final assignment

global

we firstly used `>Rscript getdata.R` to get 'dat.csv' from 'data_cn/tickers.csv' , 'data_cn/adjusted.csv' and 'data_cn/in_univ.csv' , all what we use is in 'dat.csv' , later all the code is start from dat.csv . during this step , we remove those don't have Gics code from our universe.

we write all the functions to modify data in 'calculate.r'

we write all the `library()` in 'library.r'

part A

1.

we first choose a specific year t . Then we remove those stocks don't exist in this year , for 21_day_lookback factor , we chose the timeseries from 21days before that year

we write this step in 'yeardat.r' , as an example , we choose 2018 and the data is saved in test18.csv .

2.

we firstly write some functions in file 'calculate.r'

- `app(dat,tau,f)` , apply function `f` to past `tau` days in every time series in `dat` . use the result to constructure a data.frame derive the timesreies in this year
- `logreturn()` to change the data in the specific year to daily log return .
- `minus()` subtract out the industry mean
- `mat()` change the data.frame to matrix purely derive values(don't contain Gics , date , etc.)
- `ran()` return the rank :

$$(2 * rank - N - 1)/(N - 1)$$

then we have these to get v :

```
#get the factor v
#grt the daily logreturn , we change NA to 0 here
logr = logreturn(year)

# a)
#get the 10-day-logreturn
logr10 = app(logr,10,mean)

# b)
#subtract out the industry return
logr_idu = minus(logr10)

# c)
#calculate the volatility
#and then divide the 10-day return by the volatility
sd21 = app(logr,21,sd)
v = mat(logr_idu)/mat(sd21)

# d
#rank the normalized factor)
v = ran(v)
```

3.

similar to 2. we have :

```
# a)
#we already get that in 2.
#logr is what we need

# b)
#subtract out the industry return
logrm_idu = minus(logr)

# c)
#get the maximum of the magnitude of the daily return
max21 = app(logrm_idu,21,max)

# d)
#rank the factor
m = ran(mat(max21))
```

4.

we add a function to 'calculate.r'

- `regr(a,b,c)` , `a,b,c` are all $i \times j$ matrix this function return the $(i-1) \times 2$ matrix $\beta = (\beta_{i,b}, \beta_{i,c})$

then this step is:

```
#calculate the daily betas
r_midu = app(logrm_idu,1,max)
beta = regr(mat(r_midu),v,m)
beta_v = beta[1,]
beta_m = beta[2,]
```

5.

we add a function to 'calculate.r'

- `ts(x)` return the t-stat of x

then this step is:

```
#calculate the 1-year average and the t-stat of beta
and the t-stat
mbm = mean(beta_m)
tsv = ts(beta_v)
tsm = ts(beta_m)
```

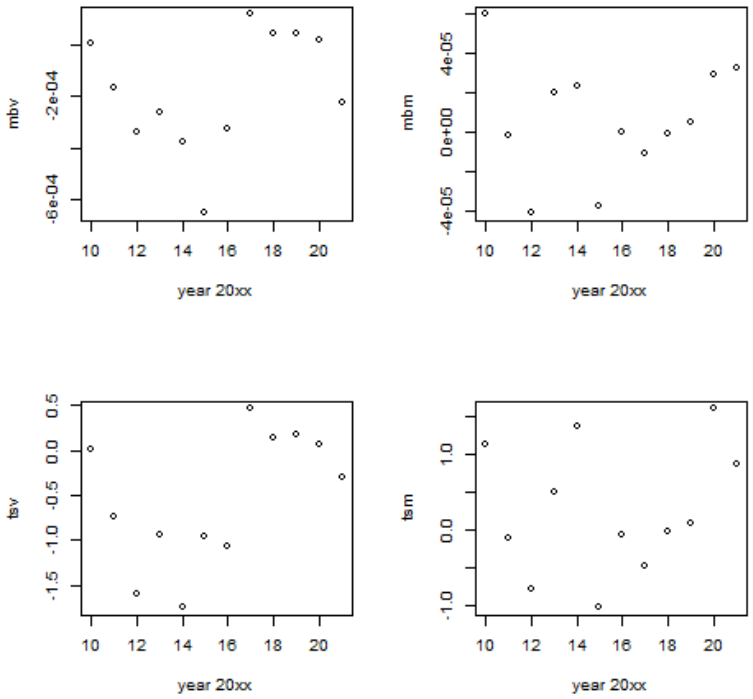
result

we do a loop to apply the steps to every year , and save the results together to 'beta.csv' , the that is:

	tlist	mbv	mbm	tsv	tsm
1	10	3.17526016374988e-06	5.94493012403889e-05	0.00904250852186723	1.12802009177008
2	11	-0.000169546405459324	-1.40568680799672e-06	-0.743844427732674	-0.105738889544941
3	12	-0.000337121878928585	-4.04223069943161e-05	-1.59739033590372	-0.787522052839338
4	13	-0.000263426853392668	1.99758178983862e-05	-0.944509292501053	0.485462700770483
5	14	-0.000378046497416255	2.34740130451825e-05	-1.73858632734512	1.35009082193967
6	15	-0.000646674391910587	-3.70838168829258e-05	-0.959337869213977	-1.02134623746244
7	16	-0.000322140793892591	-2.30772652374582e-07	-1.06358126806893	-0.0656071719105106
8	17	0.000116431689160235	-1.04917971054775e-05	0.46542267123334	-0.491093673168217
9	18	4.29763623128189e-05	-1.09755671884696e-06	0.144642150856474	-0.037695401968076
10	19	4.10803934318818e-05	4.78439271915337e-06	0.177094367562872	0.0818549889870174
11	20	1.76587028826167e-05	2.92763675704891e-05	0.0639050539645888	1.59132242966391
12	21	-0.00022108257125467	3.26849885416974e-05	-0.297899566343352	0.85074801546716

we can also plot

```
png('beta.png')
p = read.csv("beta.csv")

par(mfrow = c(2,2))
for(i in 1:4){
    names(p)[i+2]
    plot(p[,2],p[,i+2],ylab = names(p)[i+2])
}
dev.off()
```



the script this part is saved as 'parta.r' , the input data it used is 'dat.csv' , the out put files are : "beta.csv" and "beta.png"

part B

we add a function to 'calculate.r'

- `expect(b,v,m)`

$$b[1] * v + b[2] * m$$

- `change()`

$$f(n) = \begin{cases} -1, & \text{if } n < \text{-}0.6 \\ 1, & \text{if } n > 0.6 \\ 0, & \text{else} \end{cases}$$

- `weight(re)` , `apply(re,change)`
- `fact()`

$$fact\_logreturn = ln(1 + \frac{\sum_i w_i(\exp(logreturn_i) - 1)}{\sum_i \frac{1}{2}|w_i|})$$

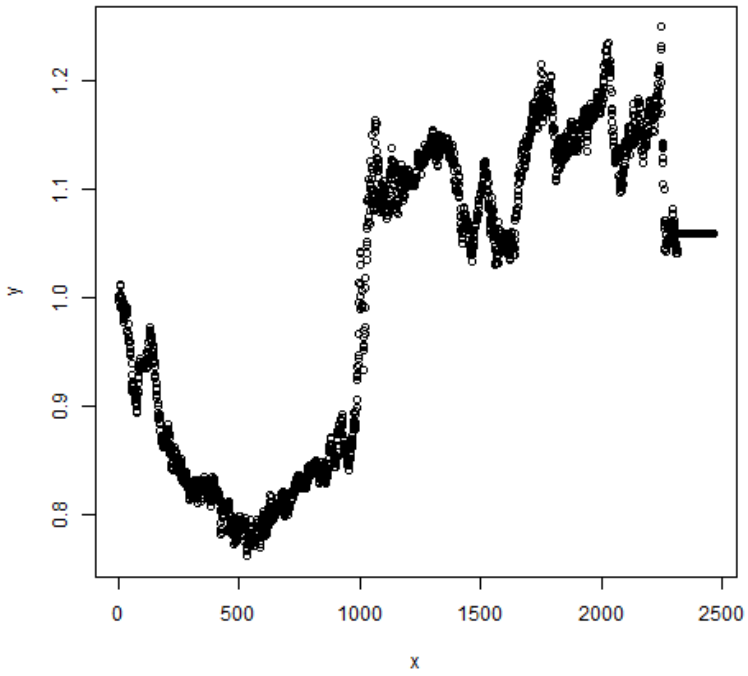
```
# 1.
#construct portfolio
re = expect(beta,v,m)
w = weight(re)

# 2.
#get the portfolio return at each time step t.
rf = fact(w,mat(logr))
```

the daily logreturn is saved in 'dailyreturn.csv' , yearly result saved in 'return.csv'

	tlist	logr_year	sd_year	sharp
1	10	0	0	0
2	11	-0.166755926528585	0.00471778231250174	-0.158503376027637
3	12	-0.0455564234791016	0.00425416779874613	-0.0484554612213748
4	13	-0.00619289657519553	0.00561371661011681	-0.00510727909315688
5	14	0.0488126400990426	0.00394187299454885	0.0555296335215965
6	15	0.252851014188804	0.0102781907104545	0.110814107703167
7	16	0.0476927788633998	0.00493277799642757	0.0435519979569541
8	17	-0.0925000505466106	0.00510015082349255	-0.0816969750060785
9	18	0.148305859081358	0.00511933457318767	0.131084856062533
10	19	0.00104325024132195	0.00472917531382879	0.000993688137891325
11	20	0.000455372486962358	0.00575956359550823	0.000357754360484334
12	21	-0.130066218388905	0.0106574661651426	-0.162723129336079

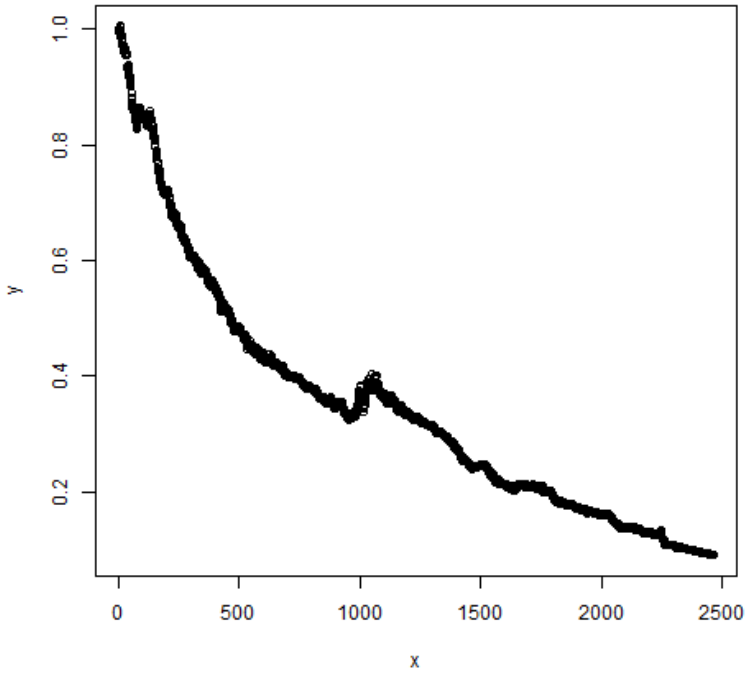
from here we know that the best year is 2015 , and the worst is 2011 .the the annual return (annually logreturn) volatility of the portfolio is `sd(read.csv('return.csv')[,3])` : 0.115089054444508

we also use function `pnl()` to plot the total PnL:



part C

when considering cost model , we simply minus 0.001 from the everyday-log-return and get the part-B-PnL considering cost (trade every day,buy cost 0.0005,sell cost 0.0005, 0.0005 << 1 ,so simply minus 0.001) `pnl(dailyreturn-0.0005,"pnlcost.png")`



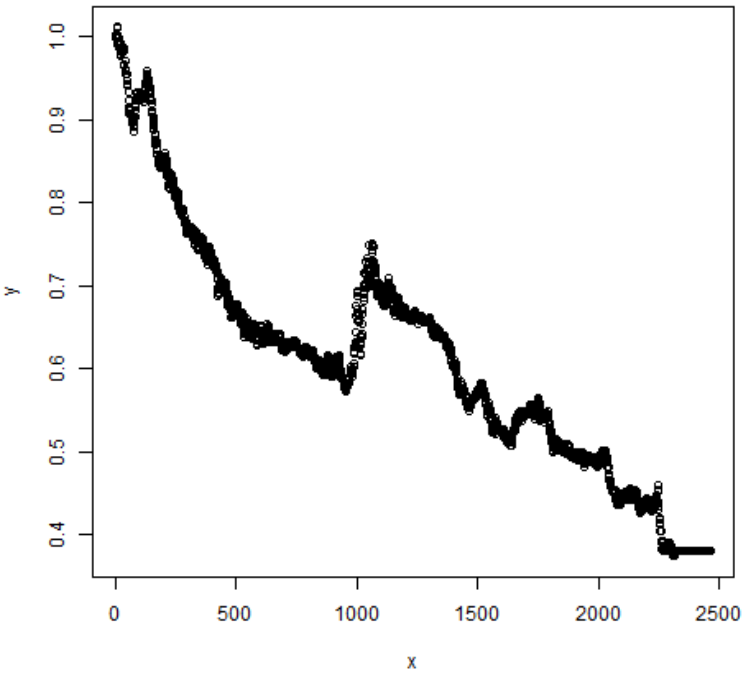
then we considering hold some position longer then one day , which means the daily log return cost is less than 0.001 (if we hold A today , and tomorrow , it doesn't make cost)

so we change the daily logreturn fomula , from $ln(1+\frac{\sum_i w_i(\exp(logreturn_i)-1)}{\sum_i \frac{1}{2}|w_i|})$ to:

$$fact\_logreturn = ln(1 + \frac{\sum_i w_i(\exp(logreturn_i) - 1)}{\sum_i \frac{1}{2}|w_i|} - \frac{\sum_i |\delta w_i| * 0.0005}{\sum_i \frac{1}{2}|w_i|})$$

we add function `factc()` to 'calculate.r' , and use it instead of fact , to get the modified resullt of part B. this is in 'partbc.r'

and the PnL is :

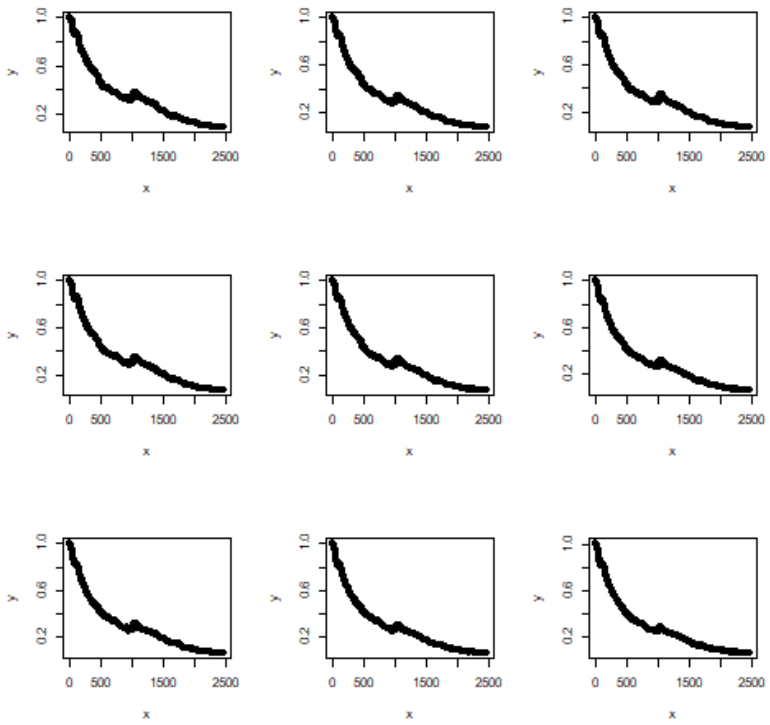


this picture beeing much better than above is because this item $-\frac{\sum_i |\delta w_i| * 0.0005}{\sum_i \frac{1}{2}|w_i|}$ take the place of minusing 0.001 directly.

to if $\sum_i |\delta w_i|$ get less , the PnL will be better , so we import a hurdle to modify the expcted return :

after import hurdle , we make the decision that if we long stock$_i$ today and want to long it tomorrow , we would holde it , instead of selling and buying again , to save the trading cost.

we choose γ from {1.1,1.2,...,1.9} while $h = \gamma * \sigma$ and here is the 9 PnLs:



the first 2 photo in partC means `factc()` works well , but the 9 PnLs means that different hurdles made little differences on PnL , so I guess the function `weightc()` may have something wrong , but i didn't find it .

part D

a) contribution to the project

the outline of the whole project is desigined by me , at the wery begining , i write an Abstract in Chinese containing all the math fomula and then we translate it together to the form we upload .

before the coding process , to make the cooperation easier , i wrote a detailed task.md file to illustrate what should every body do , as well as the form of intermediate-step-file . i thought that this could help people start the later part of the whole project easily , after all , without data , they can know the data form from that file . you can refer it at the bottom of this file.

during the coding process , i found that the early part and the later part both didn't meet the requirement of our every-onr-tesk , none of them can offer me a standard data file or the script . they didn't give me the monthly industry return , the factors they didn't modified successfully , there wasn't r.csv or e.csv either . so i need to adjust the plan near the ddl. and i have to pick up all the troubles and nearly do the whole works again.

for the strategy part , i communicate with them and tell them which strategy may be useful.

during the report writting and ppt making part , i write our regrassion part . How we get the α , β ,and why.

b) difficulty

i think coding and debug is the most difficult part for me , especially when there must be some cooperation . coding alone is really different from coding with each other , though i've tried a lot to make the cooperation being more smooth , it's still bother me a lot . even cooperation with myself from different time makes a lot trouble.

before coding line by line , i thought that the difficult part is drawing the outline , dividing the whole project into specific tesk , writting the math behind the model , even writting the pseudocode . however , infact , the time transforming the pseudocode into real code is the same as summing all other together .

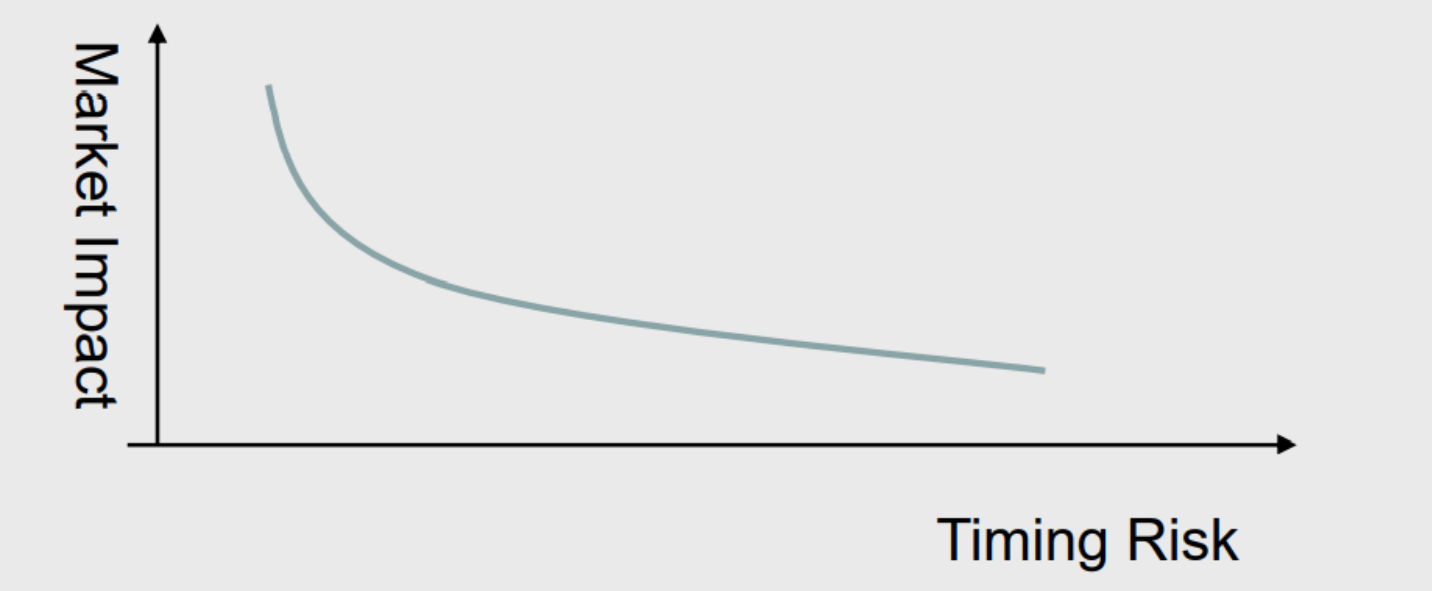
the 3 assignment contain the same part . in principle , we can use the earlier one to solve the later one . however , because the lack of experience , i write the first one by pyhton , the second one R , when i want to use the second one to do the last one , i find i just write all the code in a single file , a single function , that is because i don't know the function `source()` in R . To make my final work more readable , i have to write all the work again.

reading my own early work can bother me that much , no matter how difficult it is to read the others. during the cooperation process , i find that reality is really different from what i thought.

at last , after this introspection , i find all my difficulty is technologically instead of theoretically . i know after 1~2 years training i can overcome all this technological problem , but i also strongly wander if there is a shortcut.

c) most interesting

i think the cost are the most interesting part , especially this photo:



the qualitative analysis is obvious , but the quantitative behand it is mysterious , mabe people have some empirical formula , but we still don't know the dynamic principle details . i think this part is interesting .

d)

i think it's not difficult at all for me to understand the topics . the logic and math behind them is intuitionistic.

---

next i will offer some refference:

- tesk.md
- functions : calculate.r
- script for each part : parta.r partb.r partbc.r partc.r

---

tesk.md :

specific task for every one

liyi and zhutianhang(2.1 in the abstract)

you two need to upload 4 files intotal , two of them will influence others' work , the other two do nothing with other members , but necessary for our final report.

return.csv

in this csv , we need summarized industry return , it's not the original price of every socks, but the logreturn every day you calculated for each industry:

label	industrycode	20070103	20070104	...	20210528
1	101020	0.01	-0.01	...	0
...	...	...	...	0	
J	601020				

that is a ((J+1) $\times$ (T+1))csv , J means there are J industries , T means there are T trading days .

factor.csv

in this file there should be I independent factors:

label	factornames	20070103	20070104	...	20210528
1	risk-free-rate	0.01	-0.01	...	0
...	...	...	...		
I	No.I factor's name				

that is a ((I+1) $\times$ (T+1))csv , I means there are I factors , T means there are T trading days .

other files

beside these two files you need to offer the original data you used in one file (you can name it 1.csv or any thing you want.) and the code you got the upside 2 file in another file(mabe called 1.py or 1.R), this 2 file don't influence other's work directly so the format is not necessary to determined.

liyusu(2.2 in the abstract)

i need to up load 3 files in total , first 2 is used in other part , the 3rd is code.

alpha.csv

the format is the same as price.csv . the only thing different is the value change from logreturn to alpha[j,t] , but because of our definetion , we dont have R(t) where t $\in$ [T,T+Δt] , so the max of the time label of alpha is T-Δt

beta.csv

label	factornames	20070103	20070104	...	20210528 -Δt

label	factornames	20070103	20070104	...	20210528 -Δt
1	$\beta_{1,1}$	0.01	-0.01	...	0
2	$\beta_{1,2}$	0.01	-0.01	...	0
...					
(i-1)*j+j	$\beta_{i,j}$				
...					
I * J	$\beta_{I,J}$				

here alpha and beta is little bit different from the abstract , each column is what we want in abstract . and the column "t1" will have the same value as _t1 _ Δt.csv in the abstract.

regrassion.R

```
>ls
price.csv       return.csv
>Rscript regrassion.R
>ls
price.csv       return.cs       alpha.csv       beta.csv
```

all of us(2.3 in the abstract)

port.R

the input of port.R is a time ,for example '20210303'

```
>ls
alpha.csv       beta.csv
>Rscript port.R 20210303
>ls
alpha.csv       beta.csv        weight.csv
```

weight.csv:

label	industry	20210303
1	101020	0.02
...	...	...
j	industry$_j$	w_j
...	...	...
J		

xuyuxin and xiaoyumeng(2.4)

you need to upload 2 files r.csv and e.csv:
r.csv:

name	t1	...	t10
liyi	3.7	...	4.8
...			

in which t_i is a random time belongs to our datset 3.7 is the total log return calculate from liyi's w_i using time input t_i .
the format of e.csv is the same as r.csv.

only different is taht the value is residuals instead of return.
you'd better also upload your code and conclussion.

---

calculate.r

```
#dat = read.csv('test18.csv')
#dat = dat[,2:dim(dat)[2]]

logreturn = function(dat){
    #dat = read.csv('test18.csv')
    #dat = dat[,2:dim(dat)[2]]
    i = dim(dat)[1]
    j = dim(dat)[2]
    mat = as.matrix(dat[4:i,3:j])
    mat = log(mat)
    mat1 = mat[2:(i-3),] - mat[1:(i-4),]
    logr = dat
    logr[5:i,3:j] = mat1
    logr[is.na(logr)] = 0
    return (logr)
}

app = function(dat,x,f){
    #dat = read.csv('logr18.csv')
    #dat = dat[,2:dim(dat)[2]]
    i = dim(dat)[1] - 24
    j = dim(dat)[2] - 2
    tem = array(0,dim = c(x,i,j))
    mat = as.matrix(dat[4:(i+24),1:j+2])
    for (i1 in 1:i) {
       tem[,i1,] = mat[(i1-x+1):i1+21,]
    }
    mat1 = apply(tem , c(2,3) , f)
    dat[1:i + 24,1:j + 2] = mat1
    dat1 = dat[c(1:3,25:(i+3)),c(1,3:(j+2))]
    return (dat1)
}


mat = function(dat){
    i = dim(dat)[1]
    j = dim(dat)[2]
    mat1 = as.matrix(dat[4:i,3:j])
    return(mat1)
}

mata = function(dat){
    i = dim(dat)[1]
    j = dim(dat)[2]
    mat1 = as.matrix(dat[4:i,2:j])
    return(mat1)
}



#minus()
minus1 = function(logr){
    tem = t(logr)
    tem = as.data.frame(tem[3:dim(tem)[1],])
    names(tem) = t(logr)[1,]
    tem$indu = as.numeric(tem$Gics) %/% 100
    gda <- group_by(tem, indu)
    industry = as.data.frame(summarise(gda))
    for(i in 1:dim(industry)[1]){
        indi = (tem$indu == industry[i,])
        industryi = as.matrix(tem[indi,(5:dim(tem)[2]-1)])
        industryi = apply(industryi,c(1,2),as.numeric)
        industryi = industryi - apply(industryi,2,mean)
        tem[indi,(5:dim(tem)[2]-1)] = industryi
    }
    result = as.data.frame(t(tem))
```

```
    result = apply(result,c(1,2),as.numeric)
    re = logr
    re[,3:dim(re)[2]] = result[2:dim(result)[1]-1,]
    return(re)
}

minus = function(logr){
    tem = t(logr)
    tem = as.data.frame(tem[3:dim(tem)[1],])
    names(tem) = t(logr)[1,]
    tem$indu = as.numeric(tem$Gics) %/% 100
    gda <- group_by(tem, indu)
    industry = as.data.frame(summarise(gda))
    m = mat(logr)
    row = as.matrix(logr[2,3:dim(logr)[2]])
    for (i in 1:dim(industry)[1]){
        indi = ((row%/%100) == industry[i,])
        industryi = as.matrix(m[,indi])
        industryi = industryi - apply(industryi,1,mean)
        m[,indi] = industryi
    }
    i = dim(logr)[1]
    j = dim(logr)[2]
    logr[4:i,3:j] = m
    return(logr)
}



ran = function(mat){
    k = t(apply(mat, 1, rank))
    n = dim(mat)[2]
    v = (2*k-n-1)/(n-1)
    return(v)
}

regr = function(mat0,mat1,mat2){
    v = mat1
    m = mat2
    lr = mat0
    t = dim(mat0)[1]
    k = dim(mat0)[2]
    beta = array(0,dim = c(2,t-1))
    for (i in 2:t-1) {
        x = cbind(v[i,],m[i])
        y = as.matrix(lr[i+1,])
        a = t(x) %*% x
        if(det(a)==0){
            x = v[i,]+m[i]
            a = sum(x*x)
        }
        b = t(x) %*% y

        beta[,i] = solve(a,b)
    }
    return(beta)
}

t_s = function(a){
    T = length(a)
    t_s = sqrt(T) * mean(a) / sd(a)
    return(t_s)
}

expect = function(b,v,m) {
    return(b[1,1] * v + b[1,2] * m)
```

```
}

change = function(n){
    if (n < - 0.6) {
       return(-1)
    }
    else if (n >0.6) {
       return(1)
    } else {
       return(0)
    }
}


changec = function(n){
    if(n > 0.6){
        return(1)
    } else {
        return(0)
    }
}

weight = function(re){
    re = ran(re)
    return(apply(re,c(1,2),change))
}

weightc = function(re,gama){
    w = re * 0
    j = dim(re)[2]
    t = dim(re)[1]
    w0 = 1:j * 0
    for (i in 1:t){
        sd = sd(re[i,])
        h = sd*gama
        x = t(as.matrix(re[i,] + h * (1+w0)))
        y = t(as.matrix(re[i,] - h * (1+w0)))
        dim(x) = c(1,length(x))
        dim(y) = c(1,length(y))
        kx = ran(x)
        ky = ran(-y)
        wx = apply(kx,c(1,2),changec)
        wy = apply(ky,c(1,2),changec)
        w0 = wx - wy
        w[i,] = w0
    }
    return(w)
}

fact = function(w,mat){
    t = dim(w)[1]-1
    t1 = dim(mat)[1]
    mat1 = mat[1:t+t1-t,]
    w1 = w[1:t+1,]
    fact = w1 * (exp(mat1)-1)
    fact = apply(fact,1,sum)
    fact = log(1+fact*2/apply(abs(w1),1,sum))
    return(fact)
}

pnl = function(logr,name){
    t = length(logr)
    x = 0:t
    y = array(1,dim = c(1,t+1))
    for(i in 1:t){
        y[i+1] = y[i] * exp(logr[i])
```

```
    }
    png(name)
    plot(x,y)
    dev.off()
}

factc = function(w,mat){
    t = dim(w)[1]-1
    t1 = dim(mat)[1]
    mat1 = mat[1:t+t1-t,]
    w1 = w[1:t+1,]
    fact = w1 * (exp(mat1)-1)
    fact = apply(fact,1,sum)
    sumw = 0.5 * apply(abs(w1),1,sum)
    sumd = apply(abs(w1-w[1:t,]), 1, sum)
    fact = log(1 + fact/sumw - 0.0005*sumd/sumw)
    return(fact)
}
```

---

parta.r

```r
#used data : 'dat.csv'
#used function files : 'yeardat.r' 'calculate.r'
#used packages : 'library.r'

#set up evironment
source('library.r')
source('yeardat.r')
source('calculate.r')

#read data
dat = read.csv('dat.csv')

#initial the yearly betameans and their t-states
tsv = 10:21 * 0
tsm = tsv
tlist = 10:21
mbv = tsv
mbm = tsv

#for every year from 2010 Jan to 2021 May
for (i in 1:12) {
   #choose a year
   t = tlist[i]

   #find this years' data
   year = yearly(dat , t)

   #get the factor v
   logr = logreturn(year)
   logr10 = app(logr,10,mean)
   logr_idu = minus(logr10)
   sd21 = app(logr,21,sd)
   v = mat(logr_idu)/mat(sd21)
   v = ran(v)

   #get the factor m
   logrm_idu = minus(logr)
   max21 = app(logrm_idu,21,max)
   m = ran(mat(max21))

   #calculate the daily betas
   r_midu = app(logrm_idu,1,max)
   beta = regr(mat(r_midu),v,m)
   beta_v = beta[1,]
   beta_m = beta[2,]

   #save the means and the t-states
   mbv[i] = mean(beta_v)
   mbm[i] = mean(beta_m)
   tsv[i] = t_s(beta_v)
   tsm[i] = t_s(beta_m)
}

#save the result to a .csv file 'beta.csv'
t = as.data.frame(cbind(tlist,mbv,mbm,tsv,tsm))
write.csv(t,'beta.csv')

png('beta.png')
p = read.csv("beta.csv")

par(mfrow = c(2,2))
for(i in 1:4){
    names(p)[i+2]
    plot(p[,2],p[,i+2],ylab = names(p)[i+2],xlab = 'year 20xx')
}
dev.off()
```

partb.r

```r
#used data : 'dat.csv'
#used function files : 'yeardat.r' 'calculate.r'
#used packages : 'library.r'

#set up evironment
source('library.r')
source('yeardat.r')
source('calculate.r')

#read data
dat = read.csv('dat.csv')
beta = read.csv('beta.csv')

#initial the yearly betameans and their t-states
logr_year = 10:21 * 0
sd_year = 10:21 * 0
sharp  =  10:21 * 0
tlist = 10:21
rf_day = array(0,dim = c(300,12))

#for every year from 2010 Jan to 2021 May
for (i in 11:21 - 9) {
    #choose a year
   t = tlist[i]

   #find this years' data
   year = yearly(dat , t)

   #get the factor v
   logr = logreturn(year)
   logr10 = app(logr,10,mean)
   logr_idu = minus(logr10)
   sd21 = app(logr,21,sd)
   v = mata(logr_idu)/mata(sd21)
   v = ran(v)

   #get the factor m
   logrm_idu = minus(logr)
   max21 = app(logrm_idu,21,max)
   m = ran(mata(max21))

   #calculate the daily logreturn in fact
   beta = read.csv('beta.csv')[i-1,3:4]
   re = expect(beta,v,m)
   w = weight(re)
   rf = fact(w,mat(logr))


   #save the means and the t-states
    sd_year[i] = sd(rf)
    logr_year[i] = sum(rf)
    sharp[i] = mean(rf)/sd(rf)
    rf_day[1:length(rf),i] = rf
}

#save the result to a .csv file
fact = as.data.frame(cbind(tlist,logr_year,sd_year,sharp))
write.csv(fact,'return.csv')
write.csv(as.data.frame(rf_day),'dailyreturn.csv')
```

partbc.r

```r
#used data : 'dat.csv'
#used function files : 'yeardat.r' 'calculate.r'
#used packages : 'library.r'

#set up evironment
source('library.r')
source('yeardat.r')
source('calculate.r')

#read data
dat = read.csv('dat.csv')
beta = read.csv('beta.csv')

#initial the yearly betameans and their t-states
logr_year = 10:21 * 0
sd_year = 10:21 * 0
sharp  =  10:21 * 0
tlist = 10:21
rf_day = array(0,dim = c(300,12))

#for every year from 2010 Jan to 2021 May
for (i in 11:21 - 9) {
    #choose a year
   t = tlist[i]

   #find this years' data
   year = yearly(dat , t)

   #get the factor v
   logr = logreturn(year)
   logr10 = app(logr,10,mean)
   logr_idu = minus(logr10)
   sd21 = app(logr,21,sd)
   v = mata(logr_idu)/mata(sd21)
   v = ran(v)

   #get the factor m
   logrm_idu = minus(logr)
   max21 = app(logrm_idu,21,max)
   m = ran(mata(max21))

   #calculate the daily logreturn in fact
   beta = read.csv('beta.csv')[i-1,3:4]
   re = expect(beta,v,m)
   w = weight(re)
   rf = factc(w,mat(logr))


   #save the means and the t-states
    sd_year[i] = sd(rf)
    logr_year[i] = sum(rf)
    sharp[i] = mean(rf)/sd(rf)
    rf_day[1:length(rf),i] = rf
}

#save the result to a .csv file
fact = as.data.frame(cbind(tlist,logr_year,sd_year,sharp))
write.csv(fact,'returnc.csv')
write.csv(as.data.frame(rf_day),'dailyreturnc.csv')


dat = as.matrix(read.csv('dailyreturnc.csv')[1:224,3:13])
dim(dat) = c(224*11,1)
pnl(dat,'pnlcostc.png')
```

partc.r

```
#used data : 'dat.csv'
#used function files : 'yeardat.r' 'calculate.r'
#used packages : 'library.r'

#set up evironment
source('library.r')
source('yeardat.r')
source('calculate.r')

#read data
dat = read.csv('dat.csv')
beta = read.csv('beta.csv')

#initial the yearly betameans and their t-states
#logr_year = 10:21 * 0
#sd_year = 10:21 * 0
#sharp  =  10:21 * 0
tlist = 10:21
glist = 11:19/10
rf_day = array(0,dim = c(300,12,9))

for(gama in 1:9){
    g =glist[gama]
    #for every year from 2010 Jan to 2021 May
    for (i in 11:21 - 9) {
        #choose a year
        t = tlist[i]

        #find this years' data
        year = yearly(dat , t)

        #get the factor v
        logr = logreturn(year)
        logr10 = app(logr,10,mean)
        logr_idu = minus(logr10)
        sd21 = app(logr,21,sd)
        v = mata(logr_idu)/mata(sd21)
        v = ran(v)

        #get the factor m
        logrm_idu = minus(logr)
        max21 = app(logrm_idu,21,max)
        m = ran(mata(max21))

        #calculate the daily logreturn in fact
        beta = read.csv('beta.csv')[i-1,3:4]
        re = expect(beta,v,m)
        w = weightc(re,g)
        rf = factc(w,mat(logr))


        #save the means and the t-states
        rf_day[1:length(rf),i,gama] = rf
    }

}

png("pnl9.png")
par(mfrow = c(3,3))
for(i in 1:9){
    logr = rf_day[1:224,2:12,i]
    dim(logr) = c(224*11,1)
    t = length(logr)
    x = 0:t
    y = array(1,dim = c(1,t+1))
    for(i in 1:t){
```

```
        y[i+1] = y[i] * exp(logr[i])
    }
    plot(x,y)
}
dev.off()
```