Project

members: Jueyi Liu(jueyiliu), Wenxin Wei(wxwei), Jiachen Ge(j5ge), Chih-Hsuan Kao(chkao831)

Problem 1

(1) WTS The Black-Litterman

$$\text{mean} = [(\tau\Sigma)^{-1} + P'\Omega^{-1}P]^{-1}[(\tau\Sigma)^{-1}\pi + P'\Omega^{-1}q]$$

$$\text{variance} = [(\tau\Sigma)^{-1} + P'\Omega^{-1}P]^{-1}$$

pf we are given that

$$\boldsymbol{P}\boldsymbol{\mu} = \boldsymbol{q} + \boldsymbol{\epsilon}^{(v)}, \quad \boldsymbol{\epsilon}^{(v)} \sim N(0, \boldsymbol{\Omega}), \qquad (4)$$

and $$\boldsymbol{\mu} = \boldsymbol{\pi} + \boldsymbol{\epsilon}^{(e)}, \quad \boldsymbol{\epsilon}^{(e)} \sim N(0, \tau\boldsymbol{\Sigma}), \Rightarrow \mu \sim (\pi, \tau\Sigma) \quad (1)$$

in which $\boldsymbol{\pi}$ corresponds to the ”equilibrium risk premium” of the risky assets.

From (1), the dist. of the data equilibrium return conditioned on investor's prior belief is

$$\pi \mid \mu \sim N(\mu, \tau\Sigma) \quad — \quad (1')$$

' denotes transpose of matrix

From (4), the pdf of $P\mu$ is given by

$$\frac{p^*}{\sqrt{2\pi_c|\Omega|}} \exp\left(-\frac{1}{2}(P\mu - q)'\Omega^{-1}(P\mu - q)\right)$$

From (1'), the pdf of $(\pi \mid \mu)$ is given by

$$\frac{p^*}{\sqrt{2\pi_c|\tau\Sigma|}} \exp\left(-\frac{1}{2}(\pi - \mu)'(\tau\Sigma)^{-1}(\pi - \mu)\right)$$

where p^* is the # exact linear belief constraints

q is $p \times 1$, view vector

P is $p \times m$, investor's individual picks of assets

μ is $m \times 1$, investor-expected excess returns

Ω is $p \times p$, diagonal cov matrix of uncertainty in the estimate μ of mean

τ is scalar of Σ

Σ is $m \times m$ covariance matrix (known) ↳ historical

$\underset{m\times 1}{\Pi} = r \Sigma w^*$, the equilibrium risk premium

By Bayesian,

$$Pr(\mu \mid \pi) = \frac{Pr(\pi \mid \mu)\, Pr(\mu)}{Pr(\pi)}$$

Substituting pdf above,
the posterior density is proportional to

$$\exp\left(-\tfrac{1}{2}(\pi-\mu))'(\tau\Sigma)^{-1}(\pi-\mu) - \tfrac{1}{2}(p\mu - q)'\Omega^{-1}(p\mu-q)\right)$$

Let $A = (\tau\Sigma)^{-1} + p'\Omega^{-1}p$

$B = (\tau\Sigma)^{-1}\pi + p'\Omega^{-1}q$

$c = \pi'(\tau\Sigma)^{-1}\pi + q'\Omega^{-1}q$ for simplification

we can rewrite as

$$\exp\left(-\tfrac{1}{2}[\mu' A \mu - 2B'\mu + c]\right)$$

$$= \exp\left(-\tfrac{1}{2}[\mu' A' A A^{-1'} \mu - 2B'A^{-1}A\mu + c]\right)$$

$$= \exp\left(-\tfrac{1}{2}[(A\mu - B)' A^{-1}(A\mu - B) - B'A^{-1}B + c]\right)$$

$$= \exp\left(-\tfrac{1}{2}[c - B'A^{-1}B]\right) \times \exp\left(-\tfrac{1}{2}(A\mu-B)'A^{-1}(A\mu-B)\right)$$

↙∵ has no dependency on μ

Incorporate unmodelled terms into the integrating constant for the posterior pdf of $(\mu \mid \pi)$.

Hence the posterior pdf $(\mu \mid \pi)$ is normal with mean

$$A^{-1}B = [(\tau\Sigma)^{-1} + p'\Omega^{-1}p]^{-1}[(\tau\Sigma)^{-1}\pi + p'\Omega^{-1}q]$$

and (co)variance $A^{-1} = [p'\Omega^{-1}p + (\tau\Sigma)^{-1}]^{-1}$ ■

↖∵ diagonal

1.(2)

```
library(MASS)
# prepare data
dat<-read.table("http://www.ams.sunysb.edu/~xing/statfinbook/_BookData/Chap03/m_ret_10stocks.txt",header=T)
dat<-as.matrix(dat[,2:11])
company<-colnames(dat)
sprf<-read.table("http://www.ams.sunysb.edu/~xing/statfinbook/_BookData/Chap03/m_sp500ret_3mtcm.txt",skip=1,
                 header=T)
rf<-sprf[,3]/1200
```

Individual views do not have influence on market portfolio. If you do not have views, you hold the market portfolio, which is the benchmark. If you do have view, your views will tilt the final weights away from the market portfolio, the degree to which depending on how confident you are about your views. In this case, we choose to calculate market porfolio without views. (By MG.ppt)

By p.74 in the book, "… weight vector $\Sigma^{-1}(\mu - r_f 1)/1^T\Sigma^{-1}(\mu - r_f 1)$, which is the fund or market portfolio in the one-fund theorem…", we use this method to calculate the market portfolio in the one-fund theorem.
The date is 12/7/2006. Since the problem mention that "Today's risk free rate today will be the same as yesterday's", we use the risk free rate at 12/1/2006 to be an estimate of the actual risk free rate.

```
covar<-unname(cov(dat))
mu<-apply(dat, 2, mean)
one<-(rep(1, dim(covar)[1]))
mktrf<-rf[156]
mktport<-as.vector(solve(covar)%*%(mu-mktrf)
                   /as.numeric((t(one)%*%solve(covar)%*%(mu-mktrf))))
print("The market portfolio in the one-fund theorem is")
```

```
## [1] "The market portfolio in the one-fund theorem is"
```

```
cbind(company, mktport)
```

```
##       company mktport
##  [1,] "AAPL"  "0.118712203787641"
##  [2,] "ADBE"  "0.657120392888832"
##  [3,] "ADP"   "-0.403862485011893"
##  [4,] "AMD"   "-0.326087948142585"
##  [5,] "DELL"  "2.12748670229292"
##  [6,] "GTW"   "-1.50356801971028"
##  [7,] "HP"    "-0.0731814638930222"
##  [8,] "IBM"   "0.183449880027483"
##  [9,] "MSFT"  "0.179971332567008"
## [10,] "ORCL"  "0.0399594051938877"
```

Etimate of implied excess equilibrium return, i.e. equilibrium risk premium of the risky assets (pi)

We use the market portfolio weights calculated above to approximate the market capitalization weights. We consider the risk aversion parameter be 2.25. (normally the risk aversion parameter is between 2 and 4)

```r
risk_aversion_parameter <- 2.25
meanas = risk_aversion_parameter*covar%*%mktport
print("The equilibrium risk premium of the risky assets are")
```

```
## [1] "The equilibrium risk premium of the risky assets are"
```

```r
meanas
```

```
##                [,1]
##  [1,]  0.0035459373
##  [2,]  0.0045305568
##  [3,] -0.0001483448
##  [4,] -0.0021541911
##  [5,]  0.0097605070
##  [6,] -0.0081146980
##  [7,]  0.0012064392
##  [8,]  0.0020330606
##  [9,]  0.0040416092
## [10,]  0.0035954305
```

The covariance matrix is estimated by the covariance of historical data

```r
covar<-cov(dat)
print("The covariance matrix is")
```

```
## [1] "The covariance matrix is"
```

```r
covar
```

```
##              AAPL         ADBE          ADP          AMD         DELL
## AAPL 0.0045492771 0.0013279090 0.0001289013 0.0026123284 0.0019279099
## ADBE 0.0013279090 0.0044543592 0.0005155629 0.0017548782 0.0011450512
## ADP  0.0001289013 0.0005155629 0.0007882798 0.0005010847 0.0003765201
## AMD  0.0026123284 0.0017548782 0.0005010847 0.0071438193 0.0019955112
## DELL 0.0019279099 0.0011450512 0.0003765201 0.0019955112 0.0035220840
## GTW  0.0022417112 0.0019385825 0.0005769367 0.0030814814 0.0025456103
## HP   0.0007299668 0.0006935191 0.0002949267 0.0009015779 0.0005459061
## IBM  0.0009500523 0.0004466988 0.0004151517 0.0015804227 0.0010751672
## MSFT 0.0008986926 0.0006104442 0.0003242627 0.0011862503 0.0016045961
## ORCL 0.0013409915 0.0011763324 0.0003809339 0.0015399549 0.0011940995
##               GTW           HP          IBM         MSFT         ORCL
## AAPL 0.0022417112 7.299668e-04 0.0009500523 8.986926e-04 0.0013409915
## ADBE 0.0019385825 6.935191e-04 0.0004466988 6.104442e-04 0.0011763324
## ADP  0.0005769367 2.949267e-04 0.0004151517 3.242627e-04 0.0003809339
## AMD  0.0030814814 9.015779e-04 0.0015804227 1.186250e-03 0.0015399549
## DELL 0.0025456103 5.459061e-04 0.0010751672 1.604596e-03 0.0011940995
## GTW  0.0065056598 4.673048e-04 0.0010237907 1.440775e-03 0.0011096197
## HP   0.0004673048 2.304971e-03 0.0004827528 8.859382e-05 0.0003080524
## IBM  0.0010237907 4.827528e-04 0.0014830329 9.009031e-04 0.0008346107
## MSFT 0.0014407745 8.859382e-05 0.0009009031 2.018664e-03 0.0009180812
## ORCL 0.0011096197 3.080524e-04 0.0008346107 9.180812e-04 0.0038527785
```

Construct P, Q, Omega

The view vector (Q) is 3x1 column vector. P is a 3x10 matrix. Omega is a 3x3 matrix.

The third row of P represents View 3, the absolute view. View 3 involves 4 assets: Apple, Microsoft, Oracle, and IBM. Sequentially, they are the 1st, 8th, 9th and 10th asset in the date, which corresponds with the "-1"s in the 1st, 8th, 9th and 10th column of Row 3 of P. This is "-1" since for each company cost would increase by 3% by change of tax policy, and we guess that this may lead to decrease in returns by 3%. So Q[3]=3.

View 1 and View 2 are represented by Row 1 and Row 2, respectively. In the case of relative views, each row sums to 0. The nominally outperforming assets receive positive weightings, while the nominally underperforming assets receive negative weightings.

According to Idzorek(2004), "Litterman (2003, p. 82) assigns a percentage value to the asset(s) in question. Satchell and Scowcroft (2000) use an equal weighting scheme. Under this system, the weightings are proportional to 1 divided by the number of respective assets outperforming or underperforming."

View 1 says that a fund consisting of HP, AAPL, MSFT, and IBM in equal weights would increase its return by 0.1%. Since four assets norminally outperform, each receiving a +0.25 weighting, and Q[1]=0.1.

View 2 says that 2*r_{IBM}-r_{GTW}=0.50%, IBM outperfoms and GTM underperforms. So the weight for IBM is 2, and for GTM is -1. Q2=0.5%.

Omega is a diagonal matrix, where the diagonal entries are the variance of the error terms. $var = sd^2 = (se * \sqrt{n})^2$, where n=156 the length of the data.

```
# P: investors' individual picks of their assets
P <- matrix( c(0.25,0,0,0,0,0,0.25,0.25,0.25,0,
               0,0,0,0,0,-1,0,2,0,0,
               -1,0,0,0,0,0,0,-1,-1,-1),3,10, byrow = TRUE)

# Omega: The variances of the individual view portfolios
# diagonal entries: variance of the error terms
Omega <- matrix(c((0.02*sqrt(156))^2,0,0,
                  0,(0.08*sqrt(156))^2,0,
                  0,0,(0.01*sqrt(156))^2),3,3)

# Q: view vector
Q <- c(0.1, 0.5, 3)
```

```
P
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7]  [,8]  [,9] [,10]
## [1,]  0.25    0    0    0    0    0 0.25  0.25  0.25     0
## [2,]  0.00    0    0    0    0   -1 0.00  2.00  0.00     0
## [3,] -1.00    0    0    0    0    0 0.00 -1.00 -1.00    -1
```

```
Q
```

```
## [1] 0.1 0.5 3.0
```

```
Omega
```

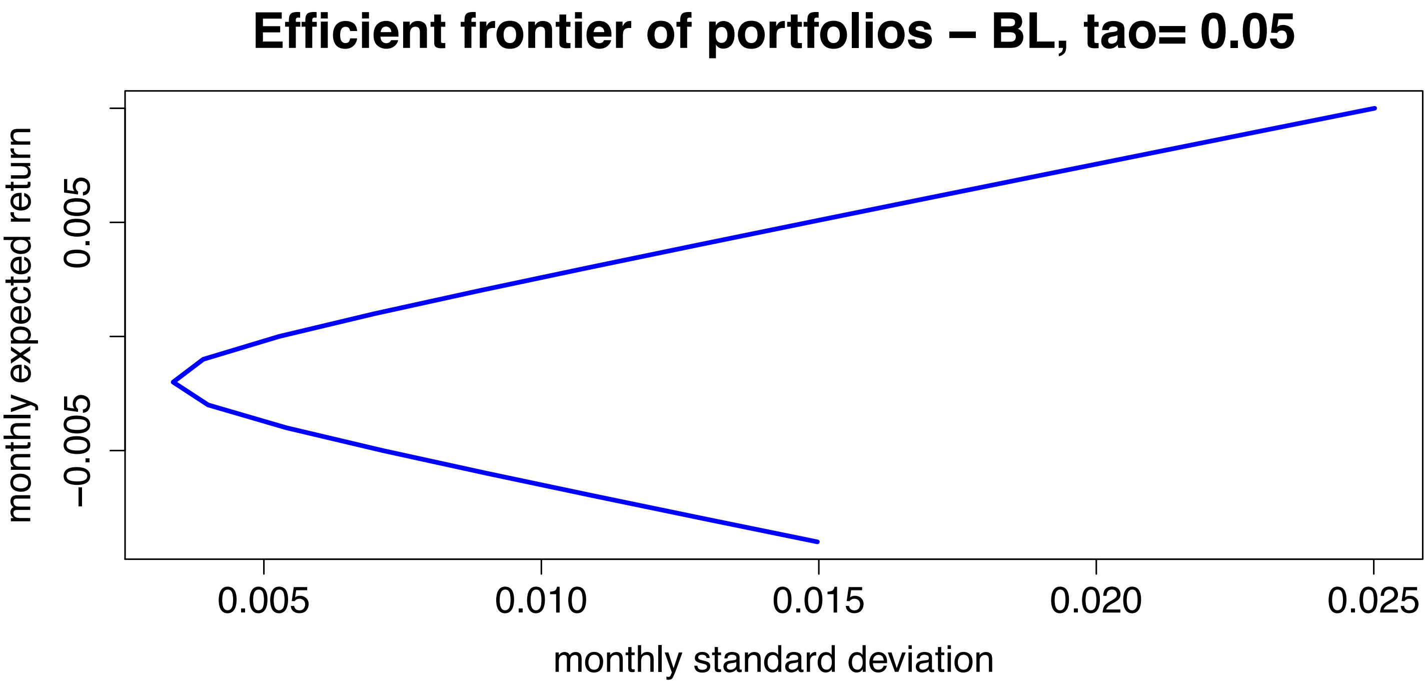
```
##        [,1]   [,2]   [,3]
## [1,] 0.0624 0.0000 0.0000
## [2,] 0.0000 0.9984 0.0000
## [3,] 0.0000 0.0000 0.0156
```

Function for Black-Litterman

```
BL<-function(tao){
  cat("tao=",tao,"\n\n")
  C <- covar

  Sigma_mu<-tao * C

  var1 <- ginv(Sigma_mu)  # ginv: create inverse of matrix
  var2 <- t(P) %*% ginv(Omega) %*% P
  var12 <- ginv(var1+var2)
  var3 <- (t(P) %*% ginv(Omega) %*% Q) + (var1 %*% meanas)
  mhat <- var12 %*% var3    # the mean of the expected return: mu proved in part(1)
  print("Calculated by the formula proved in part(1), the mean of the expected return is")
  print(cbind(company, mhat))
  cat("\n\n")

  varhat<-var12 # the variance of the expected return: sigma proved in part(1)
  print("Calculated by the formula proved in part(1), the sigma of the expected return is")
  print(cbind(company, varhat))
  cat("\n\n")

  BmuP=vector()
  BsigmaP=vector()
  w=matrix(0,nrow=10, ncol=20)

  for (i in 1:20){
   mustar=-0.1+i*0.01

   one<-(rep(1, dim(varhat)[1]))
   invCov<-solve(varhat)
   A<-as.vector(one%*%invCov%*%mhat)
   B<-as.vector(t(mhat)%*%invCov%*%mhat)
   C<-as.vector(one%*%invCov%*%one)
   D<-B*C-A*A
   w[,i]<-(B*as.vector(invCov%*%one)-A*as.vector(invCov%*%mhat)+mustar*(C*as.vector(invCov%*%mhat)
                                                                     -A*as.vector(invCov%*%one)))/D
   BmuP[i]=mean(w[,i]*mhat)
   BsigmaP[i]=sqrt(var(w[,i]*mhat))
  }


  #Efficient frontier Plot - BL
  BLplot <- plot(BsigmaP, BmuP,type = 'l',lty = 1,lwd=3,
                 xlab = 'monthly standard deviation',ylab = 'monthly expected return',
                 main = paste('Efficient frontier of portfolios - BL, tao=',tao),col = 'blue',
                 cex.lab=1.5, cex.main=2, cex.axis=1.5, cex.sub=1.5)
}
```

```
BL(0.05)  # tao=0.05
```

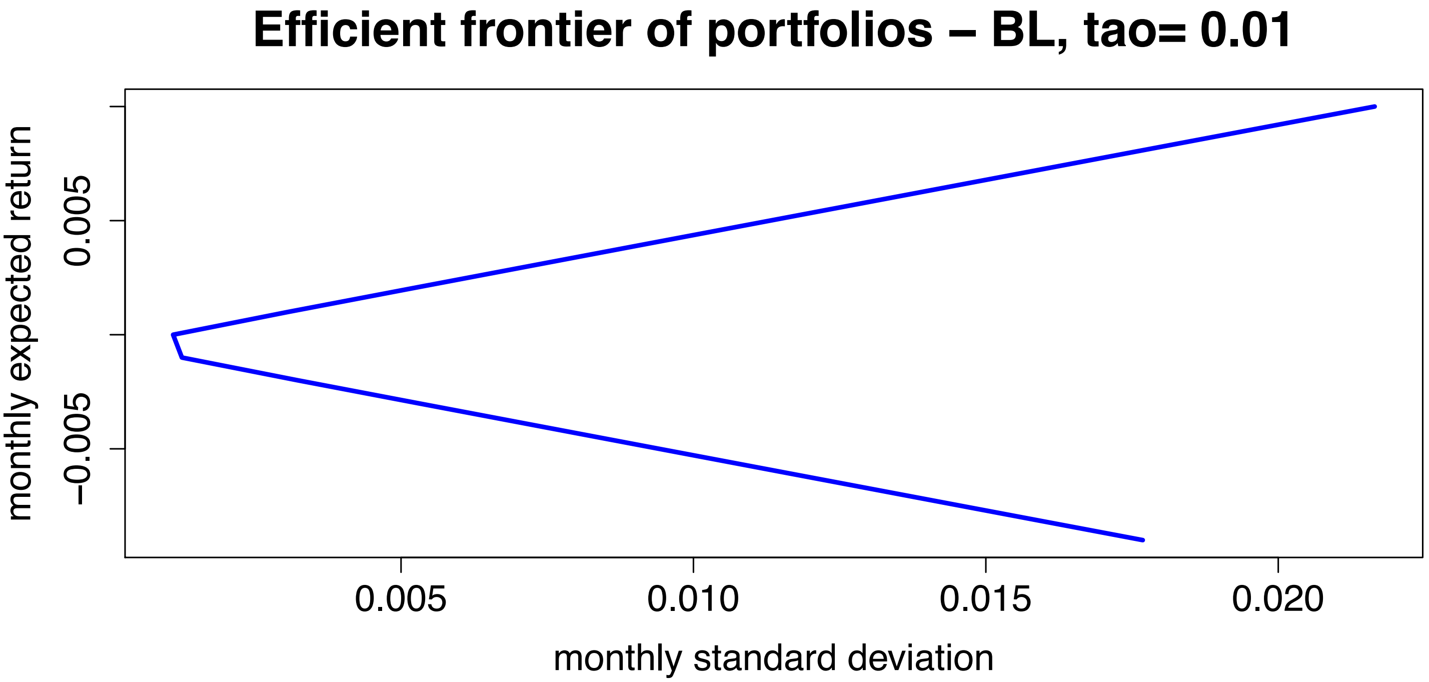
```
## tao= 0.05
##
## [1] "Calculated by the formula proved in part(1), the mean of the expected return is"
##        company
##  [1,] "AAPL"  "-0.0657661305005849"
##  [2,] "ADBE"  "-0.0273941421882248"
##  [3,] "ADP"   "-0.0113286284249031"
##  [4,] "AMD"   "-0.0641143897434025"
##  [5,] "DELL"  "-0.0422100834641445"
##  [6,] "GTW"   "-0.0603153756598774"
##  [7,] "HP"    "-0.0131340297966591"
##  [8,] "IBM"   "-0.0352480033874087"
##  [9,] "MSFT"  "-0.0383759963815814"
## [10,] "ORCL"  "-0.0586791724984168"
##
##
## [1] "Calculated by the formula proved in part(1), the sigma of the expected return is"
##       company
##  [1,] "AAPL"  "0.000218427481916572" "6.2239532632313e-05"
##  [2,] "ADBE"  "6.22395326323137e-05" "0.000220804537116594"
##  [3,] "ADP"   "4.98648988811562e-06" "2.51080255666682e-05"
##  [4,] "AMD"   "0.000122539511369869" "8.40300676546928e-05"
##  [5,] "DELL"  "8.96241735950419e-05" "5.4137958814523e-05"
##  [6,] "GTW"   "0.000105293555497983" "9.37959858845949e-05"
##  [7,] "HP"    "3.45841116850272e-05" "3.37979093194664e-05"
##  [8,] "IBM"   "4.26377127595543e-05" "2.01022350373533e-05"
##  [9,] "MSFT"  "3.94125089480361e-05" "2.79835729654117e-05"
## [10,] "ORCL"  "5.89897610136153e-05" "5.51102612559639e-05"
##
##  [1,] "4.98648988811557e-06" "0.000122539511369869" "8.96241735950415e-05"
##  [2,] "2.51080255666684e-05" "8.40300676546943e-05" "5.41379588145233e-05"
##  [3,] "3.91783199898675e-05" "2.37502361584036e-05" "1.77331503653257e-05"
##  [4,] "2.37502361584033e-05" "0.000349971250530893" "9.37232907206716e-05"
##  [5,] "1.77331503653257e-05" "9.37232907206729e-05" "0.000171030034099055"
##  [6,] "2.77537773791054e-05" "0.000148007229919755" "0.00012218943528172"
##  [7,] "1.44368478384817e-05" "4.33678475016843e-05" "2.58620944763954e-05"
##  [8,] "1.99706913581643e-05" "7.46707226255877e-05" "5.01136163953968e-05"
##  [9,] "1.5321377168875e-05"  "5.43760690863067e-05" "7.6091838630761e-05"
## [10,] "1.77451722878199e-05" "6.97921912720835e-05" "5.36643230603446e-05"
##
##  [1,] "0.000105293555497981" "3.45841116850274e-05" "4.26377127595544e-05"
##  [2,] "9.37959858845944e-05" "3.37979093194668e-05" "2.01022350373536e-05"
##  [3,] "2.77537773791051e-05" "1.44368478384817e-05" "1.99706913581644e-05"
##  [4,] "0.000148007229919751" "4.33678475016843e-05" "7.46707226255875e-05"
##  [5,] "0.000122189435281719" "2.58620944763958e-05" "5.01136163953971e-05"
##  [6,] "0.000320133655615849" "2.19334132726709e-05" "4.75552782872714e-05"
##  [7,] "2.193341327267e-05"   "0.000114831082677618" "2.31041604074013e-05"
##  [8,] "4.75552782872704e-05" "2.31041604074014e-05" "7.1521213623219e-05"
##  [9,] "6.78939246808772e-05" "3.26148504489129e-06" "4.20691555233673e-05"
## [10,] "4.94307267009608e-05" "1.37097540239905e-05" "3.73866136501052e-05"
##
##  [1,] "3.94125089480358e-05" "5.89897610136155e-05"
##  [2,] "2.79835729654118e-05" "5.51102612559644e-05"
##  [3,] "1.53213771688749e-05" "1.77451722878199e-05"
##  [4,] "5.4376069086306e-05"  "6.97921912720833e-05"
##  [5,] "7.6091838630761e-05"  "5.36643230603452e-05"
##  [6,] "6.78939246808778e-05" "4.94307267009622e-05"
##  [7,] "3.26148504489112e-06" "1.37097540239904e-05"
##  [8,] "4.20691555233671e-05" "3.73866136501052e-05"
##  [9,] "9.75573243012207e-05" "4.09741115789165e-05"
## [10,] "4.09741115789162e-05" "0.000185426085742785"
```



```
BL(0.01) # tao=0.01
```

```
## tao= 0.01
##
## [1] "Calculated by the formula proved in part(1), the mean of the expected return is"
##       company
##  [1,] "AAPL"  "-0.0111515297634662"
##  [2,] "ADBE"  "-0.00223824620605981"
##  [3,] "ADP"   "-0.00251926849933499"
##  [4,] "AMD"   "-0.0152927478859061"
##  [5,] "DELL"  "-0.00125936318845288"
##  [6,] "GTW"   "-0.0191814018574585"
##  [7,] "HP"    "-0.00183814667036984"
##  [8,] "IBM"   "-0.00587329743935492"
##  [9,] "MSFT"  "-0.00495248500033975"
## [10,] "ORCL"  "-0.00960531605326458"
##
##
## [1] "Calculated by the formula proved in part(1), the sigma of the expected return is"
##       company
##  [1,] "AAPL"  "4.51095906659633e-05" "1.31028665582767e-05"
##  [2,] "ADBE"  "1.31028665582768e-05" "4.44624642107206e-05"
##  [3,] "ADP"   "1.22716114972444e-06" "5.12721154377139e-06"
##  [4,] "AMD"   "2.5780786106333e-05"  "1.7391300380568e-05"
##  [5,] "DELL"  "1.89919666587517e-05" "1.13184447233751e-05"
##  [6,] "GTW"   "2.2129130924374e-05"  "1.92530063051504e-05"
##  [7,] "HP"    "7.21850669893742e-06" "6.89795835230991e-06"
##  [8,] "IBM"   "9.29421597440799e-06" "4.37229605457519e-06"
##  [9,] "MSFT"  "8.75275766941284e-06" "5.99679059781529e-06"
## [10,] "ORCL"  "1.30681187300263e-05" "1.16061502771214e-05"
```

```
##
##  [1,] "1.22716114972441e-06" "2.57807861063332e-05" "1.89919666587516e-05"
##  [2,] "5.1272115437714e-06"  "1.73913003805682e-05" "1.13184447233751e-05"
##  [3,] "7.87280440733741e-06" "4.95555057977538e-06" "3.71885776723032e-06"
##  [4,] "4.95555057977532e-06" "7.11320416439822e-05" "1.96984673875944e-05"
##  [5,] "3.71885776723033e-06" "1.96984673875947e-05" "3.50056743723413e-05"
##  [6,] "5.72301207350325e-06" "3.05575679798388e-05" "2.5240244413041e-05"
##  [7,] "2.93614555208431e-06" "8.94323156253823e-06" "5.39829184618567e-06"
##  [8,] "4.11814964936531e-06" "1.56197408740702e-05" "1.05971060574706e-05"
##  [9,] "3.20481162431596e-06" "1.16531702171328e-05" "1.58704884087531e-05"
## [10,] "3.75414396283621e-06" "1.5093980624504e-05"  "1.16848291705723e-05"
##
##  [1,] "2.21291309243737e-05" "7.21850669893743e-06" "9.29421597440798e-06"
##  [2,] "1.92530063051503e-05" "6.89795835230996e-06" "4.37229605457522e-06"
##  [3,] "5.72301207350322e-06" "2.93614555208432e-06" "4.11814964936532e-06"
##  [4,] "3.05575679798381e-05" "8.94323156253821e-06" "1.56197408740702e-05"
##  [5,] "2.52402444130408e-05" "5.39829184618573e-06" "1.05971060574707e-05"
##  [6,] "6.48383933563126e-05" "4.61232990767161e-06" "1.00837498868304e-05"
##  [7,] "4.61232990767148e-06" "2.30320398634332e-05" "4.78371315935698e-06"
##  [8,] "1.00837498868303e-05" "4.783713159357e-06"   "1.47187984784196e-05"
##  [9,] "1.42319867218232e-05" "8.36402324787265e-07" "8.8828300852095e-06"
## [10,] "1.08396281947895e-05" "3.00870851175007e-06" "8.16189064584864e-06"
##
##  [1,] "8.75275766941275e-06" "1.30681187300263e-05"
##  [2,] "5.99679059781527e-06" "1.16061502771215e-05"
##  [3,] "3.20481162431596e-06" "3.75414396283624e-06"
##  [4,] "1.16531702171326e-05" "1.5093980624504e-05"
##  [5,] "1.58704884087531e-05" "1.16848291705724e-05"
##  [6,] "1.42319867218232e-05" "1.08396281947897e-05"
##  [7,] "8.36402324787224e-07" "3.00870851175006e-06"
##  [8,] "8.88283008520947e-06" "8.16189064584867e-06"
##  [9,] "2.00434852940282e-05" "8.97175056609741e-06"
## [10,] "8.97175056609733e-06" "3.82219415910193e-05"
```



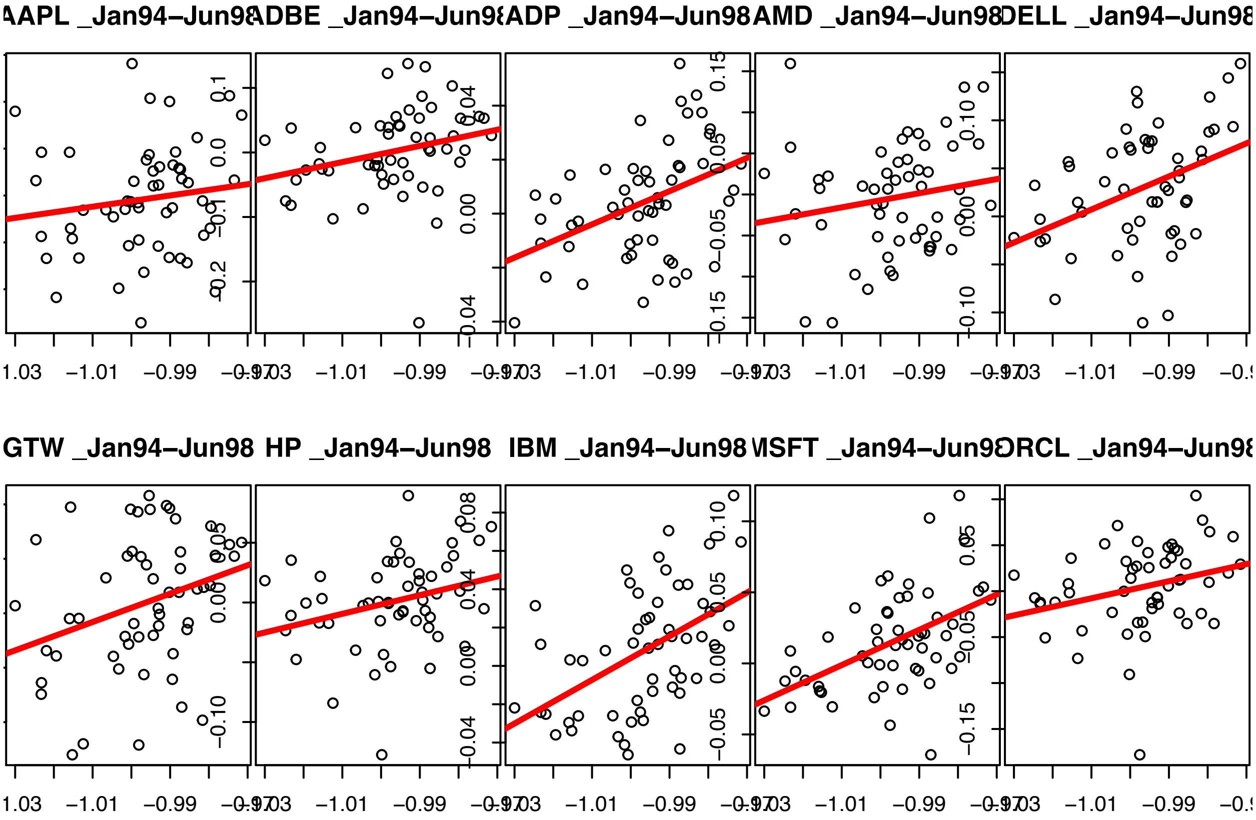
Efficient frontier of portfolios – BL, tao= 0.01

We can see from the result above, the efficient frontier for different value of τ is very diffrent. We know that τ is a scalar reflecting the uncertainty in the prior distribution of μ. $\mu = \pi + \epsilon$ and ϵ follows $N(0, \tau\Sigma)$. The bigger the τ is, the bigger the standard deviation of μ.

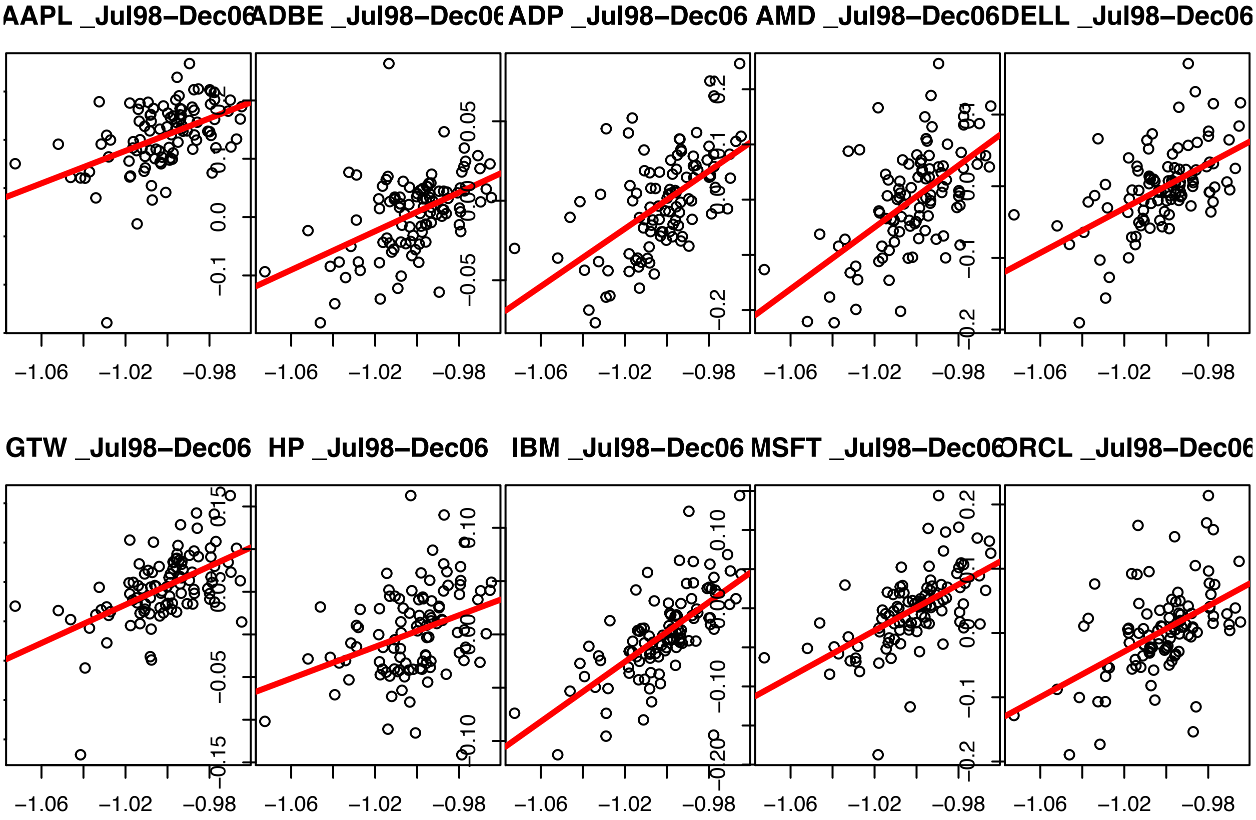
2

5.10 (a) Fit CAPM for the period (i) from Jan 1994 to June 1998 (ii) from Jul 1998 to Dec 2006 and estimate beta

```
#create sp500ret table from the data that contains the monthly returns of market (S&P500) index
#and the monthly rates of the 3-month US T-bill of Federal Reserve (risk-free)
sp500ret = read.table("https://web.stanford.edu/~xing/statfinbook/_BookData/Chap03/m_sp500ret_3mtcm.txt",
                   skip = 1,
                   sep="\t",
                   header = TRUE,
                   fill=FALSE,
                   strip.white=TRUE)
#create stock table from data that contains monthly rate of ten stocks from Jan 1994 to Dec 2016
stock = read.table("https://web.stanford.edu/~xing/statfinbook/_BookData/Chap03/m_logret_10stocks.txt",
               sep="\t",
               header = TRUE,
               fill=FALSE,
               strip.white=TRUE)
name <- colnames(stock[,2:11])

par(mar = c(3, 0.1, 3, 0.1))
par(mfrow=c(2,5))

stocks <- stock[1:156,-1]
rf <- sp500ret[,5] / 1200
stocks_ex <- apply(stocks,2,function(x){x - rf})
sp_ex <- sp500ret[,3]-rf
capm_before <- lm((stocks_ex[1:54,]) ~sp_ex[1:54]-1)
for (i in (1:10)){
  Y = (stocks_ex[1:54,i])
  X = sp_ex[1:54]-1
  title = paste(name[i],"_Jan94-Jun98")
  plot(Y~X,main=title)
  abline(lm(Y~X),col=2,lwd=3)
}
```



```
beta_before = coef(capm_before)[1,]

capm_after <- lm((stocks_ex[55:156,]) ~sp_ex[55:156]-1)
for (i in (1:10)){
  Y = (stocks_ex[55:156,i])
  X = sp_ex[55:156]-1
  title = paste(name[i],"_Jul98-Dec06")
  plot(Y~X,main=title)
  abline(lm(Y~X),col=2,lwd=3)
}
```



```r
beta_after = coef(capm_after)[1,]
cbind(beta_before,beta_after)
```

```
##      beta_before beta_after
## AAPL   0.5112057  1.6311096
## ADBE   1.2196229  1.6176307
## ADP    0.6477905  0.9041194
## AMD    0.7560834  2.7700785
## DELL   2.0812867  1.5548681
## GTW    1.3775690  2.4788492
## HP     0.7575166  0.9077438
## IBM    1.1994294  1.3888152
## MSFT   1.4070321  1.4710070
## ORCL   0.9537838  1.7401961
```

The estimated betas significantly differ for the two specified periods. In practice, beta is the measure of the volatility, or systematic risk, of an individual stock in comparison to the unsystematic risk of the entire market. This results show that the systematic risks for the two periods respectively differ a lot. Statistically speaking, beta represents the slope of the line through a regression of data points from a stock's returns against those of the market. This can be seen from the plots illustrated above.

5.10 (b) Consider dynamic linear model 5.40 for CAPM with time-varying betas. Use Kalman filter with $sigma_w = 0.2$ to estimate $beta_t$ sequentially

```r
library(dlm)
residuals<-capm_before$residuals
```

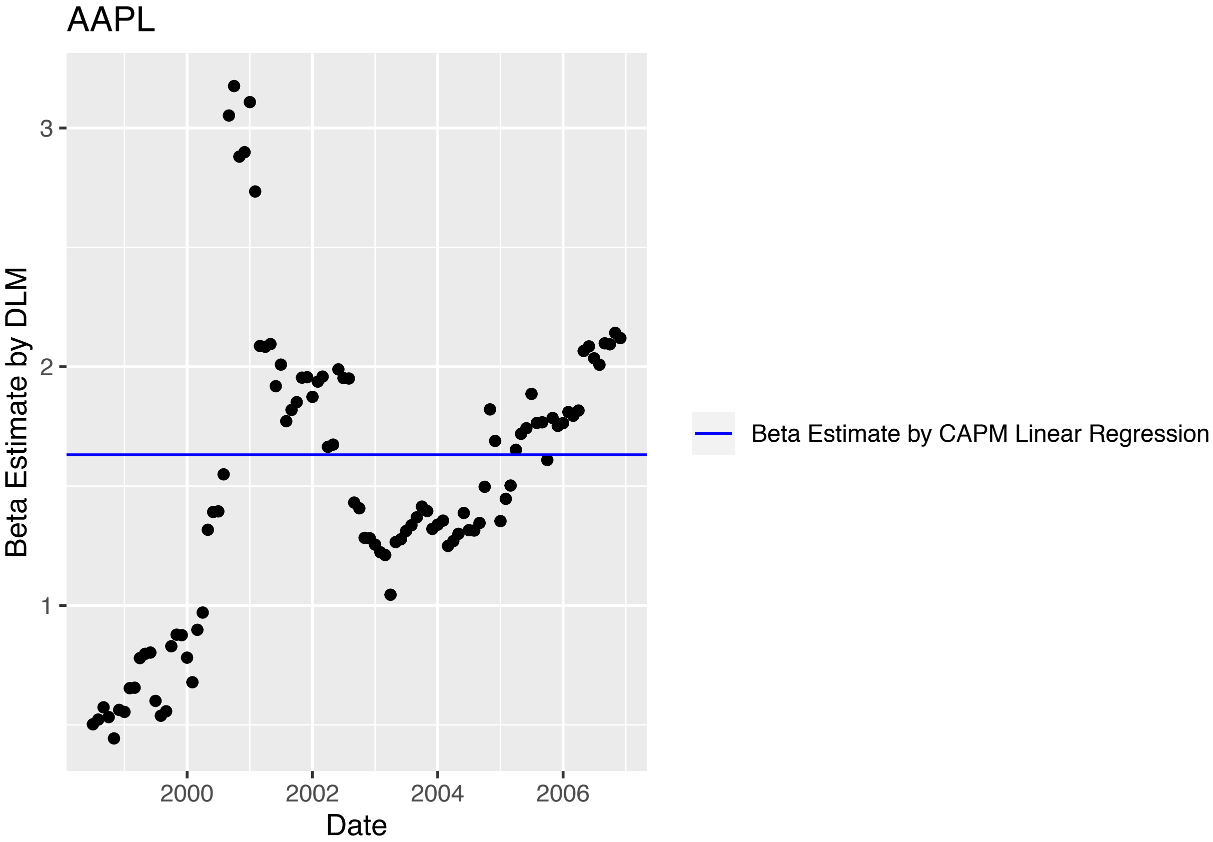
```
out_list <- vector(mode = "list", length = 10)
for(i in 1:10){
  error_variance <- var(residuals[,i])
  dlm = dlmModReg(matrix(sp_ex[55:156]),dV = error_variance, dW=0.04, m0=beta_before[i],C0=0*diag(nrow=1),
                addInt=FALSE)
  out = dlmFilter(as.numeric(stocks_ex[55:156,i]),dlm)
  ele <- c(out$m[2:103]) #strip out the first item since it's the base
  out_list[[i]] = ele
}
```

5.10 (c) Compare and discuss estimated beta from CAPM Linear Regression (part a) and from DLM with Kelman Filter (part b)

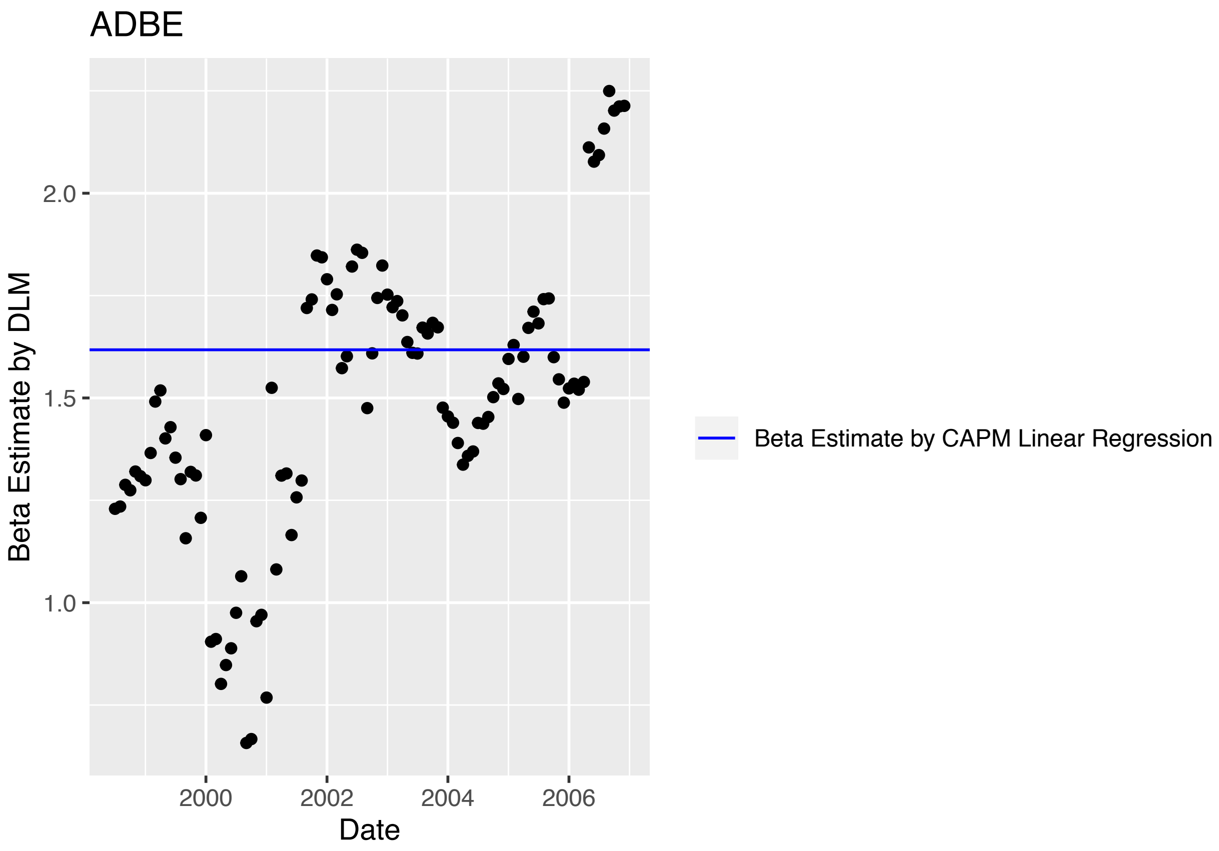
```
sp500_post = sp500ret[c(55:156),]
library(flipTime)
sp500_post$newdate <- AsDate(sp500_post$Date)

library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:dlm':
##
##     %+%
```

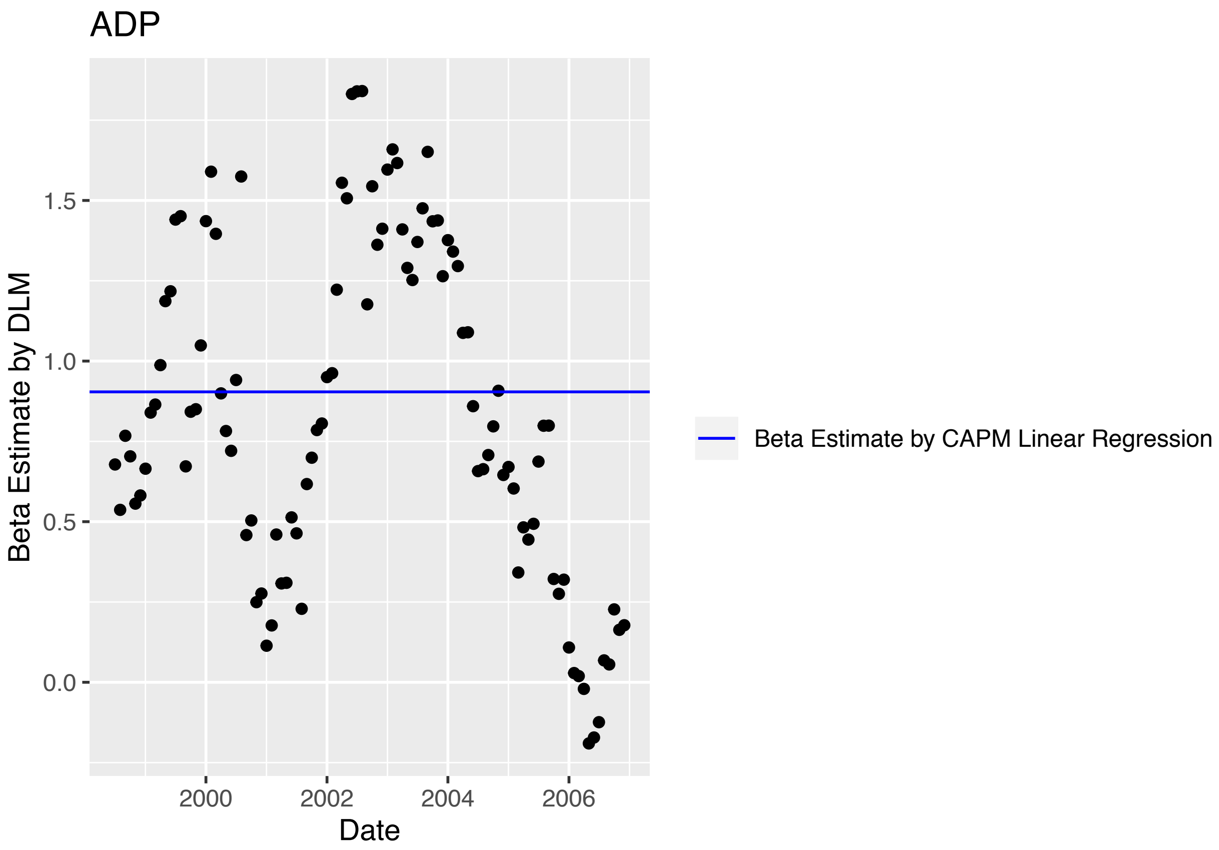
```
base <- ggplot(data = sp500_post, mapping = aes(x = newdate, y = out_list[[1]])) + geom_point() + geom_hline(aes
scale_linetype_manual(name='',values = c(1, 1), guide = guide_legend(override.aes = list(color = c("blue"))))
base + labs(title="AAPL", x ="Date", y = "Beta Estimate by DLM")
```



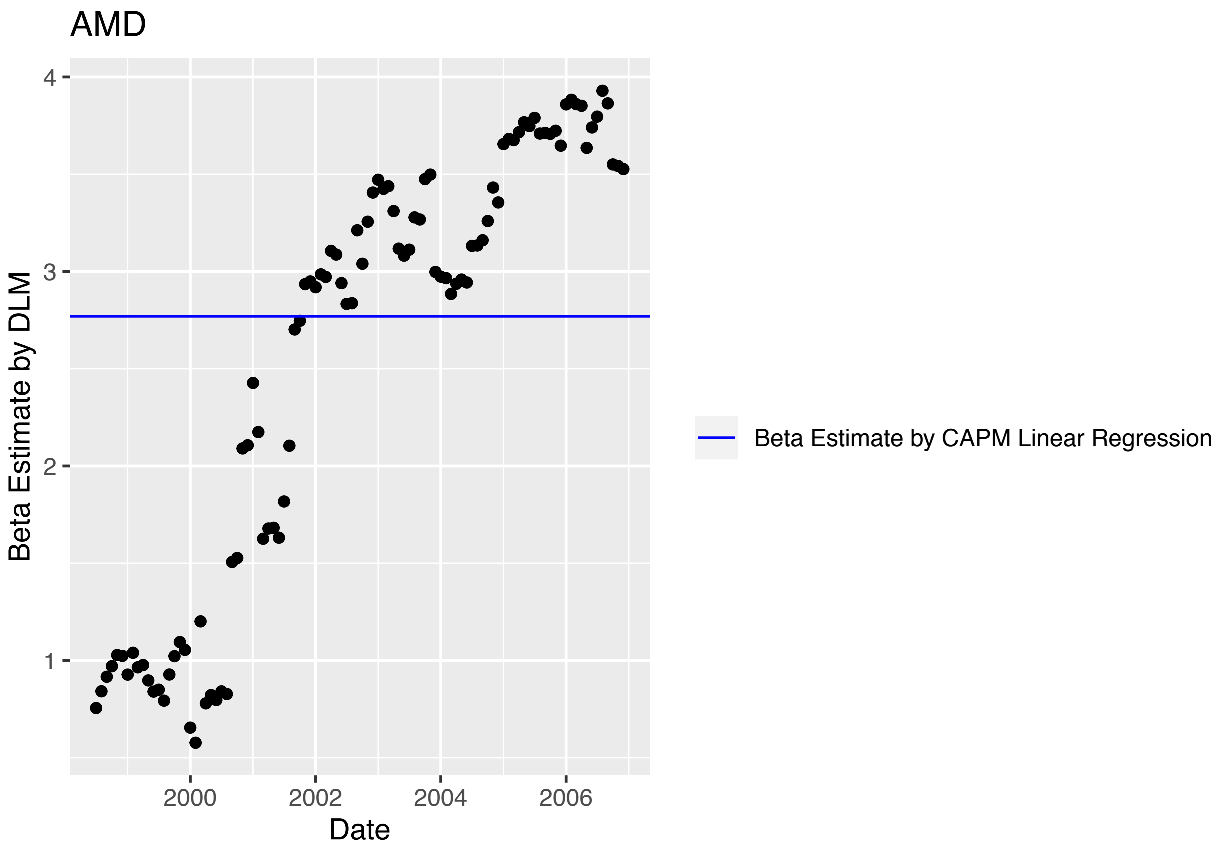
```
base <- ggplot(data = sp500_post, mapping = aes(x = newdate, y = out_list[[2]])) + geom_point() + geom_hline(aes
scale_linetype_manual(name='',values = c(1, 1), guide = guide_legend(override.aes = list(color = c("blue"))))
base + labs(title="ADBE", x ="Date", y = "Beta Estimate by DLM")
```



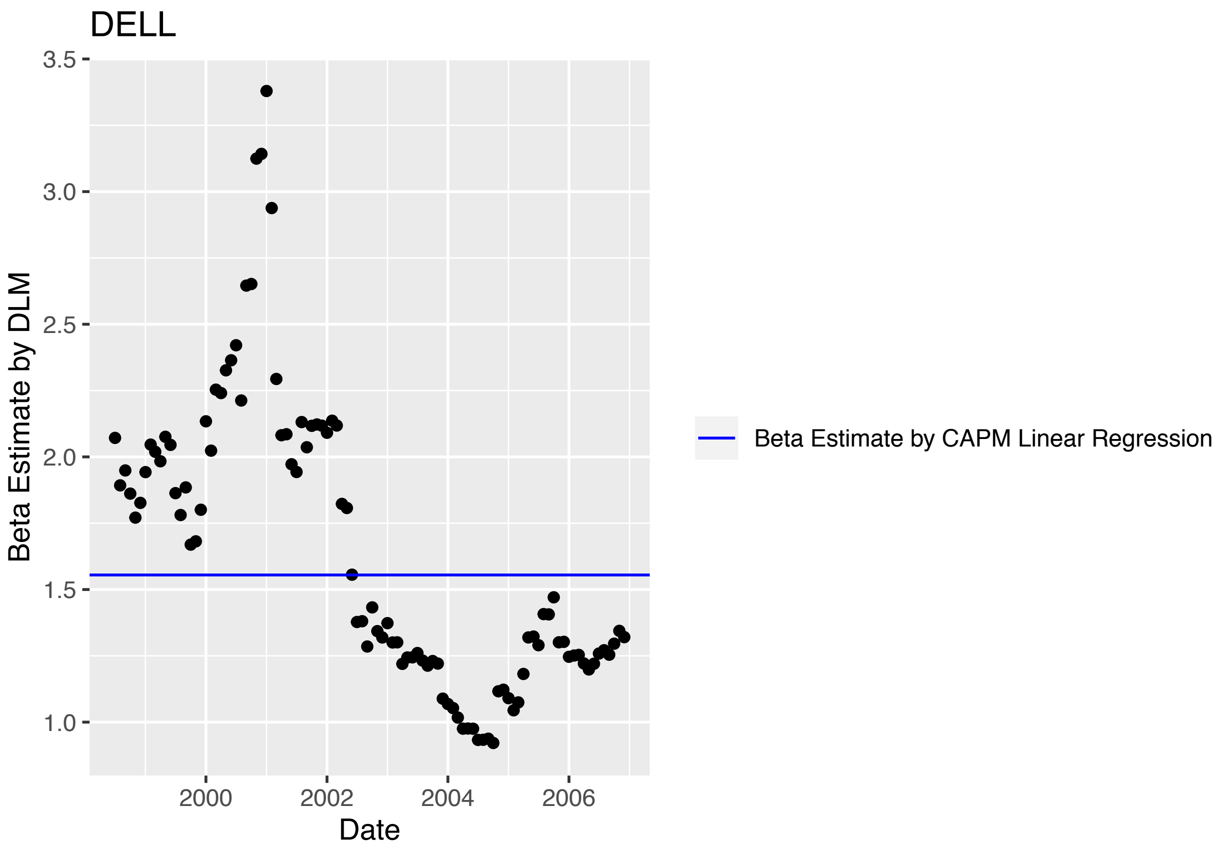
```
base <- ggplot(data = sp500_post, mapping = aes(x = newdate, y = out_list[[3]])) + geom_point() + geom_hline(aes
scale_linetype_manual(name='',values = c(1, 1), guide = guide_legend(override.aes = list(color = c("blue"))))
base + labs(title="ADP", x ="Date", y = "Beta Estimate by DLM")
```



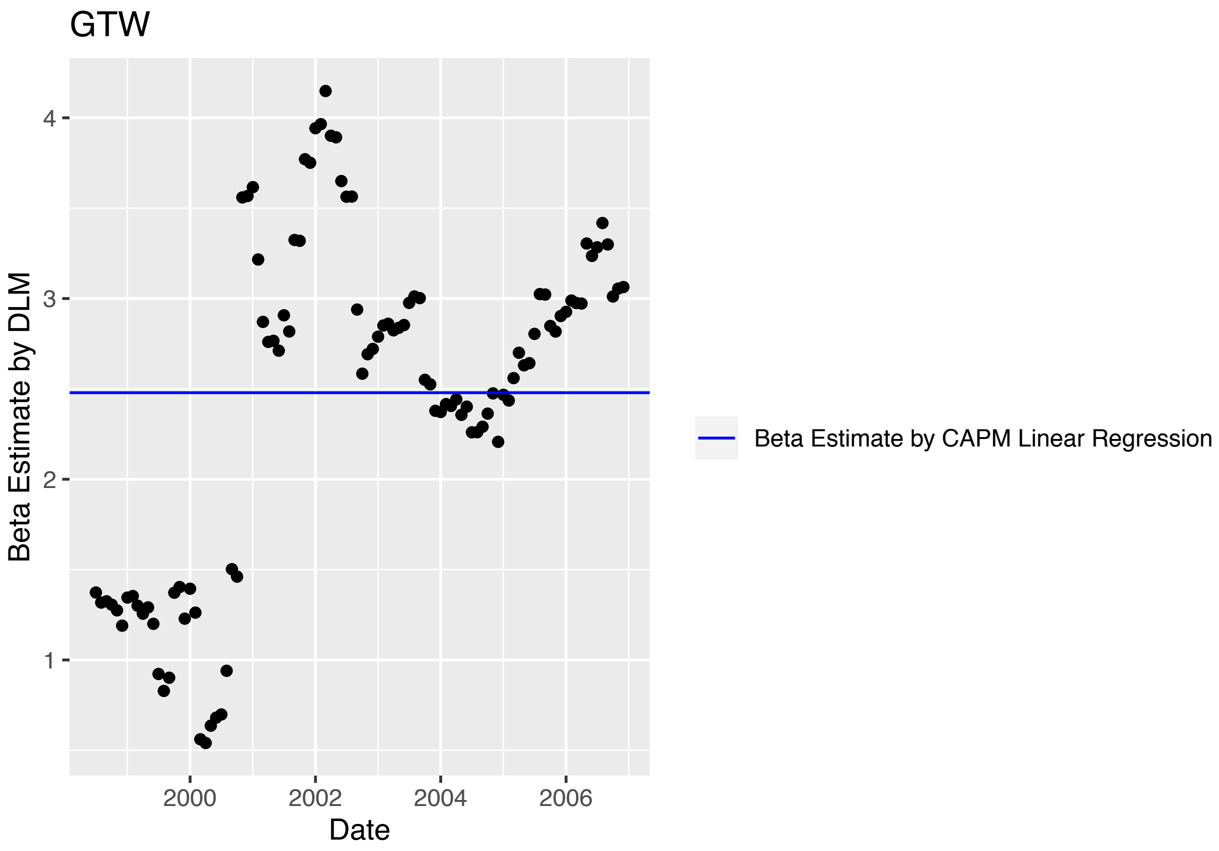
```
base <- ggplot(data = sp500_post, mapping = aes(x = newdate, y = out_list[[4]])) + geom_point() + geom_hline(aes
scale_linetype_manual(name='',values = c(1, 1), guide = guide_legend(override.aes = list(color = c("blue"))))
base + labs(title="AMD", x ="Date", y = "Beta Estimate by DLM")
```



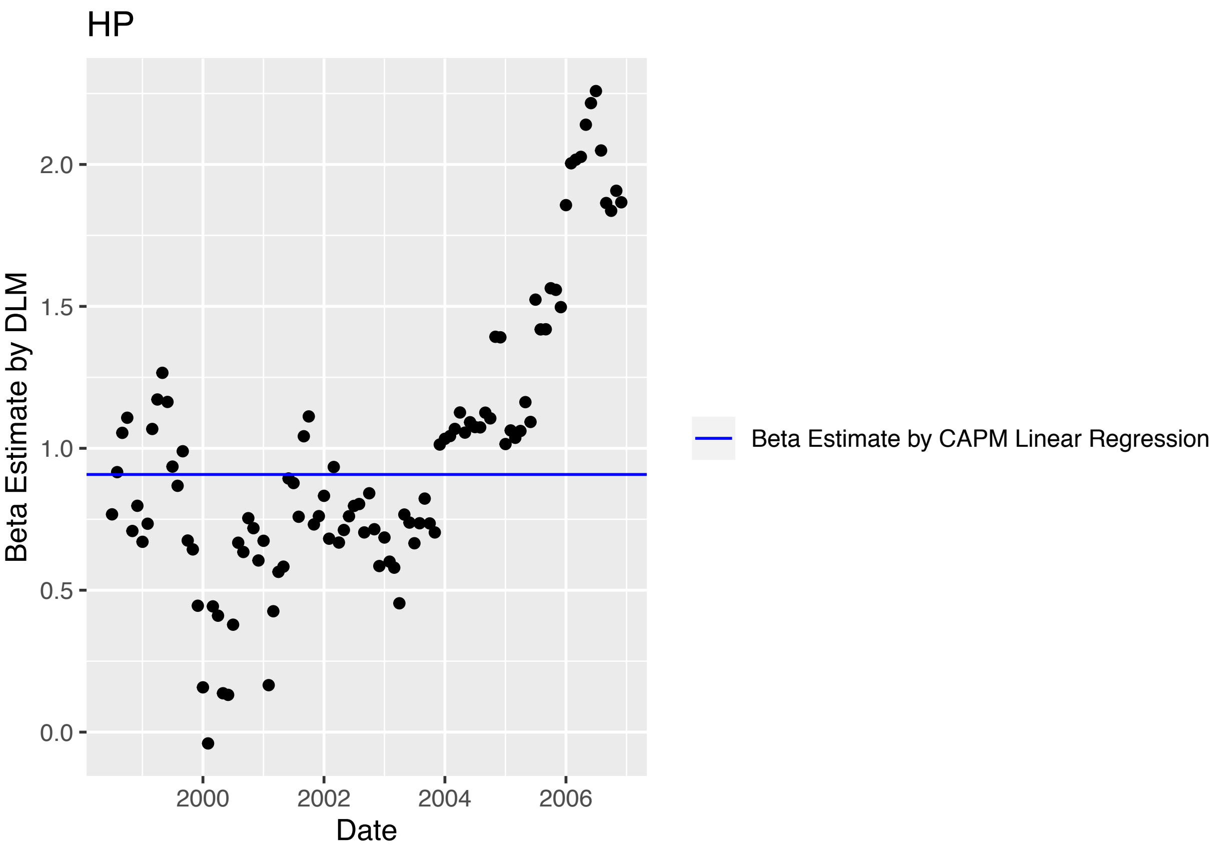
```
base <- ggplot(data = sp500_post, mapping = aes(x = newdate, y = out_list[[5]])) + geom_point() + geom_hline(aes
scale_linetype_manual(name='',values = c(1, 1), guide = guide_legend(override.aes = list(color = c("blue"))))
base + labs(title="DELL", x ="Date", y = "Beta Estimate by DLM")
```



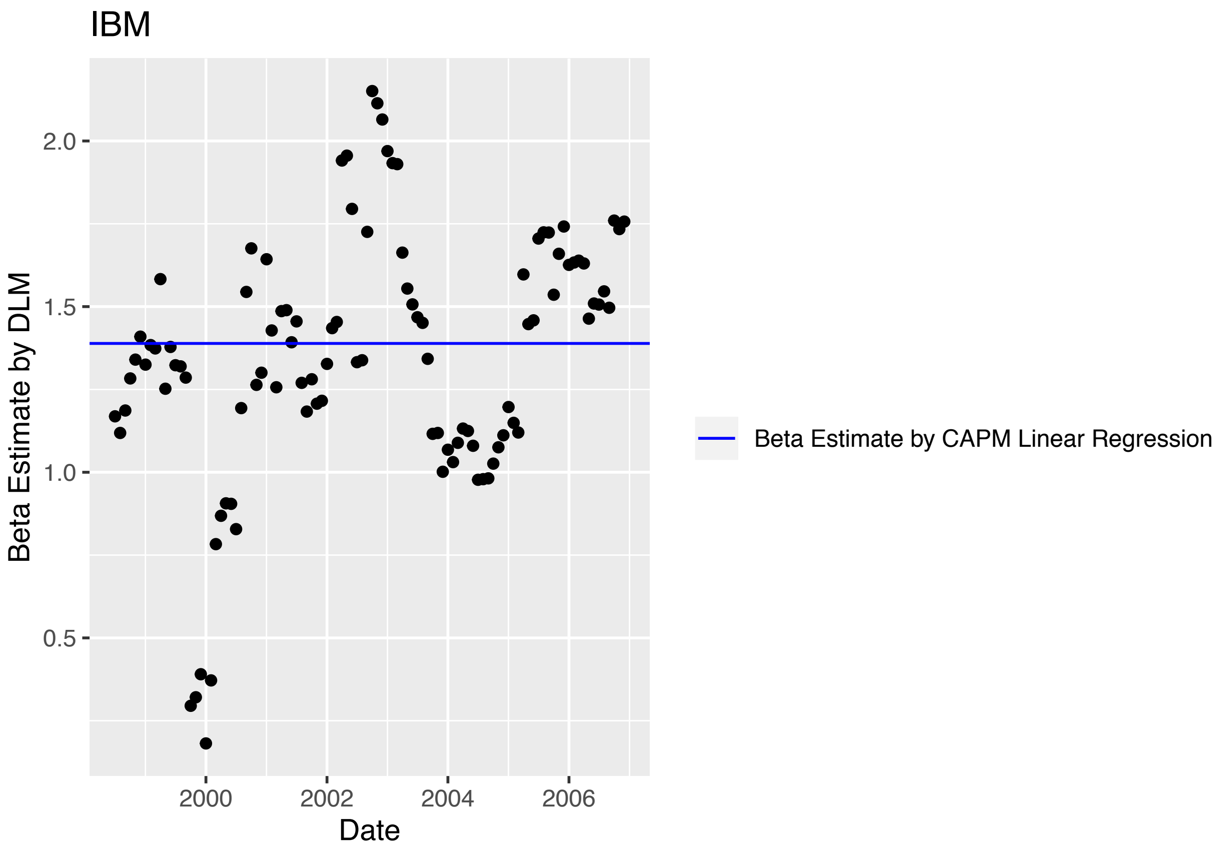
```
base <- ggplot(data = sp500_post, mapping = aes(x = newdate, y = out_list[[6]])) + geom_point() + geom_hline(aes
scale_linetype_manual(name='',values = c(1, 1), guide = guide_legend(override.aes = list(color = c("blue"))))
base + labs(title="GTW", x ="Date", y = "Beta Estimate by DLM")
```



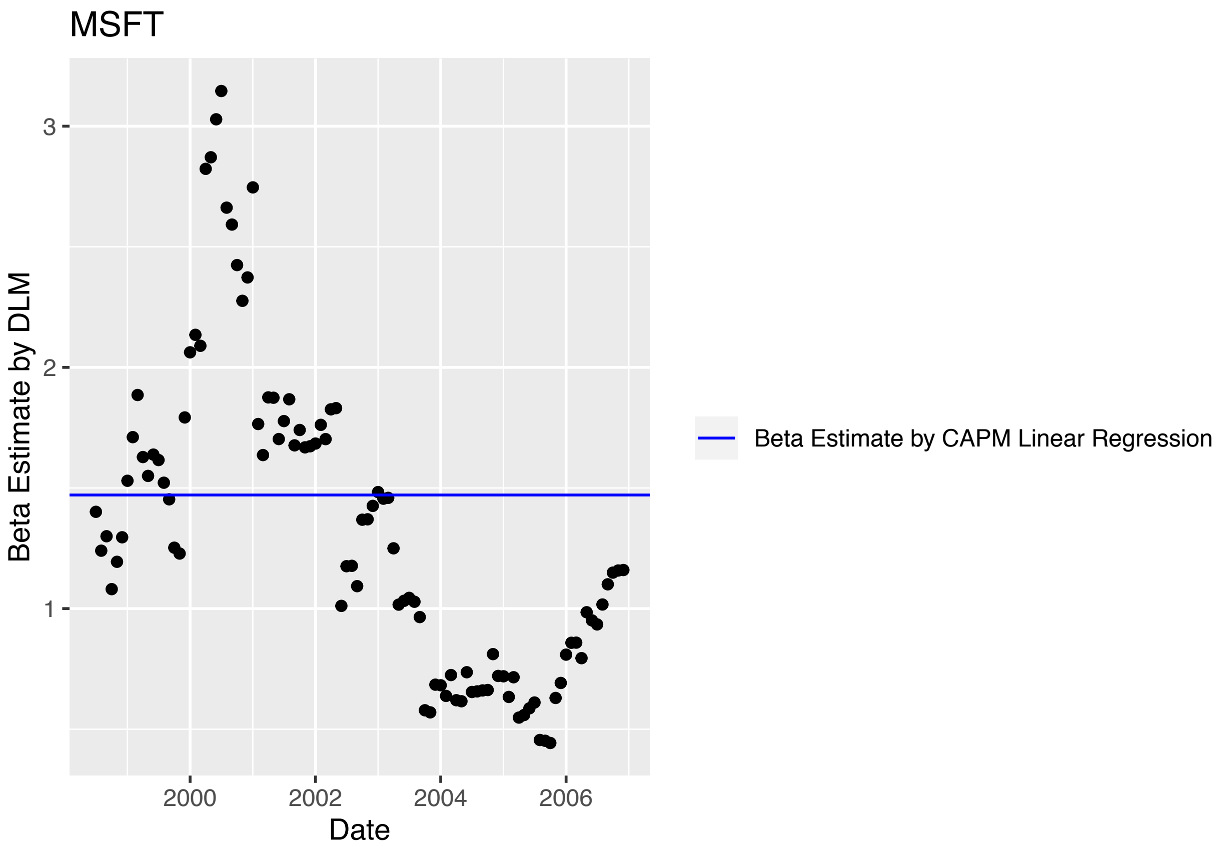
```
base <- ggplot(data = sp500_post, mapping = aes(x = newdate, y = out_list[[7]])) + geom_point() + geom_hline(aes
scale_linetype_manual(name='',values = c(1, 1), guide = guide_legend(override.aes = list(color = c("blue"))))
base + labs(title="HP", x ="Date", y = "Beta Estimate by DLM")
```



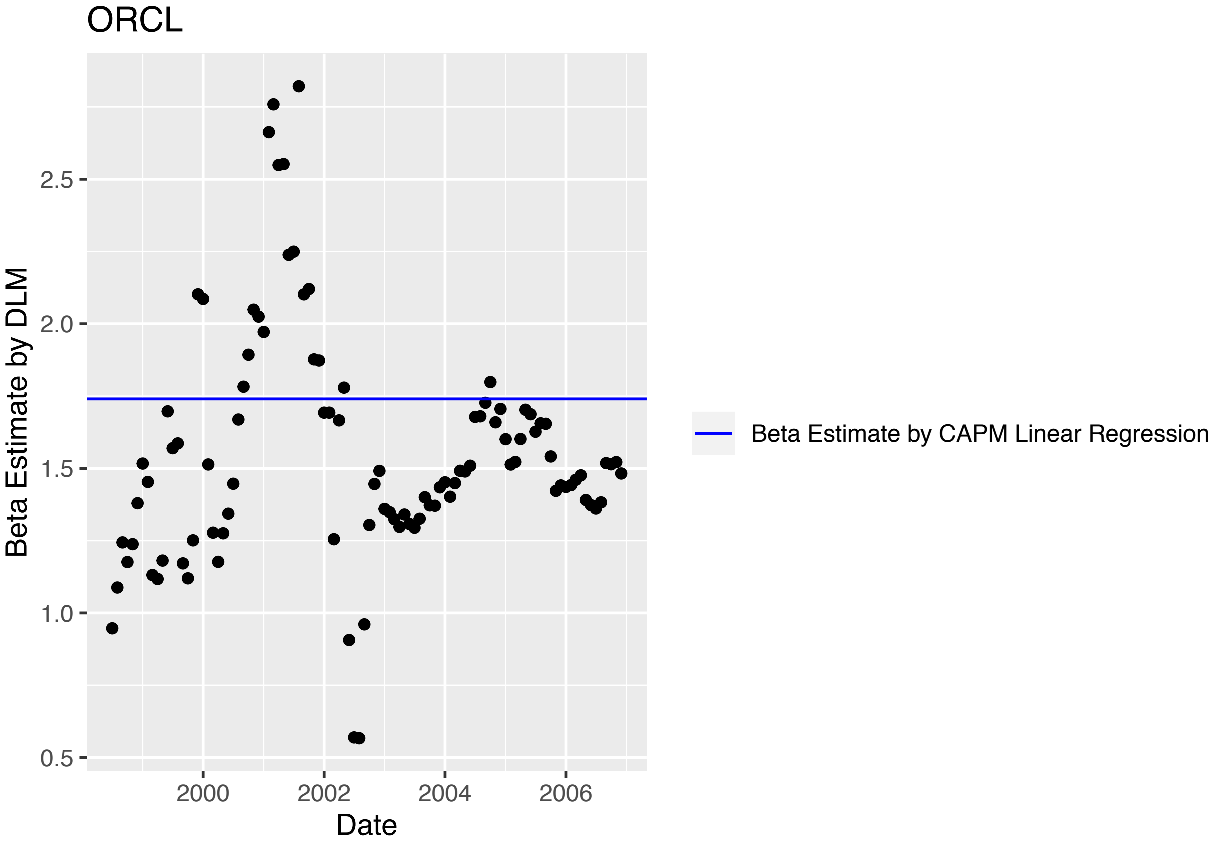
```
base <- ggplot(data = sp500_post, mapping = aes(x = newdate, y = out_list[[8]])) + geom_point() + geom_hline(aes
scale_linetype_manual(name='',values = c(1, 1), guide = guide_legend(override.aes = list(color = c("blue"))))
base + labs(title="IBM", x ="Date", y = "Beta Estimate by DLM")
```



```
base <- ggplot(data = sp500_post, mapping = aes(x = newdate, y = out_list[[9]])) + geom_point() + geom_hline(aes
scale_linetype_manual(name='',values = c(1, 1), guide = guide_legend(override.aes = list(color = c("blue"))))
base + labs(title="MSFT", x ="Date", y = "Beta Estimate by DLM")
```



```
base <- ggplot(data = sp500_post, mapping = aes(x = newdate, y = out_list[[10]])) + geom_point() + geom_hline(ae
scale_linetype_manual(name='',values = c(1, 1), guide = guide_legend(override.aes = list(color = c("blue"))))
base + labs(title="ORCL", x ="Date", y = "Beta Estimate by DLM")
```



In the 10 figures above (for 10 stocks), each black dot represents the beta estimate by Dynamic Linear Model (DLM) across time; each blue horizontal line represents the beta estimate by Simple Linear Regression for period July 1998 to December 2006. Although seemingly there's no apparent pattern across time, we can see that the DLM estimates for each stock fluctuate near the linear model estimate (blue line) evenly without concentrating too much above or below the horizontal line. Kalman filtering tends to produce estimates of unknown variables by allowing for variability in the regression coefficients and estimating a joint probability distribution over the variables for each timeframe. By the end of year 2002, the Dot-com bubble occurred and this fact was generally reflected in the DLM Estimates (except for stock AMD). Hence, DLM captures the information of the stock performance better than Linear Regression. From the figures and analyses above, we could observe and conclude that both regression approaches for beta estimates align nicely with each other.

Reference:

Q1.1

http://www.globalriskguard.com/resources/assetman/bayes_0008.pdf

Q1.2

https://faculty.fuqua.duke.edu/~charvey/Teaching/BA491_2005/MG/MG.ppt

https://faculty.fuqua.duke.edu/~charvey/Teaching/IntesaBci_2001/GS_The_intuition_behind.pdf

https://faculty.fuqua.duke.edu/~charvey/Teaching/BA453_2006/Idzorek_onBL.pdf

https://rpubs.com/RajeshVegi/Portfolio_optimization

https://pdfs.semanticscholar.org/75b1/45e5fefffcf3c190f676784861b9e65c0089.pdf

https://rpubs.com/saeed349/337816