In [1]:
```
from typing import Optional, List

import numpy as np
import pandas as pd
import seaborn as sn
import statsmodels.api as sm
import requests
from loguru import logger
import ccxt
import time
```

In [2]:
```
_binance = ccxt.binanceusdm()
_TARGET_COIN_TO_LIST = [
    'ethereum',
    'bitcoin',
    'binancecoin',
    'solana',
    'avalanche-2',
    'matic-network',
    'tron',
]
#_TVL_NAME_LIST = [coin_id + '_tvl_usd' for coin_id in _TARGET_COIN_TO_LI
```

Download Data from Coingecko

In [4]:
```
def fetch_price_usd_daily_all(coin_id: str = 'solana') -> Optional[pd.Dat
    url = 'https://api.coingecko.com/api/v3/coins/' + coin_id + \
    '/market_chart?vs_currency=usd&days=365&interval=daily'
    response = requests.get(url)
    data_dict = response.json()
    price_list = data_dict.get('prices')
    if price_list is None:
        logger.error(f'cannot fetch data from api, coin_id={coin_id}')
        return
    data_df = pd.DataFrame(price_list)
    data_df.columns = ['ts', coin_id + '_price_usd']
    data_df['ts'] = data_df['ts'].apply(lambda x: pd.to_datetime(_binance
    data_df = data_df.set_index('ts')
    return data_df
```

In [5]:
```
def download_coins_price(coin_id_list: List[str] = _TARGET_COIN_TO_LIST,
                         save_path: str = 'coins_price_data.xlsx') -> Opt
    first_coin_id = coin_id_list[0]
    df_init = fetch_price_usd_daily_all(first_coin_id)
    if df_init is None:
        logger.warning(f'no data for coin, id={first_coin_id}')
        return
    merged_df: pd.DataFrame = pd.DataFrame(index=df_init.index)
    for coin_id in coin_id_list:
        time.sleep(20)
        logger.info(f'fetch price data, id={coin_id}')
        df = fetch_price_usd_daily_all(coin_id)
        if df is None:
            logger.warning(f'no data for coin, id={coin_id}')
        merged_df = pd.merge(merged_df, df, how='left', left_index=True,
```

```
    merged_df = merged_df.dropna()
    merged_df.index = merged_df.index.date
    merged_df.to_excel(save_path)
    return merged_df
```

In [6]:
```
def fetch_mkt_cap_usd_daily_all(coin_id: str = 'solana') -> Optional[pd.D
    url = 'https://api.coingecko.com/api/v3/coins/' + coin_id + \
    '/market_chart?vs_currency=usd&days=365&interval=daily'
    response = requests.get(url)
    data_dict = response.json()
    mkt_cap_list = data_dict.get('market_caps')
    if mkt_cap_list is None:
        logger.error(f'cannot fetch data from api, coin_id={coin_id}')
        return
    data_df = pd.DataFrame(mkt_cap_list)
    data_df.columns = ['ts', coin_id + '_mkt_cap_usd']
    data_df['ts'] = data_df['ts'].apply(lambda x: pd.to_datetime(_binance
    data_df = data_df.set_index('ts')
    return data_df
```

In [7]:
```
def download_coins_mkt_cap(coin_id_list: List[str] = _TARGET_COIN_TO_LIST
                           save_path: str = 'coins_mkt_cap_data.xlsx') -> O
    first_coin_id = coin_id_list[0]
    df_init = fetch_mkt_cap_usd_daily_all(first_coin_id)
    if df_init is None:
        logger.warning(f'no data for coin, id={first_coin_id}')
        return
    merged_df: pd.DataFrame = pd.DataFrame(index=df_init.index)
    for coin_id in coin_id_list:
        time.sleep(20)
        logger.info(f'fetch mkt cap data, id={coin_id}')
        df = fetch_mkt_cap_usd_daily_all(coin_id)
        if df is None:
            logger.warning(f'no data for coin, id={coin_id}')
        merged_df = pd.merge(merged_df, df, how='left', left_index=True,

    merged_df = merged_df.dropna()
    merged_df.index = merged_df.index.date
    merged_df.to_excel(save_path)
    return merged_df
```

In [8]:
```
price_df = download_coins_price()
```

```
2025-02-23 22:03:23.827 | INFO     | __main__:download_coins_price:11 - fe
tch price data, id=ethereum
2025-02-23 22:03:44.109 | INFO     | __main__:download_coins_price:11 - fe
tch price data, id=bitcoin
2025-02-23 22:04:04.543 | INFO     | __main__:download_coins_price:11 - fe
tch price data, id=binancecoin
2025-02-23 22:04:25.165 | INFO     | __main__:download_coins_price:11 - fe
tch price data, id=solana
2025-02-23 22:04:45.569 | INFO     | __main__:download_coins_price:11 - fe
tch price data, id=avalanche-2
2025-02-23 22:05:06.205 | INFO     | __main__:download_coins_price:11 - fe
tch price data, id=matic-network
2025-02-23 22:05:26.643 | INFO     | __main__:download_coins_price:11 - fe
tch price data, id=tron
```

In [9]:
```
mkt_cap_df = download_coins_mkt_cap()
```

```
2025-02-23 22:05:47.834 | INFO     | __main__:download_coins_mkt_cap:11 -
fetch mkt cap data, id=ethereum
2025-02-23 22:06:08.229 | INFO     | __main__:download_coins_mkt_cap:11 -
fetch mkt cap data, id=bitcoin
2025-02-23 22:06:28.624 | INFO     | __main__:download_coins_mkt_cap:11 -
fetch mkt cap data, id=binancecoin
2025-02-23 22:06:49.286 | INFO     | __main__:download_coins_mkt_cap:11 -
fetch mkt cap data, id=solana
2025-02-23 22:07:09.913 | INFO     | __main__:download_coins_mkt_cap:11 -
fetch mkt cap data, id=avalanche-2
2025-02-23 22:07:30.390 | INFO     | __main__:download_coins_mkt_cap:11 -
fetch mkt cap data, id=matic-network
2025-02-23 22:07:50.801 | INFO     | __main__:download_coins_mkt_cap:11 -
fetch mkt cap data, id=tron
```

Load TVL Data

In [11]:
```python
_CHAINS_TO_COINS_MAPPING_DICT = {
    'ethereum': 'Ethereum',
    'bitcoin': 'Bitcoin',
    'binancecoin': 'BSC',
    'solana': 'Solana',
    'avalanche-2': 'Avalanche',
    'matic-network': 'Polygon',
    'tron': 'Tron',
}

def load_tvl(chains_to_coins_mapping_dict: dict = _CHAINS_TO_COINS_MAPPIN
    reverse_dict = {chains_to_coins_mapping_dict[x]: x + '_tvl_usd' for x

    tvl_df = pd.read_csv('chains.csv')
    del tvl_df['Timestamp']
    tvl_df['Date'] = tvl_df['Date'].apply(lambda x: pd.to_datetime(x, for
    tvl_df.set_index('Date', inplace=True)
    target_coins = list(reverse_dict.keys())
    tvl_df = tvl_df[target_coins]
    tvl_df.rename(columns=reverse_dict, inplace=True)

    return tvl_df
```

In [12]:
```python
tvl_df = load_tvl()
```

Merge Datasets & Calculate MTVL

In [14]:
```python
dfm = pd.merge(price_df, mkt_cap_df, how='left', left_index=True, right_i
dfm = pd.merge(dfm, tvl_df, how='left', left_index=True, right_index=True
dfm = dfm.dropna()
```

In [15]:
```python
dfm.head()
```

Out[15]:

	ethereum_price_usd	bitcoin_price_usd	binancecoin_price_usd	solana_pric
2024-02-25	2988.317384	51553.096713	381.927653	104.1
2024-02-26	3113.408298	51751.884055	388.562714	103.4
2024-02-27	3173.629947	54478.191083	401.214285	109.8
2024-02-28	3242.047654	57003.526737	395.055555	108.1
2024-02-29	3380.803956	62558.582024	416.253849	117.4

5 rows × 21 columns

In [16]:
```
dfm['solana_mtvl'] = dfm['solana_mkt_cap_usd'] / dfm['solana_tvl_usd']
dfm['solana_mtvl_pct_chg'] = dfm['solana_mtvl'].pct_change()
dfm['solana_mtvl_pct_chg_t-1'] = dfm['solana_mtvl_pct_chg'].shift(1)
dfm['solana_usd_r'] = dfm['solana_price_usd'].pct_change()
dfm = dfm.dropna()
```

In [17]:
```
dfm.head()
```

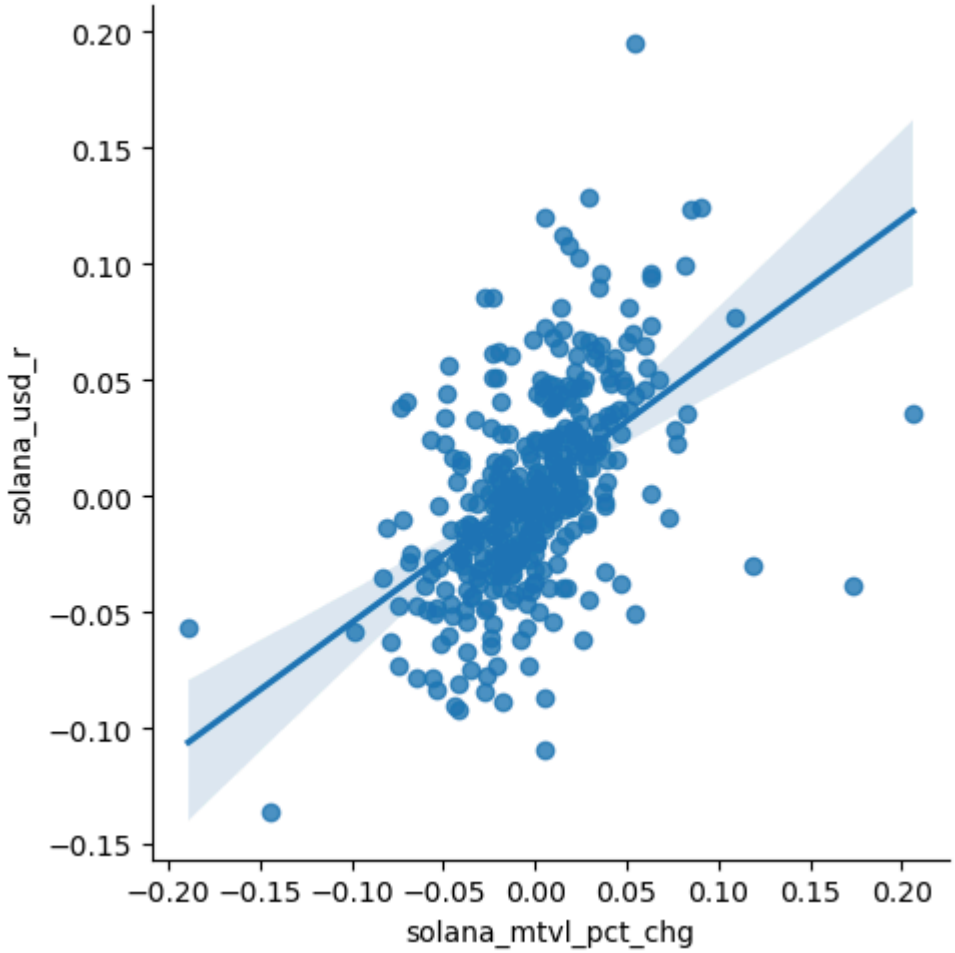
Out[17]:

	ethereum_price_usd	bitcoin_price_usd	binancecoin_price_usd	solana_pric
2024-02-27	3173.629947	54478.191083	401.214285	109.8
2024-02-28	3242.047654	57003.526737	395.055555	108.1
2024-02-29	3380.803956	62558.582024	416.253849	117.4
2024-03-01	3347.690472	61298.216861	399.786496	125.6
2024-03-02	3431.751998	62426.640529	407.348845	130.0

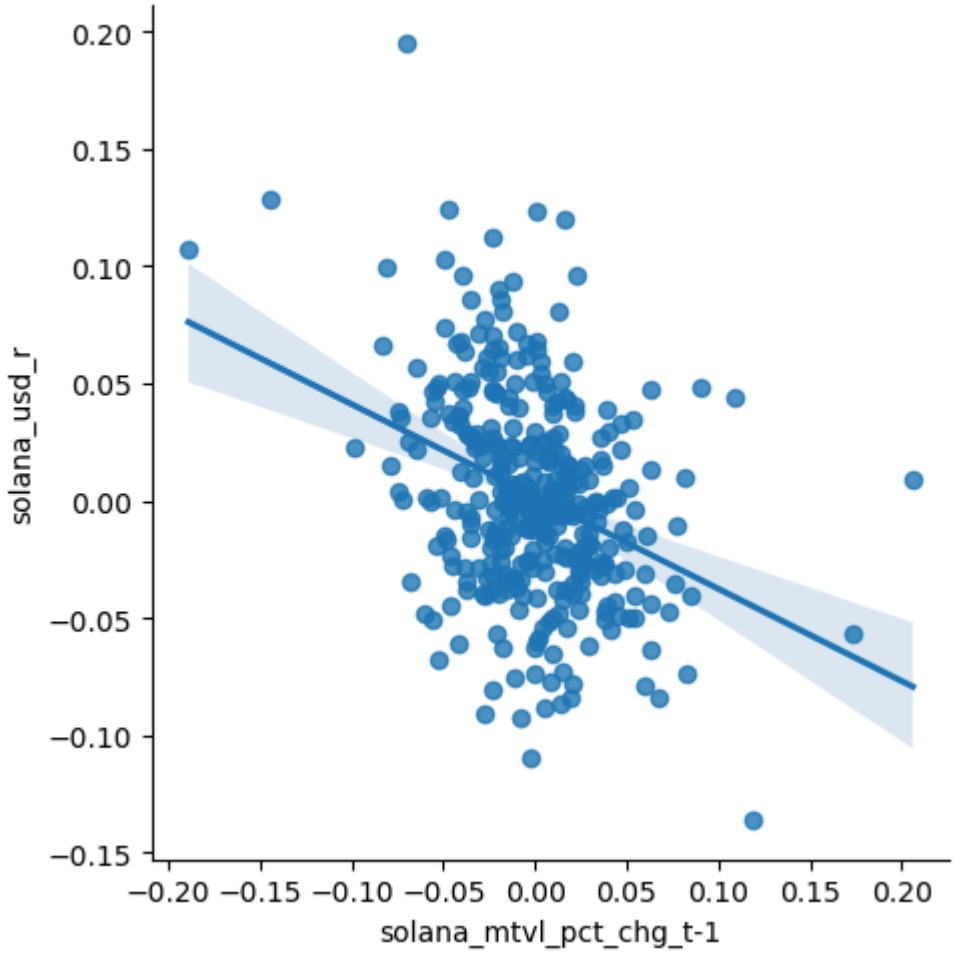
5 rows × 25 columns

Linear Regression: Can MTVL predict price change?

In [19]:
```
sn.lmplot(x='solana_mtvl_pct_chg', y='solana_usd_r', data=dfm, fit_reg=Tr
```



In [20]:
```
sn.lmplot(x='solana_mtvl_pct_chg_t-1', y='solana_usd_r', data=dfm, fit_re
```



In [21]:
```
mod1 = sm.OLS(dfm['solana_usd_r'], sm.add_constant(dfm['solana_mtvl_pct_c
res = mod1.fit()
print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
====
Dep. Variable:           solana_usd_r   R-squared:
0.261
Model:                            OLS   Adj. R-squared:
0.259
Method:                 Least Squares   F-statistic:                     1
27.5
Date:                Sun, 23 Feb 2025   Prob (F-statistic):           1.59
e-25
Time:                        22:07:52   Log-Likelihood:                 67
5.58
No. Observations:                 363   AIC:                            -1
347.
Df Residuals:                     361   BIC:                            -1
339.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
=============
                          coef    std err          t      P>|t|      [0.02
5      0.975]
------------------------------------------------------------------------------
-------------
const                   0.0032      0.002      1.624      0.105      -0.00
1       0.007
solana_mtvl_pct_chg     0.5790      0.051     11.292      0.000       0.47
8       0.680
==============================================================================
====
Omnibus:                       18.559   Durbin-Watson:
1.926
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               4
0.220
Skew:                           0.230   Prob(JB):                     1.85
e-09
Kurtosis:                       4.565   Cond. No.
25.9
==============================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
```

In [22]:
```
mod2 = sm.OLS(dfm['solana_usd_r'], sm.add_constant(dfm['solana_mtvl_pct_c
res = mod2.fit()
print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
====
Dep. Variable:           solana_usd_r   R-squared:
0.120
Model:                            OLS   Adj. R-squared:
0.118
Method:                 Least Squares   F-statistic:                      4
9.34
Date:                Sun, 23 Feb 2025   Prob (F-statistic):           1.08
e-11
Time:                        22:07:52   Log-Likelihood:                  64
3.94
No. Observations:                 363   AIC:                             -1
284.
Df Residuals:                     361   BIC:                             -1
276.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
=================
                              coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
-----------------
const                         0.0017      0.002      0.797      0.426      -
0.003       0.006
solana_mtvl_pct_chg_t-1      -0.3934      0.056     -7.024      0.000      -
0.504      -0.283
==============================================================================
====
Omnibus:                       10.172   Durbin-Watson:
1.738
Prob(Omnibus):                  0.006   Jarque-Bera (JB):                1
1.846
Skew:                           0.295   Prob(JB):                       0.0
0268
Kurtosis:                       3.659   Cond. No.
25.9
==============================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
```

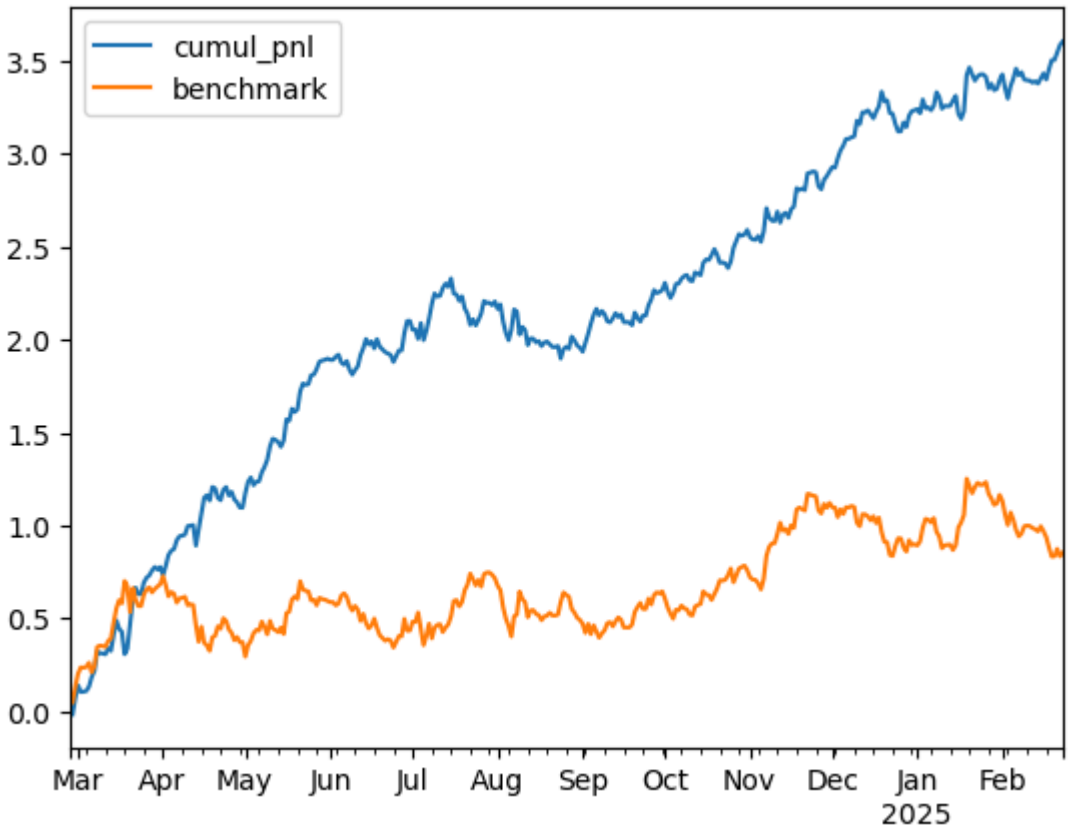
Strategy Performance

In [24]:
```python
def compute_SR(r_data: pd.Series) -> float:
    return r_data.mean() / r_data.std() * np.sqrt(365)
```

In [25]:
```python
bt_df = dfm[['solana_mtvl_pct_chg', 'solana_usd_r']]
bt_df['position'] = np.where(dfm['solana_mtvl_pct_chg'] < 0, 1,
                             np.where(dfm['solana_mtvl_pct_chg'] > 0, -1,
bt_df['pnl'] = bt_df['position'].shift() * bt_df['solana_usd_r']
bt_df['cumul_pnl'] = bt_df['pnl'].cumsum()
bt_df['benchmark'] = bt_df['solana_usd_r'].cumsum()
```

```
/var/folders/jz/_62zm7jn3y754w_n7gcp4v3m0000gn/T/ipykernel_23357/382552911
7.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  bt_df['position'] = np.where(dfm['solana_mtvl_pct_chg'] < 0, 1,
/var/folders/jz/_62zm7jn3y754w_n7gcp4v3m0000gn/T/ipykernel_23357/382552911
7.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  bt_df['pnl'] = bt_df['position'].shift() * bt_df['solana_usd_r']
/var/folders/jz/_62zm7jn3y754w_n7gcp4v3m0000gn/T/ipykernel_23357/382552911
7.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  bt_df['cumul_pnl'] = bt_df['pnl'].cumsum()
/var/folders/jz/_62zm7jn3y754w_n7gcp4v3m0000gn/T/ipykernel_23357/382552911
7.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  bt_df['benchmark'] = bt_df['solana_usd_r'].cumsum()
```

In [26]:
```
bt_df[['cumul_pnl', 'benchmark']].plot();
```



In [27]:
```
print('strat SR ' + str(compute_SR(bt_df['pnl'])))
print('benchmark SR ' + str(compute_SR(bt_df['solana_usd_r'])))
```

```
strat SR 4.45417713735506
benchmark SR 1.0254353617538454
```