TimeSeries Momentum - India (EOD Data)

Here I have created a time series momentum strategy for liquid futures contracts on Indian markets.

Methodology:

1. Aggregate Time Series Momentum

Universe: Futures contracts on the 60 stocks in the Nifty

Signal: N-day risk-scaled trend

Leverage Constraint: 1.5

Cost: 5 bps roundtrip

Position Sizing: Risk budgeting position sizing proportional to indicator values scaled to a risk of 15%

The lookbacks and rebalancing frequencies we try are -

<table>
<thead>
<tr>
<th>Lookback</th>
<th>Rebalancing Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>252</td>
<td>21</td>
</tr>
<tr>
<td>126</td>
<td>21</td>
</tr>
<tr>
<td>63</td>
<td>10</td>
</tr>
<tr>
<td>21</td>
<td>5</td>
</tr>
</tbody>
</table>

2. Relative Sector Momentum

Universe: AI liquid futures contracts

Signal: N-day risk-scaled trend

Leverage Constraint: 1.5

Cost: 5 bps roundtrip

Position Sizing: Risk budgeting position sizing proportional to indicator values scaled to a risk of 15%

Portfolio Combination: Equal Weighted

<table>
<thead>
<tr>
<th>Lookback</th>
<th>Rebalancing Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>252</td>
<td>21</td>
</tr>
<tr>
<td>126</td>
<td>21</td>
</tr>
<tr>
<td>63</td>
<td>10</td>
</tr>
<tr>
<td>21</td>
<td>5</td>
</tr>
</tbody>
</table>

Authors

- Sonam Srivastava
- Gaurav Chakravarty
- Manel Bragh

Optimal Project Management Link

```python
from datetime import date, timedelta
import pandas as pd
import plotly.express as px
import numpy as np
import matplotlib.pyplot as plt
import os
import matplotlib.pyplot as plt
import pandas as pd
import plotly.express as px

get_sector_information

RBA Logic

```python
df = pd.read_csv('file.csv')
sectors = {}
for i in df['sector'].unique():
    sectors[i] = df[df['sector'] == i]['Symbol', 'total_ton'].sum()
ds = df[df['sector'].isin(['Index', 'index'])]
df['Nifty'] = df['sector'].isin(['BSE500', 'index'])

def LEVERAGE_LMT = 1.5
    risk_budgeting_optimization
    def calculate_portfolio_var(w, V):
        # function that calculates portfolio risk
        v = np.sqrt(w.T @ V @ w)
        return v
    def calculate_risk_contribution(w, V):
        # function that calculates asset contribution to total risk
        w = np.abs(w)
        sig = np.sqrt(calculate_portfolio_var(w, V))
        # Marginal Risk Contribution
        MRC = V @ w,
        # Risk Contribution
        RC = np.multiply(MRC, V) / sig
        return RC
    def risk_budget_objective(c, target_risk, R, IR):
        # calculate portfolio risk
        x = np.abs(c) @ V @ c
        if target_risk > np.sqrt(c @ V @ c):
            # set target risk to np.sqrt(c @ V @ c)
            target_risk = np.sqrt(c @ V @ c)
            asset_risk = calculate_risk_contribution(c, V)
            J = np.sum(asset_risk / target_risk) ** 0.5
            return J
    def total_leverage_constraint(s):
        return LEVERAGE_LMT * np.sum(np.abs(s))
    def min_leverage_constraint(s):
        return np.sum(np.abs(s)) > 0.5
    def plot_risk_budgeting(c, target_risk, R, IR):
        c = np.abs(c) @ R @ c
        if target_risk > np.sqrt(c @ R @ c):
            target_risk = np.sqrt(c @ R @ c)
            asset_risk = calculate_risk_contribution(c, R)
            J = np.sum(asset_risk / target_risk) ** 0.5
            return J
```

```
```python
def get_pseudo_ma_weights(indicators, covariance, dates):
    weights = pd.DataFrame(index=indicators, columns=indicators.columns)
    for date in dates:
        if covariance(dates):
            x = indicators.loc(date, 't').tolist()
        else:
            x = self.ma_weighting_calculator(x, 3, v)
        weights.loc[date, :] = w
    return weights

Functions for strategy analysis

```
```python
import pickle
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import os
from datetime import datetime
import cftime
import matplotlib.dates as mdates
import seaborn as sns
from scipy.optimize import curve_fit
import statsmodels.api as sm

# Load data and process it

# Plotting

# Example of a bar chart
fig = plt.figure(figsize=(12, 8))
data = [30, 50, 80, 100]
labels = ['Group A', 'Group B', 'Group C', 'Group D']
plt.bar(labels, data)
plt.xlabel('Groups')
plt.ylabel('Values')
plt.title('Example Bar Chart')
plt.show()

# Example of a line chart
fig = plt.figure(figsize=(10, 6))
times = np.arange(0, 10, 0.1)
data1 = np.sin(2 * np.pi * times)
data2 = np.cos(2 * np.pi * times)
plt.plot(times, data1, label='Sine Wave')
plt.plot(times, data2, label=' cosine Wave')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Sine and Cosine Waves')
plt.legend()
plt.show()

# Example of a histogram
fig, ax = plt.subplots()
data = np.random.normal(0, 1, 1000)
bins = np.linspace(-3.5, 3.5, 20)
ax.hist(data, bins=bins, alpha=0.5, label='Normal Distribution')
av.set_ylabel('Frequency')
av.set_xlabel('Value')
av.legend()
plt.show()
```

---

The above code snippet demonstrates various plotting styles using matplotlib, such as bar charts, line charts, and histograms. It also includes examples of data processing and the use of libraries like numpy, pandas, and scipy for data manipulation and optimization.
Relative Sector Time Series Momentum

```python
import pickle as pkl
if not os.path.isfile('/home/sam/data/Sector_Neutral_75_EQ Option.pkl'):
    data_store = {}
else:
    data_store = pkl.load(open('/home/sam/data/Sector_Neutral_75_EQ Option.pkl', 'rb'))
print('Using pickled values')

LEVERAGE_LMT = 1.5

lookback = 252 days, rebalancing frequency = 21 days

lookback=252
rebalancing_freq=21
no_cost_ret = pd.DataFrame()
factor = []

def analyze(lookback, rebalancing_freq):
    global no_cost_ret, cost_ret
    if (lookback, rebalancing_freq) not in data_store:
        data_store[lookback, rebalancing_freq] = (all_returns_df, all_weights, eq_returns, stats)
    else:
        all_returns_df, all_weights, eq_returns, stats = data_store[lookback, rebalancing_freq]

    scaling_factor = get_annualized_sharpe_returns(eq_returns) / 10
    weights_df = pd.concat([all_weights, weight_corrected(all_returns_df, weight_df)]).resample('D').sum()
    cost_ret = pd.concat([no_cost_ret, cost_ret]).resample('D').sum()

    plt.rc('font', family='serif')
    plt.rc('figure', figsize=(6, 4))
    plt.rc('xtick', labelsize=10)
    plt.rc('ytick', labelsize=10)

    fig = plt.figure(figsize=(6, 2))
    ax = fig.add_subplot(1, 1, 1)

cost_ret.cumsum().plot(ax=ax)
plt.show()

get_summary_stats(...)  # use pd.Series(...)
```