data_from_wrds

February 5, 2024

1 Working with CRSP Data from WRDS

1.0.1 Import the relevant packages

[ ]:
```
import wrds
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import scipy.optimize as sco

plt.style.use('fivethirtyeight')
np.random.seed(777)

%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

1.1 Provided Code

1.1.1 Create connection to the WRDS servers and download CRSP Data

Create connection

[ ]:
```
conn = wrds.Connection()
```

```
WRDS recommends setting up a .pgpass file.
You can create this file yourself at any time with the create_pgpass_file()
function.
Loading library list…
Done
```

List CRSP Files and Tables

[ ]:
```
conn.list_tables(library='crsp')
```

[ ]:
```
['acti',
 'asia',
 'asib',
 'asic',
```

```
'asio',
'asix',
'bmdebt',
'bmheader',
'bmpaymts',
'bmquotes',
'bmyield',
'bndprt06',
'bndprt12',
'bxcalind',
'bxdlyind',
'bxmthind',
'bxquotes',
'bxyield',
'cap',
'ccm_lookup',
'ccm_qvards',
'ccmxpf_linktable',
'ccmxpf_lnkhist',
'ccmxpf_lnkrng',
'ccmxpf_lnkused',
'comphead',
'comphist',
'compmaster',
'contact_info',
'crsp_cik_map',
'crsp_daily_data',
'crsp_header',
'crsp_monthly_data',
'crsp_names',
'crsp_portno_map',
'crsp_ziman_daily_index',
'crsp_ziman_monthly_index',
'cs20yr',
'cs5yr',
'cs90d',
'cst_hist',
'daily_nav',
'daily_nav_ret',
'daily_returns',
'dividends',
'dport1',
'dport2',
'dport3',
'dport4',
'dport5',
'dport6',
```

```
'dport7',
'dport8',
'dport9',
'dsbc',
'dsbo',
'dse',
'dse62',
'dse62delist',
'dse62dist',
'dse62exchdates',
'dse62names',
'dse62nasdin',
'dse62shares',
'dseall',
'dseall62',
'dsedelist',
'dsedist',
'dseexchdates',
'dsenames',
'dsenasdin',
'dseshares',
'dsf',
'dsf62',
'dsf62_v2',
'dsf_v2',
'dsfhdr',
'dsfhdr62',
'dsi',
'dsi62',
'dsia',
'dsib',
'dsic',
'dsio',
'dsir',
'dsix',
'dsiy',
'dsp500',
'dsp500_v2',
'dsp500list',
'dsp500list_v2',
'dsp500p',
'dssc',
'dsso',
'eod_cap',
'eod_sector',
'eod_vg',
'erdport1',
```

```
'erdport2',
'erdport3',
'erdport4',
'erdport5',
'erdport6',
'erdport7',
'erdport8',
'erdport9',
'ermport1',
'ermport2',
'ermport3',
'ermport4',
'ermport5',
'fbpri',
'fbyld',
'front_load',
'front_load_det',
'front_load_grp',
'fund_fees',
'fund_flows',
'fund_hdr',
'fund_hdr_hist',
'fund_names',
'fund_style',
'fund_summary',
'fund_summary2',
'fwdask06',
'fwdask12',
'fwdave06',
'fwdave12',
'fwdbid06',
'fwdbid12',
'hldask06',
'hldask12',
'hldave06',
'hldave12',
'hldbid06',
'hldbid12',
'holdings',
'holdings_co_info',
'idx_const_close_pf_v2',
'idx_const_close_v2',
'idx_const_open_pf_v2',
'idx_const_open_proj_v2',
'idx_const_open_v2',
'idx_levels',
'inddlyseriesdata',
```

```
'inddlyseriesdata62',
'inddlyseriesdata_ind',
'index_descriptions',
'index_type_map',
'indfamilyinfohdr',
'indfamilyinfohdr62',
'indfamilyinfohdr_ind',
'indissrebalancesummary_ind',
'indmthseriesdata',
'indmthseriesdata62',
'indmthseriesdata_ind',
'indsecrebalancesummary_ind',
'indseriesinfohdr',
'indseriesinfohdr62',
'indseriesinfohdr_ind',
'mbi',
'mbmdat',
'mbmhdr',
'mbx',
'mbxid',
'mcti',
'metacalendarperiod',
'metacalendarperiod62',
'metacalendarperiod_ind',
'metacolumncoverage',
'metacolumncoverage62',
'metacolumncoverage_ind',
'metacolumninfo',
'metacolumninfo62',
'metacolumninfo_ind',
'metaexchangecalendar',
'metaexchangecalendar62',
'metaexchangecalendar_ind',
'metafileinfo',
'metafileinfo62',
'metafileinfo_ind',
'metaflagcoverage',
'metaflagcoverage62',
'metaflagcoverage_ind',
'metaflaginfo',
'metaflaginfo62',
'metaflaginfo_ind',
'metaflagtype',
'metaflagtype62',
'metaflagtype_ind',
'metaiteminfo',
'metaiteminfo62',
```

```
'metaiteminfo_ind',
'metasiztociz',
'metasiztociz62',
'metasiztociz_ind',
'mfdbname',
'mhista',
'mhistn',
'mhistq',
'monthly_nav',
'monthly_returns',
'monthly_tna',
'monthly_tna_ret_nav',
'mport1',
'mport2',
'mport3',
'mport4',
'mport5',
'mse',
'mse62',
'mse62delist',
'mse62dist',
'mse62exchdates',
'mse62names',
'mse62nasdin',
'mse62shares',
'mseall',
'mseall62',
'msedelist',
'msedist',
'mseexchdates',
'msenames',
'msenasdin',
'mseshares',
'msf',
'msf62',
'msf62_v2',
'msf_v2',
'msfhdr',
'msfhdr62',
'msi',
'msi62',
'msia',
'msib',
'msic',
'msio',
'msir',
'msix',
```

```
'msiy',
'msp500',
'msp500_v2',
'msp500list',
'msp500list_v2',
'msp500p',
'portnomap',
'priask06',
'priask12',
'priave06',
'priave12',
'pribid06',
'pribid12',
'price_type',
'property_type',
'qcti',
'qsia',
'qsib',
'qsic',
'qsio',
'qsix',
'rear_load',
'rear_load_det',
'rear_load_grp',
'rebala',
'rebaln',
'rebalq',
'reit_type',
'riskfree',
's6z_agg_ann',
's6z_agg_ann_legacy',
's6z_agg_mth',
's6z_agg_mth_legacy',
's6z_agg_qtr',
's6z_agg_qtr_legacy',
's6z_del',
's6z_del_legacy',
's6z_dind',
's6z_dind_legacy',
's6z_dis',
's6z_dis_legacy',
's6z_dp_dly',
's6z_dp_dly_legacy',
's6z_ds_dly',
's6z_ds_dly_legacy',
's6z_hdr',
's6z_hdr_legacy',
```

```
's6z_indhdr',
's6z_indhdr_legacy',
's6z_mdel',
's6z_mdel_legacy',
's6z_mind',
's6z_mind_legacy',
's6z_mth',
's6z_mth_legacy',
's6z_nam',
's6z_nam_legacy',
's6z_ndi',
's6z_ndi_legacy',
's6z_shr',
's6z_shr_legacy',
'saz_agg_ann',
'saz_agg_ann_legacy',
'saz_agg_mth',
'saz_agg_mth_legacy',
'saz_agg_qtr',
'saz_agg_qtr_legacy',
'saz_del',
'saz_del_legacy',
'saz_dind',
'saz_dind_legacy',
'saz_dis',
'saz_dis_legacy',
'saz_dp_dly',
'saz_dp_dly_legacy',
'saz_ds_dly',
'saz_ds_dly_legacy',
'saz_hdr',
'saz_hdr_legacy',
'saz_indhdr',
'saz_indhdr_legacy',
'saz_mdel',
'saz_mdel_legacy',
'saz_mind',
'saz_mind_legacy',
'saz_mth',
'saz_mth_legacy',
'saz_nam',
'saz_nam_legacy',
'saz_ndi',
'saz_ndi_legacy',
'saz_shr',
'saz_shr_legacy',
'sechead',
```

```
'sechist',
'sector',
'sfz_dind',
'sfz_indhdr',
'sfz_mbr',
'sfz_mind',
'sfz_portd',
'sfz_portm',
'sfz_rb',
'stkannsecuritydata',
'stkannsecuritydata62',
'stkdelists',
'stkdelists62',
'stkdistributions',
'stkdistributions62',
'stkdlysecuritydata',
'stkdlysecuritydata62',
'stkdlysecurityprimarydata',
'stkdlysecurityprimarydata62',
'stkindissuerstatistics_ind',
'stkindmembership_ind',
'stkindsecuritystatistics_ind',
'stkissuerinfohdr',
'stkissuerinfohdr62',
'stkissuerinfohist',
'stkissuerinfohist62',
'stkmthfloatshares',
'stkmthfloatshares62',
'stkmthsecuritydata',
'stkmthsecuritydata62',
'stkqtrsecuritydata',
'stkqtrsecuritydata62',
'stksecurityinfohdr',
'stksecurityinfohdr62',
'stksecurityinfohist',
'stksecurityinfohist62',
'stkshares',
'stkshares62',
'stock_qvards',
'stocknames',
'stocknames62',
'stocknames62_v2',
'stocknames_v2',
'sub_property_type',
'tfz_dly',
'tfz_dly_cd',
'tfz_dly_cpi',
```

```
 'tfz_dly_ft',
 'tfz_dly_rf2',
 'tfz_dly_ts2',
 'tfz_idx',
 'tfz_iss',
 'tfz_mast',
 'tfz_mth',
 'tfz_mth_bp',
 'tfz_mth_cd',
 'tfz_mth_cpi',
 'tfz_mth_fb',
 'tfz_mth_ft',
 'tfz_mth_rf',
 'tfz_mth_rf2',
 'tfz_mth_ts',
 'tfz_mth_ts2',
 'tfz_pay',
 'vg',
 'wrds_dailyindexret62_query',
 'wrds_dailyindexret_query',
 'wrds_dsf62v2_query',
 'wrds_dsfv2_query',
 'wrds_inddlytranspose_query',
 'wrds_indmthtranspose_query',
 'wrds_monthlyindexret62_query',
 'wrds_monthlyindexret_query',
 'wrds_msf62v2_query',
 'wrds_msfv2_query',
 'wrds_names62_query',
 'wrds_names_query',
 'yldask06',
 'yldask12',
 'yldave06',
 'yldave12',
 'yldbid06',
 'yldbid12',
 'ziman_reit_info',
 'zr_hdrnames']
```

[ ]:
```
conn.describe_table('crsp', 'dsf')
```

```
Approximately 105231600 rows in crsp.dsf.
```

[ ]:

	name	nullable	type	comment
0	cusip	True	VARCHAR(8)	None
1	permno	True	INTEGER	None
2	permco	True	INTEGER	None
3	issuno	True	INTEGER	None

4	hexcd	True	SMALLINT	None
5	hsiccd	True	INTEGER	None
6	date	True	DATE	None
7	bidlo	True	NUMERIC(11, 5)	None
8	askhi	True	NUMERIC(11, 5)	None
9	prc	True	NUMERIC(11, 5)	None
10	vol	True	NUMERIC(10, 0)	None
11	ret	True	NUMERIC(10, 6)	None
12	bid	True	NUMERIC(11, 5)	None
13	ask	True	NUMERIC(11, 5)	None
14	shrout	True	DOUBLE PRECISION	None
15	cfacpr	True	DOUBLE PRECISION	None
16	cfacshr	True	DOUBLE PRECISION	None
17	openprc	True	NUMERIC(11, 5)	None
18	numtrd	True	INTEGER	None
19	retx	True	NUMERIC(10, 6)	None

[ ]:
```
conn.describe_table('crspm', 'dsfhdr')
```

Approximately 37776 rows in crspm.dsfhdr.

[ ]:

	name	nullable	type	comment
0	permno	True	INTEGER	None
1	permco	True	INTEGER	None
2	hshrcd	True	SMALLINT	None
3	dlstcd	True	SMALLINT	None
4	hcusip	True	VARCHAR(8)	None
5	htick	True	VARCHAR(8)	None
6	hcomnam	True	VARCHAR(35)	None
7	htsymbol	True	VARCHAR(10)	None
8	hnaics	True	VARCHAR(7)	None
9	hprimexc	True	VARCHAR(1)	None
10	htrdstat	True	VARCHAR(1)	None
11	hsecstat	True	VARCHAR(1)	None
12	cusip	True	VARCHAR(8)	None
13	compno	True	DOUBLE PRECISION	None
14	issuno	True	DOUBLE PRECISION	None
15	hexcd	True	DOUBLE PRECISION	None
16	hsiccd	True	DOUBLE PRECISION	None
17	numnam	True	DOUBLE PRECISION	None
18	numdis	True	DOUBLE PRECISION	None
19	numshr	True	DOUBLE PRECISION	None
20	numdel	True	DOUBLE PRECISION	None
21	numndi	True	DOUBLE PRECISION	None
22	begdat	True	DATE	None
23	enddat	True	DATE	None
24	begprc	True	DATE	None
25	endprc	True	DATE	None

```
26    begret     True                DATE    None
27    endret     True                DATE    None
28    begrtx     True                DATE    None
29    endrtx     True                DATE    None
30  begbidlo     True                DATE    None
31  endbidlo     True                DATE    None
32  begaskhi     True                DATE    None
33  endaskhi     True                DATE    None
34    begvol     True                DATE    None
35    endvol     True                DATE    None
36    begbid     True                DATE    None
37    endbid     True                DATE    None
38    begask     True                DATE    None
39    endask     True                DATE    None
40    begopr     True                DATE    None
41    endopr     True                DATE    None
42    hsicmg     True    DOUBLE PRECISION    None
43    hsicig     True    DOUBLE PRECISION    None
```

[ ]:
```
conn.describe_table('crsp', 'dsfhdr')
```

```
Approximately 37776 rows in crsp.dsfhdr.
```

[ ]:
```
        name  nullable              type  comment
0     permno      True           INTEGER     None
1     permco      True           INTEGER     None
2     hshrcd      True          SMALLINT     None
3     dlstcd      True          SMALLINT     None
4     hcusip      True        VARCHAR(8)     None
5      htick      True        VARCHAR(8)     None
6    hcomnam      True       VARCHAR(35)     None
7   htsymbol      True       VARCHAR(10)     None
8     hnaics      True        VARCHAR(7)     None
9   hprimexc      True        VARCHAR(1)     None
10  htrdstat      True        VARCHAR(1)     None
11  hsecstat      True        VARCHAR(1)     None
12     cusip      True        VARCHAR(8)     None
13    compno      True  DOUBLE PRECISION     None
14    issuno      True  DOUBLE PRECISION     None
15     hexcd      True  DOUBLE PRECISION     None
16    hsiccd      True  DOUBLE PRECISION     None
17    numnam      True  DOUBLE PRECISION     None
18    numdis      True  DOUBLE PRECISION     None
19    numshr      True  DOUBLE PRECISION     None
20    numdel      True  DOUBLE PRECISION     None
21    numndi      True  DOUBLE PRECISION     None
22    begdat      True              DATE     None
23    enddat      True              DATE     None
```

24	begprc	True	DATE	None
25	endprc	True	DATE	None
26	begret	True	DATE	None
27	endret	True	DATE	None
28	begrtx	True	DATE	None
29	endrtx	True	DATE	None
30	begbidlo	True	DATE	None
31	endbidlo	True	DATE	None
32	begaskhi	True	DATE	None
33	endaskhi	True	DATE	None
34	begvol	True	DATE	None
35	endvol	True	DATE	None
36	begbid	True	DATE	None
37	endbid	True	DATE	None
38	begask	True	DATE	None
39	endask	True	DATE	None
40	begopr	True	DATE	None
41	endopr	True	DATE	None
42	hsicmg	True	DOUBLE PRECISION	None
43	hsicig	True	DOUBLE PRECISION	None

[ ]:
```
conn.describe_table('crsp', 'dse')
```

Approximately 12648999 rows in crsp.dse.

[ ]:

	name	nullable	type	comment
0	event	True	VARCHAR(8)	None
1	date	True	DATE	None
2	hsicmg	True	DOUBLE PRECISION	None
3	hsicig	True	DOUBLE PRECISION	None
4	comnam	True	VARCHAR(32)	None
5	cusip	True	VARCHAR(8)	None
6	dclrdt	True	DATE	None
7	dlamt	True	NUMERIC(11, 5)	None
8	dlpdt	True	DATE	None
9	dlstcd	True	SMALLINT	None
10	hsiccd	True	INTEGER	None
11	issuno	True	INTEGER	None
12	ncusip	True	VARCHAR(8)	None
13	nextdt	True	DATE	None
14	paydt	True	DATE	None
15	rcrddt	True	DATE	None
16	shrcls	True	VARCHAR(1)	None
17	shrflg	True	SMALLINT	None
18	ticker	True	VARCHAR(5)	None
19	permno	True	INTEGER	None
20	nameendt	True	DATE	None
21	shrcd	True	SMALLINT	None

22	exchcd	True	SMALLINT	None
23	siccd	True	INTEGER	None
24	tsymbol	True	VARCHAR(10)	None
25	naics	True	VARCHAR(7)	None
26	primexch	True	VARCHAR(1)	None
27	trdstat	True	VARCHAR(1)	None
28	secstat	True	VARCHAR(1)	None
29	permco	True	INTEGER	None
30	compno	True	INTEGER	None
31	hexcd	True	SMALLINT	None
32	distcd	True	SMALLINT	None
33	divamt	True	NUMERIC(11, 5)	None
34	facpr	True	NUMERIC(10, 5)	None
35	facshr	True	NUMERIC(10, 5)	None
36	acperm	True	INTEGER	None
37	accomp	True	INTEGER	None
38	nwperm	True	INTEGER	None
39	nwcomp	True	INTEGER	None
40	dlretx	True	NUMERIC(10, 6)	None
41	dlprc	True	NUMERIC(11, 5)	None
42	dlret	True	NUMERIC(10, 6)	None
43	shrout	True	NUMERIC(10, 0)	None
44	shrenddt	True	DATE	None
45	trtscd	True	SMALLINT	None
46	trtsendt	True	DATE	None
47	nmsind	True	SMALLINT	None
48	mmcnt	True	SMALLINT	None
49	nsdinx	True	SMALLINT	None

[ ]:
```
conn.describe_table('crspm', 'dsf')
```

Approximately 105270144 rows in crspm.dsf.

[ ]:

	name	nullable	type	comment
0	cusip	True	VARCHAR(8)	None
1	permno	True	INTEGER	None
2	permco	True	INTEGER	None
3	issuno	True	INTEGER	None
4	hexcd	True	SMALLINT	None
5	hsiccd	True	INTEGER	None
6	date	True	DATE	None
7	bidlo	True	NUMERIC(11, 5)	None
8	askhi	True	NUMERIC(11, 5)	None
9	prc	True	NUMERIC(11, 5)	None
10	vol	True	NUMERIC(10, 0)	None
11	ret	True	NUMERIC(10, 6)	None
12	bid	True	NUMERIC(11, 5)	None
13	ask	True	NUMERIC(11, 5)	None

```
14   shrout      True  DOUBLE PRECISION    None
15   cfacpr      True  DOUBLE PRECISION    None
16  cfacshr      True  DOUBLE PRECISION    None
17  openprc      True     NUMERIC(11, 5)    None
18   numtrd      True            INTEGER    None
19     retx      True     NUMERIC(10, 6)    None
```

[ ]:
```
conn.describe_table('crsp', 'stocknames')
```

```
Approximately 80790 rows in crsp.stocknames.
```

[ ]:
```
         name  nullable              type comment
0      permno      True           INTEGER    None
1      namedt      True              DATE    None
2   nameenddt      True              DATE    None
3       shrcd      True          SMALLINT    None
4      exchcd      True          SMALLINT    None
5       siccd      True           INTEGER    None
6      ncusip      True        VARCHAR(8)    None
7      ticker      True        VARCHAR(8)    None
8      comnam      True       VARCHAR(35)    None
9      shrcls      True        VARCHAR(4)    None
10     permco      True           INTEGER    None
11      hexcd      True          SMALLINT    None
12      cusip      True        VARCHAR(8)    None
13    st_date      True              DATE    None
14   end_date      True              DATE    None
15    namedum      True  DOUBLE PRECISION    None
```

[ ]:
```
conn.describe_table('crsp', 'dsenames')
```

```
Approximately 113885 rows in crsp.dsenames.
```

[ ]:
```
         name  nullable         type comment
0      permno      True      INTEGER    None
1      namedt      True         DATE    None
2   nameendt      True         DATE    None
3       shrcd      True     SMALLINT    None
4      exchcd      True     SMALLINT    None
5       siccd      True      INTEGER    None
6      ncusip      True   VARCHAR(8)    None
7      ticker      True   VARCHAR(8)    None
8      comnam      True  VARCHAR(35)    None
9      shrcls      True   VARCHAR(4)    None
10    tsymbol      True  VARCHAR(10)    None
11      naics      True   VARCHAR(7)    None
12   primexch      True   VARCHAR(1)    None
13    trdstat      True   VARCHAR(1)    None
```

```
14    secstat      True    VARCHAR(1)     None
15     permco      True       INTEGER     None
16     compno      True       INTEGER     None
17     issuno      True       INTEGER     None
18      hexcd      True      SMALLINT     None
19     hsiccd      True       INTEGER     None
20      cusip      True    VARCHAR(8)     None
```

[ ]:
```
conn.describe_table('crsp', 'crsp_header')
```

```
Approximately 679 rows in crsp.crsp_header.
```

[ ]:
```
      name  nullable         type  comment
0   permno      True      INTEGER     None
1   permco      True      INTEGER     None
2    begdt      True         DATE     None
3    enddt      True         DATE     None
4   comnam      True  VARCHAR(64)     None
5  hdlstcd      True     SMALLINT     None
```

Begin coding

[ ]:
```
check = {'tickers': ('JNJ', 'COP')}
check
```

[ ]:
```
{'tickers': ('JNJ', 'COP')}
```

[ ]:
```
checking = conn.raw_sql('select a.permno, a.ticker, a.comnam, b.date, b.prc, b.
↪ret, b.retx, b.cfacpr from crsp.stocknames a join crsp.dsf b on a.permno = b.
↪permno WHERE a.ticker in %(tickers)s and a.st_date <= b.date and b.date <= a.
↪end_date', params=check)
checking
del checking
```

[ ]:
```
stocks = ('AAPL', 'TSLA', 'AMZN', 'GOOG', 'MSFT', 'WMT', 'HD', 'JNJ', 'JPM',␣
↪'T', 'SPY', 'EWJ')
```

[ ]:
```
stocks_dict = {'tickers': ('AAPL', 'TSLA', 'AMZN', 'GOOG', 'MSFT', 'WMT', 'HD',␣
↪'JNJ', 'JPM', 'T', 'SPY', 'EWJ')}
print(stocks_dict)
type(stocks_dict)
```

```
{'tickers': ('AAPL', 'TSLA', 'AMZN', 'GOOG', 'MSFT', 'WMT', 'HD', 'JNJ', 'JPM',
'T', 'SPY', 'EWJ')}
```

[ ]:
```
dict
```

[ ]:
```

```

```
raw_data_from_crsp = conn.raw_sql('select a.permno, a.ticker, a.comnam, a.
 ↪tsymbol, b.date, b.prc, b.ret, b.retx, b.cfacpr from crsp.dsenames a join␣
 ↪crsp.dsf b on a.permno = b.permno WHERE a.tsymbol in %(tickers)s and a.
 ↪namedt <= b.date and b.date <= a.nameendt', params=stocks_dict)
pd_data = pd.DataFrame(raw_data_from_crsp)
pd_data.info()

del raw_data_from_crsp
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73632 entries, 0 to 73631
Data columns (total 9 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   permno   73632 non-null  int64
 1   ticker   73632 non-null  object
 2   comnam   73632 non-null  object
 3   tsymbol  73632 non-null  object
 4   date     73632 non-null  object
 5   prc      73632 non-null  float64
 6   ret      73627 non-null  float64
 7   retx     73627 non-null  float64
 8   cfacpr   73632 non-null  float64
dtypes: float64(4), int64(1), object(4)
memory usage: 5.1+ MB
```

[ ]:
```
print(pd_data.head())
print(pd_data.tail())
```

```
   permno ticker              comnam tsymbol        date    prc       ret  \
0   14593   AAPL  APPLE COMPUTER INC    AAPL  1982-11-01  26.75  0.054187
1   10107   MSFT      MICROSOFT CORP    MSFT  1986-03-13  28.00       NaN
2   10107   MSFT      MICROSOFT CORP    MSFT  1986-03-14  29.00  0.035714
3   10107   MSFT      MICROSOFT CORP    MSFT  1986-03-17  29.50  0.017241
4   10107   MSFT      MICROSOFT CORP    MSFT  1986-03-18  28.75 -0.025424

       retx  cfacpr
0  0.054187   224.0
1       NaN   288.0
2  0.035714   288.0
3  0.017241   288.0
4 -0.025424   288.0
       permno ticker      comnam tsymbol        date         prc       ret  \
73627   90319   GOOG  GOOGLE INC    GOOG  2014-03-27  1114.28003 -0.015628
73628   90319   GOOG  GOOGLE INC    GOOG  2014-03-28  1120.15002  0.005268
73629   90319   GOOG  GOOGLE INC    GOOG  2014-03-31  1114.51001 -0.005035
73630   90319   GOOG  GOOGLE INC    GOOG  2014-04-01  1134.89001  0.018286
73631   90319   GOOG  GOOGLE INC    GOOG  2014-04-02  1135.09998  0.000185
```

```
           retx  cfacpr
73627 -0.015628  39.938
73628  0.005268  39.938
73629 -0.005035  39.938
73630  0.018286  39.938
73631  0.000185  39.938
```

Convert dates to datetime and sort for proper indexing

[ ]:
```
pd_data['date'] = pd.to_datetime(pd_data['date'], format='%Y-%m-%d')

pd_data = pd_data.sort_values(by='date')
```

[ ]:
```
pd_data_index = pd_data.set_index('date')
pd_data_index.head()
```

[ ]:
```
            permno ticker             comnam tsymbol     prc       ret  \
date
1982-11-01   14593   AAPL  APPLE COMPUTER INC    AAPL  26.750  0.054187
1982-11-02   14593   AAPL  APPLE COMPUTER INC    AAPL  28.625  0.070094
1982-11-03   14593   AAPL  APPLE COMPUTER INC    AAPL  30.750  0.074236
1982-11-04   14593   AAPL  APPLE COMPUTER INC    AAPL  31.000  0.008130
1982-11-05   14593   AAPL  APPLE COMPUTER INC    AAPL  30.125 -0.028226

                retx  cfacpr
date
1982-11-01  0.054187   224.0
1982-11-02  0.070094   224.0
1982-11-03  0.074236   224.0
1982-11-04  0.008130   224.0
1982-11-05 -0.028226   224.0
```

[ ]:
```
# Adjust price for splits
pd_data_index['adj price'] = pd_data_index['prc'] / pd_data_index['cfacpr']
pd_data_index.info()

del pd_data
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 73632 entries, 1982-11-01 to 2023-12-29
Data columns (total 9 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   permno     73632 non-null  int64
 1   ticker     73632 non-null  object
 2   comnam     73632 non-null  object
 3   tsymbol    73632 non-null  object
 4   prc        73632 non-null  float64
```

```
 5   ret        73627 non-null  float64
 6   retx       73627 non-null  float64
 7   cfacpr     73632 non-null  float64
 8   adj price  73632 non-null  float64
dtypes: float64(5), int64(1), object(3)
memory usage: 5.6+ MB
```

Slicing the ticker and price column

[ ]:
```
pd_data_index_tic_prc = pd_data_index[['ticker', 'adj price']]
pd_data_index_tic_prc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 73632 entries, 1982-11-01 to 2023-12-29
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   ticker     73632 non-null  object
 1   adj price  73632 non-null  float64
dtypes: float64(1), object(1)
memory usage: 1.7+ MB
```

[ ]:
```
# Slice out data from January 1, 2016 to March 30, 2018
b = pd_data_index_tic_prc[('2016-01-01' <= pd_data_index_tic_prc.index) &␣
 ↪(pd_data_index_tic_prc.index <= '2023-12-31')]
print(b.head(), b.tail())
```

```
           ticker  adj price
date
2016-01-04    SPY   201.0192
2016-01-04   AMZN    31.8495
2016-01-04    JNJ   100.4800
2016-01-04   AAPL    26.3375
2016-01-04   TSLA    14.8940             ticker  adj price
date
2023-12-29    EWJ   64.14000
2023-12-29   MSFT  376.04001
2023-12-29   AAPL  192.53000
2023-12-29    JPM  170.10001
2023-12-29    SPY  475.31000
```

[ ]:
```
b.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 24144 entries, 2016-01-04 to 2023-12-29
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   ticker     24144 non-null  object
 1   adj price  24144 non-null  float64
```

```
dtypes: float64(1), object(1)
memory usage: 565.9+ KB
```

[ ]:
```
table = b.pivot(columns='ticker')
table.columns = [col[1] for col in table.columns]
table.head()
```

[ ]:
```
                AAPL       AMZN    EWJ       GOOG         HD     JNJ    JPM  \
date
2016-01-04  26.3375  31.849500  47.72  37.092001  131.07001  100.48  63.62
2016-01-05  25.6775  31.689499  48.32  37.129001  130.42999  100.90  63.73
2016-01-06  25.1750  31.632501  47.48  37.181000  129.08000  100.39  62.81
2016-01-07  24.1125  30.397000  46.76  36.319500  125.40000   99.22  60.27
2016-01-08  24.2400  30.352500  45.76  35.723498  123.90000   98.16  58.92

             MSFT        SPY          T       TSLA    WMT
date
2016-01-04  54.80  201.01920  26.314076  14.894000  61.46
2016-01-05  55.05  201.36000  26.497930  14.895333  62.92
2016-01-06  54.05  198.82001  26.091919  14.602666  63.55
2016-01-07  52.17  194.05000  25.670588  14.376666  65.03
2016-01-08  52.33  191.92300  25.693569  14.066667  63.54
```

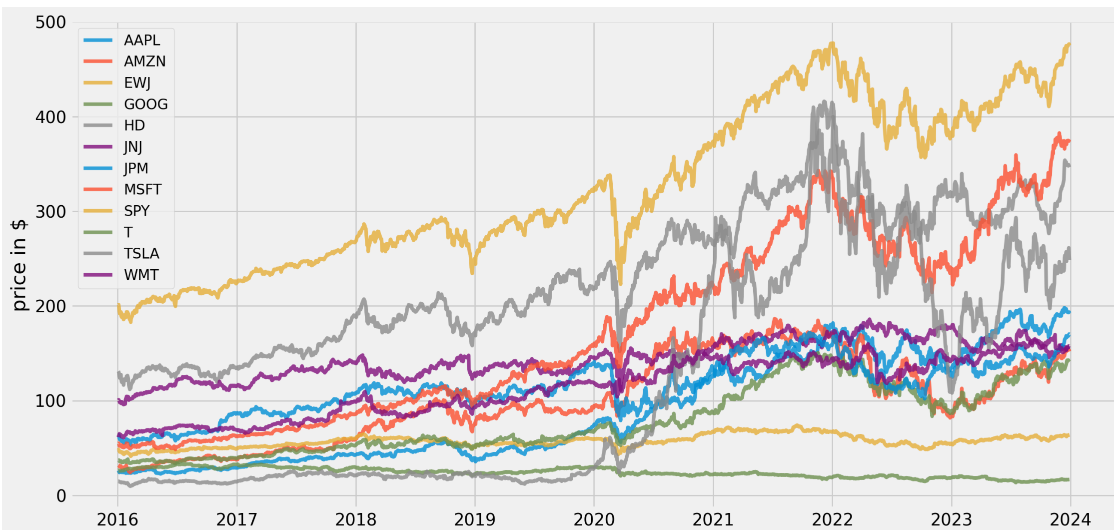
[ ]:
```
table.describe()
```

[ ]:
```
              AAPL         AMZN          EWJ         GOOG           HD  \
count  2012.000000  2012.000000  2012.000000  2012.000000  2012.000000
mean     90.791868   100.472923    57.301663    79.061171   234.346635
std      55.870061    44.802150     6.474248    35.156013    74.411156
min      22.585000    24.103501    41.280000    33.413000   111.850000
25%      41.314373    59.763623    53.057500    51.427498   172.390000
50%      66.661251    94.572501    56.995000    64.682751   225.655000
75%     145.860000   140.577510    60.810000   110.598874   298.920003
max     198.110000   186.570495    74.120000   150.708996   416.179990

               JNJ          JPM         MSFT          SPY            T  \
count  2012.000000  2012.000000  2012.000000  2012.000000  2012.000000
mean    144.412646   114.531426   176.226521   325.329161    23.972375
std      20.077548    28.720264    97.166219    82.962306     4.943863
min      95.750000    53.070000    48.430000   182.860000    13.450000
25%     129.637500    95.507500    86.250000   258.267495    20.154913
50%     142.585010   113.345000   153.435005   300.199995    23.571590
75%     162.764993   137.582500   259.447492   407.132502    28.583521
max     186.009990   171.780000   382.700010   477.709990    33.300521

              TSLA          WMT
count  2012.000000  2012.000000
mean    113.851756   113.965268
```

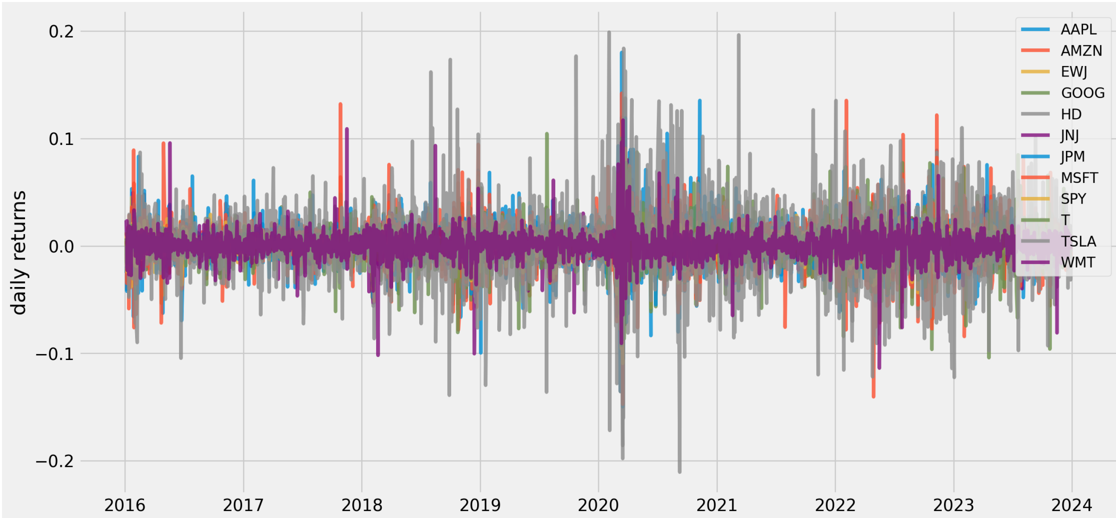
```
std     112.995334    30.335506
min       9.578000    60.840000
25%      18.944666    86.267500
50%      28.505667   118.130000
75%     223.411667   141.695002
max     409.970010   169.780000
```

Buid tables and calculate returns

[ ]:
```
plt.figure(figsize=(14, 7))
for c in table.columns.values:
    plt.plot(table.index, table[c], lw=3, alpha=0.8,label=c)
plt.legend(loc='upper left', fontsize=12)
plt.ylabel('price in $')

plt.show()
```



[ ]:
```
returns = table.pct_change()

plt.figure(figsize=(14,7))
for c in returns.columns.values:
    plt.plot(returns.index, returns[c], lw=3, alpha=0.8, label=c)
plt.legend(loc='upper right', fontsize=12)
plt.ylabel('daily returns')
plt.show()
```



[ ]:
```
returns.describe()
```

[ ]:

	AAPL	AMZN	EWJ	GOOG	HD
count	2011.000000	2011.000000	2011.000000	2011.000000	2011.000000
mean	0.001160	0.000995	0.000205	0.000825	0.000612
std	0.018477	0.020883	0.010722	0.017934	0.015941
min	-0.128647	-0.140494	-0.098047	-0.111008	-0.197938
25%	-0.007397	-0.008626	-0.005151	-0.006919	-0.006432
50%	0.000951	0.001224	0.000484	0.000979	0.000974
75%	0.010258	0.011161	0.005884	0.009408	0.008267
max	0.119808	0.135359	0.069444	0.104485	0.137508

\

	JNJ	JPM	MSFT	SPY	T
count	2011.000000	2011.000000	2011.000000	2011.000000	2011.000000
mean	0.000290	0.000649	0.001111	0.000496	-0.000110
std	0.011716	0.017896	0.017512	0.011635	0.015067
min	-0.100379	-0.149649	-0.147390	-0.109424	-0.104061
25%	-0.004967	-0.007483	-0.006874	-0.003757	-0.006606
50%	0.000225	0.000352	0.000982	0.000596	0.000338
75%	0.005922	0.008676	0.010073	0.005971	0.007076
max	0.079977	0.180125	0.142169	0.090603	0.100223

\

	TSLA	WMT
count	2011.000000	2011.000000
mean	0.002071	0.000561
std	0.036667	0.013590
min	-0.210628	-0.113757
25%	-0.016192	-0.005575
50%	0.001367	0.000625

```
75%        0.019509     0.006763
max        0.198949     0.117085
```

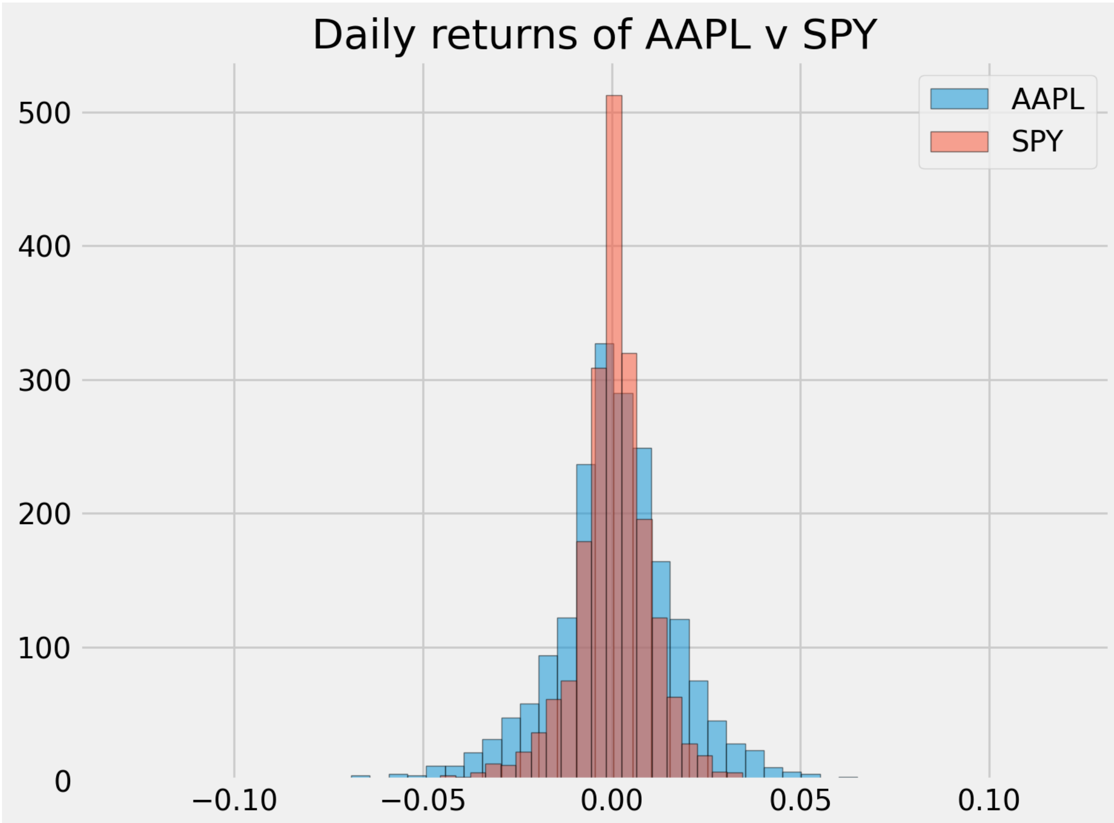
1.2 Student Code

```
[ ]: monthly_returns = returns.groupby([returns.index.year, returns.index.month]).
      ↪apply(lambda x: (1 + x).prod() -1)
     monthly_returns.describe()
```

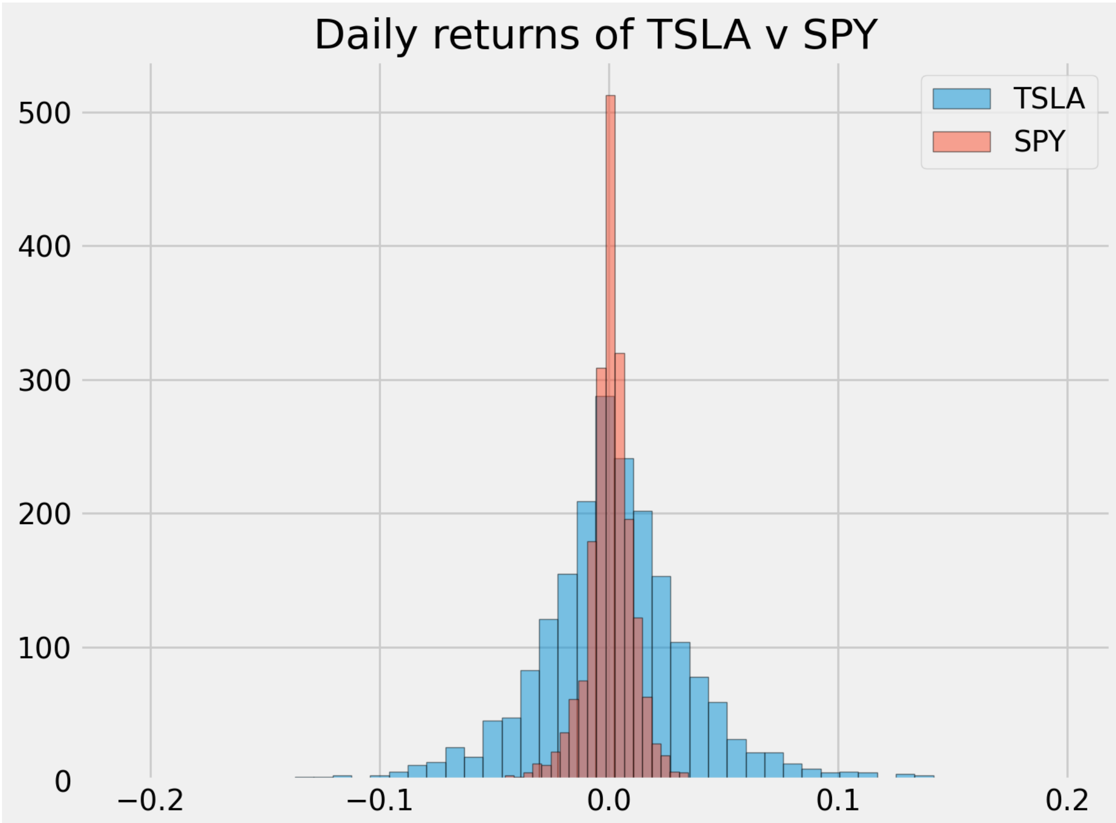
```
[ ]:             AAPL       AMZN        EWJ       GOOG         HD        JNJ  \
     count  96.000000  96.000000  96.000000  96.000000  96.000000  96.000000
     mean    0.024379   0.020310   0.003904   0.016407   0.012246   0.005736
     std     0.083816   0.089667   0.040680   0.069796   0.064747   0.047036
     min    -0.184045  -0.237525  -0.089490  -0.176750  -0.150953  -0.121511
     25%    -0.036637  -0.043764  -0.020767  -0.026062  -0.030735  -0.019814
     50%     0.030846   0.026152   0.005946   0.017290   0.011719   0.008246
     75%     0.090949   0.074636   0.025178   0.066462   0.060079   0.036493
     max     0.214380   0.270596   0.116223   0.165080   0.181582   0.144208

                  JPM       MSFT        SPY          T       TSLA        WMT
     count  96.000000  96.000000  96.000000  96.000000  96.000000  96.000000
     mean    0.012896   0.021952   0.010080  -0.002779   0.045409   0.011206
     std     0.072456   0.058945   0.046473   0.061706   0.188524   0.051929
     min    -0.224615  -0.109267  -0.129987  -0.172345  -0.367334  -0.159226
     25%    -0.033512  -0.013035  -0.011706  -0.037891  -0.077007  -0.022039
     50%     0.017286   0.021351   0.014942  -0.005132   0.011420   0.013629
     75%     0.058547   0.059079   0.036582   0.036854   0.128318   0.043628
     max     0.204593   0.176291   0.126984   0.188396   0.741452   0.117353
```

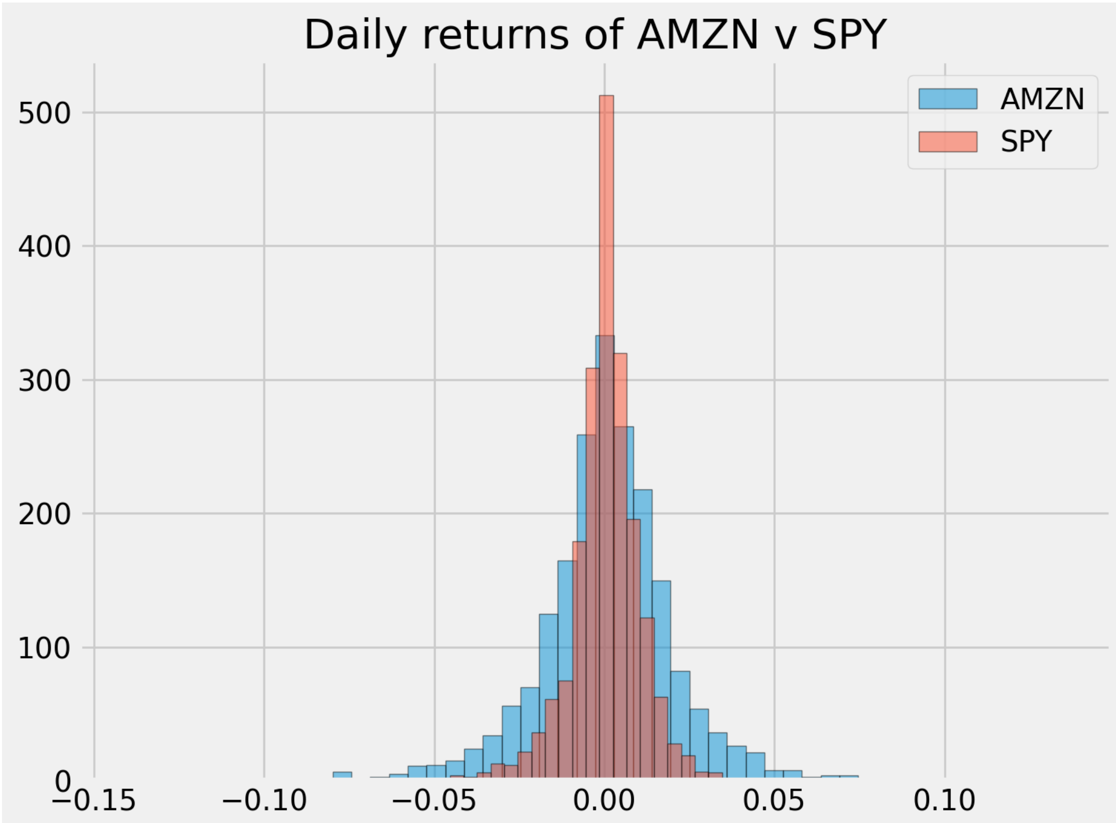
```
[ ]: def stock_histogram(dataframe, asset, title, market='SPY'):
         fig, ax = plt.subplots(figsize=(8, 6))
         plt.hist(dataframe[asset], bins=50, alpha=0.5, label=asset,␣
      ↪edgecolor='black')
         plt.hist(dataframe[market], bins=50, alpha=0.5, label=market,␣
      ↪edgecolor='black')
         plt.title(title)
         plt.legend(loc='upper right')
         plt.show()
```

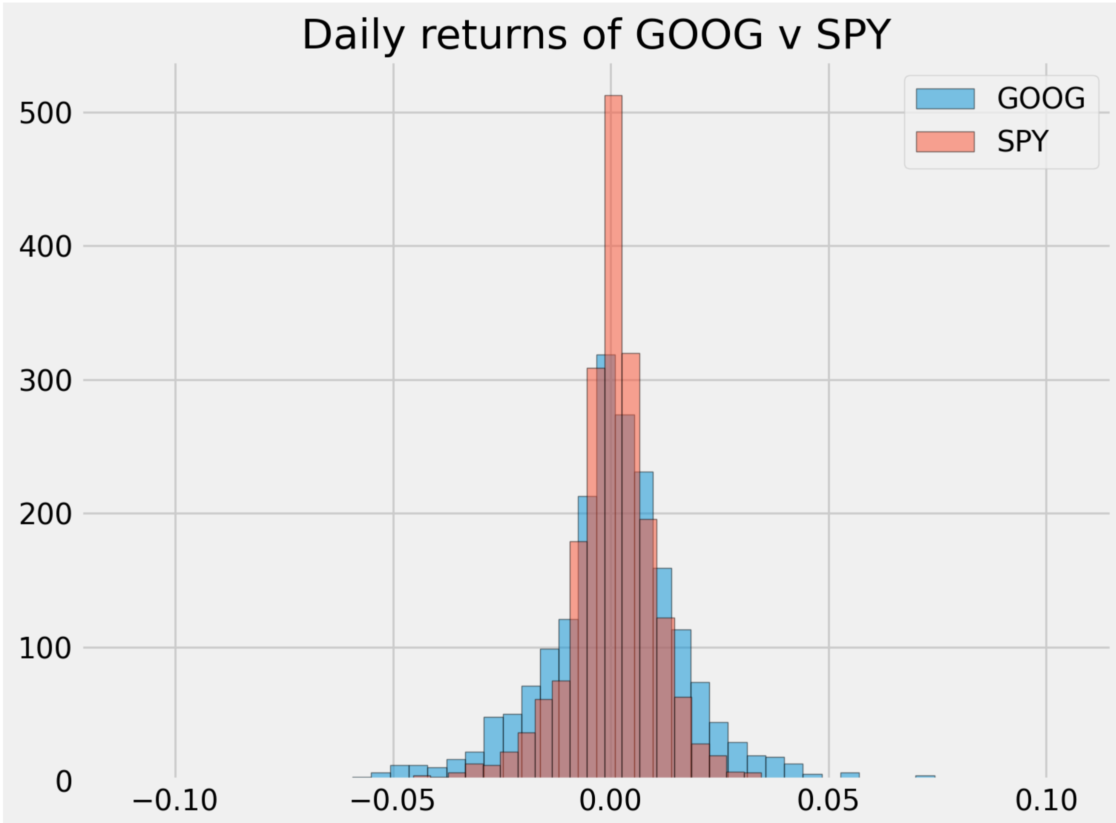
```
[ ]: stocks = ['AAPL', 'TSLA', 'AMZN', 'GOOG', 'MSFT', 'WMT', 'HD', 'JNJ', 'JPM',␣
      ↪'T', 'SPY', 'EWJ']
     for stock in stocks:
         title = f"Daily returns of {stock} v SPY"
         if stock != 'SPY':
             stock_histogram(returns, stock, title)
         else:
             continue
```

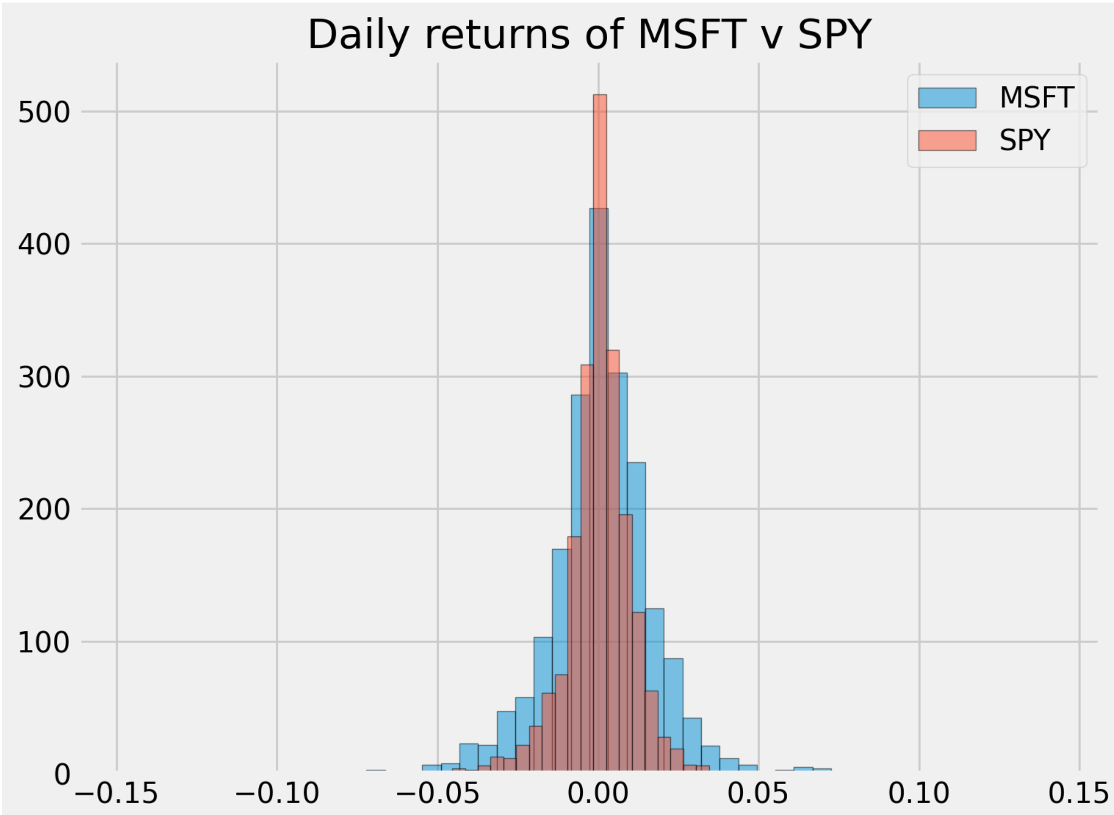


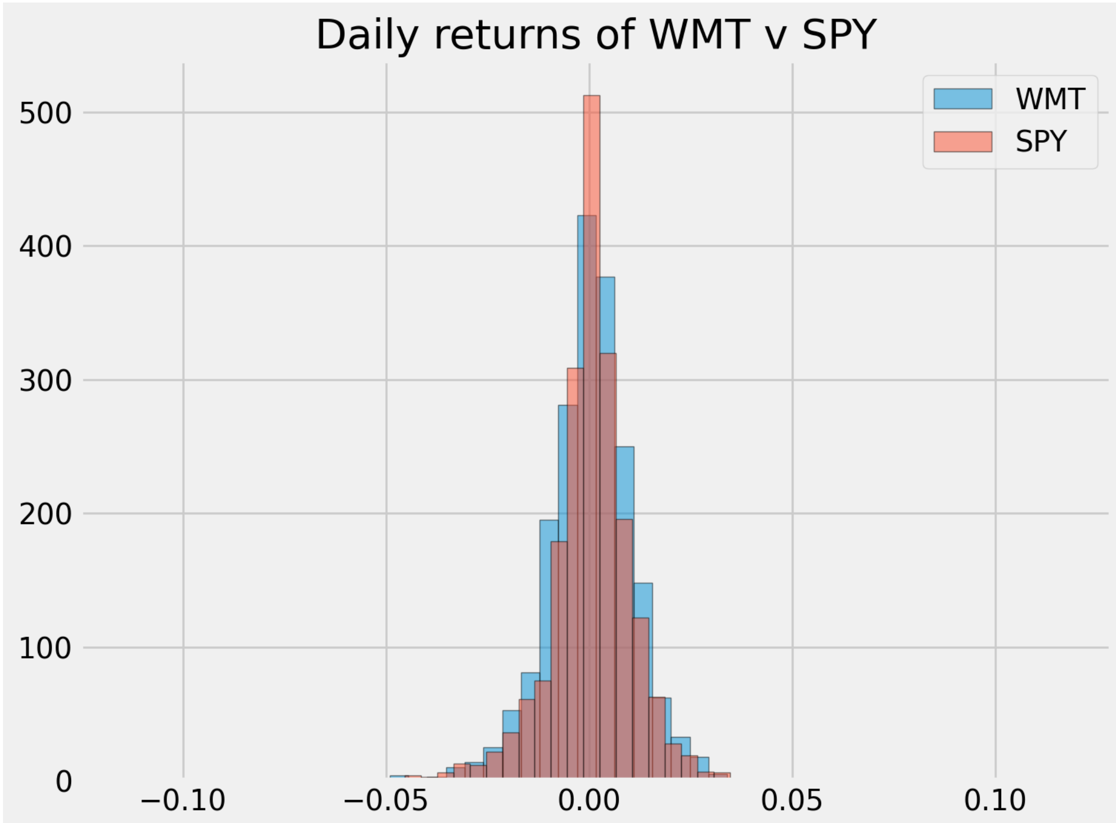
Daily returns of AAPL v SPY
AAPL
SPY
500
400
300
200
100
0
−0.10
−0.05
0.00
0.05
0.10

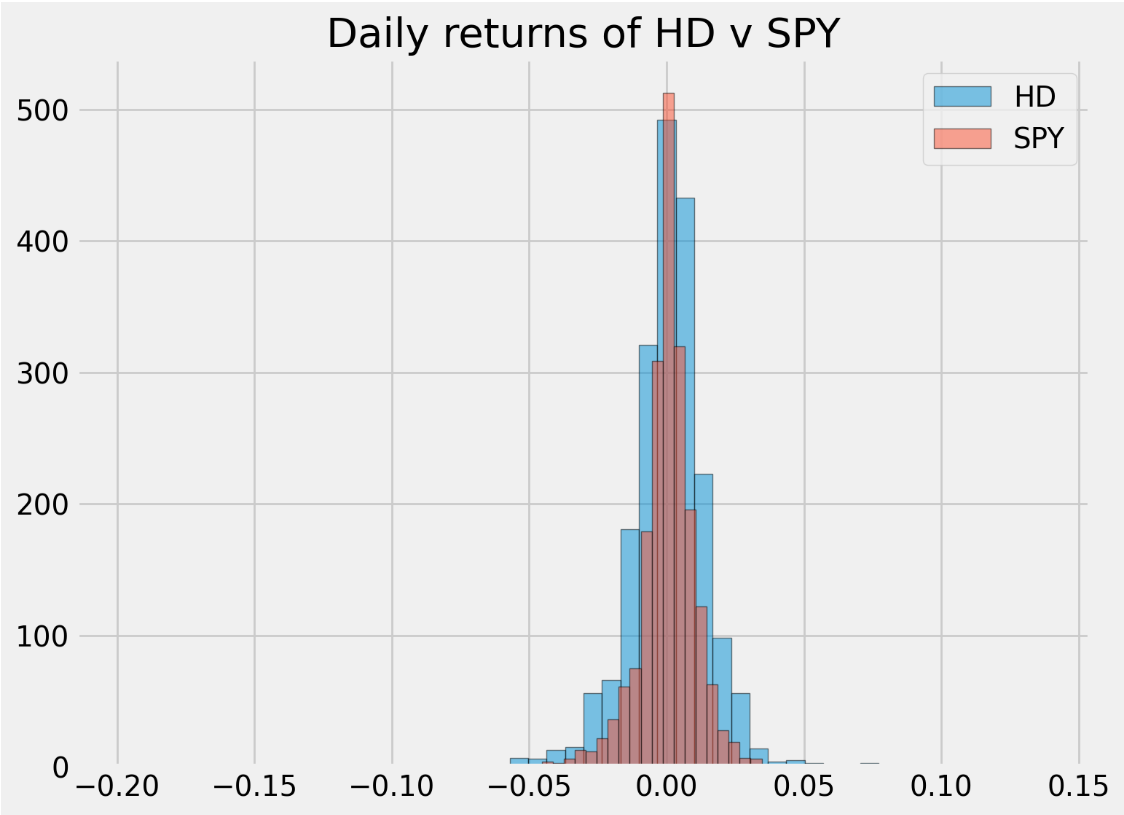

Daily returns of TSLA v SPY
TSLA
SPY
500
400
300
200
100
0
−0.2
−0.1
0.0
0.1
0.2

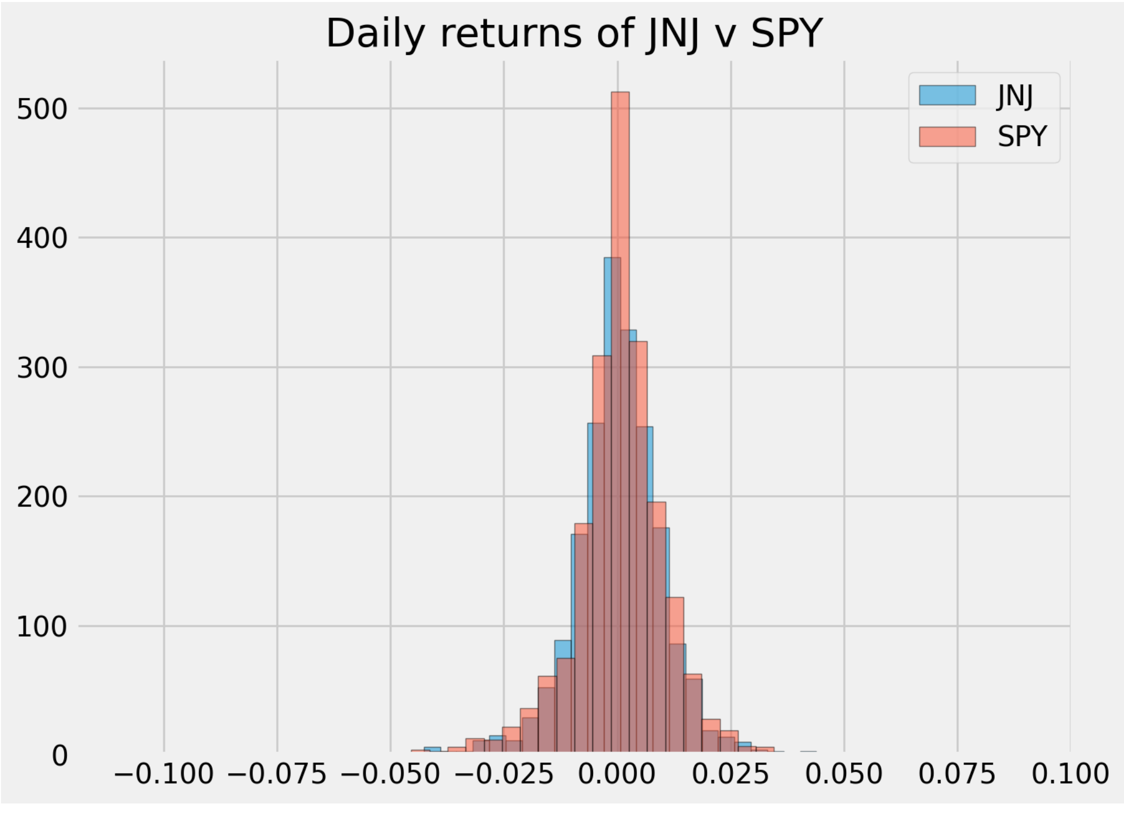

Daily returns of AMZN v SPY
AMZN
SPY
500
400
300
200
100
0
−0.15
−0.10
−0.05
0.00
0.05
0.10

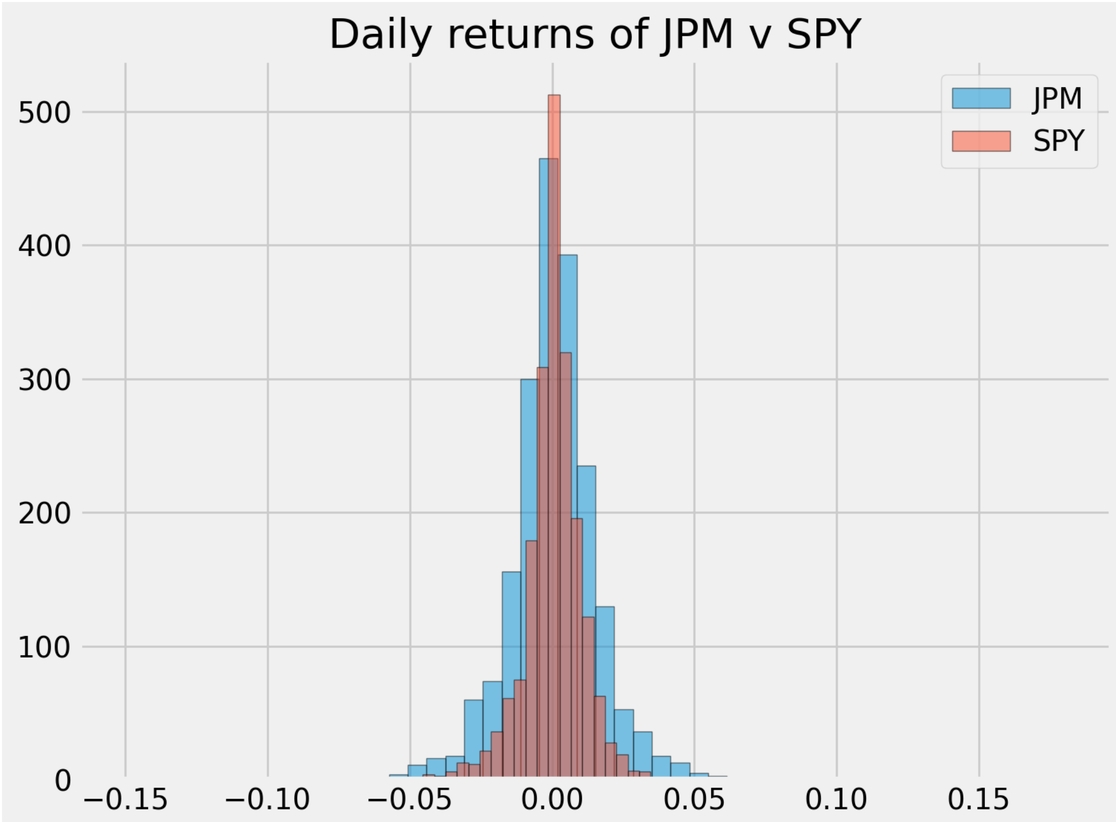

Daily returns of GOOG v SPY
GOOG
SPY
500
400
300
200
100
0
−0.10
−0.05
0.00
0.05
0.10


Daily returns of MSFT v SPY
MSFT
SPY
500
400
300
200
100
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15

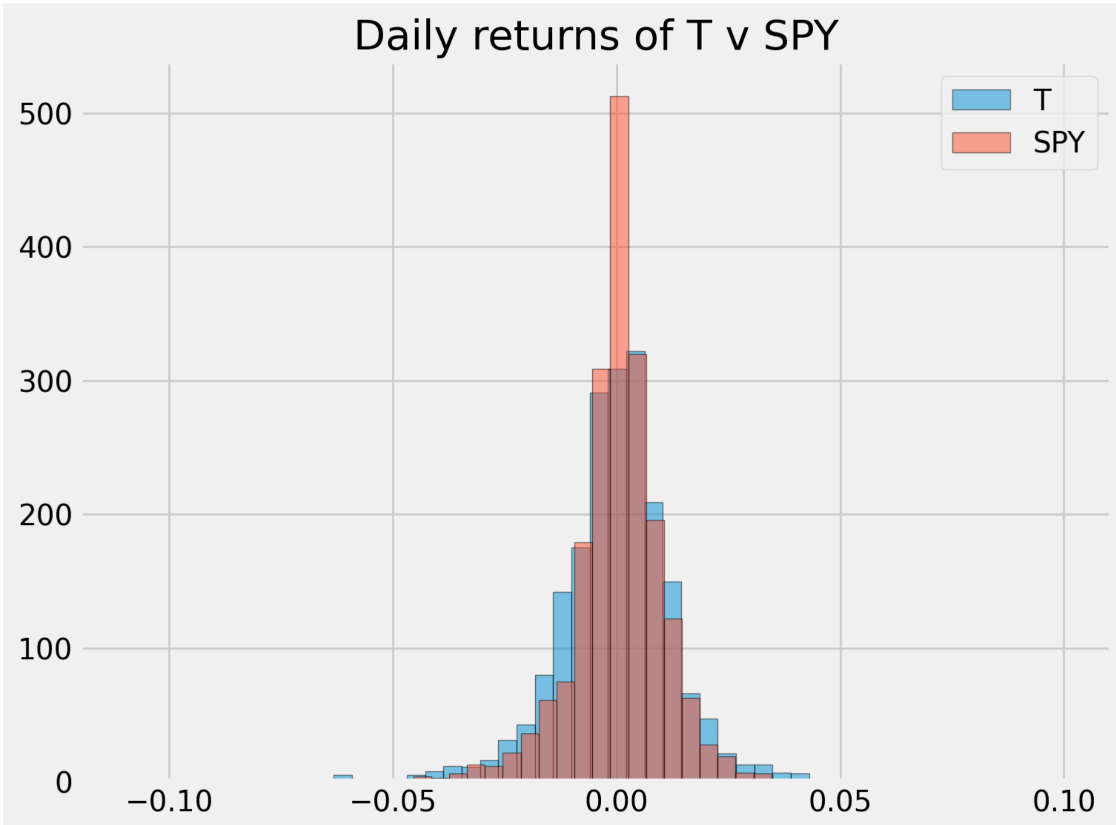

Daily returns of WMT v SPY
WMT
SPY
500
400
300
200
100
0
−0.10
−0.05
0.00
0.05
0.10

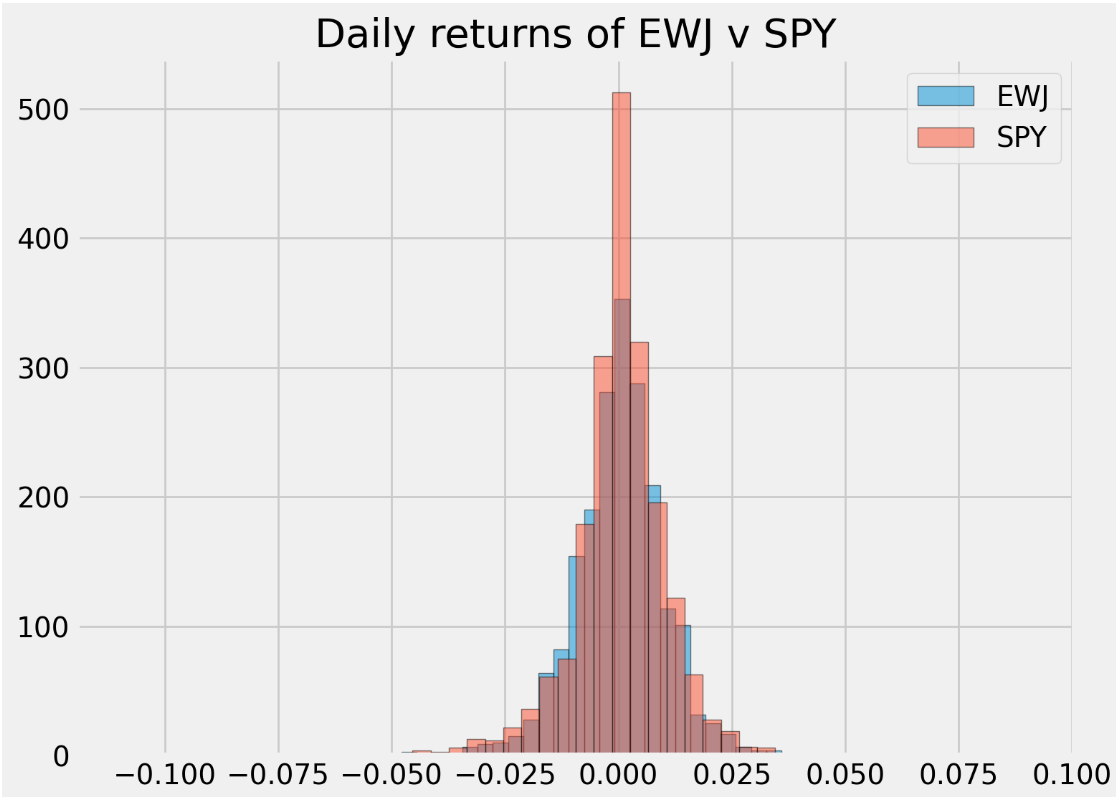

Daily returns of HD v SPY
HD
SPY
500
400
300
200
100
0
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15


Daily returns of JNJ v SPY
JNJ
SPY
500
400
300
200
100
0
−0.100
−0.075
−0.050
−0.025
0.000
0.025
0.050
0.075
0.100

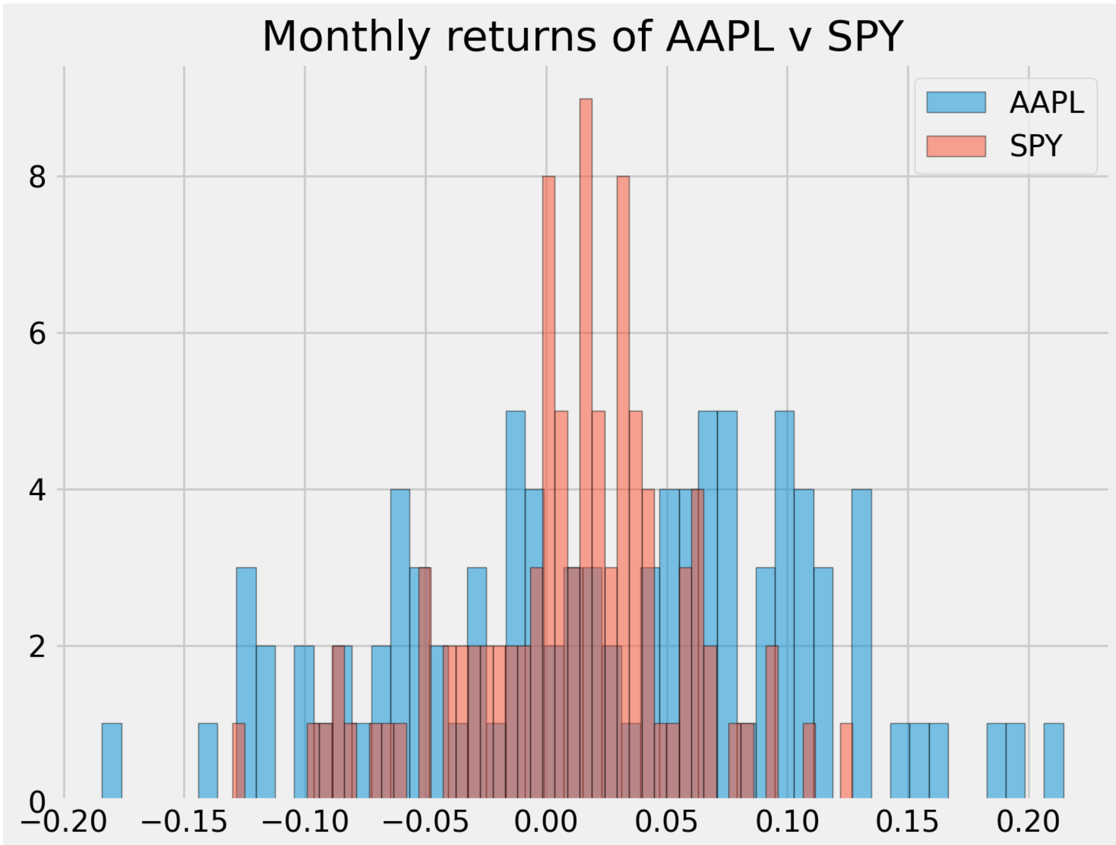


Daily returns of JPM v SPY
JPM
SPY
500
400
300
200
100
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15

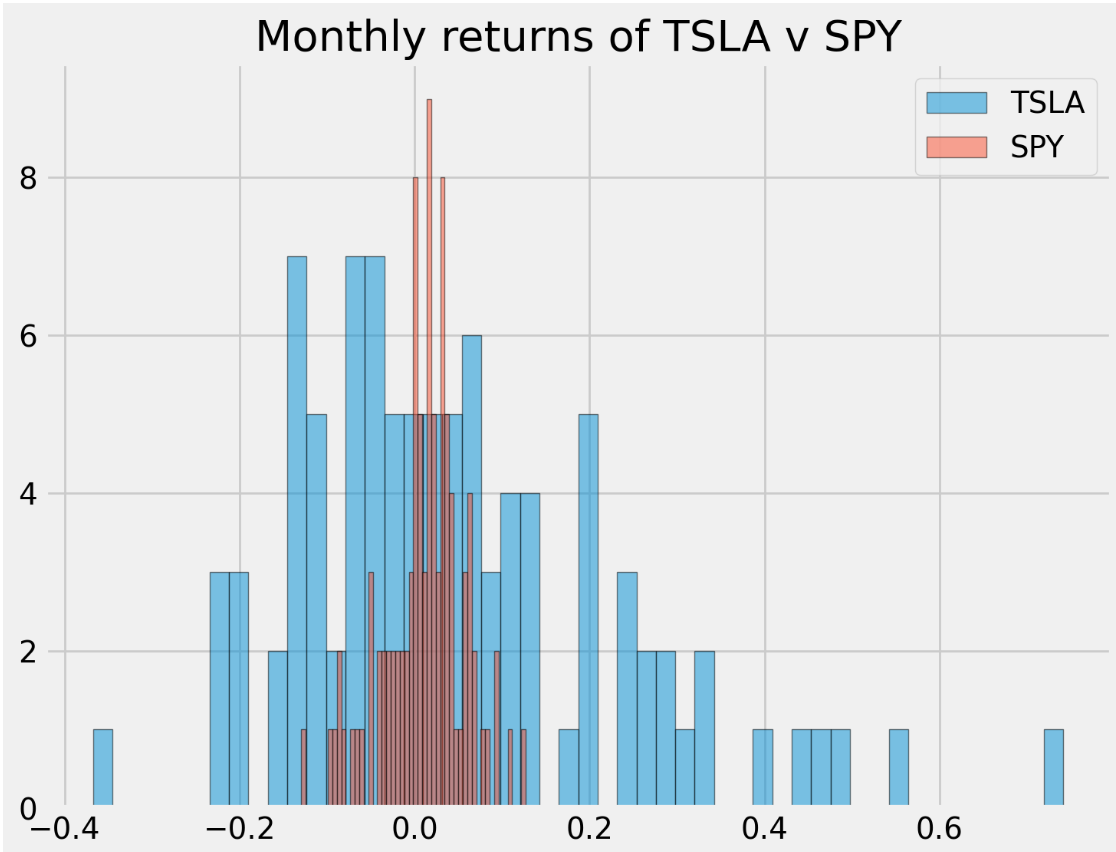

Daily returns of T v SPY
T
SPY
500
400
300
200
100
0
−0.10
−0.05
0.00
0.05
0.10

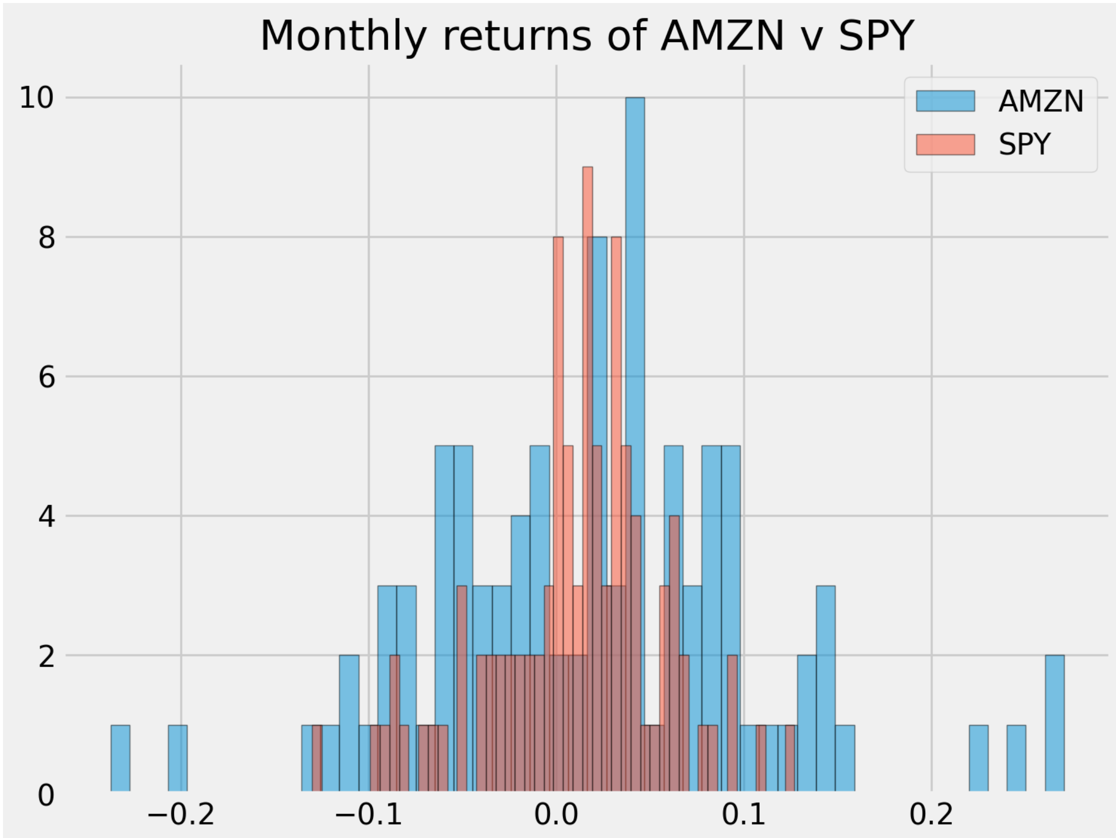


[ ]:
```
# Create graphs for monthly returns
for stock in stocks:
    title = f"Monthly returns of {stock} v SPY"
    if stock != 'SPY':
        stock_histogram(monthly_returns, stock, title)
    else:
        continue
```

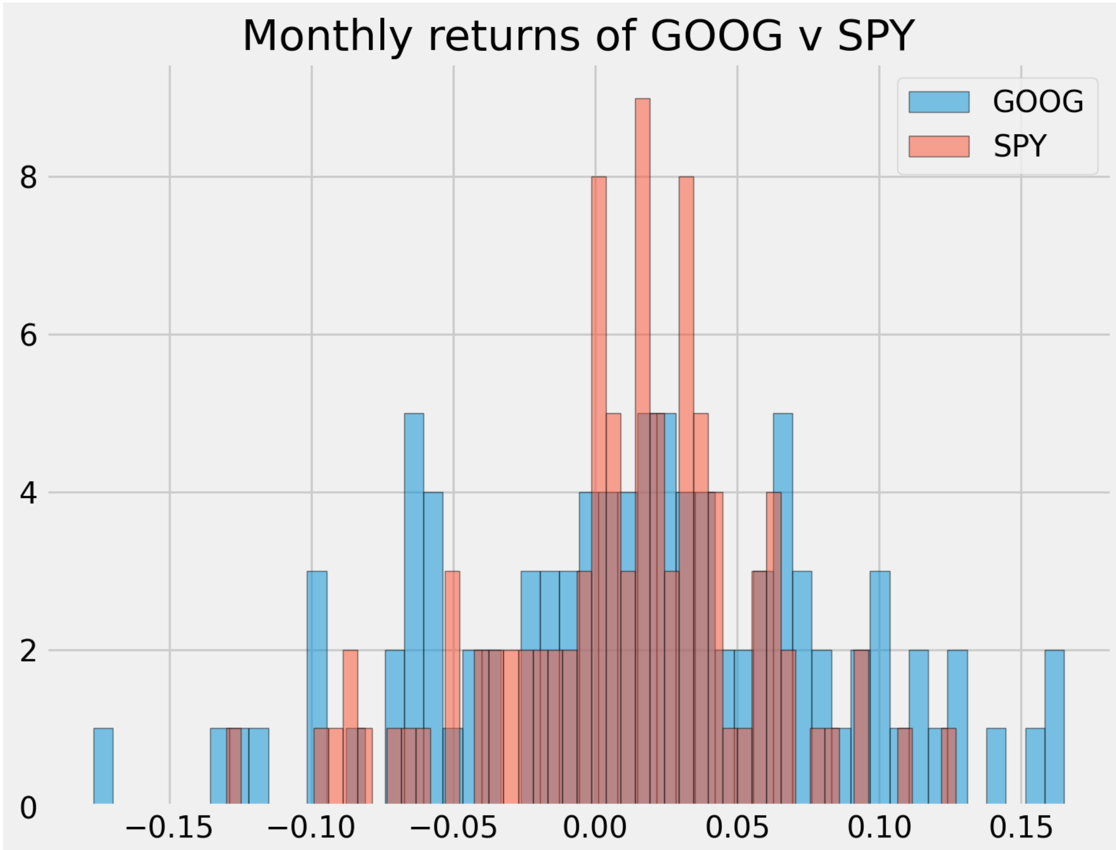


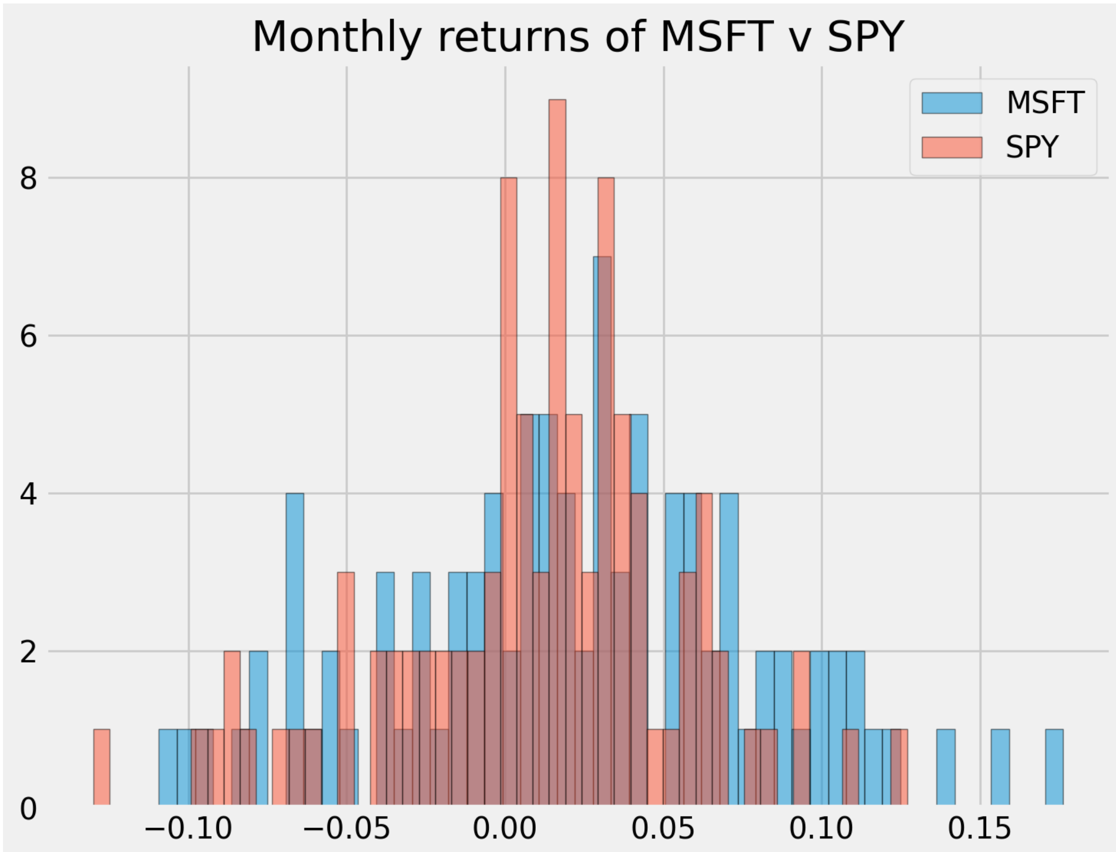
Monthly returns of AAPL v SPY
AAPL
SPY
0
2
4
6
8
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20

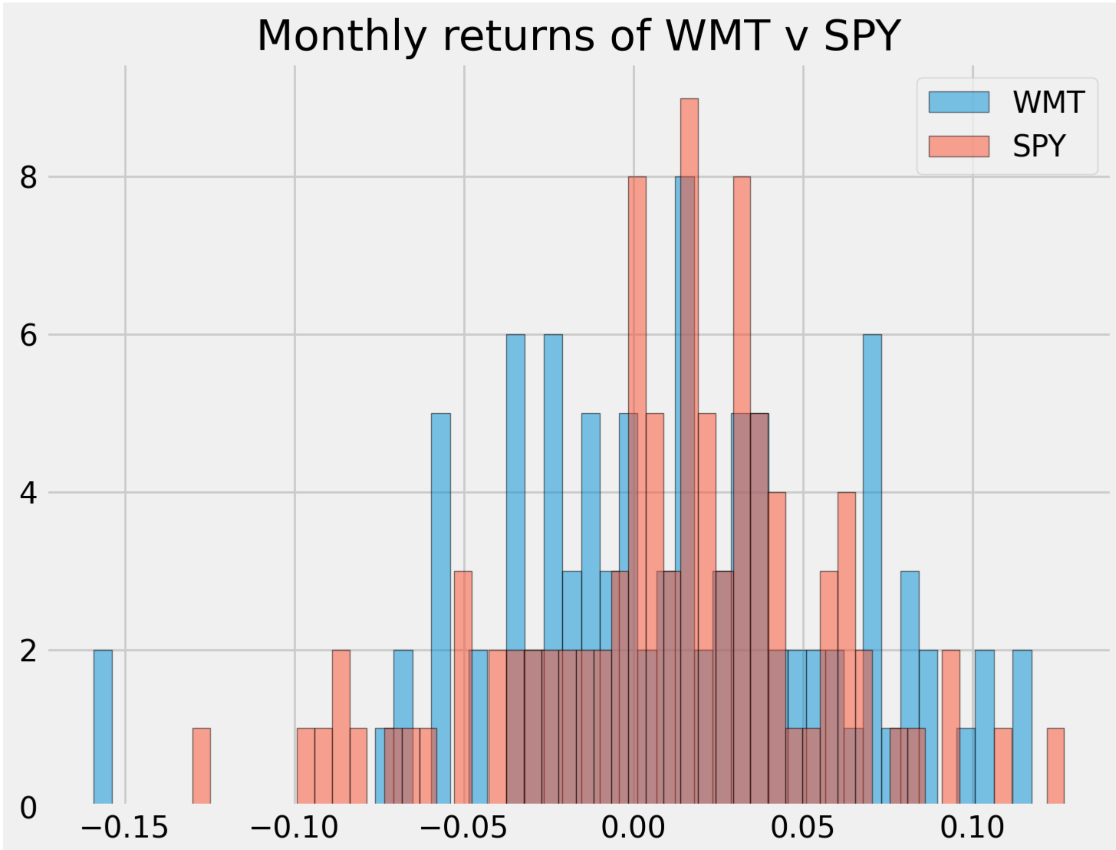

Monthly returns of TSLA v SPY
TSLA
SPY
0
2
4
6
8
−0.4
−0.2
0.0
0.2
0.4
0.6

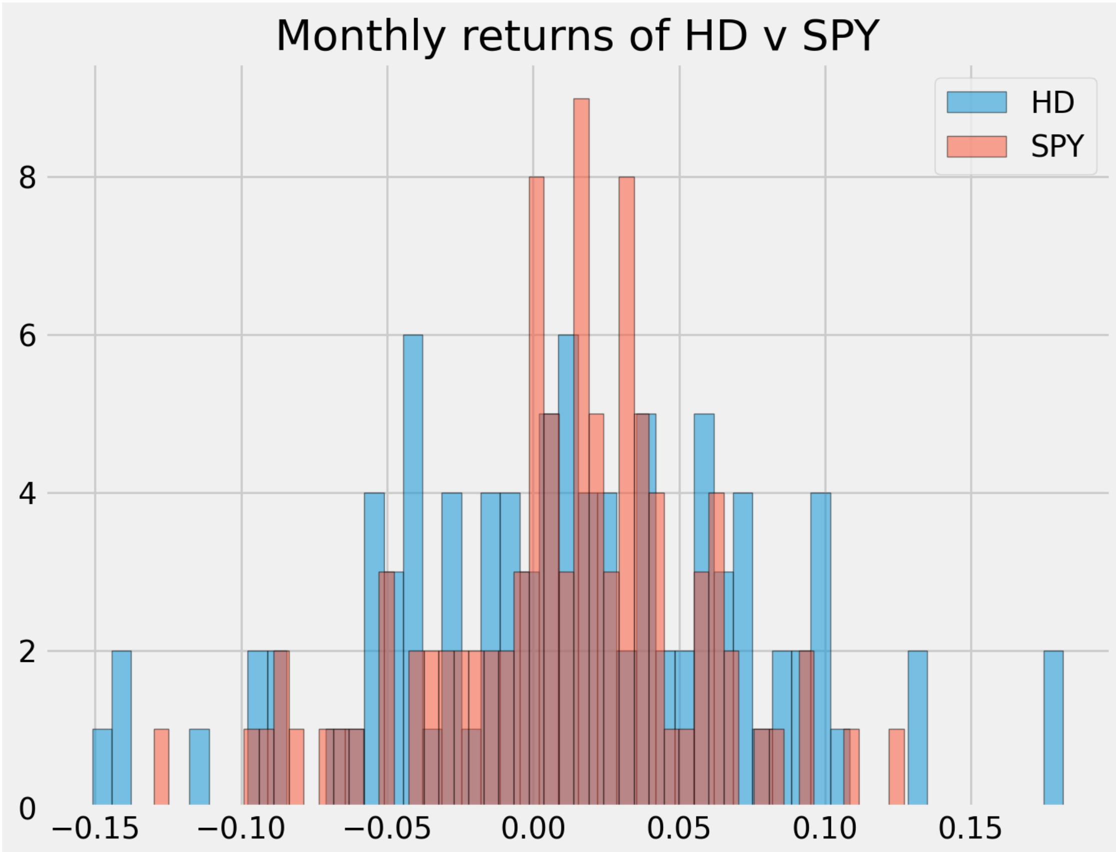


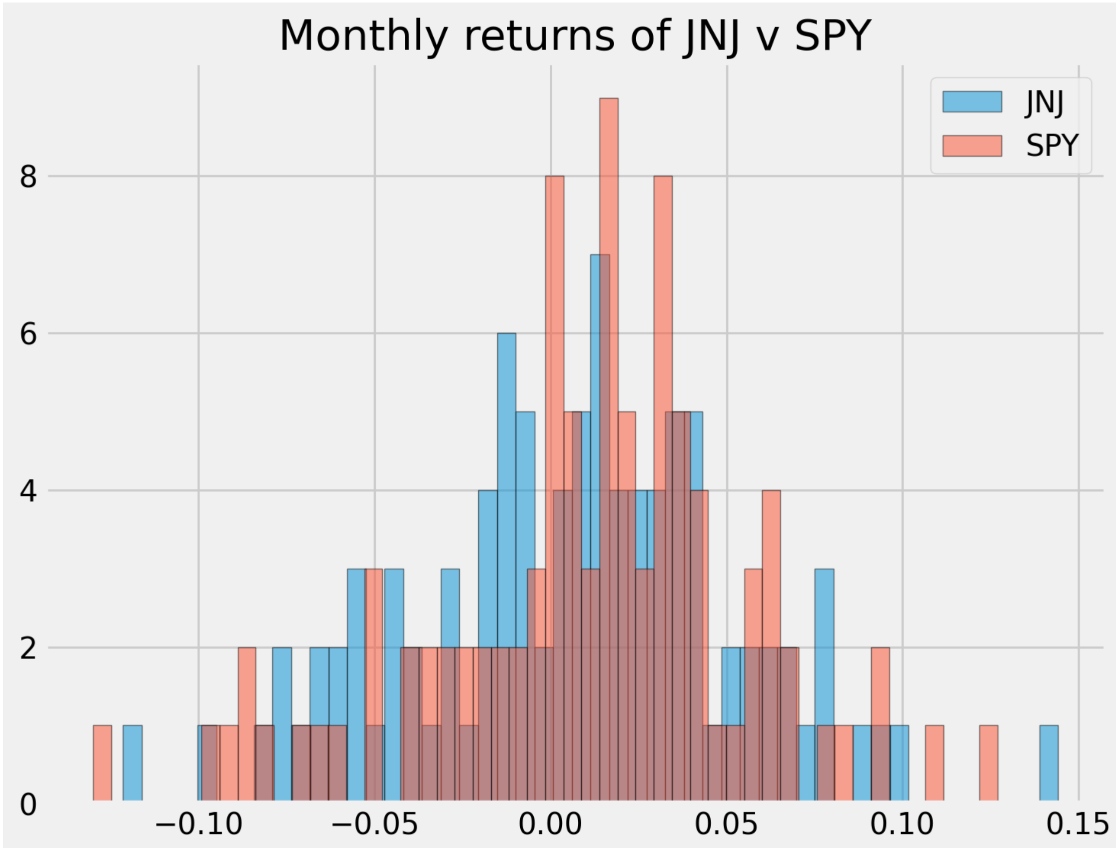
Monthly returns of AMZN v SPY
AMZN
SPY
10
8
6
4
2
0
−0.2
−0.1
0.0
0.1
0.2

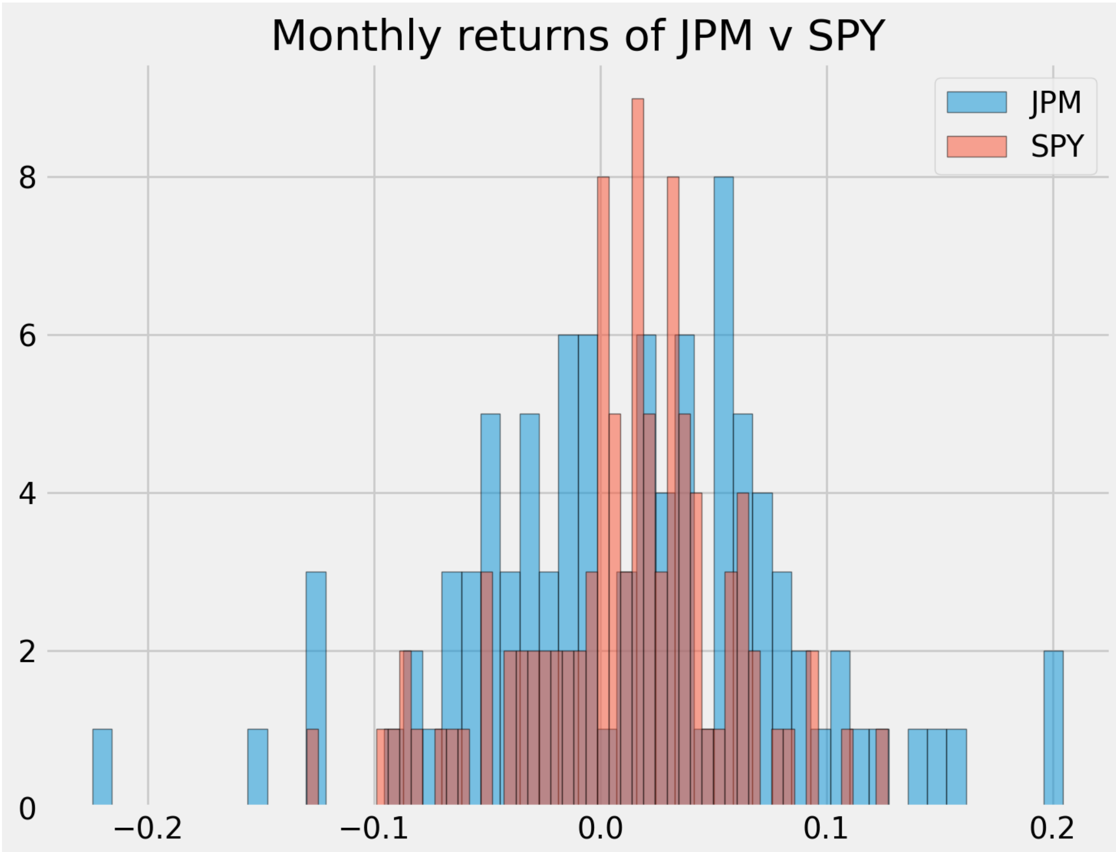

Monthly returns of GOOG v SPY
GOOG
SPY
0
2
4
6
8
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15


Monthly returns of MSFT v SPY
MSFT
SPY
0
2
4
6
8
−0.10
−0.05
0.00
0.05
0.10
0.15

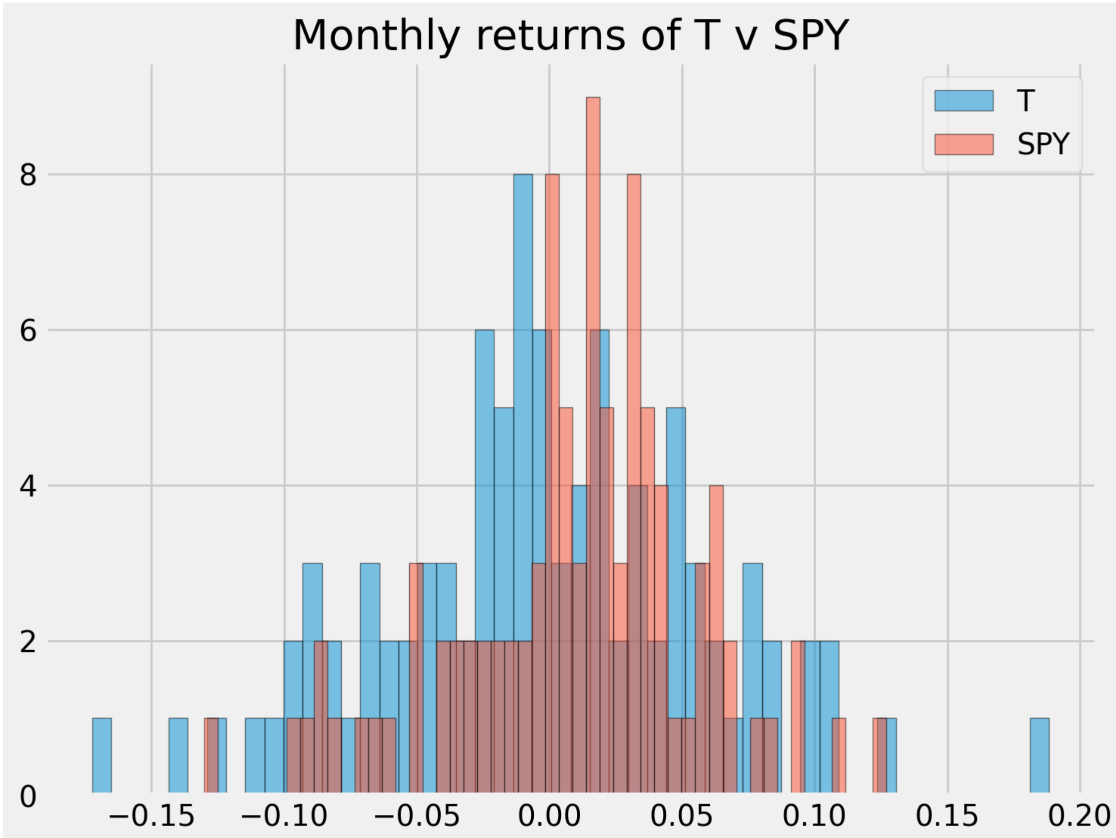

Monthly returns of WMT v SPY
WMT
SPY
8
6
4
2
0
−0.15
−0.10
−0.05
0.00
0.05
0.10

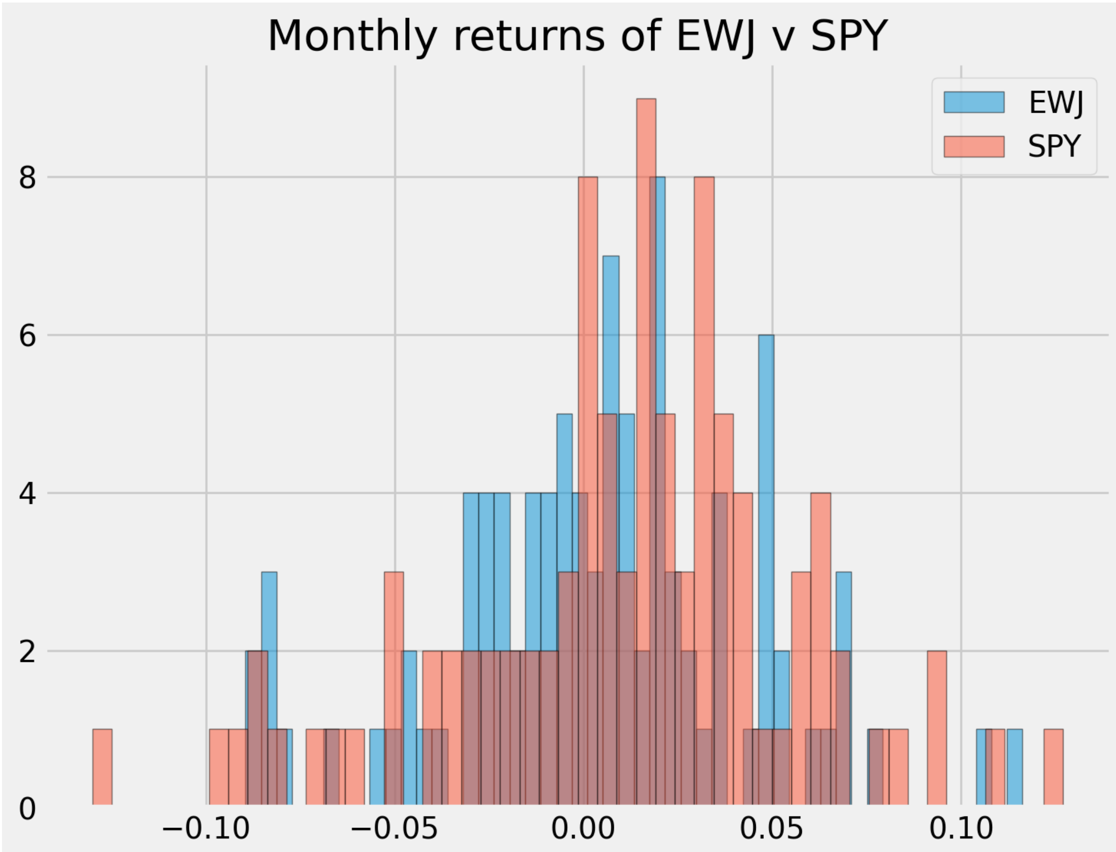

Monthly returns of HD v SPY
HD
SPY
0
2
4
6
8
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15


Monthly returns of JNJ v SPY
JNJ
SPY
0
2
4
6
8
−0.10
−0.05
0.00
0.05
0.10
0.15



Monthly returns of JPM v SPY
JPM
SPY
0
2
4
6
8
−0.2
−0.1
0.0
0.1
0.2


Monthly returns of T v SPY
T
SPY
8
6
4
2
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20



1.2.1 Calculate annualized stats.

[ ]:
```
for stock in stocks:
    mean = returns[stock].mean()
    annual_mean = (1 + mean)**252 - 1
    std_dev = returns[stock].std() * np.sqrt(252)
    print("----------------------------------------")
    print(f"\t{stock}\nMean: {annual_mean}\nStDv: {std_dev}")
    print("----------------------------------------")
```

```
----------------------------------------
        AAPL
Mean: 0.3394340998507035
StDv: 0.29330967234736355
----------------------------------------
----------------------------------------
        TSLA
Mean: 0.684416034306883
StDv: 0.5820732004229534
----------------------------------------
----------------------------------------
```

```
        AMZN
Mean: 0.2847541353191092
StDv: 0.3315069929104289
---------------------------------------
---------------------------------------
        GOOG
Mean: 0.23097755504379047
StDv: 0.2846952257452138
---------------------------------------
---------------------------------------
        MSFT
Mean: 0.3230444760283311
StDv: 0.2779885803854891
---------------------------------------
---------------------------------------
        WMT
Mean: 0.15171410242805528
StDv: 0.2157290208341175
---------------------------------------
---------------------------------------
        HD
Mean: 0.16676516193623447
StDv: 0.25304793947039567
---------------------------------------
---------------------------------------
        JNJ
Mean: 0.07577329293717039
StDv: 0.18598713604181938
---------------------------------------
---------------------------------------
        JPM
Mean: 0.1775640155514988
StDv: 0.284097688731136
---------------------------------------
---------------------------------------
        T
Mean: -0.02727400081767961
StDv: 0.2391886601239021
---------------------------------------
---------------------------------------
        SPY
Mean: 0.13310830964765064
StDv: 0.18470413196534724
---------------------------------------
---------------------------------------
        EWJ
Mean: 0.05293948001765014
StDv: 0.17021163985631338
```

1.2.2 Repreat the Mean /StDv Calculations using two other packages

Yfinance and Pandas-datareader

```
[ ]: import yfinance as yf
     yf.pdr_override()
     import warnings
     warnings.simplefilter(action='ignore', category=FutureWarning)
     import pandas as pd
     import pandas_datareader.data as web
     from datetime import datetime
```

```
[ ]: start = datetime(2022,1,1)
     end = datetime(2023,12,31)

     aapl = web.get_data_yahoo('AAPL', start=start, end=end)

     print(type(aapl))
     aapl.info()
     aapl
```

```
[*********************100%%**********************]  1 of 1 completed
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 501 entries, 2022-01-03 to 2023-12-29
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Open       501 non-null    float64
 1   High       501 non-null    float64
 2   Low        501 non-null    float64
 3   Close      501 non-null    float64
 4   Adj Close  501 non-null    float64
 5   Volume     501 non-null    int64
dtypes: float64(5), int64(1)
memory usage: 27.4 KB
```

```
[ ]:                   Open        High         Low       Close   Adj Close  \
      Date
      2022-01-03  177.830002  182.880005  177.710007  182.009995  179.953903
      2022-01-04  182.630005  182.940002  179.119995  179.699997  177.669983
      2022-01-05  179.610001  180.169998  174.639999  174.919998  172.944000
      2022-01-06  172.699997  175.300003  171.639999  172.000000  170.056961
      2022-01-07  172.889999  174.139999  171.029999  172.169998  170.225052
      …                  …           …           …           …           …
      2023-12-22  195.179993  195.410004  192.970001  193.600006  193.600006
```

```
2023-12-26  193.610001  193.889999  192.830002  193.050003  193.050003
2023-12-27  192.490005  193.500000  191.089996  193.149994  193.149994
2023-12-28  194.139999  194.660004  193.169998  193.580002  193.580002
2023-12-29  193.899994  194.399994  191.729996  192.529999  192.529999

               Volume
Date
2022-01-03  104487900
2022-01-04   99310400
2022-01-05   94537600
2022-01-06   96904000
2022-01-07   86709100
…                 …
2023-12-22   37122800
2023-12-26   28919300
2023-12-27   48087700
2023-12-28   34049900
2023-12-29   42628800

[501 rows x 6 columns]
```

[ ]:
```
# Get data for all stocks using pandas data reader
start = datetime(2014,1,1)
end = datetime(2023,12,31)

for ticker in stocks:
    # Get returns of stocks
    stock_data = web.get_data_yahoo(str(ticker), start=start, end=end)
    stock_data['Returns'] = stock_data['Adj Close'].pct_change()

    # Calculate annualzied mean and std
    mean_stock = stock_data['Returns'].mean()
    annual_mean_stock = (1 + mean_stock)**252 -1
    std_dev_stock = stock_data['Returns'].std() * np.sqrt(252)

    # Print results
    print("---------------------------------------")
    print(f"\t{ticker}\nMean: {annual_mean_stock}\nStDv: {std_dev_stock}")
    print("---------------------------------------")
```

```
[*********************100%%**********************]  1 of 1 completed
---------------------------------------
        AAPL
Mean: 0.3251997854827633
StDv: 0.2838052556524042
---------------------------------------
[*********************100%%**********************]  1 of 1 completed
---------------------------------------
```

```
        TSLA
Mean: 0.6096841091220577
StDv: 0.5566101755543822
---------------------------------------
[*********************100%***********************]  1 of 1 completed
---------------------------------------
        AMZN
Mean: 0.2949744203105513
StDv: 0.33171987111189244
---------------------------------------
[*********************100%***********************]  1 of 1 completed
---------------------------------------
        GOOG
Mean: 0.2236223612265349
StDv: 0.27941470292688925
---------------------------------------
[*********************100%***********************]  1 of 1 completed
---------------------------------------
        MSFT
Mean: 0.33067124157179717
StDv: 0.2706928682464443
---------------------------------------
[*********************100%***********************]  1 of 1 completed
---------------------------------------
        WMT
Mean: 0.11881081376626579
StDv: 0.20784625433162202
---------------------------------------
[*********************100%***********************]  1 of 1 completed
---------------------------------------
        HD
Mean: 0.21744566462596526
StDv: 0.24070071824570038
---------------------------------------
[*********************100%***********************]  1 of 1 completed
---------------------------------------
        JNJ
Mean: 0.1029046578843118
StDv: 0.18004821021137782
---------------------------------------
[*********************100%***********************]  1 of 1 completed
---------------------------------------
        JPM
Mean: 0.18641445389713374
StDv: 0.2693783907850355
---------------------------------------
[*********************100%***********************]  1 of 1 completed
---------------------------------------
```

```
        T
Mean: 0.056203672183316566
StDv: 0.22235064948381422
---------------------------------------
[*********************100%%**********************]  1 of 1 completed
---------------------------------------
        SPY
Mean: 0.13792709100991907
StDv: 0.17525680902827667
---------------------------------------
[*********************100%%**********************]  1 of 1 completed
---------------------------------------
        EWJ
Mean: 0.06159654989425212
StDv: 0.1689365130942255
---------------------------------------
```

[ ]:
```
# Get data for all stocks using yfinance
stocks = ['AAPL', 'TSLA', 'AMZN', 'GOOG', 'MSFT', 'WMT', 'HD', 'JNJ', 'JPM',␣
 ↪'T', 'SPY', 'EWJ']

start = datetime(2014,1,1)
end = datetime(2023,12,31)

yf_returns_dict = {}

for ticker_yf in stocks:
    # Get returns of stocks
    stock_data_yf = yf.download(str(ticker_yf), start=start, end=end,␣
 ↪progress=False)
    yf_returns_dict[(ticker_yf + ' Returns')] = stock_data_yf['Adj Close'].
 ↪pct_change()

yf_combined_returns = pd.DataFrame.from_dict(yf_returns_dict)
yf_combined_returns = yf_combined_returns.dropna(how='all')

print("---------------------------------------")
for ticker_yf in stocks:
    mean_stock = yf_combined_returns[(ticker_yf + ' Returns')].mean()
    annual_mean_stock = (1 + mean_stock) ** 252 - 1
    std_dev_stock = yf_combined_returns[(ticker_yf + ' Returns')].std() * np.
 ↪sqrt(252)
    print(f"{(ticker_yf + ' Returns')}\nMean: {annual_mean_stock}\nStDv:␣
 ↪{std_dev_stock}")
    print("---------------------------------------")
```

```
---------------------------------------
AAPL Returns
```

```
Mean: 0.3251998307524451
StDv: 0.283805259147745
---------------------------------------
TSLA Returns
Mean: 0.6096841091220577
StDv: 0.5566101755543822
---------------------------------------
AMZN Returns
Mean: 0.2949744203105513
StDv: 0.33171987111189244
---------------------------------------
GOOG Returns
Mean: 0.2236223612265349
StDv: 0.27941470292688925
---------------------------------------
MSFT Returns
Mean: 0.3306713159161627
StDv: 0.2706929604384544
---------------------------------------
WMT Returns
Mean: 0.11881081912923186
StDv: 0.20784624662546367
---------------------------------------
HD Returns
Mean: 0.21744564069913097
StDv: 0.24070068860329963
---------------------------------------
JNJ Returns
Mean: 0.10290468516949547
StDv: 0.18004816664578446
---------------------------------------
JPM Returns
Mean: 0.18641444736419488
StDv: 0.2693784326655183
---------------------------------------
T Returns
Mean: 0.05620368755216165
StDv: 0.2223505057934908
---------------------------------------
SPY Returns
Mean: 0.13792708053159997
StDv: 0.1752568116754903
---------------------------------------
EWJ Returns
Mean: 0.06159653714244406
StDv: 0.16893644305573868
---------------------------------------
```

1.2.3 Calculate and show differences in CSRP and Yfinance

(Only those two, since yfinance and pandas_datareader are the same)

[ ]:
```python
for stock in stocks:
    mean_crsp = returns[stock].mean()
    annual_mean_crsp = (1 + mean_crsp)**252 -1
    std_dev_crsp = returns[stock].std() * np.sqrt(252)

    mean_yf = yf_combined_returns[(stock + ' Returns')].mean()
    annual_mean_yf = (1 + mean_yf)**252 -1
    std_dev_yf = yf_combined_returns[(stock + ' Returns')].std() * np.sqrt(252)

    diff_mean_crsp_yf = annual_mean_crsp - annual_mean_yf
    diff_std_crsp_yf = std_dev_crsp - std_dev_yf

    print("---------------------------------------")
    print(f"Mean and Std Difference for {stock} between CRSP & Yfinance:")
    print(f"CRSP - Yf Mean: {diff_mean_crsp_yf}")
    print(f"CRSP - Yf Std: {diff_std_crsp_yf}")
    print("---------------------------------------")
```

```
---------------------------------------
Mean and Std Difference for AAPL between CRSP & Yfinance:
CRSP - Yf Mean: 0.014234269098258423
CRSP - Yf Std: 0.009504413199618533
---------------------------------------
---------------------------------------
Mean and Std Difference for TSLA between CRSP & Yfinance:
CRSP - Yf Mean: 0.0747319251848253
CRSP - Yf Std: 0.025463024868571216
---------------------------------------
---------------------------------------
Mean and Std Difference for AMZN between CRSP & Yfinance:
CRSP - Yf Mean: -0.010220284991442119
CRSP - Yf Std: -0.00021287820146353997
---------------------------------------
---------------------------------------
Mean and Std Difference for GOOG between CRSP & Yfinance:
CRSP - Yf Mean: 0.007355193817255579
CRSP - Yf Std: 0.005280522818324529
---------------------------------------
---------------------------------------
Mean and Std Difference for MSFT between CRSP & Yfinance:
CRSP - Yf Mean: -0.007626839887831638
CRSP - Yf Std: 0.007295619947034704
---------------------------------------
---------------------------------------
Mean and Std Difference for WMT between CRSP & Yfinance:
```

```
CRSP - Yf Mean: 0.032903283298823416
CRSP - Yf Std: 0.007882774208653825
---------------------------------------
---------------------------------------
Mean and Std Difference for HD between CRSP & Yfinance:
CRSP - Yf Mean: -0.0506804787628965
CRSP - Yf Std: 0.012347250867096038
---------------------------------------
---------------------------------------
Mean and Std Difference for JNJ between CRSP & Yfinance:
CRSP - Yf Mean: -0.02713139223232508
CRSP - Yf Std: 0.0059389693960349155
---------------------------------------
---------------------------------------
Mean and Std Difference for JPM between CRSP & Yfinance:
CRSP - Yf Mean: -0.008850431812696069
CRSP - Yf Std: 0.014719256065617692
---------------------------------------
---------------------------------------
Mean and Std Difference for T between CRSP & Yfinance:
CRSP - Yf Mean: -0.08347768836984126
CRSP - Yf Std: 0.016838154330411303
---------------------------------------
---------------------------------------
Mean and Std Difference for SPY between CRSP & Yfinance:
CRSP - Yf Mean: -0.00481877088394933
CRSP - Yf Std: 0.009447320289856953
---------------------------------------
---------------------------------------
Mean and Std Difference for EWJ between CRSP & Yfinance:
CRSP - Yf Mean: -0.008657057124793921
CRSP - Yf Std: 0.001275196800574696
---------------------------------------
```