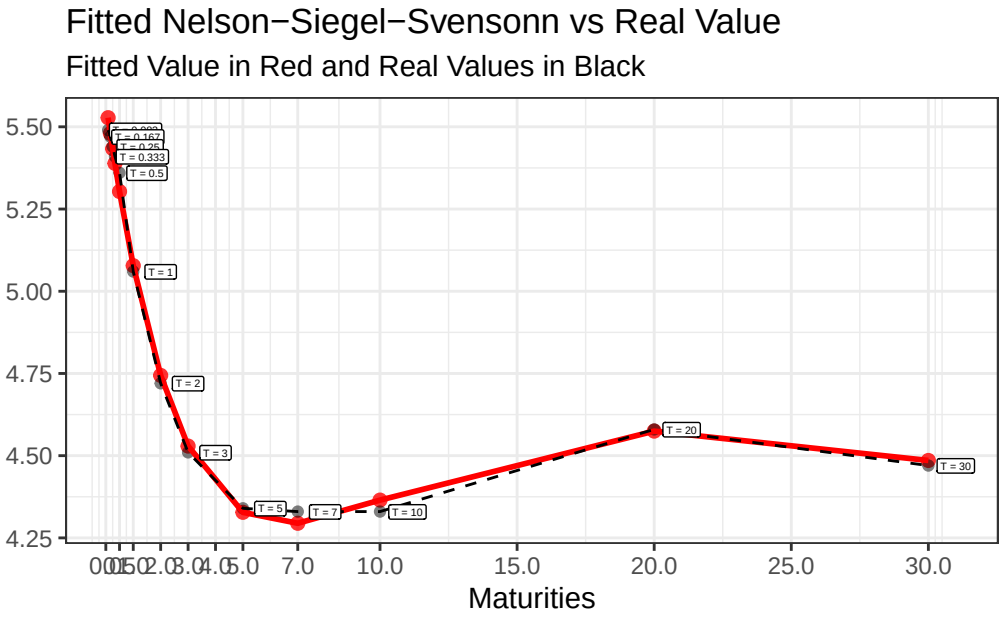
Mean Squared Error of the Fit: 0.041482

```
Mar$plot_ns
```

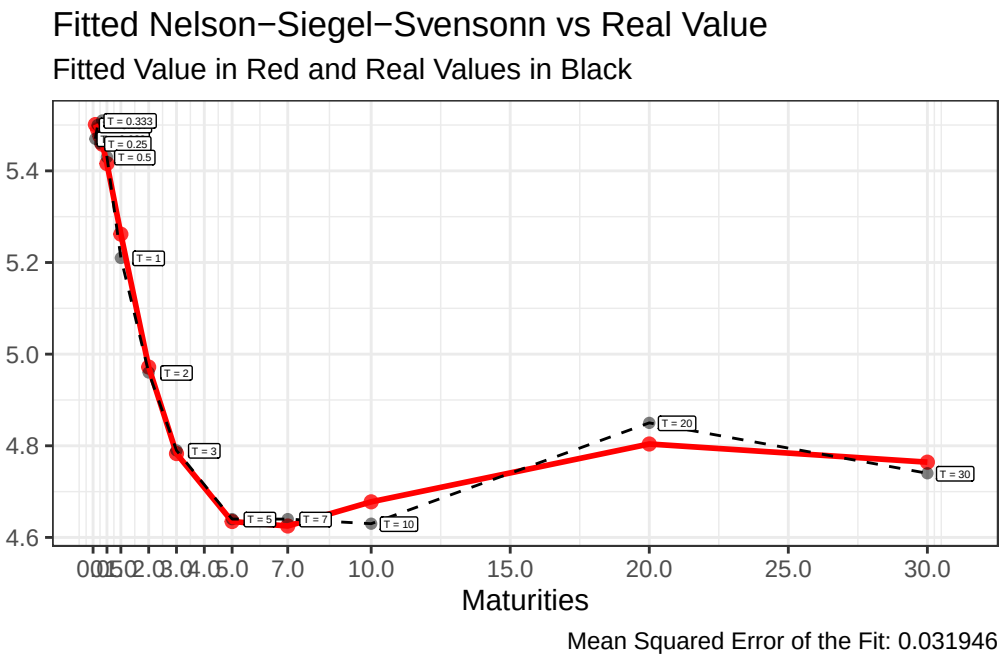


Mean Squared Error of the Fit: 0.030313

```
Apr$plot_ns
```




```
May$plot_ns
```



Problem 4

위 문제에서 구한 모수값들을 가지고 6월 첫날의 금리커브를 예측하려고 한다. 모수값들에 대하여 적절한 (예: 엑셀에서 제공하는 선형 추세선을 찾아 사용) 예측값들을 구하고 이를 바탕으로 금리 커브를 구하시오.

Answer

먼저, 문제3에서 최종적으로 산출한 NSS fitted curve에 대하여 각각의 파라미터는 아래와 같습니다.

```
optim_parameters <- Jan$optim_params %>%
  union_all(Feb$optim_params) %>%
  union_all(Mar$optim_params) %>%
  union_all(Apr$optim_params) %>%
  union_all(May$optim_params) %>%
  mutate(month=c("Jan","Feb","Mar","Apr","May"))
optim_parameters
```

```
# A tibble: 5 x 7
  beta0 beta1 beta2 beta3  tau1   tau2 month
  <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl> <chr>
1  2.35 3.35  -2.33  5.88 2.10  11.8   Jan
2  4.32 0.961 -3.16  1.34 2.06   0.127 Feb
3  3.32 2.30  -3.09  4.07 2.63  10.3   Mar
4  2.69 2.89  -3.15  6.44 3.45  11.7   Apr
5  3.46 2.05   1.74  4.09 0.802 14.1   May
```

이를 이용하여 6월달의 US par-yield curve를 예측하고자 할 때, 모수값들을 활용하여 NSS모델의 6월 예상파라미터를 추정할 수 있습니다.

여기에는 AR, ARMA, GARCH 등의 여러 모델을 생각할 수 있겠으나, 저는 직전 달의 모수값이 다음 달의 모수값에 대한 최적 추정치라고 생각합니다. 모수값의 시계열자료가 5개에 불과하고, **특별한 선형관계나 추세가 관찰되지 않기 때문**입니다. 이에 따라 **5월달의 NSS모델의 모수값을 6월의 예상파라미터**로 고려하겠습니다.

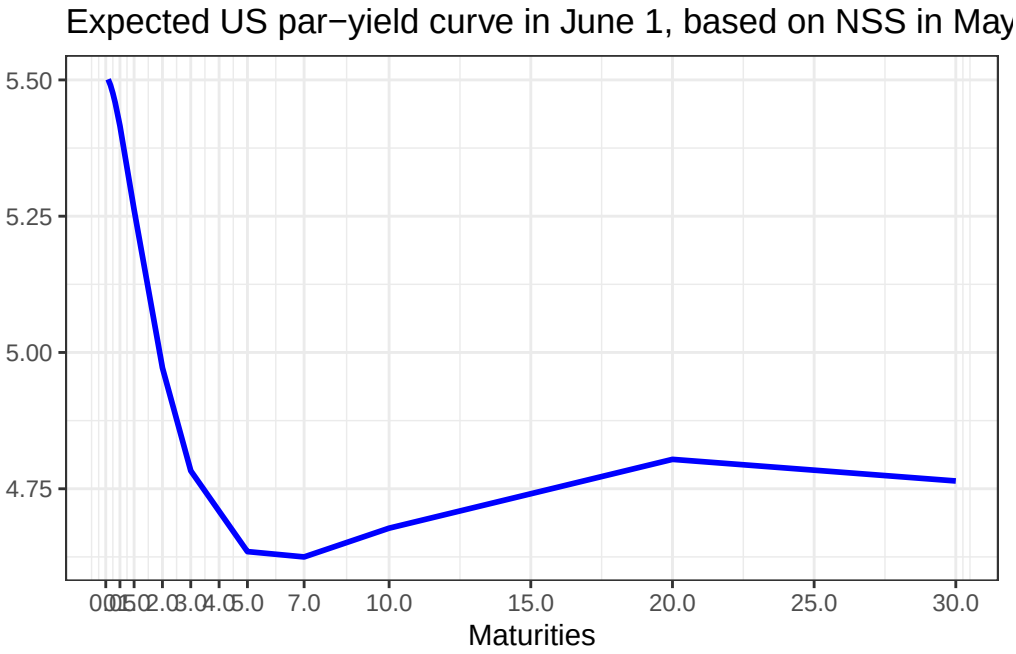
이를 이용하여 산출한 **6월 예상 US par-yield 및 curve는 아래와 같습니다.**

```
est_jun <- tibble(maturity=uspar_may$maturity,
                  par=May$df_optim$fit[[1]])
est_jun
```

```
# A tibble: 13 x 2
   maturity   par
      <dbl> <dbl>
 1   0.0833  5.50
```

```
 2    0.167    5.49
 3    0.25     5.48
 4    0.333    5.46
 5    0.5      5.42
 6    1        5.26
 7    2        4.97
 8    3        4.78
 9    5        4.63
10    7        4.62
11   10        4.68
12   20        4.80
13   30        4.76
```

```
ggplot(est_jun,aes(x=maturity,y=par))+
  geom_line(colour="blue", size=1)+
  scale_x_continuous(breaks=c(0, 0.5, 1,2,3,4,5,7,10,15, 20, 25, 30))+
  ggtitle("Expected US par-yield curve in June 1, based on NSS in May 1.")+
  xlab("Maturities")+
  ylab("") +
  theme_bw()
```



Problem 5

hw2.xlsx 에는 2011-10-28 부터 2016-10-28 까지의 스왑 데이터가 결측치가 있는 날들은 제외하고 들어 있다. 이를 이용하여 일별 금리변화를 주도하는 세 가지 주성분을 구하여 리포트하시오.

Answer

먼저, 예제파일을 읽어와 일별/만기별 스왑금리의 차이를 bp단위로 계산하겠습니다.

```
pca <- read_csv("investment_hw/term_hw2.csv") %>%
  pivot_longer(cols=contains("DSWP"),names_to = "maturity", values_to = "rate") %>%
  arrange(maturity,observation_date) %>%
  mutate(changes=(rate-lag(rate))*100) %>%
  filter(maturity==lag(maturity)) %>%
  mutate(date=observation_date,
         maturity=substr(maturity,5,nchar(maturity)) %>% as.integer()) %>%
  select(date,maturity,changes)
```

다음으로, 각 만기별 covariance matrix를 산출하면 아래와 같습니다.

```
cov_matrix <- pca %>%
  pivot_wider(names_from = "maturity", values_from = "changes")
cov_matrix <- cov_matrix[,2:ncol(cov_matrix)] %>%
  select(`1`,`2`,`3`,`4`,`5`,`7`,`10`,`30`) %>%
  cov()

options(digits=4)
cov_matrix
```

```
       1     2      3      4      5      7     10     30
1  2.213 3.055  3.523  3.696  3.704  3.431  2.955  2.098
2  3.055 5.697  7.042  7.839  8.177  8.046  7.338  5.748
3  3.523 7.042  9.896 11.304 12.095 12.314 11.528  9.366
4  3.696 7.839 11.304 13.825 14.949 15.678 15.021 12.600
5  3.704 8.177 12.095 14.949 16.859 17.865 17.439 15.007
7  3.431 8.046 12.314 15.678 17.865 20.054 20.057 18.130
10 2.955 7.338 11.528 15.021 17.439 20.057 21.084 19.854
```

```
30 2.098 5.748  9.366 12.600 15.007 18.130 19.854 21.117
```

이제, 공분산행렬을 통해 eigenvalue와 eigenvector를 계산하겠습니다.

```
eigenstuff <- eigen(cov_matrix)
eigenstuff
```

```
eigen() decomposition
$values
[1] 97.7556  9.9072  1.7153  0.5056  0.2766  0.2129  0.1903  0.1832

$vectors
         [,1]     [,2]    [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
[1,] -0.08566 -0.28417  0.5080  0.59000  0.44064 -0.32532 -0.06279  0.04302
[2,] -0.19278 -0.39768  0.4590  0.05126 -0.39712  0.61693  0.16366 -0.16203
[3,] -0.28762 -0.39863  0.1332 -0.40842 -0.31307 -0.42430 -0.22186  0.49654
[4,] -0.35997 -0.30873 -0.1546 -0.29835  0.19393 -0.30442  0.32447 -0.65316
[5,] -0.40648 -0.19686 -0.3123 -0.04801  0.46000  0.44075 -0.53295  0.07933
[6,] -0.44890  0.04588 -0.3212  0.29086  0.01963  0.06934  0.62949  0.45513
[7,] -0.45266  0.28606 -0.1312  0.43332 -0.50253 -0.19855 -0.36367 -0.28993
[8,] -0.41420  0.61983  0.5212 -0.34465  0.21905  0.04865  0.04712  0.03509
```

해당 **eigenvalue는 각 주성분의 설명력**을 나타냅니다. 각 eigenvalue의 전체 합에 대한 **비율은 전체 금리변화에서 해당 주성분이 설명하는 비율**을 나타냅니다.

```
eigenstuff$ratio=eigenstuff$values/sum(eigenstuff$values)
eigenstuff$ratio
```

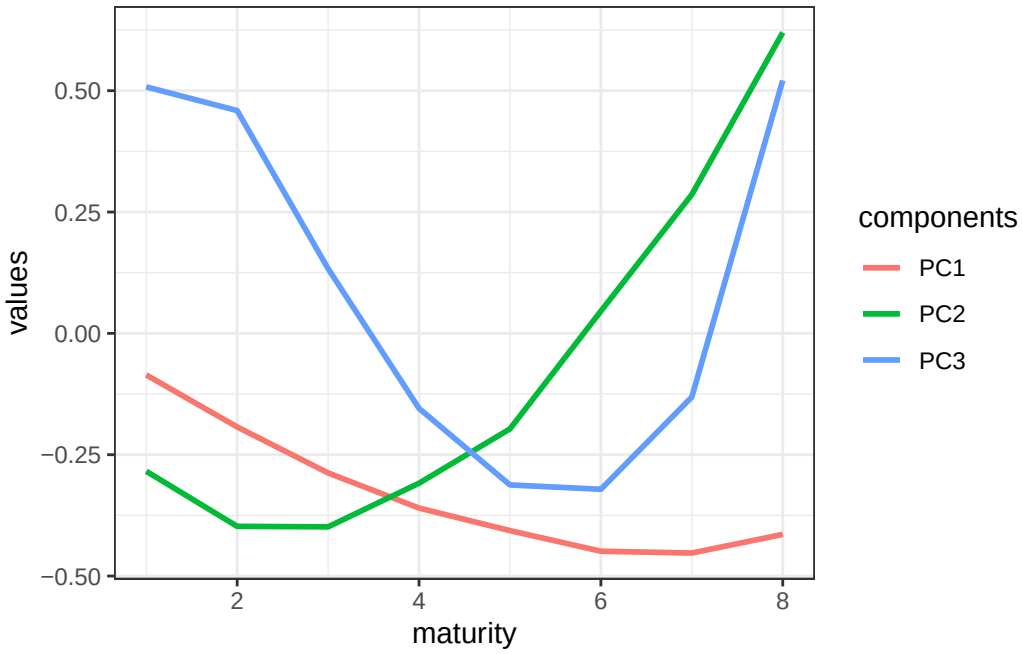
```
[1] 0.882696 0.089458 0.015488 0.004565 0.002498 0.001922 0.001718 0.001654
```

이에 따르면, **첫번째 주성분이 약 88.3%, 두번째 주성분이 8.9%, 세번째 주성분이 1.5%를 설명**하고 있습니다. 이는 **3가지의 주성분만으로 전체 swap rate curve 변화의 약 98.8%를 설명**할 수 있다는 의미입니다.

여기에서, **첫번째 주성분은 전체적인 금리수준의 변화(level change effect)**라고 할 수 있고 **두번째 주성분은 금리곡선의 기울기 변화(slope change effect)**, **세번째 주성분은 금리곡선의 볼록성 변화(convecture change effect)**라고 할 수 있습니다.

세가지 주성분의 eigenvector를 도식화하면 아래와 같습니다.

```
plot_pca <- tibble(maturity=c(1:8),
                   PC1=eigenstuff$vectors[,1],
                   PC2=eigenstuff$vectors[,2],
                   PC3=eigenstuff$vectors[,3]) %>%
  pivot_longer(cols=contains("PC"),names_to = "components", values_to = "values") %>%
  ggplot(.,aes(x=maturity,y=values,colour=components))+
  geom_line(size=1)+
  theme_bw()
plot_pca
```



이자율기간구조 과제3

Problem 1.

1. 위험중립측도 하에서 주어진 short rate 모형 $dr_t = \lambda dt + \sigma dW_t$을 고려하자. 아래 표에 있는 데이터를 이용하여 이 모형에 대한 이항트리를 구성하는 것이 목적이다. 그 외의 정보는 다음과 같이 주어져 있다:

   - $r_0 = 1.73\%$
   - historical $\sigma = 0.0173$

   이항트리의 time step 이 6개월 단위일 때, 다음 SSE 를 최소화하는 λ 를 구하시오:

$$\text{SSE} = \sum_i (\mathsf{P}^{\text{model}}(t_i) - \mathsf{P}^{\text{market}}(t_i))^2.$$

maturity (years)	price (face value 100)
.5	99.1338
1	97.8925
1.5	96.1462
2	94.1011
2.5	91.7136
3	89.2258
3.5	86.8142

Answer

$\lambda = 0.0168$

lambda	**0.01680**							
sig	1.73%	per year						
dt	0.5	6-month						
time	0	0.5	1	1.5	2	2.5	3	3.5
	1.730%	3.793%	5.856%	7.919%	9.983%	12.046%	14.109%	16.172%
		1.347%	3.410%	5.473%	7.536%	9.599%	11.662%	13.726%
			0.963%	3.026%	5.089%	7.153%	9.216%	11.279%
				0.580%	2.643%	4.706%	6.769%	8.832%
					0.196%	2.259%	4.323%	6.386%
						-0.187%	1.876%	3.939%
							-0.571%	1.493%
								-0.954%
price of a security that pays 1 at (t,i) only								
time	0	0.5	1	1.5	2	2.5	3	3.5
	1	0.49571209	0.24324275	0.11816142	0.05683037	0.02706432	0.01276343	0.00596118
		0.49571209	0.48944118	0.35877978	0.2314421	0.1385833	0.07888166	0.04322887
			0.24619843	0.36312761	0.35346915	0.28386774	0.20315137	0.13436904
				0.12250924	0.239935	0.29075163	0.27906687	0.23206675
					0.06107758	0.14891173	0.21565785	0.24051311
					0	0.03050885	0.08889282	0.14958109
					0	0	0.01526871	0.05168957
					0	0	0	0.0076562
disc factor	1	0.991424181	0.97888235	0.96257805	0.94275421	0.91968757	0.89368272	0.86506581
market price		0.991338	0.978925	0.961462	0.941011	0.917136	0.892258	0.868142
difference^2		7.42714E-09	1.819E-09	1.2456E-06	3.0388E-06	6.5105E-06	2.0298E-06	9.463E-06
SSE	**0.000022**							

Problem 2.

2. 위험중립측도에서 주어진 Ho-Lee 모형 $dr_t = \lambda_t dt + \sigma dW_t$ 을 고려하자. 위의 문제와 동일하게 초기 short rate r_0 및 volatility σ 을 두고 6개월 단위 이항트리를 구성하되, 위의 채권 시장가격을 재생산 할 수 있는 $\lambda_{0.5}, \lambda_1, \ldots, \lambda_3$ 의 값들을 구하시오. 즉, 이 이항트리는 $\mathsf{P}^{\text{model}}(t_i) = \mathsf{P}^{\text{market}}(t_i)$ for all i 를 만족해야 한다.

Answer :

$\lambda_{0.5} \sim \lambda_3$은 각각 0.01662, 0.02187, 0.01472, 0.01779, 0.00812, 0.00046 입니다.

sig	0.0173	per year						
dt	0.5	6-month						
time	0	0.5	1	1.5	2	2.5	3	3.5
lambda at t		**0.01662**	**0.02187**	**0.01472**	**0.01779**	**0.00812**	**0.00046**	**0.10000**
	1.730%	3.784%	6.101%	8.060%	10.173%	11.802%	13.049%	19.272%
		1.338%	3.655%	5.614%	7.726%	9.356%	10.602%	16.825%
			1.208%	3.167%	5.280%	6.909%	8.156%	14.379%
				0.721%	2.833%	4.463%	5.709%	11.932%
					0.387%	2.016%	3.262%	9.486%
						-0.431%	0.816%	7.039%
							-1.631%	4.592%
								2.146%
price of a security that pays 1 at (t,i) only								
time	0	0.5	1	1.5	2	2.5	3	3.5
	1	0.49571209	0.24325322	0.1180261	0.05672686	0.02699055	0.01274327	0.00598139
		0.49571209	0.48946238	0.35836553	0.23101751	0.13820293	0.07875669	0.04337735
			0.24620916	0.3627049	0.352816	0.28308313	0.20282855	0.13483681
				0.12236547	0.23948845	0.28994234	0.27862213	0.23288567
					0.06096309	0.14849432	0.2153132	0.24137352
					0	0.03042273	0.08875038	0.15012363
					0	0	0.01524418	0.05187967
					0	0	0	0.00768475
disc factor	1	0.991424181	0.97892476	0.961462	0.94101191	0.917136	0.8922584	0.8681428
market price		0.991338	0.978925	0.961462	0.941011	0.917136	0.892258	0.868142

Problem 3.

3. 연속시간에서 정의된 Vasicek 모형 $dr_t = k(\theta - r_t)dt + \sigma dW_t$ 을 고려하자. 액면가 1 이고 만기 t 인 무이표 채권의 가격 $\mathsf{d}(t) = \mathsf{E}\left[e^{-\int_0^t r_s ds}\right]$ (또는 discount factor) 수식을 구하시오.

Answer

다음 3단계로 나누어 Vasicek 모형 하의 무이표채권 가격 수식을 유도하겠습니다.

(1) Integrating Factor Method를 이용하여 Short rate r_t 의 SDE를 풀고 확률분포 유도

(2) Spot rate $R_t = -\int_0^t r_s ds$ 의 확률분포 유도

(3) $X \sim N(\mu, \sigma^2)$ 에 대해 정규분포의 적률생성함수가 $E[e^X] = e^{\mu + \frac{1}{2}\sigma^2}$ 임을 이용하여 $d(t) = E[e^{-\int_0^t r_s ds}]$ 를 계산

(1) *Short rate*의 확률분포 계산

$Vasicek \ : \ dr_t = k(\theta - r_t)dt + \sigma dW_t \Rightarrow \ dr_t + kr_t dt = k\theta dt + \sigma dW_t$ 에서,

양변에 Integrating Factor e^{kt} 를 곱하면, $e^{kt}dr_t + e^{kt}kr_t dt = e^{kt}k\theta dt + e^{kt}\sigma dW_t$

$d(e^{kt}r_t) = e^{kt}dr_t + e^{kt}kr_t dt$ 임을 통해 $d(e^{kt}r_t) = e^{kt}k\theta dt + e^{kt}\sigma dW_t$ 라고 할 수 있습니다.

양 변을 $(0, t)$ 에 대해 적분하면, $e^{kt}r_t - r_0 = \theta(e^{kt} - 1) + \sigma \int_0^t e^{ks} dW_s$ 이 됩니다.

이제, 위 식을 r_t에 대해 정리하면 현물이자율에 관한 식을 얻을 수 있습니다.

$$r_t = r_0 e^{-kt} + \theta(1 - e^{-kt}) + \sigma \int_0^t e^{k(s-t)} dW_s$$

이를 이용하여 $E[r_t],\ Var[r_t]$를 계산해보겠습니다.

먼저, Ito integral의 기대값은 0이므로 $E[\int_0^t e^{k(s-t)} dW_s] = 0$이 되며, $E[r_t] = r_0 e^{-kt} + \theta(1 - e^{-kt})$가 됩니다.

다음으로, $Var[r_t] = \sigma^2 Var[\int_0^t e^{k(s-t)} dW_s]$이므로,

$Var[\int_0^t e^{k(s-t)} dW_s] = E[(\int_0^t e^{k(s-t)} dW_s)^2] - E[\int_0^t e^{k(s-t)} dW_s]^2_{=0} = E[(\int_0^t e^{k(s-t)} dW_s)^2]$,

$dW_t^2 = dt$이므로, $E[(\int_0^t e^{k(s-t)} dW_s)^2] = E[\int_0^t e^{2k(s-t)} ds] = \int_0^t e^{2k(s-t)} ds = \frac{1}{2k}(1 - e^{-2kt})$

즉, $Var[r_t] = \frac{\sigma^2}{2k}(1 - e^{-2kt})$

최종적으로 Short rate의 확률분포는 아래의 정규분포입니다.

$$r_t \sim N\left(r_0 e^{-kt} + \theta(1 - e^{-kt}),\ \frac{\sigma^2}{2k}(1 - e^{-2kt}) \right)$$

(2) $R_t = -\int_0^t r_s ds$의 확률분포 계산

먼저, $r_t = r_0 e^{-kt} + \theta(1 - e^{-kt}) + \sigma \int_0^t e^{k(s-t)} dW_s$를 이용하여 $R_t = -\int_0^t r_s ds$를 표현하겠습니다.

$$\int_0^t r_s ds = \int_0^t r_0 e^{-ks} ds + \int_0^t \theta(1 - e^{-ks}) ds + \int_0^t \sigma \int_0^s e^{k(u-s)} dW_u ds$$

$$\Rightarrow \int_0^t r_s ds = \frac{r_0(e^{-kt} - 1)}{-k} + t\theta - \frac{\theta(e^{-kt} - 1)}{-k} + \sigma \int_0^t \int_0^s e^{k(u-s)} dW_u ds$$

$$\therefore -\int_0^t r_s ds = \frac{r_0(e^{-kt} - 1)}{k} - t\theta - \frac{\theta(e^{-kt} - 1)}{k} - \sigma \int_0^t \int_0^s e^{k(u-s)} dW_u ds$$

이제, $-\int_0^t r_s ds = R_t$라고 하겠습니다.

위 (1)의 방법과 유사하게 $E[\int_0^t \int_0^s e^{k(u-s)} dW_u ds] = 0$이므로, R_t의 기대값은 아래와 같습니다.

$$E[R_t] = \frac{r_0(e^{-kt} - 1)}{k} - t\theta - \frac{\theta(e^{-kt} - 1)}{k}$$

다음으로, $Var[R_t] = Var[-\sigma \int_0^t \int_0^s e^{k(u-s)} dW_u ds] = \sigma^2 Var[\int_0^t \int_0^s e^{k(u-s)} dW_u ds]$로 표현할 수 있습니다.

해당 이중적분식은 푸비니정리에 따라 $\int_0^t \int_0^s e^{k(u-s)} dW_u ds = \int_0^t \int_u^t e^{k(u-s)} ds dW_u = \int_0^t \frac{e^{k(u-t)}-1}{-k} dW_u$ 입니다.

따라서, $Var[\int_0^t \int_0^s e^{k(u-s)} dW_u ds] = Var[\int_0^t \frac{e^{k(u-t)}-1}{-k} dW_u]$ 이고, $Var[X] = E[X^2] - E[X]^2$ 임을 적용하면,

$$Var[\int_0^t \frac{e^{k(u-t)}-1}{-k} dW_u] = E[(\int_0^t \frac{e^{k(u-t)}-1}{-k} dW_u)^2] - E[\int_0^t \frac{e^{k(u-t)}-1}{-k} dW_u]^2 = E[(\int_0^t \frac{e^{k(u-t)}-1}{-k} dW_u)^2]$$

$dW_t^2 = dt$ 이므로, $E[(\int_0^t \frac{e^{k(u-t)}-1}{-k} dW_u)^2] = E[\int_0^t (\frac{e^{k(u-t)}-1}{-k})^2 du] = \int_0^t \frac{e^{2k(u-t)}-2e^{k(u-t)}+1}{k^2} du$

$\Rightarrow = \frac{1}{k^2}(\frac{1-e^{-2kt}}{2k} - \frac{2(1-e^{-kt})}{k} + t)$

정리하면, R_t 의 분산 및 확률분포는 아래와 같습니다.

$$Var[R_t] = \sigma^2 Var[\int_0^t \int_0^s e^{k(u-s)} dW_u ds] = \frac{\sigma^2}{k^2}(\frac{1-e^{-2kt}}{2k} - \frac{2(1-e^{-kt})}{k} + t)$$

$$R_t \sim N\left(\frac{r_0(e^{-kt}-1)}{k} - t\theta - \frac{\theta(e^{-kt}-1)}{k},\ \frac{\sigma^2}{k^2}(\frac{1-e^{-2kt}}{2k} - \frac{2(1-e^{-kt})}{k} + t) \right)$$

(2) $d(t) = E[e^{-\int_0^t r_s ds}]$ **유도**

우리는 정규분포를 따르는 확률변수 X에 대해, $E[e^X] = e^{\mu + \frac{1}{2}\sigma^2}$ 임을 알고 있습니다.

따라서 $d(t) = E[e^{-\int_0^t r_s ds}] = E[e^{R_t}] = e^{E[R_t] + \frac{1}{2}Var[R_t]}$ 이므로, R_t가 정규분포를 따르고 평균과 분산을 알고있으면 $d(t)$를 알 수 있습니다.

위 결과를 통해, $d(t) = e^{E[R_t] + \frac{1}{2}Var[R_t]}$를 정리하겠습니다.

$$E[R_t] + \frac{1}{2}Var[R_t] = \frac{r_0(e^{-kt}-1)}{k} - t\theta - \frac{\theta(e^{-kt}-1)}{k} + \frac{1}{2}\left(\frac{\sigma^2}{k^2}(\frac{1-e^{-2kt}}{2k} - \frac{2(1-e^{-kt})}{k} + t)\right)$$

여기서, $B(t) = \frac{1-e^{-kt}}{k}$ 라고 하면,

$$= -B(t)r_0 - (t - B(t))\theta + \frac{\sigma^2}{2k^2}\left(t - 2B(t) + \frac{(1+e^{-kt})(1-e^{-kt})}{2k}\right)$$

$B(t) = \frac{1-e^{-kt}}{k}$ 에서, $e^{-kt} = 1 - kB(t)$ 이므로,

$$= -B(t)r_0 - (t - B(t))\theta + \frac{\sigma^2}{2k^2}\left(t - 2B(t) + \frac{1}{2}(2 - kB(t))B(t)\right)$$

$$= -B(t)r_0 - (t - B(t))\theta + \frac{\sigma^2}{2k^2}\left(t - B(t) - \frac{kB(t)^2}{2}\right)$$

$$= -B(t)r_0 - (t - B(t))\theta + \frac{\sigma^2}{2k^2}(t - B(t)) - \frac{\sigma^2}{2k^2}(\frac{kB(t)^2}{2})$$

$$= -B(t)r_0 + (B(t) - t)\theta - \frac{\sigma^2}{2k^2}(B(t) - t) - \frac{(\sigma B(t))^2}{4k}$$

$$= -B(t)r_0 + (B(t) - t)\left(\theta - \frac{\sigma^2}{2k^2}\right) - \frac{(\sigma B(t))^2}{4k}$$

$$\therefore d(t) = A(t)e^{-B(t)r_0} \quad where \quad B(t) = \frac{1 - e^{-kt}}{k}, \quad A(t) = e^{(B(t)-t)\left(\theta - \frac{\sigma^2}{2k^2}\right) - \frac{(\sigma B(t))^2}{4k}}$$

Problem 4.

4. 위의 문제에서 얻어진 해를 이용하여 앞에서 주어진 7개의 무이표 채권들의 모델 가격 $100\mathsf{d}^{\text{vasicek}}(.5)$ 부터 $100\mathsf{d}^{\text{vasicek}}(3.5)$ 까지 구할 수 있다. 모수값 r_0, σ 가 1번 문제와 같을 때, SSE 를 최소화하는 k, θ 의 값을 구하시오.

Answer

1번의 모수값과 다양한 k, θ의 초기값 조합을 통해 SSE를 최소화시키는 최적화를 반복수행하겠습니다.

```
library(tidyverse)
price_market <- tibble(maturity=seq(0.5,3.5,0.5),
                       price=c(99.1338, 97.8925, 96.1462, 94.1011,
                               91.7136, 89.2258, 86.8142))


r0=0.0173
sigma=0.0173
```

R 코드로 초기값 조합에 따른 최적화를 300회 수행한 결과,

SSE를 최소화시키는 $k = 0.0134$, $\theta = 1.05$이며, 이때의 SSE는 0.19(액면 100원 기준) 및 초기값은 $k = 1$, $\theta = 0.05$ 입니다.

```
vasicek_discount <- function(r0,sigma,k,theta,t){
  B=(1-exp(-k*t))/k
  A=exp((B-t)*(theta-sigma^2/(2*k^2))-(sigma*B)^2/(4*k))

  vasicek=A*exp(-B*r0)
  return(vasicek)
}

SSE_vasicek <- function(params){
  SSE=(vasicek_discount(0.0173,0.0173,params[1],params[2],price_market$maturity)*100-price_mar
  return(SSE)
}

montecarlo <- tibble(k0=double(),theta0=double(),k=double(),theta=double(),SSE=double())

for(k in seq(0.1,3,0.1)){
  for(theta in seq(0.01,0.1,0.01)){
    tmp <- optim(par=c(k,theta),SSE_vasicek)
    tmp_tibble <- tibble(k0=k,theta0=theta,k=tmp$par[1],theta=tmp$par[2],SSE=tmp$value)
    montecarlo <- montecarlo %>% bind_rows(tmp_tibble)
  }
}

montecarlo %>% arrange(SSE) %>% slice(1:5)
```

```
# A tibble: 5 x 5
     k0 theta0      k theta   SSE
  <dbl>  <dbl>  <dbl> <dbl> <dbl>
1   0.1   0.05 0.0134 1.05  0.190
2   0.1   0.03 0.0160 0.889 0.191
3   0.1   0.07 0.0171 0.833 0.192
4   0.2   0.02 0.0174 0.818 0.192
5   0.1   0.06 0.0181 0.788 0.192
```

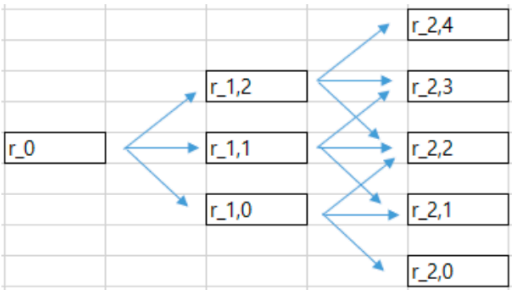
Problem 5.

5. Vasicek 모형을 트리로 구현할 때, 우리는 이항트리가 아닌 삼항트리를 이용할 수 있다. 각 노드 $r_{i,j}$ 에서 아래와 같이 세 개의 노드로 분기할 수 있고 이들은 $r_{i+1,j}, r_{i+1,j+1}, r_{i+1,j+2}$ 로 나타낼 수 있다. 그리고 $r_{i+1,j}$ 로 갈 확률 p_d, $r_{i+1,j+1}$ 로 갈 확률 p_m, $r_{i+1,j+2}$ 로 갈 확률 p_u 이 있다. 이들은 물론 양수이며 $p_u + p_m + p_d = 1$을 만족한다. 각 시점에서 가능한 금리의 값들을 정하는 방법은 다양하다. 그 중 하나의 방법으로 다음을 고려할 수 있다.

   우선 어떤 정수 l 에 대하여 $r_{i,i} = \theta + l\square$, $\square^2 = 3\frac{\sigma^2}{2k}\left[1 - e^{-2k\Delta}\right]$ 로 두자. 여기에서 Δ 는 시점간 간격을 의미하고 $\square = \sqrt{3\mathsf{Var}(r_{i+1}|r_i)}$ 이다. 이 정수 l 은 $\mathsf{E}[r_i|r_0]$ 값이 $r_{i,i}$ 에 가장 가깝도록 정해진다. 즉 $l = \mathsf{round}\left(\frac{\mathsf{E}[r_i|r_0]-\theta}{\square}\right) = \mathsf{round}\left(\frac{(r_0-\theta)e^{-k\Delta i}}{\square}\right)$.

   위와 같이 정해진 $r_{i,i}$ 에 대하여 우리는 $r_{i,j} = r_{i,i} + (j-i)\square$ 로 각 노드 $j = 0.1, \ldots, 2i$ 를 정의할 수 있다. 이 작업은 각 $i = 1, 2, \ldots$ 에 대해 이루어진다. 남은 작업은 주어진 노드 $r_{i,j}$ 에서의 분기확률 p_u, p_m, p_d 를 정하는 것이다. $\mathsf{E}[r_{i+1}|r_{i,j}]$ 와 $\mathsf{Var}(r_{i+1}|r_{i,j})$ 를 매칭할 수 있도록 이 값들을 정하시오. 보다 정확히 p_u, p_m, p_d 를 ξ 와 $\square$ 로 표현하시오. 여기에서 ξ 는 다음으로 정의된다:

$$\xi = \mathsf{E}[r_{i+1}|r_{i,j}] - r_{i+1,j+1}.$$



Answer

먼저, 문제 3에서 Vasicek모형의 short rate에 대한 확률분포를 아래와 같이 유도하였습니다.

$$r_t \sim N\left(r_0 e^{-kt} + \theta(1 - e^{-kt}),\ \frac{\sigma^2}{2k}(1 - e^{-2kt}) \right)$$

이를 이용하면 시간간격이 Δt로 주어진 r_i에 대하여, $Var[r_{i+1}|r_i] = \frac{\sigma^2}{2k}(1 - e^{-2k\Delta t})$ 임을 알 수 있습니다.

즉, 문제에서 주어진 $\square^2 = 3\frac{\sigma^2}{2k}(1 - e^{-2k\Delta}) = 3Var[r_{i+1}|r_i]$ 이며, 이는 $\square$가 Vasicek trinomial tree에서 상승 및 하락폭임을 의미합니다.

따라서 문제에서 주어진 trinomial node에서 상승/보합/하락시의 short rate는 다음과 같습니다.

$$r_{i+1,u} = r_{i+1,j+2} = r_{i+1,j+1} + \square,\ r_{i+1,m} = r_{i+1,j+1},\ r_{i+1,d} = r_{i+1,j} = r_{i+1.j+1} - \square$$

한편, trinomial node에서 상승/보합/하락확률이 각각 p_u, p_m, p_d이므로 상승/보합/하락시의 short rate의 평균은 $E[r_{i+1}|r_i]$와 같아야하고, 분산 역시 $Var[r_{i+1}|r_i] = \square^2/3$과 같아야합니다.

이러한 평균과 분산의 equilibrium을 수식으로 표현하면 아래와 같습니다.

1. $E[r_{i+1}|r_{i,j}] = r_i e^{-k\Delta t} + \theta(1 - e^{-k\Delta t}) = p_u r_{i+1,j+2} + p_m r_{i+1,j+1} + p_d r_{i+1,j}$

2. $Var[r_{i+1}|r_{i,j}] = \frac{\sigma^2}{2k}(1 - e^{-2k\Delta t}) = p_u(r_{i+1,j+2} - E[r_{i+1}|r_i])^2 + p_m(r_{i+1,j+1} - E[r_{i+1}|r_i])^2 + p_d(r_{i+1,j} - E[r_{i+1}|r_i])^2$

이제, 두 식을 만족하는 $p_{u,m,d}$를 구하도록 하겠습니다.

Trinomial node의 변동폭, $\xi = E[r_{i+1}|r_{i,j}] - r_{i+1,j+1}$ 및 $p_u + p_m + p_d = 1$임을 이용하면,

1. $E[r_{i+1}|r_{i,j}] = p_u(r_{i+1,j+1}+\square)+p_m r_{i+1,j+1}+p_d(r_{i+1,j+1}-\square) = r_{i+1,j+1}+(p_u-p_d)\square \;\Rightarrow\; p_u-p_d = \frac{\xi}{\square}$...(1)

2. $Var[r_{i+1}|r_{i,j}] = p_u(-\xi+\square)^2+p_m(-\xi)^2+p_d(-\xi-\square)^2 = \xi^2-2\xi\square(p_u-p_d)+\square^2(p_u+p_d)\left(=\frac{\square^2}{3}\right)$...(2)

이제, $p_{u,m,d}$를 $\square$, ξ로 표현할 수 있습니다.

$$(1)\ p_u - p_d = \frac{\xi}{\square}\ \ \&\ \ (2)\ \xi^2 - 2\xi\square(p_u - p_d) + \square^2(p_u + p_d) = \frac{\square^2}{3}\ \ \&\ \ (3)\ p_u + p_m + p_d = 1$$

(1), (3)을 이용하면, (2) $\xi^2 - 2\xi\square(\frac{\xi}{\square}) + \square^2(1 - p_m) = \frac{\square^2}{3}$

$$\therefore p_m = \frac{2}{3} - \frac{\xi^2}{\square^2}$$

(3)+(1)에서, $2p_u + p_m = 1 + \frac{\xi}{\square}$

$$\therefore p_u = \frac{1}{2} + \frac{\xi}{2\square} - \frac{p_m = \frac{2}{3} - \frac{\xi^2}{\square^2}}{2} = \frac{1}{6} + \frac{\xi(\xi+\square)}{2\square^2}$$

(3)-(1)에서, $2p_d + p_m = 1 - \frac{\xi}{\square}$

$$\therefore p_u = \frac{1}{2} - \frac{\xi}{2\square} - \frac{p_m = \frac{2}{3} - \frac{\xi^2}{\square^2}}{2} = \frac{1}{6} + \frac{\xi(\xi-\square)}{2\square^2}$$

Part III

투자분석('24봄)

투자분석 과제1

Question

You are a stock analyst and want to evaluate stocks. You may select any 3 stocks and get information from data source you prefer. You need a risk-free rate and market risk premium for this assignment. Choose appropriate risk-free rate such as 3-month T-bill rate or 1-month LIBOR and risk premium for the market portfolio. Use the constant growth dividend discount model. You may use any computer tool but do not hand in the data or the program code.

a) What are the names of stocks you choose?
b) Specify risk-free rate, risk premium and the data source.
c) What are the required rates of return? Use the CAPM. You need to specify firm's beta and the expected rate of return of the market index portfolio.
d) Calculate present value of growth opportunity (PVGO) for each stock.
e) Find the intrinsic value (0) and 1-year value (1) for each stock.
f) Find the expected rate of return for each stock, i.e., (1 − 0)/ 0 where 0 is the current market price.
g) Based on the calculation above, which stock do you buy or sell? Explain briefly

문제풀이

(a) 주식 선정

저는 ***CME group*(*CME*)**, ***ICE*(*ICE*)**, ***Nasdaq*(*NDAQ*)** 세가지 종목을 선정하였습니다.

선정 배경으로는,

(1) 제가 거래소 산업에 관심이 많고,
(2) 세 주식 모두 미국에 상장되어있는 대표적인 글로벌 거래소이며,
(3) 동일한 거래소 산업이고 S&P500지수의 구성종목이라 동일 선상에서 비교하기 적합할 것으로 보이기 때문입니다.

관련 자료는 Yahoo Finance를 참조하였습니다.

(b) 무위험이자율 및 리스크프리미엄

risk-free rate는 최근 미국에서 대체지표금리로 선정한 SOFR기반의 금리를 누적하여 3개월 금리로 환산한 3-month SOFR Average를 risk-free rate로 선정하였습니다.

NewYork Fed : SOFR Average

각 주식과 시장(S&P500)의 리스크프리미엄은 과거 3년간('21~'23)의 연평균수익률 및 무위험이자율을 이용하여 산출할 예정입니다.

각 데이터를 취합하여 정리한 결과 및 무위험이자율, 리스크프리미엄의 산출은 아래와 같습니다.

```
# import data&library
rm(list = ls())
library(tidyverse)
nasdaq <- read_csv("investment_hw/투자분석_NDAQ.csv")
cme <- read_csv("investment_hw/투자분석_CME.csv")
ice <- read_csv("investment_hw/투자분석_ICE.csv")
rf <- read_csv("investment_hw/투자분석_sofrai.csv")
market <- read_csv("investment_hw/투자분석_snp500.csv")

# tidy data
rf2 <- rf %>%
  filter(`90-Day Average SOFR` %>% is.na()==FALSE) %>%
  mutate(date=paste0(substr(`Effective Date`,7,10),
                     substr(`Effective Date`,1,2),
                     substr(`Effective Date`,4,5)) %>% as.integer(),
         year=substr(`Effective Date`,7,10),
         month=substr(`Effective Date`,1,2),
         group="rfr",
         price=`90-Day Average SOFR`) %>%
  select(year,month,date,group,price)

# Data Pre-processing
tidydata <- tibble()
tidydata <- nasdaq %>%
  mutate(group="nasdaq") %>%
  union_all(cme %>% mutate(group="cme")) %>%
```

```
  union_all(ice %>% mutate(group="ice")) %>%
  union_all(market %>% mutate(group="market")) %>%
  mutate(date=paste0(substr(Date,1,4),
                     substr(Date,6,7),
                     substr(Date,9,10)) %>% as.integer(),
         year=substr(Date,1,4),
         month=substr(Date,6,7),
         price=`Adj Close`) %>%
  select(year,month,date,group,price) %>%
  union_all(rf2) %>%
  arrange(date,group)

# Use daily adj. closing price, if rfr is NA then use yesterday rfr.
tidydata_wide <- tidydata %>%
  pivot_wider(names_from = "group", values_from = "price") %>%
  filter(nasdaq %>% is.na()==FALSE,
         cme %>% is.na()==FALSE,
         ice %>% is.na()==FALSE,
         market %>% is.na()==FALSE) %>%
  mutate(rfr=if_else(rfr %>% is.na(),lag(rfr)/100,rfr/100)) %>%
  mutate(cme_yield=log(cme)-log(lag(cme)),
         ice_yield=log(ice)-log(lag(ice)),
         nasdaq_yield=log(nasdaq)-log(lag(nasdaq)),
         market_yield=log(market)-log(lag(market))) %>%
  filter(cme_yield %>% is.na()==FALSE)
```

```
rf3 <- tidydata_wide %>%
  filter(year!=2020&year!=2024,as.integer(month)%%3==0) %>%
  arrange(date %>% desc()) %>%
  group_by(year,month) %>%
  slice(1) %>%
  ungroup() %>%
  summarise(rfr_3year=mean(rfr))

tidydata_wide %>% filter(date==20231229) %>% select(rfr) %>% bind_cols(rf3)
```

```
# A tibble: 1 x 2
     rfr rfr_3year
   <dbl>     <dbl>
1 0.0536    0.0223
```

```r
tidydata_wide %>%
  filter(year!=2020&year!=2024) %>%
  summarise(cme_premium=sum(cme_yield)/3-rf3$rfr_3year,
            ice_premium=sum(ice_yield)/3-rf3$rfr_3year,
            nasdaq_premium=sum(nasdaq_yield)/3-rf3$rfr_3year,
            market_premium=sum(market_yield)/3-rf3$rfr_3year)
```

```
# A tibble: 1 x 4
  cme_premium ice_premium nasdaq_premium market_premium
        <dbl>       <dbl>          <dbl>          <dbl>
1      0.0679      0.0272         0.0826         0.0573
```

무위험이자율은 기간말 기준 5.36%, 과거 3개년 동안 무위험이자율에 투자한 연평균수익률은 약 2.23%, 과거 3개년 동안 실현된 각 주식과 시장의 연환산 리스크프리미엄은 아래 표와 같습니다.

구 분	S&P500	CME	ICE	NASDAQ
연수익률	7.96%	9.03%%	4.95%	10.49%
프리미엄	5.74%	6.81%%	2.73%	8.27%

⚠ Excess return vs. Risk premium

초과수익률은 과거의 값을 바탕으로 산출되며, 리스크프리미엄은 기대수익률을 기반으로 산출.
위의 풀이는 초과수익률을 산출하였으므로 오답이며, 리스크프리미엄은 CAPM 등의 기대수익률을 가지고와서 산출해야 함.

$$Excess\ return = r - r_f$$

$$Risk\ premium = E[r] - r_f$$

(c) CAPM

각 종목의 베타를 산출하기 위해 필요한 파라미터는 아래와 같습니다.

- 무위험이자율, R_f : 기간말('23.12월말)의 3-month SOFR average
- 시장수익률, R_m : S&P500지수의 수익률
- 개별수익률, R_{stock} : 개별 주식의 수익률

수익률은 일간/월간 두가지를 사용하여 비교할 예정이며, 회귀분석에 사용한 값은 아래와 같습니다.

- 기간 : (일별) 2023년 250거래일, (월간) 2021년~2023년 36개월
- 산출방법 : (일별) 전일 종가 대비 당일 종가의 로그수익률, (월별) 이전달 말 종가 대비 이번달 말 종가의 로그수익률

마지막으로, 베타를 이용하여 기대수익률을 산출할 때 이용한 R_m은 다음과 같습니다.

- (일별) 250일간 S&P500지수 일평균수익률의 연환산(=연수익률)
- (월별) 36개월간 S&P500지수 월평균수익률의 연환산(=3개년수익률/3)

> **Warning**
>
> 로그수익률(연속복리수익률)을 사용하였으므로 보유기간수익률로 표현할 때 단순 덧셈만으로 환산할 수 있습니다.
> 또한, 기간 중 배당 및 CA로 인한 차이는 수정주가(Adj. close price)를 사용해서 보정하였습니다.

이에 따라, R코드로 산출한 개별종목별 **베타 / 기대수익률 / 산출기간동안 실현수익률**은 아래와 같습니다.

```
# calcuate beta, required return


# daily beta in '23
dailybeta <- tidydata_wide %>%
  filter(year=="2023") %>%
  summarise(beta_cme=cov(market_yield,cme_yield)/var(market_yield),
            beta_ice=cov(market_yield,ice_yield)/var(market_yield),
            beta_nasdaq=cov(market_yield,nasdaq_yield)/var(market_yield),
            market_avg_yield=sum(market_yield),
            cme_avg_yield=sum(cme_yield),
            ice_avg_yield=sum(ice_yield),
            nasdaq_avg_yield=sum(nasdaq_yield)) %>%
  mutate(group="daily")


# monthly beta
monthlybeta <- tidydata_wide %>%
  arrange(year,month,date %>% desc) %>%
  group_by(year,month) %>%
```

```
  slice(1) %>%
  ungroup() %>%
  mutate(cme_yield=log(cme)-log(lag(cme)),
         ice_yield=log(ice)-log(lag(ice)),
         nasdaq_yield=log(nasdaq)-log(lag(nasdaq)),
         market_yield=log(market)-log(lag(market))) %>%
  filter(cme_yield %>% is.na()==FALSE) %>%
  filter(year!="2024",year!="2020") %>%
  summarise(beta_cme=cov(market_yield,cme_yield)/var(market_yield),
            beta_ice=cov(market_yield,ice_yield)/var(market_yield),
            beta_nasdaq=cov(market_yield,nasdaq_yield)/var(market_yield),
            market_avg_yield=sum(market_yield)/3,
            cme_avg_yield=sum(cme_yield)/3,
            ice_avg_yield=sum(ice_yield)/3,
            nasdaq_avg_yield=sum(nasdaq_yield)/3) %>%
  mutate(group="monthly")

beta <- tibble()
beta <- dailybeta %>%
  union_all(monthlybeta) %>%
  mutate(tidydata_wide %>% filter(date==20231229) %>% select(rfr)) %>%
  select(group,market_avg_yield,cme_avg_yield,ice_avg_yield,nasdaq_avg_yield,
         rfr,beta_cme,beta_ice,beta_nasdaq)

capm <- tibble()
capm <- beta %>%
  pivot_longer(cols = c("beta_cme","beta_ice","beta_nasdaq"),
               names_to = "stock", values_to = "beta") %>%
  mutate(expected_return=beta*(market_avg_yield-rfr)+rfr) %>%
  mutate(stock=substr(stock,6,nchar(stock))) %>%
  mutate(realized_return=if_else(stock=="cme",cme_avg_yield,
                                 if_else(stock=="ice",ice_avg_yield,nasdaq_avg_yield))) %>%
  select(group,stock,beta,expected_return,realized_return,market_avg_yield,rfr)

capm
```

```
# A tibble: 6 x 7
  group   stock   beta expected_return realized_return market_avg_yield    rfr
  <chr>   <chr>  <dbl>           <dbl>           <dbl>            <dbl>  <dbl>
1 daily   cme    0.315          0.105           0.272            0.217  0.0536
2 daily   ice    0.828          0.189           0.240            0.217  0.0536
3 daily   nasdaq 0.866          0.195          -0.0373           0.217  0.0536
4 monthly cme    0.353          0.0628          0.0903           0.0796 0.0536
5 monthly ice    1.12           0.0829          0.0495           0.0796 0.0536
6 monthly nasdaq 0.993          0.0795          0.105            0.0796 0.0536
```

일별/월별로 구분한 각 주식의 베타와 기대수익률입니다.

보다 일반적이고 평탄화된 결과를 위해 월별 데이터를 중심으로 설명하겠습니다.

먼저 **각 주식의 베타는 CME:0.35, ICE:1.12, Nasdaq 0.99**입니다. 이는 거래소산업이 특성상 변동성이 크지 않고 경제상황에 민감하지 않아 베타가 시장포트폴리오와 유사한 1 수준에서 형성되는 것으로 보입니다. 특히, CME는 세계 최대의 파생상품거래소로서 규모가 매우 큰 고배당 우량주인 특징이 있어 변동성이 낮아 베타가 0.35로 낮게 형성된 것으로 추정됩니다.

베타 산출방법

베타는 $R_{stock,t} = \alpha + \beta R_{m,t} + \epsilon$의 선형회귀분석 결과로 산출되는 계수입니다.
즉, $\beta = \rho_{stock,m} \frac{\sigma_{stock}}{\sigma_m} = \frac{Cov_{stock,m}}{\sigma_m^2}$로 산출할 수 있습니다.

월별수익률을 산출한 과거 3년(2021~2023)간 평균월별수익률을 연율화하면 로그수익률 특성상 지난 3년간 연평균수익률과 같습니다. 즉, 시장포트폴리오의 기대수익률은 약 7.96%라고 할 수 있으며, 무위험이자율은 2023.12월말 값인 5.36%를 사용합니다.

이제, $R_{stock} = \beta(R_m - R_f) + R_f$를 통해 각 주식의 요구수익률을 산출할 수 있습니다.

구 분	CME	ICE	NASDAQ
Beta	0.35	1.12	0.99
Required Return	6.28%	8.29%	7.95%

(d) PVGO

먼저, 거래소산업은 성장주보다는 우량주 성향이 강하고, 배당성향이 다소 높고 정기적인 배당을 하는 경향이 있습니다.. 따라서 배당할인모형(Constant-growth DDM)을 사용할 예정입니다.

먼저, 기업의 내재가치에 배당할인모형을 적용한 PVGO의 산식은 아래와 같습니다.

$$PVGO = V_0 - \frac{E_1}{k} = \frac{D_1}{k-g} - \frac{E_1}{k}$$

여기서, 요구수익률 k는 (c)에서 CAPM으로 구한 요구수익률을 사용할 예정이며, D_1, E_1, g를 구하기 위해 직전주당배당(D_0), 직전 주당수익(E_0), 직전재투자성향($b_0 = \frac{E_0 - D_0}{E_0}$), 직전 자기자본수익률 ROE_0를 이용하여 다음과 같이 산출하도록 하겠습니다.

$$g = b_0 \times ROE_0 = \frac{E_0 - D_0}{E_0} \times ROE_0, \quad D_1 = D_0(1+g), \quad E_1 = E_0(1+g)$$

$$\Rightarrow PVGO = \frac{D_0(1+g)}{k-g} - \frac{E_0(1+g)}{k}$$

> **주당순이익 산출 : DDM**
>
> 다음 년도의 주당순이익 E_1은 올해의 순이익 E_0에 배당할인모형의 배당성장률 g를 반영하여 산출하였습니다.

이제, 각 주식의 재무상태표를 참조하여 D_0, E_0, b_0, ROE_0를 산출하겠습니다. 다만, 직전사업연도(2023)는 시장벤치마크(S&P500)의 수익률이 약 21%로, 과거 평균치의 약 3배를 상회하고 있어 각 기업의 EPS, DPS 등이 평소 수치보다 고평가되었을 가능성이 있습니다. 이를 감안하여 D_0, E_0 산출시에는 과거 3개년 당기순이익, 현금배당금의 산술평균을 사용할 것이며 ROE_0는 과거 3개년의 기하평균을 사용하겠습니다.

각 값들은 Yahoo Finance 및 Stockrow를 참조하였습니다.

```
financial <- read_csv("investment_hw/투자분석_financialstatement.csv")
```

```
# DDM 이용
financial <- financial %>%
  group_by(stock) %>%
  summarise(eps=sum(earning)/sum(shares),
            dps=sum(dividend)/sum(shares),
            bps=sum(capital)/sum(shares),
            roe=prod(1+roe)^(1/3)-1) %>%
  mutate(reinvestment=(eps-dps)/eps) %>%
  mutate(growth=roe*reinvestment) %>%
  mutate(d1=dps*(1+growth),
         e1=eps*(1+growth)) %>%
  left_join(.,capm %>% filter(group=="monthly") %>% select(stock,expected_return,rfr),
```

```
            by="stock") %>%
  mutate(v0_ddm=d1/(expected_return-growth)) %>%
  mutate(pvgo_ddm=v0_ddm-e1/expected_return)
financial
```

```
# A tibble: 3 x 13
  stock   eps   dps   bps   roe reinvestment  growth    d1    e1 expected_return
  <chr> <dbl> <dbl> <dbl> <dbl>        <dbl>   <dbl> <dbl> <dbl>           <dbl>
1 cme    7.86 7.49   75.3 0.105       0.0475 0.00497 7.53   7.90          0.0628
2 ice    4.67 1.52   42.3 0.116       0.675  0.0786  1.64   5.04          0.0829
3 nasd~  2.25 0.785  15.6 0.163       0.652  0.106   0.868  2.49          0.0795
# i 3 more variables: rfr <dbl>, v0_ddm <dbl>, pvgo_ddm <dbl>
```

산출 결과는 아래와 같습니다.

구 분	CME	ICE	NASDAQ
E_0	7.86	4.67	2.25
D_0	7.49	1.52	0.79
ROE_0	10.5%	11.6%	16.3%
b_0	4.7%	67.5%	65.2%
g	0.5%	7.9%	10.6%
D_1	7.53	1.64	0.87
E_1	7.90	5.04	2.49
k	6.28%	8.29%	7.95%
V_0	130.15	380.07	-32.82(NaN)
$PVGO$	4.32	319.28	-64.19(NaN)

배당할인모형을 사용하여 평가해본 결과, **PVGO는 CME가 4.32$, ICE가 319.28$, NASDAQ은 -64.19$로** 산출되었습니다. 여기서 **NASDAQ 거래소**의 경우 **기업의 요구수익률보다 성장률이 높아**($k < g$) 기업의 내재가치(V_0) 평가에 **배당할인모형을 사용할 수 없는 케이스**입니다. 즉, **PVGO가 정상적으로 산출되지 않았습니다.**

NASDAQ 거래소의 요구수익률은 7.95% 및 성장률은 10.6%로 산출되었는데, 현재시점에서 배당할인모형을 통해 이 주식을 평가한다는 것은 기업에 요구되는 수익률보다 성장하는 속도가 높아서 기업의 미래 배당가치가 발산($\to \infty$)한다는 것을 의미합니다. 이러한 이유가 발생하는 이유를 추정해보면, 아마도 NASDAQ의 현재주가가 저평가되어있거나, 최근 성과가 좋아서 단기적으로 ROE_0가 높게 산출되었거나, CAPM의 요구수익률이 실제 주식의 요구수익률보다 낮기 때문일 것 입니다.

NASDAQ 거래소의 과거 추이를 볼 때, 최근 3년간 실현수익이 연평균 약 10.5%로 요구수익률보다 높고 성장률과 유사하므로 CAPM의 요구수익률이 실제 주식의 요구수익률을 정확하게 평가하지 못한 것이 주된 이유일 것으로 보입니다.

이러한 문제점은,

(1) CAPM 대신 APT 등의 다른 모형으로 요구수익률(k)을 평가해보거나,

(2) 배당할인모형(DDM) 대신 다른 모형을 이용함으로써 해결할 수 있습니다.

> **CAPM의 한계**
>
> CAPM으로 산출한 요구수익률 k는 개별주식의 여러 특성을 정확히 반영하지 못하였을 가능성이 있습니다. CAPM은 시장에 대한 민감도 β와 시장포트폴리오의 리스크 대비 수익률($R_m - R_f$)을 이용해 주가의 기대수익률(요구수익률)을 평가하는데, 이 한가지 요인(factor)으로 기대수익률이 결정된다고 가정하기 때문입니다.

Multi-factor 모형은 배우기 이전이므로 수업시간에 배운 **초과이익모형(Residual Income Model)을 이용하여 세 주식의 V_0, $PVGO$를 다시 평가**해보도록 하겠습니다.

먼저, RIM의 산식은 다음과 같습니다.

$$V_0 = BE_0 + \sum_{i=1}^{\infty} \frac{E[RE_i]}{(1+k)^i}$$

여기서 BE_0은 주당순자산으로 D_0, E_0과 같이 과거 3개년 평균치를 사용하고, 초과이익의 기대값 $E[RE_i]$는 i년도의 주당순이익(E_i)-주당배당금(D_i)으로 결정되며, 주당순이익은 $E_i = ROE_0 \times BE_{i-1}$, 주당배당금은 $k \times BE_{i-1}$로 산출, $BE_i = BE_{i-1} \times (1 + R_f)$(즉, 초과이익이 무위험이자율만큼 성장)하도록 하겠습니다.

$$V_0 = BE_0 + \sum_{i=1}^{\infty} \frac{(ROE_0 - k)BE_0(1+r_f)^i}{(1+k)^i} = BE_0 + \frac{(ROE_0 - k)BE_0}{k - r_f}$$

위와 같이 RIM으로 기업의 내재가치를 산출할 수 있으며, 산출결과는 아래와 같습니다.

```
financial_rim <- financial %>%
  mutate(v0_rim=bps+(roe-expected_return)*bps/(expected_return-rfr)) %>%
  mutate(pvgo_rim=v0_rim-(eps*(1+rfr))/expected_return) %>%
  select(stock,bps,roe,rfr,expected_return,v0_rim,pvgo_rim)

financial_rim
```

```
# A tibble: 3 x 7
  stock    bps   roe    rfr expected_return v0_rim pvgo_rim
```

```
  <chr>  <dbl> <dbl>  <dbl>           <dbl>  <dbl>    <dbl>
1 cme     75.3 0.105 0.0536          0.0628  417.     285.
2 ice     42.3 0.116 0.0536          0.0829   90.5     31.2
3 nasdaq  15.6 0.163 0.0536          0.0795   65.8     35.9
```

> 주당순이익 산출 : RIM
>
> 다음 년도의 주당순이익 E_1은 올해의 순이익 E_0에 초과이익성장률인 무위험이자율만큼 가산하여 산출하였습니다.

구 분	CME	ICE	NASDAQ
$V_0,\ DDM$	130.15	380.07	-
$PVGO,\ DDM$	4.32	319.28	-
$V_0,\ RIM$	417.10	90.55	65.81
$PVGO,\ RIM$	291.86	34.20	37.43

RIM 방식으로 내재가치를 평가한 결과, **PVGO의 값은 CME가 291.86$, ICE가 34.20$, NASDAQ에서 37.43$로 산출**되었습니다.

(e) Find $V_0,\ V_1$

제가 사용한 RIM 산출식에서 $V_1 = V_0(1 + r_f)$입니다. (DDM은 $V_1 = V_0(1 + g)$)

NASDAQ에서 DDM은 사용할 수 없으므로 RIM을 기준으로 산출하겠습니다.

```
financial_rim %>%
  mutate(v1_rim=v0_rim*(1+rfr)) %>%
  select(stock,v0_rim,v1_rim)
```

```
# A tibble: 3 x 3
  stock  v0_rim v1_rim
  <chr>   <dbl>  <dbl>
1 cme     417.   439.
2 ice      90.5   95.4
3 nasdaq   65.8   69.3
```

기존에 산출하였던 V_0와 산출한 V_1은 아래와 같습니다.

구 분	CME	ICE	NASDAQ
$V_0,\ RIM$	417.10	90.55	65.81
$V_1,\ RIM$	439.44	95.40	69.33

(f) Find the expected rate of return for each stock

```
stock_return <- financial_rim %>%
  mutate(v1_rim=v0_rim*(1+rfr)) %>%
  select(stock,v1_rim) %>%
  left_join(.,tidydata %>% filter(date==20240228) %>% select(group,price),
            by=c('stock'='group')) %>%
  mutate(return=(v1_rim-price)/price)
stock_return
```

```
# A tibble: 3 x 4
  stock  v1_rim price return
  <chr>   <dbl> <dbl>  <dbl>
1 cme     439.  220.   1.00
2 ice      95.4 138.  -0.311
3 nasdaq   69.3  56.1  0.236
```

2월말 기준 각 주식의 종가를 기준으로 계산한 **연환산 기대수익률**은 다음과 같습니다.

구 분	CME	ICE	NASDAQ
$V_1,\ RIM$	439.44	95.40	69.33
$ClosePrice$	219.68	138.39	56.11
$ExpectedReturn,\ annum$	100.03%	-31.06%	23.55%

(g) Which stock is the best?

NASDAQ exchange 주식

단순히 (f)의 기대수익률로 볼 때는 CME가 가장 높지만, 이는 BPS가 높고 배당성향이 1에 가까운 CME 거래소의 특성상 RIM 방식에서 높은 내재가치를 평가받은 것으로 보입니다. 이는 DDM 방식으로 평가하였을 때, 성장률 g가 매우 낮아 내재가치가 다소 낮게 산출된 점에서 잘 드러납니다.

또한, CME는 시장 베타도 0.35수준으로 낮아 절대적인 요구수익률이 높지 않고 변동성도 낮아 흔히 말하는 재미없는 주식입니다. 장기적인 투자를 선호하고 자본금이 많은 기관투자자 등이 주로 투자하는 주식일 것으로 예상됩니다.

반면에 NASDAQ은 RIM방식으로 평가한 기대수익률이 연 23.55% 수준으로 양호하고, 시장 베타도 거의 1에 가까워 적당한 변동성을 가지고 있습니다. DDM방식에서는 평가가 불가능하였으나, 이 의미는 ROE와 배당성향이 높아 기대되는 배당성장률 g가 CAPM의 요구수익률 k보다 높다는 뜻입니다. 즉, 초과이익을 발생시켜 시장을 outperform할 확률이 높다고 생각하였습니다.

즉, **배당성향이 매우 높고 변동성이 낮아 안정적인 우량주인 CME보다**는 최근 성과가 훌륭하여 **성장성이 있으며, 안정적으로 요구수익률 대비 초과이익**을 달성할 것으로 예상되고, **적정 수준의 변동성을 지닌 NASDAQ 거래소이 투자대상으로 가장 적합**하다고 판단됩니다.

한편, ICE는 DDM에서는 내재가치가 좋게 평가되었으나 RIM에서는 낮게 평가되어 결국 기대수익률이 (-)되는 분석결과를 고려하여 선택하지 않았습니다.

투자분석 과제2

Problems

You are a portfolio manager and want to create a portfolio. Choose any three stocks which are traded in the same currency. market. Collect monthly returns of stock prices for 5 years at least. You need to have a risk-free rate for this assignment. Assume that a short-sale is not allowed. You may use any computer tool but do not hand in the data or the program code.

a) What are the names of the stocks you have chosen? Mention the database to provide data.

b) Find annualized arithmetic average and variance of each stock's excess return and variance-covariance matrix of 3 stocks' excess returns.

c) Assume the expected rate of return and future standard deviation (SD) are same as historical arithmetic average and SD for each stock. Plot investment opportunity set for the three stocks. Explain briefly how you do it.

d) Find the Tangent portfolio and Capital Allocation Line.

e) By some reason, you should hold more than 30% of one stock among three stocks. (Select one stock and assign weight more of 30% to the stock. Then the weights for other stocks do not exceed 70%). How does the efficient frontier change? What is the optimal portfolio that maximizes the Sharpe ratio? What are the expected return, the standard deviation, and the Sharpe ratio of the optimal portfolio? Explain why you would have this portfolio.

f) You are worried about a worst scenario. What is the (monthly) 5% VaR of the optimal portfolio in e) under the normality assumption? What is the 5% expected shortfall without the normality and explain how you compute it.

g) Provide suggestions on the proportion of risk-free assets to invest in when you choose the optimal risky portfolio in e). You may consider three values for the coefficient of risk aversion: 2, 3.5 and 5.

Answer

(*a*)

저는 ***CME group*(*CME*)**, ***ICE*(*ICE*)**, ***Nasdaq*(*NDAQ*)**K 세가지 종목을 선정하였습니다.

선정 배경으로는,

(1) 제가 거래소 산업에 관심이 많고,
(2) 세 주식 모두 미국에 상장되어있는 대표적인 글로벌 거래소이며,
(3) 동일한 거래소 산업이고 S&P500지수의 구성종목이라 동일 선상에서 비교하기 적합할 것으로 보이기 때문입니다.

세 주식의 일별수정주가(Adj. close), 벤치마크지수인 S&P500지수의 일별수정가격, 무위험이자율로 채택한 미연준의 일별 Effective-FFR(Federel Funds Rate)를 활용하였습니다.

기간 : 직전 10년(2014.4 ~ 2024.3)
출처 : [Yahoo Finance](https://finance.yahoo.com/)
산출방법 :
(월수익률) 지난달 말 대비 월말 수익률
(월초과수익률) 월수익률 - 월말FFR/12
(월분산) 월/월초과수익률의 표본표준편차
(평균수익률) 월/월초과수익률을 산술평균하여 연환산 (x12)
(평균분산) 월분산을 산술평균하여 연환산 (x12)

```
library(tidyverse)
# stocks and market index S&P500 from yahoo finance
cme <- read_csv("investment_hw/cme.csv") %>% tibble()
ndaq <- read_csv("investment_hw/ndaq.csv") %>% tibble()
ice <- read_csv("investment_hw/ice.csv") %>% tibble()
spx <- read_csv("investment_hw/spx.csv") %>% tibble()
# risk-free rate is effective-FFR(federal funds rate)
ffr <- read_csv("investment_hw/fedfunds.csv") %>% tibble()
# set period 10years
strt_dd='20140101'; end_dd='20240331'
```

(*b*)

```
# tidy data
raw_data <- tibble()
raw_data <- cme %>% mutate(cme=`Adj Close`) %>% select(Date,cme) %>%
  left_join(ndaq %>% mutate(ndaq=`Adj Close`) %>% select(Date,ndaq)) %>%
  left_join(ice %>% mutate(ice=`Adj Close`) %>% select(Date,ice)) %>%
  left_join(spx %>% mutate(spx=`Adj Close`) %>% select(Date,spx)) %>%
  mutate(day=gsub("-","",Date)) %>%
  mutate(year=substr(day,1,4)) %>%
  mutate(month=substr(day,1,6)) %>%
  filter(day>=strt_dd,day<=end_dd) %>%
  left_join(ffr %>%
              mutate(ffr=FEDFUNDS/100,
                     month=substr(gsub("-","",as.character(DATE)),1,6)) %>%
              select(month,ffr)) %>%
  select(year,month,day,cme,ndaq,ice,spx,ffr,Date)
# using monthly return
monthly_raw <- tibble()
monthly_raw <- raw_data %>%
  group_by(year,month) %>%
  arrange(day %>% desc()) %>%
  slice(1) %>%
  pivot_longer(.,c("cme","ndaq","ice","spx"),
               names_to = "name",values_to = "price") %>%
  ungroup() %>%
  arrange(name,year,month) %>%
  mutate(return=price/lag(price)-1) %>%  # monthly return
  mutate(excess_return=return-ffr*(1/12)) %>%
  filter(as.integer(month)>=201404)

# calculate arithmatic mean&var and var-cov matrix of excess return of stocks
monthly_stat <- tibble()
monthly_stat <- monthly_raw %>%
  ungroup() %>%
  group_by(name) %>%
  summarise(avg_excess=mean(excess_return)*12,
```

```
            var_excess=sd(excess_return)^2*12,
            avg_return=mean(return)*12,
            var_return=sd(return)^2*12,
            avg_rf=mean(ffr)) # Annualized
# Cov matrix
variance_matrix <- monthly_raw %>%
  ungroup() %>%
  select(month,name,excess_return) %>%
  pivot_wider(names_from = "name", values_from = "excess_return") %>%
  select(cme,ice,ndaq,spx) %>%
  var()
# Corr matrix
cor_matrix <- monthly_raw %>%
  ungroup() %>%
  select(month,name,excess_return) %>%
  pivot_wider(names_from = "name", values_from = "excess_return") %>%
  select(cme,ice,ndaq,spx) %>%
  cor()
```

위 방법을 기준으로 산출한 각 주식의 평균초과수익률(연환산) 및 공분산행렬은 아래와 같습니다.

Excess return : CME 15.0%, ICE 14.4%, Nasdaq 18.8%

```
monthly_stat
```

```
# A tibble: 4 x 6
  name  avg_excess var_excess avg_return var_return avg_rf
  <chr>      <dbl>      <dbl>      <dbl>      <dbl>  <dbl>
1 cme        0.150     0.0340      0.165     0.0340 0.0141
2 ice        0.144     0.0414      0.158     0.0415 0.0141
3 ndaq       0.188     0.0415      0.202     0.0413 0.0141
4 spx        0.101     0.0229      0.115     0.0229 0.0141
```

Variance-Covariance Matrix of three stocks&Benchmark index

```
variance_matrix
```

```
              cme         ice        ndaq          spx
cme  0.0028310476 0.001767299 0.001138796 0.0008194417
ice  0.0017672988 0.003452184 0.002149990 0.0016652315
ndaq 0.0011387956 0.002149990 0.003456322 0.0016308630
spx  0.0008194417 0.001665231 0.001630863 0.0019071007
```

Correlation Matrix of three stocks&Benchmark index

```
cor_matrix
```

```
           cme       ice      ndaq       spx
cme  1.0000000 0.5653136 0.3640534 0.3526612
ice  0.5653136 1.0000000 0.6224185 0.6489942
ndaq 0.3640534 0.6224185 1.0000000 0.6352191
spx  0.3526612 0.6489942 0.6352191 1.0000000
```

(*c*)

각 주식의 기대수익률과 미래변동성이 과거실현값과 동일하다고 가정하겠습니다.

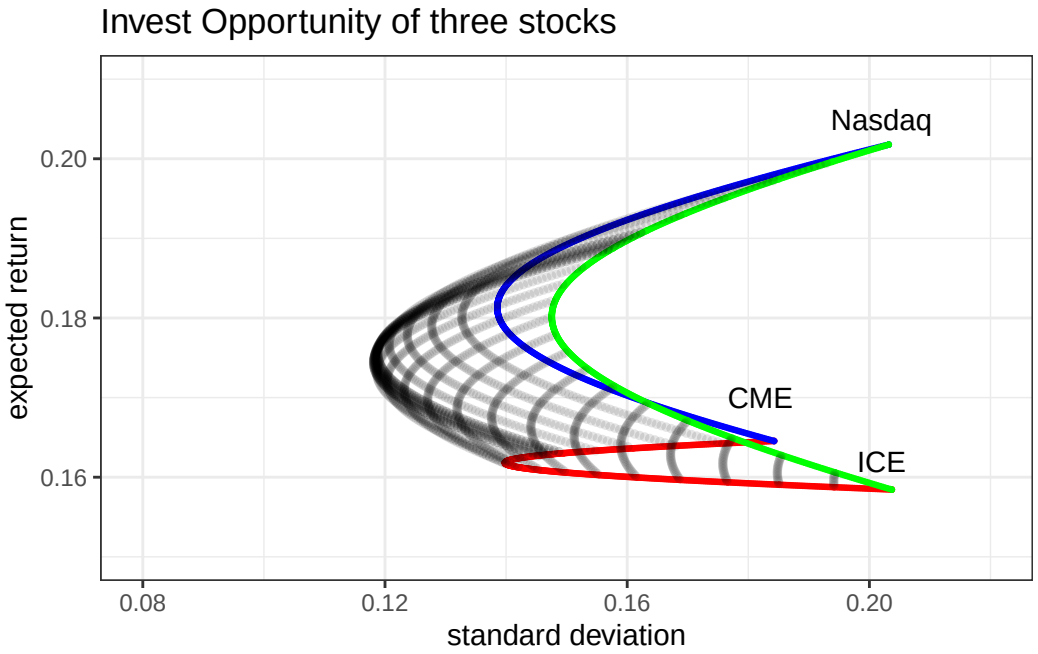
공매도는 없다고 가정하였으므로 각 주식의 비중이 양수가 되도록 설정하고, 먼저 두개의 주식으로 구성된 포트폴리오의 efficient frontier를 각각 구하도록 하겠습니다.

그 다음, 하나의 주식과 다른 두 주식의 포트폴리오를 재결합하면 새로운 포트폴리오를 구성할 수 있으며, 이를 반복하여 efficient frontier를 도식화할 수 있습니다.

아래 코드에서는 ICE의 구성비중을 5%, 10%, ⋯ , 95%로 고정하고 CME-Nasdaq 포트폴리오와 재결합을 반복하여 efficient frontier를 구현하였습니다.

```
# Portfolio return/vol of variance combinations three stocks
portfolio <- tibble()
portfolio <- tibble(cme=seq(0,1,0.002),ice=seq(1,0,-0.002)) %>%
  union_all(.,tibble(cme=seq(0,1,0.002),ice=0)) %>%
  union_all(.,tibble(cme=0,ice=seq(1,0,-0.002)))

for(i in seq(0.05,0.95,0.05)){portfolio <- portfolio %>%
  union_all(.,tibble(cme=seq(0,1-i,0.005),ice=i))}
for(i in seq(0.25,0.30,0.01)){portfolio <- portfolio %>%
  union_all(.,tibble(cme=seq(0,1-i,0.005),ice=i))}
```

```
portfolio <- portfolio %>%
  mutate(ndaq=1-cme-ice) %>%
  mutate(return=cme*monthly_stat$avg_return[1]
         +ice*monthly_stat$avg_return[2]
         +ndaq*monthly_stat$avg_return[3],
         vol=sqrt(cme^2*monthly_stat$var_return[1]
                  +ice^2*monthly_stat$var_return[2]
                  +ndaq^2*monthly_stat$var_return[3]
                  +2*cme*ice*variance_matrix[2]
                  +2*cme*ndaq*variance_matrix[3]
                  +2*ndaq*ice*variance_matrix[7])) %>%
  mutate(sharpe=(return-monthly_stat$avg_rf[1])/vol)

plot_portfolio <- ggplot(data=portfolio,mapping=aes(x=vol,y=return))+
  geom_point(data=portfolio %>% filter(ndaq==0),size=0.5,color="red")+
  geom_point(data=portfolio %>% filter(ice==0),size=0.5,color="blue")+
  geom_point(data=portfolio %>% filter(cme==0),size=0.5,color="green")+
  scale_x_continuous(limits=c(0.08,0.22))+
  scale_y_continuous(limits=c(0.15,0.21))+
  labs(title = "Invest Opportunity of three stocks",
       x="standard deviation",y="expected return") +
  annotate(geom="text",x=0.182,y=0.17,label="CME")+
  annotate(geom="text",x=0.202,y=0.205,label="Nasdaq")+
  annotate(geom="text",x=0.202,y=0.162,label="ICE")+
  theme_bw()

plot_combination <- plot_portfolio +
  geom_point(data=portfolio %>% filter(cme!=0&ice!=0&ndaq!=0),size=1,
  color="black",alpha=0.1)

plot_combination
```



(*d*)

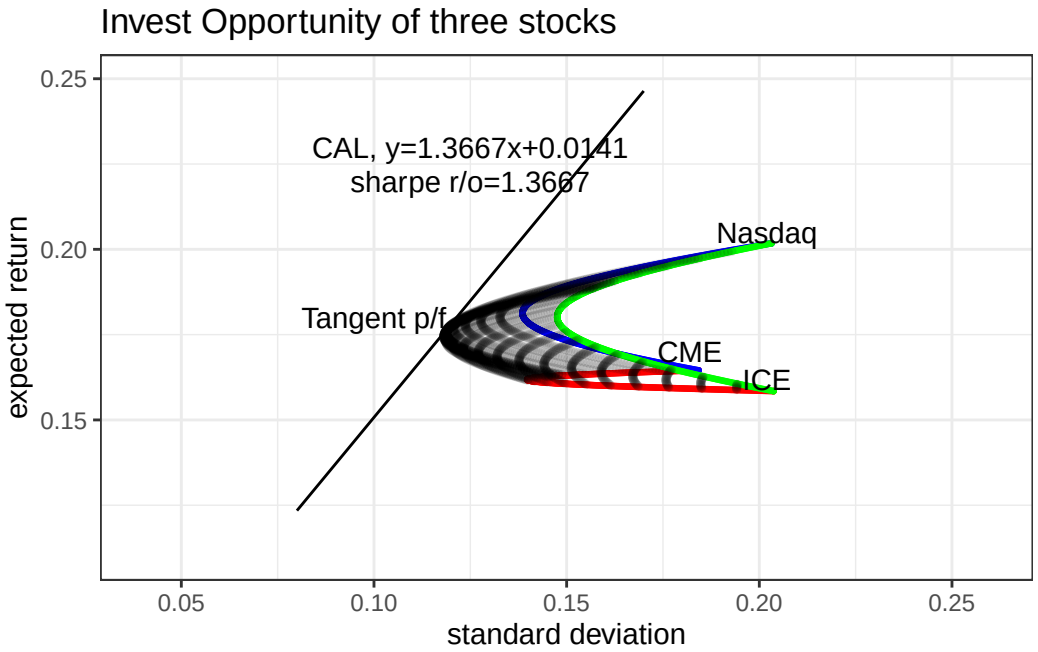
Tangent p/f는 Sharpe r/o를 최대화시키는 세 주식의 조합으로, 위에서 도식화한 투자기회에 대하여 무위험이자율을 y 절편으로 가지는 접선을 그려서 시각화할 수 있습니다.

접선의 기울기는 포트폴리오의 초과수익률을 변동성으로 나눈 값으로 Sharpe r/o가 되므로, 이 접선이 CAL에 해당합니다.

```
sharpe=max(portfolio$sharpe)
tangent <- tibble(vol=seq(0.08,0.17,0.01)) %>%
  mutate(return=monthly_stat$avg_rf[1]+sharpe*vol)

plot_tangent <- plot_combination+
  geom_line(data=tangent)+
  scale_x_continuous(limits=c(0.04,0.26))+
  scale_y_continuous(limits=c(0.11,0.25))+
  annotate(geom="text",x=0.125,y=0.23,label="CAL, y=1.3667x+0.0141")+
  annotate(geom="text",x=0.125,y=0.22,label="sharpe r/o=1.3667")+
  annotate(geom="text",x=0.1,y=0.18,label="Tangent p/f")
```

```
Scale for x is already present.
Adding another scale for x, which will replace the existing scale.
Scale for y is already present.
Adding another scale for y, which will replace the existing scale.
```

```
plot_tangent
```



이 때, Tangent p/f의 기대수익률/변동성/구성비율은 아래와 같습니다.

```
portfolio %>% arrange(sharpe %>% desc()) %>% slice(1)
```

```
# A tibble: 1 x 6
    cme   ice  ndaq return   vol sharpe
  <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>
1  0.36  0.27  0.37  0.177 0.119   1.37
```

(*e*)

Tangent p/f에서 CME, Nasdaq의 비중은 각각 36%, 37%로 이미 30%를 초과하였습니다. 따라서 30%를 초과보유할 주식을 선정할 때 두 주식을 선정한다면 최적포트폴리오는 (d)의 Tangent p/f와 동일할 것 입니다.

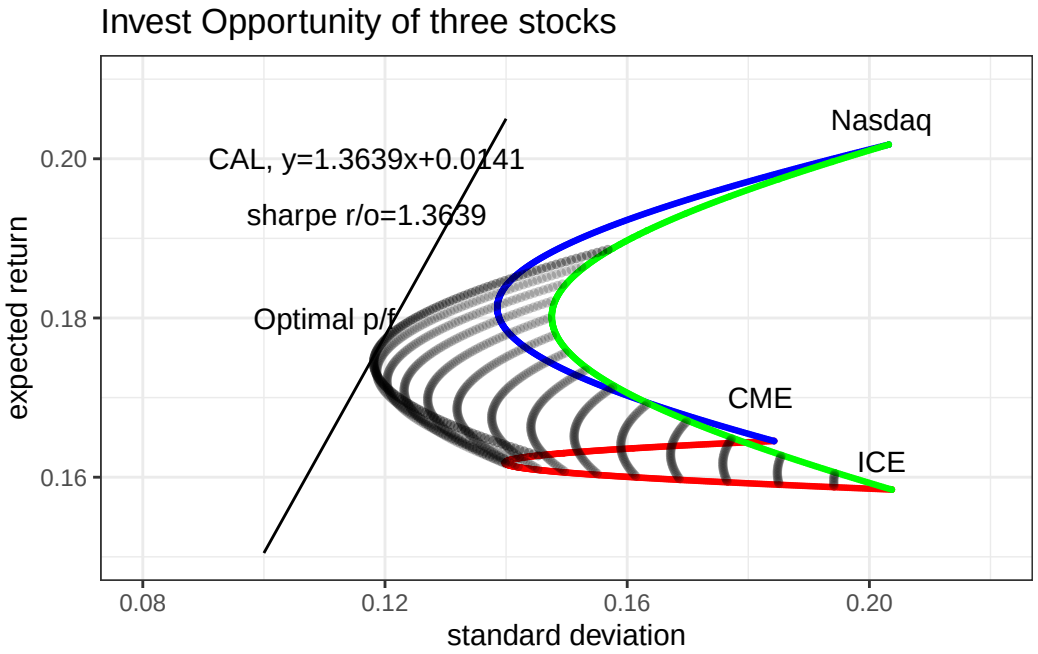
한편, ICE를 선정한다면 (d)의 Tangent p/f 구성이 불가능합니다. 이 경우 새로운 efficient frontier는 아래와 같이 도식화할 수 있으며, 새로운 최적포트폴리오는 ICE의 비중이 정확히 30%일 때 결정됩니다.

```
# Set ice>=0.3
portfolio2 <- portfolio %>% filter(ice>=0.3)
sharpe2=max(portfolio2$sharpe)
tangent2 <- tibble(vol=seq(0.1,0.14,0.01)) %>%
  mutate(return=monthly_stat$avg_rf[1]+sharpe2*vol)
```

```
portfolio2 %>% arrange(desc(sharpe))
```

```
# A tibble: 1,905 x 6
     cme   ice  ndaq return   vol sharpe
   <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>
 1 0.34    0.3 0.36   0.176 0.119   1.36
 2 0.34    0.3 0.36   0.176 0.119   1.36
 3 0.345   0.3 0.355  0.176 0.119   1.36
 4 0.345   0.3 0.355  0.176 0.119   1.36
 5 0.335   0.3 0.365  0.176 0.119   1.36
 6 0.335   0.3 0.365  0.176 0.119   1.36
 7 0.35    0.3 0.35   0.176 0.119   1.36
 8 0.35    0.3 0.35   0.176 0.119   1.36
 9 0.33    0.3 0.37   0.176 0.119   1.36
10 0.33    0.3 0.37   0.176 0.119   1.36
# i 1,895 more rows
```

```
plot_tangent2 <- plot_portfolio +
  geom_point(data=portfolio2 %>% filter(cme!=0&ice!=0&ndaq!=0),
             size=1,color="black",alpha=0.2)+
  geom_line(data=tangent2)+
  annotate(geom="text",x=0.117,y=0.2,label="CAL, y=1.3639x+0.0141")+
  annotate(geom="text",x=0.117,y=0.193,label="sharpe r/o=1.3639")+
  annotate(geom="text",x=0.11,y=0.18,label="Optimal p/f")
plot_tangent2
```



Optimal p/f의 기대수익률/변동성/구성비율은 아래와 같습니다.

```
optimal <- portfolio2 %>% arrange(sharpe %>% desc()) %>% slice(1)
optimal
```

```
# A tibble: 1 x 6
    cme   ice  ndaq return   vol sharpe
  <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>
1  0.34   0.3  0.36  0.176 0.119   1.36
```

(*f*)

먼저, 포트폴리오의 연환산수익률이 정규분포를 따른다면 표준정규분포표를 참조하여 **Value at Risk를 다음과 같이 산출**할 수 있습니다.

$$5\%\ VaR = E(r_p) - 1.65\sigma_p = 0.176 - 1.65 \times 0.119 = -1.99\%$$

```
VaR=optimal$return-1.65*optimal$vol; VaR
```

```
[1] -0.01990571
```

만약 수익률의 분포가 정규분포가 아니라면, Historical VaR 및 ES(Expected Shortfall)을 산출할 수 있습니다. 이를 데이터의 참조기간인 **과거 10년간 최적포트폴리오의 월수익률이 필요**합니다.

위의 최적포트폴리오는 **CME 34%, ICE 30%, Nasdaq 36%로 구성된 포트폴리오**이므로, 거래비용 등을 무시하고 매월말 포트폴리오의 구성비율을 조정한다고 가정하고 포트폴리오의 명목금액을 P_t, 각 주식의 $(t, t+1)$ 월수익률을 $r_{t,k}$ 라고 한다면 $(t, t+1)$간 포트폴리오의 월수익률 r_t는 다음과 같습니다.

$$P_{t+1} = 0.34 \times P_t \times (1 + r_{t,cme}) + 0.3 \times P_t \times (1 + r_{t,ice}) + 0.36 \times P_t \times (1 + r_{t,ndaq})$$

$$\Rightarrow 1 + r_t = \frac{P_{t+1}}{P_t} = 0.34 \times (1 + r_{t,cme}) + 0.3 \times (1 + r_{t,ice}) + 0.36 \times (1 + r_{t,ndaq})$$

이를 적용하면 과거 10년간 최적포트폴리오의 월수익률 120개를 얻을 수 있습니다.

이를 오름차순으로 정렬하면 **Historical 5% VaR은 6번째 관측값이며, 5% ES는 6개의 관측값의 평균**으로 산출할 수 있습니다. 이는 **월수익률 기반 VaR 및 ES이므로 $\sqrt{12}$를 곱하여 연환산**하도록 하겠습니다.

```
optimal_monthly <- tibble()
optimal_monthly <- monthly_raw %>%
  select(month,name,return) %>%
  pivot_wider(names_from = "name", values_from = "return") %>%
  mutate(pf_return=optimal$cme*(1+cme)+optimal$ice*(1+ice)+optimal$ndaq*(1+ndaq)-1) %>%
  select(month,cme,ice,ndaq,pf_return) %>%
  arrange(pf_return) %>%
  slice(1:6)
optimal_monthly
```

```
# A tibble: 6 x 5
  month       cme     ice    ndaq pf_return
  <chr>     <dbl>   <dbl>   <dbl>     <dbl>
1 202204 -0.0779  -0.123  -0.117    -0.106
2 202002 -0.0842  -0.105  -0.119    -0.103
3 202003 -0.127   -0.0917 -0.0697   -0.0957
4 202209 -0.0899  -0.101  -0.0448   -0.0769
5 202201  0.00455 -0.0739 -0.147    -0.0734
6 202205 -0.0935  -0.116  -0.0134   -0.0714
```

Annualized 5% VaR은 -24.78%, ES는 -30.38%입니다.

정규분포 가정 하에 산출한 VaR과 큰 차이가 나는 이유는 지난 22년 Covid-19 팬데믹으로 인해 이례적으로 주가가 월 7% 이상 하락하는 급락장이 지속(Left fat tail)되었는데, 이때의 outlier 표본이 Historical VaR 산출을 지배한 반면 정규분포 근사시에는 반영되지 않아 차이가 발생하는 것으로 추정됩니다.

```
hist_VaR <- optimal_monthly$pf_return[6]*sqrt(12)
hist_ES <- mean(optimal_monthly$pf_return[1:6])*sqrt(12)
paste(round(hist_VaR,6), round(hist_ES,6) , round(VaR,6), sep=" / ")
```

```
[1] "-0.247279 / -0.303873 / -0.019906"
```

(g)

위의 최적포트폴리오와 무위험자산을 이용하여 투자를 결정한다면, 최종적인 포트폴리오의 기대수익률 및 변동성은 자본배분선($E(r_p) = 1.3639\sigma_p + 0.0141$) 위에서 결정될 것입니다.

이때, 무위험자산의 비중은 투자자의 위험회피정도에 따른 효용함수 $U = E(r_p) - \frac{1}{2}A\sigma_p^2$를 최대화시키는 수준에서 결정됩니다.

이에 따라 산출한 **무위험자산의 비중은 모든 경우(A=2, 3.5, 5)에서 0**이 됩니다. 이는 과거기간 중 제로금리 기간이 다소 길어 무위험자산의 수익률이 낮은 반면, 선정한 주식의 기대수익률은 상대적으로 높아 일어난 것으로 추정됩니다.

부가적으로, 무위험자산의 비중 산식 $r^* = 1 - \frac{E(r_p)-r_f}{A\sigma_p^2}$를 역산하여 **위험회피정도 A를 역산해보면, 약 11.48까지는 최적포트폴리오를 100% 보유하는 것이 효용을 극대화**시키는 투자결정입니다.

```
# Utility function to verify risk-free asset ratio
utility <- tibble(vol=seq(optimal$vol,0,-0.0001)) %>%
  mutate(return=sharpe2*vol+monthly_stat$avg_rf[1]) %>%
  mutate(rf_ratio=(optimal$return-return)/(optimal$return-monthly_stat$avg_rf[1])) %>%
  mutate(a2=return-0.5*2*vol^2,
         a3.5=return-0.5*3.5*vol^2,
         a5=return-0.5*5*vol^2)
utility %>% arrange(desc(a5)) %>% slice(1)
```

```
# A tibble: 1 x 6
    vol return rf_ratio    a2  a3.5    a5
  <dbl>  <dbl>    <dbl> <dbl> <dbl> <dbl>
1 0.119  0.176        0 0.162 0.151 0.141
```

```
(optimal$return-monthly_stat$avg_rf[1])/optimal$vol^2
```

```
[1] 11.48027
```

투자분석 과제3

Question

Select any four stocks from the pool of market index. Make sure all four stocks belong to four different industries. You will estimate the single index model for each of the chosen stocks.

a) Collect the 60 recent monthly returns of the chosen stocks, and the monthly T-bill rates for the same period. Also, you need market index returns for the same period. Run regression model with this data. Report the alpha and beta estimates.

b) Interpret each estimate of alpha and beta. Consider the smallest and largest betas among the four stocks. To which industries do the two companies belong? Is the business consistent with the estimated beta for the two companies?

c) Use the first 30 months only and run the regression. Report the alpha and beta estimates.

d) Use the last 30 months only and run the regression. Report the alpha and beta estimates.

e) Are the three sets of estimates identical? Discuss the result of a), c) and d).

Answer

먼저, 제가 선정한 주식은 금융산업의 뱅크오브아메리카(BAC), 거래소산업의 시카고거래소그룹(CME), IT업종의 IBM(IBM), 외식업종의 맥도날드(MCD)입니다.

네 종류의 주식은 모두 S&P500 지수의 구성종목이며, 출처는 야후파이낸스입니다.

무위험이자율은 4 weeks T-bill rate이며, 출처는 미 재무부 홈페이지입니다.

2019~2023년 5년간 데이터를 월별로 취합하여 지수 및 주식의 월별수익률을 산출한 다음, 월환산 무위험이자율을 차감하여 초과수익률을 산출하였고, 각 주식의 초과수익률과 지수의 초과수익률에 대해서 단순선형회귀분석을 진행하였습니다.

```
rm(list=ls())
library(tidyverse)

# stocks and market index S&P500 from yahoo finance
bac <- read_csv("investment_hw/bac.csv") %>% tibble()
cme <- read_csv("investment_hw/cme.csv") %>% tibble()
ibm <- read_csv("investment_hw/ibm.csv") %>% tibble()
mcd <- read_csv("investment_hw/mcd.csv") %>% tibble()
spx <- read_csv("investment_hw/spx.csv") %>% tibble()
# risk-free rate is effective-FFR(federal funds rate)
tbill <- read_csv("investment_hw/tbill.csv") %>% tibble()

# set period 5years
strt_dd='20181201'
end_dd='20231231'

# tidy date
rfr <- tbill %>% mutate(tbill=`4 WEEKS BANK DISCOUNT`) %>%
  mutate(y=paste0("20",substr(Date,nchar(Date)-1,nchar(Date)))) %>%
  mutate(m=if_else(substr(Date,2,2)=="/",
                   paste0("0",substr(Date,1,1)),substr(Date,1,2))) %>%
  mutate(d=if_else(substr(Date,2,2)=="/",
                   if_else(substr(Date,4,4)=="/",
                           paste0("0",substr(Date,3,3)),substr(Date,3,4)),
                   if_else(substr(Date,5,5)=="/",
                           paste0("0",substr(Date,4,4)),substr(Date,4,5)))) %>%
  mutate(day=paste0(y,m,d)) %>%
  select(day,tbill)

raw_data <- tibble()
raw_data <- bac %>% mutate(bac=`Adj Close`) %>% select(Date,bac) %>%
  left_join(cme %>% mutate(cme=`Adj Close`) %>% select(Date,cme)) %>%
  left_join(ibm %>% mutate(ibm=`Adj Close`) %>% select(Date,ibm)) %>%
  left_join(mcd %>% mutate(mcd=`Adj Close`) %>% select(Date,mcd)) %>%
  left_join(spx %>% mutate(spx=`Adj Close`) %>% select(Date,spx)) %>%
```

```
  mutate(day=gsub("-","",Date)) %>%
  mutate(year=substr(day,1,4)) %>%
  mutate(month=substr(day,1,6)) %>%
  left_join(rfr,by="day") %>%
  filter(day>=strt_dd,day<=end_dd) %>%
  select(year,month,day,bac,cme,ibm,mcd,spx,tbill,Date)

# using monthly return
monthly_raw <- tibble()
monthly_raw <- raw_data %>%
  group_by(year,month) %>%
  arrange(day %>% desc()) %>%
  slice(1) %>%
  pivot_longer(.,c("bac","cme","ibm","mcd","spx"),
               names_to = "name",values_to = "price") %>%
  ungroup() %>%
  arrange(name,year,month) %>%
  mutate(return=price/lag(price)-1) %>%  # monthly return
  mutate(ex_return=return-tbill/100/12) %>% # excess return
  filter(as.integer(month)>=201901)
```

(a)

각 회귀분석의 결과로 산출된 절편(알파, 연환산)과 계수(베타)는 아래와 같습니다.

구 분	alpla	beta
Band of America	-0.0571	1.42
CME	0.0054	0.45
IBM	0.0439	0.76
McDonald's	0.0352	0.71

```
regression <- tibble()
regression <- monthly_raw %>%
  select(day,name,ex_return) %>%
  pivot_wider(.,values_from = "ex_return", names_from = "name")
```

```r
reg_bac <- lm(regression$bac~regression$spx) %>% coef() %>% as_tibble()
reg_cme <- lm(regression$cme~regression$spx) %>% coef() %>% as_tibble()
reg_ibm <- lm(regression$ibm~regression$spx) %>% coef() %>% as_tibble()
reg_mcd <- lm(regression$mcd~regression$spx) %>% coef() %>% as_tibble()

result_reg <- tibble(name=c("bac","cme","ibm","mcd"),
                     alpha=c(reg_bac$value[1],reg_cme$value[1],reg_ibm$value[1],reg_mcd$value[
                     beta=c(reg_bac$value[2],reg_cme$value[2],reg_ibm$value[2],reg_mcd$value[2
result_reg
```

```
# A tibble: 4 x 3
  name     alpha  beta
  <chr>    <dbl> <dbl>
1 bac   -0.0571  1.42
2 cme    0.00536 0.449
3 ibm    0.0439  0.758
4 mcd    0.0352  0.708
```

(b)

베타가 가장 높은 주식은 뱅크오브아메리카이며, 가장 낮은 주식은 시카고거래소그룹입니다.

일반적으로 금융업종이 기간산업인 거래소업종에 비해 변동성이 큰 편이며, 실제로 기간 중 BOA의 연환산 변동성이 33.4%로 CME의 20.9%를 크게 상회하였습니다.

물론 기간 중 변동성이 가장 낮았던 주식은 맥도날드(19.1%)로, 베타는 0.71이 산출되어 변동성과 베타의 관계가 선형적이지는 않지만, 거래소산업의 베타가 일반적으로 0.5수준임을 고려할 때 **합리적인 결과가 산출**된 것으로 보입니다.

```r
monthly_raw %>%
  group_by(name) %>%
  summarise(vol=sd(ex_return)*sqrt(12))
```

```
# A tibble: 5 x 2
  name    vol
  <chr> <dbl>
1 bac   0.334
```

```
2 cme   0.209
3 ibm   0.240
4 mcd   0.191
5 spx   0.185
```

(c)~(d)

먼저, (a), (c), (d)의 결과를 각각 정리하면 아래와 같습니다.

연환산 *Alpha* 및 *Beta*

구 분	(a)α	(c)α	(d)α	(a)β	(c)β	(d)β
Band of America	-0.0571	-0.0814	-0.0699	1.42	1.61	1.23
CME	0.0054	-0.0314	0.0196	0.45	0.56	0.35
IBM	0.0439	-0.0804	0.0940	0.76	1.14	0.41
McDonald's	0.0352	-0.0241	0.0895	0.71	0.73	0.71

알파의 경우 앞 30개월의 결과가 낮은 경향이 있고, 베타의 경우 앞 30개월의 결과가 높은 경향이 있습니다.

앞 30개월의 기간은 2019.1~2021.6이며, 뒤 30개월은 2021.7~2023.12입니다.

기간의 특징적인 부분은 **앞 30개월의 기간에 코로나19 팬데믹으로 인해 주가변동성이 굉장히 높고, 주식이 급락했던 시기를 포함**하고 있다는 것 입니다.

따라서 앞 30개월 기간동안은 평균적으로 주식의 변동성이 높았고, 주가급락으로 인해 평균적인 수익률은 낮아 시장지수인 S&P500을 Underperform하였을 가능성이 높습니다.

이로 인해 **시장지수 대비 초과수익률인 알파는 (-)를 기록**하였으며, **베타는 전체기간(a)과 후반기간(d) 대비 높은 수치가 산출**된 것으로 해석할 수 있습니다.

오히려, 최근 30개월간의 주식시장을 평균적인 흐름이라고 본다면, 뒤 30개월의 결과값이 해당 주식들의 일반적인 알파 및 베타라고 볼 수 있습니다. 따라서, **코로나19 기간동안의 왜곡된 값으로 인해 전체 기간의 알파가 과소평가, 베타가 과대평가되었을 가능성**이 있습니다.

```
regression2 <- tibble()
regression2 <- monthly_raw %>%
  select(day,name,ex_return) %>%
  pivot_wider(.,values_from = "ex_return", names_from = "name") %>%
```

```
  slice(1:30)

reg_bac2 <- lm(regression2$bac~regression2$spx) %>% coef() %>% as_tibble()
reg_cme2 <- lm(regression2$cme~regression2$spx) %>% coef() %>% as_tibble()
reg_ibm2 <- lm(regression2$ibm~regression2$spx) %>% coef() %>% as_tibble()
reg_mcd2 <- lm(regression2$mcd~regression2$spx) %>% coef() %>% as_tibble()

result_reg2 <- tibble(name=c("bac","cme","ibm","mcd"),
                      alpha2=c(reg_bac2$value[1],reg_cme2$value[1],
                               reg_ibm2$value[1],reg_mcd2$value[1])*12,
                      beta2=c(reg_bac2$value[2],reg_cme2$value[2],
                              reg_ibm2$value[2],reg_mcd2$value[2]))

regression3 <- tibble()
regression3 <- monthly_raw %>%
  select(day,name,ex_return) %>%
  pivot_wider(.,values_from = "ex_return", names_from = "name") %>%
  slice(31:60)

reg_bac3 <- lm(regression3$bac~regression3$spx) %>% coef() %>% as_tibble()
reg_cme3 <- lm(regression3$cme~regression3$spx) %>% coef() %>% as_tibble()
reg_ibm3 <- lm(regression3$ibm~regression3$spx) %>% coef() %>% as_tibble()
reg_mcd3 <- lm(regression3$mcd~regression3$spx) %>% coef() %>% as_tibble()

result_reg3 <- tibble(name=c("bac","cme","ibm","mcd"),
                      alpha3=c(reg_bac3$value[1],reg_cme3$value[1],
                               reg_ibm3$value[1],reg_mcd3$value[1])*12,
                      beta3=c(reg_bac3$value[2],reg_cme3$value[2],
                              reg_ibm3$value[2],reg_mcd3$value[2]))

result <- tibble()
result <- result_reg %>%
  left_join(result_reg2,by="name") %>%
  left_join(result_reg3,by="name")
```

```
result
```

```
# A tibble: 4 x 7
  name     alpha  beta  alpha2 beta2  alpha3 beta3
  <chr>    <dbl> <dbl>   <dbl> <dbl>   <dbl> <dbl>
1 bac   -0.0571  1.42  -0.0814 1.61  -0.0699 1.23
2 cme    0.00536 0.449 -0.0314 0.564  0.0196 0.345
3 ibm    0.0439  0.758 -0.0804 1.14   0.0940 0.413
4 mcd    0.0352  0.708 -0.0241 0.729  0.0895 0.713
```

투자분석 과제4

Question

You need to construct a portfolio consisting of a passive portfolio and three stocks you already selected, doing the Assignment 2. If you selected Korea (USA) stocks, use KOSPI 200 (S&P 500) for the passive portfolio.

You construct your optimal risky portfolio based on **single-index model** that was covered in class. Don't hand in data or your program code.

(a) What were the names of the individual stocks you have chosen in assignment 2? Indicate the source database.

(b) With the 60 recent monthly returns of the chosen stocks, and the monthly T-bill rates for the same period, **run regressions** and report the estimated **beta**, **alpha**, **t-value** or **p-value**, **R squared** and **residual SD** of each stock and documents them.

(c) Assume that the average of the risk premium of the index, forecasted by investment companies, is **6%** and its standard deviation is **15%**. Or you may use the past average excess return for the index.

If you make the optimal risky portfolio according to SIM, you **need alpha** of the expected return for each stock. **There are three options** such as

(i) you may randomly choose these numbers. If this is the case, be reasonable to set the value for alpha, or

(ii) you may use the estimated alpha in (b). for future alpha (As you know, this won't hold generally), or

(iii) you may accept the 12-month future price suggested by analysts and calculate alphas using SCL.

What is your choice? **Construct and report your optimal risky portfolio** consisting of the index and the three stocks **maximizing Sharpe ratio**. Assume short-sale is allowed.

(d) What is your **Information ratio** for the active portfolio you have chosen?

(e) **Compare Sharpe ratio** of the optimal risky portfolio with Sharpe ratio of the index portfolio.

Answer

(a)

저는 HW2에서 ***CME group*(*CME*)**, ***ICE*(*ICE*)**, ***Nasdaq*(*NDAQ*)** 세가지 종목을 선정하였습니다.

선정 배경으로는,

(1) 제가 거래소 산업에 관심이 많고,
(2) 세 주식 모두 미국에 상장되어있는 대표적인 글로벌 거래소이며,
(3) 동일한 거래소 산업이고 S&P500지수의 구성종목이라 동일 선상에서 비교하기 적합할 것으로 보이기 때문입니다.

세 주식의 일별수정주가(Adj. close), 벤치마크지수인 S&P500지수의 일별수정가격을 활용하였고, 무위험이자율은 4 weeks T-bill rate를 사용하였습니다.

기간 : 직전 5년(2019.1 ~ 2023.12)
출처 : Yahoo Finance 및 미 재무부
산출방법 :
(월수익률) 지난달 말 대비 월말 수익률
(월초과수익률) 월수익률 - 월말T-bill/12
(월분산) 월/월초과수익률의 표본표준편차
(평균수익률) 월/월초과수익률을 산술평균하여 연환산 (x12)
(평균분산) 월분산을 산술평균하여 연환산 (x12)

Data import and pre-processing

```
rm(list=ls())


setwd("/Users/hwan/Desktop/Homepage/study_24spring")
getwd()
library(tidyverse)

# stocks and market index S&P500 from yahoo finance
cme <- read_csv("investment_hw/cme.csv") %>% tibble()
ndaq <- read_csv("investment_hw/ndaq.csv") %>% tibble()
```

```r
ice <- read_csv("investment_hw/ice.csv") %>% tibble()
spx <- read_csv("investment_hw/spx.csv") %>% tibble()
# risk-free rate is T-bill rate
tbill <- read_csv("investment_hw/tbill.csv") %>% tibble()

# set period 10years
strt_dd='20140101'
end_dd='20231231'

# tidy date
rfr <- tbill %>% mutate(tbill=`4 WEEKS BANK DISCOUNT`) %>%
  mutate(y=paste0("20",substr(Date,nchar(Date)-1,nchar(Date)))) %>%
  mutate(m=if_else(substr(Date,2,2)=="/",paste0("0",substr(Date,1,1)),substr(Date,1,2))) %>%
  mutate(d=if_else(substr(Date,2,2)=="/",
                   if_else(substr(Date,4,4)=="/",paste0("0",substr(Date,3,3)),substr(Date,3,4)
                   if_else(substr(Date,5,5)=="/",paste0("0",substr(Date,4,4)),substr(Date,4,5)
  mutate(day=paste0(y,m,d)) %>%
  select(day,tbill)

raw_data <- tibble()
raw_data <- cme %>% mutate(cme=`Adj Close`) %>% select(Date,cme) %>%
  left_join(ndaq %>% mutate(ndaq=`Adj Close`) %>% select(Date,ndaq)) %>%
  left_join(ice %>% mutate(ice=`Adj Close`) %>% select(Date,ice)) %>%
  left_join(spx %>% mutate(spx=`Adj Close`) %>% select(Date,spx)) %>%
  mutate(day=gsub("-","",Date)) %>%
  mutate(year=substr(day,1,4)) %>%
  mutate(month=substr(day,1,6)) %>%
  left_join(rfr,by="day") %>%
  filter(day>=strt_dd,day<=end_dd) %>%
  select(year,month,day,cme,ndaq,ice,spx,tbill,Date)

# using monthly return
monthly_raw <- tibble()
monthly_raw <- raw_data %>%
  group_by(year,month) %>%
```

```
  arrange(day %>% desc()) %>%
  slice(1) %>%
  pivot_longer(.,c("cme","ndaq","ice","spx"),
               names_to = "name",values_to = "price") %>%
  ungroup() %>%
  arrange(name,year,month) %>%
  mutate(return=price/lag(price)-1) %>%  # monthly return
  mutate(ex_return=return-tbill/100/12) %>% # excess return
  filter(as.integer(month)>=201901)
```

(b)

SIM(Single Index Model)을 기반으로 분석하므로, 회귀식은 아래와 같습니다.

$$r_{i,t} - r_{f,t} = \alpha_i + \beta_i(r_{M,t} - r_{i,t}) + \epsilon_{i,t}$$

이에 따라 각 주식의 월초과수익률을 S&P500지수의 월초과수익률로 회귀분석을 실시하겠습니다.

```
regression <- tibble()
regression <- monthly_raw %>%
  select(day,name,ex_return) %>%
  pivot_wider(.,values_from = "ex_return", names_from = "name")

reg_cme <- lm(regression$cme~regression$spx) %>% summary()
reg_ndaq <- lm(regression$ndaq~regression$spx) %>% summary()
reg_ice <- lm(regression$ice~regression$spx) %>% summary()

result_reg <- tibble(name=c("cme","ndaq","ice"),
                     alpha=c(reg_cme$coefficients[1],
                             reg_ndaq$coefficients[1],
                             reg_ice$coefficients[1]),
                     beta=c(reg_cme$coefficients[2],
                            reg_ndaq$coefficients[2],
                            reg_ice$coefficients[2]),
                     sd_residual=c(reg_cme$sigma,
```

```
                                    reg_ndaq$sigma,
                                    reg_ice$sigma))
```

```
reg_cme
```

```
Call:
lm(formula = regression$cme ~ regression$spx)

Residuals:
      Min        1Q    Median        3Q       Max
-0.114664 -0.035431 -0.006045  0.029527  0.124369

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.0004469  0.0073717   0.061  0.95187
regression$spx 0.4493575  0.1364499   3.293  0.00169 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05598 on 58 degrees of freedom
Multiple R-squared:  0.1575,    Adjusted R-squared:  0.143
F-statistic: 10.85 on 1 and 58 DF,  p-value: 0.001691
```

```
reg_ndaq
```

```
Call:
lm(formula = regression$ndaq ~ regression$spx)

Residuals:
      Min        1Q    Median        3Q       Max
-0.161675 -0.021231  0.003611  0.030794  0.094846

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.004646   0.005915   0.785    0.435
```

```
regression$spx 0.947498   0.109480   8.655 4.95e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.04491 on 58 degrees of freedom
Multiple R-squared:  0.5636,    Adjusted R-squared:  0.5561
F-statistic:  74.9 on 1 and 58 DF,  p-value: 4.949e-12
```

```
reg_ice
```

```
Call:
lm(formula = regression$ice ~ regression$spx)

Residuals:
      Min         1Q     Median         3Q        Max
-0.116179 -0.027230 -0.007334  0.028422  0.137285

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.0002405  0.0057437   0.042    0.967
regression$spx 0.9886906  0.1063155   9.300 4.26e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.04362 on 58 degrees of freedom
Multiple R-squared:  0.5986,    Adjusted R-squared:  0.5916
F-statistic: 86.48 on 1 and 58 DF,  p-value: 4.264e-13
```

결과를 표로 정리하면 아래와 같습니다.

구 분	CME	Nasdaq	ICE
α	0.0004	0.0046	0.0002
β	0.4494	0.9475	0.9887
$t-value$	3.293	8.655	9.300
$p-value$	0.0017	0.0000	0.0000
$Adj.R-squared$	0.143	0.5561	0.5916

구 분	CME	Nasdaq	ICE
$residualSD$	0.0560	0.0449	0.0436

(c)

위에서 각 주식의 alpha는 모두 양수로 산출되었습니다. (ii)를 채택하여 (b)의 회귀분석 결과로 도출된 alpha를 사용하겠습니다.

Optimal-risky portfolio 구축을 위해 먼저 passive 및 active p/f를 정의하겠습니다.

Passive p/f는 문제에서 $E(R_M) = E(r_M - r_f) = 0.06,\ \sigma_M = 0.15$로 주어져있습니다.

이는 연환산이므로 월수익률을 사용한 앞선 분석에 적용하기 위해 월단위로 환산하겠습니다.

즉, **월환산 평균 리스크프리미엄은 0.005 및 표준편차는** $0.15/\sqrt{12} = 0.0433$**입니다.**

Active p/f는 (b)에서 도출한 alpha, beta, residual SD를 통해 구축하도록 하겠습니다.

각 주식의 구성비율은 리스크대비 초과수익률($\frac{\alpha_i}{\sigma^2(\epsilon_i)}$)을 가중평균하여 산출할 수 있습니다.

```
active_pf <- result_reg %>% mutate(w0=alpha/sd_residual^2)
active_pf <- active_pf %>% mutate(w1=w0/sum(active_pf$w0))
active_pf
```

```
# A tibble: 3 x 6
  name     alpha  beta sd_residual    w0     w1
  <chr>    <dbl> <dbl>       <dbl> <dbl>  <dbl>
1 cme   0.000447 0.449      0.0560 0.143 0.0554
2 ndaq  0.00465  0.947      0.0449 2.30  0.895
3 ice   0.000241 0.989      0.0436 0.126 0.0492
```

Active p/f의 alpha, beta, residual SD는 아래와 같습니다.

```
active_pf <- active_pf %>%
  summarise(alpha=sum(w1*alpha), beta=sum(w1*beta),
            sd_residual=sum(w1^2*sd_residual^2) %>% sqrt())
active_pf
```

```
# A tibble: 1 x 3
    alpha  beta sd_residual
    <dbl> <dbl>       <dbl>
1 0.00420 0.922      0.0404
```

이제, passive & active p/f를 결합하여 Sharpe ratio가 최대값이 되는 최적 비중 w_A^*를 산출하도록 하겠습니다.

```
Sharpe_SIM <- function(alpha_A,beta_A,SD_residual_A,weight_A=0.5,excess_M,SD_M){
  alpha_pf=weight_A*alpha_A
  beta_pf=weight_A*beta_A+1-weight_A
  excess_pf=alpha_pf+beta_pf*excess_M
  Var_residual_pf=weight_A^2*SD_residual_A^2
  SD_pf=sqrt(beta_pf^2*SD_M^2+Var_residual_pf)
  result <- tibble(alpha=alpha_pf,
                   beta=beta_pf,
                   excess_return=excess_pf,
                   SD=SD_pf,
                   Sharpe=excess_pf/SD_pf)
  return(result)
}

maximize_SIM <- function(weight_A){
  result <- Sharpe_SIM(active_pf$alpha,active_pf$beta,active_pf$sd_residual,
                       weight_A,0.06/12,0.15/sqrt(12))
  return(result$Sharpe)}

optimal_weight=optimize(maximize_SIM,c(0,1),maximum = TRUE)
optimal_weight
```

```
$maximum
[1] 0.8969585

$objective
[1] 0.1553268
```

Active p/f의 최적비중 $w_a^* = 0.8969$이며, 이때 Optimal risky p/f의 Sharpe ratio는 0.1553입니다.

Optimal risky p/f의 alpha, beta, 초과수익률, 표준편차는 아래와 같습니다.

```
SIM_optimal <- Sharpe_SIM(active_pf$alpha,active_pf$beta,active_pf$sd_residual,
                          optimal_weight$maximum,0.06/12,0.15/sqrt(12))
SIM_optimal
```

```
# A tibble: 1 x 5
    alpha  beta excess_return     SD Sharpe
    <dbl> <dbl>         <dbl>  <dbl>  <dbl>
1 0.00376 0.930       0.00841 0.0542  0.155
```

(d)

Information ratio $= \frac{\alpha_A}{\sigma(\epsilon_A)} = 0.1039$

```
info_ratio=active_pf$alpha/active_pf$sd_residual
info_ratio
```

```
[1] 0.1038897
```

(e)

각 포트폴리오의 Sharpe ratio는 아래와 같습니다.

$$Sharpe\ r/o\ (Optimal\ risky)\ =\ 0.1553,\ \ Sharpe\ r/o\ (Index)\ =\ \frac{0.005}{0.0433} = 0.1155$$

Optimal risky p/f가 보다 나은 성과를 보여주며, 정보비율로 그 차이를 표현할 수 있습니다.

```
sqrt(0.1155^2+info_ratio^2)
```

```
[1] 0.1553491
```

Part IV

재무회계('24봄)

재무회계 Ch1-3 과제

(1) Quetion 1-31

31. Accrual versus cash basis of accounting. The following information is based on the financial statements of Hewston, a large manufacturing firm. Annual revenues are $66,387 million and net expenses (including income taxes) are $62,313 million. During the year, the firm collected $65,995 million in cash from customers and had cash outflows associated with payments to suppliers and vendors of $56,411 million.

a. Calculate net income and net cash flow for the year.

b. How can cash collected from customers be less than revenues?

c. How can cash payments to suppliers and vendors be less than expenses?

Figure 7.1: Chapter1-31

Answer

(a) $net\ income = 66387 - 62313 = 4,074 million\$$

$net\ cash\ flow = 65995 - 56411 = 9,584 million\$$

(b) If customer buy a product of Hewston and want to pay later, then Hewston get a certain amount of revenue but get no cash. Instead, Hewston get a asset, namely accrued revenue, that will be paid for cash.

In that case, a company can get revenues more than collected cash.

(c) If Hewston buy material from suppliers and vendors to produce its own products but want to pay after selling product, then there exists a certain amount of expense and occurs a liability, namely accrued expense, that will pay for cash.

In that case, a company's expenses is bigger than its cash payments.

(2) Quetion 1-34

34. Balance sheet relations. Selected balance sheet amounts for SinoTwelve, a Chinese manufacturer, appear next. All amounts are in thousands of U.S. dollars ($). Compute the missing amounts.

	2013	2012
Total Assets	?	$5,450,838
Current Liabilities	$4,488,461	3,527,504
Current Assets	?	3,062,449
Total Liabilities and Shareholders' Equity	7,199,847	?
Noncurrent Liabilities	1,098,123	?
Shareholders' Equity	?	1,134,276
Noncurrent Assets	2,494,481	?
Total Liabilities.	?	?

Figure 7.2: Chapter1-34

Answer

Total Assets = Current Assets + Noncurrent Assets

Total Liabilities = Current Liabilities + Noncurrent Liabilities

Total Liabilities and Shareholders' Equity = Total Liabilities + Shareholders' Equity

And, *Total Assets = Total Liabilities and Shareholders' Equity*

Apply above,

Category	2013	2012
Total Assets	**7,199,847**	5,450,838
Current Liabilities	4,488,461	3,527,504
Current Assets	**4,705,366**	3,062,449
Total Liabilities and Shareholders' Equity	7,199,847	**5,450,838**
Noncurrent Liabilities	1,098,123	789,058
Shareholders' Equity	**1,613,263**	1,134,276
Noncurrent Assets	2,494,481	2,388,389
Total Liabilities	**5,586,594**	**4,316,562**

(3) Quetion 2-10

PROBLEMS

10. Dual effects of transactions on balance sheet equation and journal entries. Assume that during Year 15, Bullseye Corporation, a U.S. retailer, engages in the following six transactions. Bullseye Corporation applies U.S. GAAP and reports its results in millions of U.S. dollars ($). Do not be concerned that after these transactions, the balance in the Cash account is negative. The firm will deal with that issue in transactions not shown here.

(1) The firm issues 20 million shares of $0.0833 par value common stock for a total of $960 million cash.

(2) It purchases merchandise costing $1,500 million on account.

(3) The firm acquires a new store location, consisting of a building costing $3,200 million and land costing $930 million. It pays cash to the owner of the property.

(4) The firm purchases fixtures for the new store costing $860 million, on account.

(5) The firm pays the merchandise supplier in transaction **(2)** the amount due.

(6) The firm pays the supplier of the fixtures in transaction **(4)** half of the amount due in cash. The firm pays the other half by issuing 8.6 million common shares to the supplier. At the time of this transaction, Bullseye Corporation's shares traded at $50 per share in the market.

a. Indicate the effects of these six transactions on the balance sheet equation using this format:

Transaction Number	Assets	=	Liabilities	+	Shareholders' Equity
(1)	+$960		$0		+$960
Subtotal	$960	=	$0	+	$960

b. Give the journal entries for each of the six transactions.

Figure 7.3: Chapter2-10

Answer

(a)

Transaction Number	Assets	=	Liabilities	+	Shareholder's Equity
(1)	+$960m		$0		+$960m
(2)	+$1,500m		+$1,500m		$0

Transaction Number	Assets	=	Liabilities	+	Shareholder's Equity
(3)	+$3,200m		$0		$0
	+$930m				
	-$4,130m				
(4)	+$860m		+$860m		$0
(5)	-$1,500m		-$1,500m		$0
(6)	-$430m		-$430m -$430m		+$430m
	+$430m				
	-$430m				
Total	**$1,390m**	=	**$0**	+	**$1,390m**

(*b*)

Transaction Num. (1)	Debit	Credit
Cash	$960m	
Common stock		$1.7m
Additional capital		$958.3m

Transaction Num. (2)	Debit	Credit
Inventory	$1,500m	
Payable		$1,500m

Transaction Num. (3)	Debit	Credit
Building	$3,200m	
Landscape	$930m	
Cash paid		$4,130m

Transaction Num. (4)	Debit	Credit
Fixture	$860m	
Payable		$860m

Transaction Num. (5)	Debit	Credit
Payable	$1,500m	
Cash		$1,500m

Transaction Num. (6)	Debit	Credit
Payable	$430m	
Cash paid		$430m
Cash	$430m	
Common stock		$0.7m
Additional capital		$429.3m
Payable	$430m	
Cash paid		$430m

(4) Quetion 3-12

12. Relations between financial statements. The following selected information is based on the Year 7 financial statements adapted from those of Beyond Petroleum (BP). BP applies IFRS and reports its results in millions of U.S. dollars. Compute the missing information in each of the following four independent cases. The letters in parentheses refer to the following:

BS—Balance sheet
IS—Income statement
SCF—Statement of cash flows

a.	Accounts Receivable, January 1, Year 7 (BS)	$?
	Sales on Account for Year 7 (IS)	288,951
	Collections from Customers on Account during Year 7 (SCF)	289,623
	Accounts Receivable, December 31, Year 7 (BS)	38,020
b.	Income Taxes Payable, January 1, Year 7 (BS)	$ 2,635
	Income Tax Expense for Year 7 (IS)	10,442
	Payments to Governments during Year 7 (SCF)	?
	Income Taxes Payable, December 31, Year 7 (BS)	3,282
c.	Trade Payables, January 1, Year 7 (BS)	$ 42,236
	Purchases of Supplies during Year 7 (SCF)	15,162
	Payments to Suppliers during Year 7 (SCF)	?
	Trade Payables, December 31, Year 7 (BS)	43,152
d.	Retained Earnings, January 1, Year 7 (BS)	$ 88,453
	Net Income for Year 7 (IS)	21,169
	Dividends Declared and Paid during Year 7 (SCF)	8,106
	Retained Earnings, December 31, Year 7 (BS)	?

Figure 7.4: Chapter3-10

Answer

(a)

$$Accounts\ Receivable\ in\ Jan. + 288,951 - 289,623 = 38,020$$

$$Accounts\ Receivable\ in\ Jan. = \$38,692$$

(b)

$$2,635 + 10,442 - Payments\,to\,Gov. = 3,282$$

$$Payments\,to\,Gov. = \$9,795$$

(c)

$$42,236 + 15,162 - Payments\,to\,suppliers = 43,152$$

$$Payments\,to\,suppliers = \$14,246$$

(d)

$$88,453 + 21,169 - 8,106 = Retained\,Earnings\,in\,Dec.$$

$$Retained\,Earnings\,in\,Dec. = \$101,516$$

재무회계 Ch4 과제

(1) Quetion 4-9

9. For each of the following items, indicate whether the item meets all of the criteria in the definition of a liability. If so, how does the firm value it?
   a. Interest accrued but not paid on a note.
   b. Advances from customers for goods and services to be delivered later.
   c. Confirmed orders from customers for goods and services to be delivered later.
   d. Product warranties.
   e. Damages the company must pay if it loses a pending lawsuit.
   f. Contractual promises to purchase specific quantities of natural gas for each of the next 10 years.
   g. Promises by an airline to provide flights in the future in exchange for miles flown, if customers accumulate a certain number of miles at regular fares.

Figure 7.5: Chapter4-9

First, the criteria in the definition of a liablilty is following :

(1) Probably, future sacrifices of economic benefits.

(2) Arising from present obligation to transfer assets of privide services.

(3) As a result of a past transaction.

In this perspective,

(a) is a liability. accrued interest payable.

(b) is a liability.(Advances from customers) It is equal to amount of received cash.

(c) is not a liability. criteria (3) is not satisfied.

(d) is a liability.(Provisions) It can be valued by some ratio of original product's price.

(e) can be liability or not. It depends on a probability of lawsuit.

If a probability of loss lawsuit is more than certain level,(In GAAP 80%, IFRS 50%) then it will be a liability. In that case, it can be valued by expected loss that should pay.

(f) is not a liability. criteria (3) is not satisfied.

(g) is a liability. It can be valued by some ratio of flight ticket's regular price.

(2) Quetion 4-15

15. Balance sheet relations. Jennings Group, a Malaysian investment management company, reported the following data for four recent years. Jennings applies Malaysian accounting standards and reports its results in millions of ringgit (RM). Compute the missing balance sheet amounts for each of the four years. In answering this question, assume that Jennings uses U.S. GAAP. (Adapted from the financial statements of Genting Group.)

	Year 7	Year 6	Year 5	Year 4
Noncurrent Assets	?	RM18,717.4	RM11,289.1	RM9,713.9
Shareholders' Equity	RM21,537.3	16,666.9	9,002.0	?
Total Assets	?	28,224.7	?	?
Current Liabilities	?	4,351.3	1,494.2	1,755.2
Current Assets	10,999.2	?	?	6,882.6
Noncurrent Liabilities	5,721.7	?	?	3,540.7
Total Liabilities and Shareholders' Equity	30,178.9	?	18,491.3	?

Figure 7.6: Chapter4-15

Total Assets = Current Assets + Noncurrent Assets

Total Liabilities = Current Liabilities + Noncurrent Liabilities

Total Liabilities and Shareholders' Equity = Total Liabilities + Shareholders' Equity

And, *Total Assets = Total Liabilities and Shareholders' Equity*

Apply above,

Category	Year7	Year6	Year5	Year4
Noncurrent Assets	**19,179.7**	18,717.4	11,289.1	9,713.9
Shareholders' Equity	21,537.3	16,666.9	9,002.0	**11,300.6**
Total Assets	**30,178.9**	28,224.7	**18,491.3**	**16,596.5**
Current Liabilities	**2,919.9**	4,351.3	1,494.2	1,755.2
Current Assets	10,999.2	**9,507.3**	**7,202.2**	6,882.6

Category	Year7	Year6	Year5	Year4
Noncurrent Liabilities	5,721.7	**7,206.5**	**7,995.1**	3,540.7
Total Liabilities and Equity	30,178.9	**28,224.7**	18,491.3	**16,596.5**

재무회계 Ch5 과제

(1) Question 5-12

12. Revenue recognition. Fonterra Cooperative Group Limited (Fonterra), a New Zealand dairy cooperative, uses the accrual basis of accounting and recognizes revenue at the time it sells products or renders services. Fonterra applies New Zealand accounting standards and reports its results in millions of New Zealand dollars (NZ$). In answering this problem, assume Fonterra uses IFRS. Indicate which of the following transactions or events immediately give rise to Fonterra's recognition of revenue.

a. Fonterra has completed the pasteurization of an order of 13,000 liters of milk it will deliver to a grocery store chain next week. Fonterra has not yet delivered the milk or invoiced the grocery store. The selling price of the milk is NZ$26,000.

b. Refer to part **a**, and assume that the grocery store paid Fonterra a deposit of NZ$5,000 on the order of the milk.

c. Fonterra delivered the milk and billed the grocery store. The grocery store has not yet paid the invoice.

d. One day after delivery, the grocery store called Fonterra and reported that it has to destroy 3,000 liters of milk because it had spoiled sometime prior to its delivery. The grocer refuses to pay for these 3,000 liters.

e. Fonterra spent NZ$10 million to develop a technique to transform a by-product of casein (a protein found in milk and cheese) into ethanol. Fonterra expects to use this technique to generate sales of at least NZ$2 million over the next year.

f. Refer to part **e**, and assume that Fonterra signed contracts worth NZ$400 million for ethanol sales.

Figure 7.7: Chapter5-12

(a) NOT recognized. It is before a completion of the earning process.

(b) NOT recognized. It is before a completion of the earning process.

(c) **Revenue recoginzed.** There is a completion of the earning process and receipt.(account receivable)

(d) NOT recognized. It can be an expense.

(e) NOT recognized. It is an investment, which is an expense.

(f) NOT recognized. There is no completion of the earning process.

(2) Question 5-18

18. Income statement relations. Selected income statement information for Years 11, 12, and 13 for SwissTek, a Swiss engineering firm. SwissTek applies U.S. GAAP and reports its results in millions of U.S. dollars.

	Year 13	Year 12	Year 11
Sales of Products	$24,816	?	$17,622
Income Tax Expense	595	?	464

	Year 13	Year 12	Year 11
Earnings Before Interest and Taxes	4,023	2,557	?
Sales of Services	4,367	3,778	3,342
Selling and Administrative Expenses	4,975	?	3,780
Cost of Services Sold	?	2,570	2,305
Income Before Taxes	?	2,076	1,199
Other Operating Income (Expense)	?	139	37
Interest and Other Financial Expense	286	?	407
Gross Profit	8,968	6,744	?
Cost of Products Sold	17,292	13,967	13,205
Other Non-operating Income (Expense)	?	(321)	(258)
Interest and Dividend Income	273	147	?
Net Income	3,757	1,390	?

Compute the missing amounts for each of the three years.

Figure 7.8: Chapter5-18

The following table is the re-organized income statement.

Category	Year 13	Year 12	Year 11
(+) Sales of Products	24,816	**19,503**	17,622
(-) Cost of Products Sold	17,292	13,967	13,205
(+) Sales of Service	4,367	3,778	3,342
(-) Cost of Services Sold	**2,923**	2,570	2,305
(=) **Gross Profit**	8,968	6,744	**5,454**
(-) Selling and Administrative Expenses	4,975	**4,326**	3,780
(+) Other Operating Income	**(30)**	139	37
(=) **Earning Before Interest and Taxes**	4,023	2,557	**1,711**
(+) Other Non-operating Income	**342**	(321)	(258)

Category	Year 13	Year 12	Year 11
(+) Interest and Dividend Income	273	147	**153**
(-) Interest and Other Financial Expense	286	**307**	407
(=) **Income Before Taxes**	**4,352**	2,076	1,199
(-) Income Tax Expense	595	686	464
(=) **Net Income**	3,757	1,390	**735**

재무회계 Ch6 과제

14. **Derive sales revenue from data in the statement of cash flows and balance sheet.** Microchem Corporation reported a balance of €5,196 million in accounts receivable at the beginning of the year and €5,334 million at the end of the year. Its statement of cash flows using the direct method reported cash collections from customers of €33,551 million for the year. Assuming that Microchem Corporation makes all sales on account, compute the amount of sales during the year.

Answer

Because a company makes all sales on account, **total sales is sum of cash collections and account receivable.**

There are EUR33,551m cash collections and EUR138m increase in account receivavle, so 33551+138=**EUR33,689m will be a total sales** of company during the year.

16. **Derive cost of goods sold from data in the statement of cash flows.** The section showing cash flow from operations, using the indirect method, for Taylor Stores reported an increase in inventories of $5.7 million during the year. It reported also that the balance in accounts payable for inventories increased by $5.9 million. The direct method would show cash payments for merchandise inventory purchased of $646.9 million. Compute Taylor's cost of goods sold for the year.

Answer

Cost will be **cash payments + increase payable - increase inventories**.

646.9+5.9-5.7=**USD647.1m is cost of goods sold** of company for the year.

26. Preparing a statement of cash flows from changes in balance sheet accounts. The comparative balance sheets of Incloud Airlines show the following information for a recent year (amounts in thousands of US$):

Change	Amount	Direction
Cash	$ 40,308[a]	Increase
Accounts Receivable	15,351	Decrease
Inventories	15,117	Increase
Prepayments	16,776	Increase
Property, Plant, and Equipment (at Cost)	1,134,644[b]	Increase
Accumulated Depreciation	264,088[b]	Increase
Other Non-operating Assets	8,711	Increase
Accounts Payable	660	Decrease
Other Current Liabilities	114,596	Increase
Long-Term Debt	244,285	Increase
Other Non-operating Liabilities	140,026	Increase
Common Stock	96,991	Increase
Retained Earnings	340,879[c]	Increase

[a]Cash was $378,511 at the beginning of the year and $418,819 at the end of the year.

[b]Incloud Airlines did not sell any property, plant, and equipment during the year.

[c]Net income was $474,378.

a. Prepare a statement of cash flows for Incloud Airlines for the year. Treat changes in non-operating assets as investing transactions and changes in non-operating liabilities as financing transactions.

Answer

Incloud Airlines

Statement of Cash Flow for the recent year(thousands$)

Category	Name	Amount
Cashflow from operations		**$835,860**
+	Net income	474378
+	Increase in accumulated depreciation	264088
+	Decrease in account receivable	15351
-	Increase in other current liabilities	114596
-	Increase in inventories	15117

Category	Name	Amount
-	Increase in Prepayments	16776
-	Decrease in account payable	660
Cashflow from investing		**($1,143,355)**
-	Increase in property, plant, equipment	1134644
-	Increasing in other non-operating assets	8711
Cashflow from financing		**$347,803**
+	Increasing in long-term debt	244285
+	Increasing in common stock	96991
+	Increasing in non-operating liabilities	140026
-	Payment of dividends	133499
Net Change in Cash		**$40,308**
Cash, beginning of year		**$378,511**
Cash, end of year		**$418,819**

29. Calculating and interpreting cash flows. Market Star is a marketing services firm that creates advertising copy for clients and places the advertising in television, magazines, and other media. Accounts receivable represent amounts owed by clients, and accounts payable represent amounts payable to various media. Market Star has purchased other marketing services firms in recent years. Selected data for Market Star for three recent years appear next (amounts in millions of US$):

	2013	2012	2011
Net Income	$ 499	$ 363	$279
Depreciation and Amortization Expense	226	196	164
Increase (Decrease) in Accounts Receivable	514	648	238
Increase (Decrease) in Inventories	98	13	35
Increase (Decrease) in Prepayments	125	(10)	64
Increase (Decrease) in Accounts Payable	277	786	330
Increase (Decrease) in Other Current Liabilities	420	278	70
Acquisition of Property, Plant, and Equipment	150	130	115
Acquisition of Investments in Securities (noncurrent)	885	643	469
Dividends Paid	122	104	88
Long-Term Debt Issued	599	83	208
Common Stock Issued (Reacquired)	(187)	(252)	42

a. Prepare a comparative statement of cash flows for Market Star for the three years. Use the indirect method of computing cash flow from operations.

Answer

Market Star

Statement of Cash Flow for 2011 $\sim$ 2013(*millions*\$)

Category	Name	2013	2012	2011
Cashflow from operations		**\$685m**	**\$972m**	**\$506m**
+	Net income	499	363	279
+	Increase in depreciation and amortizaion	226	196	279
+	Increase in account payable	277	786	330
-	Increase in account receivable	514	648	238
-	Increase in other current liabilities	420	278	70
-	Increase in inventories	98	13	35
-	Increase in Prepayments	125	(10)	64
Cashflow from investing		**(\$1,035)**	**(\$773)**	**(\$584)**
-	Increase in property, plant, equipment	150	130	115
-	Increase in investments in securities	885	643	469
Cashflow from financing		**\$290**	**(\$273)**	**\$162**
+	Increasing in long-term debt	599	83	208
+	Increasing in common stock	(187)	(252)	42
-	Payment of dividends	122	104	88
Net Change in Cash		**(\$60)**	**(\$74)**	**\$84**

재무회계 Ch8 과제

16. **Revenue recognition at time of sale and advances from customers.** Pret a Manger is a food retailer with stores in the United Kingdom and the United States and is known for its fast but fresh food menu. A customer shopping at a London Heathrow store purchased a ham and cheese baguette (£4.50), a small fruit salad (£2.40), and a banana-bran muffin (£1.50). The customer paid with cash.
    a. What journal entry will Pret a Manger record for this transaction?
    b. Suppose that, in addition to the above items, the customer purchased a Pret a Manger card (to use for future purchases at Pret a Manger stores) for £40.00. What journal entry will Pret a Manger record for this transaction?
    c. Suppose that, for the original transaction described in this problem, the customer did not pay with cash but used a Pret a Manger card purchased a month earlier. Assume there is a sufficient balance on the card to cover the cost of his purchases. What journal entry did Pret a Manger record?

Answer

(***a***)

debit : cash EUR 8.4

credit : revenue EUR 8.4

(***b***)

debit : cash EUR 48.4

credit : revenue EUR 8.4 / advances from customer EUR 40

(***c***)

debit : advances from customer EUR 8.4

credit : revenue EUR 8.4

26. Reconstructing events when using the allowance method. Selected data from the accounts of Seward Corporation appear next; the firm's fiscal year ends on December 31.

	January 1	December 31
Accounts Receivable, Gross	$82,900 Dr.	$ 87,300 Dr.
Allowance for Uncollectible Accounts	8,700 Cr.	9,100 Cr.
Bad Debt Expense	—	4,800 Dr.
Sales Revenue	—	240,000 Cr.

The firm makes all sales on account. There were no recoveries during the year of accounts written off in previous years.

Give the journal entries for the following transactions and events during the year:

a. Sales on account.

b. Recognition of bad debt expense.

c. Write-off of uncollectible accounts.

d. Collection of cash from customers from sales on account.

(***a***)

debit : account receivable USD 240,000

credit : sales revenue UST 240,000

(***b***)

debit : bad debt expense USD 4,800

credit : allowance for uncollectible accounts USD 4,800

(***c***)

debit : allowance for uncollectible accounts USD 4,400

credit : account receivable USD 4,400

(***d***)

debit : cash USD 231,200

credit : account receivavle USD 231,200

34. Percentage-of-completion and completed contract methods of income recognition. The Shannon Construction Company agreed to build a warehouse for $6,000,000. Expected and actual costs to construct the warehouse were as follows: 2012, $1,200,000; 2013, $3,000,000; and 2014, $600,000. The firm completed the warehouse in 2014. Compute revenue, expense, and income before income taxes for 2012, 2013, and 2014 using the percentage-of-completion method and the completed contract method.

category	2014	2013	2012
Cost	600000	3000000	1200000
Ratio	12.5%	62.5%	25.0%
Revenue	750000	3750000	1500000
Income	150000	750000	300000

재무회계 Ch9 과제

17. Effect of inventory valuation on the balance sheet and net income. ResellFast purchases residential and commercial real estate for resale. ResellFast has a December 31 year-end and prepares financial statements quarterly. On February 5, 2012, ResellFast acquired an open-air mall in Miami, Florida, with space for 15 retail businesses, for $20 million. On April 12, 2012, a storm flooded a portion of the mall, reducing the mall's fair value to $16.5 million. On August 14, 2012, a large retailer announced plans to build a new store adjacent to the open-air mall; this announcement, combined with ResellFast's repairs, attracted several smaller retailers to inquire about acquiring space in the open-air mall. By September 30, 2012, the fair value of the open-air mall had increased to $26 million. On November 8, 2012, ResellFast sold the mall for $27.5 million. Compute the carrying value of the open-air mall on ResellFast's balance sheet and the related income statement effect for each quarter of 2012. ResellFast follows U.S. GAAP.

Answer

Category	Inventory	Income
Querter 1	USD 20m	-
Querter 2	USD 16.5m	-USD 3.5m
Querter 3	USD 16.5m	-
Querter 4	-	+USD 11m

21. Income computation for a manufacturing firm. The following data relate to GenMet, a U.S.-based consumer goods manufacturing firm, for the fiscal year ending October 31, 2013. Reported amounts are in millions of U.S. dollars ($).

	October 31, 2013	October 31, 2012
Raw Materials Inventory	$101.5	$ 73.7
Work-in-Process Inventory	119.1	100.8
Finished Goods Inventory	322.3	286.2

GenMet incurred manufacturing costs (direct material, direct labor, manufacturing overhead) during fiscal 2013 totaling $2,752.0. Sales revenue was $6,700.2, selling and administrative expenses were $2,903.7, and interest expense was $151.9. The income tax rate is 35%. Round arithmetic computations to one digit after the decimal point. Compute GenMet's net income for fiscal year 2013.

Cost of sold	Amount(m$)
Total manufacturing cost	2752.0
Work in process inventory	100.8-119.1=(18.3)
Finished goods inventory	286.2-322.3=(36.1)
Cost of sold	2697.6

Net income	Amount(m$)
Sales	6700.2
Cost of sold	(2697.6)
Selling and ad. expenses	(2903.7)
Interest expense	(151.9)
Income tax	947*35%=(331.5)
Net income	**615.5**

26. Computations involving different cost-flow assumptions. Sun Health Food's purchases of vitamins during its first year of operations were as follows:

	Quantity	Cost per Unit	Total Cost
January 5 Purchase	460	$4.30	$ 1,978
April 16 Purchase	670	4.20	2,814
August 26 Purchase	500	4.16	2,080
November 13 Purchase	870	4.10	3,567
Totals	2,500		$10,439

The inventory on December 31 was 420 units. Compute the cost of the inventory on December 31 and the cost of goods sold for the year under each of the following cost-flow assumptions:

a. FIFO.

The amount of sales is 2080.

Under FIFO, cost will be : $1978 + 2814 + 2080 + 450 \times 4.1 = 8717$

재무회계 Ch10 과제

Problem 10-21

21. Change in depreciable life and salvage value. Thom Corporation acquired a computer on January 1, 2011, for $10,000,000. The computer had an estimated useful life of six years and $1,000,000 estimated salvage value. The firm uses the straight-line depreciation method. On January 1, 2013, Thom Corporation discovers that new technologies make it likely that the computer will last only four years in total and that the estimated salvage value will be only $600,000. Compute the amount of depreciation expense for 2013 for this change in depreciable life and salvage value. Assume that the change does not represent an impairment loss.

Answer

$\text{Carrying value} : 10,000,000 - (2 \times \frac{10,000,000-1,000,000}{6}) = 7,000,000USD$

$\text{Depreciation} : \frac{7,000,000-600,000}{2} = 3,200,000USD$

Problem 10-29

29. Recording transactions involving tangible and intangible assets. Present journal entries for each of the following transactions of Moon Macrosystems:

a. Acquired computers costing $400,000 and computer software costing $40,000 on January 1, 2011. Moon expects the computers to have a service life of 10 years and $40,000 salvage value. It expects the computer software to have a service life of four years and zero salvage value.

b. Paid $20,000 to install the computers in the office. Paid $10,000 to install and test the computer software.

c. Recorded depreciation and amortization using the straight-line method for 2011 and 2012. Moon records a full year of depreciation in the year of acquisition. Treat depreciation and amortization as a period expense.

d. On January 1, 2013, new software offered on the market made the software acquired in part **a** completely obsolete. Give any required journal entry.

e. On January 2, 2013, Moon revised the depreciable life of the computers to a total of 14 years and the salvage value to $56,000. Give the entry to record depreciation for 2013.

f. On December 31, 2014, Moon sold the computers for $260,000. Give the required journal entries for 2014.

Answer

(a)

debit : Computer 400,000 / Software 40,000

credit : Cash 440,000

(b)

debit : Computer 20,000 / Software 10,000

credit : Cash 30,000

(c)

debit : Depreciation 76,000(=2(*420000-40000)/10*) / *Amortization 25,000*(=*2*50000/4)

credit : Accumulated Depreciation 76,000 / Software 25,000

(d)

debit : Loss of software 25,000

credit : Software 25,000

(e)

debit : Depreciation 24,000(=(420000-38000*2-56000)/12)

credit : Accumulated depreciation 24,000

(f)

debit : Depreciation 24,000 / Cash 260,000 / Accumulated depreciation 124,000 / Loss of computer 36,000

credit : Accumulated depreciation 24,000 / Computer 420,000

재무회계 Ch11 과제

Problem 11-18.

18. Amortization schedule for bonds. On January 1 of the current year, Womack Company issues 10% semiannual coupon bonds maturing five years from the date of issue. The firm issues the bonds to yield 8% compounded semiannually. The bonds have a face value of $100,000.

a. Compute the initial issue proceeds of these bonds.

b. Construct an amortization schedule, similar to that in **Exhibit 11.2**, for this bond issue, assuming that Womack Company uses amortized cost measurement based on the historical market interest rate to account for the bonds.

Answer

(a)

Initial Price of bond : Present value of bond's cashflow

$= \sum_{i=1}^{10} \frac{5000}{(1.04)^i} + \frac{100000}{(1.04)^{10}} = 108111$

Therefore, it is 108,111$

(b)

Six-month period	Liability at start of period	Interest at 4% for Period	Cash payment	Decrease in Carrying Value of Liability	Liability at End of Period
0					$108,111
1	$108,111	$4,324	$5,000	$(676)	107,435
2	107,435	4,297	5,000	(703)	106,732
3	106,732	4,269	5,000	(731)	106,001
4	106,001	4,240	5,000	(760)	105,241
5	105,241	4,210	5,000	(790)	104,451
6	104,451	4,178	5,000	(822)	103,629
7	103,629	4,145	5,000	(855)	102,774
8	102,774	4,111	5,000	(889)	101,885
9	101,885	4,075	5,000	(925)	100,960
10	100,960	4,040	5,000	(960)	100,000
Total		$41,889	$50,000	$(8,111)	

Problem 11-19.

19. Amortization schedule for bonds. On January 1, 2012, Seward Corporation issues $100,000 face value, 8% semiannual coupon bonds maturing three years from the date of issue. The coupons, dated for June 30 and December 31 of each year, each promise 4% of the face value, 8% total for a year. The firm issues the bonds to yield 10%, compounded semiannually.

a. Compute the initial issue proceeds of these bonds.

b. Construct an amortization schedule, similar to that in **Exhibit 11.2**, for this bond issue, assuming that Seward Company uses amortized cost measurement based on the historical market interest rate to account for the bonds.

c. Give the journal entries related to these bonds for 2012. Seward uses the calendar year as its reporting period.

Answer

(a)

Initial Price of bond : Present value of bond's cashflow

$= \sum_{i=1}^{6} \frac{4000}{(1.05)^i} + \frac{100000}{(1.05)^6} = 94925$

Therefore, it is 94,025$

(b)

Six-month period	Liability at start of period	Interest at 5% for Period	Cash payment	Increase in Carrying Value of Liability	Liability at End of Period
1	$94,925	$4,746	$4,000	$746	$95,671
2	95,671	4,784	4,000	784	96,455
3	96,455	4,823	4,000	823	97,278
4	97,278	4,864	4,000	864	98,142
5	98,142	4,907	4,000	907	99,049
6	99,049	4,951	4,000	951	100,000
Total		$29,075	$24,000	$5,075	

(c)

January 2

debit : Cash 94925

credit : Bonds payable 94925

June 30

debit : interest expense 4746

credit : Cash 4000 / Bonds payable 746

December 31

debit : interest expense 4746

credit : Cash 4000 / Bonds payable 746

Exhibit 11.2

EXHIBIT 11.2

Amortization Schedule for $125,000 Loan, Repaid in Nine Semiannual Installments of $17,000 and a Final Payment of $16,781. Interest Rate Is 12% Compounded Semiannually (6% compounded each six months)[4]

Period (1)	Balance at Beginning of Period (2)	Interest Expense for Period (3)	Cash Payment (4)	Portion of Payment Reducing Principal (5)	Balance at End of Period (6)
0					$125,000
1	$125,000	$7,500	$17,000	$ 9,500	115,500
2	115,500	6,930	17,000	10,070	105,430
3	105,430	6,326	17,000	10,674	94,756
4	94,756	5,685	17,000	11,315	83,441
5	83,441	5,006	17,000	11,994	71,448
6	71,448	4,287	17,000	12,713	58,734
7	58,734	3,524	17,000	13,476	45,259
8	45,259	2,716	17,000	14,284	30,974
9	30,974	1,858	17,000	15,142	15,832
10	15,832	950	16,782	15,832	0

Note: In preparing this table, we rounded numbers to the nearest dollar, but the underlying computations, in Excel®, used several significant digits after the decimal point.

Column (2) = Column (6) from previous period.
Column (3) = 0.06 × Column (2), except for period 10, where it is the amount such that Column (3) = Column (4) − Column (5).
Column (4) is given.
Column (5) = Column (4) − Column (3).
Column (6) = Column (2) − Column (5).

Part V

경영통계분석('24봄)

경영통계분석 과제1

1. Type of variables

Question

Identify whether the following variables are numerical or categorical. If numerical, state whether the variable is discrete or continuous. If categorical, state whether the variable is nominal or ordinal.

a. Number of companies going bankrupt in a year.

: Numerical and Discrete

b. Petal area of rose flowers.

: Numerical and Continuous

c. Key on the musical scale.

: Categorical and Ordinal

d. Heart beats per minute of a Tour de France cyclist, averaged over the duration of the race.

: Numerical and Continuous

e. Stage of fruit ripeness.

: Categorical and Ordinal

f. Angle of flower orientation relative to position of the sun.

: Numerical and Continuous

g. Tree species

: Categorical and Nominal

h. year of birth

: Numerical and Discrete

i. Gender

: Categorical and Nominal

j. Birth weight

: Numerical and Continuous

2. Discrete data

Question

Birds of the Caribbean islands of the Lesser Antilles are descended from immigrants originating from larger islands and the nearby mainland. The data presented here are the approximate dates of immigration, in millions of years, of each of 37 bird species now present on the Lesser Antilles. The dates were calculated from the difference in mitochondrial DNA sequences between each of the species and its closest living relative on larger islands or the mainland.

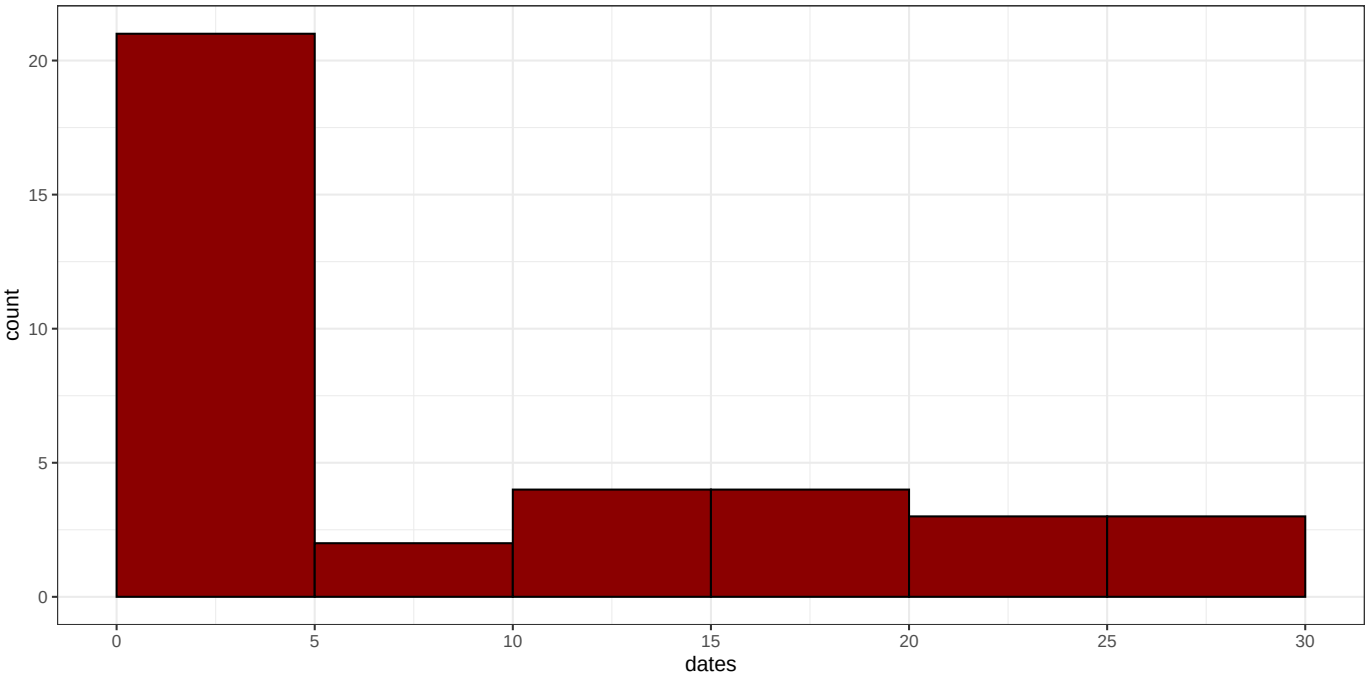
```
library(tidyverse)
birds_caribbean <- tibble("dates"=c(0.00,0.00,0.04,0.21,0.29,
0.54,0.63,0.88,0.96,1.25,
1.67,1.75,1.84,1.96,2.01,
2.51,2.72,3.30,3.51,4.05,
4.85,6.94,8.73,10.57,11.11,
12.45,14.00,17.30,17.92,18.05,
18.43,22.48,22.48,23.48,26.32,
26.45,28.87))
```

```
birds_caribbean$dates
```

```
 [1]  0.00  0.00  0.04  0.21  0.29  0.54  0.63  0.88  0.96  1.25  1.67  1.75
[13]  1.84  1.96  2.01  2.51  2.72  3.30  3.51  4.05  4.85  6.94  8.73 10.57
[25] 11.11 12.45 14.00 17.30 17.92 18.05 18.43 22.48 22.48 23.48 26.32 26.45
[37] 28.87
```

a. Plot the data in a histogram and describe the shape of the frequency distribution.

```
ggplot(data=birds_caribbean, aes(x=dates))+
  geom_histogram(binwidth=5, boundary=0, color="black",fill="darkred")+
  scale_x_continuous(breaks = seq(0,30,5)) +
  theme_bw()
```



히스토그램은 위와 같으며, 0~5구간에 전체 37종의 조류 중 20종 이상이 넘는 빈도가 집중되어 있습니다. 이는 과반 이상의 조류가 비교적 최근인 500만년 이내에 Lesser Antilles섬으로 이주해왔다는 것을 의미합니다. 나머지 약 15종의 조류는 각각 2~4종씩 ~1천만년, ~1500만년, ⋯ , ~3천만년 구간에 고르게 분포되어 있습니다.

b. By viewing the graph alone, approximate the mean and median of the distribution. Which should be greater? Explain your reasoning.

: 히스토그램은 왼쪽으로 치우쳐져 있으며, 이는 $Skewness >> 0$이라는 것을 말합니다. 이러한 경우 평균은 중간값보다 크게 형성되고, 오른쪽 꼬리가 다소 긴 점을 감안하여 추정해보면, **중간값은 약 4, 평균은 약 10**일 것으로 보입니다.

c. Calculate the mean and median. Was your intuition in part (b) correct?

```
birds_caribbean %>%
  summarise(mean=mean(dates),
            mid=median(dates))
```

```
# A tibble: 1 x 2
   mean   mid
  <dbl> <dbl>
```

```
1  8.66  3.51
```

: 대략적인 경향성은 맞습니다. 실제 중간값은 3.51 및 평균은 8.66입니다.

d. Calculate the first and third quantiles and the inter quartile range.

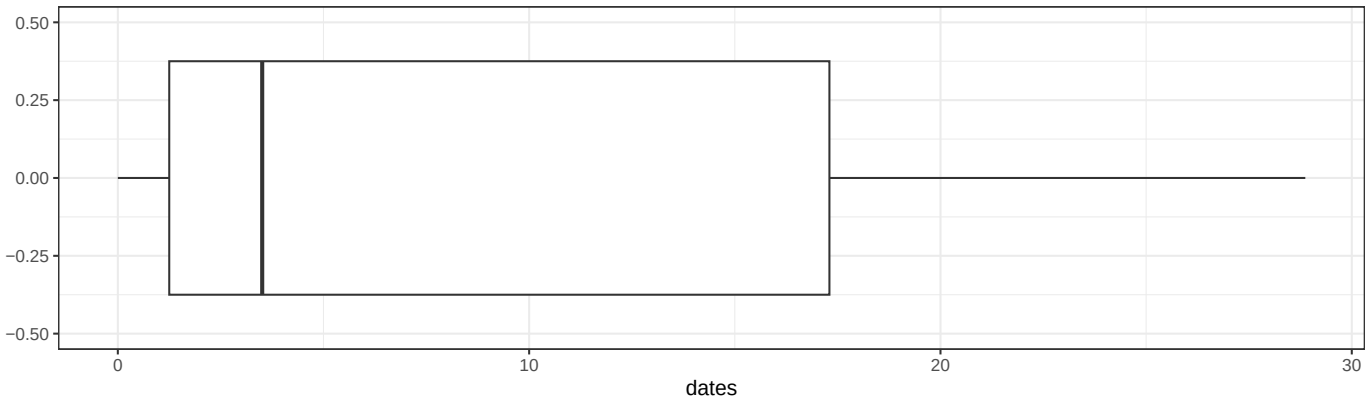
```
birds_caribbean %>%
  reframe(quantile=quantile(dates))
```

```
# A tibble: 5 x 1
  quantile
     <dbl>
1     0
2     1.25
3     3.51
4    17.3
5    28.9
```

: First(25%) quantiles은 1.25이며, third(75%) quantiles은 17.30입니다. IQR은 16.05입니다.

e. Draw a box plot for these data

```
ggplot(birds_caribbean, aes(x=dates))+
  geom_boxplot()+
  scale_y_continuous(limits=c(-0.5,0.5))+
  theme_bw()
```



3. Histogram

Question

Francis Galton presented the following data on the flight speeds of 3207 "old" homing pigeons traveling at least 90 miles.

a. What type of graph is this?

: Histogram

b. Examine the graph and visually determine the approximate value of the mean (to the nearest 100 yards per minute). Explain how you obtained your estimate.

: 그래프는 bell-shape의 히스토그램이며, 오른쪽 꼬리가 긴 형태입니다. 아마 $skewness > 0$일 것으로 보이며, 평균은 중간값보다 클 것 입니다.

그래프에서 중간값은 오른쪽 꼬리가 긴 분포를 고려할 때 약 1,000일 것으로 추정할 수 있는데, **평균값은 이보다 큰 약 1,100**이 될 것으로 추정됩니다.

c. Examine the graph and visually determine the approximate value of the median (to the nearest 100 yards per minute). Explain how you obtain your estimate.

: 그래프에서 900~1000이 중심막대인데, bell-shape와 오른쪽 꼬리가 긴 분포를 고려할 때, **중간값은 약 1,000**이 될 것으로 추정됩니다.

d. Examine the graph and visually determine the approximate value of the mode (to the nearest 100 yards per minute). Explain how you obtained your estimate.

: 최빈값은 가장 많은 빈도를 가지는 값 또는 구간으로, 히스토그램의 가장 높은 막대인 1000~1100에 존재할 것입니다. 막대 왼쪽에 빈도가 더 많은 점을 고려할 때 **최빈값은 약 1,000**일 것으로 보입니다.

e. Examine the graph and visually determine the approximate value of the standard deviation (to the nearest 100 yards per minute). Explain how you obtained your estimate.

: 먼저, 정규분포를 따르는 확률변수에 대해 다음 식이 성립합니다.

$$for\ X\ \sim\ Normal\ Dist.\ then\ P[-\sigma < X < \sigma] \approx 68.2\%$$

문제의 히스토그램은 bell-shape로 정규분포를 따른다고 가정하는 것에 큰 문제는 없어보입니다. 히스토그램에서 어림잡아봤을 때, 약 $1,100 \pm 200$ 구간이 70~80%에 해당한다고 보여지는데, ($P[900 < x < 1300] = 70 \sim 80\%$)

이는 (b)의 추정 평균을 이용하면 ± 200입니다. 즉, 히스토그램이 정규분포를 따른다고 가정하면, 히스토그램의 형태는 $N(1050, \sigma^2 = (200-\alpha)^2)$의 확률밀도함수가 유사할 것입니다. 한편, 히스토그램은 오른쪽 꼬리가 긴 형태를 가지고 있어 변동성은 정규분포 대비 다소 클 것입니다.

따라서, 최종적으로 **히스토그램 데이터의 표준편차는 약 200으로 추정**됩니다.

4. Sample Statistics

```
handheight <- tibble("person"=c("A","B","C","D","E"),
                     "hand"=c(17,15,19,17,21),
                     "height"=c(150,154,169,172,175))
handheight
```

```
# A tibble: 5 x 3
  person  hand height
  <chr>  <dbl>  <dbl>
1 A         17    150
2 B         15    154
3 C         19    169
4 D         17    172
5 E         21    175
```

a. Calculate the sample variances for hand and height, respectively.

표본 분산은 $s_x^2 = \frac{\sum_{k=1}^{n}(X_k - \bar{X})^2}{n-1}$ 입니다.

```
sample_var <- handheight %>%
  mutate(hand_tmp=hand-mean(handheight$hand),
         height_tmp=height-mean(handheight$height)) %>%
  mutate(hand_tmp=hand_tmp^2,
         height_tmp=height_tmp^2) %>%
  summarise(s_var_hand=sum(hand_tmp)/4,
            s_var_height=sum(height_tmp)/4)
sample_var
```

```
# A tibble: 1 x 2
```

```
  s_var_hand s_var_height
       <dbl>         <dbl>
1        5.2          126.
```

b. Calculate the sample covariance.

표본 공분산은 $s_{xy} = \frac{\sum_{k=1}^{n}(X_k-\bar{X})(Y_k-\bar{Y})}{n-1}$ 입니다.

```
sample_cov <- handheight %>%
  mutate(s_cov_tmp=(hand-mean(handheight$hand))*(height-mean(handheight$height))) %>%
  summarise(s_cov=sum(s_cov_tmp)/4)
sample_cov
```

```
# A tibble: 1 x 1
  s_cov
  <dbl>
1  18.5
```

c. Calculate the sample correlation and interpret the result.

표본 상관계수는 $r_{xy} = \frac{s_{x,y}}{s_x s_y}$ 입니다.

```
sample_corr=sample_cov$s_cov/sqrt(sample_var$s_var_hand*sample_var$s_var_height)
sample_corr
```

```
[1] 0.7213147
```

표본상관계수는 약 0.72이며, 이는 두 변수간에 강한 양의 선형관계가 존재하는 것을 말합니다. 즉, 손의 크기와 키 사이에는 양의 상관관계가 있어 손이 큰 집단은 키도 큰 경향이 있고, 키가 큰 집단은 손도 큰 경향이 있다는 것을 의미합니다.

> R 내장함수
>
> 참고로, R 내장함수는 표본연산을 기본으로 하고 있어 내장함수를 이용하여 표현 가능합니다.
>
> ```
> var(handheight$hand)
> ```
>
> ```
> [1] 5.2
> ```
>
> ```
> var(handheight$height)
> ```

```
[1] 126.5
```

```
cov(handheight$hand,handheight$height)
```

```
[1] 18.5
```

```
cor(handheight$hand,handheight$height)
```

```
[1] 0.7213147
```

경영통계분석 과제2

Question 1

At one large midwest university, about 40% of the college seniors have a social science major. Five seniors will be selected at random. Let X denote the number that **don't** have a social science major.

(a) List the probability distribution

학생을 선정할 때, 사회과학 전공이 아닌 경우를 1로, 사회과학 전공인 경우를 0으로 하면 이는 확률 0.6의 독립된 베르누이 시행으로 볼 수 있습니다.

따라서, 이를 5번 반복하는 확률변수 X는 $X \sim B(5, 0.6)$를 따른다고 볼 수 있으며, $P(X = k) = \binom{5}{k} 0.6^k 0.4^{(5-k)}$ 이므로 이에 따른 확률분포는 아래와 같습니다.

k X	0	1	2	3	4	5
P(X=k)	0.0102	0.0768	0.230	0.346	0.259	0.0778

```
library(tidyverse)
```

```
binomial <- tibble(k=c(0,1,2,3,4,5), prob=choose(5,k)*0.6^k*0.4^(5-k))
binomial
```

```
# A tibble: 6 x 2
      k   prob
  <dbl>  <dbl>
1     0 0.0102
2     1 0.0768
3     2 0.230
4     3 0.346
5     4 0.259
6     5 0.0778
```

(b) Calculate mean and variance from the entries in the list from part (a).

평균은 3, 분산은 1.2입니다.

```
mean=sum(binomial$k*binomial$prob)
variance=sum(binomial$k^2*binomial$prob)-mean^2
paste(mean,variance,sep=" / ")
```

```
[1] "3 / 1.2"
```

(c) Calculate E(X)=np and Var(X)=np(1-p) and compare your answer with part (b).

np와 np(1-p)의 값은 각각 3, 1.2로, 평균 및 분산과 같습니다.

Question 2

If X has a normal distribution with = 100 and = 5 , find b such that

(a) P(X<b)=0.67

b=102.2

```
# Set parameter
mean <- 100; std <- 5
# (a) P(X<b)=0.67, Using bisection method
i <- mean-3*std;j <- mean+3*std;mid <- (i+j)/2
while(abs(pnorm(mid,mean,std)-0.67)>0.000001){
  if(pnorm(mid,mean,std)-0.67>=0){j <- mid}
  if(pnorm(mid,mean,std)-0.67<0){i <- mid}
  mid=(i+j)/2
}
paste(j,pnorm(j,mean,std),sep=" / ")
```

```
[1] "102.200012207031 / 0.670032330457297"
```

(b) P(X>b)=0.011

b=111.451

```r
# (b) P(X>b)=0.011, Using bisection method
i <- mean-3*std;j <- mean+3*std;mid <- (i+j)/2
while(abs(pnorm(mid,mean,std)-(1-0.011))>0.000001){
  if(pnorm(mid,mean,std)-(1-0.011)>=0){j <- mid}
  if(pnorm(mid,mean,std)-(1-0.011)<0){i <- mid}
  mid=(i+j)/2}
paste(i,1-pnorm(i,mean,std),sep=" / ")
```

```
[1] "111.451416015625 / 0.0110024524392477"
```

(c) P(|X-100|<b)=0.966

b=10.601

```r
# (c) P(|X-100|<b)=0.966, Using bisection method
i <- mean-3*std;j <- mean+3*std;mid <- (i+j)/2
while(abs(1-(1-pnorm(j,mean,std))*2-0.966)>0.000001){
  if(pnorm(mid,mean,std)-(1-(1-0.966)/2)>=0){j <- mid}
  if(pnorm(mid,mean,std)-(1-(1-0.966)/2)<0){i <- mid}
  mid=(i+j)/2
}
paste(j-100,1-(1-pnorm(j,mean,std))*2,sep=" / ")
```

```
[1] "10.6003761291504 / 0.966000298159686"
```

(d) P(X<110)=b

b=0.9772

```r
# (d) P(X<110)=b, Using bisection method
b_d <- pnorm(110,mean,std); b_d
```

```
[1] 0.9772499
```

(e) P(X>95)=b

b=0.8413

```r
# (e) P(X>95)=b
b_e <- 1-pnorm(95,mean,std); b_e
```

```
[1] 0.8413447
```

Question 3

Suppose the amount of sun block lotion in plastic bottles leaving a filling machine has a normal distribution. The bottles are labeled 300 milliliter(ml) but the actual mean is 302 ml and the standard deviation is 2 ml.

(a) What is the probability that an individual bottle will contain less than 299 ml?

로션의 양(ml)에 대한 확률변수 X는 $X \sim N(302, 2^2)$를 따르므로, $P(X < 299) \approx 6.68\%$

```r
pnorm(299,302,2)
```

```
[1] 0.0668072
```

(b) If you pick up 5 bottles and check the amount of sun block lotion in each bottle, what is the probability that all of 5 bottles contain less than 299 ml? (Assume that they are independent.)

각각의 병이 확률변수 $X_i,\ i = 1, 2, 3, 4, 5$라고 하면 $X_i \sim N(302, 2^2)$이며 각 확률변수는 독립이므로 $P(X_i \in k_1,\ X_j \in k_2) = P(X_i \in k_1)P(X_j \in k_2)$가 성립합니다.

따라서, 모든 병이 299ml 미만일 확률 $P(X_i < 299\ for\ all\ i) = \prod_{k=1}^{5} P(X_i < 299) \approx 0.668^5$

```r
pnorm(299,302,2)^5
```

```
[1] 1.330811e-06
```

Question 4

The number of complaints per day, X, received by a cable TV distributor has the probability distribution

x	0	1	2	3
f(x)	0.4	0.3	0.1	0.2

(a) Find the expected value and the standard deviation of the number of complaints per day.

평균은 1.1, 표준편차는 $\sqrt{1.29} \approx 1.136$ 입니다.

```
complaints <- tibble(x=c(0,1,2,3), prob=c(0.4,0.3,0.1,0.2))
mean <- sum(complaints$x*complaints$prob)
vol <- sum(complaints$x^2*complaints$prob)-mean^2
std <- sqrt(vol)
paste(mean,std,sep=" / ")
```

```
[1] "1.1 / 1.13578166916005"
```

(b) What is the approximate probability that the distributor will receive more than 2 complaints in average during 100 days?

먼저, X_i를 i일 뒤에 컴플레인이 들어오는 횟수라고 한다면 X_i는 위 분포를 따르게 될 것입니다. 100일 뒤까지의 컴플레인이 들어오는 횟수를 $X_1,\ X_2\ , ..., X_{100}$이라고 한다면, 각각의 X_i는 독립이고 동일한 분포(iid)를 가지는 표본입니다.

한편, 중심극한정리(Central Limit Theorem)에 따라, 표본의 크기가 충분히 크다면 해당 표본들을 정규분포로 근사시킬 수 있습니다.

$$if\ X_i\ are\ iid,\ \frac{\sum_{k=1}^{n}(X_i-\mu)}{\sqrt{n}\times\sigma} \sim N(0,1)\ for\ large\ n$$

100번의 독립시행 $X_1 \sim X_{100}$은 정규근사하기 충분한 표본이므로 이를 적용하면 다음과 같습니다.

$$By\ CLT,\ \frac{\sum_{k=1}^{100}(X_i-1.1)}{\sqrt{100}\times\sqrt{1.29}} = \frac{\sum_{k=1}^{100}X_i-110}{\sqrt{129}} \sim N(0,1)$$

이제, 100일 동안의 일평균 컴플레인 횟수가 2보다 클 확률은 다음과 같습니다.

$$P[\frac{\sum_{k=1}^{100}X_i}{100} > 2] = P[\frac{\sum_{k=1}^{100}X_i-110}{\sqrt{129}} = Z > \frac{90}{\sqrt{129}}]$$

해당 확률을 R을 통해 구해보면, 0에 가깝습니다.

```
1-pnorm(90/sqrt(129),0,1)
```

```
[1] 1.110223e-15
```

(c) If we observe for five days, what is the probability that the distributor will receive exactly 1 complaint only for two days?

컴플레인을 1개만 받을 확률은 0.3입니다.

하루를 컴플레인을 1개만 받는 경우를 1, 1개가 아닌 경우를 0으로만 나눈다면, 해당 시행은 확률 0.3의 베르누이 시행입니다. 이를 5일 동안 관측한다면 각각은 독립이므로 해당 분포는 $B(5, 0.3)$인 이항분포입니다.

즉, 5일중 2일만 컴플레인을 정확히 1개 받을 확률은 $\binom{5}{2}0.3^2 0.7^3 = 30.87\%$

```
choose(5,2)*0.3^2*0.7^3
```

```
[1] 0.3087
```

(d) If we observe for 100 days, what is the approximate distribution of the number of days that the distributor will not receive any complaint?

하루에 컴플레인이 없을 확률은 0.4로, 컴플레인이 없는 경우를 1, 있는 경우를 0으로 나눈다면 해당 시행은 확률 0.4의 베르누이 시행으로 볼 수 있습니다.

이를 100일간의 표본으로 나누어 시행하면 하루단위 시행은 각각 독립이므로, 이는 확률 0.4 및 시행횟수 100의 이항분포 ($B(100, 0.4)$)를 따르게 됩니다.

한편, 해당 이항분포는 동일한 확률을 가진 100개의 베르누이시행($B_i \sim B(1, 0.4)$)으로 나누어 볼 수 있고, (b)와 동일하게 CLT를 이용한다면 아래와 같이 표현할 수 있습니다.

$$\frac{\sum_{k=1}^{100}(B_i - 0.4)}{\sqrt{100}\sqrt{1 \times 0.4 \times 0.6}} \sim N(0,1),\ Since\ X = \sum_{k=1}^{100} B_i,\ \frac{X - 40}{\sqrt{24}} \sim N(0,1)\ \Rightarrow\ X \sim N(40, 24)$$

즉, 평균이 40이고 분산이 24인 정규분포로 근사할 수 있습니다.

(e) Using the approximate distribution in (d), what is the approximate probability that the number of days that the distributor will not receive any complaint is at most 30 days?

해당 확률은 $P(X \leq 30)\ for\ X \sim B(100, 0.4)$과 같습니다.

위의 정규분포 근사를 활용하면 $X \sim N(40, 24),\ then\ P(X \leq 30) \approx 2.06\%$

```
pnorm(30,40,sqrt(24))
```

```
[1] 0.02061342
```

Question 5

The probability that a voter will believe a rumor about a politician is 0.3.

(a) Find the probability that the first 3 voters don't believe the rumor but the 4th voter believe it.

$0.7 \times 0.7 \times 0.7 \times 0.3 = 0.1029 = 10.29\%$

(b) Find the probability that the exactly one person believe the rumor if 5 voters are told individually

각 유권자는 소문을 개별적으로 확인하였으므로 유권자들이 루머를 믿는 확률변수는 각각 독립입니다.

루머를 믿을 확률은 0.3이므로 루머를 믿는 경우를 1, 안믿는 경우를 0이라고 한다면 이는 확률 0.3의 베르누이 시행입니다. 이를 5번 반복하면 5명의 유권자 중 루머를 믿는 유권자의 수는 $B(5, 0.3)$를 따릅니다.

정확히 한사람만 루머를 믿을 확률은, $\binom{5}{1}0.3^1 0.7^4 \approx 36.02\%$

```
choose(5,1)*0.3*0.7^4
```

```
[1] 0.36015
```

Question 6

Here is the assignment of probabilities that describes the age (in years) and the gender of a randomly selected American College student.

Category	14~18	18~25	25~35	35~
Male	0.01	0.28	0.13	0.04
Femail	0.02	0.3	0.14	0.08

A college student will be selected at random. Let A=[student is Female] and B=[student is at least 25 but less than 35 years old]. Find,

(a) P(A) and P(B)

$P(A) = 0.02 + 0.3 + 0.14 + 0.08 = 0.54$

$P(B) = 0.13 + 0.14 = 0.27$

(b) P(A or B)

$P(A\,or\,B) = P(A \cup B) = P(A) + P(B) - P(A \cap B) = 0.54 + 0.27 - 0.14 = 0.67$

Question 7

Show the following statement:If $X \sim t[k]$, then $X^2 \sim F[1, k]$

$$X \sim t[k] \;\Leftrightarrow\; X = \frac{Z}{\sqrt{(V/k)}}\; for\; Z \sim N(0,1),\; V \sim \chi^2(k)$$

$$\Leftrightarrow X^2 = \frac{Z^2}{V/k} = \frac{U/1}{V/k}\; for\; U = Z^2 \sim \chi^2(1)$$

$$\Leftrightarrow F = X^2 = \frac{U/1}{V/k} \sim F(1,k)\; for\; U \sim \chi^2(1),\; V \sim \chi^2(k) \quad \square$$

경영통계분석 과제3

Question 1

if $X_1,\ X_2,\ \ldots,\ X_n$ are i.i.d random variables with mean μ and variance σ^2, calculate the covariance $\bar{X}$ and $X_i - \bar{X}$ for any $i = 1, \ldots, n$.

Answer

먼저, 임의의 확률변수 U, V에 대하여 $Cov(U,V) = E[(U - E[U])(V - E[V])] = E[UV] - E[U]E[V]$ 입니다.

$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \Rightarrow E[\bar{X}] = \mu,\ E[X_i - \bar{X}] = 0\ for\ all\ i$이고, 각각의 X_i는 독립이므로 $E[X_iX_j] = E[X_i]E[X_j] = \mu^2\ for\ i \neq j$이며, $E[X_i^2] = Var[X_i] + E[X_i]^2 = \sigma^2 + \mu^2$입니다.

이를 이용하면 $E[\bar{X}^2] = E[\frac{\sum_i X_i^2 + \sum_{i,j}^{i\neq j} X_iX_j}{n^2}] = \frac{n(\sigma^2+\mu^2)+n(n-1)\mu^2}{n^2} = \frac{n\sigma^2+n^2\mu^2}{n^2} = \frac{\sigma^2}{n} + \mu^2$입니다.

이제 공분산을 구해보겠습니다.

$$Cov[\bar{X}, X_i - \bar{X}] = E[\bar{X}(X_i - \bar{X})] - \mu \times 0 = E[\bar{X}X_i] - E[\bar{X}^2]$$

$$= \frac{E[X_i(X_1 + \ldots + X_n)]}{n} - \frac{\sigma^2}{n} - \mu^2 = \frac{E[X_i^2] + (n-1)\mu^2 - \sigma^2 - n\mu^2}{n}$$

$$= \frac{\sigma^2 + \mu^2 - \mu^2 - \sigma^2}{n} = 0$$

즉, 공분산은 0입니다.

> **i** 간결한 풀이방법
>
> $Cov[X_i, X_j] = 0\ for\ i \neq j,\ Cov[X_i, X_i] = Var[X_i] = \sigma^2,\ Var[\bar{X}] = \frac{\sigma^2}{n}$이므로,
> $Cov[\bar{X}, X_i - \bar{X}] = Cov[\bar{X}, X_i] - Cov[\bar{X}, \bar{X}] = \frac{Var[X_i]}{n} - Var[\bar{X}] = 0$

Question 2

A commercial for a manufacturer of household appliance claims that 3% of all its product require a service call in the first year. A consumer protection association wants to check the claim by surveying 400 households that recently purchased one of the company's appliances. What is the probability that more than 5% require a service call within the first year?

(a) Calculate the probability using the approximate normal approach.

(b) Calculate the probability using the binomial distribution.

Answer

각각의 가구를 조사하는 일은 독립이며, 클레임이 있는 경우를 1, 없는 경우를 0으로 나누면 이는 $Ber(0.03)$의 베르누이 시행입니다.

이를 400번 반복하므로 분포는 $B(400, 0.03)$의 이항분포이며, 정규분포로 근사하면 아래와 같습니다.

$$By\ CLT,\ \frac{\sum_{i=1}^{400}(X_i - 0.03)}{\sqrt{400}\sqrt{0.03 \times 0.97}} \sim N(0,1) \Rightarrow X \sim N(12, 11.64)\ for\ X \sim B(400, 0.03)$$

클레임이 있는 가구가 5%를 초과할 확률은 20가구를 초과할 확률 $P(X > 20)$이므로, **(a) 0.95% 및 (b) 1.05%** 입니다.

정규분포 근사는 n이 매우 크고 p가 0.5와 인접할 때 정확도가 높은데, **n은 충분히 크지만 p가 매우 작아 다소 오차가 발생**하는 것으로 추정됩니다.

```
# (a) normal dist.
1-pnorm(20,12,sqrt(11.64))
```

```
[1] 0.009517584
```

```
# (b) binomial dist.
prob=0
for(i in 0:20){
    prob=prob+choose(400,i)*0.03^i*0.97^(400-i)
}
1-prob
```

```
[1] 0.01045374
```

Question 3

Suppose that X has normal dist. with $\mu = 10,\ \sigma = 2$

(a) $P(6 < X < 14)$: 0.9545

```
mean=10; std=2
pnorm(14,mean,std)-pnorm(6,mean,std)
```

```
[1] 0.9544997
```

> Note
>
> 이는 $P(X \in (\mu - 2\sigma, \mu + 2\sigma))$이므로, 95.45%임이 잘 알려져있습니다.

(b) $P(X \leq c) = 0.95$: c=13.29

```
# (a) P(X<=c)=0.95, Using bisection method
i <- mean-3*std
j <- mean+3*std
mid <- (i+j)/2
while(abs(pnorm(mid,mean,std)-0.95)>0.000001){
  if(pnorm(mid,mean,std)-0.95>=0){j <- mid}
  if(pnorm(mid,mean,std)-0.95<0){i <- mid}
  mid=(i+j)/2
}
paste(j,pnorm(j,mean,std),sep=" / ")
```

```
[1] "13.2897644042969 / 0.950002947051978"
```

(c) for sample $X_1, ..., X_4$, $P(\bar{X} \leq 12)$: 0.9772

$E(\bar{X}) = 10,\ Var(\bar{X}) = \frac{4}{4} = 1$이고, 각각의 X_i는 i.i.d 정규분포이므로 $\bar{X} \sim N(10, 1)$입니다.

```
pnorm(12,10,1)
```

```
[1] 0.9772499
```

Question 4

Bits are sent over a communications channel in packets of 160. If the probability of a bit being corrupted (one error) over this channel is 0.2 and such errors are independent. Let X denotes the number of bits that are corrupted over this channels

(a) What is the distribution of X? Can it be approximated as normal distribution?
(b) Approximately, what is the probability that more than 50 bits in a packet are corrupted?

Answer

한번의 전송에 에러가 발생하는 경우를 1, 아닌 경우를 0으로 나누면 이는 확률 0.2의 베르누이 시행입니다. 이를 160번 반복하고 에러가 발생하는 횟수인 X는 $B(160, 0.2)$ **이항분포**를 따를 것 입니다.

해당 **이항분포는 시행횟수가 충분히 크지만($>$30), 확률이 0.2로 다소 낮아 정규분포 $N(32, 25.6)$를 사용할 때 근소한 오차**가 발생할 수 있습니다.

$P(X > 30)$을 구해보면,정규분포는 34.63%, 이항분포는 39.05%으로 오차가 다소 존재하므로 이에 유의해야합니다.

```
prob=0
for(i in 0:30){
    prob=prob+choose(160,i)*0.2^i*0.8^(160-i)
}
paste(pnorm(30,32,sqrt(25.6)),prob,sep=" / ")
```

```
[1] "0.34631639202098 / 0.390465138866041"
```

에러가 50번을 초과하여 발생할 확률은 $P(X > 50)$이므로, **정규분포를 이용한 확률은 0.0187%**입니다. 한편, 이항분포를 이용한 확률은 0.0265%입니다.

```
prob=0
for(i in 0:50){
    prob=prob+choose(160,i)*0.2^i*0.8^(160-i)
}
paste(1-pnorm(50,32,sqrt(25.6)),1-prob,sep=" / ")
```

```
[1] "0.000187156123554355 / 0.000264520708086802"
```

Question 5

A large population is described by the probability distribution, Let X_1 and X_2 be a random sample of size2 from the distribution.

X	f(x)
0	0.1
1	0.2
2	0.7

(a) Determine the sampling distribution of $max(X_1, X_2)$

(b) Determine the sampling distribution of $X_1 + X_2$

Answer

먼저 X_1, X_2는 모집단으로부터 추출한 표본으로 i.i.d를 만족할 것 입니다. 따라서 $f(X_1 = x_1, X_2 = x2) = f(x_1)f(x_2)$로 산출할 수 있습니다.

$x_m \in \{0, 1, 2\}$ for $x_m \sim max(X_1, X_2)$이고, 순서쌍 (x_1, x_2)로 경우의 수를 나타내면 $(0, 0) \Rightarrow x_m = 0$ 및 $(0, 1)$, $(1, 0)$, $(1, 1) \Rightarrow x_m = 1$ 및 $(2, 0)$, $(2, 1)$, $(2, 2)$, $(0, 2)$, $(1, 2) \Rightarrow x_m = 2$입니다. 따라서, $max(X_1, X_2)$의 분포는,

x_m	$max(X_1, X_2)$
0	0.01
1	0.02+0.02+0.04=0.08
2	0.7+0.21=0.91

위와 유사한 방법으로 $X_1 + X_2$의 경우의 수를 나타내면 다음과 같습니다.

$x_1 \in X_1$	$x_2 \in X_2$	$\Rightarrow$	$x_+ \in X_1 + X_2$
0	0		0
0	1		1
1	0		1
1	1		2
2	0		2
0	2		2

$x_1 \in X_1$	$x_2 \in X_2$	$\Rightarrow$	$x_+ \in X_1 + X_2$
1	2		3
2	1		3
2	2		4

따라서, $X_1 + X_2$의 분포는,

x_+	$X_1 + X_2$
0	0.01
1	0.02+0.02=0.04
2	0.04+0.07+0.07=0.18
3	0.14+0.14=0.28
4	0.49

경영통계분석 과제4

Problem 1.

Assume that the standard deviation of the heights of five-year-old boys is 3.5 inches. How many five-year-old need to be sampled if we want to be 90% sure that the population mean height is estimated within .5 inch?

Answer

크기가 N인 표본을 임의추출하고, 충분히 크다고 가정하겠습니다.

모표준편차가 3.5(inch)이므로, 크기가 N인 표본평균 $\bar{X}$은 중심극한정리(CLT)에 의해 정규분포 $N(\mu, \frac{3.5^2}{N})$을 따르게 될 것입니다.

이를 이용하여 신뢰수준이 90%인 모평균의 신뢰구간을 양편으로 추정해보겠습니다.

$$P(z_{0.95} = -z_{0.05} < \frac{\bar{X} - \mu}{3.5/\sqrt{N}} < z_{0.05}) = 0.9 \Rightarrow P(\bar{X} - z_{0.05}\frac{3.5}{\sqrt{N}} < \mu < \bar{X} + z_{0.05}\frac{3.5}{\sqrt{N}}) = 0.9$$

여기서, 모평균을 0.5inch 이내로 추정하는 것의 의미는, 표본오차가 0.5이내, 즉 $z_{0.05}\frac{3.5}{\sqrt{N}} < 0.5$라는 뜻입니다.

$z_{0.05} \approx 1.65$이므로, 이를 정리하면 $1.65 \times 7 < \sqrt{N} \Rightarrow 133.4025 < N$

즉, 표본의 수가 134명 이상이라면 90% 신뢰구간을 0.5inch 오차 이내로 추정할 수 있습니다.

또한, 이는 충분히 큰 표본이므로 앞서 CLT를 이용하는 것에 위배되지 않습니다.

Problem 2.

An employee of an on-campus copy center wants to determine the mean number of copies before a cartridge needs to be replaced. She records the life length in thousands of copies for 43 cartridges and obtains $n = 43, \bar{x} = 8.12, s = 1.78$ thousand copies Obtain a 90% confidence interval for the population mean, μ , number of copies in thousands before a cartridge should be replaced.

Answer

먼저, 43개의 샘플이 충분히 크다고 생각하고 시작하겠습니다.

CLT에 따라 표본평균 $\bar{X} \sim N(\mu, \sigma/\sqrt{43})$이며, 표본분산 $\frac{42\times s^2}{\sigma^2} \sim \chi^2(42)$입니다.

이를 정리하면 $\frac{\bar{X}-\mu}{s/\sqrt{43}} \sim t(42)$가 되는데, 표본의 크기가 충분히 크므로 $t(n-1) \to N(0,1)$로 근사할 수 있습니다.

이제, 이를 이용하여 모평균의 90% 신뢰구간을 추정해보겠습니다.

$$P(\mu \in (\bar{X} \pm z_{0.05}\frac{s}{\sqrt{43}})) = 0.90 \Rightarrow P(\mu \in (7.672, 8.568)) = 0.90$$

즉, 신뢰구간은 (7.672, 8.568)입니다.

Problem 3.

Data on the average weekly earnings were obtained from a survey of 50 nonsupervisory production workers in the mining industry. The sample mean and standard deviation were found to be \$630 and \$35, respectively.

(a) Estimate the true mean weekly earnings and determine the 95% error margin.
(b) Construct a 95% confidence interval for the true mean weekly earnings

Answer

먼저, 표본평균 $\bar{X}$의 성질에 따라 $E(\bar{X}) = \mu,\ \bar{X} \to^p \mu$이므로 $\bar{X}$는 모평균에 대한 불편(unbiased) 및 일치(consistent) 추정량입니다.

따라서 **모평균의 estimator로 표본평균을 선정하고, 모평균을 \$630으로 추정**하겠습니다.

한편, 표본의 수가 50으로 충분히 크므로 앞서 이용한 논리를 그대로 준용하여 $\frac{\bar{X}-\mu}{s/\sqrt{50}} \sim t(49) \approx N(0,1)$라고 할 수 있습니다.

따라서 **95%의 error margin은 $z_{0.05}\frac{s}{\sqrt{50}} = 8.167$입니다.**

마지막으로, 95%의 신뢰수준으로 추정한 모평균의 신뢰구간은 아래와 같습니다.

$$P(\bar{X} - z_{0.025}\frac{s}{\sqrt{50}} < \mu < \bar{X} - z_{0.025}\frac{s}{\sqrt{50}}) \Rightarrow Confidence\ interval\ is\ (620.299, 639.702)$$

경영통계분석 과제5

Problem 1.

1. In a given situation, suppose H0 was not rejected at =.02. Answer the following questions as "yes", "no", or "can't tell" as the case may be.

(a) Would H0 also be retained at =.01?
(b) Would H0 also be retained at =.05?
(c) Is the p-value smaller than .02?

Answer

H0가 $\alpha = 0.02$에서 기각되지 않았다는 것은, 주장하고자 하는 대립가설에 대한 통계량의 p-value가 0.02보다 크다는 의미입니다.

따라서, 우리가 알 수 있는 것은 0.02보다 낮은 신뢰수준에서 H0를 기각할 수 없다는 것 뿐입니다.

(a) Yes
(b) Can't tell
(c) No

Problem 2.

2. A company's mixed nuts are sold in cans and the label says that 25% of the contents is cashews. Suspecting that this might be an overstatement, an inspector takes a random sample of 35 cans and measures the percent weight of cashews [i.e. 100(weight of cashews/weight of all nuts)] in each can. The mean and standard deviation of these measurements are found to be 23.5 and 3.1, respectively. Do these results constitute strong evidence in support of the inspector's belief?

(a) Identify H0 and H1.

(b) Carry out the hypothesis test at the 5% level of significance using the p-value (use normal distribution).

Answer

H0 : 캐슈넛의 포함비율은 25%이다. ($\mu = 25$)

H1 : 캐슈넛의 포함비율은 25%보다 낮아 표기가 과장되었다. ($\mu < 25$)

정규분포 가정 하에서 표본의 크기가 충분히 크므로 z검정을 수행할 수 있으며, z통계량 z_0 및 p-value는 아래와 같이 산출할 수 있습니다.

$$z_0 = \frac{\bar{X} - \mu}{s/\sqrt{35}} = \frac{-1.5}{3.1/\sqrt{35}} = -2.86,\ P(Z < z_0 = -2.86) = 0.0021$$

p-value=0.0021〈0.05이므로, 5% 신뢰수준에서 귀무가설을 기각할 수 있습니다.

이는 캐슈넛의 포함비율이 25% 보다 낮다는 조사자의 추정을 지지하는 결과입니다.

심지어, p-value가 매우 낮아 1% 신뢰수준이라고 하더라도 귀무가설을 기각할 수 있습니다.

결론적으로, 캐슈넛의 포함비율이 25% 보다 낮다는 조사자의 추정은 합리적으로 보입니다.

Problem 3

3. From extensive records, it is known that the duration of treating a disease by a standard therapy has a mean of 15 days. It is claimed that a new therapy can reduce the treatment time. To test this claim, the new therapy is tried on 70 patients, and from the data of their homes to recovery, the sample mean and standard deviation are found to be 14.6 and 3.0 days, respectively.

(a) Perform the hypothesis test using a 2.5% level of significance, based on the rejection region.

(b) Calculate the p-value and interpret the result.

(c) State any assumptions you make about the populations.

Answer

(a)

표본의 수는 70이며(충분히 큼), 표본의 평균은 14.6 및 표준편차는 3입니다.

새로운 치료법의 치료기간이 평균적으로 15일보다 낮다면, 새로운 치료법이 효율적이라고 할 수 있습니다.

따라서, 새로운 치료법의 평균치료기간(모평균)이 15일보다 낮다는 가설을 주장하기 위해 표준정규분포를 이용한 z-검정을 실시하겠습니다.

H0 : $\mu = 15$

H1 : $\mu < 15$

단측검정이며, 신뢰수준 2.5%에 대한 임계값은 $P(Z < x) \approx 0.025,\ x = -1.96$ 입니다.

z검정에 이용하는 통계량은 $z_0 = \frac{14.6-15}{3/\sqrt{(70)}} = -1.1155 > -1.96 = z_{0.025}$ 입니다.

따라서, 귀무가설을 기각할 수 없으며 동 가정 하에서 새로운 치료법이 개선되었다는 주장을 지지하지 않습니다.

(b)

p-value는 $P(Z < z_0 = -1.1155) \approx 0.132$ 입니다.

이 뜻은 실제 새로운 치료방법의 치료기간이 15일보다 짧지 않을 확률이 약 13.2%라는 의미입니다.

만약 신뢰수준을 15%로 상향하여 동 검정을 수행한다면 귀무가설을 기각할 수 있겠으나, 의사결정이 잘못되었을 리스크가 13.2%로 존재한다는 의미입니다. (1종오류)

(c)

저는 z검정을 사용하였습니다.

표본의 크기가 충분히 크므로 CLT에 따라 표본평균이 정규분포를 따르게 되며, 표본표준편차는 모표준편차의 불편 및 일치추정량이므로 $s \to \sigma\ for\ large\ sample$인 점을 이용한 것 입니다.

따라서, **1. 임의추출(random samples)** 이외의 별도의 가정은 없습니다.

Problem 4.

4. In a study of interspousal aggression and its possible effect on child behavior, the behavior problem checklist scores were recorded for 47 children whose parents were classified as aggressive. The sample mean and standard deviation were 7.92 and 3.45, respectively. For a sample of 38 children whose parents were classified as nonaggressive, the mean and standard deviation of the BPC scores were 5.80 and 2.87, respectively.

(a) Do these observations substantiate the conjecture that the children of aggressive families have a higher mean BPC than those of nonaggressive families? (Answer by calculating the p-value.)

(b) State any assumptions you make about the populations.

Answer

(a)

먼저, 각 표본의 수가 30보다 크므로 z검정을 수행하도록 하겠습니다.

부모가 공격적이라고 응답한 아이 집단을 A, 공격적이지 않다고 응답한 아이 집단을 B라고 하고 가설을 수립하면,

H0 : $\mu_A = \mu_B \ \Rightarrow \ \mu_A - \mu_B = 0$

H1 : $\mu_A > \mu_B \ \Rightarrow \ \mu_A - \mu_B > 0$

한편, 각각 독립인 표본에 대하여 두 집단의 평균의 차이를 검정할 때 필요한 표본평균의 분포는 아래와 같습니다.

$$\bar{X^*} = \bar{X_A} - \bar{X_B} \sim N(\mu_A - \mu_B, \frac{\sigma_A^2}{47} + \frac{\sigma_B^2}{38} \approx \frac{s_A^2}{47} + \frac{s_B^2}{38})$$

따라서, z통계량은 $z_0 = \frac{\mu_A - \mu_B - 0}{\frac{s_A}{\sqrt{47}} + \frac{s_B}{\sqrt{38}}} = \frac{7.92 - 5.80}{3.45/\sqrt{47} + 2.87/\sqrt{38}} = 2.188$ 이며,

P-value는 $P(Z > 2.188) = P(Z < -2.188) = 0.0143$ 입니다.

따라서, 신뢰수준 약 1.5%에서 귀무가설을 기각할 수 있으며 공격적으로 응답한 아이 집단의 BPC 점수가 평균적으로 높다는 주장을 강하게 지지한다고 할 수 있습니다.

(b)

위와 마찬가지로 표본의 크기가 충분하므로 CLT에 따라 표본평균이 정규분포를 따르게 되며, 표본표준편차는 모표준편차로 수렴하므로 이를 대용치로 이용한 것 입니다.

다만, 두 집단의 평균차이를 검정한 것이므로 두 모집단이 서로 독립이여야 합니다.

따라서, **1. 임의추출(random samples)**, **2. 두 모집단이 독립**이라고 가정하였습니다.

Problem 5.

5. The data on the weight of male and female wolves are as follows:

구 분											
Female	57	84	90	71	71	77	68	73			
Male	71	93	101	84	88	117	86	86	93	86	106

(a) Test the null hypothesis that the mean weights of males and females are equal versus a two-sided alternative. Take $\alpha = 0.05$

(b) State any assumptions you make about the populations for (a).

(c) Test the null hypothesis that the variances of weights of males and females are equal. Take $\alpha = 0.05$

(d) State any assumptions you make about the populations for (c).

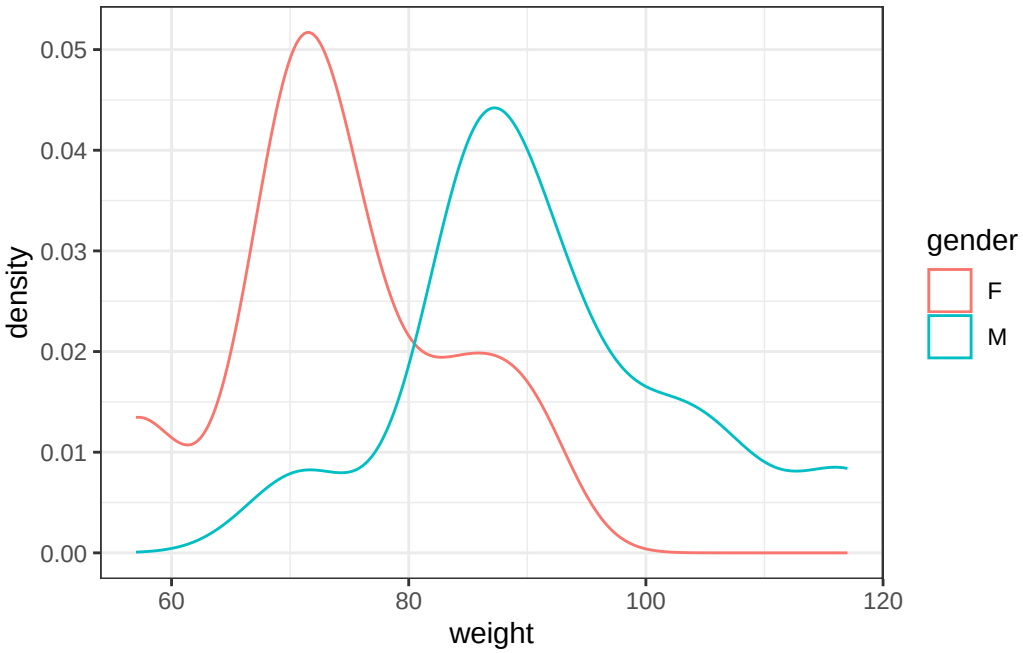
Answer

(a)

먼저, 각 샘플의 모양을 살펴보면 아래와 같습니다.

```
library(tidyverse)
library(patchwork)
weight <- tibble(gender=c(rep("F",8),rep("M",11)),
                 weight=c(57,84,90,71,71,77,68,73,
                          71,93,101,84,88,117,86,86,93,86,106))
```

```
ggplot(weight,aes(x=weight,colour=gender))+
  geom_density() +
  theme_bw()
```



여성, 남성별로 표본의 수, 표본평균, 표본표준편차는 아래와 같습니다.

```
weight %>%
  group_by(gender) %>%
  mutate(cnt=1) %>%
  summarise(sample=sum(cnt),mean=mean(weight),vol=sd(weight))
```

```
# A tibble: 2 x 4
  gender sample  mean   vol
  <chr>   <dbl> <dbl> <dbl>
1 F           8  73.9  10.1
2 M          11  91.9  12.4
```

각 표본이 대략적으로 bell-shape를 형성하고 있으므로, 정규성 가정에 큰 무리는 없어보입니다.

따라서, 평균의 차이가 있는지 알아보기 위해서 t검정을 수행하겠습니다.

H0 : $\mu_F = \mu_M$, H1 : $\mu_F \neq \mu_M$

```
t.test(weight~gender,data=weight,alternative="two.sided",paired=F)
```

```
    Welch Two Sample t-test

data:  weight by gender
t = -3.4971, df = 16.715, p-value = 0.002821
alternative hypothesis: true difference in means between group F and group M is not equal to 0
95 percent confidence interval:
 -28.928413  -7.139769
sample estimates:
mean in group F mean in group M
       73.87500        91.90909
```

p-value는 0.0028로 신뢰수준 5% 하에서 귀무가설을 기각할 수 있습니다.

(b)

모집단의 분포를 모르고 표본의 크기도 충분하지 않으므로, 표본평균의 정규성이 보장되지 않습니다.

또한, 두 집단의 평균 차이를 검정하기 위해서는 독립일 필요가 있으며, t분포 유도를 위해 두 집단의 분산이 동일해야합니다.

따라서, **1. 임의추출(random samples)**, **2. 모집단의 정규성**, **3. 두 모집단은 독립**, **4. 두 모집단은 등분산**이라고 가정하였습니다.

(c)

두 집단의 분산이 같은지에 대해 검정하기 위해 분산차이에 대한 F검정을 수행하겠습니다.

H0 : $\frac{\sigma_M}{\sigma_F} = 1$, H1 : $\frac{\sigma_M}{\sigma_F} \neq 1$

```
var.test(weight~gender,data=weight)
```

```
    F test to compare two variances

data:  weight by gender
F = 0.66063, num df = 7, denom df = 10, p-value = 0.5981
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.1672544 3.1453141
sample estimates:
ratio of variances
         0.6606253
```

P-value는 0.598로 귀무가설을 기각하지 못하였습니다. 이는 두 집단의 분산이 같다는 것을 지지하고 있습니다.

(d)

표본의 크기가 충분하지 않으므로 정규성이 담보되지 않아 F분포 및 카이제곱분포 유도를 위해 정규성 가정이 필요합니다.

또한, F분포 유도를 위해 두 집단이 독립일 필요가 있습니다.

따라서, **1. 임의추출(random samples)**, **2. 모집단의 정규성**, **3. 두 모집단은 독립**이라고 가정하였습니다.

경영통계분석 과제6

Problem 1

1. Given the five pairs of (x, y) values,

```
rm(list=ls())
library(tidyverse)

x=c(0,1,6,3,5)
y=c(4,3,0,2,1)
```

(a)

Find the least squares estimates of slope and intercept, determine the best fitting straight line (use R program).

Answer

R을 이용한 회귀분석 결과, 회귀식은 $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x = 3.84615 - 0.61538x$ **입니다.**

```
lm(y~x) %>% summary()
```

```
Call:
lm(formula = y ~ x)

Residuals:
         1          2          3          4          5
 1.538e-01 -2.308e-01 -1.538e-01 -1.880e-16  2.308e-01
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.84615    0.16736   22.98 0.000180 ***
x           -0.61538    0.04441  -13.86 0.000814 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2265 on 3 degrees of freedom
Multiple R-squared:  0.9846,    Adjusted R-squared:  0.9795
F-statistic:   192 on 1 and 3 DF,  p-value: 0.0008136
```

(b)

Test H0: 1 = 0 versus H1: 1 0 with = 0.05.

Answer

상기 회귀분석 결과에서 β_1에 대한 t통계량은 -13.86, 이에 이용한 p-value는 0.000814입니다.

따라서 95% 신뢰수준에서 가설검정시 **H0가 reject되며 H1이 채택**됩니다.

또한, 신뢰수준을 약 99.91%까지 높게 설정하여도 가설검정 결과는 동일합니다.

(c)

Obtain a 95% confidence interval for the fitted value given x=1.

Answer

먼저, 회귀분석을 통해 추정된 회귀계수 $\hat{\beta}_0,\ \hat{\beta}_1$는 모계수 $\beta_0,\ \beta_1$의 불편추정량이며, 회귀계수의 분포는 아래와 같습니다.

$$\hat{\beta}_1 = \frac{S_{xy}}{S_{xx}} \sim N(\beta_1, \frac{\sigma^2}{S_{xx}}),\ \ \hat{\beta}_0 = \bar{y} - \hat{\beta}_1\bar{x} \sim N(\beta_0, (\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}})\sigma^2)$$

주어진 x_0에 대한 $y_0 = \beta_0 + \beta_1 x_0$의 조건부기대값은 $E(y_0|x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$이며, 회귀계수의 분포를 이용해 추정한 $E(y_0|x_0)$의 분포는 다음과 같습니다.

$$E(y_0|x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 \sim N(\beta_0 + \beta_1 x_0, (\frac{1}{n} + \frac{(x_0 - \bar{x})^2}{S_{xx}})\sigma^2)$$

이제, 오차의 표준편차 σ 대신에 $MSE = \hat{\sigma}$를 이용하여 t분포를 통해 신뢰구간을 추정하면,

$$CI_{95\%} \ = \ ((\hat{\beta}_0 + \hat{\beta}_1 x_0) \pm t_{0.025,3} \sqrt{\hat{\sigma}^2 (\frac{1}{n} + \frac{(x_0 - \bar{x})^2}{S_{xx}})})$$

이제 $S_{xx} = \sum (x_i - \bar{x})^2 = 3^2 + 2^2 + 3^2 + 2^2 = 26$ 및 $\hat{\beta}_0 = 3.846,\ \hat{\beta}_1 = -0.615,\ \hat{\sigma} = 0.227,\ n = 5,\ x_0 = 1$를 적용하면 신뢰구간은 $(2.802,\ \ 3.659)$ 입니다.

(d)

Calculate R-squared.

Answer

결정계수는 $\frac{SSR}{SST}$ 이며, 이는 **위 회귀분석의 결과에서 0.9846으로 산출**되었습니다.

Problem 2

2. This is the R output of a linear regression model:

```
Call:
lm(formula = y ~ x)
Residuals:
    Min      1Q  Median      3Q     Max
-2.7252 -1.2076 -0.3564  1.2183  2.8928
Coefficients:
            Estimate  Std. Error  t value  Pr(>|t|)
(Intercept) 23.6409   16.4171     1.440    0.1754
x           0.6527    0.2416      2.702    0.0192 *
---
Signif. codes: 0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1
Residual standard error: 1.779 on 12 degrees of freedom
Multiple R-squared: 0.3782, Adjusted R-squared: 0.3264
F-statistic: 7.3 on 1 and 12 DF, p-value: 0.01924
```

(a)

What are the estimates of β_0 and β_1?

Answer

각 회귀계수의 추정량은 $\hat{\beta}_0 = 23.6409$, $\hat{\beta}_1 = 0.6527$로 위 표에서 확인할 수 있습니다.

(b)

Using = 0.05 to test on H0 : 1 = 0 vs H1 : 1 = 0. What is the conclusion?

Answer

$\hat{\beta}_1$ 에 대한 p-value는 위 표에서 0.0192로 산출되었습니다.

따라서, 95% 신뢰수준에서는 H0를 기각하고 H1을 채택할 수 있습니다. 즉, 두 변수간에는 선형관계가 있다고 추론할 수 있습니다.

그러나, 99% 신뢰수준에서는 H0를 기각할 수 없습니다.

(c)

Compute the sum of squares error and sum of squares total for this model.

Answer

위 표에서 $1.779 = \sqrt{MSE} = \sqrt{\frac{SSE}{12}}$ 이므로, $SSE = 37.978$ 입니다.

또한, $0.3782 = R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$ 이므로, $SST = 61.078$ 입니다.

(d)

For x = 70, use the model to predict y and construct prediction interval with 95% confidence level.

Answer

$x_1 = 70$ 에 대한 y_1 의 95% 예측구간을 산출하도록 하겠습니다.

앞서 1-(c)에서, 주어진 x_0 에 대하여 $y_0 = \beta_0 + \beta_1 x_0$ 이며, 이에 대한 조건부기대값의 분포를 통해 신뢰구간을 산출하였습니다.

그러나 예측구간의 경우, 새로운 observation x_1 에 대하여 새로 발생하는 오차 $\epsilon_1 \sim N(0, \sigma^2)$ 을 고려해야 합니다.

즉, $y_1 = \beta_0 + \beta_1 x_1 + \epsilon_1$ 이므로 y_1 의 추정량 $\hat{y_1}$ 의 분산은 $Var(\hat{y_1}) = Var(\hat{\beta}_0 + \hat{\beta}_1 x_1) + Var(\epsilon_1)$ 이 됩니다. 따라서 분포는 아래와 같습니다.

$$\hat{y_1} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \epsilon_1 \sim N(\beta_0 + \beta_1 x_0, (\frac{1}{n} + \frac{(x_0 - \bar{x})^2}{S_{xx}})\sigma^2 + \sigma^2)$$

이제, 이를 이용한 95% 예측구간은 다음과 같습니다.

$$CI_{95\%} \;=\; ((\hat{\beta}_0 + \hat{\beta}_1 x_1) \pm t_{0.025,12}\sqrt{\hat{\sigma}^2 + \hat{\sigma}^2(\frac{1}{n} + \frac{(x_1 - \bar{x})^2}{S_{xx}})})$$

먼저, 파라미터 $\hat{\beta}_0 = 23.6409,\ \hat{\beta}_1 = 0.6527,\ n = 14,\ x_1 = 70$으로 주어져 있습니다.

또한 1-(c)의 회귀계수의 분포에 따라 $Var(\hat{\beta}_0) = (\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}})\sigma^2 =,\ Var(\hat{\beta}_1) = \frac{\sigma^2}{S_{xx}}$ 입니다.

즉, 표준오차를 통해 $S.E(\hat{\beta}_0) = 16.4171 = \sqrt{(\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}})MSE},\quad S.E(\hat{\beta}_1) = 0.2416 = \sqrt{\frac{MSE}{S_{xx}}}$임을 알 수 있고, 주어진 파라미터를 통해 $\bar{x} = 67.92,\ S_{xx} = 54.22$를 도출할 수 있습니다.

이제, **각 파라미터를 대입한** $x_1 = 70$ 일 때의 y_1 의 95% 예측구간은 $(65.17,\ 73.49)$입니다.