Homework 5

Lydia Strebe

May 1, 2019

Problem 2: Optimal Portfolio Models

Optimal portfolio with simple covariance matrix:

```
library(boot)
library(gurobi)

## Loading required package: slam

## Warning: package 'slam' was built under R version 3.5.2

library(leaps)

## Warning: package 'leaps' was built under R version 3.5.2

library(glmnet)

## Warning: package 'glmnet' was built under R version 3.5.2

## Loading required package: Matrix

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 3.5.2

## Loaded glmnet 2.0-16

finance_data = read.csv('C:/Users/lydia/Documents/finance.csv')

#function for bootstrap (2 arguments: data set and index-which rows of data
set)
#returns opt weights and sdev
get_opt_sdev_and_x_column_wise = function(input_data,index=NULL){
  #if no index is given, use entire data set
  if(is.null(index)){
    index = 1:dim(input_data)[1]
  }


  returns_data = input_data[index,2:30]
  factor_data = input_data[index,31:42]
  #how many stocks and how many days (columns and rows)
  nstocks = dim(returns_data)[2]
  ndays = dim(returns_data)[1]
```

```
  cov_mat = cov(returns_data)

  #stuff to feed into gurobi
  port = list()
  port$modelsense = 'min'
  port$Q = cov_mat
  #contraint sum of xs
  port$A = array(1,c(1,nstocks))
  #constraint - must be equal to 1
  port$rhs = 1
  port$sense = '='

  #tell gurobi to shut up
  params = list()
  params$outputflag = 0;
  #plug into gurobi
  port_results = gurobi(port,params=params)

  output_variable = list()
  output_variable$sdev = sqrt(port_results$objval)
  output_variable$x = port_results$x

  output_variable
}

#this function calls 1st function and returns opt st dev
get_opt_sdev_column_wise = function(input_data,index=NULL){
  if(is.null(index)){
    index = 1:dim(input_data)[1]
  }

  val = get_opt_sdev_and_x_column_wise(input_data,index)

  val$sdev
}

boot_opt_sdev =
boot(finance_data[1:442,],get_opt_sdev_column_wise,R=500,parallel="multicore"
)


opt_port = get_opt_sdev_and_x_column_wise(finance_data[1:442,])
opt_port_sd=opt_port$sdev
opt_port = opt_port$x

#out of sample returns
ret_validate = array(0,100)
for(day in 443:542){      #weights times returns
```

```
  ret_validate[day-442] = sum(opt_port*finance_data[day,2:30])
}

quantile(boot_opt_sdev$t[,1], probs=c(.025,.975))

##        2.5%       97.5%
## 0.004557625 0.006330212

#standard deviation of returns
sd(ret_validate)

## [1] 0.01039603

#standard deviation from optimal model
opt_port_sd

## [1] 0.005537111
```

Optimal portfolio with ordinary least squares covariance matrix:

```
#function for bootstrap (2 arguments: data set and index-which rows of data
set)
#returns opt weights and sdev
get_opt_sdev_and_x_column_wise = function(input_data,index=NULL){
  #if no index is given, use entire data set
  if(is.null(index)){
    index = 1:dim(input_data)[1]
  }


  returns_data = input_data[index,2:30]
  factor_data = input_data[index,31:42]
  #how many stocks and how many days (columns and rows)
  nstocks = dim(returns_data)[2]
  ndays = dim(returns_data)[1]

  covf=cov(factor_data)

  M=matrix(nrow=12,ncol=29)
  v=matrix(0,nrow=29,ncol=29)

  for(i in 2:30){
    stock=input_data[,i]

lin_mod=glm(stock~Mkt+SMB+HML+RMW+CMA+DGS10+VAW+VCR+VDE+VFH+VGT+VHT,data=inpu
t_data)
    M[,i-1]=lin_mod$coefficients[2:13]
    v[i-1,i-1]=var(lin_mod$residuals)
  }

  #Covariance for linear reg model
```

```
  covs=t(M)%*%covf%*%M+v

  #stuff to feed into gurobi
  port = list()
  port$modelsense = 'min'
  port$Q = covs
  #contraint sum of xs
  port$A = array(1,c(1,nstocks))
  #constraint - must be equal to 1
  port$rhs = 1
  port$sense = '='

  #tell gurobi to shut up
  params = list()
  params$outputflag = 0;
  #plug into gurobi
  port_results = gurobi(port,params=params)

  output_variable = list()
  output_variable$sdev = sqrt(port_results$objval)
  output_variable$x = port_results$x

  output_variable
}

#this function calls 1st function and returns opt st dev
get_opt_sdev_column_wise = function(input_data,index=NULL){
  if(is.null(index)){
    index = 1:dim(input_data)[1]
  }

  val = get_opt_sdev_and_x_column_wise(input_data,index)

  val$sdev
}

boot_opt_sdev_ols =
boot(finance_data[1:442,],get_opt_sdev_column_wise,R=500,parallel="multicore"
)


opt_port_ols = get_opt_sdev_and_x_column_wise(finance_data[1:442,])
opt_port_ols_sd = opt_port_ols$sdev
opt_port_ols = opt_port_ols$x

#out of sample returns
ret_validate_ols = array(0,100)
for(day in 443:542){      #weights times returns
  ret_validate_ols[day-442] = sum(opt_port_ols*finance_data[day,2:30])
```

```
}

quantile(boot_opt_sdev_ols$t[,1], probs=c(.025,.975))

##        2.5%       97.5% 
## 0.004755649 0.005984499

#standard deviation of returns
sd(ret_validate_ols)

## [1] 0.01042301

#sd from QP solution
opt_port_ols_sd

## [1] 0.00535836

#difference
sd(ret_validate_ols)-opt_port_ols_sd

## [1] 0.005064649
```

Optimal portfolio with LASSO covariance matrix (best lambda):

```
#function for bootstrap (2 arguments: data set and index-which rows of data
set)
#returns opt weights and sdev
get_opt_sdev_and_x_column_wise = function(input_data,index=NULL){
  #if no index is given, use entire data set
  if(is.null(index)){
    index = 1:dim(input_data)[1]
  }


  returns_data = input_data[index,2:30]
  factor_data = input_data[index,31:42]
  #how many stocks and how many days (columns and rows)
  nstocks = dim(returns_data)[2]
  ndays = dim(returns_data)[1]

  covf=cov(factor_data)

  M_lasso=matrix(nrow=12,ncol=29)
  v_lasso=matrix(0,nrow=29,ncol=29)

  for(i in 2:30){

    #Prep data
    y=input_data[,i]

x=model.matrix(y~Mkt+SMB+HML+RMW+CMA+DGS10+VAW+VCR+VDE+VFH+VGT+VHT,input_data
```

```
)[,-1]

    #Cross validation
    cv.out=cv.glmnet(x,y,alpha=1)

    #best lambda
    bestlam=cv.out$lambda.min

    #best lambda model
    lasso_mod = glmnet(x,y,alpha=1,lambda=bestlam)

    M_lasso[,i-1]=as.numeric(coef(lasso_mod))[2:13]

    ypred=predict(lasso_mod,x)
    v_lasso[i-1,i-1]=var(ypred-y)
  }

  #Covariance for LASSO model - best lambda
  covs_lasso=t(M_lasso)%*%covf%*%M_lasso+v_lasso
  cov_mat = covs_lasso

  #stuff to feed into gurobi
  port = list()
  port$modelsense = 'min'
  port$Q = cov_mat
  #contraint sum of xs
  port$A = array(1,c(1,nstocks))
  #constraint - must be equal to 1
  port$rhs = 1
  port$sense = '='

  #tell gurobi to shut up
  params = list()
  params$outputflag = 0;
  #plug into gurobi
  port_results = gurobi(port,params=params)

  output_variable = list()
  output_variable$sdev = sqrt(port_results$objval)
  output_variable$x = port_results$x

  output_variable
}

#this function calls 1st function and returns opt st dev
get_opt_sdev_column_wise = function(input_data,index=NULL){
  if(is.null(index)){
    index = 1:dim(input_data)[1]
  }
```

```
  val = get_opt_sdev_and_x_column_wise(input_data,index)

  val$sdev
}

boot_opt_sdev_lasso =
boot(finance_data[1:442,],get_opt_sdev_column_wise,R=500,parallel="multicore"
)


opt_port_lasso = get_opt_sdev_and_x_column_wise(finance_data[1:442,])
opt_port_lasso_sd = opt_port_lasso$sdev
opt_port_lasso = opt_port_lasso$x

#out of sample returns
ret_validate_lasso = array(0,100)
for(day in 443:542){      #weights times returns
  ret_validate_lasso[day-442] = sum(opt_port_lasso*finance_data[day,2:30])
}

quantile(boot_opt_sdev_lasso$t[,1], probs=c(.025,.975))
```

```
##        2.5%       97.5%
## 0.004655214 0.005753269
```

```
#out of sample standard deviation
sd(ret_validate_lasso)
```

```
## [1] 0.01028593
```

```
#sd from QP solution
opt_port_lasso_sd
```

```
## [1] 0.005237066
```

```
#difference
sd(ret_validate_lasso)-opt_port_lasso_sd
```

```
## [1] 0.005048863
```

Optimal portfolio with Ridge covariance matrix (best lambda):

```
#function for bootstrap (2 arguments: data set and index-which rows of data
set)
#returns opt weights and sdev
get_opt_sdev_and_x_column_wise = function(input_data,index=NULL){
  #if no index is given, use entire data set
  if(is.null(index)){
    index = 1:dim(input_data)[1]
  }
```

```
  returns_data = input_data[index,2:30]
  factor_data = input_data[index,31:42]
  #how many stocks and how many days (columns and rows)
  nstocks = dim(returns_data)[2]
  ndays = dim(returns_data)[1]

  covf=cov(factor_data)


  M_ridge=matrix(nrow=12,ncol=29)
  v_ridge=matrix(0,nrow=29,ncol=29)

  for(i in 2:30){

    #Prep data
    y=input_data[,i]

x=model.matrix(y~Mkt+SMB+HML+RMW+CMA+DGS10+VAW+VCR+VDE+VFH+VGT+VHT,input_data
)[,-1]

    #Cross validation
    cv.out=cv.glmnet(x,y,alpha=0)

    #best lambda
    bestlam=cv.out$lambda.min

    #best lambda model
    ridge_mod = glmnet(x,y,alpha=0,lambda=bestlam)

    M_ridge[,i-1]=coef(ridge_mod)[2:13]

    ypred=predict(ridge_mod,x)
    v_ridge[i-1,i-1]=var(ypred-y)
  }

  #Covariance for ridge model - best lambda
  covs_ridge=t(M_ridge)%*%covf%*%M_ridge+v_ridge

  cov_mat = covs_ridge

  #stuff to feed into gurobi
  port = list()
  port$modelsense = 'min'
  port$Q = cov_mat
  #contraint sum of xs
  port$A = array(1,c(1,nstocks))
```

```
  #constraint - must be equal to 1
  port$rhs = 1
  port$sense = '='

  #tell gurobi to shut up
  params = list()
  params$outputflag = 0;
  #plug into gurobi
  port_results = gurobi(port,params=params)

  output_variable = list()
  output_variable$sdev = sqrt(port_results$objval)
  output_variable$x = port_results$x

  output_variable
}

#this function calls 1st function and returns opt st dev
get_opt_sdev_column_wise = function(input_data,index=NULL){
  if(is.null(index)){
    index = 1:dim(input_data)[1]
  }

  val = get_opt_sdev_and_x_column_wise(input_data,index)

  val$sdev
}

boot_opt_sdev_ridge =
boot(finance_data[1:442,],get_opt_sdev_column_wise,R=500,parallel="multicore"
)



opt_port_ridge = get_opt_sdev_and_x_column_wise(finance_data[1:442,])
opt_port_ridge_sd = opt_port_ridge$sdev
opt_port_ridge = opt_port_ridge$x

#out of sample returns
ret_validate_ridge = array(0,100)
for(day in 443:542){      #weights times returns
  ret_validate_ridge[day-442] = sum(opt_port_ridge*finance_data[day,2:30])
}

quantile(boot_opt_sdev_ridge$t[,1], probs=c(.025,.975))

##        2.5%       97.5%
## 0.004544510 0.005561628
```

```r
#out of sample sd
sd(ret_validate_ridge)
```

```
## [1] 0.01017155
```

```r
#sd from QP solution
opt_port_ridge_sd
```

```
## [1] 0.005064793
```

```r
#difference
sd(ret_validate_ridge)-opt_port_ridge_sd
```

```
## [1] 0.005106758
```

Optimal portfolio with forward stepwise selection covariance matrix:

```r
#function for bootstrap (2 arguments: data set and index-which rows of data
set)
#returns opt weights and sdev
get_opt_sdev_and_x_column_wise = function(input_data,index=NULL){
  #if no index is given, use entire data set
  if(is.null(index)){
    index = 1:dim(input_data)[1]
  }


  returns_data = input_data[index,2:30]
  factor_data = input_data[index,31:42]
  #how many stocks and how many days (columns and rows)
  nstocks = dim(returns_data)[2]
  ndays = dim(returns_data)[1]

  covf=cov(factor_data)


  M_fwd=matrix(0,nrow=12,ncol=29)
  v_fwd=matrix(0,nrow=29,ncol=29)

  for(i in 2:30){

    #Prep data
    fwd_stocks_train=input_data[,c(i,31:42)]
    names(fwd_stocks_train)[1]="stock"

    #Cross validation
    k=10
    #separating data into folds
    folds=sample(1:k,nrow(fwd_stocks_train),replace=TRUE)
    cv.errors=matrix(0,k,12)
    for(j in 1:k){
```

```
fwd_fit=regsubsets(stock~Mkt+SMB+HML+RMW+CMA+DGS10+VAW+VCR+VDE+VFH+VGT+VHT,da
ta=fwd_stocks_train[folds!=j,],nvmax=12,method='forward')
      for(h in 1:12){
        form=stock~Mkt+SMB+HML+RMW+CMA+DGS10+VAW+VCR+VDE+VFH+VGT+VHT
        mat=model.matrix(form,fwd_stocks_train[folds==j,])
        coefi=coef(fwd_fit,h)
        xvars=names(coefi)
        pred=mat[,xvars]%*%coefi
        cv.errors[j,h]=mean((fwd_stocks_train$stock[folds==j]-pred)^2)
      }
    }

    mean.cv.errors=colMeans(cv.errors)
    #pinpointing the number of regressors that results in the lowest cross
validation error
    mincv = which.min(mean.cv.errors)

    #best model

fwd_mod=regsubsets(stock~Mkt+SMB+HML+RMW+CMA+DGS10+VAW+VCR+VDE+VFH+VGT+VHT,da
ta=fwd_stocks_train,nvmax=12,method='forward')

    #MUST INCLUDE 0 where there are none
    fwd_coefs=matrix(0,nrow=1,ncol=12)

    for(n in 2:(mincv+1)){
      for(p in 2:13){
        if(names(coef(fwd_mod,mincv))[n]==names(fwd_stocks_train)[p]){
          fwd_coefs[,p-1]=coef(fwd_mod,mincv)[n]
        }
      }
    }


    M_fwd[,i-1]=fwd_coefs

    #variance of residuals
    form=stock~Mkt+SMB+HML+RMW+CMA+DGS10+VAW+VCR+VDE+VFH+VGT+VHT
    mat=model.matrix(form,fwd_stocks_train)
    coefi=coef(fwd_mod,mincv)
    xvars=names(coefi)
    ypred=mat[,xvars]%*%coefi
    v_fwd[i-1,i-1]=var(ypred-fwd_stocks_train$stock)
  }

  #Covariance for fwd stepwise model
  covs_fwd=t(M_fwd)%*%covf%*%M_fwd+v_fwd
```

```
  cov_mat = covs_fwd

  #stuff to feed into gurobi
  port = list()
  port$modelsense = 'min'
  port$Q = cov_mat
  #contraint sum of xs
  port$A = array(1,c(1,nstocks))
  #constraint - must be equal to 1
  port$rhs = 1
  port$sense = '='

  #tell gurobi to shut up
  params = list()
  params$outputflag = 0;
  #plug into gurobi
  port_results = gurobi(port,params=params)

  output_variable = list()
  output_variable$sdev = sqrt(port_results$objval)
  output_variable$x = port_results$x

  output_variable
}

#this function calls 1st function and returns opt st dev
get_opt_sdev_column_wise = function(input_data,index=NULL){
  if(is.null(index)){
    index = 1:dim(input_data)[1]
  }

  val = get_opt_sdev_and_x_column_wise(input_data,index)

  val$sdev
}


boot_opt_sdev_fwd =
boot(finance_data[1:442,],get_opt_sdev_column_wise,R=500,parallel="multicore"
)


opt_port_fwd = get_opt_sdev_and_x_column_wise(finance_data[1:442,])
opt_port_fwd_sd = opt_port_fwd$sdev
opt_port_fwd = opt_port_fwd$x

#out of sample returns
ret_validate_fwd = array(0,100)
for(day in 443:542){      #weights times returns
```

```
  ret_validate_fwd[day-442] = sum(opt_port_fwd*finance_data[day,2:30])
}

quantile(boot_opt_sdev_fwd$t[,1], probs=c(.025,.975))

##        2.5%       97.5%
## 0.004658309 0.005865613

#out of sample sd
sd(ret_validate_fwd)

## [1] 0.01035183

#sd from QP solution
opt_port_fwd_sd

## [1] 0.005305865

#difference
sd(ret_validate_fwd)-opt_port_fwd_sd

## [1] 0.005045963
```

Optimal portfolio with LASSO covariance matrix (lambda 1se):

```
#function for bootstrap (2 arguments: data set and index-which rows of data
set)
#returns opt weights and sdev
get_opt_sdev_and_x_column_wise = function(input_data,index=NULL){
  #if no index is given, use entire data set
  if(is.null(index)){
    index = 1:dim(input_data)[1]
  }


  returns_data = input_data[index,2:30]
  factor_data = input_data[index,31:42]
  #how many stocks and how many days (columns and rows)
  nstocks = dim(returns_data)[2]
  ndays = dim(returns_data)[1]

  covf=cov(factor_data)


  M_lasso_1se=matrix(nrow=12,ncol=29)
  v_lasso_1se=matrix(0,nrow=29,ncol=29)

  for(i in 2:30){

    #Prep data
    y=input_data[,i]
```

```r
x=model.matrix(y~Mkt+SMB+HML+RMW+CMA+DGS10+VAW+VCR+VDE+VFH+VGT+VHT,input_data
)[,-1]

    #Cross validation
    cv.out=cv.glmnet(x,y,alpha=1)

    #lambda 1se
    lam_1se=cv.out$lambda.1se

    #lambda 1se model
    lasso_mod_1se=glmnet(x,y,alpha=1,lambda=lam_1se)

    M_lasso_1se[,i-1]=as.numeric(coef(lasso_mod_1se))[2:13]

    ypred_1se=predict(lasso_mod_1se,x)
    v_lasso_1se[i-1,i-1]=var(ypred_1se-y)
  }

  #Covariance for LASSO model - lambda 1se
  covs_lasso_1se=t(M_lasso_1se)%*%covf%*%M_lasso_1se+v_lasso_1se
  cov_mat = covs_lasso_1se

  #stuff to feed into gurobi
  port = list()
  port$modelsense = 'min'
  port$Q = cov_mat
  #contraint sum of xs
  port$A = array(1,c(1,nstocks))
  #constraint - must be equal to 1
  port$rhs = 1
  port$sense = '='

  #tell gurobi to shut up
  params = list()
  params$outputflag = 0;
  #plug into gurobi
  port_results = gurobi(port,params=params)

  output_variable = list()
  output_variable$sdev = sqrt(port_results$objval)
  output_variable$x = port_results$x

  output_variable
}

#this function calls 1st function and returns opt st dev
get_opt_sdev_column_wise = function(input_data,index=NULL){
  if(is.null(index)){
```

```
    index = 1:dim(input_data)[1]
  }

  val = get_opt_sdev_and_x_column_wise(input_data,index)

  val$sdev
}


boot_opt_sdev_l1se =
boot(finance_data[1:442,],get_opt_sdev_column_wise,R=500,parallel="multicore"
)


opt_port_l1se = get_opt_sdev_and_x_column_wise(finance_data[1:442,])
opt_port_l1se_sd = opt_port_l1se$sdev
opt_port_l1se = opt_port_l1se$x

#out of sample returns
ret_validate_l1se = array(0,100)
for(day in 443:542){      #weights times returns
  ret_validate_l1se[day-442] = sum(opt_port_l1se*finance_data[day,2:30])
}

quantile(boot_opt_sdev_l1se$t[,1], probs=c(.025,.975))

##        2.5%       97.5%
## 0.003342387 0.004005222

#out of sample sd
sd(ret_validate_l1se)

## [1] 0.01055318

#sd from QP solution
opt_port_l1se_sd

## [1] 0.00373507

#difference
sd(ret_validate_l1se)-opt_port_l1se_sd

## [1] 0.00681811
```

Optimal portfolio with Ridge covariance matrix (lambda 1se):

```
#function for bootstrap (2 arguments: data set and index-which rows of data
set)
#returns opt weights and sdev
get_opt_sdev_and_x_column_wise = function(input_data,index=NULL){
  #if no index is given, use entire data set
```

```
  if(is.null(index)){
    index = 1:dim(input_data)[1]
  }


  returns_data = input_data[index,2:30]
  factor_data = input_data[index,31:42]
  #how many stocks and how many days (columns and rows)
  nstocks = dim(returns_data)[2]
  ndays = dim(returns_data)[1]

  covf=cov(factor_data)


  M_ridge_1se=matrix(nrow=12,ncol=29)
  v_ridge_1se=matrix(0,nrow=29,ncol=29)

  for(i in 2:30){

    #Prep data
    y=input_data[,i]

x=model.matrix(y~Mkt+SMB+HML+RMW+CMA+DGS10+VAW+VCR+VDE+VFH+VGT+VHT,input_data
)[,-1]

    #Cross validation
    cv.out=cv.glmnet(x,y,alpha=0)

    #lambda 1se
    lam_1se=cv.out$lambda.1se

    #lambda 1se model
    ridge_mod_1se=glmnet(x,y,alpha=0,lambda=lam_1se)

    M_ridge_1se[,i-1]=as.numeric(coef(ridge_mod_1se))[2:13]

    ypred_1se=predict(ridge_mod_1se,x)
    v_ridge_1se[i-1,i-1]=var(ypred_1se-y)
  }

  #Covariance for ridge model - lambda 1se
  covs_ridge_1se=t(M_ridge_1se)%*%covf%*%M_ridge_1se+v_ridge_1se

  cov_mat = covs_ridge_1se

  #stuff to feed into gurobi
  port = list()
  port$modelsense = 'min'
```

```
  port$Q = cov_mat
  #contraint sum of xs
  port$A = array(1,c(1,nstocks))
  #constraint - must be equal to 1
  port$rhs = 1
  port$sense = '='

  #tell gurobi to shut up
  params = list()
  params$outputflag = 0;
  #plug into gurobi
  port_results = gurobi(port,params=params)

  output_variable = list()
  output_variable$sdev = sqrt(port_results$objval)
  output_variable$x = port_results$x

  output_variable
}

#this function calls 1st function and returns opt st dev
get_opt_sdev_column_wise = function(input_data,index=NULL){
  if(is.null(index)){
    index = 1:dim(input_data)[1]
  }

  val = get_opt_sdev_and_x_column_wise(input_data,index)

  val$sdev
}



boot_opt_sdev_r1se =
boot(finance_data[1:442,],get_opt_sdev_column_wise,R=500,parallel="multicore"
)



opt_port_r1se = get_opt_sdev_and_x_column_wise(finance_data[1:442,])
opt_port_r1se_sd = opt_port_r1se$sdev
opt_port_r1se = opt_port_r1se$x

#out of sample returns
ret_validate_r1se = array(0,100)
for(day in 443:542){      #weights times returns
  ret_validate_r1se[day-442] = sum(opt_port_r1se*finance_data[day,2:30])
}

quantile(boot_opt_sdev_r1se$t[,1], probs=c(.025,.975))
```

```
##        2.5%       97.5% 
## 0.003261720 0.003762758
```

```
#out of sample sd
sd(ret_validate_r1se)
```

```
## [1] 0.01033679
```

```
#sd from QP solution
opt_port_r1se_sd
```

```
## [1] 0.003487385
```

```
#difference
sd(ret_validate_r1se)-opt_port_r1se_sd
```

```
## [1] 0.006849406
```

Equally weighted portfolio:

```
#create an equally weighted portfolio
eq_port=array(1/29,dim=29)

#out of sample returns for an equally weighted portfolio
ret_validate_equal = array(0,100)
for(day in 443:542){
  ret_validate_equal[day-442] = sum(eq_port*finance_data[day,2:30])
}

#out of sample sd
sd(ret_validate_equal)
```

```
## [1] 0.01267459
```

Overall, the Ridge model had the best out of sample standard deviation. However, LASSO had the smallest difference between the out of sample standard deviation and the standard deviation from the original optimization problem. In general, all the models did somewhat better than the simple covariance model except for the ordinary least squares model (and the LASSO model with 1se lambda). However, the equally weighted portfolio performed the worst. None of the factor models fell within the confidence interval for out of sample standard deviation.