A few weeks ago, Richard asked a question for the Grandmasters Challenge or something like that. Although I am not really engaged at the moment, I thought it was interesting, mainly, what would it take for the correlation of correlations to be 0. We haven't used the white board in a while, so we like any excuse to break it out, but the math is absolutely heinous. Just ignore the other notes/shopping list. I ran out of windex.

[

image

1439×1079 88 KB

](https://forum.numer.ai/uploads/default/original/2X/6/61b78fc33526a39a5e988807b988597598b91e20.jpeg)

There was another forum post that showed model predictions can be uncorrelated fairly easily. All you have to do is orthogonalize the features, with PCA for example, and separate those features into separate models. With linear models, the math is very clear the two model predictions should be uncorrelated. With nonlinear models, my intuition tells me that having x uncorrelated with y may still help f(x) be uncorrelated with g(y) especially if the nonlinear terms do not contribute as much as the linear terms. However, the post later notes that the model performance across eras were still correlated at about 0.50.

Essentially, we want the features to be orthogonal over time. Such as have one model track uncorrelated features set 1 and another model track uncorrelated features set 2. But when we PCA our panel data, I presume it doesn't specifically focus on the time component so the model performance is still correlated.

In my work, I used the very old method of factor models to try to predict returns. For those who are unfamiliar, a long time ago, people were like, "the market factor is all you need" and they were all doing asset pricing with CAPM. Then later on, like 30 years ago, Fama French found a few other factors that seemed helpful for predicting asset returns. I think I heard they had an investment firm based on the idea. Today, there's a factor zoo, and the latest I have heard is a paper proposing lasso to select interesting factors, but it was ultimately rejected from one of the top 3 finance journals. Essentially, the lasso is just a statistically method that can't really tell the difference between noise and real variation.

A factor is like some sort of underlying trait that some or all assets can have. By having this trait to some degree, the asset's performance is impacted accordingly. An example is a leverage factor. Some assets are more levered than others. If there was a macroeconomic shock that would make leverage more or less favorable, you would expect the asset's performance to move in accordance to this one factor. There's an idea that exposure to these factors (maybe you can consider as risk) should be correlated with some asset return/reward. These factors are time series, not panel data. I thought if one model would follow some orthogonal factors and another model follow other ones, that would lead to uncorrelated model performance.

Alright cool. All I did was apply the Fama French factor model method to estimate potential returns of individual assets. You can see my code below. The only difference was that I orthogonalized the factors with PCA first, divided up the variation, and then reconstituted the original values. I had to use signals data for this because that is panel data. However, if we ever get an asset id on the traditional data (or if you want to do the painstakingly laborious matching of assets over time yourself), I am hopeful it will work better with 1000s of factors than only the 22 in signals. This was the result 10% of the way through, but it did not hold. The correlation was still near 0, but model 2 had negative spearman corr with the target variable.

[

image

1439×1079 84.8 KB

](https://forum.numer.ai/uploads/default/original/2X/4/42faff0248b4bc541b467bc24e144d0f1072e8dd.jpeg)

I tried several things to make it better. One idea was the models were overfitting so I limited the pca variation to just the top 80% of explained variance and I tried some RIdge regression. Didn't seem to help. I also tried adjusting the period length to estimate the fama french risk premias and factor loadings, didn't seem to help. My code is also stupid slow. Ideally, I am wondering if we can somehow reconstitute the panel data from uncorrelated time series factors, but it's not clear to me at all. Good luck everyone.

```
from numerapi import NumerAPI napi = NumerAPI() napi.download_dataset("signals/v1.0/train.parquet", "signals_train.parquet") napi.download_dataset("signals/v1.0/validation.parquet", "signals_valid.parquet")

import numpy as np import pandas as pd from sklearn.linear_model import LinearRegression, Ridge from sklearn.decomposition import PCA import time

train = pd.read_parquet('signals_train.parquet') valid = pd.read_parquet('signals_valid.parquet') data = pd.concat((train, valid))

data_columns = data.columns other_cols = ['numerai_ticker','composite_figi','date','data_type'] feature_columns = [c for c in data_columns if c[0:4] == 'feat'] feature_columns.remove('feature_country') target_columns = [c for c in data_columns if
```

c[0:4] == 'targ']

data = data[feature_columns + ['numerai_ticker','date','target']] data.dropna(inplace=True) keep_assets = data['numerai_ticker'].unique() data = data.loc[data['numerai_ticker'].isin(keep_assets)]

unique_dates = data['date'].unique() period_pcorr1 = [] period_pcorr2 = [] period_rcorr1 = [] period_rcorr2 = [] period_length = 52*3 start = time.time() for date in unique_dates[period_length:]:

```
start_date = unique_dates[unique_dates.tolist().index(date)-period_length]
modeling_data = data.loc[(data['date'] < date) & (data['date'] >= start_date)]

factor_returns = pd.DataFrame()
for feature in feature_columns:
    temp = modeling_data.copy()
    temp['grouped_feature'] = pd.qcut(temp[feature], q=3, labels=False, duplicates='drop')
    test = temp.groupby(['date','grouped_feature'])['target'].mean().reset_index()
    test = test.loc[test['grouped_feature'].isin([0,2])]
    test = test.pivot(index='date', columns='grouped_feature', values='target')
    test[feature+'_r'] = test[2] - test[0]
    factor_returns[feature+'_r'] = test[feature+'_r']

pca = PCA(n_components=len(feature_columns))
components = pca.fit_transform(factor_returns.values)
variance_explained = np.cumsum(pca.explained_variance_ratio_)

p40 = np.argmin(np.abs(variance_explained - 0.4))
model1_features = components.copy()
model1_features[:, p40+1:] = 0
model1_features_recovered = pca.inverse_transform(model1_features)

p80 = np.argmin(np.abs(variance_explained - 0.8))
model2_features = components.copy()
model2_features[:, :p40+1] = 0
model2_features[:, p80+1:] = 0
model2_features_recovered = pca.inverse_transform(model2_features)

assets = modeling_data['numerai_ticker'].unique()
model1_asset_betas = {}
model2_asset_betas = {}
for asset in assets:
    temp = modeling_data.loc[modeling_data['numerai_ticker'] == asset]
    asset_dates = temp['date']
    keep_features_index = factor_returns.index.isin(asset_dates)
    asset_targets = temp['target']

    model1 = Ridge(alpha=1).fit(model1_features_recovered[keep_features_index, :], asset_targets)
    # model1 = LinearRegression().fit(model1_features_recovered[keep_features_index, :], asset_targets)
    model1_asset_betas[asset] = model1.coef_

    model2 = Ridge(alpha=1).fit(model2_features_recovered[keep_features_index, :], asset_targets)
    # model2 = LinearRegression().fit(model2_features_recovered[keep_features_index, :], asset_targets)
    model2_asset_betas[asset] = model2.coef_

model1_risk_premias = model1_features_recovered.mean(axis = 0)
model2_risk_premias = model2_features_recovered.mean(axis = 0)

period_predictions = []
period_data = data.loc[data['date'] == date]
period_assets = period_data['numerai_ticker'].unique()
for asset in period_assets:
    try:
        model1_pred = np.dot(model1_asset_betas[asset], model1_risk_premias)
        model2_pred = np.dot(model2_asset_betas[asset], model2_risk_premias)
        target = period_data.loc[period_data['numerai_ticker'] == asset]['target'].values[0]
        period_predictions.append({'asset':asset, 'date':date, 'model1':model1_pred, 'model2':model2_pred, 'target':target})
    except:
        pass

period_df = pd.DataFrame(period_predictions)
period_pcorr1.append(period_df['model1'].corr(period_df['target'], method='pearson'))
period_rcorr1.append(period_df['model1'].corr(period_df['target'], method='spearman'))
period_pcorr2.append(period_df['model2'].corr(period_df['target'], method='pearson'))
period_rcorr2.append(period_df['model2'].corr(period_df['target'], method='spearman'))

print(date, len(period_pcorr1), round((time.time()-start)/60))
print(np.mean(period_pcorr1), np.mean(period_pcorr2))
print(np.corrcoef([period_pcorr1, period_pcorr2]))
print(np.mean(period_rcorr1), np.mean(period_rcorr2))
print(np.corrcoef([period_rcorr1, period_rcorr2]))
print()
```