Final Project

Individual Project

2019/12/11

Member:

Dominique Nie MSFTA 474148

```
library(bayesm)
library(m537)
```

```
## Loading required package: coda

## Loading required package: devtools

## Loading required package: usethis

## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

##
## Attaching package: 'm537'

## The following object is masked from 'package:stats':
##
##     sigma
```

```
library(m537tools)
library(xts)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(ggplot2)
library(future.apply)
```

```
## Loading required package: future
```

```
library(sqldf)
```

```
## Loading required package: gsubfn

## Loading required package: proto

## Loading required package: RSQLite
```

```
library(quantmod)
```

```
## Loading required package: TTR

## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```
library(parallel)
rm(list = ls())
```

Diversification

1.Select 30 stocks that you are interested in. Find their yahoo symbols.

```
company_symbols = c("AAPL","MSFT","AMZN","FB","GOOG","GOOGL","INTC","CMCSA","CSCO","PEP",
"ADBE","AMGN","COST","NFLX","NVDA","AVGO","TXN","CHTR","QCOM","SBUX","BKNG","GILD","CELG",
"MDLZ","FISV","ADP","TMUS","INTU","ISRG","CSX","^gspc")

company_names = c("apple","microsoft","amazon","facebook","alphabetc","alphabeta","intel",
"comcast","cisco","pepsico","adobe","amgen","costco","netflix","nvidia","broadcom",
"texasinstruments","chater","qualcomm","starbucks","booking","gilead","celgene","mondelez",
"fiserv","automaticdata","tmobile","intuit","intuitivesurg","csx","sp500")
```

Above are the 30 stocks I selected.

2.Download 4 years of weekly price data for each stock from June 1, 2015 to June 1, 2019.

```
prmdf = getfinwdat(symbols = company_symbols,
                   symnames = company_names,
                   from = "2015-06-01",
                   to = "2019-06-01")
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
## Warning: ^IRX contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## Warning in to.period(datxts, indexAt = "endof", period = "weeks"): missing
## values removed from data
```

4 years of weekly price data for each stock I choose from June 1, 2015 to June 1, 2019 have been downloaded and stored in "prmdf". (All 30 stocks I select in step 1 are available for these 4 years)

3.Assume that the desired portfolio mean return levels are .03, .06, .09 and .12, each in annual terms.

```
desired_mean_weekly_return = c(.03,.06,.09,.12)/52
```

The desired portfolio mean return levels are specified in weekly terms as above.

4.Now form 6 portfolios for each desired return level (each portfolio includes the risk-free asset). Each portfolio has 5, 10, 15, 20, 25 and 30 stocks. The smaller set of assets should be a subset of the larger set.

5.Assume that the stock premium is explained by the Gaussian SURE CAPM model without an intercept.

6.Now form each portfolio, use the default training sample prior. Comment on the prior in the case of the SURE model with 15 assets.

7.Now compute the optimal portfolios for each group of assets at each target portfolio return level. Give the weights of each asset in each portfolio as well as the standard deviation of the optimal portfolios.

```
set.seed(1)
portfolio_name = c("sp500")
loop_names = c("apple","microsoft","amazon","facebook","alphabetc","alphabeta","intel",
"comcast","cisco","pepsico","adobe","amgen","costco","netflix","nvidia","broadcom",
"texasinstruments","chater","qualcomm","starbucks","booking","gilead","celgene","mondelez",
"fiserv","automaticdata","tmobile","intuit","intuitivesurg","csx")
modelsize = c(1:6)
mean_std = rep(0,6)
return_sd = c(.03,.06,.09,.12)
```

```r
#Question 4
for (i in 1:6){
    new_portfolio_name = sample(loop_names,5,replace = FALSE)
    portfolio_name = c(portfolio_name,new_portfolio_name)   #company names used in portfolio
    loop_names = setdiff(loop_names,portfolio_name)
    facfrmls = list()
    for (j in portfolio_name[-1]){
        frm = paste(j , "~ prmsp500-1")
        frm = paste("prm",frm,sep = "")
        frm = as.formula(frm)
        facfrmls = append(facfrmls,frm)
    }   #set model frame list
    #Question 6
    if(i == 3){
        s = 2   #use the last 2 as prediction
        ns = dim(prmdf)[1]
        prmdfn = prmdf[1:(ns-s),]   #sample data
        n = dim(prmdfn)[1]
        prmdff = prmdf[-(1:n),]
        priols = trainpriorsureg(modelfrmls = facfrmls,
                                 data = prmdfn,trainpct = .15)
        print("The prior sample size of portfolio with 15 assets ")
        print(priols$nt)
        print("The proir data size of portfolio with 15 assets ")
        print(priols$n)
        print("The total proir data size of portfolio with 15 assets ")
        print(priols$nn)
        print("The number of prior beta of portfolio with 15 assets ")
        print(priols$k)
        print("The number of eigenvalue of covariance matrix in portfolio with 15 assets ")
        print(priols$d)
    }
    #Question 7
    capmportls = makebayesportfolioaftersureg(portmean = desired_mean_weekly_return,
                                              modelfrmls = facfrmls,data = prmdf)
    #get optimal portfolio solution under four expected return
    print(paste("The weight of optimal portfolio ", i))
    print(capmportls$weights)   #the weights of each asset in each portfolio
    print(paste("The standard deviation of optimal portfolio ", i))
    print(capmportls$portsd)   #the standard deviation of the optimal portfolios
    return_sd = cbind(return_sd,capmportls$portsd)
}
```

```
## Warning: ^IRX contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## Warning in to.period(datxts, indexAt = "endof", period = "weeks"): missing
## values removed from data
```

```
## [1] "The weight of optimal portfolio  1"
##       wt_fiserv   wt_facebook    wt_intel     wt_apple wt_microsoft
## [1,] 0.05528699 -0.0001086864 0.004383067 -0.003347729   0.03920882
## [2,] 0.29820422 -0.0005862275 0.023641169 -0.018056816   0.21148260
## [3,] 0.54112146 -0.0010637685 0.042899271 -0.032765902   0.38375638
## [4,] 0.78403870 -0.0015413095 0.062157374 -0.047474989   0.55603016
##       wt_rskfree
## [1,]  0.90457754
## [2,]  0.48531505
## [3,]  0.06605256
## [4,] -0.35320993
## [1] "The standard deviation of optimal portfolio  1"
##             [,1]
## [1,] 0.002156981
## [2,] 0.011634216
## [3,] 0.021111451
## [4,] 0.030588686
```

```
## Warning: ^IRX contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## Warning: missing values removed from data
```

```
## [1] "The weight of optimal portfolio  2"
##       wt_fiserv wt_facebook   wt_intel   wt_apple wt_microsoft
## [1,] 0.00824978 -0.01125343 0.02384963 0.00841848   0.02262593
## [2,] 0.04449726 -0.06069821 0.12863894 0.04540718   0.12203865
## [3,] 0.08074473 -0.11014299 0.23342825 0.08239589   0.22145136
## [4,] 0.11699221 -0.15958777 0.33821756 0.11938459   0.32086408
##      wt_starbucks   wt_tmobile  wt_gilead wt_mondelez wt_texasinstruments
## [1,] -0.003888413 -0.003934457 0.01492658  0.01275877         0.004206648
## [2,] -0.020973132 -0.021221482 0.08051023  0.06881764         0.022689612
## [3,] -0.038057850 -0.038508507 0.14609389  0.12487650         0.041172576
## [4,] -0.055142569 -0.055795532 0.21167755  0.18093537         0.059655540
##       wt_rskfree
## [1,]  0.92404048
## [2,]  0.59029331
## [3,]  0.25654614
## [4,] -0.07720103
## [1] "The standard deviation of optimal portfolio  2"
##             [,1]
## [1,] 0.001811433
## [2,] 0.009770419
## [3,] 0.017729405
## [4,] 0.025688391
## [1] "The prior sample size of portfolio with 15 assets "
## [1] 31
## [1] "The proir data size of portfolio with 15 assets "
## [1] 175
## [1] "The total proir data size of portfolio with 15 assets "
## [1] 2625
## [1] "The number of prior beta of portfolio with 15 assets "
## [1] 15
## [1] "The number of eigenvalue of covariance matrix in portfolio with 15 assets "
## [1] 15
## Warning: ^IRX contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## Warning: missing values removed from data
```

```
## [1] "The weight of optimal portfolio  3"
##       wt_fiserv wt_facebook     wt_intel    wt_apple wt_microsoft
## [1,] 0.01814819  0.02228251 -0.006558705 0.007339159   0.01319103
## [2,] 0.09788683  0.12018628 -0.035376019 0.039585595   0.07114910
## [3,] 0.17762547  0.21809005 -0.064193333 0.071832031   0.12910718
## [4,] 0.25736411  0.31599383 -0.093010647 0.104078466   0.18706526
##      wt_starbucks   wt_tmobile   wt_gilead wt_mondelez wt_texasinstruments
## [1,]   0.01388127 -0.004404992 0.004109161 0.001899457        -0.009157062
## [2,]   0.07487210 -0.023759426 0.022163792 0.010245196        -0.049390909
## [3,]   0.13586294 -0.043113861 0.040218422 0.018590935        -0.089624756
## [4,]   0.19685377 -0.062468295 0.058273052 0.026936674        -0.129858602
##      wt_intuitivesurg wt_alphabetc     wt_adobe   wt_costco     wt_csx
## [1,]       0.01386800  -0.01745209 -0.001938986 0.001330842 0.01884956
## [2,]       0.07480054  -0.09413221 -0.010458408 0.007178229 0.10166985
## [3,]       0.13573308  -0.17081233 -0.018977830 0.013025615 0.18449014
## [4,]       0.19666562  -0.24749245 -0.027497253 0.018873002 0.26731042
##       wt_rskfree
## [1,]  0.92461265
## [2,]  0.59337945
## [3,]  0.26214624
## [4,] -0.06908696
## [1] "The standard deviation of optimal portfolio  3"
##             [,1]
## [1,] 0.001737806
## [2,] 0.009373294
## [3,] 0.017008782
## [4,] 0.024644269
```

```
## Warning: ^IRX contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## Warning: missing values removed from data
```

```
## [1] "The weight of optimal portfolio  4"
##        wt_fiserv wt_facebook     wt_intel   wt_apple wt_microsoft
## [1,] -0.01208133 0.007058689 0.0002693334 0.01260803 -0.001111623
## [2,] -0.06516367 0.038072810 0.0014527171 0.06800457 -0.005995815
## [3,] -0.11824601 0.069086931 0.0026361008 0.12340110 -0.010880008
## [4,] -0.17132835 0.100101052 0.0038194845 0.17879764 -0.015764201
##      wt_starbucks  wt_tmobile   wt_gilead wt_mondelez wt_texasinstruments
## [1,]  0.005551236 0.005517323 0.007944582 -0.00634209         -0.01321852
## [2,]  0.029941982 0.029759062 0.042851094 -0.03420765         -0.07129743
## [3,]  0.054332729 0.054000802 0.077757605 -0.06207322         -0.12937633
## [4,]  0.078723476 0.078242542 0.112664117 -0.08993878         -0.18745523
##      wt_intuitivesurg  wt_alphabetc   wt_adobe   wt_costco      wt_csx
## [1,]     5.940023e-05 -0.0009426369 0.01370934 -0.00257129 0.005518676
## [2,]     3.203900e-04 -0.0050843483 0.07394476 -0.01386890 0.029766361
## [3,]     5.813798e-04 -0.0092260597 0.13418018 -0.02516650 0.054014046
## [4,]     8.423696e-04 -0.0133677711 0.19441560 -0.03646411 0.078261731
##      wt_comcast     wt_amgen    wt_cisco   wt_nvidia    wt_chater
## [1,] 0.03692496 -0.007525889 0.007883048 0.008233167 -0.007938241
## [2,] 0.19916405 -0.040592767 0.042519195 0.044407650 -0.042816891
## [3,] 0.36140313 -0.073659645 0.077155341 0.080582132 -0.077695541
## [4,] 0.52364221 -0.106726523 0.111791488 0.116756615 -0.112574191
##      wt_rskfree
## [1,]  0.9404538
## [2,]  0.6788228
## [3,]  0.4171918
## [4,]  0.1555608
## [1] "The standard deviation of optimal portfolio  4"
##             [,1]
## [1,] 0.001611222
## [2,] 0.008690532
## [3,] 0.015769841
## [4,] 0.022849150

## Warning: ^IRX contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.

## Warning: missing values removed from data

## [1] "The weight of optimal portfolio  5"
##        wt_fiserv  wt_facebook   wt_intel    wt_apple wt_microsoft
## [1,] 0.001425422 -0.004075808 0.01023777 0.002753808  0.005502859
## [2,] 0.007688369 -0.021983890 0.05521996 0.014853354  0.029681050
## [3,] 0.013951316 -0.039891972 0.10020215 0.026952900  0.053859242
## [4,] 0.020214263 -0.057800055 0.14518434 0.039052446  0.078037433
##      wt_starbucks   wt_tmobile   wt_gilead  wt_mondelez
## [1,] -0.002884477 -0.006631351 0.005391713 -0.006060065
## [2,] -0.015558148 -0.035767856 0.029081556 -0.032686481
## [3,] -0.028231819 -0.064904360 0.052771398 -0.059312897
## [4,] -0.040905491 -0.094040864 0.076461241 -0.085939313
##      wt_texasinstruments wt_intuitivesurg wt_alphabetc    wt_adobe
## [1,]          0.01850780      0.006912725    0.1853385 -0.00839714
## [2,]          0.09982647      0.037285515    0.9996697 -0.04529208
## [3,]          0.18114513      0.067658305    1.8140010 -0.08218702
## [4,]          0.26246380      0.098031096    2.6283322 -0.11908196
##       wt_costco      wt_csx  wt_comcast     wt_amgen    wt_cisco
## [1,] 0.01669344 0.007677786 -0.01313468 -0.004930785 -0.01321403
## [2,] 0.09004023 0.041412065 -0.07084517 -0.026595423 -0.07127317
## [3,] 0.16338703 0.075146343 -0.12855566 -0.048260062 -0.12933232
## [4,] 0.23673383 0.108880621 -0.18626615 -0.069924701 -0.18739146
##        wt_nvidia   wt_chater wt_booking  wt_netflix wt_alphabeta
## [1,] -0.00185536 0.005440551 0.01134154 0.001174348   -0.1676736
## [2,] -0.01000735 0.029344977 0.06117342 0.006334142   -0.9043895
## [3,] -0.01815934 0.053249403 0.11100530 0.011493937   -1.6411054
## [4,] -0.02631133 0.077153830 0.16083719 0.016653731   -2.3778213
##       wt_celgene  wt_pepsico wt_rskfree
## [1,] 0.004361523 -0.01017002  0.9562675
## [2,] 0.023524965 -0.05485458  0.7641178
## [3,] 0.042688408 -0.09953913  0.5719681
## [4,] 0.061851851 -0.14422369  0.3798184
## [1] "The standard deviation of optimal portfolio  5"
##             [,1]
## [1,] 0.001469681
## [2,] 0.007927092
## [3,] 0.014384503
## [4,] 0.020841914
```

```
## Warning: ^IRX contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## Warning: missing values removed from data
```

```
## [1] "The weight of optimal portfolio  6"
##       wt_fiserv wt_facebook    wt_intel   wt_apple wt_microsoft
## [1,] 0.01988714 0.004612868 -0.00260188 0.00476825   0.03347306
## [2,] 0.10726629 0.024880663 -0.01403389 0.02571875   0.18054534
## [3,] 0.19464544 0.045148457 -0.02546590 0.04666925   0.32761762
## [4,] 0.28202458 0.065416251 -0.03689792 0.06761975   0.47468991
##      wt_starbucks   wt_tmobile   wt_gilead wt_mondelez wt_texasinstruments
## [1,]  0.002798272 -0.002042617 0.003588516  -0.0198209         0.004970194
## [2,]  0.015093183 -0.011017369 0.019355558  -0.1069090         0.026807989
## [3,]  0.027388093 -0.019992120 0.035122601  -0.1939971         0.048645783
## [4,]  0.039683004 -0.028966872 0.050889643  -0.2810852         0.070483578
##      wt_intuitivesurg wt_alphabetc     wt_adobe  wt_costco       wt_csx
## [1,]       0.00527168  -0.03547325 0.0008446581 0.01558055 -0.002623346
## [2,]       0.02843413  -0.19133387 0.0045558754 0.08403758 -0.014149674
## [3,]       0.05159658  -0.34719449 0.0082670928 0.15249462 -0.025676002
## [4,]       0.07475902  -0.50305512 0.0119783101 0.22095166 -0.037202330
##       wt_comcast    wt_amgen   wt_cisco    wt_nvidia  wt_chater
## [1,] 0.002396123 0.002319339 0.00855665 -0.003731798 0.01206320
## [2,] 0.012924089 0.012509935 0.04615244 -0.020128389 0.06506587
## [3,] 0.023452056 0.022700532 0.08374822 -0.036524980 0.11806855
## [4,] 0.033980022 0.032891128 0.12134401 -0.052921570 0.17107122
##       wt_booking    wt_netflix wt_alphabeta   wt_celgene   wt_pepsico
## [1,] 0.001270675 -0.0003751828   0.01420750 -0.004799083 -0.005964935
## [2,] 0.006853705 -0.0020236423   0.07663171 -0.025885059 -0.032173374
## [3,] 0.012436735 -0.0036721018   0.13905593 -0.046971034 -0.058381814
## [4,] 0.018019764 -0.0053205613   0.20148014 -0.068057010 -0.084590253
##         wt_intuit wt_qualcomm wt_automaticdata    wt_amazon wt_broadcom
## [1,] -0.006472245  0.00202708    -0.0003790098 -0.003627842  0.01161060
## [2,] -0.034909678  0.01093357    -0.0020442843 -0.019567674  0.06262469
## [3,] -0.063347111  0.01984005    -0.0037095588 -0.035507507  0.11363877
## [4,] -0.091784544  0.02874654    -0.0053748334 -0.051447340  0.16465286
##      wt_rskfree
## [1,]  0.9376657
## [2,]  0.6637845
## [3,]  0.3899033
## [4,]  0.1160221
## [1] "The standard deviation of optimal portfolio  6"
##             [,1]
## [1,] 0.001527265
## [2,] 0.008237688
## [3,] 0.014948112
## [4,] 0.021658535
```

The optimal portfolio with 5 stocks is: fiserv, facebook, intel, apple, microsoft, rskfree. The weights of each asset and the standard deviation of the optimal portfolio for each annual return level are shown as above.

The optimal portfolio with 10 stocks is: fiserv, facebook, intel, apple, microsoft, mondelez, comcast, intuitivesurg, alphabetc, gilead, rskfree. The weights of each asset and the standard deviation of the optimal portfolio for each annual return level are shown as above.

The optimal portfolio with 15 stocks is: fiserv, facebook, intel, apple, microsoft, mondelez, comcast, intuitivesurg, alphabetc, gilead, nvidia, automaticdata, intuit, amazon, chate, rskfree. The weights of each asset and the standard deviation of the optimal portfolio for each annual return level are shown as above.

The optimal portfolio with 20 stocks is: fiserv, facebook, intel, apple, microsoft, mondelez, comcast, intuitivesurg, alphabetc, gilead, nvidia, automaticdata, intuit, amazon, chate, broadcom, starbucks, cisco, adobe, booking, rskfree. The weights of each asset and the standard deviation of the optimal portfolio for each annual return level are shown as above.

The optimal portfolio with 25 stocks is: fiserv, facebook, intel, apple, microsoft, mondelez, comcast, intuitivesurg, alphabetc, gilead, nvidia, automaticdata, intuit, amazon, chate, broadcom, starbucks, cisco, adobe, booking, pepsico, costco, tmobile, csx, celgene, rskfree. The weights of each asset and the standard deviation of the optimal portfolio for each annual return level are shown as above.

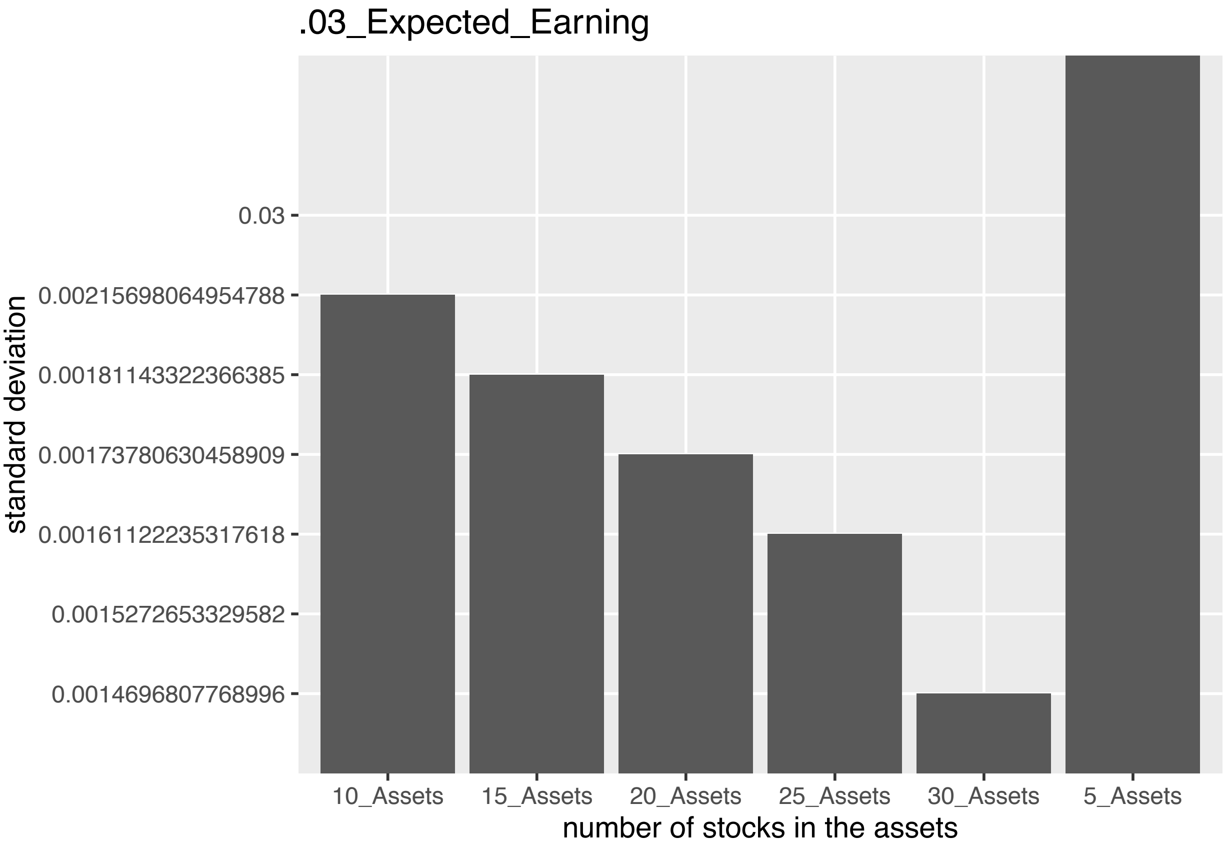
The optimal portfolio with 30 stocks is: fiserv, facebook, intel, apple, microsoft, mondelez, comcast, intuitivesurg, alphabetc, gilead, nvidia, automaticdata, intuit, amazon, chate, broadcom, starbucks, cisco, adobe, booking, pepsico, costco, tmobile, csx, celgene, netflix, qualcomm, texasinstruments, amgen, alphabeta, rskfree. The weights of each asset and the standard deviation of the optimal portfolio for each annual return level are shown as above.

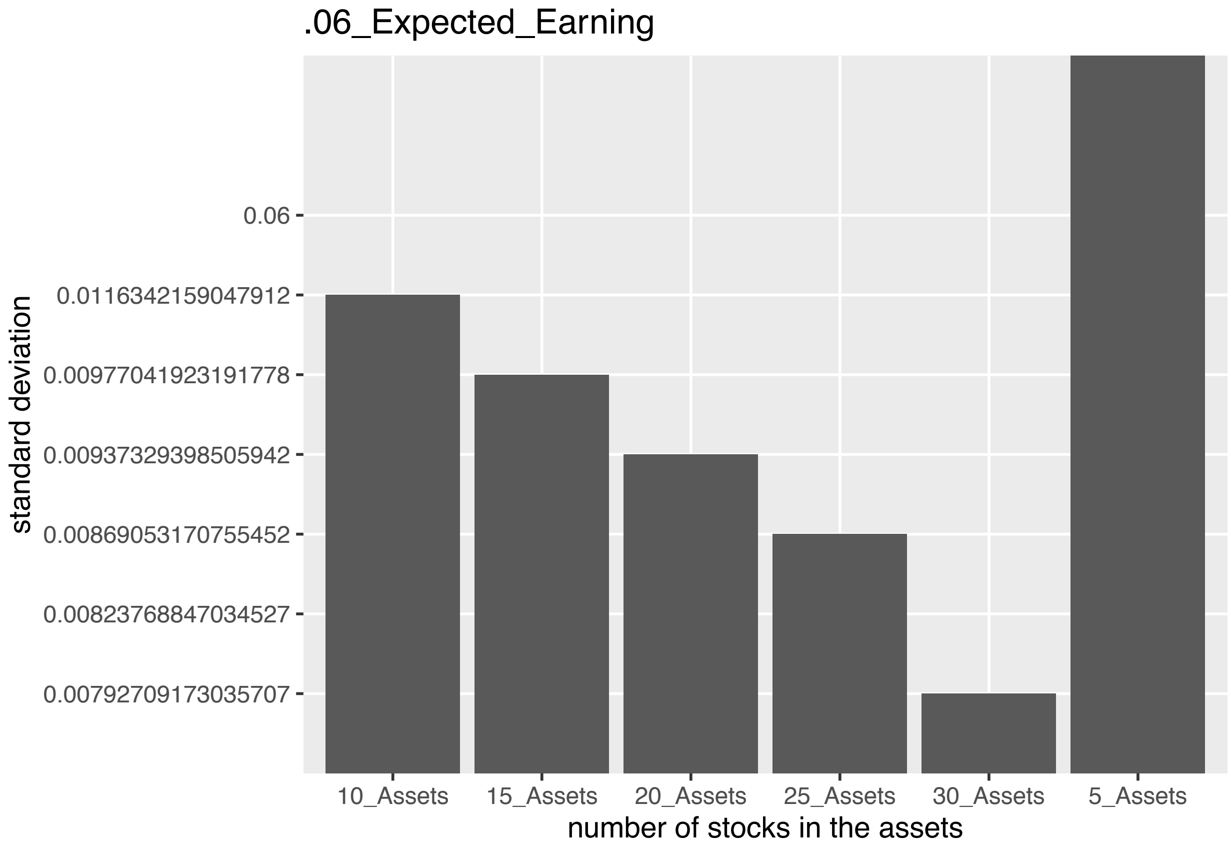
8.Use ggplot to plot the standard deviation vs. the number of stocks in the assets. Comment on your findings.

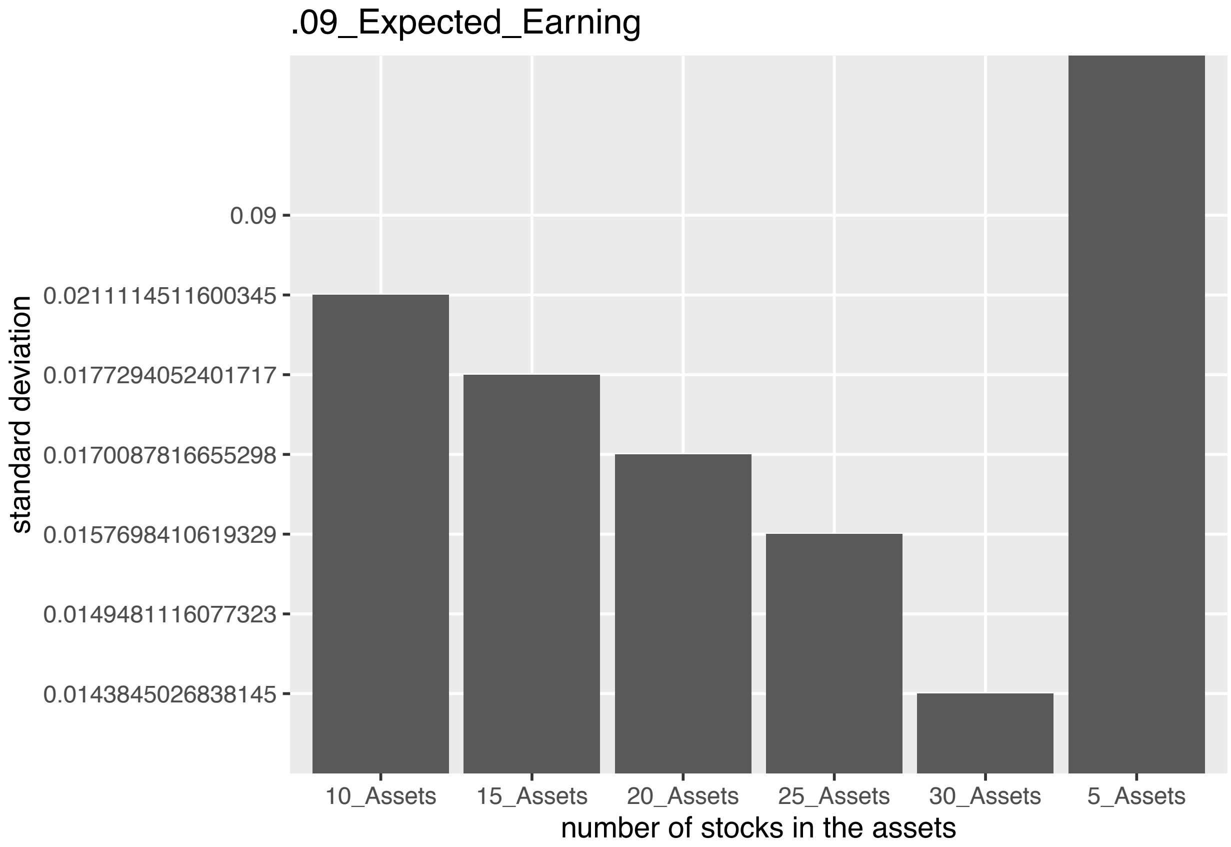
```r
return_sd = t(return_sd)   #6*4 matrix
rownames(return_sd) = NULL
c = c("5_Assets","10_Assets","15_Assets","20_Assets","25_Assets","30_Assets")
return_sd = cbind(c, return_sd)
```

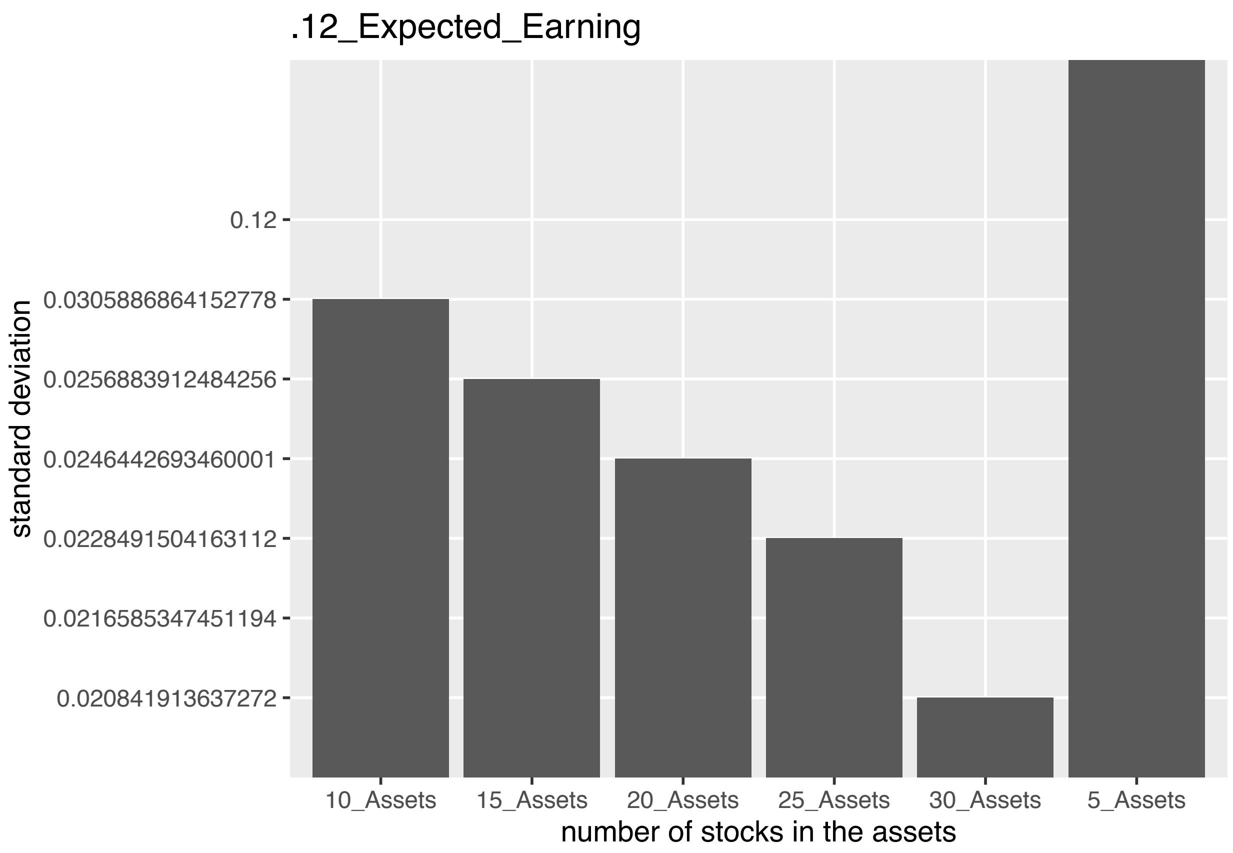
```
## Warning in cbind(c, return_sd): number of rows of result is not a multiple
## of vector length (arg 1)
```

```r
plt_data = as.data.frame(return_sd)   #6*5 matrix
colnames(plt_data) = c("Assets_Type",".03_Expected_Earning",".06_Expected_Earning",
  ".09_Expected_Earning",".12_Expected_Earning")   #rename colunms
for(i in 2:5){
    p = ggplot(data = plt_data,aes(x=Assets_Type,y=plt_data[,i]))
    p1 = p + geom_bar(stat = "identity") + labs(title = colnames(plt_data)[i])
    p2 = p1 + labs(x = "number of stocks in the assets",
                   y = "standard deviation")
    print(p2)   #plot
}
```


.03_Expected_Earning
standard deviation
0.03
0.00215698064954788
0.00181143322366385
0.00173780630458909
0.00161122235317618
0.0015272653329582
0.0014696807768996
10_Assets
15_Assets
20_Assets
25_Assets
30_Assets
5_Assets
number of stocks in the assets


.06_Expected_Earning
standard deviation
0.06
0.0116342159047912
0.00977041923191778
0.00937329398505942
0.00869053170755452
0.00823768847034527
0.00792709173035707
10_Assets
15_Assets
20_Assets
25_Assets
30_Assets
5_Assets
number of stocks in the assets


.09_Expected_Earning
standard deviation
0.09
0.0211114511600345
0.0177294052401717
0.0170087816655298
0.0157698410619329
0.0149481116077323
0.0143845026838145
10_Assets
15_Assets
20_Assets
25_Assets
30_Assets
5_Assets
number of stocks in the assets



9.Redo questions 5-8 with student-t errors. For each set of assets, use log-marginal likelihoods to find the appropriate-degrees of freedom of the student-t distribution on a grid of 10 equally-spaced values between 3 and 5.

```
set.seed(1)
portfolio_name = c("sp500")
loop_names = c("apple","microsoft","amazon","facebook","alphabetc","alphabeta","intel",
"comcast","cisco","pepsico","adobe","amgen","costco","netflix","nvidia","broadcom",
"texasinstruments","chater","qualcomm","starbucks","booking","gilead","celgene","mondelez",
"fiserv","automaticdata","tmobile","intuit","intuitivesurg","csx")
modelsize = c(1:6)
mean_std = rep(0,6)
```

```
nug = seq(from = 3,
          to = 5,
          by = .20)
nug = as.matrix(nug);
return_sd = c(.03,.06,.09,.12)
```

```
for(i in 1:6){
    new_portfolio_name = sample(loop_names,5,replace = FALSE)
    portfolio_name = c(portfolio_name,new_portfolio_name)   #company names used in portfolio
    loop_names = setdiff(loop_names,portfolio_name)
    facfrmls = list()
    for (j in portfolio_name[-1]){
        frm = paste(j , "~ prmsp500-1")
        frm = paste("prm",frm,sep = "")
        frm = as.formula(frm)
        facfrmls = append(facfrmls,frm)
    }   #set model frame list

    datls = suremat(modelfrmls = facfrmls,datdf = prmdf)
    outls = future_mapply("MCMCsuret",nu = nug,
                          MoreArgs = list(modelfrm = facfrmls,
                                          data = prmdf),
                          SIMPLIFY = FALSE)
    nug_list = t(logmarglik(outls))
    nug_df = cbind(nug,nug_list)
    best_nu = nug_df[which.max(nug_df[,2]),1]
    print(paste("The best nu of optimal portfolio ", i))
    print(best_nu)   #find the appropriate-degrees of freedom

    capmportls = makebayesportfolioaftersuret(portmean = desired_mean_weekly_return,
                                              modelfrmls = facfrmls,data = prmdf,nu = best_nu)
    #get optimal portfolio solution under four expected return
    print(paste("The weight of optimal portfolio ", i))
    print(capmportls$weights)   #the weights of each asset in each portfolio
    print(paste("The weight of optimal portfolio ", i))
    print(capmportls$portsd)   #the standard deviation of the optimal portfolios
    return_sd = cbind(return_sd,capmportls$portsd)
}
```

```
## [1] "The best nu of optimal portfolio  1"
## [1] 4
```

```
## Warning: ^IRX contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## Warning in to.period(datxts, indexAt = "endof", period = "weeks"): missing
## values removed from data
```

```
## [1] "The weight of optimal portfolio  1"
##       wt_fiserv wt_facebook   wt_intel   wt_apple  wt_microsoft
## [1,] 0.03611085   0.0257180 0.02738947 0.01049297 -0.0007006469
## [2,] 0.19477292   0.1387165 0.14773200 0.05659647 -0.0037791148
## [3,] 0.35343499   0.2517149 0.26807452 0.10269996 -0.0068575826
## [4,] 0.51209706   0.3647134 0.38841704 0.14880346 -0.0099360505
##       wt_rskfree
## [1,]  0.90098935
## [2,]  0.46596126
## [3,]  0.03093317
## [4,] -0.40409493
## [1] "The weight of optimal portfolio  1"
##             [,1]
## [1,] 0.002871138
## [2,] 0.015486204
## [3,] 0.028101270
## [4,] 0.040716335
## [1] "The best nu of optimal portfolio  2"
## [1] 4.4

## Warning: ^IRX contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.

## Warning: missing values removed from data

## [1] "The weight of optimal portfolio  2"
##         wt_fiserv wt_facebook    wt_intel    wt_apple wt_microsoft
## [1,] -0.005762338 0.001710718 0.002868108 -0.01206272   0.03466039
## [2,] -0.031080614 0.009227184 0.015469859 -0.06506330   0.18694949
## [3,] -0.056398890 0.016743650 0.028071611 -0.11806388   0.33923859
## [4,] -0.081717166 0.024260117 0.040673362 -0.17106446   0.49152770
##       wt_intuit wt_starbucks wt_qualcomm  wt_gilead   wt_chater wt_rskfree
## [1,] 0.01490605   0.01463618  0.01472957 0.01202226 0.007038532  0.9152532
## [2,] 0.08039954   0.07894392  0.07944761 0.06484508 0.037964087  0.5428971
## [3,] 0.14589303   0.14325166  0.14416565 0.11766791 0.068889642  0.1705410
## [4,] 0.21138652   0.20755940  0.20888369 0.17049073 0.099815197 -0.2018151
## [1] "The weight of optimal portfolio  2"
##            [,1]
## [1,] 0.00225371
## [2,] 0.01215595
## [3,] 0.02205819
## [4,] 0.03196043
## [1] "The best nu of optimal portfolio  3"
## [1] 3.8

## Warning: ^IRX contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.

## Warning: missing values removed from data

## [1] "The weight of optimal portfolio  3"
##       wt_fiserv  wt_facebook    wt_intel    wt_apple wt_microsoft
## [1,] 0.01439533 -0.004219228 0.002158904 0.006275974   0.02316819
## [2,] 0.07764484 -0.022757466 0.011644587 0.033851038   0.12496344
## [3,] 0.14089435 -0.041295703 0.021130271 0.061426102   0.22675869
## [4,] 0.20414386 -0.059833941 0.030615955 0.089001165   0.32855394
##        wt_intuit wt_starbucks wt_qualcomm   wt_gilead   wt_chater
## [1,] 0.006646003   0.01297703  0.01208679 0.008080213 0.001622878
## [2,] 0.035846884   0.06999489  0.06519313 0.043582655 0.008753399
## [3,] 0.065047765   0.12701274  0.11829948 0.079085097 0.015883921
## [4,] 0.094248646   0.18403059  0.17140582 0.114587539 0.023014442
##       wt_celgene    wt_amazon   wt_costco    wt_cisco  wt_tmobile
## [1,] 0.002027741 -0.008217989 0.003337648 -0.01719474 0.004423061
## [2,] 0.010937131 -0.044325783 0.018002440 -0.09274413 0.023856889
## [3,] 0.019846520 -0.080433578 0.032667232 -0.16829353 0.043290716
## [4,] 0.028755910 -0.116541372 0.047332023 -0.24384292 0.062724544
##      wt_rskfree
## [1,]  0.9324322
## [2,]  0.6355561
## [3,]  0.3386799
## [4,]  0.0418038
## [1] "The weight of optimal portfolio  3"
##             [,1]
## [1,] 0.001777580
## [2,] 0.009587823
## [3,] 0.017398067
## [4,] 0.025208310
## [1] "The best nu of optimal portfolio  4"
## [1] 3
```

```
## Warning: ^IRX contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

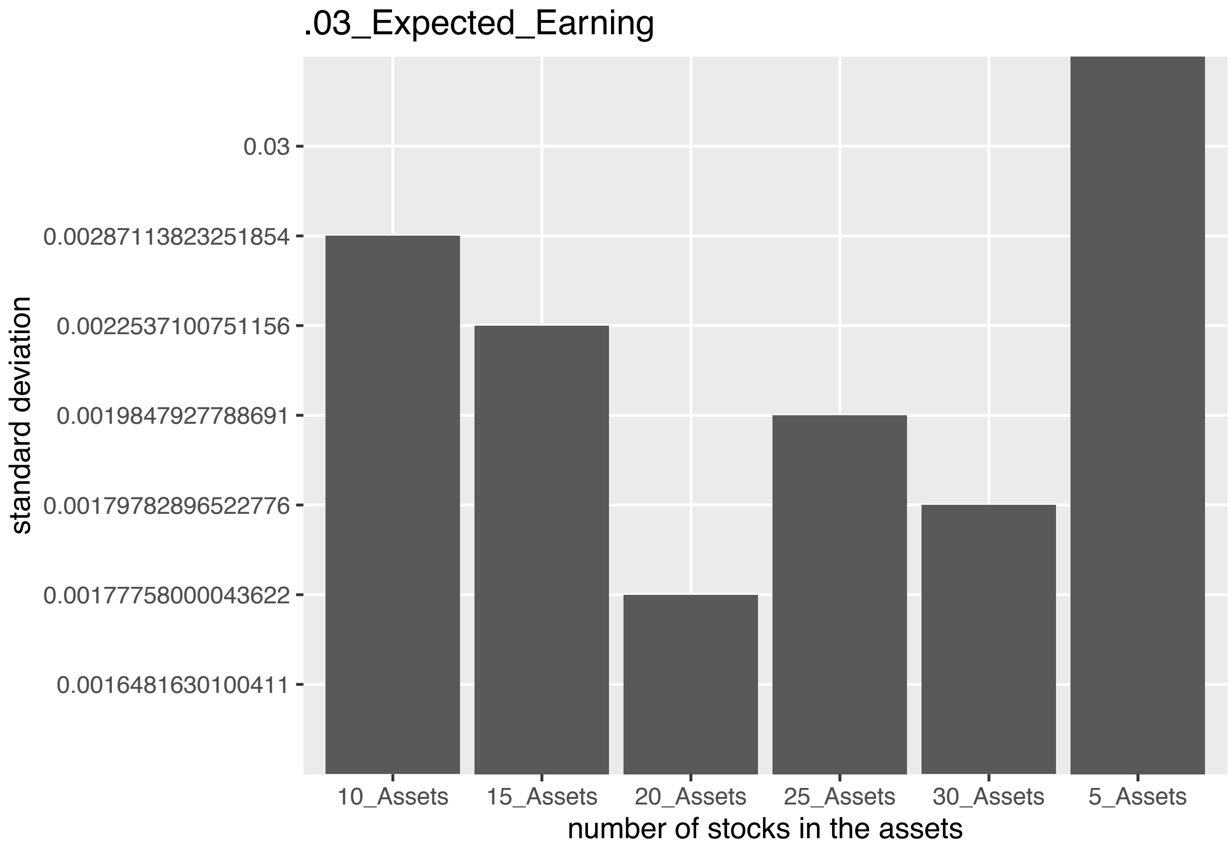
```
## Warning: missing values removed from data
```

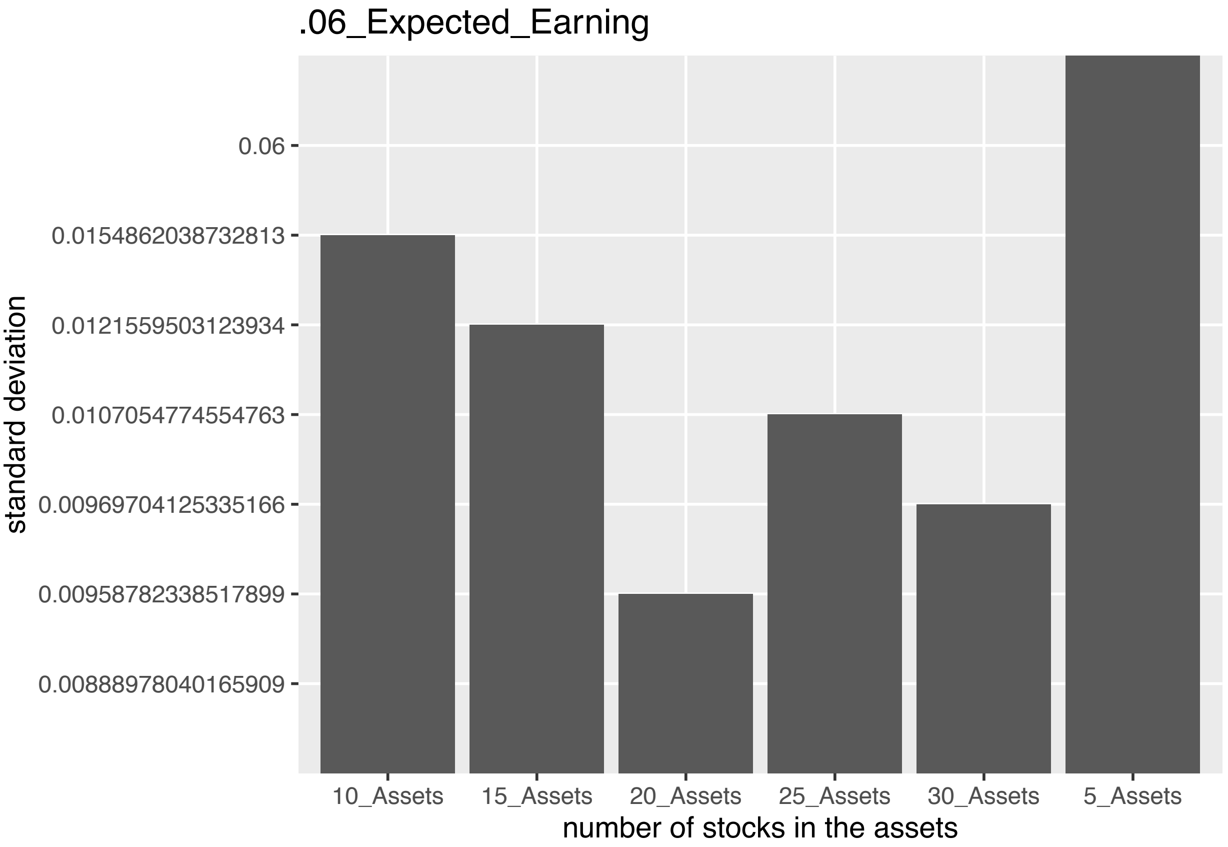
```
## [1] "The weight of optimal portfolio  4"
##          wt_fiserv wt_facebook    wt_intel     wt_apple wt_microsoft
## [1,] 0.0008781142  0.00476583 0.002724884 -0.007354983   0.01970415
## [2,] 0.0047363291  0.02570570 0.014697346 -0.039670946   0.10627925
## [3,] 0.0085945440  0.04664557 0.026669807 -0.071986908   0.19285436
## [4,] 0.0124527589  0.06758543 0.038642268 -0.104302871   0.27942947
##        wt_intuit wt_starbucks  wt_qualcomm  wt_gilead   wt_chater
## [1,] 0.003688245 -0.004708733 0.0006459151 0.01055212 0.008833055
## [2,] 0.019893477 -0.025397734 0.0034839050 0.05691550 0.047643299
## [3,] 0.036098708 -0.046086734 0.0063218948 0.10327888 0.086453543
## [4,] 0.052303939 -0.066775734 0.0091598847 0.14964226 0.125263787
##       wt_celgene   wt_amazon  wt_costco     wt_cisco  wt_tmobile
## [1,] 0.002277917 -0.01076127 0.01189111 0.0006848698 0.002404512
## [2,] 0.012286515 -0.05804359 0.06413769 0.0036940169 0.012969338
## [3,] 0.022295114 -0.10532591 0.11638427 0.0067031640 0.023534163
## [4,] 0.032303712 -0.15260823 0.16863084 0.0097123111 0.034098989
##         wt_nvidia    wt_adobe wt_broadcom wt_texasinstruments  wt_comcast
## [1,] -0.001096644 0.008480856 0.006863408         0.009823494 0.001778300
## [2,] -0.005915024 0.045743624 0.037019512         0.052985478 0.009591708
## [3,] -0.010733405 0.083006392 0.067175616         0.096147463 0.017405115
## [4,] -0.015551785 0.120269160 0.097331720         0.139309447 0.025218522
##       wt_rskfree
## [1,]  0.92792485
## [2,]  0.61124460
## [3,]  0.29456436
## [4,] -0.02211589
## [1] "The weight of optimal portfolio  4"
##             [,1]
## [1,] 0.001984793
## [2,] 0.010705477
## [3,] 0.019426162
## [4,] 0.028146847
## [1] "The best nu of optimal portfolio  5"
## [1] 3

## Warning: ^IRX contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.

## Warning: missing values removed from data

## [1] "The weight of optimal portfolio  5"
##        wt_fiserv wt_facebook    wt_intel    wt_apple wt_microsoft
## [1,] -0.01024983 0.001526624 0.004564706 -0.00650549  0.001072481
## [2,] -0.05528501 0.008234230 0.024620884 -0.03508899  0.005784693
## [3,] -0.10032019 0.014941836 0.044677063 -0.06367250  0.010496906
## [4,] -0.14535538 0.021649442 0.064733241 -0.09225600  0.015209119
##         wt_intuit wt_starbucks wt_qualcomm   wt_gilead     wt_chater
## [1,] -0.007372379   0.01115981 0.005848681 0.006470893 -0.0009299532
## [2,] -0.039764772   0.06019325 0.031546327 0.034902385 -0.0050159355
## [3,] -0.072157165   0.10922669 0.057243972 0.063333878 -0.0091019179
## [4,] -0.104549558   0.15826014 0.082941618 0.091765370 -0.0131879002
##        wt_celgene   wt_amazon   wt_costco   wt_cisco  wt_tmobile
## [1,] -0.003525625 0.007768401 0.008520378 0.01319592 0.001234474
## [2,] -0.019016343 0.041900818 0.045956794 0.07117551 0.006658443
## [3,] -0.034507061 0.076033235 0.083393210 0.12915510 0.012082412
## [4,] -0.049997779 0.110165651 0.120829626 0.18713469 0.017506382
##         wt_nvidia    wt_adobe  wt_broadcom wt_texasinstruments  wt_comcast
## [1,] -0.002505376 0.001608672 -0.004596726          0.01627385 0.004748321
## [2,] -0.013513376 0.008676774 -0.024793596          0.08777711 0.025611258
## [3,] -0.024521376 0.015744876 -0.044990466          0.15928036 0.046474195
## [4,] -0.035529375 0.022812979 -0.065187336          0.23078362 0.067337132
##      wt_automaticdata wt_alphabeta  wt_netflix wt_alphabetc   wt_booking
## [1,]       0.01786406   0.02813529 0.001155187  -0.02047537 -0.006880666
## [2,]       0.09635426   0.15175476 0.006230792  -0.11043907 -0.037112599
## [3,]       0.17484446   0.27537423 0.011306397  -0.20040276 -0.067344531
## [4,]       0.25333467   0.39899370 0.016382002  -0.29036645 -0.097576463
##      wt_rskfree
## [1,] 0.93189366
## [2,] 0.63265140
## [3,] 0.33340913
## [4,] 0.03416686
## [1] "The weight of optimal portfolio  5"
##             [,1]
## [1,] 0.001797829
## [2,] 0.009697041
## [3,] 0.017596254
## [4,] 0.025495466
## [1] "The best nu of optimal portfolio  6"
## [1] 3

## Warning: ^IRX contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.

## Warning: missing values removed from data
## [1] "The weight of optimal portfolio  6"
##        wt_fiserv  wt_facebook    wt_intel    wt_apple wt_microsoft
## [1,] 0.005768571 -0.006573374 0.002923185 0.000319776  0.003484457
## [2,] 0.031114231 -0.035455138 0.015766932 0.001724792  0.018794292
## [3,] 0.056459892 -0.064336903 0.028610679 0.003129808  0.034104126
## [4,] 0.081805553 -0.093218667 0.041454425 0.004534824  0.049413961
##         wt_intuit wt_starbucks  wt_qualcomm   wt_gilead   wt_chater
## [1,] -0.009529846 -0.002849572 -0.005674269 0.006754322 0.002525438
## [2,] -0.051401611 -0.015369882 -0.030605591 0.036431131 0.013621586
## [3,] -0.093273377 -0.027890193 -0.055536914 0.066107939 0.024717733
## [4,] -0.135145143 -0.040410503 -0.080468236 0.095784748 0.035813880
##       wt_celgene    wt_amazon  wt_costco   wt_cisco  wt_tmobile
## [1,] 0.001682197 -0.001725777 0.01932934 0.01281240 0.004883358
## [2,] 0.009073349 -0.009308411 0.10425766 0.06910689 0.026339614
## [3,] 0.016464502 -0.016891044 0.18918597 0.12540138 0.047795870
## [4,] 0.023855655 -0.024473678 0.27411429 0.18169586 0.069252126
##        wt_nvidia   wt_adobe wt_broadcom wt_texasinstruments  wt_comcast
## [1,] 0.004073353 0.00778746 -0.00365574        0.0007310388 0.002149027
## [2,] 0.021970650 0.04200362 -0.01971815        0.0039430410 0.011591318
## [3,] 0.039867947 0.07621978 -0.03578056        0.0071550431 0.021033609
## [4,] 0.057765245 0.11043594 -0.05184297        0.0103670453 0.030475900
##      wt_automaticdata wt_alphabeta    wt_netflix wt_alphabetc   wt_booking
## [1,]      0.006903179   0.08212542 -5.211117e-05  -0.07973113 -0.000663732
## [2,]      0.037234027   0.44296406 -2.810747e-04  -0.43004986 -0.003580005
## [3,]      0.067564875   0.80380269 -5.100381e-04  -0.78036859 -0.006496278
## [4,]      0.097895722   1.16464133 -7.390016e-04  -1.13068731 -0.009412551
##       wt_pepsico  wt_mondelez     wt_amgen wt_intuitivesurg     wt_csx
## [1,] 0.002731708 -0.008028903 -0.001137292    -0.0003287453 0.01025939
## [2,] 0.014734154 -0.043305900 -0.006134268    -0.0017731701 0.05533661
## [3,] 0.026736600 -0.078582898 -0.011131244    -0.0032175949 0.10041383
## [4,] 0.038739046 -0.113859895 -0.016128220    -0.0046620197 0.14549105
##      wt_rskfree
## [1,]  0.9427069
## [2,]  0.6909751
## [3,]  0.4392434
## [4,]  0.1875116
## [1] "The weight of optimal portfolio  6"
##             [,1]
## [1,] 0.001648163
## [2,] 0.008889780
## [3,] 0.016131398
## [4,] 0.023373015
```

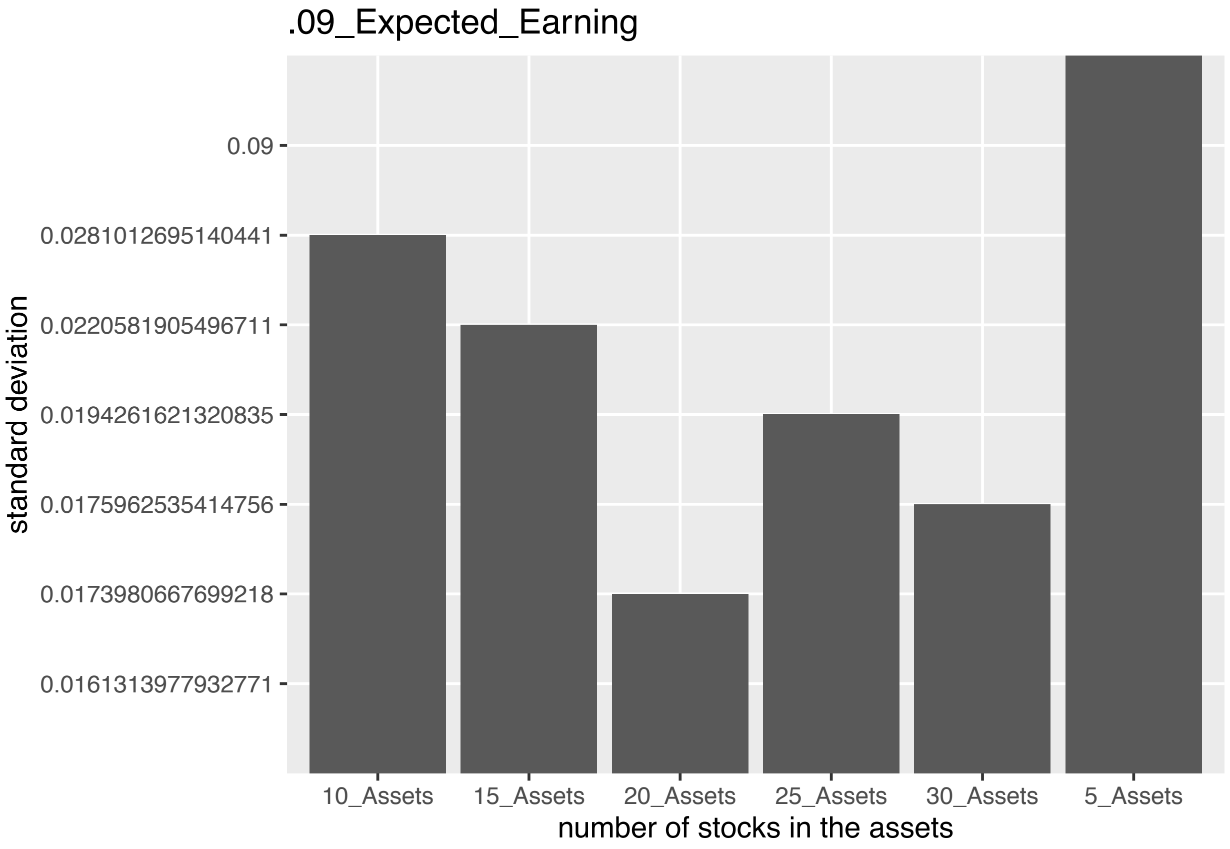
```
return_sd = t(return_sd)
rownames(return_sd) = NULL
c = c("5_Assets","10_Assets","15_Assets","20_Assets","25_Assets","30_Assets")
return_sd = cbind(c, return_sd)
```

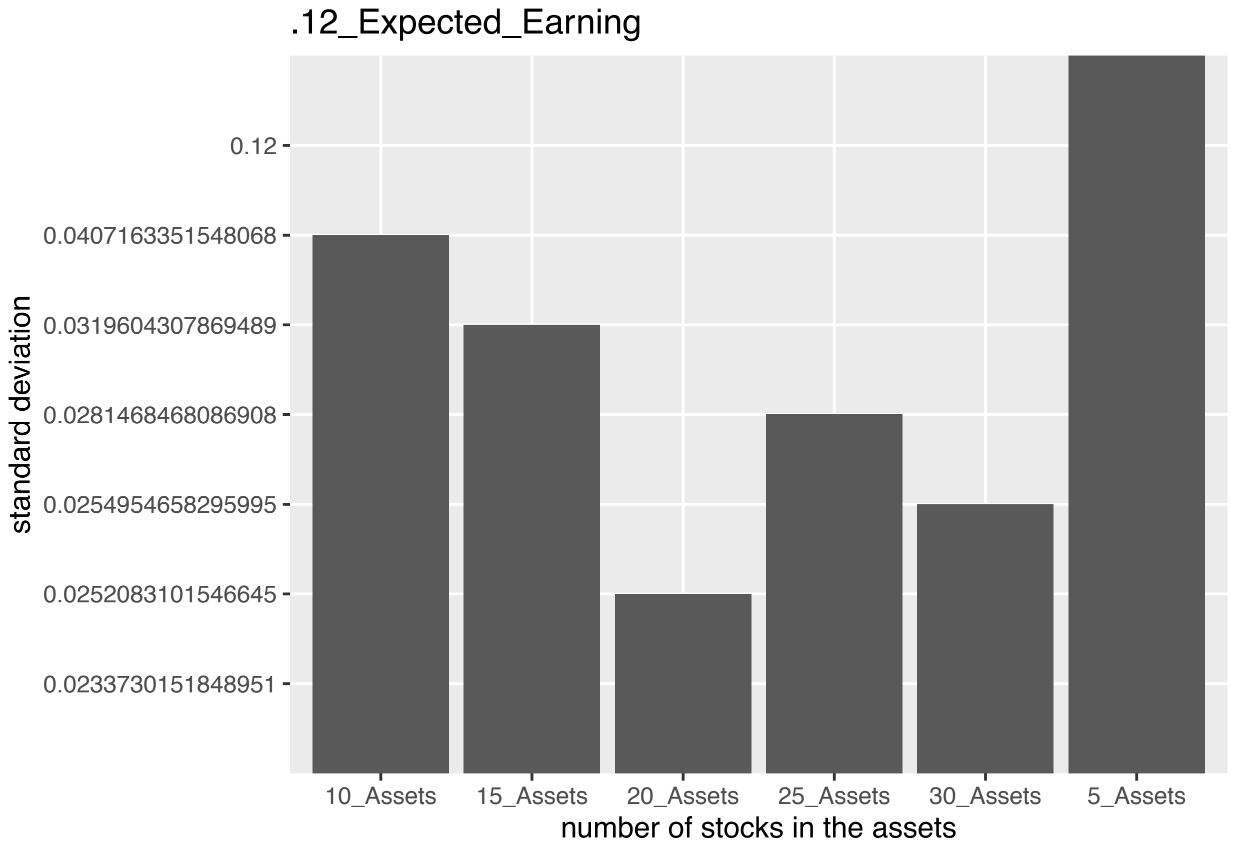
```
## Warning in cbind(c, return_sd): number of rows of result is not a multiple
## of vector length (arg 1)
```

```
plt_data = as.data.frame(return_sd)
colnames(plt_data) = c("Assets_Type",".03_Expected_Earning",".06_Expected_Earning",
  ".09_Expected_Earning",".12_Expected_Earning")    #rename colunms
for(i in 2:5){
    p = ggplot(data = plt_data,aes(x=Assets_Type,y=plt_data[,i]))
    p1 = p + geom_bar(stat = "identity") + labs(title = colnames(plt_data)[i])
    p2 = p1 + labs(x = "number of stocks in the assets",
                   y = "standard deviation")
    print(p2)    #plot
}
```




.06_Expected_Earning
standard deviation
0.06
0.0154862038732813
0.0121559503123934
0.0107054774554763
0.00969704125335166
0.00958782338517899
0.00888978040165909
10_Assets
15_Assets
20_Assets
25_Assets
30_Assets
5_Assets
number of stocks in the assets


.09_Expected_Earning
standard deviation
0.09
0.0281012695140441
0.0220581905496711
0.0194261621320835
0.0175962535414756
0.0173980667699218
0.0161313977932771
10_Assets
15_Assets
20_Assets
25_Assets
30_Assets
5_Assets
number of stocks in the assets



The best nu for the optimal portfolio with 5 stocks is 4. The optimal portfolio is: fiserv, facebook, intel, apple, microsoft, rskfree. The weights of each asset and the standard deviation of the optimal portfolio for each annual return level are shown as above.

The best nu for the optimal portfolio with 10 stocks is 3.8. The optimal portfolio is: fiserv, facebook, intel, apple, microsoft, qualcomm, gilead, adobe, netflix, starbucks, rskfree. The weights of each asset and the standard deviation of the optimal portfolio for each annual return level are shown as above.

The best nu for the optimal portfolio with 15 stocks is 3.8. The optimal portfolio is: fiserv, facebook, intel, apple, microsoft, qualcomm, gilead, adobe, netflix, starbucks, alphabetc, costco, chater, booking, tmobile, rskfree. The weights of each asset and the standard deviation of the optimal portfolio for each annual return level are shown as above.

The best nu for the optimal portfolio with 20 stocks is 3. The optimal portfolio is: fiserv, facebook, intel, apple, microsoft, qualcomm, gilead, adobe, netflix, starbucks, alphabetc, costco, chater, booking, tmobile, mondelez, texasinstruments, amazon, cisco, automaticdata, rskfree. The weights of each asset and the standard deviation of the optimal portfolio for each annual return level are shown as above.

The best nu for the optimal portfolio with 25 stocks is 3. The optimal portfolio is: fiserv, facebook, intel, apple, microsoft, qualcomm, gilead, adobe, netflix, starbucks, alphabetc, costco, chater, booking, tmobile, mondelez, texasinstruments, amazon, cisco, automaticdata, intuit, broadcom, comcast, csx, nvidia, rskfree. The weights of each asset and the standard deviation of the optimal portfolio for each annual return level are shown as above.

The best nu for the optimal portfolio with 30 stocks is 3. The optimal portfolio is: fiserv, facebook, intel, apple, microsoft, qualcomm, gilead, adobe, netflix, starbucks, alphabetc, costco, chater, booking, tmobile, mondelez, texasinstruments, amazon, cisco, automaticdata, intuit, broadcom, comcast, csx, nvidia, alphabeta, pepsico, celgene, intuitivesurg, amgen, rskfree. The weights of each asset and the standard deviation of the optimal portfolio for each annual return level are shown as above.

Tuna Market Share

1.Load the tuna data set from package bayesm. There are seven brands in the data set. For each brand, estimate separate independent student-t models where logsales for each product is regressed on an intercept, the product's log price and display activity. Use the default training sample prior and use log-marginal likelihoods to find the appropriate-degrees of freedom of the student-t distribution on a grid of 20 equally-spaced values between 3 and 6.

```
#extract data and specify models
skdf = extracttunabrand("sk")
modelsk = log(salessk)~log(pricesk)+dispsk
csdf = extracttunabrand("cs")
modelcs = log(salescs)~log(pricecs)+dispcs
bbdf = extracttunabrand("bb")
modelbb = log(salesbb)~log(pricebb)+dispbb
bbcdf = extracttunabrand("bbc")
modelbbc = log(salesbbc)~log(pricebbc)+dispbbc
gedf = extracttunabrand("ge")
modelge = log(salesge)~log(pricege)+dispge
bbldf = extracttunabrand("bbl")
modelbbl = log(salesbbl)~log(pricebbl)+dispbbl
hhcldf = extracttunabrand("hhcl")
modelhhcl = log(saleshhcl)~log(pricehhcl)+disphhcl
```

```
#generate a grid of 20 equally-spaced values between 3 and 6
nug = seq(from = 3,to = 6,by = .15)
```

```
#sk - estimate independent student-t model
outls = future_mapply("MCMCregresst",nu = nug,
                      MoreArgs = list(data = skdf,
                                      modelfrm = modelsk),
                      SIMPLIFY = FALSE)
A = cbind(nug,t(logmarglik(outls)))
colnames(A) = c("nu","logmarg")
ind = which.max(A[,2])
A1 = A[ind,,drop = F]
print(A[ind,,drop = F])
```

```
##      nu   logmarg
## [1,]  3 -176.7638
```

```
#cs - estimate independent student-t model
outls = future_mapply("MCMCregresst",nu = nug,
                      MoreArgs = list(data = csdf,
                                      modelfrm = modelcs),
                      SIMPLIFY = FALSE)
A = cbind(nug,t(logmarglik(outls)))
colnames(A) = c("nu","logmarg")
ind = which.max(A[,2])
```

```r
A2 = A[ind,,drop = F]
print(A[ind,,drop = F])
```

```
##         nu   logmarg
## [1,] 4.05 -212.5567
```

```r
#bb - estimate independent student-t model
outls = future_mapply("MCMCregresst",nu = nug,
                      MoreArgs = list(data = bbdf,
                                      modelfrm = modelbb),
                      SIMPLIFY = FALSE)
A = cbind(nug,t(logmarglik(outls)))
colnames(A) = c("nu","logmarg")
ind = which.max(A[,2])
A3 = A[ind,,drop = F]
print(A[ind,,drop = F])
```

```
##       nu   logmarg
## [1,]  3 -121.4391
```

```r
#bbc - estimate independent student-t model
outls = future_mapply("MCMCregresst",nu = nug,
                      MoreArgs = list(data = bbcdf,
                                      modelfrm = modelbbc),
                      SIMPLIFY = FALSE)
A = cbind(nug,t(logmarglik(outls)))
colnames(A) = c("nu","logmarg")
ind = which.max(A[,2])
A4 = A[ind,,drop = F]
print(A[ind,,drop = F])
```

```
##        nu   logmarg
## [1,] 3.3 -218.8886
```

```r
#ge - estimate independent student-t model
outls = future_mapply("MCMCregresst",nu = nug,
                      MoreArgs = list(data = gedf,
                                      modelfrm = modelge),
                      SIMPLIFY = FALSE)
A = cbind(nug,t(logmarglik(outls)))
colnames(A) = c("nu","logmarg")
ind = which.max(A[,2])
A5 = A[ind,,drop = F]
print(A[ind,,drop = F])
```

```
##        nu  logmarg
## [1,] 5.4 12.43656
```

```r
#bbl - estimate independent student-t model
outls = future_mapply("MCMCregresst",nu = nug,
                      MoreArgs = list(data = bbldf,
                                      modelfrm = modelbbl),
                      SIMPLIFY = FALSE)
A = cbind(nug,t(logmarglik(outls)))
colnames(A) = c("nu","logmarg")
ind = which.max(A[,2])
A6 = A[ind,,drop = F]
print(A[ind,,drop = F])
```

```
##      nu   logmarg
## [1,]  3 -100.2465
```

```
#hhcl - estimate independent student-t model
outls = future_mapply("MCMCregresst",nu = nug,
                      MoreArgs = list(data = hhcldf,
                                      modelfrm = modelhhcl),
                      SIMPLIFY = FALSE)
A = cbind(nug,t(logmarglik(outls)))
colnames(A) = c("nu","logmarg")
ind = which.max(A[,2])
A7 = A[ind,,drop = F]
print(A[ind,,drop = F])
```

```
##      nu   logmarg
## [1,]  3 -233.4531
```

```
result = rbind(A1,A2,A3,A4,A5,A6,A7)
rownames(result) = c("sk", "cs", "bb", "bbc", "ge", "bbl", "hhcl")
result
```

```
##        nu    logmarg
## sk   3.00 -176.76375
## cs   4.05 -212.55673
## bb   3.00 -121.43915
## bbc  3.30 -218.88859
## ge   5.40   12.43656
## bbl  3.00 -100.24651
## hhcl 3.00 -233.45310
```

The estimated separate independent student-t models for each brand are shown as above.

2.Now estimate a SURE student-t model for the seven brands. Again use marginal likelihoods to find the appropriate degrees of freedom on a grid of 20 equally-spaced values between 3 and 6. From your estimation results, which pair of products is the most correlated?

```
suremodel = list(modelsk,modelcs,modelbb,modelbbc,modelge,modelbbl,modelhhcl)
suredf = cbind(skdf,csdf,bbdf,bbcdf,gedf,bbldf,hhcldf)
outls = future_mapply("MCMCsuret",nu = nug,
                      MoreArgs = list(data = suredf,
                                      modelfrm = suremodel),
                      SIMPLIFY = FALSE)
```

```
A = cbind(nug,t(logmarglik(outls)))
colnames(A) = c("nu","logmarg")
ind = which.max(A[,2])
A = A[ind,,drop = F]
print(A[ind,,drop = F])
```

```
##      nu   logmarg
## [1,]  3 -1181.632
```

The best degree of freedom for the SURE student-t model is 3.

```
tunasuret = MCMCsuret(modelfrmls = suremodel,
                      data = suredf,nu = 3)
```

```
dd = attr(tunasuret, "dd")
k = attr(tunasuret, "k")
sigm = tunasuret[,(k+1):(k+dd)]
Sigma = apply(sigm,2,"mean")
Sigma = xpnd(Sigma)
rho = cov2cor(Sigma)
diag(rho) = rep(0,7)
cbind(c("sk", "cs", "bb", "bbc","ge","bbl","hhcl")[which.max(rho)%%7],
      c("sk", "cs","bb","bbc","ge","bbl","hhcl")[which.max(rho)%/%7+1])
```

```
##      [,1]  [,2]
## [1,] "bbl" "bb"
```

The most correlated pair of products is "bb" and "bbl".

3.Now suppose you are managing the sales of Star Kist 6 oz, and you want to know what price to charge for your product, given the other six other products in the market. Suppose your main competitor is Chicken of the Sea 6 oz and you would like to generate (on average) twice the total sales compared to Chicken of the Sea 6 oz. How would you determine your own price?

```
result2 = cbind()
psk = c(0.44,0.54,0.64,0.74,0.84)
for (i in 1:5){
    b = cbind(0,psk[i],0.31,0,0.70,0.35,0,1.80,0.29,0,0.85,
              0.23,0,1.40,0.35,0,3.49,0.25,0,0.75,0.24)
    predf1 = data.frame(b)
    colnames(predf1) = names(suredf)
    pre1 = predictsuret(thetam = tunasuret,pdatdf = predf1,
                        nu = 3,logr = F)
    pre1 = pre1[[1]]
    sk = exp(mean(pre1[1,]))
    cs = exp(mean(pre1[2,]))
    ratio = (sk)/(cs)
    temp = c((sk),(cs),ratio)
    result2 = rbind(result2,temp)
}
rownames(result2) = c(0.44,0.54,0.64,0.74,0.84)
result2
```

```
##            [,1]     [,2]      [,3]
## 0.44 106955.90 13211.27 8.0958064
## 0.54  50980.85 13100.32 3.8915719
## 0.64  27266.80 13132.63 2.0762632
## 0.74  16517.91 13425.93 1.2302991
## 0.84  10083.01 13294.23 0.7584505
```

I would determine Star Kist price as 0.44, which is expected to be the best among the prices given.