Multi risk factor model
Constraint Variance optimization
Long/short 100/100 market portfolio
Long/short 130/30 portfolio

DUPREY Stefan

Plan

1. Factor model and stochastic processes
2. Covariance matrix
3. Regressing returns over factor loading/exposure
4. Minimum variance optimization rebalancement at each date
5. Long/short 130/30 portfolio
6. Long/Short 100/100 Market Neutral portfolio
7. R code : computing factors loadings
8. R code : estimating factors covariance matrix and idiosyncratic risk
9. R code : quadratic programming, constraints and multi risk factor models

1. Factor model and stochastic processes

We have $N \in \mathbb{N}$ stocks. Their returns $R_i, \forall i \in \{1, \cdots, N\}$ are modeled by N random processes χ_i corresponding to idiosyncratic risk together with K random processes f_A :

$$R_i = \chi_i + \sum_{A=1}^{K} \widetilde{\Omega}_{iA} f_A \tag{1}$$

$$\langle \chi_i, \chi_j \rangle = \xi_{ij}^2 \delta_{ij} \tag{2}$$

$$\langle \chi_i, f_A \rangle = 0 \tag{3}$$

$$\langle f_A, f_B \rangle = \Phi_{AB} \tag{4}$$

$$\langle R_i, R_j \rangle = \Theta_{ij} \tag{5}$$

2. Covariance matrix

The R_i covariance matrix can be written as :

$$\Theta = \Xi + \widetilde{\Omega}\Phi\widetilde{\Omega}^{\mathsf{T}} \tag{6}$$

where $\Xi_{ij} = \xi_{ij}^2 \delta_{ij}$ is the diagonal idiosyncratic part of the variance and Φ the $K \times K$ factor covariance matrix, $\widetilde{\Omega}$ is the $K \times K$ factor loading matrix. Note that $K << N$ gives more out of samples stability. Through Cholesky :

$$\Theta = \Xi + \Omega\Omega^{\mathsf{T}} \tag{7}$$

where $\Omega = \widetilde{\Omega}\widetilde{\Phi}$ and $\widetilde{\Phi}\widetilde{\Phi}^{\mathsf{T}} = \Phi$
Note that the standard variance for a portfolio with weights ω decomposes as :

$$\omega^{\mathsf{T}}\Omega\omega = \sum_{i=1}^{N} \xi_{ij}^2 + f^{\mathsf{T}}\Phi f \tag{8}$$

where $f = \Omega\omega$ is the factor values for our portfolio with weights ω

3. Regressing returns over factor loading/exposure

At each observation time we have to regress our stock returns to our stock factors :

$$R(t) = \sum_{k=1}^{K} \beta_k(t) \times F_k(t) + \epsilon(t) \tag{9}$$

$$R(t) = \sum_{k=1}^{K} f_k(t) \times \Omega_k(t) + \epsilon(t) \tag{10}$$

Ω_i, F_i is the factor exposure or loading for the stock i and β_i, f_i is the factor return or factor itself.

The variance of $\epsilon(t)$ is the idiosyncratic risk for stock i

The covariance of $(f_k(t))_{k\in\{1,\cdots,K\}}$ is the factor covariance matrix.

Here a proprietary strategy is to be defined to compute the factor matrix : a rolling standard deviation or garch volatility estimation for idiosyncratic variance and a rolling Ledoit-Wolf covariance matrix estimation

4. Minimum variance optimization rebalancement at each date

$$\operatorname{argmin} \quad \left\{\omega\Sigma\omega^{\intercal} - q \times \mu^{\intercal}\omega\right\}, \textit{becomes}$$
$$\begin{cases} \omega \in \mathbb{R}^N, \omega_i > 0, \omega_i < 1 \\ \sum_{i=1}^{N} \omega_i = 1 \\ \sum_{i=1}^{n} \omega_i\beta_i = \sum_{i=1}^{n} \omega_i \frac{\mathrm{Cov}(R_i, R_{market})}{\mathrm{Var}(R_i)} = 1 \end{cases} \tag{11}$$

$$\operatorname{argmin} \quad \left\{xQx^{\intercal} - q \times \widetilde{\mu}^{\intercal}x\right\} \tag{12}$$
$$\begin{cases} x = (f, \omega) \in \mathbb{R}^{(N+K)}, \omega_i > 0, \omega_i < 1 \\ \sum_{i=1}^{N} \omega_i = 1 \\ \sum_{i=1}^{n} \omega_i\beta_i = \sum_{i=1}^{n} \omega_i \frac{\mathrm{Cov}(R_i, R_{market})}{\mathrm{Var}(R_i)} = 1 \\ \Omega\omega = f \end{cases}$$

where :

$$Q = \left(\begin{array}{c|ccc} \Phi & & 0 \cdots 0 & \\ \hline & \xi_{11}^2 & & \mathbf{0} \\ 0 & & \ddots & \\ & & & \ddots \\ \vdots & \mathbf{0} & & & \xi_{nn}^2 \\ 0 & & & \end{array}\right), \widetilde{\mu} = \begin{pmatrix} 0 \\ \mu \end{pmatrix} \tag{13}$$

5. Long/short 130/30 portfolio

$$\operatorname{argmin} \quad \{xQx^{\intercal} - q \times \widetilde{\mu}^{\intercal} x\} \tag{14}$$

$$\begin{cases} x = (f, \omega) \in \mathbb{R}^{(N+K)} \\ v \in \mathbb{R}^{(N)} \\ -0.1 < \omega_i < 0.1 \\ \sum_{i=1}^{N} \omega_i = 1 \\ \sum_{i=1}^{n} \omega_i \beta_i = \sum_{i=1}^{n} \omega_i \frac{\mathrm{Cov}(R_i, R_{market})}{\mathrm{Var}(R_i)} = 1 \\ \Omega\omega = f \\ -v < \omega < v \\ v > 0 \\ \sum_{i=1}^{N} v_i = 1.6 \end{cases}$$

where :

$$Q = \left(\begin{array}{c|ccc} \Phi & & 0 \cdots 0 & \\ \hline & \xi_{11}^2 & & \mathbf{0} \\ & & \ddots & \\ 0 & & & \\ & & & \ddots \\ \vdots & \mathbf{0} & & \xi_{nn}^2 \\ 0 & & & \end{array} \right), \widetilde{\mu} = \begin{pmatrix} 0 \\ \mu \end{pmatrix} \qquad (15)$$

6. Long/Short 100/100 Market Neutral portfolio

$$\text{argmin} \qquad \{xQx^{\intercal} - q \times \widetilde{\mu}^{\intercal} x\} \tag{16}$$

$$\begin{cases} x = (f, l\omega, s\omega) \in \mathbb{R}^{(N+3*K)} \\ b \in \mathbb{R}^{(N)} \\ 0 < l\omega_i < 0.1, 0 < s\omega_i < 0.1 \\ \sum_{i=1}^{N} l\omega_i = \sum_{i=1}^{N} s\omega_i \\ \sum_{i=1}^{N} l\omega_i + \sum_{i=1}^{N} s\omega_i = 2 \\ \sum_{i=1}^{n} l\omega_i\beta_i = \sum_{i=1}^{n} s\omega_i\beta_i \\ \Omega l\omega - \Omega s\omega = f \\ l\omega < b \\ s\omega < 1 - b \end{cases}$$

where :

$$Q = \left(\begin{array}{c|cc} \Phi & 0 \cdots 0 & \\ \hline 0 & Q_{idio} & -Q_{idio} \\ 0 & -Q_{idio} & Q_{idio} \end{array}\right), Q_{idio} = \begin{pmatrix} \xi_{11}^2 & & & \mathbf{0} \\ & \ddots & & \\ & & \ddots & \\ \mathbf{0} & & & \xi_{nn}^2 \end{pmatrix} \tag{17}$$

7. R code : computing factors loadings

```r
# combine sector dummies and all factors
all.data = abind(sectors.matrix, factors.matrix)

# create betas and specific.return
beta = all.data[,1,] * NA
specific.return = next.month.ret * NA
nfactors = ncol(beta)

# append next.month.ret to all.data
all.data = abind(next.month.ret, all.data, along = 3)
dimnames(all.data)[[3]][1] = 'Ret'

# estimate betas (factor returns)
for(t in 12:(nperiods-1)) {
  temp = all.data[t:t,,]
  # first methodology : we deal ourself with the intercept by removing the first column and readjusting the b afterwards
#   x = temp[,-c(1:2)]
#   y = temp[,1]
#   # b = lm(y~x)$coefficients
#   b = tryCatch( lm(y~x)$coefficients,
#                error = function(e) {matrix(NaN,nrow=1,ncol=(dim(x)[2]+1))})
#   # robust linear model
#   #   load.packages('MASS')
#   #   temp = rlm(y~x)$coefficients
#   # quantile regression
#   #   load.packages('quantreg')
#   #   temp = rq(y ~ x, tau = 0.5)$coefficients
#   b[2:nsectors] = b[1] + b[2:nsectors]

  ### other methodology : we use wilkinson notation in R lm function to specify that we want to drop the intercept
  # which is already a linear combination of the columns
  x = temp[,-c(1)]
  y = temp[,1]
  # b = lm(y~x)$coefficients
  b = tryCatch( lm(y~-1+x)$coefficients,
               error = function(e) {matrix(NaN,nrow=1,ncol=(dim(x)[2]))})

  beta[(t+1),] = b

  specific.return[(t+1),] = y - rowSums(temp[,-1] * matrix(b, n, nfactors, byrow=T), na.rm=T)
}
```

8. R code : estimating factors covariance matrix and idiosyncratic risk

```
factor.covariance = array(double(), c(nperiods, nfactors, nfactors))
dimnames(factor.covariance)[[2]] = colnames(beta)
dimnames(factor.covariance)[[3]] = colnames(beta)

# estimate factor covariance with a one month rolling window
for(t in 36:nperiods) {
  factor.covariance[t,,] = tryCatch(var.shrink.eqcor(beta[(t-23):t,]),
                                    error = function(e) {matrix(NaN,nrow=nfactors,ncol=nfactors)})
}

#*****************************************************************
# Compute stocks specific variance foreasts using GARCH(1,1)
#*****************************************************************
specific.variance = next.month.ret * NA
for(i in 1:n) {
  specific.variance[,i] = bt.forecast.garch.volatility(specific.return[,i], 24)
}

# compute historical specific.variance
hist.specific.variance = next.month.ret * NA
for(i in 1:n) hist.specific.variance[,i] = runSD(specific.return[,i], 24)

# if specific.variance is missing use historical specific.variance
specific.variance[] = ifna(coredata(specific.variance), coredata(hist.specific.variance))
```

9. R code : quadratic programming, constraints and multi risk factor models

```r
for(t in 36:nperiods) {
  #--------------------------------------------------------------------------
  # Split x into x.long and x.short, x_long and x_short >= 0
  # SUM(x.long) - SUM(x.short) = 0
  #--------------------------------------------------------------------------
  # x.long and x.short >= 0
  # x.long <= 0.1
  # x.short <= 0.1
  constraints = new.constraints(2*n, lb = 0, ub = c(rep(0.1,n), rep(0.1,n)))
  # SUM (x.long - x.short) = 0
  constraints = add.constraints(c(rep(1,n), -rep(1,n)), 0, type = '=', constraints)
  # SUM (x.long + x.short) = 2
  constraints = add.constraints(c(rep(1,n), rep(1,n)), 2, type = '=', constraints)
  #--------------------------------------------------------------------------
  # beta of portfolio is the weighted average of the individual asset betas       # http://www.duke.edu/~charvey/Classes/ba350/riskman/riskman.htm
  #--------------------------------------------------------------------------
  temp = ifna(as.vector(beta[t,]),0)
  constraints = add.constraints(c(temp, -temp), type = '=', b = 0, constraints)
  #--------------------------------------------------------------------------
  # Create factor exposures constraints adjust prior constraints, add factor exposures
  #--------------------------------------------------------------------------
  constraints = add.variables(nfactors, constraints)
  # BX - X1 = 0
  temp = ifna(factor.exposures[t,,], 0)
  constraints = add.constraints(rbind(temp, -temp, -diag(nfactors)), rep(0, nfactors), type = '=', constraints)
  #--------------------------------------------------------------------------
  # Add binary constraints    # adjust prior constraints: add b.i
  #--------------------------------------------------------------------------
  constraints = add.variables(n, constraints)
  # index of binary variables b.i
  constraints$binary.index = (2*n + nfactors + 1):(3*n + nfactors)
  # binary variable b.i : x.long < b, x.short < (1 - b)
  # x.long < b
  constraints = add.constraints(rbind(diag(n), 0*diag(n), matrix(0,nfactors,n), -diag(n)), rep(0, n), type = '<=', constraints)
  # x.short < (1 - b)
  constraints = add.constraints(rbind(0*diag(n), diag(n), matrix(0,nfactors,n), diag(n)), rep(1, n), type = '<=', constraints)
  #--------------------------------------------------------------------------
  # set expected return
  #--------------------------------------------------------------------------
  # set expected return
  alpha = factors.avg$AVG[t,] / 5
  temp = ifna(coredata(alpha),0)
  expected.return = c(temp, -temp, rep(0, nfactors), rep(0, n))
  #--------------------------------------------------------------------------
  # Create Covariance matrix
  # [Qu  0]
  # [ 0 Qf]
  #--------------------------------------------------------------------------
  temp = diag(n)
  diag(temp) = ifna(specific.variance[t,], mean(coredata(specific.variance[t,]), na.rm=T))^2
  # | cov -cov |
  # |-cov  cov |
  temp = cbind( rbind(temp, -temp), rbind(-temp, temp) )
  cov.temp = 0*diag(2*n + nfactors + n)
  cov.temp[1:(2*n),1:(2*n)] = temp
  cov.temp[(2*n+1):(2*n+nfactors),(2*n+1):(2*n+nfactors)] = factor.covariance[t,,]
  #-
```