Replication report of

"Eureka, a version of Momentum that Works in Japan" by Denis Chaves

Tse Chun Hei Vincent
Chan Kwai Chuen

Introduction

The replication report focuses on a quantitative trading strategy from the paper “Eureka, a version of Momentum that Works in Japan” written by Denis Chaves (Chaves, 2012). Denis Chaves used idiosyncratic returns to generate an alternative momentum, which is different from the traditional momentum strategies and the new momentum strategies (IMOM) could create alpha against fama french factors models. In this strategy replication report, we would like to study the new strategy invented by Denis Chaves and investigate whether it is profitable and reliable.

Data processing

In this report, we used monthly stock data from CRSP. There were permno, date, close price, volumn, exchange code, risk free rate, market return and stock return. We construct market capital, stock excess return and excess market return. Also, following the paper, stocks are picked starting from 1964. Fama French data was grabbed from the ff parquet file of the professor.

```
msf = pd.read_parquet('msf.parquet',columns = ['permno', 'date', 'prc',
'shrout', 'hexcd', 'ret'])
msf = msf.loc[msf['date'] > '1964-12-31']
msf = msf[msf['hexcd'].isin(set([1,2,3]))]
msf = msf.assign(end_of_month = lambda df:
df['date'].apply(end_of_month))
```

Methodology

Our goal is to find out idiosyncratic momentum and regress the trading strategy on Fama French 3 factors/4 factors/5 factors models.
From the author, idiosyncratic return is:

$$excess\ stock\ return\ =\ \alpha\ +\ \beta\ *\ excess\ market\ return\ +\ \varepsilon$$

idiosyncratic momentum is:

$$IMOM\ =\ \prod_{i=2}^{12}(1+\varepsilon_{t-i})\ -\ 1$$

```
##Beta estimate
container = pd.DataFrame(columns = ['const', 'mkt_rf'])
for i in range (10000, int(merge['permno'].max())+1):
  temp = merge.loc[merge['permno'] == i]
  if temp.shape[0] <36:
    pass
  else:
    fit = RollingOLS(temp['ret_rf'], smf.add_constant(temp['mkt_rf']),
window = 36).fit()
    params = fit.params
    container = container.append(params)

##Residual
merge[['constant', 'beta']] = container
```

```
merge['residual'] = merge['ret_rf'] - merge['constant'] -
merge['beta']*merge['mkt_rf']
```

Data filtering and requirements

We create the lag variables for stock price, return and market capital. We need to ensure that there are at least 8 valid returns. Also, We cannot miss returns 2 months ago, price 13 months ago and last month's return. Besides, as required by the author, we need a full previous year of data.

```
merge2 = merge[
    (
        merge['Momentum'].notnull() &
        merge['Residual_Momentum'].notnull() &
        merge['ret'].notnull() &
        merge['market_cap_lag1'].notnull() &
        merge['prc_lag13'].notnull() &
        (merge['rollvalidobs_lag1'] == 12) &
        merge['prc_lag13'].notnull()
    )
].copy().reset_index(drop=True)
```

Creating signal

After getting all the data we need, we merge the msf file and the fama french data file in order to prepare for rolling regression. In rolling regression, the rolling window is set to be 36 months of data, following Denis Chaves. Stocks lie within 36 months of data point are eliminated. Calling the function from statsmodels.regression.rolling, we get the parameters, which is a series of coefficients of the constant and the independent variable. After that, we can calculate the residuals by the following equation:

$$\varepsilon = excess\ stock\ return - \alpha - \beta * excess\ market\ return$$

With the residuals, we can calculate the rolling residual cumulative return. Also, we can calculate the classic momentum with the excess return.

```
def rolling_prod(a, n=11) :
    ret = np.cumprod(a.values)
    ret[n:] = ret[n:] / ret[:-n]
    ret[:n-1] = np.nan
    return pd.Series(ret, index=a.index)

merge['rolling_residual_11_months'] = (
    merge
    .assign(ret=(merge['residual'].fillna(0)+1))
    .groupby('permno')['ret']
```

```
    .apply(rolling_prod)
) - 1

#Rolling Return
def rolling_prod(a, n=11) :
    ret = np.cumprod(a.values)
    ret[n:] = ret[n:] / ret[:-n]
    ret[:n-1] = np.nan
    return pd.Series(ret, index=a.index)

merge['rolling_return_11_months'] = (
    merge
    .assign(ret=(merge['ret'].fillna(0)+1))
    .groupby('permno')['ret']
    .apply(rolling_prod)
) - 1


merge['Momentum'] =
merge.groupby('permno')['rolling_return_11_months'].shift(2)
merge['Residual_Momentum'] =
merge.groupby('permno')['rolling_residual_11_months'].shift(2)
```

Portfolio sort

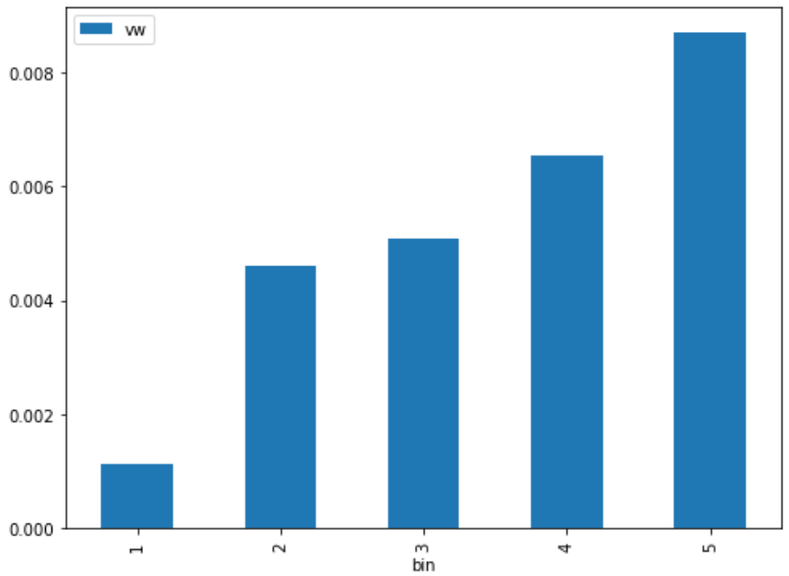
Firstly, we apply quantiles to the IMOM, and create 5 bins according to the author, which are called winners minus losers. The stocks inside the fifth bin are the winners and the stocks inside the first bin are the losers. Every stock is assigned to the bins based on their lag return. From that, we can find the return for the next month. By applying equal weight and value weight, we create two types of portfolio. From each type of portfolio, portfolio returns are calculated by winners minus losers (long the winners and short the losers).

```
def apply_quantiles(x, include_in_quantiles=None, bins=5):

    if include_in_quantiles is None:
        include_in_quantiles = [True] * len(x)

    x = pd.Series(x)
    quantiles = np.quantile(
        x[x.notnull() & include_in_quantiles],
        np.linspace(0, 1, bins+1)
    )
    quantiles[0] = x.min() - 1
    quantiles[-1] = x.max() + 1

    return pd.cut(x, quantiles, labels=False) + 1

merge2 = merge2[merge2.Residual_Momentum.notnull()]

merge2['bin'] = (
```

```
    merge2
    .groupby('date')
    .apply(lambda group: apply_quantiles(group['Residual_Momentum'],
bins=5, include_in_quantiles=(group['hexcd']==1)))
).reset_index(level=[0], drop=True).sort_index()

portfolios = (
    merge2
    .groupby(['date', 'bin'])
    .apply(
        lambda g: pd.Series({
            'portfolio_ew': g['ret_rf'].mean(),
            'portfolio_vw': (g['ret_rf'] * g['market_cap_lag1']).sum() /
g['market_cap_lag1'].sum()
        })
    )
).reset_index()

portfolios2 = pd.merge(
    portfolios.query('bin==5'),
    portfolios.query('bin==1'),
    suffixes=['_long', '_short'],
    on='date'
)

portfolios2['strategy_ew'] = portfolios2['portfolio_ew_long'] -
portfolios2['portfolio_ew_short']
portfolios2['strategy_vw'] = portfolios2['portfolio_vw_long'] -
portfolios2['portfolio_vw_short']
```

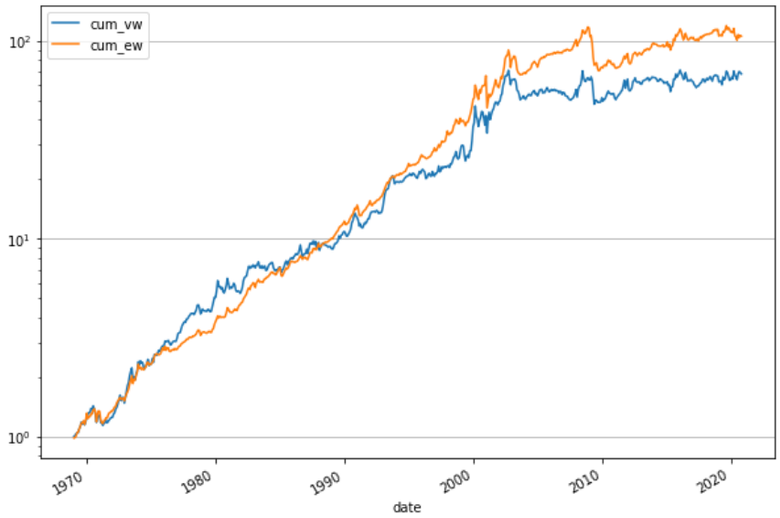
Plots

Bar plot of the bins:




```
dta = portfolios.groupby('bin').agg(vw=('portfolio_vw',
'mean')).sort_values('bin')
plt.rcParams["figure.figsize"] = (8,6)
dta.plot(kind='bar')
```

P&L curve of the strategy:



The value weight strategy has lower cumulative returns, while the equal weight strategy has higher returns that are biased upward. For larger investors, It is hard to trade with the equal weight strategy towards smaller stocks without moving the market. Since they buy large blocks of stocks and do not involve themselves as frequently in small-cap offerings. They would own the most of the controlling portions of these smaller companies if they did. They will lose liquidity. Therefore, traders would be more suitable to trade at bigger companies.
Also, small companies' stocks tend to be more volatile and are riskier investments. Large companies are more stable and transparent. It allows high volume trading, while the small companies could not afford that.

```
plt.rcParams["figure.figsize"] = (10,7)
portfolios2 = portfolios2.sort_values('date')
portfolios2['cum_vw'] = (portfolios2['strategy_vw'] + 1).cumprod() - 1
portfolios2['cum_ew'] = (portfolios2['strategy_ew'] + 1).cumprod() - 1
(
    portfolios2
    .assign(date=pd.to_datetime(portfolios2['date']))
    .assign(cum_vw=portfolios2['cum_vw']+1)
    .assign(cum_ew=portfolios2['cum_ew']+1)
    .plot(x='date', y=['cum_vw','cum_ew'] , logy=True).grid(axis='y')
)
```

Sharpe ratio

The annualized value-weighted sharpe ratio is 0.6420.
The annualized equal-weighted sharpe ratio is 0.8195.

```
def sharpe(x):
    return x.mean() / x.std()
sharpe_vw = sharpe(portfolios2['strategy_vw'])
sharpe_ew = sharpe(portfolios2['strategy_ew'])
```

Benchmarking

We would like to test how this IMOM strategy outperforms the Fama French 3 factors model. We regress on the 3 factors plus the classic MOM for both value weight and equal weight portfolios.

```
EW_1_fit = sm.ols('strategy_ew ~ 1 + mkt_rf',
data=pd.merge(four_factor_monthly, portfolios2,
```

```
on='end_of_month')).fit()
EW_1 = EW_1_fit.params


EW_2_fit = sm.ols('strategy_ew ~ 1 + mkt_rf + smb + hml',
data=pd.merge(four_factor_monthly, portfolios2,
on='end_of_month')).fit()
EW_2 = EW_2_fit.params


EW_3_fit = sm.ols('strategy_ew ~ 1 + mkt_rf + smb + hml + Mom',
data=pd.merge(four_factor_monthly, portfolios2,
on='end_of_month')).fit()
EW_3 = EW_3_fit.params


VW_1_fit = sm.ols('strategy_vw ~ 1 + mkt_rf',
data=pd.merge(four_factor_monthly, portfolios2,
on='end_of_month')).fit()
VW_1 = VW_1_fit.params


VW_2_fit = sm.ols('strategy_vw ~ 1 + mkt_rf + smb + hml',
data=pd.merge(four_factor_monthly, portfolios2,
on='end_of_month')).fit()
VW_2 = VW_2_fit.params


VW_3_fit = sm.ols('strategy_vw ~ 1 + mkt_rf + smb + hml + Mom',
data=pd.merge(four_factor_monthly, portfolios2,
on='end_of_month')).fit()
VW_3 = VW_3_fit.params

print(smf.iolib.summary2.summary_col([EW_1_fit,EW_2_fit,EW_3_fit,VW_1_fi
t,VW_2_fit,VW_3_fit], stars = True))
```

Risk analysis

The annualized alphas are significant for CAPM, 3-factor, 4-factor. Except for the strategy 3 of both value weight and equal weight portfolios, strategies 1 and 2, which are the CAPM and Fama French 3 factors model, have about annualized 10% - 11% alpha. For the Fama French 4 factors model, the annualized alpha is about 4.4% for equal weight and 2.7% for value weight portfolios.

	strategy_ew I	strategy_ew II	strategy_ew III	strategy_vw I	strategy_vw II	strategy_vw III
Intercept	0.0090***	0.0092***	0.0036***	0.0086***	0.0090***	0.0023**
	(0.0013)	(0.0014)	(0.0008)	(0.0016)	(0.0016)	(0.0011)
Mom			0.6465***			0.7609***
			(0.0195)			(0.0248)
R-squared	0.0559	0.0643	0.6628	0.0440	0.0506	0.6245
R-squared Adj.	0.0543	0.0598	0.6606	0.0425	0.0460	0.6220
hml		-0.0690	0.1385***		-0.0771	0.1671***
		(0.0468)	(0.0288)		(0.0566)	(0.0365)
mkt_rf	-0.1779***	-0.1694***	-0.0072	-0.1898***	-0.2179***	-0.0269
	(0.0293)	(0.0314)	(0.0195)	(0.0355)	(0.0380)	(0.0247)
smb		-0.0925**	-0.0760***		0.0771	0.0966***
		(0.0461)	(0.0277)		(0.0558)	(0.0351)

R-squared corresponds to each independent variables:
The IMOM strategy has low R-squared with the 3 factors, which are all below 0.1 and they are not significant. But it is highly correlated with the momentum strategy like robust-minus-weak (0.26) and momentum (0.75). Therefore, the IMOM can outperform the CAPM and Fama French 3 factors, but not 4 factors and 5 factors which includes the momentum effect. The model regression shows a positive correlation with the momentum factor, indicating that the IMOM on the portfolio can be attributed to the momentum factor. Since the model can explain more portfolio returns, it reduces the manager's original excess returns. The IMOM strategy is exposed to a higher risk from the momentum, this means that in some time periods, following a momentum-driven strategy will produce a negative return relative to the market.

	strategy_ew I	strategy_ew II	strategy_ew III	strategy_ew IIII	strategy_ew IIIII	strategy_vw I
Intercept	0.0090***	0.0081***	0.0082***	0.0074***	0.0073***	0.0028***
	(0.0013)	(0.0014)	(0.0014)	(0.0014)	(0.0014)	(0.0010)
Mom						0.7454***
						(0.0241)
R-squared	0.0559	0.0001	0.0237	0.0197	0.0280	0.6071
R-squared Adj.	0.0543	-0.0015	0.0221	0.0182	0.0264	0.6064
cma				0.2413***		
				(0.0682)		
hml		0.0111				
		(0.0465)				
mkt_rf	-0.1779***					
	(0.0293)					
rmw					0.2599***	
					(0.0615)	
smb			-0.1747***			
			(0.0450)			

Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01

```
mkt_rf = sm.ols('strategy_ew ~ 1 + mkt_rf', data=portfolios3).fit()
hml = sm.ols('strategy_ew ~ 1 + hml', data=portfolios3).fit()
smb = sm.ols('strategy_ew ~ 1 + smb', data=portfolios3).fit()
cma = sm.ols('strategy_ew ~ 1 + cma', data=portfolios3).fit()
rmw = sm.ols('strategy_ew ~ 1+ rmw', data=portfolios3).fit()
Mom = sm.ols('strategy_vw ~ 1 + Mom', data=pd.merge(four_factor_monthly,
portfolios2, on='end_of_month')).fit()
print(smf.iolib.summary2.summary_col([mkt_rf,hml,smb,cma,rmw,Mom], stars
= True))


print(smf.iolib.summary2.summary_col([mkt_rf,hml,smb,cma,rmw,Mom], stars
= True))
```

Conclusion

This report shows that the IMOM strategy invented by Denis Chaves can outperform the CAPM and Fama French 3 factors models. However, this strategy is less profitable compared to the Fama French 4 factors and 5 factors models, and it is exposed to the risk from the traditional momentum effect.

References

Chaves, D. B. (2012). Eureka! A momentum strategy that also works in Japan. A Momentum Strategy that Also Works in Japan (January 9, 2012).