# Python3 codes for thesis "How bond risk affects risk parity portfolios"

- most packages used could be easily installed through pip
- package tool9 could be found at https://github.com/nightttt7/tool9
- PredictorData2018.xlsx is from http://www.hec.unil.ch/agoyal/docs/PredictorData2019.xlsx
- "dataset3.csv" is integrated daily log returns

## Daily Data Integration

```
# %%
import pandas as pd
import numpy as np
from pathlib import Path
import matplotlib
import matplotlib.pyplot as plt
import datetime
import missingno as msno

# the relative path of folder
data_path = Path('raw-data/')
output_path = Path('output/')

# %%
# monthly treasury return (to compare)
data_tbond_return = pd.read_excel(data_path / "PredictorData2018.xlsx",
                                  sheet_name="Monthly")
data_tbond_return['Date'] = pd.to_datetime(data_tbond_return['yyyymm'],
                                           format='%Y%m', errors='coerce')
data_tbond_return.set_index('Date', inplace=True)
monthly_return = data_tbond_return[['ltr']].copy(deep=True)

# %%
# daily treasury return
raw_data = pd.read_csv(data_path / 'CRSP_treasury_all.csv')
raw_data_2020 = pd.read_csv(data_path / 'CRSP_treasury_all_2020.csv')
raw_data = pd.concat([raw_data, raw_data_2020], axis=0)

# ===============|==============|====================
# Variable  Name | Data Type    | Variable Description
# ---------------|--------------|--------------------
# KYTREASNO      | NUM          | Treasury Record Identifier
# ITYPE          | CHAR         | Type of Issue
# TDATDT         | DATE         | Date Dated by Treasury
# TMATDT         | DATE         | Maturity Date at Time of Issue
# TCOUPRT        | NUM          | Coupon Rate
# IWHY           | CHAR         | Reason for End of Data
# IYMCN          | DATE         | Year and Month of First Call Notice
# ---------------|--------------|--------------------
```

```
# CALDT          | DATE         | Quotation date
# TDACCINT       | NUM          | Daily Series of Total Accrued Interest
# TDNOMPRC       | NUM          | Daily Nominal Price
# TDPDINT        | NUM          | Daily Series of Paid Interest
# TDYLD          | NUM          | Daily Series of Promised Daily Yield
# TDRETNUA       | NUM          | Daily Unadjusted Return
# ===============|==============|====================

# ITYPE
# TYPE OF ISSUE
# 1 = NONCALLABLE BOND
# 2 = NONCALLABLE NOTE
# 3 = CERTIFICATE OF INDEBTEDNESS
# 4 = TREASURY BILL
# 5 = CALLABLE BOND
# 6 = CALLABLE NOTE
# 7 = TAX ANTICIPATION CERTIFICATE OF INDEBTEDNESS
# 8 = TAX ANTICIPATION BILL
# 9 = OTHER, FLAGS ISSUES WITH UNUSUAL PROVISIONS
# 10= RESERVED FOR FUTURE USE
# 11= INFLATION-ADJUSTED BONDS
# 12= INFLATION-ADJUSTED NOTES

# IWHY
# REASON FOR END OF DATA
# 0 = STILL QUOTED ON LAST UPDATE OF FILE
# 1 = MATURED
# 2 = CALLED FOR REDEMPTION
# 3 = ALL EXCHANGED
# 4 = SOURCES NO LONGER QUOTE ISSUE (not appear here)

# get the data **we need** from raw_data
data = raw_data[['KYTREASNO', 'ITYPE', 'TDATDT', 'TMATDT', 'TCOUPRT',
                 'IWHY', 'IYMCN', 'CALDT', 'TDACCINT', 'TDNOMPRC', 'TDPDINT',
                 'TDYLD', 'TDRETNUA']]

# format date string to datatime
data['TDATDT'] = pd.to_datetime(data['TDATDT'])
data['TMATDT'] = pd.to_datetime(data['TMATDT'])
data['CALDT'] = pd.to_datetime(data['CALDT'])

data['is_called'] = data['IYMCN'].notnull()

data.loc[data['is_called'], 'IYMCN'] = \
    data.loc[data['is_called'], 'IYMCN'].apply(lambda x: str(x)[:-2])
data['IYMCN'] = pd.to_datetime(data['IYMCN'], format='%Y%m')

# set CALDT as index (and keep CALDT column)
data.set_index('CALDT', drop=False, inplace=True)

# type (from ITYPE)
data['type'] = ''
# bills
data.loc[data['ITYPE'].isin([4]), 'type'] = 'bill'
```

```
# noncallable notes (there's no callable notes)
data.loc[data['ITYPE'].isin([2]), 'type'] = 'note'
# inflation-adjusted notes
data.loc[data['ITYPE'].isin([12]), 'type'] = 'ianote'
# callable and noncallable bonds
data.loc[data['ITYPE'].isin([1, 5]), 'type'] = 'bond'
# inflation-adjusted bonds
data.loc[data['ITYPE'].isin([11]), 'type'] = 'iabond'

# term
# m3 = less than 3 months
# m6 = less than 6 months (and greater than 3 month)
# y1 = less than 1 year (...)
# y2 = less than 2 years
# y3 = less than 3 years
# y5 = less than 5 years
# y7 = less than 7 years
# y10= less than 10 years
# y20= less than 20 years
# y30= less than 30 years

data['term_days'] = (data['TMATDT'] - data['TDATDT'])
data['term'] = ''

data.loc[(data['term_days'] >= pd.Timedelta('1D')) &
         (data['term_days'] <= pd.Timedelta('60D')) &
         (data['type'] == 'bill'), 'term'] = 'm3'
data.loc[(data['term_days'] >= pd.Timedelta('123D')) &
         (data['term_days'] <= pd.Timedelta('191D')) &
         (data['type'] == 'bill'), 'term'] = 'm6'
data.loc[(data['term_days'] >= pd.Timedelta('210D')) &
         (data['term_days'] <= pd.Timedelta('372D')) &
         (data['type'] == 'bill'), 'term'] = 'm12'

data.loc[(data['term_days'] >= pd.Timedelta('433D')) &
         (data['term_days'] <= pd.Timedelta('875D')) &
         (data['type'] == 'note'), 'term'] = 'y1'
data.loc[(data['term_days'] >= pd.Timedelta('1004D')) &
         (data['term_days'] <= pd.Timedelta('1188D')) &
         (data['type'] == 'note'), 'term'] = 'y3'
data.loc[(data['term_days'] >= pd.Timedelta('1212D')) &
         (data['term_days'] <= pd.Timedelta('1903D')) &
         (data['type'] == 'note'), 'term'] = 'y5'
data.loc[(data['term_days'] >= pd.Timedelta('1936D')) &
         (data['term_days'] <= pd.Timedelta('2646D')) &
         (data['type'] == 'note'), 'term'] = 'y7'
data.loc[(data['term_days'] >= pd.Timedelta('2830D')) &
         (data['term_days'] <= pd.Timedelta('3653D')) &
         (data['type'] == 'note'), 'term'] = 'y10'

data.loc[(data['term_days'] >= pd.Timedelta('1857D')) &
         (data['term_days'] <= pd.Timedelta('6239D')) &
         (data['type'] == 'bond'), 'term'] = 'y10_bond'
data.loc[(data['term_days'] >= pd.Timedelta('7305D')) &
```

```
         (data['term_days'] <= pd.Timedelta('7693D')) &
         (data['type'] == 'bond'), 'term'] = 'y20'
data.loc[(data['term_days'] >= pd.Timedelta('9128D')) &
         (data['term_days'] <= pd.Timedelta('14610D')) &
         (data['type'] == 'bond'), 'term'] = 'y30'

# # first trading date (or first trading date in our data)
# CALDT_f = data.groupby('KYTREASNO').first()['CALDT']
# CALDT_l = data.groupby('KYTREASNO').last()['CALDT']

# %%
# # test part
# test = data[(data['KYTREASNO'] == 202479) &
#             (data['CALDT'] >= '1973-01-01') &
#             (data['CALDT'] < '1974-01-01')].copy(deep=True)
# test['logr'] = np.log(1+test['TDRETNUA'])
# monthly_return_set3 = test[['logr']] \
#     .resample('MS', closed='left', label='left') \
#     .sum().apply(lambda x: np.exp(x)-1)

# # compare them
# pd.concat([monthly_return['1973-01-01':'1973-12-01'],
#            monthly_return_set3], axis=1).plot()
# # almost the same for month 2 to 8 and 12
# # different for month 1, 9, 10 and 11
# # it works!

# # find the id

# data[(data['TMATDT'] == '2036-02-15')].groupby('KYTREASNO').first()

# data[(data['TMATDT'] >= '2037-01-01') &
#      (data['TMATDT'] <= '2041-01-31') &
#      (data['term'] == 'y30')].groupby('KYTREASNO').first()

# data[(data['TCOUPRT'] == 11.875)].groupby('KYTREASNO').last()

# data[(data['KYTREASNO'] == 204015)]

# %%
# compare daily data and monthly data

# # 1961-6-15 to 1965
# # id: 201846
# pd.concat([
#     data[(data['KYTREASNO'] == 201846) &
#          (data['CALDT'] >= '1961-07-01') &
#          (data['CALDT'] < '1966-01-01')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['1961-07-01':'1965-12-01']], axis=1) \
#     .plot()
```

```
# # 1966 to 1973-01-05
# # id: 202439
# pd.concat([
#     data[(data['KYTREASNO'] == 202439) &
#          (data['CALDT'] >= '1966-01-01') &
#          (data['CALDT'] < '1973-01-01')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['1966-01-01':'1972-12-01']], axis=1) \
#     .plot()

# # 1973-01-08 to 1974
# # id: 202479
# pd.concat([
#     data[(data['KYTREASNO'] == 202479) &
#          (data['CALDT'] >= '1973-01-01') &
#          (data['CALDT'] < '1975-01-01')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['1973-01-01':'1974-12-01']], axis=1) \
#     .plot()

# # 1975 to 1976
# # id: 203044
# pd.concat([
#     data[(data['KYTREASNO'] == 203044) &
#          (data['CALDT'] >= '1975-01-01') &
#          (data['CALDT'] < '1977-01-01')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['1975-01-01':'1976-12-01']], axis=1) \
#     .plot()

# # 1977 to 1980
# # id: 203113
# pd.concat([
#     data[(data['KYTREASNO'] == 203113) &
#          (data['CALDT'] >= '1977-01-01') &
#          (data['CALDT'] < '1981-01-01')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['1977-01-01':'1980-12-01']], axis=1) \
#     .plot()

# # 1981
# # id: 203253
# pd.concat([
#     data[(data['KYTREASNO'] == 203253) &
#          (data['CALDT'] >= '1981-01-01') &
#          (data['CALDT'] < '1982-01-01')][['TDRETNUA']]
```

```
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['1981-01-01':'1981-12-01']], axis=1) \
#     .plot()

# # 1982
# # id: 203251
# pd.concat([
#     data[(data['KYTREASNO'] == 203251) &
#          (data['CALDT'] >= '1982-01-01') &
#          (data['CALDT'] < '1983-01-01')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['1982-01-01':'1982-12-01']], axis=1) \
#     .plot()

# # 1983
# # id: 203380
# pd.concat([
#     data[(data['KYTREASNO'] == 203380) &
#          (data['CALDT'] >= '1983-01-01') &
#          (data['CALDT'] < '1984-01-01')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['1983-01-01':'1983-12-01']], axis=1) \
#     .plot()

# # 1984
# # id: 203446
# pd.concat([
#     data[(data['KYTREASNO'] == 203446) &
#          (data['CALDT'] >= '1984-01-01') &
#          (data['CALDT'] < '1985-01-01')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['1984-01-01':'1984-12-01']], axis=1) \
#     .plot()

# # 1985
# # id: 203955
# pd.concat([
#     data[(data['KYTREASNO'] == 203955) &
#          (data['CALDT'] >= '1985-01-01') &
#          (data['CALDT'] < '1986-01-01')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['1985-01-01':'1985-12-01']], axis=1) \
#     .plot()
```

```
# # 1986 to 1989
# # id: 203963
# pd.concat([
#     data[(data['KYTREASNO'] == 203963) &
#          (data['CALDT'] >= '1986-01-01') &
#          (data['CALDT'] < '1990-01-01')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['1986-01-01':'1989-12-01']], axis=1) \
#     .plot()

# # 1990 to 1992
# # id: 204015
# pd.concat([
#     data[(data['KYTREASNO'] == 204015) &
#          (data['CALDT'] >= '1990-01-01') &
#          (data['CALDT'] < '1993-01-01')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['1990-01-01':'1992-12-01']], axis=1) \
#     .plot()

# # 1993 to 1996
# # id: 204052
# pd.concat([
#     data[(data['KYTREASNO'] == 204052) &
#          (data['CALDT'] >= '1993-01-01') &
#          (data['CALDT'] < '1996-12-31')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['1993-01-01':'1996-12-01']], axis=1) \
#     .plot()

# # 1997 to 1998
# # id: 204071
# pd.concat([
#     data[(data['KYTREASNO'] == 204071) &
#          (data['CALDT'] >= '1997-01-01') &
#          (data['CALDT'] < '1998-12-31')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['1997-01-01':'1998-12-01']], axis=1) \
#     .plot()

# # 1999 to 2001
# # id: 204077
# pd.concat([
#     data[(data['KYTREASNO'] == 204077) &
#          (data['CALDT'] >= '1999-01-01') &
#          (data['CALDT'] < '2001-12-31')][['TDRETNUA']]
```

```
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['1999-01-01':'2001-12-01']], axis=1) \
#     .plot()

# # 2002
# # id: 204082
# pd.concat([
#     data[(data['KYTREASNO'] == 204082) &
#          (data['CALDT'] >= '2002-01-01') &
#          (data['CALDT'] < '2002-12-31')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['2002-01-01':'2002-12-01']], axis=1) \
#     .plot()

# # 2003 to 2004
# # id: 204083
# pd.concat([
#     data[(data['KYTREASNO'] == 204083) &
#          (data['CALDT'] >= '2003-01-01') &
#          (data['CALDT'] < '2004-12-31')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['2003-01-01':'2004-12-01']], axis=1) \
#     .plot()

# # 2005
# # id: 204085
# pd.concat([
#     data[(data['KYTREASNO'] == 204085) &
#          (data['CALDT'] >= '2005-01-01') &
#          (data['CALDT'] < '2005-12-31')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['2005-01-01':'2005-12-01']], axis=1) \
#     .plot()

# # 2006
# # id: 204087
# pd.concat([
#     data[(data['KYTREASNO'] == 204087) &
#          (data['CALDT'] >= '2006-01-01') &
#          (data['CALDT'] < '2006-12-31')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['2006-01-01':'2006-12-01']], axis=1) \
#     .plot()
```

```
# # 2007
# # id: 204090
# pd.concat([
#     data[(data['KYTREASNO'] == 204090) &
#          (data['CALDT'] >= '2007-01-01') &
#          (data['CALDT'] < '2007-12-31')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['2007-01-01':'2007-12-01']], axis=1) \
#     .plot()

# # 2008
# # id: 204092
# pd.concat([
#     data[(data['KYTREASNO'] == 204092) &
#          (data['CALDT'] >= '2008-01-01') &
#          (data['CALDT'] < '2008-12-31')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['2008-01-01':'2008-12-01']], axis=1) \
#     .plot()

# # 2009
# # id: 204094
# pd.concat([
#     data[(data['KYTREASNO'] == 204094) &
#          (data['CALDT'] >= '2009-01-01') &
#          (data['CALDT'] < '2009-12-31')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['2009-01-01':'2009-12-01']], axis=1) \
#     .plot()

# # 2010 to 2012
# # id: 204097
# pd.concat([
#     data[(data['KYTREASNO'] == 204097) &
#          (data['CALDT'] >= '2010-01-01') &
#          (data['CALDT'] < '2012-12-31')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['2010-01-01':'2012-12-01']], axis=1) \
#     .plot()

# # 2013 to 2016
# # id: 204098
# pd.concat([
#     data[(data['KYTREASNO'] == 204098) &
#          (data['CALDT'] >= '2013-01-01') &
#          (data['CALDT'] < '2016-12-31')][['TDRETNUA']]
```

```
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['2013-01-01':'2016-12-01']], axis=1) \
#     .plot()

# # 2017
# # id: 204100
# pd.concat([
#     data[(data['KYTREASNO'] == 204100) &
#          (data['CALDT'] >= '2017-01-01') &
#          (data['CALDT'] < '2017-12-31')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['2017-01-01':'2017-12-01']], axis=1) \
#     .plot()

# # 2018
# # id: 204102
# pd.concat([
#     data[(data['KYTREASNO'] == 204102) &
#          (data['CALDT'] >= '2018-01-01') &
#          (data['CALDT'] < '2018-12-31')][['TDRETNUA']]
#     .apply(lambda x: np.log(1+x))
#     .resample('MS', closed='left', label='left')
#     .sum().apply(lambda x: np.exp(x)-1),
#     monthly_return['2018-01-01':'2018-12-01']], axis=1) \
#     .plot()

# # no dataset2 month data from 2019

# %%
# treasury daily return
treasury_return = pd.concat([
    data[(data['KYTREASNO'] == 201846) &
         (data['CALDT'] >= '1961-6-15') &
         (data['CALDT'] <= '1965-12-31')][['TDRETNUA']],
    data[(data['KYTREASNO'] == 202439) &
         (data['CALDT'] >= '1966-01-05') &
         (data['CALDT'] <= '1972-12-31')][['TDRETNUA']],
    data[(data['KYTREASNO'] == 202479) &
         (data['CALDT'] >= '1973-01-01') &
         (data['CALDT'] <= '1974-12-31')][['TDRETNUA']],
    data[(data['KYTREASNO'] == 203044) &
         (data['CALDT'] >= '1975-01-01') &
         (data['CALDT'] <= '1976-12-31')][['TDRETNUA']],
    data[(data['KYTREASNO'] == 203113) &
         (data['CALDT'] >= '1977-01-01') &
         (data['CALDT'] <= '1980-12-31')][['TDRETNUA']],
    data[(data['KYTREASNO'] == 203253) &
         (data['CALDT'] >= '1981-01-01') &
         (data['CALDT'] <= '1981-12-31')][['TDRETNUA']],
    data[(data['KYTREASNO'] == 203251) &
```

```
          (data['CALDT'] >= '1982-01-01') &
          (data['CALDT'] <= '1982-12-31')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 203380) &
          (data['CALDT'] >= '1983-01-01') &
          (data['CALDT'] <= '1983-12-01')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 203446) &
          (data['CALDT'] >= '1984-01-01') &
          (data['CALDT'] <= '1984-12-31')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 203955) &
          (data['CALDT'] >= '1985-01-01') &
          (data['CALDT'] <= '1985-12-31')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 203963) &
          (data['CALDT'] >= '1986-01-01') &
          (data['CALDT'] <= '1989-12-31')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 204015) &
          (data['CALDT'] >= '1990-01-01') &
          (data['CALDT'] <= '1992-12-31')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 204052) &
          (data['CALDT'] >= '1993-01-01') &
          (data['CALDT'] <= '1996-12-31')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 204071) &
          (data['CALDT'] >= '1997-01-01') &
          (data['CALDT'] <= '1998-12-31')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 204077) &
          (data['CALDT'] >= '1999-01-01') &
          (data['CALDT'] <= '2001-12-31')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 204082) &
          (data['CALDT'] >= '2002-01-01') &
          (data['CALDT'] <= '2002-12-31')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 204083) &
          (data['CALDT'] >= '2003-01-01') &
          (data['CALDT'] <= '2004-12-31')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 204085) &
          (data['CALDT'] >= '2005-01-01') &
          (data['CALDT'] <= '2005-12-31')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 204087) &
          (data['CALDT'] >= '2006-01-01') &
          (data['CALDT'] <= '2006-12-31')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 204090) &
          (data['CALDT'] >= '2007-01-01') &
          (data['CALDT'] <= '2007-12-31')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 204092) &
          (data['CALDT'] >= '2008-01-01') &
          (data['CALDT'] <= '2008-12-31')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 204094) &
          (data['CALDT'] >= '2009-01-01') &
          (data['CALDT'] <= '2009-12-31')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 204097) &
          (data['CALDT'] >= '2010-01-01') &
          (data['CALDT'] <= '2012-12-31')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 204098) &
          (data['CALDT'] >= '2013-01-01') &
          (data['CALDT'] <= '2016-12-31')][['TDRETNUA']],
      data[(data['KYTREASNO'] == 204100) &
```

```
          (data['CALDT'] >= '2017-01-01') &
          (data['CALDT'] <= '2017-12-31')][['TDRETNUA']],
     data[(data['KYTREASNO'] == 204102) &
          (data['CALDT'] >= '2018-01-01') &
          (data['CALDT'] <= '2018-12-31')][['TDRETNUA']],
     data[(data['KYTREASNO'] == 206059) &
          (data['CALDT'] >= '2019-01-01') &
          (data['CALDT'] <= '2019-12-31')][['TDRETNUA']],
     data[(data['KYTREASNO'] == 206112) &
          (data['CALDT'] >= '2020-01-01') &
          (data['CALDT'] <= '2020-03-31')][['TDRETNUA']]], axis=0)

# daily treasury log return
treasury_return['rb'] = np.log(1+treasury_return['TDRETNUA'])
treasury_return.index.name = 'Date'

# %%
# daily spx
# import spx data
path_spx_bb = data_path / 'SPX-daily-1928.txt'
spx_bb = pd.read_table(path_spx_bb, sep='\t')
# delete rows with missing value (reasonable missing)
spx_bb = spx_bb.loc[spx_bb.iloc[:, 1].notnull()]
# price be float
spx_bb.loc[spx_bb.loc[:, 'price'].notnull(), 'price'] = \
    spx_bb.loc[spx_bb.loc[:, 'price'].notnull(), 'price'] \
    .map(lambda x: x.replace(',', '.'))
spx_bb.loc[:, 'price'] = spx_bb.loc[:, 'price'].map(float)
# use datetime type
spx_bb['Date'] = pd.to_datetime(spx_bb.Date)
# order from past to now
spx_bb.sort_values(by='Date', inplace=True)
# set Date as index
spx_bb.reset_index(drop=True, inplace=True)
spx_bb.set_index('Date', drop=True, inplace=True)
# copy to data_spx
data_spx = spx_bb.copy()
# delete volumn column
data_spx.drop(['volumn'], axis=1, inplace=True)
# change colname 'price' to 'spx'
data_spx.columns = ['spx']

# %%
# daily spx (2019 to 2020)
path_spx_slf = data_path / 'SP500_2009-2020.csv'
spx_slf = pd.read_table(path_spx_slf, sep=',')
# replace '.' to nan
spx_slf[spx_slf.loc[:, 'SP500'] == '.'] = (
     spx_slf[spx_slf.loc[:, 'SP500'] == '.'].replace('.', np.nan))
# delete missing value
spx_slf = spx_slf.loc[spx_slf.loc[:, 'SP500'].notnull(), ]
# price be float
spx_slf.loc[:, 'SP500'] = spx_slf.loc[:, 'SP500'].map(float)
# use datetime type
```

```
spx_slf.DATE = pd.to_datetime(spx_slf.DATE)
# set Date as index
spx_slf.reset_index(drop=True, inplace=True)
spx_slf.set_index('DATE', drop=True, inplace=True)
# add 2019-07-13 to 2020-10-25 to data_spx
spx_slf.index.name = 'Date'
spx_slf.columns = ['spx']
data_spx = pd.concat([data_spx, spx_slf['2019-10-25': '2020-06-15']],
                     axis=0)

# %%
# daily log risk free rate
data_rf = pd.read_csv(data_path / "F-F_Research_Data_Factors_daily.csv",
                      header=3, usecols=[0, 4])[:-1]
data_rf['Date'] = pd.to_datetime(data_rf['Unnamed: 0'], format='%Y%m%d')
# the RF is in percent
data_rf['RF'] = data_rf['RF']/100
# daily log risk free rate
data_rf['rf'] = np.log(1+data_rf['RF'])
data_rf = data_rf[['Date', 'rf']]
data_rf.set_index('Date', drop=True, inplace=True)

# %%
data_output = (pd.concat([treasury_return['rb'], data_spx, data_rf], axis=1)
               ['1961-06-15': '2020-03-31'])
# delete NaN rows
# for both rb and rf are NaN, delete (then spx and rf NaN rows are same)
data_output = data_output[(data_output['rb'].notnull() |
                           data_output['rf'].notnull())]
# for deleted rf (where rb is NaN), add to next date
data_temp = data_output.copy(deep=True)
data_temp.reset_index(inplace=True)
nan_index = data_temp[data_temp['rb'].isnull()].index
for i in nan_index:
    data_output.iloc[i+1]['rf'] = (data_output.iloc[i+1]['rf'] +
                                   data_output.iloc[i]['rf'])
# for deleted spx (where rb is NaN), delete
data_output = data_output[data_output['rb'].notnull()]
# for deleted rb (where spx and rf are NaN), add to next date
data_temp = data_output.copy(deep=True)
data_temp.reset_index(inplace=True)
nan_index = data_temp[data_temp['spx'].isnull()].index
for i in nan_index:
    data_output.iloc[i+1]['rb'] = (data_output.iloc[i+1]['rb'] +
                                   data_output.iloc[i]['rb'])
data_output = data_output[data_output['spx'].notnull()]
# rs
data_output['rs'] = np.log(data_spx['spx']).diff()
data_output = data_output[1:]
data_output = data_output[['rb', 'rs', 'rf']]
# output
data_output.to_csv(output_path / "dataset3.csv")
```

```
# %%
```

## Daily Data Description

```
# %%
import pandas as pd
import numpy as np
from pathlib import Path
import matplotlib
import matplotlib.pyplot as plt
import datetime
import missingno as msno
from pandas.plotting import register_matplotlib_converters
from tool9 import (RRP, period, es, performance1, portfolio_risk,
                   period_4_plot, describe)

# the relative path of folder
output_path = Path('output/')
data_path = Path('raw-data/')
graphics_path = Path('../LaTeX_thesis/graphics/')
# read data from csv file
data = pd.read_csv(output_path / "dataset3.csv", parse_dates=['Date'],
                   index_col='Date')

days_of_year = 252
# historical volatility (std)
std_count = days_of_year//2
data[['std_rs', 'std_rb']] = (
    data[['rs', 'rb']].rolling(std_count).std()*np.sqrt(days_of_year))
# realized volatility
rv_count = days_of_year//2
data[['rv_rs', 'rv_rb']] = (
    np.sqrt((data[['rs', 'rb']]**2).rolling(rv_count).sum()) *
    np.sqrt(days_of_year/rv_count))
# excess return
data['excess_rs'] = data['rs'] - data['rf']
data['excess_rb'] = data['rb'] - data['rf']

# delete NaN
data = data[data['std_rs'].notnull()]

# %%
# rolling correlation
corr_count = days_of_year
corre = data[['rs', 'rb']].rolling(corr_count).corr()['rs']. \
    swaplevel().loc['rb']
corre = corre[corre.notnull()]

# positive correlation percentage
print('positive correlation percentage')
```

```
print((corre > 0).sum() / corre.count())

# save to csv
# corre.to_csv(output_path / 'corre.csv')

# plot
fig, ax = plt.subplots(nrows=1, ncols=1,
                       figsize=(12.5, 6), sharex=False)
corre.plot(ax=ax, linewidth=0.5)
ax.set_ylabel('rolling correlation')
ax.set_title('252-day rolling Correlation of S&P 500 and LTR')
fig.tight_layout()
fig.savefig(graphics_path /'252-day rolling corre of daily sp500 and ltr')

# %%
# base statistics table
pd.set_option('display.precision', 4)
description = pd.concat(
    [describe('rs', data), describe('rb', data), describe('std_rs', data),
     describe('std_rb', data), describe('rv_rs', data),
     describe('rv_rb', data)], axis=0)
description
# pd.reset_option('display.precision')

# %%
# assets performance table
pd.set_option('display.precision', 4)
table = pd.concat([performance1(data['rb'], data['rf']),
                   performance1(data['rs'], data['rf'])], axis=0)
table.index = ('rb', 'rs')
table

# %%
pd.reset_option('display.precision')

# %%
# plot cumulative rs, rb and cumulative excess rs, excess rb
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(12, 10), sharex=True)

pd.concat([data[['rb']].cumsum(), data[['rs']].cumsum()], axis=1). \
    plot(ax=axes[0], linewidth=0.5)
axes[0].set_ylabel('cumulative log return')
axes[0].set_title('a. cumulative log return of S&P 500 and LTR')
pd.concat([data[['excess_rb']].cumsum(), data[['excess_rs']].cumsum()],
          axis=1).plot(ax=axes[1], linewidth=0.5)
axes[1].set_ylabel('cumulative excess log return')
axes[1].set_title('b. cumulative excess log return of S&P 500 and LTR')
fig.tight_layout()
fig.savefig(graphics_path /'cum return of daily sp500 and ltr')

# %%
data[['rb']].cumsum()
data[['rs']].cumsum()
np.exp(4.183181)
```

```
np.exp(3.221999)

# %%
# rv_rb - rv_rs
data[(data['rv_rb'] - data['rv_rs']) > 0]. \
    to_csv(output_path / 'rblargerb.csv')

# %%
# plot volatility and cumulative log return together
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(12, 10), sharex=True)
pd.concat([data[['rv_rs']], data[['rs']].cumsum()], axis=1). \
    plot(ax=axes[0], secondary_y=['rs'], linewidth=0.5)
axes[0].set_ylabel('volatility')
axes[0].right_ax.set_ylabel('cumulative log return')
axes[0].set_title('rv_rs and cumulative rs')
pd.concat([data[['rv_rb']], data[['rb']].cumsum()], axis=1). \
    plot(ax=axes[1], secondary_y=['rb'], linewidth=0.5)
axes[1].set_ylabel('volatility')
axes[1].right_ax.set_ylabel('cumulative log return')
axes[1].set_title('rv_rb and cumulative rb')

# %%
# plot historical volatility in rising and falling interest rate environments
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(12.5, 6), sharex=False)
data[['rv_rs', 'rv_rb']].plot(ax=ax, linewidth=0.5)
ax.set_ylabel('historical volatility')
ax.set_title('volatility in rising and falling interest rate environments')
y1, y2 = ax.get_ylim()
ax.fill_between(['1926-01', '1945-01'], y1=y1, y2=y2, alpha=.25, color='r')
ax.fill_between(['1945-01', '1970-01'], y1=y1, y2=y2, alpha=.25, color='g')
ax.fill_between(['1970-01', '1992-01'], y1=y1, y2=y2, alpha=.25, color='y')
ax.fill_between(['1992-01', '2005-01'], y1=y1, y2=y2, alpha=.25, color='g')
ax.fill_between(['2005-01', '2013-01'], y1=y1, y2=y2, alpha=.25, color='y')
ax.fill_between(['2013-01', '2020-01'], y1=y1, y2=y2, alpha=.25, color='g')
```

# RPP Construction and Performance (Daily data)

```
# %%
import pandas as pd
import numpy as np
from pathlib import Path
import matplotlib
import matplotlib.pyplot as plt
import datetime
import missingno as msno
from arch import arch_model
from tool9 import (RRP, period, es, performance1, portfolio_risk,
                   period_4_plot, drawdown, maxdrawdown, var)
from sklearn.linear_model import LinearRegression
```

```
# global settings
output_path = Path('output/')
data_path = Path('raw-data/')
graphics_path = Path('../LaTeX_thesis/graphics/')

# read data from csv file
data = pd.read_csv(output_path / "dataset3.csv", parse_dates=['Date'],
                   index_col='Date')

# period
[start, end] = period('rs', data=data)

# %%
days_of_year = 252

# correlation between rs and rb
corr_count = days_of_year
corr_rolling = data[['rs', 'rb']].rolling(corr_count).corr().shift(2)
# because of the rolling, we have NaN values, delete the missing rows
corr_rolling = corr_rolling[corr_rolling['rs'].notnull()]
corr_start = corr_rolling[:1].index[0][0]

# realized volatility
rv_count = days_of_year//2
data[['rv_rs', 'rv_rb']] = (
    np.sqrt((data[['rs', 'rb']]**2).rolling(rv_count).sum()) *
    np.sqrt(days_of_year/rv_count)
).shift(1)

# rolling yearly return
data[['rolling_rs', 'rolling_rb']] = (
    (data[['rs', 'rb']].rolling(rv_count).mean()*days_of_year)
    .shift(1))

# because of the rolling, we have NaN values, delete the missing rows
data = data[data['rv_rs'].notnull()]
data = data[data['rolling_rs'].notnull()]
data_start = data[:1].index[0]

# from the same start point
data_start = max(corr_start, data_start)
corr_rolling = corr_rolling[data_start:]
data = data[data_start:]

# %%
# 60/40 portfolio
p0 = RRP(
    ratio_fixed=[0.6, 0.4],
    logr=data[['rs', 'rb']],
    risk=data[['rv_rs', 'rv_rb']],
    corr=corr_rolling[['rs', 'rb']],
    reset_months=12,
    get_actual=True
```

```
)
# reset date shift half year
p0_h = RRP(
    first_reset_date='1962-12-20',
    ratio_fixed=[0.6, 0.4],
    logr=data[['rs', 'rb']],
    corr=corr_rolling[['rs', 'rb']],
    reset_months=12,
    get_actual=False
)

plt.plot(p0.logr_p.cumsum())
plt.plot(p0_h.logr_p.cumsum())

# %%
# risk parity portfolio (target volatility)
# the correlation and risk in the whole period
data_corr = data[['rs', 'rb']].corr()
data_vol = np.sqrt((data[['rs', 'rb']]**2).sum()) * \
    np.sqrt(days_of_year/len(data))
data_ratio = [0.6, 0.4]
t_target_risk = portfolio_risk(data_corr=data_corr,
                               data_vol=data_vol, data_ratio=data_ratio)
# constract portfolio
p1 = RRP(
    logr=data[['rs', 'rb']],
    risk=data[['rv_rs', 'rv_rb']],
    corr=corr_rolling[['rs', 'rb']],
    reset_months=12,
    target_risk=t_target_risk,
    leverage_limit=6,
    get_actual=True
)
# reset date shift half year
p1_h = RRP(
    first_reset_date='1962-12-20',
    logr=data[['rs', 'rb']],
    risk=data[['rv_rs', 'rv_rb']],
    corr=corr_rolling[['rs', 'rb']],
    reset_months=12,
    target_risk=t_target_risk,
    leverage_limit=6,
    get_actual=True
)

plt.plot(p1.logr_p.cumsum())
plt.plot(p1_h.logr_p.cumsum())

# %%
# performance of p0 and p1
pd.set_option('display.precision', 4)
table = pd.concat([performance1(p0.logr_p.iloc[:, 0], data['rf']),
                   performance1(p1.logr_p.iloc[:, 0], data['rf'])], axis=0)
table.index = ('p0', 'p1')
```

```
table

# %%
pd.reset_option('display.precision')

# %%
# plot RPP
fig, axes = plt.subplots(nrows=5, ncols=1, figsize=(12, 12), sharex=True)

tmp = pd.concat([(p1.logr_p['portfolio']-data['rf']).cumsum(),
                 (p0.logr_p['portfolio']-data['rf']).cumsum()], axis=1)
tmp.columns = ['RPP', '60/40']
tmp.plot(ax=axes[0], linewidth=0.5)
axes[0].set_title('Cumulative excess log returns')
axes[0].set_xlim(start, end)
y1, y2 = axes[0].get_ylim()
axes[0].fill_between(['1969-12', '1970-11'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['1973-11', '1975-03'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['1980-01', '1980-07'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['1981-07', '1982-11'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['1990-07', '1991-03'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['2001-03', '2001-11'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['2007-12', '2009-06'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['2020-02', '2020-03'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
tmp = p1.logr_p.rolling(days_of_year).sum() - \
    p0.logr_p.rolling(days_of_year).sum()
tmp.columns = ['RPP minus 60/40']
tmp.plot(ax=axes[1], linewidth=0.5)
axes[1].set_title('1-year rolling log returns difference')
y1, y2 = axes[1].get_ylim()
axes[1].fill_between(['1969-12', '1970-11'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[1].fill_between(['1973-11', '1975-03'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[1].fill_between(['1980-01', '1980-07'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[1].fill_between(['1981-07', '1982-11'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[1].fill_between(['1990-07', '1991-03'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[1].fill_between(['2001-03', '2001-11'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[1].fill_between(['2007-12', '2009-06'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[1].fill_between(['2020-02', '2020-03'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
```

```
p1.ratio['rb'].plot(ax=axes[2], linewidth=0.5, label='theoretical bond weight')
p1.ratio_actual['rb'].plot(ax=axes[2], linewidth=0.5,
                           label='actual bond weight')
axes[2].set_ylim(0.3, 1)
axes[2].set_title('weight of bond in RPP')
axes[2].legend()

tmp = drawdown(p1.logr_p)-drawdown(p0.logr_p)
tmp.columns = ['RPP minus 60/40']
tmp.plot(ax=axes[3], linewidth=0.5)
axes[3].set_title('portfolio drawdown difference')
y1, y2 = axes[3].get_ylim()
axes[3].fill_between(['1969-12', '1970-11'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[3].fill_between(['1973-11', '1975-03'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[3].fill_between(['1980-01', '1980-07'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[3].fill_between(['1981-07', '1982-11'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[3].fill_between(['1990-07', '1991-03'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[3].fill_between(['2001-03', '2001-11'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[3].fill_between(['2007-12', '2009-06'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[3].fill_between(['2020-02', '2020-03'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
p1.leverage.plot(ax=axes[4], linewidth=0.5)
axes[4].set_ylim(0, 6.5)
axes[4].set_title('leverage')
fig.tight_layout()
fig.savefig(graphics_path / 'portfolio performance')

# %%
# plot RPP
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(12.5, 6), sharex=True)
tmp = pd.concat([drawdown(p1.logr_p), drawdown(p0.logr_p)], axis=1)
tmp.columns = ['RPP', '60/40']
tmp.plot(ax=ax, linewidth=0.5)
ax.set_title('portfolio drawdown')
y1, y2 = ax.get_ylim()
ax.fill_between(['1969-12', '1970-11'], y1=y1,
                y2=y2, alpha=.15, color='tab:gray')
ax.fill_between(['1973-11', '1975-03'], y1=y1,
                y2=y2, alpha=.15, color='tab:gray')
ax.fill_between(['1980-01', '1980-07'], y1=y1,
                y2=y2, alpha=.15, color='tab:gray')
ax.fill_between(['1981-07', '1982-11'], y1=y1,
                y2=y2, alpha=.15, color='tab:gray')
ax.fill_between(['1990-07', '1991-03'], y1=y1,
                y2=y2, alpha=.15, color='tab:gray')
ax.fill_between(['2001-03', '2001-11'], y1=y1,
                y2=y2, alpha=.15, color='tab:gray')
```

```
ax.fill_between(['2007-12', '2009-06'], y1=y1,
                y2=y2, alpha=.15, color='tab:gray')
ax.fill_between(['2020-02', '2020-03'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
fig.tight_layout()
fig.savefig(graphics_path / 'portfolio performance2')

# %%
# plot volatility of rs and rb
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(12.5, 6), sharex=False)
data[['rv_rs', 'rv_rb']].plot(ax=axes[0], linewidth=0.5)
axes[0].set_ylabel('126-day realized volatility')
axes[0].set_ylim(0, 0.6)
axes[0].set_title('Realized Volatility of S&P 500 and LTR')
# plot density function of rs and rb
y1, y2 = axes[0].get_ylim()
axes[0].fill_between(['1969-12', '1970-11'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['1973-11', '1975-03'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['1980-01', '1980-07'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['1981-07', '1982-11'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['1990-07', '1991-03'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['2001-03', '2001-11'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['2007-12', '2009-06'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['2020-02', '2020-03'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
data[['rv_rs', 'rv_rb']].plot.kde(bw_method=0.8, ax=axes[1], linewidth=0.5)
axes[1].set_ylabel('Density')
axes[1].set_ylim(0, 8)
axes[1].set_xlabel('RV Values')
axes[1].set_title('Unconditional RV distributions of S&P 500 and LTR')
fig.tight_layout()
fig.savefig(graphics_path / 'rv and kernal')

# %%
# import Treasury Constant Maturity Rate data (data_tr)
# list of files
file_list = ['DGS1MO.csv', 'DGS3MO.csv', 'DGS6MO.csv',
             'DGS1.csv', 'DGS2.csv', 'DGS3.csv',
             'DGS5.csv', 'DGS7.csv', 'DGS10.csv',
             'DGS20.csv', 'DGS30.csv']
# list of colnames
colname_list = ['tr_1m', 'tr_3m', 'tr_6m', 'tr_1y',
                'tr_2y', 'tr_3y', 'tr_5y', 'tr_7y',
                'tr_10y',  'tr_20y',  'tr_30y', ]
data_tr = pd.DataFrame()
# inport 6 file into data_move
for i in range(len(file_list)):
```

```
    # read file into df_file
    path_file = data_path / file_list[i]
    df_file = pd.read_table(path_file, sep=',', )
    # '.' is NaN value
    df_file.iloc[:, 1].replace('.', np.nan, inplace=True)
    # delete mising value
    df_file.dropna(axis=0, inplace=True)
    # price be float
    df_file.iloc[:, 1] = df_file.iloc[:, 1].map(float)
    # change colnames
    df_file.columns = ['Date', colname_list[i]]
    # use datetime type
    df_file.Date = pd.to_datetime(df_file.Date)
    # set Date as index
    df_file.set_index('Date', drop=True, inplace=True)
    # cancat with data_move
    data_tr = pd.concat([data_tr, df_file], axis=1)
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(12.5, 6), sharex=False)
data_tr['tr_10y']['1962-06-20':'2020-03-31'].plot(ax=ax, linewidth=0.5)
ax.set_ylabel('Yield')
ax.set_title('10-Year Treasury Constant Maturity Rate ')
y1, y2 = ax.get_ylim()
ax.fill_between(['1962-06-20', '1979-09-30'], y1=y1,
                y2=y2, alpha=.15, color='tab:grey')
ax.fill_between(['1979-10-01', '1981-09-30'], y1=y1,
                y2=y2, alpha=.15, color='tab:green')
ax.fill_between(['1981-10-01', '2020-03-31'], y1=y1,
                y2=y2, alpha=.15, color='tab:blue')
fig.tight_layout()
fig.savefig(graphics_path / '10 year yield')

# %%
# falling and increasing interest rate
# y is moderate increasing, r is sharply rising, b is falling
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(12, 12), sharex=False)
tmp = pd.concat([(p1.logr_p['portfolio']-data['rf']).cumsum(),
                 (p0.logr_p['portfolio']-data['rf']).cumsum()], axis=1)
tmp.columns = ['RPP', '60/40']
tmp.plot(ax=axes[0], linewidth=0.5)
axes[0].set_ylabel('portfolio cumulative log return')
axes[0].set_title(
    'portfolio cumulative excess log return in rising ' +
    'and falling interest rate situations')
y1, y2 = axes[0].get_ylim()
axes[0].fill_between(['1962-06-20', '1979-09-30'], y1=y1,
                     y2=y2, alpha=.15, color='tab:grey')
axes[0].fill_between(['1979-10-01', '1981-09-30'], y1=y1,
                     y2=y2, alpha=.15, color='tab:green')
axes[0].fill_between(['1981-10-01', '2020-03-31'], y1=y1,
                     y2=y2, alpha=.15, color='tab:blue')

tmp = pd.concat([drawdown(p1.logr_p), drawdown(p0.logr_p)], axis=1)
tmp.columns = ['RPP', '60/40']
tmp.plot(ax=axes[1], linewidth=0.5)
```

```
axes[1].set_ylabel('portfolio cumulative log return ')
axes[1].set_title(
    'portfolio drawdown in rising ' +
    'and falling interest rate situations')
y1, y2 = axes[1].get_ylim()
axes[1].fill_between(['1962-06-20', '1979-09-30'], y1=y1,
                     y2=y2, alpha=.15, color='tab:grey')
axes[1].fill_between(['1979-10-01', '1981-09-30'], y1=y1,
                     y2=y2, alpha=.15, color='tab:green')
axes[1].fill_between(['1981-10-01', '2020-03-31'], y1=y1,
                     y2=y2, alpha=.15, color='tab:blue')
fig.tight_layout()
fig.savefig(graphics_path / 'interest rate')

# %% interest division
pd.set_option('display.precision', 4)
table = pd.concat([
    performance1(p0.logr_p['1962-06-20': '1979-09-30'].iloc[:, 0],
                 data['rf']['1962-06-20': '1979-09-30']),
    performance1(p1.logr_p['1962-06-20':'1979-09-30'].iloc[:, 0],
                 data['rf']['1962-06-20': '1979-09-30'])], axis=0)
table.index = ('p0', 'p1')
table

pd.set_option('display.precision', 4)
table = pd.concat([
    performance1(p0.logr_p['1979-10-01': '1981-09-30'].iloc[:, 0],
                 data['rf']['1979-10-01': '1981-09-30']),
    performance1(p1.logr_p['1979-10-01': '1981-09-30'].iloc[:, 0],
                 data['rf']['1979-10-01': '1981-09-30'])], axis=0)
table.index = ('p0', 'p1')
table

pd.set_option('display.precision', 4)
table = pd.concat([
    performance1(p0.logr_p['1981-10-01': '2020-03-31'].iloc[:, 0],
                 data['rf']['1981-10-01': '2020-03-31']),
    performance1(p1.logr_p['1981-10-01': '2020-03-31'].iloc[:, 0],
                 data['rf']['1981-10-01': '2020-03-31'])], axis=0)
table.index = ('p0', 'p1')
table

pd.reset_option('display.precision')
# %% interest division
## positive and rising
pd.set_option('display.precision', 4)
table = pd.concat([
    performance1(p1.logr_p['1967-11-01': '1979-09-30'].iloc[:, 0],
                 data['rf']['1967-11-01': '1979-09-30']),
    performance1(p0.logr_p['1967-11-01': '1979-09-30'].iloc[:, 0],
                 data['rf']['1967-11-01': '1979-09-30'])], axis=0)
table.index = ('p1', 'p0')
table
## positive and rising
```

```
pd.set_option('display.precision', 4)
table = pd.concat([
    performance1(p1.logr_p['1979-10-01': '1981-09-30'].iloc[:, 0],
                 data['rf']['1979-10-01': '1981-09-30']),
    performance1(p0.logr_p['1979-10-01': '1981-09-30'].iloc[:, 0],
                 data['rf']['1979-10-01': '1981-09-30'])], axis=0)
table.index = ('p1', 'p0')
table
## positive and falling
pd.set_option('display.precision', 4)
table = pd.concat([
    performance1(p1.logr_p['1981-10-01': '1998-07-01'].iloc[:, 0],
                 data['rf']['1981-10-01': '1998-07-01']),
    performance1(p0.logr_p['1981-10-01': '1998-07-01'].iloc[:, 0],
                 data['rf']['1981-10-01': '1998-07-01'])], axis=0)
table.index = ('p1', 'p0')
table

pd.set_option('display.precision', 4)
table = pd.concat([
    performance1(p1.logr_p['1967-11-01': '1998-07-01'].iloc[:, 0],
                 data['rf']['1967-11-01': '1998-07-01']),
    performance1(p0.logr_p['1967-11-01': '1998-07-01'].iloc[:, 0],
                 data['rf']['1967-11-01': '1998-07-01'])], axis=0)
table.index = ('p1', 'p0')
table


# %%
# drawdown
# positive and negative asset correlation
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(12, 12), sharex=False)
tmp = pd.concat([(p1.logr_p['portfolio']-data['rf']).cumsum(),
                 (p0.logr_p['portfolio']-data['rf']).cumsum()], axis=1)
tmp.columns = ['RPP', '60/40']
tmp.plot(ax=axes[0], linewidth=0.5)
axes[0].set_ylabel('cumulative log return ')
axes[0].set_title(
    'cumulative excess log return in positive ' +
    'and negative asset correlation situations')
y1, y2 = axes[0].get_ylim()
axes[0].fill_between(['1962-06-20', '1967-10-31'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['1967-11-01', '1998-07-01'], y1=y1,
                     y2=y2, alpha=.15, color='tab:green')
axes[0].fill_between(['1998-07-02', '2020-03-31'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')

tmp = pd.concat([drawdown(p1.logr_p), drawdown(p0.logr_p)], axis=1)
tmp.columns = ['RPP', '60/40']
tmp.plot(ax=axes[1], linewidth=0.5)
axes[1].set_ylabel('drawdown ')
axes[1].set_title(
    'drawdown in positive and negative asset correlation situations')
```

```
y1, y2 = axes[1].get_ylim()
axes[1].fill_between(['1962-06-20', '1967-10-31'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[1].fill_between(['1967-11-01', '1998-07-01'], y1=y1,
                     y2=y2, alpha=.15, color='tab:green')
axes[1].fill_between(['1998-07-02', '2020-03-31'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
fig.tight_layout()
fig.savefig(graphics_path / 'correlation effect')

# %%
pd.set_option('display.precision', 4)
table = pd.concat([
    performance1(p0.logr_p['1962-06-20': '1967-10-31'].iloc[:, 0],
                 data['rf']['1962-06-20': '1967-10-31']),
    performance1(p1.logr_p['1962-06-20':'1967-10-31'].iloc[:, 0],
                 data['rf']['1962-06-20': '1967-10-31'])], axis=0)
table.index = ('p0', 'p1')
table

table = pd.concat([
    performance1(p0.logr_p['1967-11-01': '1998-07-01'].iloc[:, 0],
                 data['rf']['1967-11-01': '1998-07-01']),
    performance1(p1.logr_p['1967-11-01': '1998-07-01'].iloc[:, 0],
                 data['rf']['1967-11-01': '1998-07-01'])], axis=0)
table.index = ('p0', 'p1')
table

table = pd.concat([
    performance1(p0.logr_p['1998-07-02': '2020-03-31'].iloc[:, 0],
                 data['rf']['1998-07-02': '2020-03-31']),
    performance1(p1.logr_p['1998-07-02': '2020-03-31'].iloc[:, 0],
                 data['rf']['1998-07-02': '2020-03-31'])], axis=0)
table.index = ('p0', 'p1')
table

pd.reset_option('display.precision')

# %% performance in every recession
# ['1969-12-01': '1970-11-30']
# ['1973-11-01': '1975-03-31']
# ['1980-01-01': '1980-07-31']
# ['1981-07-01': '1982-11-30']
# ['1990-07-01': '1991-03-31']
# ['2001-03-01': '2001-11-30']
# ['2007-12-01': '2009-06-30']
# ['2020-02-01': '2020-03-31']
table = pd.concat([
    performance1(p1.logr_p['1969-12-01': '1970-11-30'].iloc[:, 0],
                 data['rf']['1969-12-01': '1970-11-30']),
    performance1(p0.logr_p['1969-12-01': '1970-11-30'].iloc[:, 0],
                 data['rf']['1969-12-01': '1970-11-30'])], axis=0)
table.index = ('p1', 'p0')
table
```

```
table = pd.concat([
    performance1(p1.logr_p['1973-11-01': '1975-03-31'].iloc[:, 0],
                 data['rf']['1973-11-01': '1975-03-31']),
    performance1(p0.logr_p['1973-11-01': '1975-03-31'].iloc[:, 0],
                 data['rf']['1973-11-01': '1975-03-31'])], axis=0)
table.index = ('p1', 'p0')
table

table = pd.concat([
    performance1(p1.logr_p['1980-01-01': '1980-07-31'].iloc[:, 0],
                 data['rf']['1980-01-01': '1980-07-31']),
    performance1(p0.logr_p['1980-01-01': '1980-07-31'].iloc[:, 0],
                 data['rf']['1980-01-01': '1980-07-31'])], axis=0)
table.index = ('p1', 'p0')
table

table = pd.concat([
    performance1(p1.logr_p['1981-07-01':'1982-11-30'].iloc[:, 0],
                 data['rf']['1981-07-01':'1982-11-30']),
    performance1(p0.logr_p['1981-07-01':'1982-11-30'].iloc[:, 0],
                 data['rf']['1981-07-01':'1982-11-30'])], axis=0)
table.index = ('p1', 'p0')
table

table = pd.concat([
    performance1(p1.logr_p['1990-07-01': '1991-03-31'].iloc[:, 0],
                 data['rf']['1990-07-01': '1991-03-31']),
    performance1(p0.logr_p['1990-07-01': '1991-03-31'].iloc[:, 0],
                 data['rf']['1990-07-01': '1991-03-31'])], axis=0)
table.index = ('p1', 'p0')
table

table = pd.concat([
    performance1(p1.logr_p['2001-03-01': '2001-11-30'].iloc[:, 0],
                 data['rf']['2001-03-01': '2001-11-30']),
    performance1(p0.logr_p['2001-03-01': '2001-11-30'].iloc[:, 0],
                 data['rf']['2001-03-01': '2001-11-30'])], axis=0)
table.index = ('p1', 'p0')
table

table = pd.concat([
    performance1(p1.logr_p['2007-12-01': '2009-06-30'].iloc[:, 0],
                 data['rf']['2007-12-01': '2009-06-30']),
    performance1(p0.logr_p['2007-12-01': '2009-06-30'].iloc[:, 0],
                 data['rf']['2007-12-01': '2009-06-30'])], axis=0)
table.index = ('p1', 'p0')
table

table = pd.concat([
    performance1(p1.logr_p['2020-02-01': '2020-03-31'].iloc[:, 0],
                 data['rf']['2020-02-01': '2020-03-31']),
    performance1(p0.logr_p['2020-02-01': '2020-03-31'].iloc[:, 0],
                 data['rf']['2020-02-01': '2020-03-31'])], axis=0)
```

```
table.index = ('p1', 'p0')
table

# %%

# data for research on "bond effect to portfolio"

# p0 and p1 data
data[['rp']] = p1.logr_p
data['lev'] = p1.leverage
# ratio of bond, not equity, because we care bond
data['ratio'] = p1.ratio['rb']
data['rp64'] = p0.logr_p

# output
# data.to_csv(output_path / "dataset3-effect-daily.csv")

# data for group
data_for_group = data.copy(deep=True)
data_for_group['group_date'] = p1._RRP__last_reset_date
data_for_group = data_for_group[data_for_group['group_date'] != '2019-06-20']

# 1. for each reset period
#   get the informations that used to constract RPP of this period
#   get the performance of RPP of this period
data_group = pd.DataFrame()

# informations (independent variables)
# return and volatility
data_group[['rv_rs', 'rv_rb', 'rolling_rs', 'rolling_rb']] = (
    data_for_group[['rv_rs', 'rv_rb', 'rolling_rs',
                    'rolling_rb', 'group_date']]
    .groupby(by='group_date').first())
# correlation
data_group['corr'] = (
    corr_rolling.swaplevel().loc['rs', 'rb'][p1.reset_date])
# sharpe ratio (risk-free rate not include)
data_group['sr_rs'] = (data_group['rolling_rs'] / data_group['rv_rs'])
data_group['sr_rb'] = (data_group['rolling_rb'] / data_group['rv_rb'])
# target volatility
data_group['target_rv'] = t_target_risk

# 2. performance in reset period (dependent variables)
# yearly return
data_group['rp'] = (
    data_for_group[['rp', 'group_date']].groupby(by='group_date').mean()
    * days_of_year)
data_group[['rs_p', 'rb_p']] = (
    data_for_group[['rs', 'rb', 'group_date']].groupby(by='group_date').mean()
    * days_of_year)
# leverage and ratio of bond
data_group[['lev', 'ratio']] = (
    data_for_group[['lev', 'ratio', 'group_date']].
    groupby(by='group_date').first())
```

```
# volatilities
reset_days = days_of_year*1
data_group['rv_rp'] = (
    data_for_group[['rp', 'group_date']].groupby(by='group_date')
    .agg(lambda x: np.sqrt((x**2).sum()) * np.sqrt(days_of_year/reset_days)))
data_group[['rv_rs_p', 'rv_rb_p']] = (
    np.sqrt((pd.concat([(data_for_group[['rs', 'rb']].shift(-1))**2,
                        data_for_group['group_date']], axis=1))
            .groupby(by='group_date').sum()) *
    np.sqrt(days_of_year/reset_days))
# correlation
data_group['corr_p'] = (
    data_for_group[['rs', 'rb', 'group_date']].groupby(by='group_date')
    .corr().swaplevel().loc['rs', 'rb'])
# risk contributions
data_group['rc_rs'] = (
    (1-data_group['ratio']) * data_group['lev']**2 *
    ((1-data_group['ratio'])*data_group['rv_rs_p']**2 +
     data_group['ratio']*data_group['rv_rs_p'] *
     data_group['rv_rb_p']*data_group['corr_p']) /
    data_group['rv_rp'])
data_group['rc_rb'] = (
    data_group['ratio'] * data_group['lev']**2 *
    (data_group['ratio']*data_group['rv_rb_p']**2 +
     (1-data_group['ratio'])*data_group['rv_rs_p'] *
     data_group['rv_rb_p']*data_group['corr_p']) /
    data_group['rv_rp'])
# sharpe ratio (risk-free rate not include)
data_group['sr_rp'] = (data_group['rp'] / data_group['rv_rp'])
# VAR and ES
data_group['var_rp'] = (
    data_for_group[['rp', 'group_date']].groupby(by='group_date')
    .agg(var))
data_group['es_rp'] = (
    data_for_group[['rp', 'group_date']].groupby(by='group_date')
    .agg(es))
# max drawdown
data_group['mdd_rp'] = (
    data_for_group[['rp', 'group_date']].groupby(by='group_date')
    .agg(maxdrawdown))

# output
data_group.to_csv(output_path / "dataset3-effect-group.csv")

# %%
```

## RPP Construction and Performance (Monthly data)

```
# %%
import pandas as pd
```

```
import numpy as np
from pathlib import Path
import matplotlib
import matplotlib.pyplot as plt
from tool9 import (RRP, period, performance2, portfolio_risk, drawdown)

# path
data_path = Path('raw-data/')
output_path = Path('output/')
graphics_path = Path('../LaTeX_thesis/graphics/')
# data
data_tbond_return = pd.read_excel(data_path / "PredictorData2018.xlsx",
                                  sheet_name="Monthly")
data_tbond_return['Date'] = pd.to_datetime(data_tbond_return['yyyymm'],
                                           format='%Y%m',
                                           errors='coerce').dropna()
data_tbond_return.set_index('Date', drop=True, inplace=True)
# log return
data_tbond_return['rs'] = np.log(data_tbond_return['Index']).diff()
data_tbond_return['rb'] = np.log(1+data_tbond_return['ltr'])
data_tbond_return['rf'] = np.log(1+data_tbond_return['Rfree'])
# data start from the date when ltr is not null
data_tbond_return = data_tbond_return['1926-01-01':]
# copy to data
data = data_tbond_return[['rs', 'rb', 'rf']].copy(deep=True)

# period
[start, end] = period('rs', data=data)

# %%
months_of_year = 12

# correlation between rs and rb
corr_count = months_of_year
corr_rolling = data[['rs', 'rb']].rolling(corr_count).corr().shift(2)
# because of the rolling, we have NaN values, delete the missing rows
corr_rolling = corr_rolling[corr_rolling['rs'].notnull()]
corr_start = corr_rolling[:1].index[0][0]

# get realized volatility
rv_count = months_of_year
data[['rv_rs', 'rv_rb']] = (
    np.sqrt((data[['rs', 'rb']]**2).rolling(rv_count).sum()) *
    np.sqrt(months_of_year/rv_count)
).shift(1)

# because of the rolling, we have NaN values, delete the missing rows
data = data[data['rv_rs'].notnull()]
data_start = data[:1].index[0]

# from the same start point
data_start = max(corr_start, data_start)
corr_rolling = corr_rolling[data_start:]
data = data[data_start:]
```

```
# %%
# 60/40 portfolio
p0 = RRP(
    ratio_fixed=[0.6, 0.4],
    logr=data[['rs', 'rb']],
    corr=corr_rolling[['rs', 'rb']],
    # first_reset_date=None,
    reset_months=12,
    get_actual=False
)

# the correlation and risk in the whole period
data_corr = data[['rs', 'rb']].corr()
data_vol = np.sqrt((data[['rs', 'rb']]**2).sum()) * \
    np.sqrt(months_of_year/len(data))
data_ratio = [0.6, 0.4]
# target risk for RPP
t_target_risk = portfolio_risk(data_corr=data_corr,
                               data_vol=data_vol, data_ratio=data_ratio)
print('target_risk is:')
print(t_target_risk)

p1 = RRP(
    logr=data[['rs', 'rb']],
    risk=data[['rv_rs', 'rv_rb']],
    corr=corr_rolling[['rs', 'rb']],
    reset_months=12,
    target_risk=t_target_risk,
    leverage_limit=6,
    get_actual=True
)

# %%
# %%
# # plot
fig, axes = plt.subplots(nrows=5, ncols=1, figsize=(12, 12), sharex=True)
tmp = pd.concat([(p1.logr_p['portfolio']['1927-01-01':'1962-07-01']-data['rf']
['1927-01-01':'1962-07-01']).cumsum(),
                 (p0.logr_p['portfolio']['1927-01-01':'1962-07-01']-data['rf']
['1927-01-01':'1962-07-01']).cumsum()], axis=1)
tmp.columns = ['RPP', '60/40']
tmp.plot(ax=axes[0], linewidth=0.5)
axes[0].set_title('Cumulative excess log returns')
axes[0].set_xlim(start, end)
y1, y2 = axes[0].get_ylim()
axes[0].fill_between(['1927-01', '1927-11'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['1929-08', '1933-03'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['1937-05', '1938-06'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['1945-02', '1945-10'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
```

```
axes[0].fill_between(['1948-11', '1949-10'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['1953-07', '1958-04'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['1960-04', '1961-02'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')

tmp = p1.logr_p['1927-01-01':'1962-07-01'].rolling(
    12).sum()-p0.logr_p['1927-01-01':'1962-07-01'].rolling(12).sum()
tmp.columns = ['RPP minus 60/40']
tmp.plot(ax=axes[1], linewidth=0.5)
axes[1].set_title('1-year rolling log returns difference')
y1, y2 = axes[1].get_ylim()
axes[1].fill_between(['1927-01', '1927-11'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[1].fill_between(['1929-08', '1933-03'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[1].fill_between(['1937-05', '1938-06'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[1].fill_between(['1945-02', '1945-10'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[1].fill_between(['1948-11', '1949-10'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[1].fill_between(['1953-07', '1958-04'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[1].fill_between(['1960-04', '1961-02'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')

p1.ratio['rb']['1927-01-01':'1962-07-01'].plot(ax=axes[2],
linewidth=0.5,label='theoretical bond weight')
p1.ratio_actual['rb'].plot(ax=axes[2], linewidth=0.5,
                           label='actual bond weight')
axes[2].set_ylim(0.4, 1)
axes[2].set_title('weight of bond in RPP')
axes[2].legend()

tmp = drawdown(p1.logr_p['1927-01-01':'1962-07-01']) - \
    drawdown(p0.logr_p['1927-01-01':'1962-07-01'])
tmp.columns = ['RPP minus 60/40']
tmp.plot(ax=axes[3], linewidth=0.5)
y1, y2 = axes[3].get_ylim()
axes[3].set_title('portfolio drawdown difference')
axes[3].fill_between(['1927-01', '1927-11'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[3].fill_between(['1929-08', '1933-03'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[3].fill_between(['1937-05', '1938-06'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[3].fill_between(['1945-02', '1945-10'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[3].fill_between(['1948-11', '1949-10'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[3].fill_between(['1953-07', '1958-04'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
```

```
axes[3].fill_between(['1960-04', '1961-02'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')

p1.leverage['1927-01-01':'1962-07-01'].plot(ax=axes[4], linewidth=0.5)
axes[4].set_ylim(0, 8)
axes[4].set_title('leverage')
fig.tight_layout()
fig.savefig(graphics_path / 'month portfolio performance')

# %%
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(12, 12), sharex=True)

tmp = pd.concat([(p1.logr_p['portfolio']['1927-01-01':'1962-07-01']-data['rf']
['1927-01-01':'1962-07-01']).cumsum(),
                 (p0.logr_p['portfolio']['1927-01-01':'1962-07-01']-data['rf']
['1927-01-01':'1962-07-01']).cumsum()], axis=1)
tmp.columns = ['RPP', '60/40']
tmp.plot(ax=axes[0], linewidth=0.5)
axes[0].set_title('Cumulative excess log returns')
axes[0].set_xlim(start, end)
y1, y2 = axes[0].get_ylim()
axes[0].fill_between(['1927-01', '1941-10'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[0].fill_between(['1941-11', '1962-07'], y1=y1,
                     y2=y2, alpha=.15, color='tab:green')

tmp = pd.concat([drawdown(p1.logr_p['1927-01-01':'1962-07-01']),
                 drawdown(p0.logr_p['1927-01-01':'1962-07-01'])], axis=1)
tmp.columns = ['RPP', '60/40']
tmp.plot(ax=axes[1], linewidth=0.5)
y1, y2 = axes[1].get_ylim()
axes[1].set_title('portfolio drawdown')
axes[1].fill_between(['1927-01-01', '1941-10-31'], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[1].fill_between(['1941-11-01', '1962-07-30'], y1=y1,
                     y2=y2, alpha=.15, color='tab:green')
fig.tight_layout()
fig.savefig(graphics_path / 'month portfolio performance2')

# %% rising and falling interest rate
pd.set_option('display.precision', 4)
table = pd.concat([
    performance2(p0.logr_p['1927-01-01': '1962-07-01'].iloc[:, 0],
                 data['rf']['1927-01-01': '1962-07-01']),
    performance2(p1.logr_p['1927-01-01': '1962-07-01'].iloc[:, 0],
                 data['rf']['1927-01-01': '1962-07-01'])], axis=0)
table.index = ('p0', 'p1')
table

table = pd.concat([
    performance2(p0.logr_p['1927-01-01': '1941-10-01'].iloc[:, 0],
                 data['rf']['1927-01-01': '1941-10-01']),
    performance2(p1.logr_p['1927-01-01': '1941-10-01'].iloc[:, 0],
                 data['rf']['1927-01-01': '1941-10-01'])], axis=0)
```

```
table.index = ('p0', 'p1')
table

table = pd.concat([
    performance2(p0.logr_p['1941-11-01': '1962-07-01'].iloc[:, 0],
                 data['rf']['1941-11-01': '1962-07-01']),
    performance2(p1.logr_p['1941-11-01': '1962-07-01'].iloc[:, 0],
                 data['rf']['1941-11-01': '1962-07-01'])], axis=0)
table.index = ('p0', 'p1')
table

pd.reset_option('display.precision')

# %% recession
# ['1927-01-01': '1927-11-01']
# ['1929-08-01': '1933-03-01']
# ['1937-05-01': '1938-06-01']
# ['1945-02-01': '1945-10-01']
# ['1948-11-01': '1949-10-01']
# ['1953-07-01': '1958-04-01']
# ['1960-04-01': '1961-02-01']
pd.set_option('display.precision', 4)
table = pd.concat([
    performance2(p1.logr_p['1927-01-01': '1927-11-01'].iloc[:, 0],
                 data['rf']['1927-01-01': '1927-11-01']),
    performance2(p0.logr_p['1927-01-01': '1927-11-01'].iloc[:, 0],
                 data['rf']['1927-01-01': '1927-11-01'])], axis=0)
table.index = ('p1', 'p0')
table

pd.set_option('display.precision', 4)
table = pd.concat([
    performance2(p1.logr_p['1929-08-01': '1933-03-01'].iloc[:, 0],
                 data['rf']['1929-08-01': '1933-03-01']),
    performance2(p0.logr_p['1929-08-01': '1933-03-01'].iloc[:, 0],
                 data['rf']['1929-08-01': '1933-03-01'])], axis=0)
table.index = ('p1', 'p0')
table

pd.set_option('display.precision', 4)
table = pd.concat([
    performance2(p1.logr_p['1937-05-01': '1938-06-01'].iloc[:, 0],
                 data['rf']['1937-05-01': '1938-06-01']),
    performance2(p0.logr_p['1937-05-01': '1938-06-01'].iloc[:, 0],
                 data['rf']['1937-05-01': '1938-06-01'])], axis=0)
table.index = ('p1', 'p0')
table

pd.set_option('display.precision', 4)
table = pd.concat([
    performance2(p1.logr_p['1945-02-01': '1945-10-01'].iloc[:, 0],
                 data['rf']['1945-02-01': '1945-10-01']),
    performance2(p0.logr_p['1945-02-01': '1945-10-01'].iloc[:, 0],
                 data['rf']['1945-02-01': '1945-10-01'])], axis=0)
```

```
table.index = ('p1', 'p0')
table

pd.set_option('display.precision', 4)
table = pd.concat([
    performance2(p1.logr_p['1948-11-01': '1949-10-01'].iloc[:, 0],
                 data['rf']['1948-11-01': '1949-10-01']),
    performance2(p0.logr_p['1948-11-01': '1949-10-01'].iloc[:, 0],
                 data['rf']['1948-11-01': '1949-10-01'])], axis=0)
table.index = ('p1', 'p0')
table

pd.set_option('display.precision', 4)
table = pd.concat([
    performance2(p1.logr_p['1953-07-01': '1958-04-01'].iloc[:, 0],
                 data['rf']['1953-07-01': '1958-04-01']),
    performance2(p0.logr_p['1953-07-01': '1958-04-01'].iloc[:, 0],
                 data['rf']['1953-07-01': '1958-04-01'])], axis=0)
table.index = ('p1', 'p0')
table

pd.set_option('display.precision', 4)
table = pd.concat([
    performance2(p1.logr_p['1960-04-01': '1961-02-01'].iloc[:, 0],
                 data['rf']['1960-04-01': '1961-02-01']),
    performance2(p0.logr_p['1960-04-01': '1961-02-01'].iloc[:, 0],
                 data['rf']['1960-04-01': '1961-02-01'])], axis=0)
table.index = ('p1', 'p0')
table

# %%
```

## Volatility Variation Effect - Theoretical Part

```
# %%
import pandas as pd
import numpy as np
from pathlib import Path
import matplotlib
import matplotlib.pyplot as plt

# global settings
output_path = Path('output/')
data_path = Path('raw-data/')
graphics_path = Path('../LaTeX_thesis/graphics/')

# %%
# ERC - theoretical analysis
plt.rcParams.update({'font.size': 12})
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 5))
```

```
p = 0.6
x = [t*0.1 for t in range(2, 20, 1)]
y = [(t+p)/(1/t+p) for t in x]
axes[0].plot(x, y, linewidth=0.8, label='correlation= 0.6')
p = 0.2
x = [t*0.1 for t in range(2, 20, 1)]
y = [(t+p)/(1/t+p) for t in x]
axes[0].plot(x, y, linewidth=0.8, label='correlation= 0.2')
p = 0
x = [t*0.1 for t in range(2, 20, 1)]
y = [(t+p)/(1/t+p) for t in x]
axes[0].plot(x, y, linewidth=0.8, label='correlation= 0.0')
p = -0.1999
x = [t*0.1 for t in range(2, 20, 1)]
y = [(t+p)/(1/t+p) for t in x]
axes[0].plot(x, y, linewidth=0.8, label='correlation=-0.2')
p = -0.4999
x = [t*0.1 for t in range(5, 20, 1)]
y = [(t+p)/(1/t+p) for t in x]
axes[0].plot(x, y, linewidth=0.8, label='correlation=-0.5')
axes[0].set_ylim(0, 1)
axes[0].set_xlim(0.2, 1)
axes[0].set_ylabel('actual ratio of risk contribution')
axes[0].set_xlabel('κ')
axes[0].legend()

p = 0.6
x = [t*0.1 for t in range(2, 50, 1)]
y = [(t+p)/(1/t+p) for t in x]
axes[1].plot(x, y, linewidth=0.8)
p = 0.2
x = [t*0.1 for t in range(2, 50, 1)]
y = [(t+p)/(1/t+p) for t in x]
axes[1].plot(x, y, linewidth=0.8)
p = 0
x = [t*0.1 for t in range(2, 50, 1)]
y = [(t+p)/(1/t+p) for t in x]
axes[1].plot(x, y, linewidth=0.8)
p = -0.1999
x = [t*0.1 for t in range(2, 50, 1)]
y = [(t+p)/(1/t+p) for t in x]
axes[1].plot(x, y, linewidth=0.8)
p = -0.4999
x = [t*0.1 for t in range(5, 20, 1)]
y = [(t+p)/(1/t+p) for t in x]
axes[1].plot(x, y, linewidth=0.8)
axes[1].set_ylim(1, 15)
axes[1].set_xlim(1, 3)
axes[1].set_ylabel('actual ratio of risk contribution')
axes[1].set_xlabel('κ')

# fig.savefig(graphics_path / 'ERC-theoretical')

# %%
```

```
# lambda - theoretical analysis
plt.rcParams.update({'font.size': 12})
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 8),
                         sharex=True, sharey=True)

p = 0.6
ka = [t*0.1 for t in range(10, 34, 1)]
la = [(t+1/t)/2 for t in ka]
vol = [np.sqrt((t+1/t+2*p)/(2*(1+p))) for t in ka]
rc1 = [(t+p)/(2*(1+p))/np.sqrt((t+1/t+2*p)/(2*(1+p))) for t in ka]
axes[0][0].fill_between(la, vol,  y2=rc1, linewidth=0.8,
                        color='tab:green', alpha=0.15)
axes[0][0].fill_between(la, rc1, y2=0, linewidth=0.8,
                        color='tab:blue', alpha=0.15)
axes[0][0].set_ylim(0, 1.5)
axes[0][0].set_xlim(1, 1.8)
axes[0][0].set_title('ρ= 0.6')
axes[0][0].set_ylabel('Risk contribution')
axes[0][0].text(1.1, 0.8, 'Risk contribution of asset 2',
                rotation=8, rotation_mode='anchor')
axes[0][0].text(1.2, 0.4, 'Risk contribution of asset 1')

p = 0.3
ka = [t*0.1 for t in range(10, 34, 1)]
la = [(t+1/t)/2 for t in ka]
vol = [np.sqrt((t+1/t+2*p)/(2*(1+p))) for t in ka]
rc1 = [(t+p)/(2*(1+p))/np.sqrt((t+1/t+2*p)/(2*(1+p))) for t in ka]
axes[0][1].fill_between(la, vol,  y2=rc1, linewidth=0.8,
                        color='tab:green', alpha=0.15)
axes[0][1].fill_between(la, rc1, y2=0, linewidth=0.8,
                        color='tab:blue', alpha=0.15)
axes[0][1].set_title('ρ= 0.3')

p = 0
ka = [t*0.1 for t in range(10, 34, 1)]
la = [(t+1/t)/2 for t in ka]
vol = [np.sqrt((t+1/t+2*p)/(2*(1+p))) for t in ka]
rc1 = [(t+p)/(2*(1+p))/np.sqrt((t+1/t+2*p)/(2*(1+p))) for t in ka]
axes[1][0].fill_between(la, vol,  y2=rc1, linewidth=0.8,
                        color='tab:green', alpha=0.15)
axes[1][0].fill_between(la, rc1, y2=0, linewidth=0.8,
                        color='tab:blue', alpha=0.15)
axes[1][0].set_title('ρ= 0.0')
axes[1][0].set_ylabel('Risk contribution')
axes[1][0].set_xlabel('λ')
axes[1][0].text(1.1, 0.93, 'Sum of risk contribution of assets')
axes[1][0].text(1.1, 0.8, 'Also the volatility of portfolio')

p = -0.3
ka = [t*0.1 for t in range(10, 34, 1)]
la = [(t+1/t)/2 for t in ka]
vol = [np.sqrt((t+1/t+2*p)/(2*(1+p))) for t in ka]
rc1 = [(t+p)/(2*(1+p))/np.sqrt((t+1/t+2*p)/(2*(1+p))) for t in ka]
axes[1][1].fill_between(la, vol,  y2=rc1, linewidth=0.8,
```

```
                        color='tab:green', alpha=0.15)
axes[1][1].fill_between(la, rc1, y2=0, linewidth=0.8,
                        color='tab:blue', alpha=0.15)
axes[1][1].set_title('ρ=-0.3')
axes[1][1].set_xlabel('λ')

# fig.savefig(graphics_path / 'lambda-theoretical')

# %%
```

## Volatility Variation Effect - Empirical Part

```
# %%
import pandas as pd
import numpy as np
from pathlib import Path
import matplotlib
import matplotlib.pyplot as plt
import datetime
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.feature_selection import SelectFromModel
import statsmodels.api as sm
from tool9 import describe

# global settings
output_path = Path('output/')
data_path = Path('raw-data/')
graphics_path = Path('../LaTeX_thesis/graphics/')

# read data from csv file
# data = pd.read_csv(output_path / "dataset3-effect-daily.csv",
#                    parse_dates=['Date'], index_col='Date')
data_group = pd.read_csv(output_path / "dataset3-effect-group.csv",
                         parse_dates=['group_date'], index_col='group_date')

# create new variables
data_group['corr_diff'] = data_group['corr_p'] - data_group['corr']
data_group['zero'] = 0
data_group['const'] = 1

# empoirical analysis data
data_group['ks'] = (data_group['rv_rs_p'] / data_group['rv_rs'])
data_group['kb'] = (data_group['rv_rb_p'] / data_group['rv_rb'])
data_group['pi'] = np.sqrt(data_group['ks']*data_group['kb'])
data_group['kappa'] = (data_group['ks']/data_group['kb'])
data_group['lambda'] = (data_group['kappa'] + (1/data_group['kappa']))/2
data_group['sqrt'] = np.sqrt((data_group['lambda']+data_group['corr_p']) /
                             (1+data_group['corr']))

data_group['rv_rp_th'] = (
```

```
    data_group['target_rv']*data_group['pi'] *
    np.sqrt((data_group['lambda']+data_group['corr_p'])/(1+data_group['corr']))
)
data_group['logks'] = np.log(data_group['ks'])
data_group['logkb'] = np.log(data_group['kb'])

# variables namesz
data_group.columns

# %%
pd.set_option('display.precision', 4)
pd.concat([describe('logkb', data_group),
           describe('logks', data_group),
           describe('rc_rb', data_group),
           describe('rc_rs', data_group)], axis=0)
pd.reset_option('display.precision')

# %%
plt.rcParams.update({'font.size': 12})
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(12, 8), sharex=False)

data_group['logks'].plot.kde(bw_method=0.6, ax=axes[0], linewidth=0.9,
                             color='tab:blue',
                             label='log(k1) for S&P 500')
data_group['logkb'].plot.kde(bw_method=0.6, ax=axes[0], linewidth=0.9,
                             color='tab:orange',
                             label='log(k2) for LTR')
axes[0].set_ylabel('Density')
axes[0].set_title('Unconditional distributions of log variation ' +
                  'of volatility of S&P 500 and LTR')
axes[0].legend()
axes[0].set_ylim(-0.006, 1.25)
axes[0].set_xlim(-2, 2.5)

axes[1].fill_between(list(data_group.index.year),
                     list(data_group['rv_rp']),
                     y2=list(data_group['rc_rb']),
                     linewidth=0.4, color='tab:blue', alpha=0.25)
axes[1].fill_between(list(data_group.index.year),
                     list(data_group['rc_rb']),
                     y2=0,
                     linewidth=0.2, color='tab:orange', alpha=0.25)
axes[1].set_xlim(1962, 2018)
axes[1].set_ylabel('Volatility')
axes[1].set_title('Volatility of RPP and risk contribution of S&P 500 and LTR')
axes[1].text(1983, 0.08, 'Risk contribution of S&P 500')
axes[1].text(1980, 0.01, 'Risk contribution of LTR')

fig.savefig(graphics_path / 'ks-kb-rc-empirical')

# %%
plt.rcParams.update({'font.size': 12})
fig, axes = plt.subplots(nrows=4, ncols=1, figsize=(12, 12), sharex=True,
                         gridspec_kw={'height_ratios': [1, 2, 1, 1]})
```

```
fig.subplots_adjust(hspace=0)
axes[0].plot(list(data_group.index.year+0.5), data_group['rv_rs_p'],
             color='tab:blue', label='rv_rs')
axes[0].plot(list(data_group.index.year+0.5), data_group['rv_rb_p'],
             color='tab:orange', label='rv_rb')
axes[0].set_ylim(0, 0.5)
axes[0].legend()
axes[0].grid(which='major')
axes[0].minorticks_on()
axes[0].grid(which='minor', axis='x', linestyle=':', linewidth='0.5',
             color='black')
axes[0].set_xlim(1962, 1991)
axes[1].plot(list(data_group.index.year),
             list(data_group['zero']), color='black')
axes[1].bar(x=list(data_group.index.year+0.3),
            height=list(np.log(data_group['ks'])),
            width=0.2, color='tab:blue', label='log(k1)')
axes[1].bar(x=list(data_group.index.year+0.5),
            height=list(np.log(data_group['kb'])),
            width=0.2, color='tab:orange', label='log(k2)')
axes[1].bar(x=list(data_group.index.year+0.7),
            height=list(np.log(data_group['kappa'])),
            width=0.2, color='green', label='log(κ)')
axes[1].plot(list(data_group.index.year+0.45),
             list(np.log(data_group['pi'])), color='purple', linewidth=2,
             label='log(π)')
axes[1].text(1965, -1.2, 'λ=1.63', color='tab:green')
axes[1].text(1972, 0.9, 'λ=1.38', color='tab:green')
axes[1].text(1977, 0.85, 'λ=1.31', color='tab:green')
axes[1].text(1979, -1.4, 'λ=1.83', color='tab:green')
axes[1].text(1987, 0.8, 'λ=1.24', color='tab:green')
axes[1].set_ylim(-1.55, 1.55)
axes[1].legend()
axes[1].grid(which='major')
axes[1].minorticks_on()
axes[1].grid(which='minor', axis='x', linestyle=':', linewidth='0.5',
             color='black')
axes[2].plot(list(data_group.index.year),
             list(data_group['zero']+1), color='tab:gray',
             linewidth=0.3)
axes[2].plot(list(data_group.index.year+0.45),
             list(data_group['sqrt']),
             color='tab:olive', linewidth=2)
ax = axes[2].twinx()
ax.bar(x=list(data_group.index.year+0.5),
       height=list((data_group['corr_p'] - data_group['corr']) *
                   (data_group['corr_p'] > data_group['corr'])),
       width=0.4, color='tab:cyan', alpha=0.5, label='increase of correlation')
ax.bar(x=list(data_group.index.year+0.5),
       height=list((data_group['corr_p'] - data_group['corr']) *
                   (data_group['corr_p'] < data_group['corr'])),
       width=0.4, color='tomato', alpha=0.5, label='decrease of correlation')
ax.set_ylim(-0.45, 0.9)
ax.legend()
```

```
axes[2].set_ylim(0.6, 1.8)
y1, y2 = axes[2].get_ylim()
axes[2].fill_between([1962, 1967.9], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[2].fill_between([1967.9, 1998.5], y1=y1,
                     y2=y2, alpha=.15, color='tab:green')
axes[2].fill_between([1998.5, 2020], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[2].grid(which='major')
axes[2].minorticks_on()
axes[2].grid(which='minor', axis='x', linestyle=':', linewidth='0.5',
             color='black')
axes[3].plot(list(data_group.index.year+0.45),
             list(data_group['target_rv']),
             color='lime', linewidth=1, linestyle='--')
axes[3].plot(list(data_group.index.year+0.45),
             list(data_group['rv_rp_th']),
             color='tab:red', linewidth=2,
             label='theoretical volatility of portfolio')
axes[3].plot(list(data_group.index.year+0.45),
             list(data_group['rv_rp']), alpha=0.8,
             color='deepskyblue', linewidth=0.8,
             label='actual volatility of portfolio')
axes[3].legend()
axes[3].grid(which='major')
axes[3].minorticks_on()
axes[3].grid(which='minor', axis='x', linestyle=':', linewidth='0.5',
             color='black')

# fig.savefig(graphics_path / 'risk-empirical-1')

# %%
plt.rcParams.update({'font.size': 12})
fig, axes = plt.subplots(nrows=4, ncols=1, figsize=(12, 12), sharex=True,
                         gridspec_kw={'height_ratios': [1, 2, 1, 1]})
fig.subplots_adjust(hspace=0)
axes[0].plot(list(data_group.index.year+0.5), data_group['rv_rs_p'],
             color='tab:blue', label='rv_rs')
axes[0].plot(list(data_group.index.year+0.5), data_group['rv_rb_p'],
             color='tab:orange', label='rv_rb')
axes[0].set_ylim(0, 0.5)
axes[0].legend()
axes[0].grid(which='major')
axes[0].minorticks_on()
axes[0].grid(which='minor', axis='x', linestyle=':', linewidth='0.5',
             color='black')
axes[0].set_xlim(1991, 2019)
axes[1].plot(list(data_group.index.year),
             list(data_group['zero']), color='black')
axes[1].bar(x=list(data_group.index.year+0.3),
            height=list(np.log(data_group['ks'])),
            width=0.2, color='tab:blue', label='log(k1)')
axes[1].bar(x=list(data_group.index.year+0.5),
            height=list(np.log(data_group['kb'])),
```

```
            width=0.2, color='tab:orange', label='log(k2)')
axes[1].bar(x=list(data_group.index.year+0.7),
            height=list(np.log(data_group['kappa'])),
            width=0.2, color='green', label='log(κ)')
axes[1].plot(list(data_group.index.year+0.45),
             list(np.log(data_group['pi'])), color='purple', linewidth=2,
             label='log(π)')
axes[1].text(2003, -0.8, 'λ=1.21', color='tab:green')
axes[1].text(2015, 0.55, 'λ=1.15', color='tab:green')
axes[1].text(2017, 0.65, 'λ=1.19', color='tab:green')
axes[1].set_ylim(-1.55, 1.55)
axes[1].legend()
axes[1].grid(which='major')
axes[1].minorticks_on()
axes[1].grid(which='minor', axis='x', linestyle=':', linewidth='0.5',
             color='black')
axes[2].plot(list(data_group.index.year),
             list(data_group['zero']+1), color='tab:gray',
             linewidth=0.3)
axes[2].plot(list(data_group.index.year+0.45),
             list(data_group['sqrt']),
             color='tab:olive', linewidth=2)
ax = axes[2].twinx()
ax.bar(x=list(data_group.index.year+0.5),
       height=list((data_group['corr_p'] - data_group['corr']) *
                   (data_group['corr_p'] > data_group['corr'])),
       width=0.4, color='tab:cyan', alpha=0.5, label='increase of correlation')
ax.bar(x=list(data_group.index.year+0.5),
       height=list((data_group['corr_p'] - data_group['corr']) *
                   (data_group['corr_p'] < data_group['corr'])),
       width=0.4, color='tomato', alpha=0.5, label='decrease of correlation')
ax.set_ylim(-0.45, 0.9)
ax.legend()
axes[2].set_ylim(0.6, 1.8)
y1, y2 = axes[2].get_ylim()
axes[2].fill_between([1962, 1967.9], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[2].fill_between([1967.9, 1998.5], y1=y1,
                     y2=y2, alpha=.15, color='tab:green')
axes[2].fill_between([1998.5, 2020], y1=y1,
                     y2=y2, alpha=.15, color='tab:gray')
axes[2].grid(which='major')
axes[2].minorticks_on()
axes[2].grid(which='minor', axis='x', linestyle=':', linewidth='0.5',
             color='black')
axes[3].plot(list(data_group.index.year+0.45),
             list(data_group['target_rv']),
             color='lime', linewidth=1, linestyle='--')
axes[3].plot(list(data_group.index.year+0.45),
             list(data_group['rv_rp_th']),
             color='tab:red', linewidth=2, label='volatility of portfolio')
axes[3].plot(list(data_group.index.year+0.45),
             list(data_group['rv_rp']), alpha=0.8,
             color='deepskyblue', linewidth=0.8,
```

```
            label='actual volatility of portfolio')
axes[3].legend()
axes[3].grid(which='major')
axes[3].minorticks_on()
axes[3].grid(which='minor', axis='x', linestyle=':', linewidth='0.5',
            color='black')

# fig.savefig(graphics_path / 'risk-empirical-2')

# %%
```