Portfolio Management Final Exam

1 Instructions

This is an *individual* exam. You are allowed to use books, notes, and online references but *not* to consult with any other person (aside from the instructor or graders). You are welcome to send any questions you may have to the instructor (at `gordon.ritter@gmail.com`) or the graders and they will be answered in a timely manner. Evidence of communication with other students or outside experts about the exam will result in a zero. Likewise, if two students submit substantially similar solutions, both will receive a zero. We will make every effort to be generous and fair with partial credit, so if you are unsure of how to complete a problem, you maximize the expected points simply by doing as much as you can without consulting others.

Completed exams must be submitted using the Blackboard system:

`https://bbhosted.cuny.edu/`

No exams will be accepted by email or hard copy, nor will late submissions be accepted. In practice you must submit your work a few hours before the deadline, since the website can freeze, your Internet connection can go down, etc. Resubmissions are allowed, so you should submit several hours before the deadline, and then double-check it and resubmit if needed.

When you submit your solutions, if a notebook is used (such as `.ipynb` format), you must submit both the notebook file and also a PDF export. The PDF file must contain all code, comments and diagrams that were available in the notebook. If you don't use notebooks, then submit your raw source code such as `.py` files, along with a human-readable PDF summarizing your results with all plots.

In any case all code should be profusely commented. Every function should come with a comment explaining what it does, and what the inputs and outputs are supposed to be.

2 Project Introduction

This project will test your knowledge and understanding of the skills presented in class by means of a semi-realistic project on a realistically-sized stock data set.

The data is provided as a sequence of compressed python pickle files. Hence you need to use python – at least initially, to load the data. You can then either continue to use python for the rest of the project, or use python to export the data in a common format, and continue in another language. However we expect most students will choose to continue in python. The hints given below also reference python code.

The model data has already been pre-processed from text files and saved into pickle files. Loading and parsing the raw files (parsing text into numbers) can slow down your backtest significantly, which is why it's important to pre-process the data beforehand.

Before proceeding further, here is a very inclusive set of imports which probably contains more packages than you will actually need.

```
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import pickle
    from statsmodels.formula.api import ols
    from scipy.stats import gaussian_kde
    import scipy
    import scipy.sparse
    import patsy
    from statistics import median
    import bz2
```

There are two types of files you need to load. Most of the cross sectional equity data is contained in files starting with `pandas-frames`. The factor covariance data is not shaped the same way at all, so factor covariance matrices are in their own files. The following code snippet will load both sets of files into memory – you will have to change `model_dir` of course.

```
model_dir = '/Users/gordonritter/Dropbox/teaching/'

frames = {}
for year in range(2003,2011):
    fil = model_dir + "pandas-frames." + str(year) + ".pickle.bz2"
    frames.update(pickle.load( bz2.open( fil, "rb" ) ))

covariance = {}
for year in range(2003,2011):
    fil = model_dir + "covariance." + str(year) + ".pickle.bz2"
    covariance.update(pickle.load( bz2.open(fil, "rb" ) ))
```

You can inspect one of the frames with a command like `frames['20040102'].head()`. Here is a dictionary which helps you interpret the data contained in each of the frames:

ID: a unique identifier that can be used to link stocks across time

BETA: risk factor computed from CAPM beta regression

MOMENTUM: 12-month growth in stock price, usually a risk factor

SIZE: risk factor based on log(market capitalization)

VALUE: risk factor based on ratio of tangible book value to current price

1DREVRSL: very short-term reversal, potential alpha factor but probably too fast-moving to be tradable

STREVRSL: short-term reversal, potential alpha factor

LTREVRSL: long-term reversal, potential alpha factor

EARNQLTY: earnings quality, potential alpha factor

EARNYILD: earnings yield (blend of forecasted earnings and historical earnings divided by market cap)

GROWTH: mix of historical and forecasted earnings growth

LEVERAGE: financial leverage of the company's balance sheet, usually a risk factor

LIQUIDTY: factor with high loadings for very liquidly traded names; usually a risk factor

MGMTQLTY: management quality, potential alpha factor which looks at quantitative measures of how well-run a company is by its management

PROFIT: profitability, potential alpha factor

PROSPECT: based on skewness of the return distribution, potential risk factor

RESVOL: risk factor computed from residual volatility

SEASON: seasonality-based potential alpha factor

SENTMT: news sentiment, potential alpha factor

INDMOM: industry momentum (defined as relative historical outperformance or underperformance of the other stocks in the same industry)

SpecRisk: specific risk is another name for predicted residual volatility.

TotalRisk: predicted total vol, including factor and idiosyncratic contributions, annualized

Ret: asset's total return on the next day after the factor loadings are known

Yield: the dividend yield of the asset

HistBeta: historically estimated CAPM beta coefficient

PredBeta: model-predicted beta coefficient in the future

IssuerMarketCap: aggregate market capitalization of the company (all share classes from the same issuer)

BidAskSpread: bid-offer spread (average for the day)

CompositeVolume: composite trading volume for the day

DataDate: the date when the data would have been known, as of the close

```
Industry factors:
      'AERODEF', 'AIRLINES', 'ALUMSTEL', 'APPAREL', 'AUTO', 'BANKS','BEVTOB', 'BIOLIFE', 'BLDGPROD',
      'CHEM', 'CNSTENG', 'CNSTMACH', 'CNSTMATL', 'COMMEQP', 'COMPELEC', 'COMSVCS', 'CONGLOM', 'CONTAINR',
      'DISTRIB', 'DIVFIN', 'DIVYILD', 'DWNRISK','ELECEQP', 'ELECUTIL', 'FOODPROD', 'FOODRET', 'GASUTIL',
      'HLTHEQP', 'HLTHSVCS', 'HOMEBLDG', 'HOUSEDUR','INDMACH', 'INSURNCE', 'INTERNET', 'LEISPROD',
      'LEISSVCS', 'LIFEINS', 'MEDIA', 'MGDHLTH','MULTUTIL','OILGSCON', 'OILGSDRL', 'OILGSEQP',
      'OILGSEXP', 'PAPER', 'PHARMA','PRECMTLS','PSNLPROD','REALEST', 'RESTAUR', 'ROADRAIL','SEMICOND',
      'SEMIEQP','SOFTWARE', 'SPLTYRET', 'SPTYCHEM', 'SPTYSTOR', 'TELECOM', 'TRADECO', 'TRANSPRT', 'WIRELESS'
```

3 Exam Problems

3.1 Factor return estimation

Problem 3.1. In short, estimate the factor returns of the factors listed as "potential alpha factors". For each such factor designated as an alpha factor in the list above, do the following. Iterate over all dates in the sample, and for each date, carry out steps (a)-(d) below:

(a) build a matrix X containing the single alpha factor, and also the "canonical risk factors". The canonical risk factors are defined to be all industry factors, and also the so-called "style factors" `BETA, SIZE, MOMENTUM, VALUE`. Use only rows where `IssuerMarketCap` is greater than 1 billion dollars. You may have to clean out any NaN values, because the matrix X needs to consist of finite numerical values.

(b) Build a vector R of the same length as the number of rows in X, using the column `Ret`. Winsorize the returns to ± 0.25, to dampen the effect of outliers.

(c) Estimate the model $R = Xf + \epsilon$ using multivariate OLS.

(d) Get the coefficient of the alpha factor in $\hat{\beta}$ from the regression in (c). You can discard the other coefficients, as they will not be used in what follows. Stuff the alpha factor coefficient in a time series associated to the given date.

Running one OLS per day over several years, where each OLS involves several thousand observations and about 50-100 independent variables, takes a few minutes (at least, it takes that long in python, because python is quite slow compared to java or c-derived languages such as c, c++, c#).

You now have a time series of coefficients for the alpha factor. Plot the cumulative sum of the coefficients and calculate the sharpe ratio. In your plots, make sure the dates appear correctly on the horizontal axis.

Comment on whether this time series is achievable as the returns on an investment strategy, both with and without trading-costs.

Here is some sample code which may help you get going. You do not have to use it.

```
def wins(x,a,b):
    return(np.where(x <= a,a, np.where(x >= b, b, x)))

def estimate_factor_returns(df, alpha):
    estu = df.loc[df.IssuerMarketCap > 1e9].copy(deep=True)
    estu['R'] = wins(estu['Ret'], -0.25, 0.25)

    myformula = # TODO something useful here
    model = ols(myformula, data=estu)
    results = model.fit()
    return(results)

facret = {}

for date in frames:
    facret[date] = estimate_factor_returns(frames[date], alpha).params
```

3.2 Hints for the next problem

In the following problem we will do a simple transaction cost-free portfolio optimization. Please refer to the course notes for notation common to factor models. Here, F is the covariance matrix of the canonical risk factors mentioned above, and X is the matrix of factor loadings, same as lecture. Also h denotes an n-vector of dollar holdings, where n is the number of assets available on a given date.

When we calculate the portfolio risk that is attributable to the risk factors: $h^T X F X^T h$, note that this can become computationally infeasible and/or slow. Use matrix factorization and carefully choose the order of matrix multiplications to avoid creating an $n \times n$ matrix.

The technique is a simple matrix factorization, wherein

$$Q := F^{1/2} X^T$$

is defined so that

$$Q^T Q = X F X^T$$

The portfolio's variance that is specific to each asset is found by combining the weights with the specific variance matrix:

$$h^T D h$$

where D is an $n \times n$ matrix, and h is an $n \times 1$ column vector.

Recall that D is a diagonal matrix, so all the off-diagonals are zero. So instead of doing the matrix multiplication, we could save computation time by working with the vector containing the diagonal values:

$$h^T D h = \sum_i^N (h_i^2 \times D_{i,i})$$

In this data set, D has already been estimated for you, in the column `SpecRisk`, but when denoting volatilities, be aware that the preparer of the data may have used annualized percent, and you are doing a *daily* optimization! More generally, always be aware of the units and numerical scales of all variables in a model.

3.3 Costless Portfolio optimization

Problem 3.2. In this problem we will set up a portfolio optimization of the same kind discussed in class.

(a) For each of the alpha factors in the previous problem, calculate its mean factor return over the first 2/3 of the date range (the "in-sample period"), and store these coefficients in a vector. In what follows, the "combined alpha" will be the linear combination of the alpha factors with these coefficients, and the combined alpha will only ever be calculated over the last 1/3 of the dates (the "out of sample period").

(b) Write code which subsets the covariance matrix to the canonical risk factors. Hence you will be able to calculate F on any given date.

(c) The objective function is: factor risk + idiosyncratic risk - expected portfolio return

$$f(h) = \frac{1}{2}\kappa\ h^T Q^T Q h + \frac{1}{2}\kappa\ h^T D h - \alpha^T h$$

Where $Q := F^{1/2} X^T$ is defined so that $Q^T Q = X F X^T$, and where α is the combined alpha vector from part (a). For each date in the out-of-sample period, find h^* that minimizes the objective, where $\kappa = 10^{-6}$ using a numerical optimization routine, such as the routines that can be found in the `scipy.optimize` package. Choose a routine that allows the user to specify the exact gradient as a function, ad give it the exact gradient. Ensure that neither your computation of $f(h)$, nor of the gradient, will ever form an $n \times n$ matrix where n is the number of assets. The largest matrix you should be using is of dimension $n \times k$, where k is the number of factors.

(d) For every day in the out of sample period, compute the pre-t-cost profit as $h^* \cdot R$ where h^* is the solution to part (c), and R is the return variable from above. Plot the cumulative sum, and print out the sharpe ratio of the out-of-sample $h^* \cdot R$.

(e) For every day in the out of sample period, do a variance decomposition on h^*. First, compute the total vol $(h'XFX'h' + h'Dh)^{1/2}$ where $h = h^*$ is the optimal portfolio, and plot the time-series of the total vol over the out of sample period. Then plot the fraction of the total variance that is idiosyncratic, industry, and style (these three kinds of variance fractions should add up to 1), again over the out of sample period. For avoidance of doubt, recall that the "style factors" are taken to be `BETA, SIZE, MOMENTUM, VALUE`.

(f) Plot the long, short, net and gross market values, in dollars, for every optimal portfolio in the out of sample period.

Here is some sample code which may help you get going. You do not have to use it.

```
XT = X.transpose()
Q = np.matmul(scipy.linalg.sqrtm(Fvar), XT);

def f(h):
    tmp = 0.0
    tmp += 0.5 * kappa * np.sum( np.matmul(Q, h) ** 2 )
    tmp += 0.5 * kappa * np.dot(h ** 2, specVar)
    tmp -= np.dot(h, alpha_vec)
    return(tmp)

def grad(h):
    # TODO calculate gradient

optimizer_result = scipy.optimize.fmin_l_bfgs_b(f, h0, fprime=grad)

# part (f)
df = pd.DataFrame(index = out_of_sample)
for dt in out_of_sample:
    h = # TODO get portfolio for this date
    df.at[dt,"long"] = np.sum(h[h>0])
    df.at[dt,"short"] = np.sum(h[h<0])
    df.at[dt,"net"] = np.sum(h)
    df.at[dt,"gross"] = np.sum(np.abs(h))
```

Problem 3.3. Derive a formula for the exact minimizer of $f(h)$ using multivariable calculus and linear algebra. Recompute the time-series $h^* \cdot R$, again over the out-of-sample period, where h^* is computed using the exact formula. Verify that, up to "small" numerical errors (not large enough to change the economic profitability in dollars), you get the same series that was previously obtained using the numerical optimization routine. (You do NOT have to re-do the other parts of the problem since the results would be substantially the same!)