

HW 4

```
In [ ]: import pandas as pd
import numpy as np
from joblib import Parallel, delayed
import quandl
import yfinance as yf
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from statsmodels.stats.sandwich_covariance import cov_hac
quandl.ApiConfig.api_key = "vBX2yLsVUpn1kpJE_Hss"

import warnings
warnings.filterwarnings('ignore')
```

1. Salience Theory

What does it mean?

Salience theory is referring to the behavioral bias that decision maker's attention is directed to the most salient payoffs of the lotteries available for choice - investors are attracted to stocks with salient upsides, which leads to an excess demand for these stocks resulting in overvaluation and lower future returns. On the other hand, stocks with salient downsides become undervalued and earn higher subsequent returns.

How does it affect investor's portfolio choice decisions?

Salient-thinking leads investors to attach more weight to a 5% stock return on a day when the market is flat than on a day when the market is also up by 5%. When evaluating a stock, they do not think about its contribution to the return of their portfolio. The salience of a stock's return is therefore determined only by its relative difference from the market return and does not depend on investor-specific characteristics. Consequently, salience-driven demand for stocks will be correlated across investors and can exert pressure on prices, given limits to arbitrage that prevent rational investors from correcting mispricing.

2. Replicate Table 2

```
In [ ]: df1 = pd.read_csv('crspData1.csv')
df2 = pd.read_csv('crspData2.csv')
```

```
In [ ]: rf = quandl.get('FRED/DTB3')
rf.columns = ['rf']
```

```
In [ ]: ticker_symbol = '^GSPC'

start_date = '1926-01-01'
end_date = '2023-11-15'

market_data = yf.download(ticker_symbol, start=start_date, end=end_date)

market_returns = pd.DataFrame(market_data['Adj Close'].pct_change())
market_returns.columns = ['mkt']
```

```
[*****100%*****] 1 of 1 completed
```

```
In [ ]: ff = pd.read_csv('F-F_Research_Data_Factors_daily.csv', skiprows=3)
ff.rename(columns={'Unnamed: 0': 'date'}, inplace=True)
ff=ff[:-1]
ff.index = pd.to_datetime(ff['date'])
ff.drop('date', axis=1, inplace=True)
```

```
In [ ]: ff
```

Out[]:

Mkt-RF SMB HML RF

date

date	Mkt-RF	SMB	HML	RF
1926-07-01	0.10	-0.25	-0.27	0.009
1926-07-02	0.45	-0.33	-0.06	0.009
1926-07-06	0.17	0.30	-0.39	0.009
1926-07-07	0.09	-0.58	0.02	0.009
1926-07-08	0.21	-0.38	0.19	0.009
...
2023-09-25	0.39	-0.11	0.35	0.021
2023-09-26	-1.46	0.39	-0.50	0.021
2023-09-27	0.13	0.69	0.12	0.021
2023-09-28	0.65	0.16	0.03	0.021
2023-09-29	-0.27	-0.03	0.30	0.021

25586 rows × 4 columns

In []: df1.head()

Out[]:

	PERMNO	date	TICKER	COMNAME	PERMCO	ACPERM	ACCOMP	NWPERM	PRC	VOL
0	10009	1986-01-17	CAYB	CAYUGA SAVINGS BANK NY	7965	NaN	NaN	NaN	-13.7500	36600.0
1	10009	1986-01-20	CAYB	CAYUGA SAVINGS BANK NY	7965	NaN	NaN	NaN	-14.2500	15900.0
2	10009	1986-01-21	CAYB	CAYUGA SAVINGS BANK NY	7965	NaN	NaN	NaN	-13.8750	22650.0
3	10009	1986-01-22	CAYB	CAYUGA SAVINGS BANK NY	7965	NaN	NaN	NaN	-13.3125	20600.0
4	10009	1986-01-23	CAYB	CAYUGA SAVINGS BANK NY	7965	NaN	NaN	NaN	-13.1875	17400.0

In []: df2.head()

```
Out[ ]:   gvkey  datadate  fyear  indfmt  consol  popsrc  datafmt  tic      conn  curcd  bkvlps  costat  mkvalt
0    1004  1966-05-31  1965.0    INDL      C       D     STD    AIR  AAR CORP    USD  3.4000      A     NaN
1    1004  1967-05-31  1966.0    INDL      C       D     STD    AIR  AAR CORP    USD  4.9673      A     NaN
2    1004  1968-05-31  1967.0    INDL      C       D     STD    AIR  AAR CORP    USD  5.0018      A     NaN
3    1004  1969-05-31  1968.0    INDL      C       D     STD    AIR  AAR CORP    USD  6.8953      A     NaN
4    1004  1970-05-31  1969.0    INDL      C       D     STD    AIR  AAR CORP    USD  7.0617      A     NaN
```

```
In [ ]: rf.head()
```

```
Out[ ]:          rf
```

Date

```
1954-01-04  1.33
1954-01-05  1.28
1954-01-06  1.28
1954-01-07  1.31
1954-01-08  1.31
```

```
In [ ]: market_returns.head()
```

```
Out[ ]:          mkt
```

Date

```
1927-12-30      NaN
1928-01-03  0.005663
1928-01-04 -0.002252
1928-01-05 -0.009594
1928-01-06  0.006268
```

```
In [ ]: PRC_df = df1.pivot_table(index='date', columns='PERMNO', values='PRC').abs()
PRC_df.index=pd.to_datetime(PRC_df.index)
ret_df = PRC_df.pct_change()
```

- ST: decile the time-series average of these monthly characteristics

```
In [ ]: def calculate_salience_weights(return_df, theta=0.1, delta=0.7):

    average_daily_returns = return_df.mean(axis=1)

    salience_df = abs(return_df.subtract(average_daily_returns, axis=0)) / \
                  (abs(return_df).add(abs(average_daily_returns), axis=0) + theta)

    salience_rank_df = salience_df.rank(axis=1, method='min', ascending=False)
    salience_prob = salience_rank_df.div(salience_rank_df.max(), axis=1)

    numerator = delta ** salience_rank_df
    denominator = delta ** salience_rank_df * salience_prob

    decision_weight_df = numerator.div(denominator.sum(axis=1), axis=0)
    norm_decision_df = decision_weight_df.div(salience_rank_df.sum(axis=1), axis=0)

    return norm_decision_df
```

```
In [ ]: grouped_ret = ret_df.groupby(pd.Grouper(freq='M'))
```

```
In [ ]: period_list = PRC_df.index.to_period('M').strftime('%Y%m').unique().tolist()
ST_df = pd.DataFrame(index=period_list, columns=PRC_df.columns)
decile_df = pd.DataFrame(index=period_list, columns=PRC_df.columns)
```

```
In [ ]: def calculate_covariance(column, decision_weight_df, ret_df):
    temp_df = pd.concat([decision_weight_df[column], ret_df[column]], axis=1).dropna()
    return np.cov(temp_df.iloc[:, 0], temp_df.iloc[:, 1])[0][1] if len(temp_df) > 1 else np.nan

def calculate_covariances(decision_weight_df, ret_df):
    return Parallel(n_jobs=-1)(delayed(calculate_covariance))(col, decision_weight_df, ret_df) for col in decision_weight_df.columns
```

```
In [ ]: for name, group in grouped_ret:
    cur_df = group.dropna(axis=1, how='all')
    decision_weight_df = calculate_salience_weights(cur_df, theta=0.1, delta=0.7)
    covariance_list = calculate_covariances(decision_weight_df, cur_df)
    ST_values_df = pd.DataFrame(covariance_list, index=decision_weight_df.columns, columns=['ST'])
    ST_values_df['Decile'] = pd.qcut(ST_values_df['ST'], 10, duplicates='drop', labels=False) + 1
    period = name.strftime('%Y%m')
    for key in ST_values_df.index:
```

```
ST_df.loc[period, key] = ST_values_df.loc[key, 'ST']
decile_df.loc[period, key] = ST_values_df.loc[key, 'Decile']
```

```
In [ ]: ST_df.tail()
```

```
Out[ ]: PERMNO 10009 10026 10028 10029 10032 10036 10038 10044 10057 10064 ... 93396 93397
```

	date	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397
201508		-0.38421	0.85452	-1.036563	-0.539591	0.433431	-1.117719	-0.584107	-0.678588	-1.509827	0.238169	...	-2.309222	-0.753522
201509		-0.38421	0.85452	-2.445425	-0.539591	0.433431	-1.117719	-0.584107	-0.678588	-1.509827	0.238169	...	-2.307898	-0.892413
201510		-0.38421	0.85452	6.546581	-0.539591	0.432432	-1.117719	-0.584107	-0.678588	-1.509827	0.238169	...	-2.309221	-0.753522
201511		-0.38421	0.85452	-1.029012	-0.539591	0.433431	-1.117719	-0.584107	-0.678588	-1.509827	0.238169	...	-2.290463	-0.753522
201512		-0.38421	0.85452	-1.022346	-0.539591	0.433431	-1.117719	-0.584107	-0.678588	-1.509827	0.238169	...	-0.608660	-0.753522

5 rows × 7217 columns

```
In [ ]: decile_df.tail()
```

```
Out[ ]: PERMNO 10009 10026 10028 10029 10032 10036 10038 10044 10057 10064 ... 93396 93397 93415 93419 93423 93426 934
```

	date	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397	93415	93419	93423	93426	934
201508		3.0	5.0	6.0	3.0	3.0	3.0	3.0	1.0	3.0	3.0	...	1.0	5.0	1.0	2.0	5.0	2.0	...
201509		3.0	4.0	1.0	3.0	5.0	3.0	3.0	2.0	3.0	3.0	...	6.0	1.0	1.0	5.0	5.0	6.0	...
201510		3.0	4.0	6.0	3.0	1.0	3.0	3.0	5.0	3.0	3.0	...	6.0	2.0	2.0	2.0	4.0	5.0	...
201511		3.0	1.0	6.0	3.0	5.0	3.0	3.0	3.0	3.0	3.0	...	6.0	6.0	5.0	5.0	3.0	5.0	...
201512		3.0	5.0	6.0	3.0	3.0	3.0	3.0	5.0	3.0	3.0	...	6.0	1.0	1.0	5.0	4.0	5.0	...

5 rows × 7217 columns

```
In [ ]: # ST_df.to_csv('ST.csv')
# decile_df.to_csv('decile.csv')
```

- PRICE: the stock price (in \$)

```
In [ ]: price_M = PRC_df.resample('M').last()
price_M.index = pd.to_datetime(price_M.index).strftime('%Y%m')
```

- BM: the book-to- market ratio

```
In [ ]: BM_df = df2.pivot_table(index='datadate', columns='gvkey', values='bkvlps')
BM_df.index = pd.to_datetime(BM_df.index).strftime('%Y%m')
```

- ME: the log of a firm's market capitalization (in \$)

```
In [ ]: ME_df = df2.pivot_table(index='datadate', columns='gvkey', values='mkvalt')
ME_df.index = pd.to_datetime(ME_df.index).strftime('%Y%m')
```

- MOM: a stock's cumulative return (in %) over the 11-month period ending two months prior to the current month

```
In [ ]: ret_M = price_M.pct_change()
mom = ret_M.rolling(window=11, min_periods=1).apply(lambda x: (x + 1).prod() - 1, raw=True)
MOM_df = mom = mom.shift(2)
```

- ILLIQ: the Amihud (2002) illiquidity measure, averaged over all trading days in a month

```
In [ ]: VOL_df = df1.pivot_table(index='date', columns='PERMNO', values='VOL')
VOL_df.index = pd.to_datetime(VOL_df.index)
dollar_VOL_df = PRC_df.multiply(VOL_df)
illiq_daily = ret_df.abs().div(dollar_VOL_df)
ILLIQ_df = illiq_daily.resample('M').mean()
```

```
In [ ]: ILLIQ_df.tail()
```

Out[]: PERMNO 10009 10026 10028 10029 10032 10036 10038 10044 10057 10064 ... 93396 93397 93415

	date	PERMNO	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397	93415
201508	NaN	1.159407e-09	0.000013	NaN	1.782756e-09	NaN	NaN	1.503839e-07	NaN	NaN	...	0.000010	3.143938e-08	3.455297e-09	E	
201509	NaN	1.429761e-09		NaN	NaN	2.636822e-09	NaN	NaN	1.585810e-07	NaN	NaN	...	NaN	2.090545e-08	5.167031e-09	
201510	NaN	1.251335e-09	0.000049	NaN	2.372137e-09	NaN	NaN	7.067061e-08	NaN	NaN	...	0.000022	1.696294e-08	3.481935e-09		
201511	NaN	1.077951e-09		NaN	NaN	1.917467e-09	NaN	NaN	1.056616e-06	NaN	NaN	...	0.000003	3.592499e-08	2.864559e-09	
201512	NaN	1.496027e-09	0.000011	NaN	2.768743e-09	NaN	NaN	1.385248e-07	NaN	NaN	...	0.000002	3.693790e-08	2.835958e-09		

5 rows × 7217 columns

- BETA: the market beta, estimated from a regression of daily excess stock returns on the daily excess market return over a one-month window
- IVOL: the idiosyncratic volatility (in %) obtained from this regression

In []: df_reg = ret_df.merge(rf, how='inner', left_index=True, right_index=True)
reg_df = df_reg.merge(market_returns, how='inner', left_index=True, right_index=True)
reg_df['mkt'] = reg_df['mkt'].sub(reg_df['rf'], axis=0)
reg_df.iloc[:, :-2] = reg_df.iloc[:, :-2].sub(reg_df['rf'], axis=0)
reg_df.drop('rf', axis=1, inplace=True)

In []: group_columns = reg_df.iloc[:, :-1].columns

In []: beta_df = pd.DataFrame()
ivol_df = pd.DataFrame()

def regression_analysis(stock_returns, market_returns):
 if len(stock_returns) > 5 and len(market_returns) > 5:
 X = sm.add_constant(market_returns)
 model = sm.OLS(stock_returns, X).fit()
 beta = model.params[1]
 residuals = model.resid
 ivol = np.std(residuals)

```

        return beta, ivol
    else:
        return np.nan, np.nan

for month, group in reg_df.groupby(pd.Grouper(freq='M')):
    betas = []
    ivols = []
    for column in group_columns:
        beta, ivol = regression_analysis(group[column], group['mkt'])
        betas.append(beta)
        ivols.append(ivol)
    beta_df[month] = betas
    ivol_df[month] = ivols

beta_df= beta_df.T
ivol_df= ivol_df.T
beta_df.index = pd.to_datetime(beta_df.index).strftime('%Y%m')
ivol_df.index = pd.to_datetime(ivol_df.index).strftime('%Y%m')
beta_df.columns = MOM_df.columns
ivol_df.columns = MOM_df.columns

```

In []: # beta_df.to_csv('betas.csv')
ivol_df.to_csv('ivol.csv')

In []: beta_df.tail()

Out[]:

	date	10009	10026	10028	10029	10032	10036	10038	10044	10057	...	93396	93397	93415
739	201508	0.789675	0.957964	1.201376	0.789675	1.000799	0.789675	0.789675	0.626389	0.789675	...	0.837181	0.986023	0.775455
740	201509	0.858696	0.928311	0.483508	0.858696	1.056992	0.858696	0.858696	0.722550	0.858696	...	0.798310	1.049245	0.871610
741	201510	0.839465	1.047803	1.159167	0.839465	1.299271	0.839465	0.839465	0.884752	0.839465	...	1.216790	1.119804	0.920796
742	201511	0.950650	0.908637	0.584735	0.950650	1.123927	0.950650	0.950650	0.964420	0.950650	...	0.842401	0.987776	0.824148
743	201512	0.926605	0.962017	0.922854	0.926605	1.088075	0.926605	0.926605	1.224763	0.926605	...	1.943523	0.990653	1.952743

5 rows × 7218 columns

In []: ivol_df.tail()

Out[]:

	date	10009	10026	10028	10029	10032	10036	10038	10044	10057	...	93396	93397	93415
739	201508	0.015625	0.008936	0.059467	0.015625	0.009512	0.015625	0.015625	0.025601	0.015625	...	0.049035	0.016814	0.028547
740	201509	0.013408	0.009757	0.087188	0.013408	0.012244	0.013408	0.013408	0.016215	0.013408	...	0.042832	0.033444	0.028213
741	201510	0.006685	0.005175	0.130684	0.006685	0.037965	0.006685	0.006685	0.016126	0.006685	...	0.041997	0.018645	0.020981
742	201511	0.007282	0.015871	0.076660	0.007282	0.010167	0.007282	0.007282	0.011177	0.007282	...	0.052289	0.022100	0.018264
743	201512	0.011121	0.008386	0.049875	0.011121	0.008621	0.011121	0.011121	0.020238	0.011121	...	0.106285	0.023618	0.044138

5 rows × 7218 columns

- REV: the stock return over the previous month (in %)

In []:

```
REV_df = ret_M.shift(1)
REV_df
```

Out[]:

PERMNO	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397	93415
	date													
192601	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
192602	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
192603	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-0.132275	NaN	...	NaN	NaN
192604	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-0.097561	NaN	...	NaN	NaN
192605	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.000000	NaN	...	NaN	NaN
...
201508	0.0	0.069486	-0.149317	0.0	-0.130811	0.0	0.0	-0.014427	0.000000	0.0	...	0.216450	-0.073774	-0.033534
201509	0.0	-0.037175	-0.298158	0.0	-0.001835	0.0	0.0	-0.077812	0.000000	0.0	...	-0.289146	-0.078193	-0.041815
201510	0.0	-0.002633	0.312336	0.0	0.013396	0.0	0.0	-0.060150	0.000000	0.0	...	-0.068836	-0.118546	-0.195915
201511	0.0	0.080327	0.028571	0.0	-0.102644	0.0	0.0	-0.014222	0.000000	0.0	...	-0.056452	-0.052002	0.144919
201512	0.0	-0.049760	0.021111	0.0	0.074523	0.0	0.0	-0.011722	0.000000	0.0	...	-0.148490	0.073770	-0.074130

1080 rows × 7217 columns

- MAX & MIN: a stock's maximum (minimum) daily return within a month (in %), as in Bali et al. (2011)

```
In [ ]: Max_df = PRC_df.resample('M').max()
Max_df.index = pd.to_datetime(Max_df.index).strftime('%Y%m')
```

```
In [ ]: Min_df = PRC_df.resample('M').min()
Min_df.index = pd.to_datetime(Min_df.index).strftime('%Y%m')
```

- TK: the prospect theory value of a stock, constructed using a five-year window of monthly returns, as in Barberis et al. (2016)

```
In [ ]: def calculate_tk(gain_loss, lambda_=2.25, alpha=0.88, beta=0.88):
    """
    Calculate TK value for a series of gains and losses.
    :param gain_loss: A Pandas Series of gains and losses.
    :param lambda_: Coefficient for loss aversion (typically > 1).
    :param alpha: Coefficient for diminishing sensitivity to gains.
    :param beta: Coefficient for diminishing sensitivity to losses.
    :return: TK value.
    """
    weighted_gains_losses = gain_loss.apply(lambda x: x**alpha if x >= 0 else -lambda_*abs(x)**beta)
    tk_value = weighted_gains_losses.sum()
    return tk_value

tk_values = ret_M.rolling(window=60, min_periods=60).apply(calculate_tk, raw=False)
```

```
In [ ]: tk_values
```

Out[]: PERMNO 10009 10026 10028 10029 10032 10036 10038 10044 10057 10064 ... 93396 93397 93415 9

	date	PERMNO	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397	93415	9
192601		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
192602		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
192603		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
192604		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
192605		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
...
201508		0.0	0.488856	-7.897477	0.0	-2.227256	0.0	0.0	-1.499669	0.0	0.0	...	-1.141606	-2.721190	-1.591522	-0.93	
201509		0.0	0.332580	-7.959257	0.0	-2.526262	0.0	0.0	-1.634804	0.0	0.0	...	-1.419714	-3.336439	-2.157416	-0.89	
201510		0.0	0.405931	-8.037565	0.0	-2.880870	0.0	0.0	-1.694807	0.0	0.0	...	-2.062214	-3.512657	-2.016325	-0.84	
201511		0.0	0.152556	-8.024943	0.0	-2.467067	0.0	0.0	-1.751041	0.0	0.0	...	-2.820000	-3.517214	-2.270185	-0.68	
201512		0.0	0.074781	-8.391391	0.0	-2.837361	0.0	0.0	-1.838433	0.0	0.0	...	-2.084789	-4.235146	-2.801986	-0.69	

1080 rows × 7217 columns

- SKEW: the skewness of daily stock returns, calculated over a one-year window

```
In [ ]: skew_df = ret_df.rolling(window=252, min_periods=252).skew()
SKEW_df = skew_df.resample('M').last()
SKEW_df.index = pd.to_datetime(SKEW_df.index).strftime('%Y%m')
```

```
In [ ]: SKEW_df.tail()
```

Out[]: PERMNO 10009 10026 10028 10029 10032 10036 10038 10044 10057 10064 ... 93396 93397 93415 93

	date	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397	93415	93
201508	0.0	-3.338193	-0.072577	0.0	-0.392391	0.0	0.0	0.184850	0.0	0.0	...	5.125254	-0.166139	-0.736083	-0.50'	
201509	0.0	-3.177815	-0.073294	0.0	-0.349498	0.0	0.0	0.136690	0.0	0.0	...	5.114800	-0.989990	-0.754296	-0.456	
201510	0.0	-3.188924	1.241365	0.0	-2.742430	0.0	0.0	0.196931	0.0	0.0	...	5.157664	-0.920797	-0.716298	-0.544	
201511	0.0	-3.127722	1.178957	0.0	-2.667381	0.0	0.0	0.237323	0.0	0.0	...	5.083206	-0.715634	-0.589164	-0.398	
201512	0.0	-3.270903	1.178219	0.0	-2.915193	0.0	0.0	0.219069	0.0	0.0	...	4.986837	-0.709800	-0.829244	-0.321	

5 rows × 7217 columns

- COSKEW: the coskewness of daily stock returns with daily market returns over a one-year window, calculated following Harvey and Siddique (2000)

In []: reg_df

Out[]:

	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93397	93415	93419
1954-01-04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-1.330000	NaN	...	NaN	NaN	NaN
1954-01-05	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-1.288197	NaN	...	NaN	NaN	NaN
1954-01-06	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-1.280000	NaN	...	NaN	NaN	NaN
1954-01-07	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-1.310000	NaN	...	NaN	NaN	NaN
1954-01-08	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-1.305868	NaN	...	NaN	NaN	NaN
...
2015-12-24	-0.20	-0.180069	-0.164992	-0.20	-0.197218	-0.20	-0.20	-0.205556	-0.200000	-0.20	...	-0.173826	-0.203324	-0.197107
2015-12-28	-0.23	-0.234822	-0.288824	-0.23	-0.240541	-0.23	-0.23	-0.269106	-0.230000	-0.23	...	-0.235886	-0.254683	-0.218821
2015-12-29	-0.23	-0.215294	-0.192187	-0.23	-0.222991	-0.23	-0.23	-0.212558	-0.230000	-0.23	...	-0.231316	-0.260096	-0.218588
2015-12-30	-0.21	-0.211005	-0.281364	-0.21	-0.222806	-0.21	-0.21	-0.221429	-0.210000	-0.21	...	-0.209341	-0.229041	-0.212116
2015-12-31	-0.16	-0.181635	-0.089961	-0.16	-0.175228	-0.16	-0.16	-0.124355	-0.160000	-0.16	...	-0.176458	-0.068699	-0.165654

15438 rows x 7218 columns

In []:

```
def calculate_coskew(excess_returns):
    market_squared = excess_returns['mkt'] ** 2
    coskew_values = {}

    for column in excess_returns.columns:
        if column != 'mkt':
            cov_with_squared_market = excess_returns[column].cov(market_squared)
            market_variance = excess_returns['mkt'].var()
            coskew = cov_with_squared_market / market_variance ** (3 / 2)
            coskew_values[column] = coskew
```

```

    return pd.Series(coskew_values)

coskew_df = pd.DataFrame()

for month in reg_df.resample('M').indices:
    start_date = month - pd.DateOffset(months=11)
    window_data = reg_df[start_date:month]
    coskew_values = calculate_coskew(window_data)
    coskew_df[month.strftime('%Y%m')] = coskew_values

```

In []:

```

coskew_df = coskew_df.T
coskew_df.index = beta_df['date']
coskew_df.columns = PRC_df.columns

```

In []:

```
coskew_df
```

Out[]:

PERMNO	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	5
date													
195401	NaN	-16.749747	NaN	...	NaN								
195402	NaN	-15.189075	NaN	...	NaN								
195403	NaN	-19.106283	NaN	...	NaN								
195404	NaN	-18.280069	NaN	...	NaN								
195405	NaN	-12.991376	NaN	...	NaN								
...	
201508	-3.674284	-4.026450	-4.561117	-3.674284	-4.057887	-3.674284	-3.674284	-3.815945	-3.674284	-3.674284	...	-4.425116	-4.1
201509	-3.499383	-3.790520	-4.197160	-3.499383	-3.782316	-3.499383	-3.499383	-3.635155	-3.499383	-3.499383	...	-4.311348	-3.9
201510	-3.059994	-3.339612	-3.703983	-3.059994	-3.385478	-3.059994	-3.059994	-3.107678	-3.059994	-3.059994	...	-3.792292	-3.5
201511	-3.722662	-3.838305	-3.353030	-3.722662	-3.840984	-3.722662	-3.722662	-3.737246	-3.722662	-3.722662	...	-4.114807	-3.7
201512	-3.295576	-3.332125	-3.219952	-3.295576	-3.373476	-3.295576	-3.295576	-3.328792	-3.295576	-3.295576	...	-3.331621	-3.36

744 rows × 7217 columns

- ISKEW: the skewness of the residuals from a Fama and French (1993) three-factor model regression estimated over a one-year window of daily returns, as in Boyer et al. (2010)

```
In [ ]: ff_reg = ret_df.merge(ff, how='inner', left_index=True, right_index=True)
```

```
In [ ]: ff_reg
```

```
Out[ ]:   10009  10026  10028  10029  10032  10036  10038  10044  10057  10064 ...  93423  93426  93427
```

date														
1926-07-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-0.007752	NaN	...	NaN	NaN	NaN
1926-07-02	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.000000	NaN	...	NaN	NaN	NaN
1926-07-06	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.031250	NaN	...	NaN	NaN	NaN
1926-07-07	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-0.053030	NaN	...	NaN	NaN	NaN
1926-07-08	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-0.024000	NaN	...	NaN	NaN	NaN
...
2015-12-24	0.0	0.019931	0.035008	0.0	0.002782	0.0	0.0	-0.005556	0.000000	0.0	...	0.011479	0.024348	-0.013131
2015-12-28	0.0	-0.004822	-0.058824	0.0	-0.010541	0.0	0.0	-0.039106	0.000000	0.0	...	0.006041	-0.012733	-0.009148
2015-12-29	0.0	0.014706	0.037813	0.0	0.007009	0.0	0.0	0.017442	0.000000	0.0	...	-0.004003	0.011178	0.015107
2015-12-30	0.0	-0.001005	-0.071364	0.0	-0.012806	0.0	0.0	-0.011429	0.000000	0.0	...	0.002192	-0.014456	-0.002480
2015-12-31	0.0	-0.021635	0.070039	0.0	-0.015228	0.0	0.0	0.035645	0.000000	0.0	...	0.001458	-0.023296	-0.012847

23637 rows × 7221 columns

```
In [ ]: def regression_and_skew(X, y):  
    model = sm.OLS(y, X).fit()  
    residuals = model.resid
```

```

iskew = residuals.skew()
return iskew

ff_reg['const'] = 1

ff_reg_df_long = ff_reg.reset_index().melt(id_vars=['date', 'Mkt-RF', 'SMB', 'HML', 'RF', 'const'], var_name='stock', \
iskew_dict = {}

chunk_size = 10000
for start in range(0, ff_reg_df_long.shape[0], chunk_size):
    end = start + chunk_size
    chunk = ff_reg_df_long[start:end]

    filtered_chunk = chunk.dropna(subset=['ret', 'Mkt-RF', 'SMB', 'HML', 'RF'])
    grouped = filtered_chunk.groupby([pd.Grouper(key='date', freq='M'), 'stock'])

    for name, group in grouped:
        X = group[['const', 'Mkt-RF', 'SMB', 'HML']]
        y = group['ret'] - group['RF']

        if y.dropna().shape[0] > 5:
            iskew_dict[name] = regression_and_skew(X, y)

iskew_df = pd.DataFrame(list(iskew_dict.items()), columns=['Date_Stock', 'ISkew'])
iskew_df[['Date', 'Stock']] = pd.DataFrame(iskew_df['Date_Stock'].tolist(), index=iskew_df.index)
iskew_df = iskew_df.pivot(index='Date', columns='Stock', values='ISkew')

```

In []: iskew_df.to_csv('iskew.csv')

In []: iskew_df

Out[]:

Stock	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397	93415
Date														
1926-07-31	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.031562	NaN	...	NaN	NaN	NaN
1926-08-31	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.483312	NaN	...	NaN	NaN	NaN
1926-09-30	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.040108	NaN	...	NaN	NaN	NaN
1926-10-31	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.114166	NaN	...	NaN	NaN	NaN
1926-11-30	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-3.216952	NaN	...	NaN	NaN	NaN
...
2015-08-31	0.0	-0.106139	0.314999	0.0	0.341414	0.0	0.0	-0.078535	0.000000	0.0	...	-0.056487	-0.423682	-0.041408
2015-09-30	0.0	-0.481084	-0.450013	0.0	0.152527	0.0	0.0	-0.260234	0.000000	0.0	...	0.570628	-1.883487	-0.229306
2015-10-31	0.0	0.135880	2.638152	0.0	-2.711235	0.0	0.0	1.545813	0.000000	0.0	...	-0.459884	-0.502306	-0.213465
2015-11-30	0.0	-2.771350	0.233215	0.0	0.426177	0.0	0.0	0.701902	0.000000	0.0	...	0.285056	0.597089	0.137966
2015-12-31	0.0	0.348947	0.816423	0.0	-0.955500	0.0	0.0	0.148231	0.000000	0.0	...	2.618703	-0.196447	0.762277

1074 rows × 7215 columns

- DBETA: the downside beta, estimated from a regression of daily excess stock returns on the daily excess market return over a one-year window, using only days on which the market return was below the average daily market return during that year, as in Ang et al. (2006)

In []:

```
def calculate_dbeta(excess_returns):
    dbeta_values = {}
```

```
avg_market_return = excess_returns['mkt'].mean()
```

```

downside_days = excess_returns[excess_returns['mkt'] < avg_market_return]

for column in group_columns:
    X = sm.add_constant(downside_days['mkt'])
    y = downside_days[column]
    model = sm.OLS(y, X).fit()

    dbeta_values[column] = model.params[1]

return pd.Series(dbeta_values)

dbeta_df = pd.DataFrame()

for month in reg_df.resample('M').sum().index:
    start_date = month - pd.DateOffset(years=1)
    window_data = reg_df[start_date:month]

    dbeta_values = calculate_dbeta(window_data)
    dbeta_df[month.strftime('%Y%m')] = dbeta_values

dbeta_df = dbeta_df.T
dbeta_df.index = beta_df.index
dbeta_df.columns = beta_df.columns

```

In []: dbeta_df.tail()

	PERMNO	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397	9:
		date													
201508		0.971323	0.949451	1.252754	0.971323	0.952098	0.971323	0.971323	0.993532	0.971323	0.971323	...	2.296619	0.974822	0.93:
201509		0.972737	0.937574	1.228833	0.972737	0.949474	0.972737	0.972737	0.999471	0.972737	0.972737	...	2.374240	0.983487	0.91
201510		0.956646	0.938153	1.258748	0.956646	1.003332	0.956646	0.956646	0.967401	0.956646	0.956646	...	2.305999	0.984062	0.85
201511		1.025536	1.011371	0.952687	1.025536	1.007223	1.025536	1.025536	1.028438	1.025536	1.025536	...	1.165497	0.925397	0.91
201512		1.011599	0.993231	0.982164	1.011599	1.010282	1.011599	1.011599	1.039887	1.011599	1.011599	...	0.940961	1.028076	1.09:

5 rows × 7217 columns

In []: # dbeta_df.to_csv('dbetas.csv')

Table 2

```
In [ ]: ST_df_normalized = (ST_df - ST_df.mean()) / ST_df.std()

ST_df_scaled = ST_df_normalized.clip(-1, 1)

# Then scale to -10 to 10
ST_df = ST_df_scaled * 10
```

```
In [ ]: ST_df
```

```
Out[ ]: PERMNO    10009    10026    10028    10029    10032    10036    10038    10044    10057    10064    ...    93396    93397
          date
192601      NaN      NaN      NaN      NaN      NaN      NaN      NaN      NaN  10.000000      NaN    ...
192602      NaN      NaN      NaN      NaN      NaN      NaN      NaN      NaN -10.000000      NaN    ...
192603      NaN      NaN      NaN      NaN      NaN      NaN      NaN      NaN   6.073152      NaN    ...
192604      NaN      NaN      NaN      NaN      NaN      NaN      NaN      NaN  -7.823317      NaN    ...
192605      NaN      NaN      NaN      NaN      NaN      NaN      NaN      NaN  -6.224325      NaN    ...
...        ...      ...      ...      ...      ...      ...      ...      ...      ...
201508 -0.38421  0.85452 -1.036563 -0.539591  0.433431 -1.117719 -0.584107 -0.678588 -1.509827  0.238169    ...
201509 -0.38421  0.85452 -2.445425 -0.539591  0.433431 -1.117719 -0.584107 -0.678588 -1.509827  0.238169    ...
201510 -0.38421  0.85452  6.546581 -0.539591  0.432432 -1.117719 -0.584107 -0.678588 -1.509827  0.238169    ...
201511 -0.38421  0.85452 -1.029012 -0.539591  0.433431 -1.117719 -0.584107 -0.678588 -1.509827  0.238169    ...
201512 -0.38421  0.85452 -1.022346 -0.539591  0.433431 -1.117719 -0.584107 -0.678588 -1.509827  0.238169    ...
...        ...      ...      ...      ...      ...      ...      ...      ...      ...
1080 rows × 7217 columns
```

```
In [ ]: decile_df
```

Out[]: PERMNO 10009 10026 10028 10029 10032 10036 10038 10044 10057 10064 ... 93396 93397 93415 93419 93423 93426 934

	date	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397	93415	93419	93423	93426	934
192601		NaN	10.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N							
192602		NaN	1.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N							
192603		NaN	9.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N							
192604		NaN	2.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N							
192605		NaN	2.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	N							
...
201508		3.0	5.0	6.0	3.0	3.0	3.0	3.0	1.0	3.0	3.0	...	1.0	5.0	1.0	2.0	5.0	2.0	1
201509		3.0	4.0	1.0	3.0	5.0	3.0	3.0	2.0	3.0	3.0	...	6.0	1.0	1.0	5.0	5.0	6.0	
201510		3.0	4.0	6.0	3.0	1.0	3.0	3.0	5.0	3.0	3.0	...	6.0	2.0	2.0	2.0	4.0	5.0	
201511		3.0	1.0	6.0	3.0	5.0	3.0	3.0	3.0	3.0	3.0	...	6.0	6.0	5.0	5.0	3.0	5.0	
201512		3.0	5.0	6.0	3.0	3.0	3.0	3.0	5.0	3.0	3.0	...	6.0	1.0	1.0	5.0	4.0	5.0	

1080 rows × 7217 columns

In []: price_M

Out[]: PERMNO 10009 10026 10028 10029 10032 10036 10038 10044 10057 10064 ... 93396 93397 93415 93419 93423 93426 93

	date																		
192601	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	11.8125	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
192602	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	10.2500	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
192603	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	9.2500	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
192604	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	9.2500	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
192605	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	8.5000	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...
201508	NaN	113.96	0.2667	NaN	38.07	NaN	NaN	11.97	NaN	NaN	...	3.9950	18.98	21.54	28.39	44.97	11.19	1	
201509	NaN	113.66	0.3500	NaN	38.58	NaN	NaN	11.25	NaN	NaN	...	3.7200	16.73	17.32	28.79	45.78	11.59	1	
201510	NaN	122.79	0.3600	NaN	34.62	NaN	NaN	11.09	NaN	NaN	...	3.5100	15.86	19.83	28.57	52.04	11.73	2	
201511	NaN	116.68	0.3676	NaN	37.20	NaN	NaN	10.96	NaN	NaN	...	2.9888	17.03	18.36	28.67	51.90	12.03	2	
201512	NaN	116.67	0.3300	NaN	34.92	NaN	NaN	10.75	NaN	NaN	...	3.3800	14.94	15.18	28.14	54.94	11.32	2	

1080 rows × 7217 columns

In []: ME_df

Out[]:

gvkey	1004	1045	1050	1062	1075	1076	1078	1104	1117	1121	...	287462	289735	294524	29
datadate															
199806	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
199807	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
199808	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
199809	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
199810	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
...
201508	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
201509	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
201510	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
201511	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
201512	NaN	26452.7417	260.4902	NaN	7155.9904	1625.514	66137.3852	58.2552	54.1001	161.9712	...	590.166	4.521	38249.035	9316

211 rows × 3025 columns

In []:

BM_df

Out[]:

gvkey	1004	1045	1050	1062	1075	1076	1078	1104	1117	1121	...	312079	315318	315629	315887	316056	31
datadate																	
195006	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
195007	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
195008	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
195009	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
195010	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
...
201508	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
201509	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
201510	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
201511	NaN	NaN	NaN	8.333	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
201512	NaN	9.0215	7.0509	NaN	41.304	18.8239	14.4031	12.8984	2.5359	36.1569	...	7.2258	9.1688	288.8586	32.5768	0.2667	

757 rows × 3427 columns

In []:

MOM_df

Out[]:	PERMNO	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397	93415	93...
date																
192601	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
192602	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
192603	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
192604	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
192605	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
...
201508	0.0	0.228438	-0.711806	0.0	0.115688	0.0	0.0	0.028906	0.0	0.0	0.0	...	-0.166065	0.008163	-0.188416	0.108
201509	0.0	0.249710	-0.739726	0.0	-0.074047	0.0	0.0	0.001543	0.0	0.0	0.0	...	0.144603	-0.150227	-0.229081	0.146
201510	0.0	0.218042	-0.805328	0.0	0.030869	0.0	0.0	-0.045455	0.0	0.0	0.0	...	0.406690	-0.198818	-0.227403	0.151
201511	0.0	0.103174	-0.695652	0.0	-0.066989	0.0	0.0	-0.069471	0.0	0.0	0.0	...	0.333286	-0.348012	-0.381429	0.054
201512	0.0	0.168872	-0.733333	0.0	-0.112535	0.0	0.0	-0.123320	0.0	0.0	0.0	...	0.324528	-0.381676	-0.280740	0.016

1080 rows × 7217 columns

```
In [ ]: ILLIQ_df.fillna(0, inplace=True)
```

In []: ILLIQ_df

Out[]:	PERMNO	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397	
	date														
	192601	0.0	0.000000e+00	0.000000	0.0	0.000000e+00	0.0	0.0	0.000000e+00	0.0	0.0	...	0.000000	0.000000e+00	0.
	192602	0.0	0.000000e+00	0.000000	0.0	0.000000e+00	0.0	0.0	0.000000e+00	0.0	0.0	...	0.000000	0.000000e+00	0.
	192603	0.0	0.000000e+00	0.000000	0.0	0.000000e+00	0.0	0.0	0.000000e+00	0.0	0.0	...	0.000000	0.000000e+00	0.
	192604	0.0	0.000000e+00	0.000000	0.0	0.000000e+00	0.0	0.0	0.000000e+00	0.0	0.0	...	0.000000	0.000000e+00	0.
	192605	0.0	0.000000e+00	0.000000	0.0	0.000000e+00	0.0	0.0	0.000000e+00	0.0	0.0	...	0.000000	0.000000e+00	0.

	201508	0.0	1.159407e-09	0.000013	0.0	1.782756e-09	0.0	0.0	1.503839e-07	0.0	0.0	...	0.000010	3.143938e-08	3
	201509	0.0	1.429761e-09	0.000000	0.0	2.636822e-09	0.0	0.0	1.585810e-07	0.0	0.0	...	0.000000	2.090545e-08	5
	201510	0.0	1.251335e-09	0.000049	0.0	2.372137e-09	0.0	0.0	7.067061e-08	0.0	0.0	...	0.000022	1.696294e-08	3
	201511	0.0	1.077951e-09	0.000000	0.0	1.917467e-09	0.0	0.0	1.056616e-06	0.0	0.0	...	0.000003	3.592499e-08	2.
	201512	0.0	1.496027e-09	0.000011	0.0	2.768743e-09	0.0	0.0	1.385248e-07	0.0	0.0	...	0.000002	3.693790e-08	2.

1080 rows × 7217 columns

In []: beta_df

Out[]:	PERMNO	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397	93
	date														
	195401	NaN	0.985880	NaN	...	NaN	NaN	NaN							
	195402	NaN	1.074246	NaN	...	NaN	NaN	NaN							
	195403	NaN	1.008448	NaN	...	NaN	NaN	NaN							
	195404	NaN	0.951724	NaN	...	NaN	NaN	NaN							
	195405	NaN	1.028708	NaN	...	NaN	NaN	NaN							

	201508	0.789675	0.957964	1.201376	0.789675	1.000799	0.789675	0.789675	0.626389	0.789675	0.789675	...	0.837181	0.986023	0.775
	201509	0.858696	0.928311	0.483508	0.858696	1.056992	0.858696	0.858696	0.722550	0.858696	0.858696	...	0.798310	1.049245	0.87
	201510	0.839465	1.047803	1.159167	0.839465	1.299271	0.839465	0.839465	0.884752	0.839465	0.839465	...	1.216790	1.119804	0.920
	201511	0.950650	0.908637	0.584735	0.950650	1.123927	0.950650	0.950650	0.964420	0.950650	0.950650	...	0.842401	0.987776	0.824
	201512	0.926605	0.962017	0.922854	0.926605	1.088075	0.926605	0.926605	1.224763	0.926605	0.926605	...	1.943523	0.990653	1.952

744 rows × 7217 columns

In []: ivol_df

Out[]: PERMNO 10009 10026 10028 10029 10032 10036 10038 10044 10057 10064 ... 93396 93397 93398

date	195401	NaN	0.018460	NaN	...	NaN	NaN							
195402	NaN	0.009603	NaN	...	NaN	NaN								
195403	NaN	0.015671	NaN	...	NaN	NaN								
195404	NaN	0.013580	NaN	...	NaN	NaN								
195405	NaN	0.010299	NaN	...	NaN	NaN								
...	
201508	0.015625	0.008936	0.059467	0.015625	0.009512	0.015625	0.015625	0.025601	0.015625	0.015625	...	0.049035	0.016814	0.028
201509	0.013408	0.009757	0.087188	0.013408	0.012244	0.013408	0.013408	0.016215	0.013408	0.013408	...	0.042832	0.033444	0.028
201510	0.006685	0.005175	0.130684	0.006685	0.037965	0.006685	0.006685	0.016126	0.006685	0.006685	...	0.041997	0.018645	0.028
201511	0.007282	0.015871	0.076660	0.007282	0.010167	0.007282	0.007282	0.011177	0.007282	0.007282	...	0.052289	0.022100	0.018
201512	0.011121	0.008386	0.049875	0.011121	0.008621	0.011121	0.011121	0.020238	0.011121	0.011121	...	0.106285	0.023618	0.04

744 rows × 7217 columns

In []: REV_df

Out[]:	PERMNO	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397	93415
	date														
	192601	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
	192602	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
	192603	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-0.132275	NaN	...	NaN	NaN	NaN
	192604	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-0.097561	NaN	...	NaN	NaN	NaN
	192605	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.000000	NaN	...	NaN	NaN	NaN

	201508	0.0	0.069486	-0.149317	0.0	-0.130811	0.0	0.0	-0.014427	0.000000	0.0	...	0.216450	-0.073774	-0.033534
	201509	0.0	-0.037175	-0.298158	0.0	-0.001835	0.0	0.0	-0.077812	0.000000	0.0	...	-0.289146	-0.078193	-0.041815
	201510	0.0	-0.002633	0.312336	0.0	0.013396	0.0	0.0	-0.060150	0.000000	0.0	...	-0.068836	-0.118546	-0.195915
	201511	0.0	0.080327	0.028571	0.0	-0.102644	0.0	0.0	-0.014222	0.000000	0.0	...	-0.056452	-0.052002	0.144919
	201512	0.0	-0.049760	0.021111	0.0	0.074523	0.0	0.0	-0.011722	0.000000	0.0	...	-0.148490	0.073770	-0.074130

1080 rows × 7217 columns

In []: Max_df

Out[]: PERMNO 10009 10026 10028 10029 10032 10036 10038 10044 10057 10064 ... 93396 93397 93415 93419 93423 93426 93

	date																		
192601	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	12.875	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
192602	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	12.000	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
192603	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	10.875	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
192604	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	9.500	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
192605	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	9.125	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...
201508	NaN	119.16	0.3700	NaN	38.88	NaN	NaN	12.9200	NaN	NaN	...	5.0000	20.72	23.50	31.68	46.86	13.83	2	2
201509	NaN	118.33	0.4300	NaN	38.58	NaN	NaN	12.2700	NaN	NaN	...	4.1900	19.48	22.05	29.67	46.46	12.43	2	2
201510	NaN	124.44	0.4500	NaN	41.22	NaN	NaN	11.8101	NaN	NaN	...	4.1301	17.17	20.89	30.79	52.59	12.27	2	2
201511	NaN	123.95	0.3836	NaN	37.96	NaN	NaN	11.1200	NaN	NaN	...	3.5700	17.57	20.15	29.62	53.11	12.52	2	2
201512	NaN	119.37	0.3700	NaN	37.78	NaN	NaN	10.8800	NaN	NaN	...	4.3800	17.33	19.01	28.74	54.96	12.60	2	2

1080 rows × 7217 columns

In []: Min_df

Out[]: PERMNO 10009 10026 10028 10029 10032 10036 10038 10044 10057 10064 ... 93396 93397 93415 93419 93423 93426 934

	date	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397	93415	93419	93423	93426	934
192601	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	11.625	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
192602	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	10.250	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
192603	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	9.250	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
192604	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	8.375	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
192605	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	7.750	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...
201508	NaN	109.69	0.2667	NaN	35.55	NaN	NaN	11.56	NaN	NaN	...	3.8200	18.02	19.65	27.90	42.34	10.60	18	18
201509	NaN	111.66	0.3000	NaN	35.98	NaN	NaN	11.25	NaN	NaN	...	3.5400	16.18	16.59	27.70	43.22	11.38	18	18
201510	NaN	113.97	0.2900	NaN	32.97	NaN	NaN	10.90	NaN	NaN	...	3.3611	15.86	17.48	28.35	46.33	11.35	18	18
201511	NaN	113.73	0.2900	NaN	35.34	NaN	NaN	10.75	NaN	NaN	...	2.9400	15.30	17.37	27.57	51.19	11.78	2	2
201512	NaN	112.69	0.3084	NaN	34.66	NaN	NaN	10.08	NaN	NaN	...	3.0038	14.86	12.70	27.17	51.76	11.20	21	21

1080 rows × 7217 columns

In []: tk_values

Out[]: PERMNO 10009 10026 10028 10029 10032 10036 10038 10044 10057 10064 ... 93396 93397 93415 9

	date	PERMNO	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397	93415	9
192601		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
192602		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
192603		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
192604		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
192605		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
...
201508		0.0	0.488856	-7.897477	0.0	-2.227256	0.0	0.0	-1.499669	0.0	0.0	...	-1.141606	-2.721190	-1.591522	-0.93	
201509		0.0	0.332580	-7.959257	0.0	-2.526262	0.0	0.0	-1.634804	0.0	0.0	...	-1.419714	-3.336439	-2.157416	-0.89	
201510		0.0	0.405931	-8.037565	0.0	-2.880870	0.0	0.0	-1.694807	0.0	0.0	...	-2.062214	-3.512657	-2.016325	-0.84	
201511		0.0	0.152556	-8.024943	0.0	-2.467067	0.0	0.0	-1.751041	0.0	0.0	...	-2.820000	-3.517214	-2.270185	-0.68	
201512		0.0	0.074781	-8.391391	0.0	-2.837361	0.0	0.0	-1.838433	0.0	0.0	...	-2.084789	-4.235146	-2.801986	-0.69	

1080 rows × 7217 columns

In []: SKEW_df

Out[]:	PERMNO	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397	93415	93
	date															
	192601	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
	192602	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
	192603	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
	192604	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
	192605	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN

	201508	0.0	-3.338193	-0.072577	0.0	-0.392391	0.0	0.0	0.184850	0.0	0.0	...	5.125254	-0.166139	-0.736083	-0.50'
	201509	0.0	-3.177815	-0.073294	0.0	-0.349498	0.0	0.0	0.136690	0.0	0.0	...	5.114800	-0.989990	-0.754296	-0.456
	201510	0.0	-3.188924	1.241365	0.0	-2.742430	0.0	0.0	0.196931	0.0	0.0	...	5.157664	-0.920797	-0.716298	-0.544
	201511	0.0	-3.127722	1.178957	0.0	-2.667381	0.0	0.0	0.237323	0.0	0.0	...	5.083206	-0.715634	-0.589164	-0.398
	201512	0.0	-3.270903	1.178219	0.0	-2.915193	0.0	0.0	0.219069	0.0	0.0	...	4.986837	-0.709800	-0.829244	-0.321

1080 rows × 7217 columns

In []: coskew_df

Out[]:	PERMNO	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	\$
date														
	195401	NaN	-16.749747	NaN	...	NaN								
	195402	NaN	-15.189075	NaN	...	NaN								
	195403	NaN	-19.106283	NaN	...	NaN								
	195404	NaN	-18.280069	NaN	...	NaN								
	195405	NaN	-12.991376	NaN	...	NaN								

	201508	-3.674284	-4.026450	-4.561117	-3.674284	-4.057887	-3.674284	-3.674284	-3.815945	-3.674284	-3.674284	...	-4.425116	-4.1
	201509	-3.499383	-3.790520	-4.197160	-3.499383	-3.782316	-3.499383	-3.499383	-3.635155	-3.499383	-3.499383	...	-4.311348	-3.9
	201510	-3.059994	-3.339612	-3.703983	-3.059994	-3.385478	-3.059994	-3.059994	-3.107678	-3.059994	-3.059994	...	-3.792292	-3.5
	201511	-3.722662	-3.838305	-3.353030	-3.722662	-3.840984	-3.722662	-3.722662	-3.737246	-3.722662	-3.722662	...	-4.114807	-3.7
	201512	-3.295576	-3.332125	-3.219952	-3.295576	-3.373476	-3.295576	-3.295576	-3.328792	-3.295576	-3.295576	...	-3.331621	-3.36

744 rows x 7217 columns

In []: iskew_df

Out[]:

Stock	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397	93415
Date														
192607	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.031562	NaN	...	NaN	NaN	NaN
192608	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.483312	NaN	...	NaN	NaN	NaN
192609	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.040108	NaN	...	NaN	NaN	NaN
192610	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.114166	NaN	...	NaN	NaN	NaN
192611	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-3.216952	NaN	...	NaN	NaN	NaN
...
201508	0.0	-0.106139	0.314999	0.0	0.341414	0.0	0.0	-0.078535	0.000000	0.0	...	-0.056487	-0.423682	-0.041408
201509	0.0	-0.481084	-0.450013	0.0	0.152527	0.0	0.0	-0.260234	0.000000	0.0	...	0.570628	-1.883487	-0.229306
201510	0.0	0.135880	2.638152	0.0	-2.711235	0.0	0.0	1.545813	0.000000	0.0	...	-0.459884	-0.502306	-0.213465
201511	0.0	-2.771350	0.233215	0.0	0.426177	0.0	0.0	0.701902	0.000000	0.0	...	0.285056	0.597089	0.137966
201512	0.0	0.348947	0.816423	0.0	-0.955500	0.0	0.0	0.148231	0.000000	0.0	...	2.618703	-0.196447	0.762277

1074 rows × 7215 columns

In []:

dbeta_df

Out[]: PERMNO 10009 10026 10028 10029 10032 10036 10038 10044 10057 10064 ... 93396 93397 9:

	date	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	93397	9:
195401		NaN	1.084026	NaN	...	NaN	NaN								
195402		NaN	0.941611	NaN	...	NaN	NaN								
195403		NaN	0.924084	NaN	...	NaN	NaN								
195404		NaN	0.978176	NaN	...	NaN	NaN								
195405		NaN	0.991650	NaN	...	NaN	NaN								
...
201508	0.971323	0.949451	1.252754	0.971323	0.952098	0.971323	0.971323	0.993532	0.971323	0.971323	...	2.296619	0.974822	0.93:	
201509	0.972737	0.937574	1.228833	0.972737	0.949474	0.972737	0.972737	0.999471	0.972737	0.972737	...	2.374240	0.983487	0.91	
201510	0.956646	0.938153	1.258748	0.956646	1.003332	0.956646	0.956646	0.967401	0.956646	0.956646	...	2.305999	0.984062	0.85	
201511	1.025536	1.011371	0.952687	1.025536	1.007223	1.025536	1.025536	1.028438	1.025536	1.025536	...	1.165497	0.925397	0.91	
201512	1.011599	0.993231	0.982164	1.011599	1.010282	1.011599	1.011599	1.039887	1.011599	1.011599	...	0.940961	1.028076	1.09:	

744 rows × 7217 columns

In []: factor_dfs = {
 'DECILE': decile_df,
 'ST': ST_df,
 'PRICE': price_M,
 'ME': ME_df,
 'BM': BM_df,
 'MOM': MOM_df,
 # 'ILLIQ': ILLIQ_df,
 'BETA': beta_df,
 'IVOL': ivol_df,
 'REV': REV_df,
 'MAX': Max_df,
 'MIN': Min_df,
 'TK': tk_values,
 'SKEW': SKEW_df,
 'COSKEW': coskew_df,
 'ISKEW': iskew_df,
 'DBETA': dbeta_df
}

```
combined_df = pd.concat(factor_dfs.values(), axis=1, keys=factor_dfs.keys())
```

```

sub_tables = {date: combined_df.loc[[date]].stack(level=-1, dropna=True) for date in combined_df.index}
averages_by_decile = {date: sub_table.groupby('DECILE').mean() for date, sub_table in sub_tables.items()}
all_averages = pd.concat(averages_by_decile, names=['Date', 'DECILE'])
table2 = all_averages.groupby('DECILE').mean()

```

In []: table2

Out[]:

	ST	PRICE	ME	BM	MOM	BETA	IVOL	REV	MAX	MIN	TK	SKEW	
DECILE													
1.0	-3.185681	18.579538	5778.973846	12.222740	0.181537	0.999349	0.036317	0.025185	22.216965	17.604644	-3.823078	-0.217600	
2.0	-0.369681	28.388402	4123.625570	200.424709	0.089344	0.980620	0.016490	0.010225	30.348882	27.270537	-2.933070	-0.227747	
3.0	-0.242086	34.467783	3872.385281	391.943230	0.042223	0.938801	0.010442	0.003948	36.169454	33.401465	-1.966968	-0.110443	
4.0	0.021728	42.742502	2583.038490	9.648616	0.074985	0.972215	0.009809	0.014815	44.280695	41.224915	-2.021697	-0.147577	
5.0	0.122577	45.152901	4016.570009	10.463821	0.124932	0.992601	0.014184	0.008118	46.644925	43.003950	-2.374772	-0.238725	
6.0	0.059124	33.918645	8367.901311	244.719908	0.148206	0.995111	0.029355	0.011180	35.272329	31.725759	-2.867205	-0.007025	
7.0	0.130652	29.026575	7019.603323	15.036887	0.109027	0.995863	0.036503	0.006138	30.348585	26.920994	-3.282684	0.098848	
8.0	0.531404	28.008762		NaN	13.399975	0.100962	0.997442	0.026907	0.006676	29.396755	25.847042	-3.392668	0.102810
9.0	2.841359	24.026203		NaN	7.820000	0.095240	0.999212	0.032351	0.004800	25.385237	21.833779	-3.788286	0.262593
10.0	7.603577	20.325538		NaN	7.465800	0.131250	0.991255	0.037122	0.011391	21.651145	17.660449	-3.941731	0.579778

3. Replicate 2 other tables from 3 to 10

Table 3

In []:

```

excess_ret = ret_df['1954-01-04':].merge(rf, left_index=True, right_index=True, how='left')
for col in excess_ret.columns:
    if col != 'rf':
        excess_ret[col] = excess_ret[col] - excess_ret['rf']
excess_returns = excess_ret.resample('M').mean()
excess_returns.drop('rf', axis=1, inplace=True)
excess_returns.index = pd.to_datetime(excess_returns.index).strftime('%Y%m')
excess_returns

```

Out[]:

	10009	10026	10028	10029	10032	10036	10038	10044	10057	10064	...	93396	9339
date													
195401	NaN	-1.174077	NaN	...	NaN	Nan							
195402	NaN	-0.967435	NaN	...	NaN	Nan							
195403	NaN	-1.026522	NaN	...	NaN	Nan							
195404	NaN	-0.960615	NaN	...	NaN	Nan							
195405	NaN	-0.758679	NaN	...	NaN	Nan							
...
201508	-0.071905	-0.073623	-0.086878	-0.071905	-0.071904	-0.071905	-0.071905	-0.075552	-0.071905	-0.071905	...	-0.086994	-0.07558
201509	-0.022381	-0.022427	-0.005507	-0.022381	-0.021581	-0.022381	-0.022381	-0.025200	-0.022381	-0.022381	...	-0.024791	-0.02776
201510	-0.015238	-0.011752	-0.006527	-0.015238	-0.020584	-0.015238	-0.015238	-0.016170	-0.015238	-0.015238	...	-0.017018	-0.01833
201511	-0.124737	-0.127851	-0.118686	-0.124737	-0.121575	-0.124737	-0.124737	-0.124982	-0.124737	-0.124737	...	-0.132413	-0.12029
201512	-0.227727	-0.227675	-0.231360	-0.227727	-0.230521	-0.227727	-0.227727	-0.228429	-0.227727	-0.227727	...	-0.216688	-0.23334

744 rows × 7217 columns

Table 4

In []:

```
factor_dfs = {
    'ST': ST_df,
    'BETA': beta_df,
    'ME': ME_df,
    'BM': BM_df,
    'MOM': MOM_df,
    'IVOL': ivol_df,
    'REV': REV_df,
    'MAX': Max_df,
    'MIN': Min_df,
    'TK': tk_values,
    'SKEW': SKEW_df,
    'COSKEW': coskew_df,
    'ISKEW': iskew_df,
    'DBETA': dbeta_df
}
```

```
In [ ]: all_model_results = []

unique_periods = excess_returns.index.unique()

while factor_dfs:
    regression_results = pd.DataFrame()

    for period in unique_periods:
        X = pd.DataFrame({factor: df.loc[period] for factor, df in factor_dfs.items() if period in df.index})
        y = excess_returns.loc[period]

        X = sm.add_constant(X, has_constant='add')
        valid_rows = ~X.isnull().any(axis=1) & ~y.isnull()
        y = y[valid_rows]
        X = X[valid_rows]

        if not X.empty:
            model = sm.OLS(y, X)
            results = model.fit()
            regression_results = regression_results.append(results.params, ignore_index=True)

    average_coefficients = regression_results.mean()
    t_stats = average_coefficients / regression_results.sem()

    all_model_results.append(pd.DataFrame({'Coefficient': average_coefficients, 't-statistic': t_stats}))

    factor_dfs.popitem()
```

```
In [ ]: index_tuples = []
for factor in factor_dfs.keys():
    index_tuples.append((factor, 'coef'))
    index_tuples.append((factor, 'tstat'))

multi_index = pd.MultiIndex.from_tuples(index_tuples, names=["Factor", "Type"])

table4 = pd.DataFrame(index=multi_index, columns=range(1, len(all_model_results) + 1))

for i, result_df in enumerate(all_model_results, 1):
    for factor in factor_dfs.keys():
        table4.loc[(factor, 'coef'), i] = result_df.loc[factor, 'Coefficient'] if factor in result_df.index else None
        table4.loc[(factor, 'tstat'), i] = result_df.loc[factor, 't-statistic'] if factor in result_df.index else None
```

```
In [ ]: table4
```

Out[]:

		1	2	3	4	5	6	7	8	9	10
Factor	Type										
ST	coef	0.00196	-0.022383	-0.022364	-0.020452	0.022453	0.029082	0.022117	0.000848	0.000554	0.00054
	tstat	8.726634	-0.839788	-0.839762	-0.764967	1.848653	2.118232	2.172967	1.589911	1.182107	1.141256
ME	coef	None	-0.005639	-0.005639	-0.005656	-0.005809	-0.005157	-0.005241	-0.00345	-0.003456	-0.003453
	tstat	None	-1.97828	-1.978318	-1.983024	-2.690439	-3.756071	-3.831376	-4.476258	-4.485568	-4.485714
BM	coef	None	-0.2353	-0.235314	-0.235318	-0.079825	-0.055327	-0.043249	-0.012622	-0.012668	-0.012666
	tstat	None	-10.57443	-10.575666	-10.58866	-5.31611	-5.071516	-5.4871	-5.401428	-5.434203	-5.442449
MOM	coef	None	0.011647	0.013407	0.016524	0.015143	-0.000686	0.000827	0.000044	0.000208	0.000068
	tstat	None	0.651292	0.752832	0.985191	1.450401	-0.092021	0.18975	0.109749	0.574297	0.169758
BETA	coef	None	-0.401413	-0.416562	-0.441362	-0.117174	-0.053323	-0.035768	-0.001025	-0.00122	0.001667
	tstat	None	-9.425402	-9.761276	-10.178667	-4.682425	-3.621694	-3.507901	-1.032765	-1.336734	0.677933
IVOL	coef	None	None	-0.006522	-0.004028	0.003942	0.000421	0.000721	-0.005697	-0.002066	-0.006876
	tstat	None	None	-0.534736	-0.558711	0.468149	0.082754	0.120512	-0.856499	-0.333934	-0.664197
REV	coef	None	None	None	-0.013587	-0.000452	0.002247	0.001447	-0.000579	-0.000628	-0.000568
	tstat	None	None	None	-1.120091	-0.040621	0.427325	0.419838	-0.834126	-0.780871	-0.680397
MAX	coef	None	None	None	None	-0.052178	-0.03801	-0.04034	-0.013173	-0.013152	-0.013063
	tstat	None	None	None	None	-5.662432	-5.046885	-6.528069	-7.619582	-7.61774	-7.571619
MIN	coef	None	None	None	None	-0.011595	-0.010481	-0.019193	-0.009815	-0.00984	-0.009917
	tstat	None	None	None	None	-1.165774	-1.366274	-3.092094	-6.523511	-6.539665	-6.617753
TK	coef	None	None	None	None	None	0.076874	0.068195	0.009374	0.009705	0.009734
	tstat	None	None	None	None	None	5.972037	5.673084	6.578809	6.770056	6.829139
SKEW	coef	None	None	None	None	None	None	-0.005265	-0.002075	-0.001792	-0.001792
	tstat	None	None	None	None	None	None	-0.557328	-1.95037	-1.810447	-1.818411
COSKEW	coef	None	None	None	None	None	None	None	0.059091	0.050671	0.051155
	tstat	None	None	None	None	None	None	None	6.114299	11.055855	11.462387
ISKEW	coef	None	None	None	None	None	None	None	None	0.000037	0.000013
	tstat	None	None	None	None	None	None	None	None	0.08532	0.029439

	1	2	3	4	5	6	7	8	9	10
Factor	Type									
DBETA	coef	None	-0.012005							
	tstat	None	-1.789523							

4. If the numbers you obtain in questions 2 and 3 deviate from those in the paper, why do you think this is? What parts of the data construction and replication were difficult? Was there any additional information the authors could have given you to make this process simpler?

Replicating the study was challenging due to several factors. Firstly, sourcing ST, Beta, and Illiq factors was arduous, especially with data dating back to the 1920s, which is scarce. Additionally, the incongruent start dates of various datasets compounded the difficulty, requiring meticulous alignment. The process was further complicated by the necessity to reference multiple papers for factor calculations. Although the authors provided formulas, the lack of step-by-step methodology led to ambiguity. More clarity on the calculation nuances and a comprehensive methodology breakdown would significantly streamline the replication process.

5. In your view, what are the key takeaways of this paper? How did the results in the tables you replicated contribute to the paper as a whole?

Replicating tables 2 and 4 was insightful, as it provided a detailed look into the empirical validation process and a clearer grasp of the study's framework. It was particularly interesting to observe deviations in results from those presented in the paper, suggesting that findings might vary with different data sources and timeframes, potentially challenging the paper's original conclusions.