STAT3799 SVM

YANG Yunqian

4 October 2020

#Data Processing

```
setwd('/Users/yangyunqian/Desktop/STAT3799/HKdata')
data_dir='/Users/yangyunqian/Desktop/STAT3799/HKdata'
library(TTR)
library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
## 
## Attaching package: 'zoo'
```

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

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

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

```
library(rvest)
```

```
## Loading required package: xml2
```

```
library(xts)
library(Hmisc)
```

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

```
## Loading required package: ggplot2
```

```
## 
## Attaching package: 'Hmisc'
```

```
## The following object is masked from 'package:rvest':
## 
##     html
```

```
## The following object is masked from 'package:quantmod':
## 
##     Lag
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
filePaths <- list.files(data_dir, "\\.csv$", full.names = TRUE)
N<-length(filePaths)
code.name<-array(1:N)
data.list<-list()
for (i in 1:N){
  name<-strsplit(filePaths[i], '/')[[1]][7]
  name<-strsplit(name,split=".",fixed=TRUE)[[1]][1]
  code.name[i]=name
  data<-read.csv(filePaths[i],i)
  data.list[[name]]=data
  rm(data)
  rm(name)
}
#max
lag.max<-function(arr,windows=9){
 N<-length(arr)
 result<-matrix(NA, N, 1)
 for (i in 1:N){
   if (i<9){
     result[i]=max(arr[1:i])
   }
   else{
     result[i]=max(arr[i-8:i])
   }
 }
 return(result)
}
#min
lag.min<-function(arr,windows=9){
  N<-length(arr)
  result<-matrix(NA, N, 1)
  for (i in 1:N){
    if (i<9){
      result[i]=min(arr[1:i])
    }
    else{
      result[i]=min(arr[i-8:i])
    }
  }
  return(result)
}

KDJ<-function(data,windows=9){
  l_temp <- nrow(data)
  KDJ <- matrix(50, l_temp, 3)
  KDJ <- as.data.frame(KDJ)
  colnames(KDJ) <- c('K', 'D', 'J')
  KDJ[1:(windows-1), ]  <- 50

  high_max <- lag.max(data$High)
  low_min <-  lag.min(data$Low)
  # rsv
  rsv <- (data$Close - low_min) / (high_max - low_min) * 100

  for (i in windows:l_temp) {

    KDJ[i, 1] <- 2/3 * KDJ[(i - 1), 1] + 1/3 * rsv[i, ]

    KDJ[i, 2] <- 2/3 * KDJ[(i - 1), 2] + 1/3 * KDJ[i, 1]
    KDJ[i, 3] <- 3 * KDJ[i, 1] - 2 * KDJ[i, 2]
  }

  return (KDJ)
}
Williams<-function(data,windows=14){
  high_max <- lag.max(data$High,windows)

  low_min <-  lag.min(data$Low,windows)
  result<-100-(data$Close-low_min)/(high_max-low_min)*100
  return(result)
}

label<-function(arr){
  N<-length(arr)
  K<-kmeans(arr,5) #k-means clustering
  K<-sort(K$centers)
  result<-matrix(NA, N, 1)
  for (i in 1:N){
    if (arr[i]>K[5]){
      result[i]=1
    }
     else if(arr[i]>K[4]){
      result[i]=2
    }
     else if(arr[i]>K[3]){
      result[i]=3
    }
    else if (arr[i]>K[2]){
      result[i]=4
    }
    else{
      result[i]=5
    }
  }
  return(result)
}
```

#Build models and forecasts

```
library(e1071)
```

```
##
## Attaching package: 'e1071'
```

```
## The following object is masked from 'package:Hmisc':
```

```
##
##     impute
```

```r
test.data.list<-list()
train.data.list<-list()
for (i in 1:N){
  name<-code.name[i]
  data<-data.list[[name]]
  data$MACD<-MACD(data$Close)[,1]
  data$RSI<-RSI(data$Close)
  data$KDJ<-KDJ(data,7)[,3] # parameter = 7 #use J value only
  impute <- function(x, x.impute){ifelse(is.na(x),x.impute,x)}

  data$KDJ<-impute(data$KDJ, 50)        #specific values
  data$KDJ<-impute(as.vector(data$KDJ), 50)
  data$Williams<-Williams(data,10) # parameter = 10
  data$return<-c(0,data$Close[2:length(data$Close)]/data$Close[1:(length(data$Close)-1)]-1)
  data$label<-label(data$return)
  #data$label<-kmeans(data$return,5)$
  data<-na.omit(data)
  train.data<-data
  test.data<-data
  train.data<-train.data[train.data$Date<'2017-01-01',] #set dates before year 2017 as training data
  test.data<-test.data[test.data$Date>='2017-01-01',] #set dates after 1 January, 2017 as testing data
  train.data.list[[name]]<-train.data
  test.data.list[[name]]<-test.data
  model<-svm(label~.,data=train.data.list[[name]][,c(8,9,10,11,13)]) #svm
  pred<-round(predict(model,test.data.list[[name]][,c(8,9,10,11)])) #prediction
  test.data$pred<-pred
  test.data.list[[name]]<-test.data
}
```

#Draw candlestick chart, MACD and RSI chart

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
## v purrr   0.3.4
```

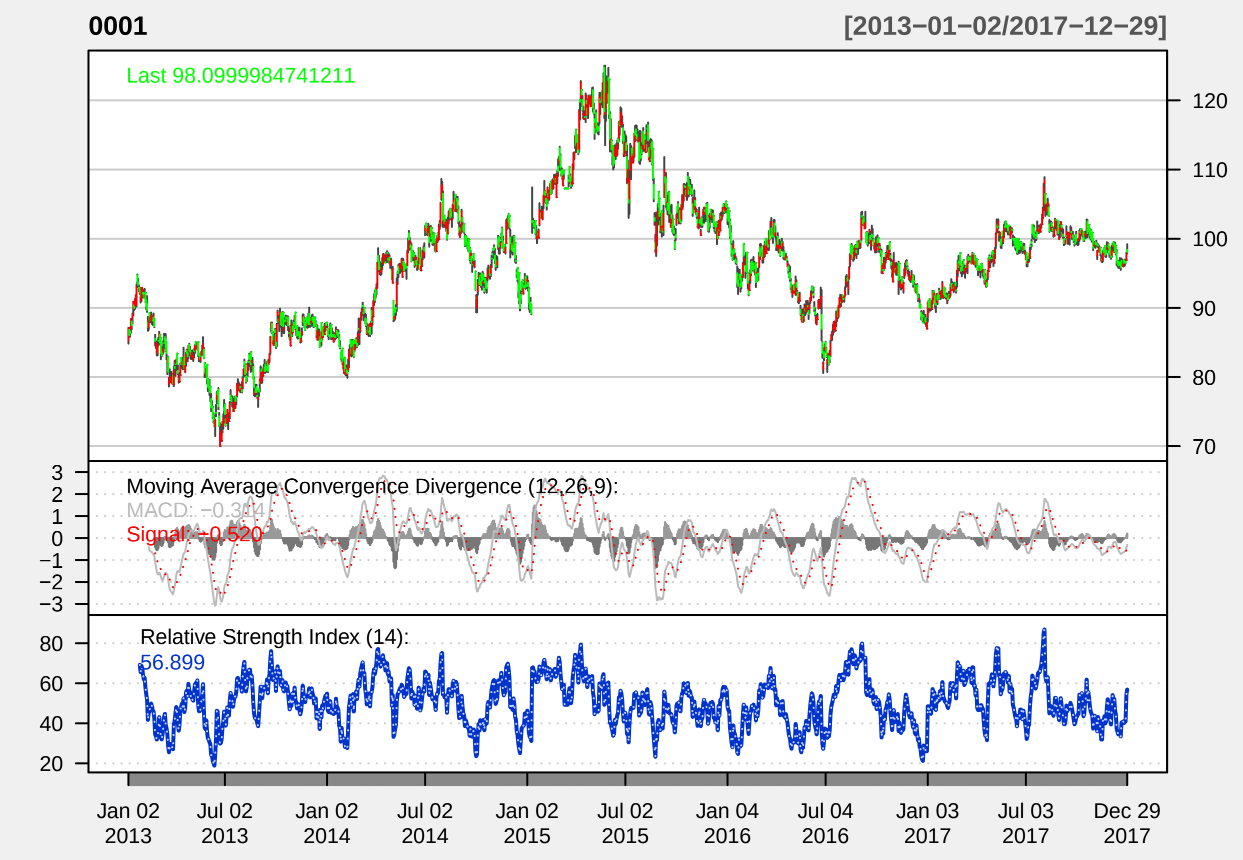
```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter()         masks stats::filter()
## x dplyr::first()          masks xts::first()
## x readr::guess_encoding() masks rvest::guess_encoding()
## x Hmisc::html()           masks rvest::html()
## x dplyr::lag()            masks stats::lag()
## x dplyr::last()           masks xts::last()
## x purrr::pluck()          masks rvest::pluck()
## x dplyr::src()            masks Hmisc::src()
## x dplyr::summarize()      masks Hmisc::summarize()
```

```r
library(gridExtra)
```

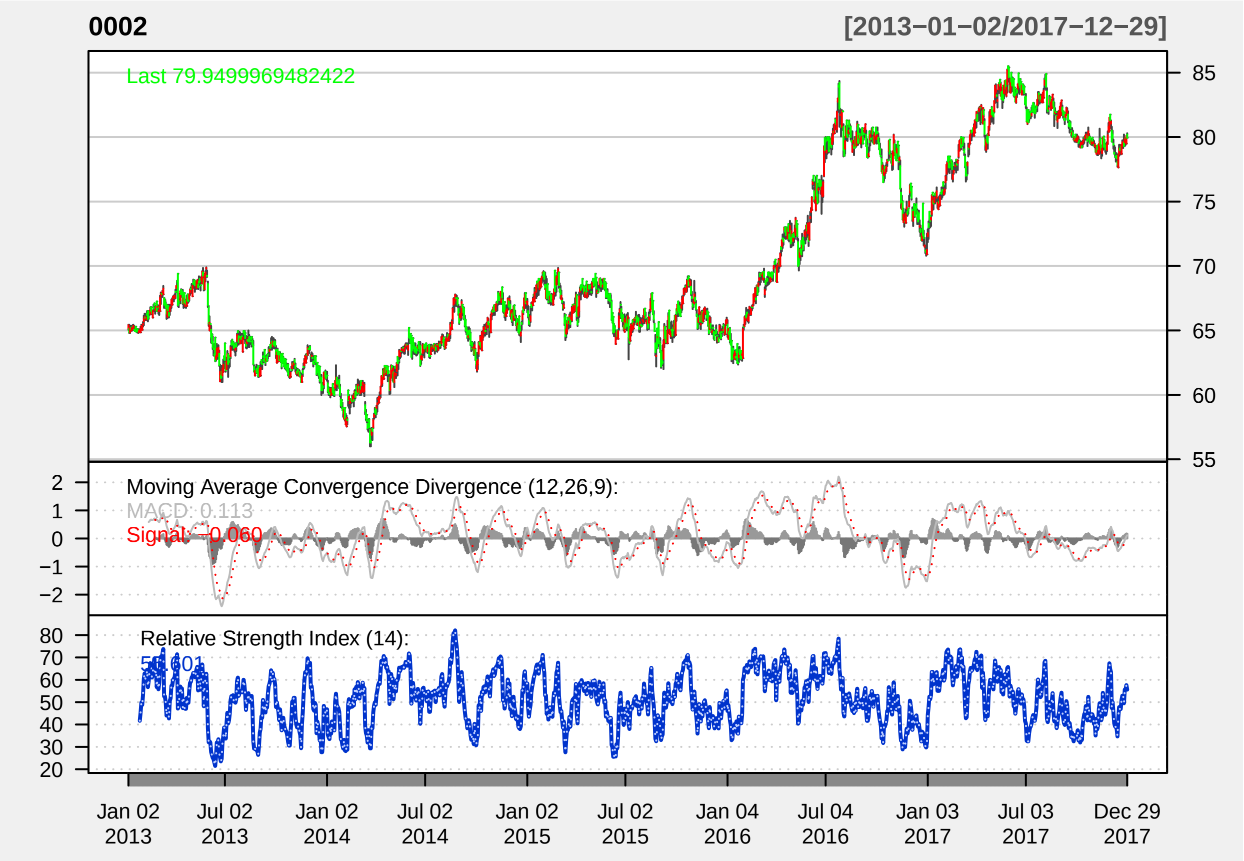
```
##
```

```
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

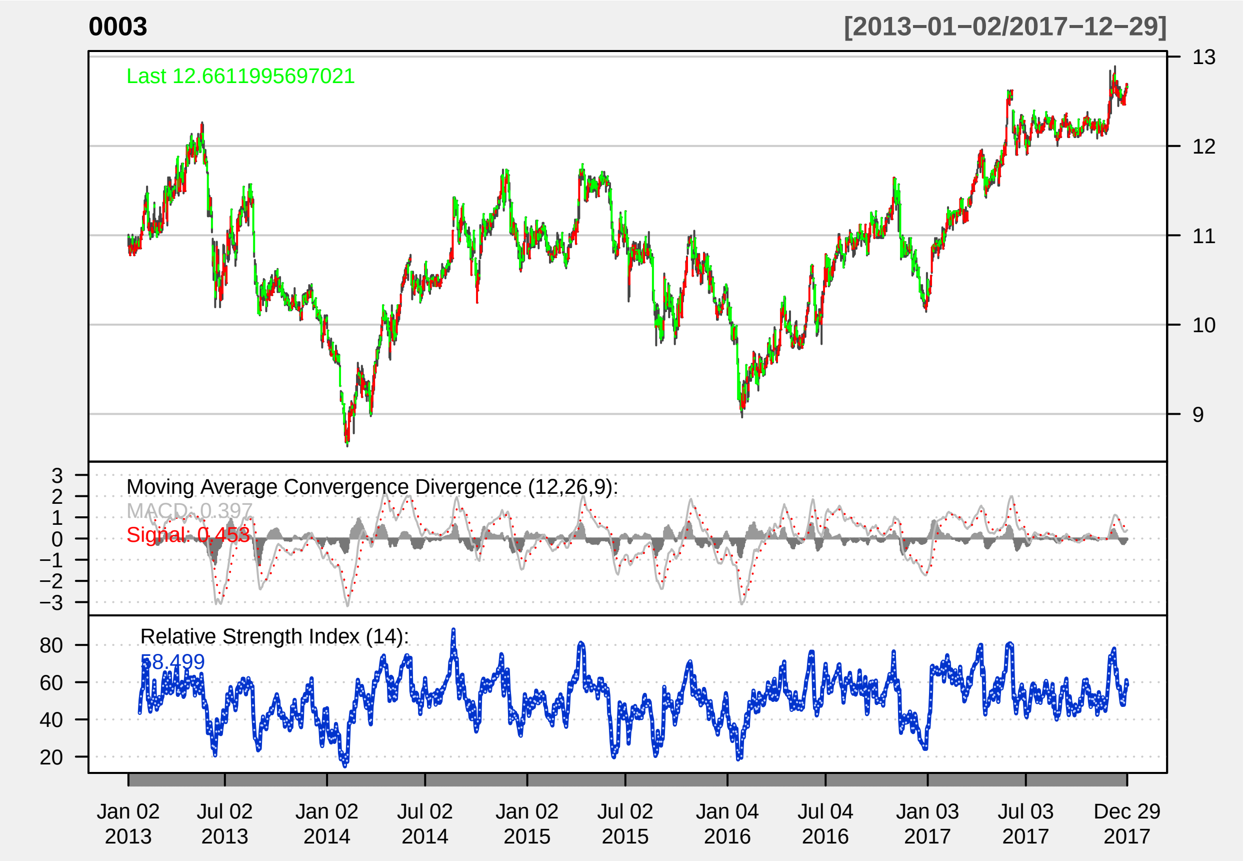
```
library(quantmod)  #need to use SVA function
k_plot<-function(code){
df<-data.list[[code]]
myvars <- c("Open","High","Low","Close","Volume")
data <- xts(df[myvars], order.by=as.Date(as.character(df[,1]),format="%Y-%m-%d"))
head(data)
stock <-data
chartSeries(x=stock["2013-01-01/"], name=code.name[code], line.type="l", bar.type="ohcl",
        theme="white", up.col='red', dn.col='green',
    TA="addMACD();addRSI();")
}
k_plot(1)
```



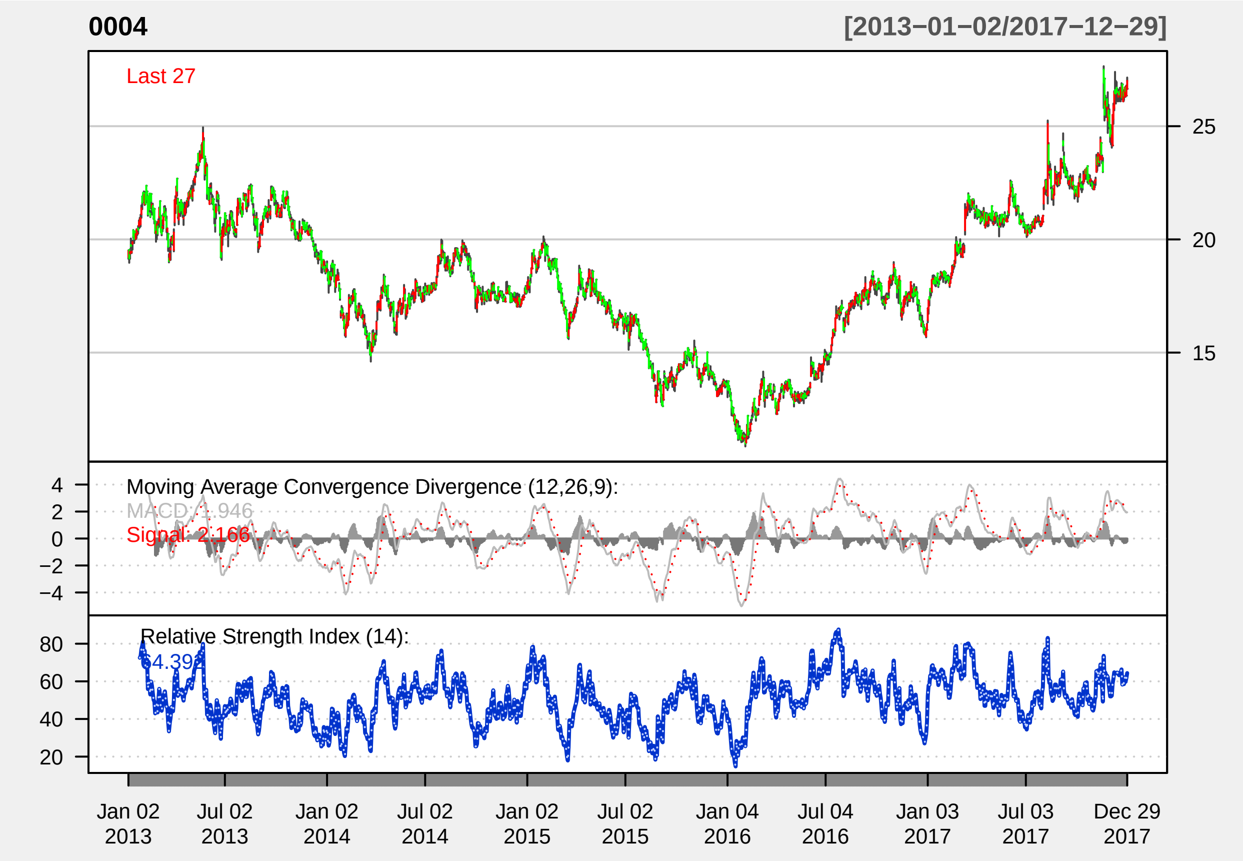
```
k_plot(2)
```


0002
[2013–01–02/2017–12–29]
Last 79.9499969482422
85
80
75
70
65
60
55
Moving Average Convergence Divergence (12,26,9):
MACD: 0.113
Signal: −0.060
2
1
0
−1
−2
Relative Strength Index (14):
57.601
80
70
60
50
40
30
20
Jan 02 2013
Jul 02 2013
Jan 02 2014
Jul 02 2014
Jan 02 2015
Jul 02 2015
Jan 04 2016
Jul 04 2016
Jan 03 2017
Jul 03 2017
Dec 29 2017

```
k_plot(3)
```

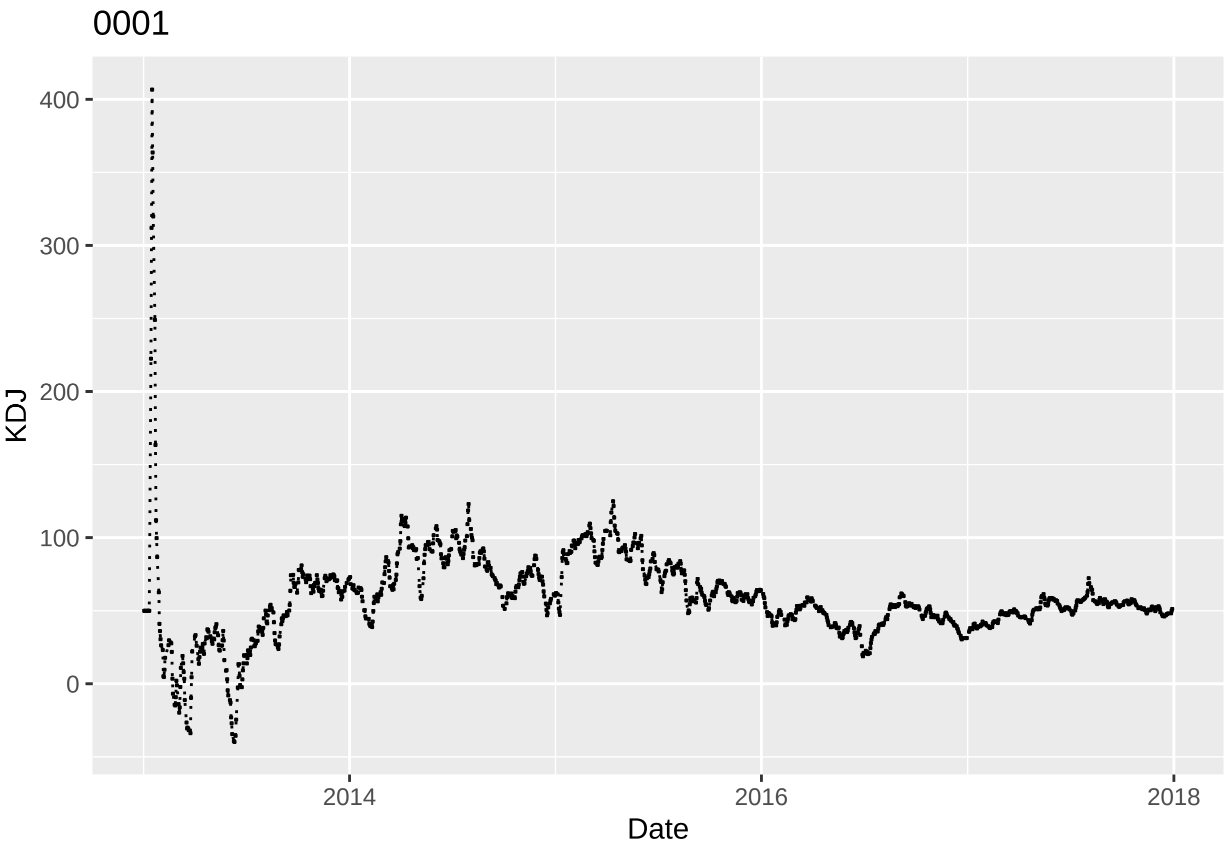

0003
[2013−01−02/2017−12−29]
Last 12.6611995697021
13
12
11
10
9
Moving Average Convergence Divergence (12,26,9):
MACD: 0.397
Signal: 0.453
3
2
1
0
−1
−2
−3
Relative Strength Index (14):
58.499
80
60
40
20
Jan 02 2013
Jul 02 2013
Jan 02 2014
Jul 02 2014
Jan 02 2015
Jul 02 2015
Jan 04 2016
Jul 04 2016
Jan 03 2017
Jul 03 2017
Dec 29 2017

```
k_plot(4)
```

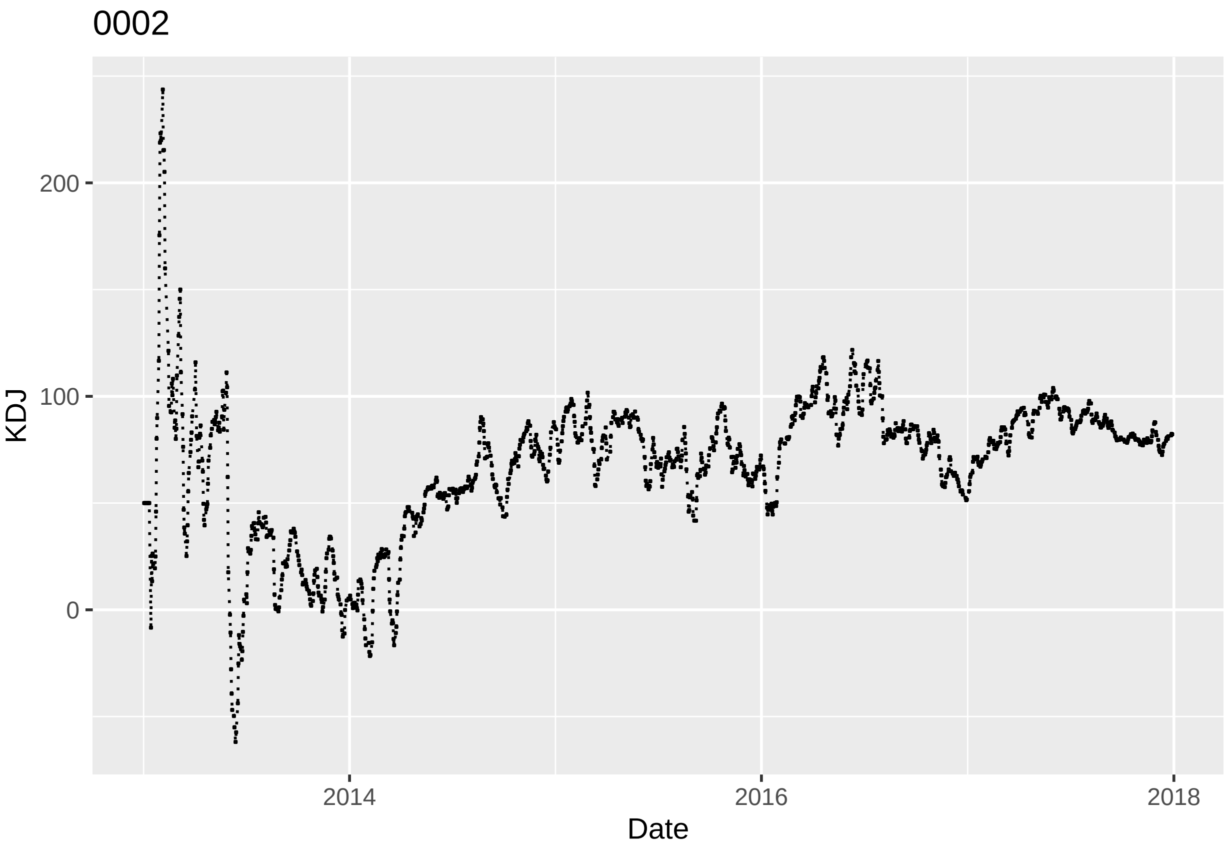


#Graph of KDJ indicator

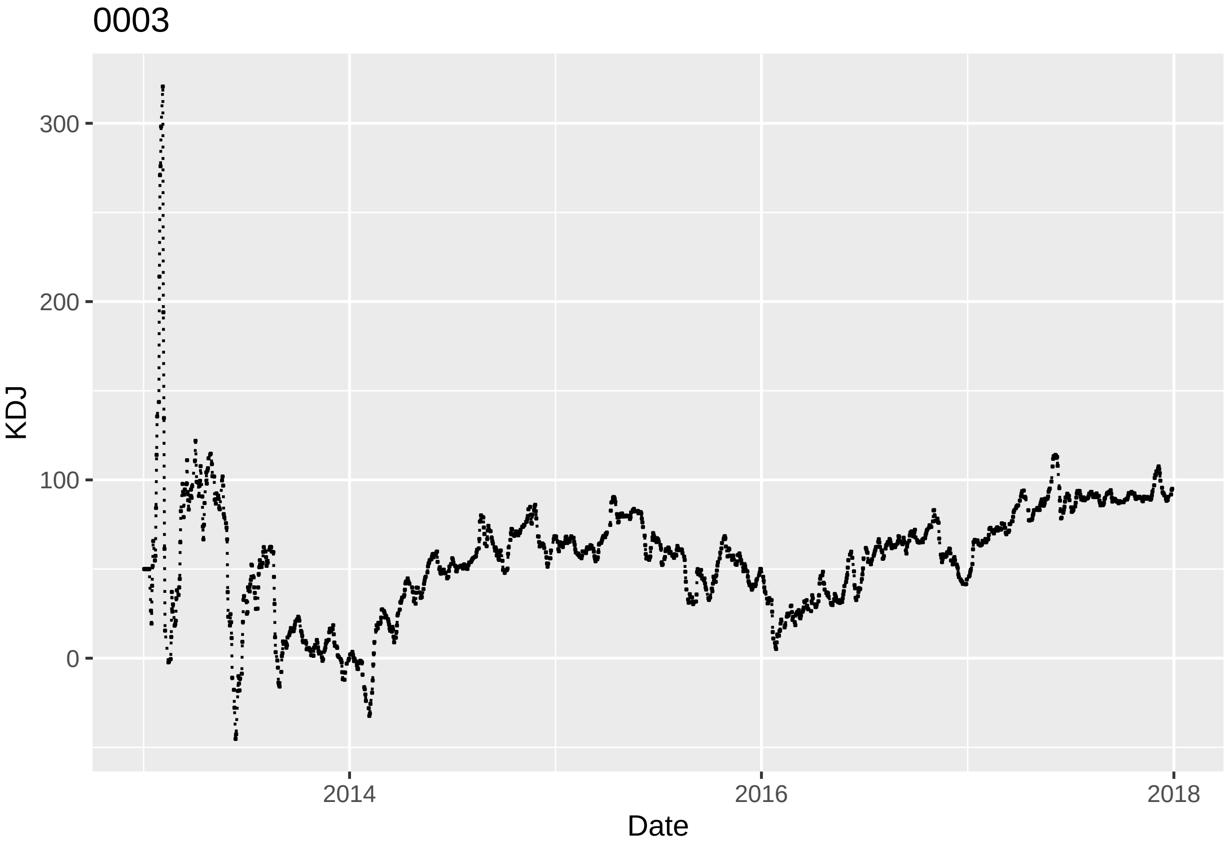
```
library(ggplot2)
plot_KDJ<-function(code){
  data<-data.list[[code]]
  data$Date<-as.Date(data$Date, '%Y-%m-%d', tz='GMT')
  data$KDJ<-KDJ(data)[,3]
  ggplot(data,aes(x = Date, y = KDJ, group = 1)) + geom_line(linetype="dotted") + geom_point(size=0.05,
    xlab("Date") + ylab("KDJ") +
    ggtitle(code.name[code])
}
plot_KDJ(0001)
```



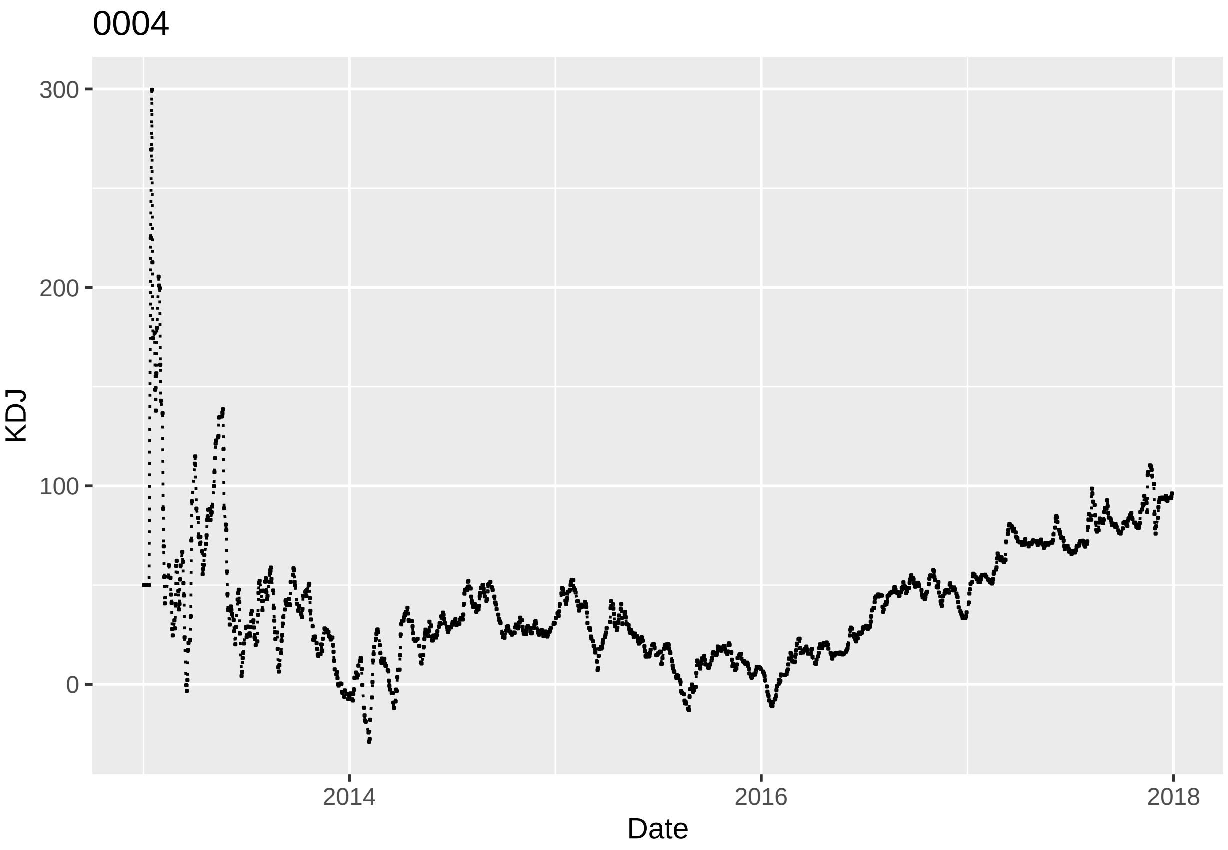
```r
plot_KDJ(0002)
```



```
plot_KDJ(0003)
```

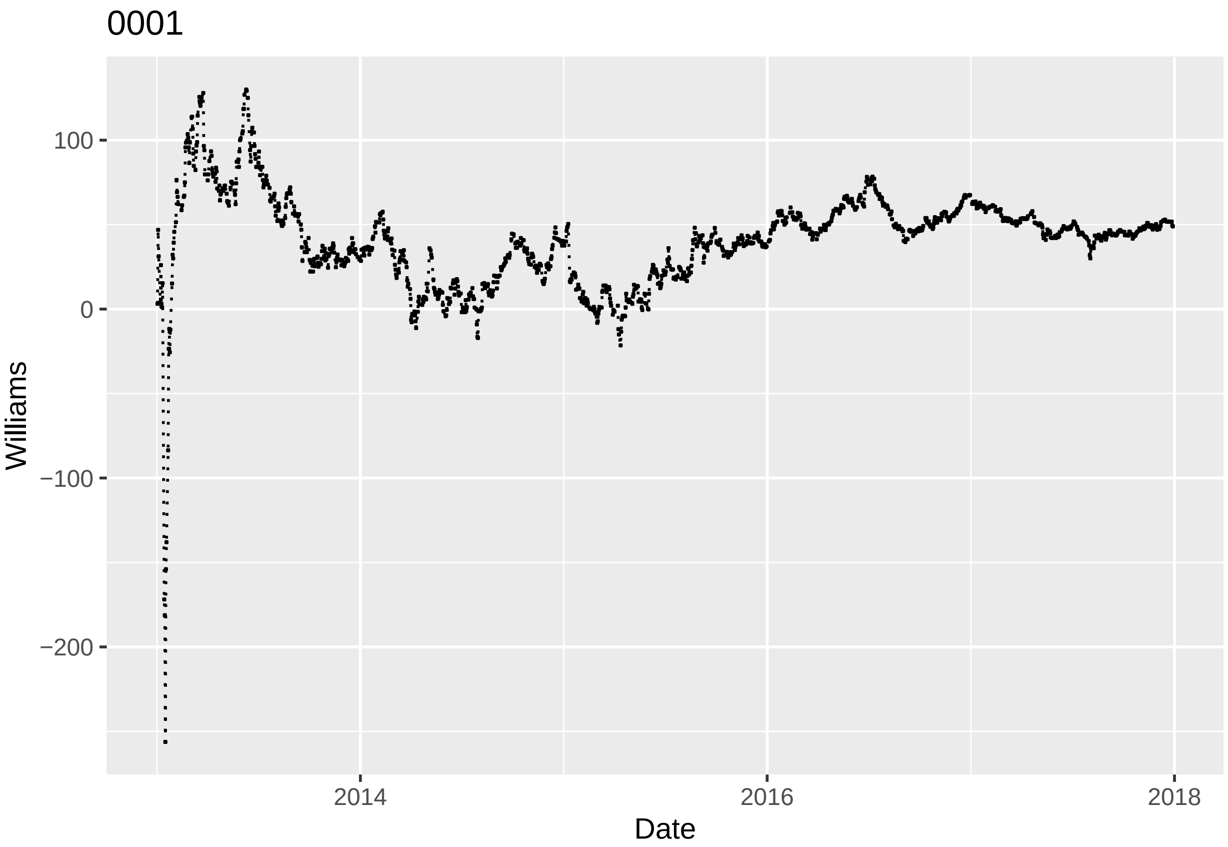


```r
plot_KDJ(0004)
```

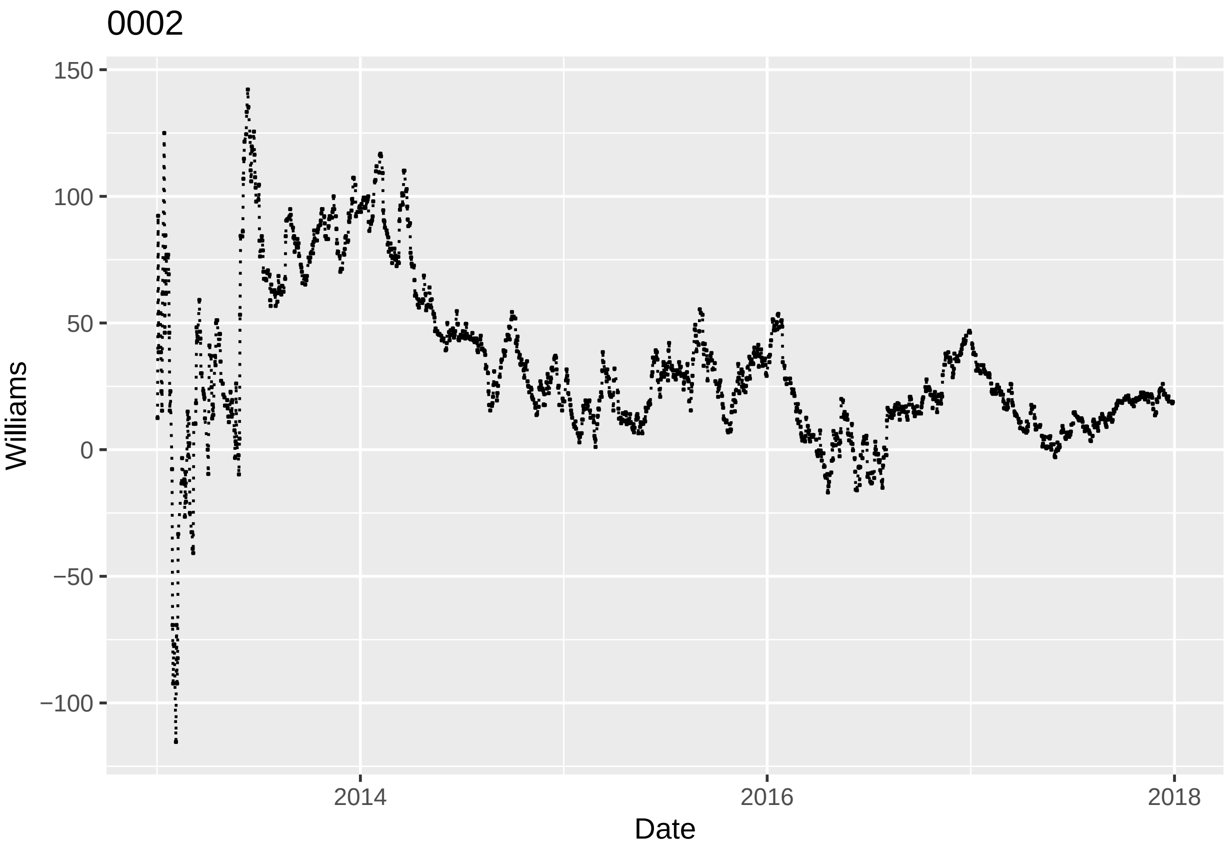


#Graph of KDJ indicator

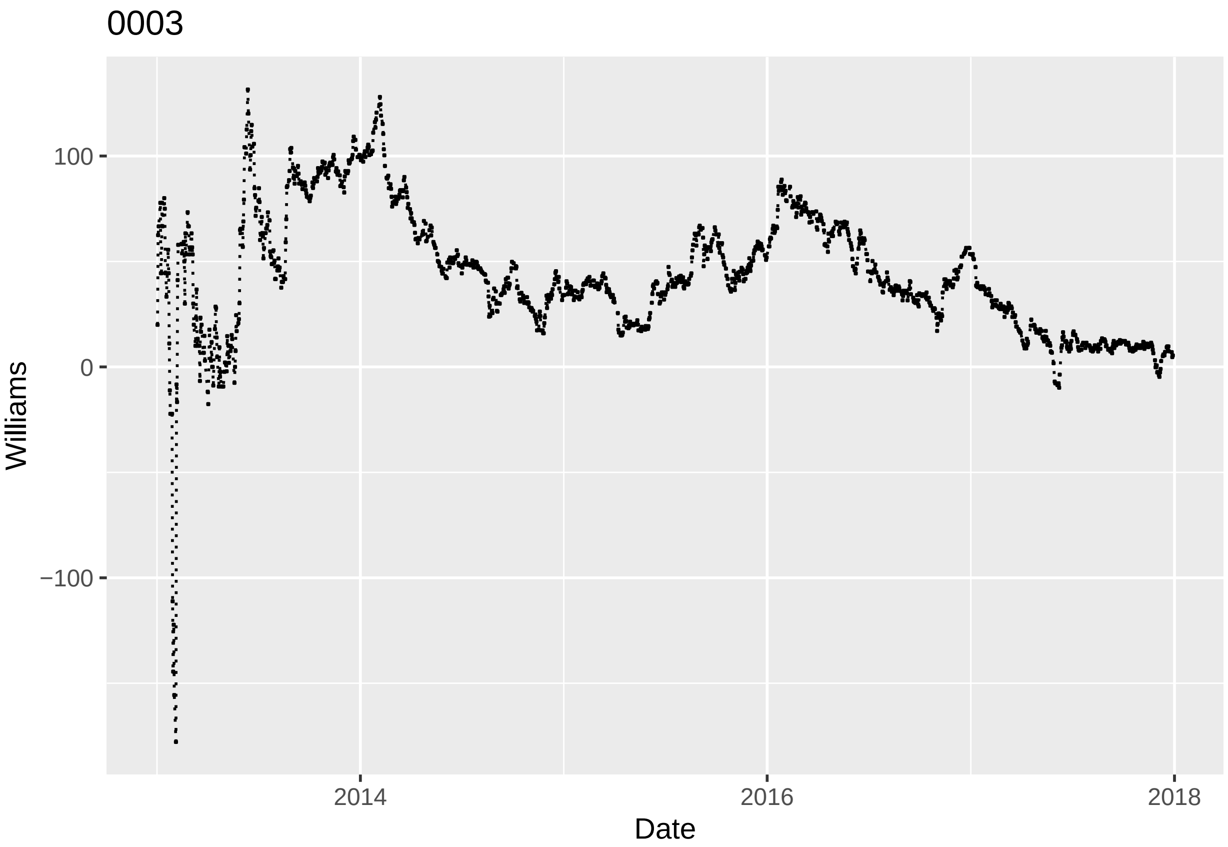
```
library(ggplot2)
plot_Williams<-function(code){
  data<-data.list[[code]]
  data$Date<-as.Date(data$Date, '%Y-%m-%d', tz='GMT')
  data$Williams<-Williams(data)
  ggplot(data,aes(x = Date, y =Williams, group = 1)) + geom_line(linetype="dotted") + geom_point(size=0
    xlab("Date") + ylab("Williams") +
    ggtitle(code.name[code])
}
plot_Williams(0001)
```



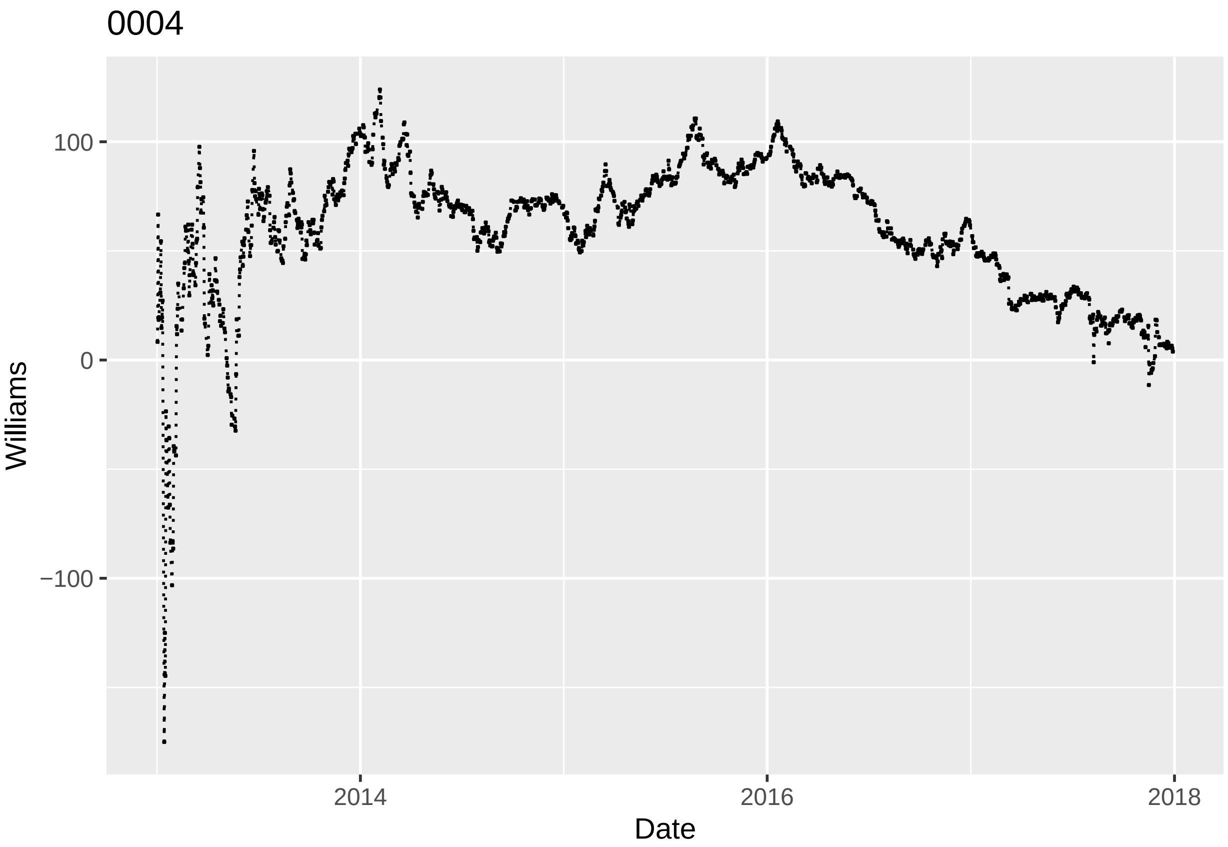
```
plot_Williams(0002)
```



```
plot_Williams(0003)
```



```r
plot_Williams(0004)
```



#Backtesting

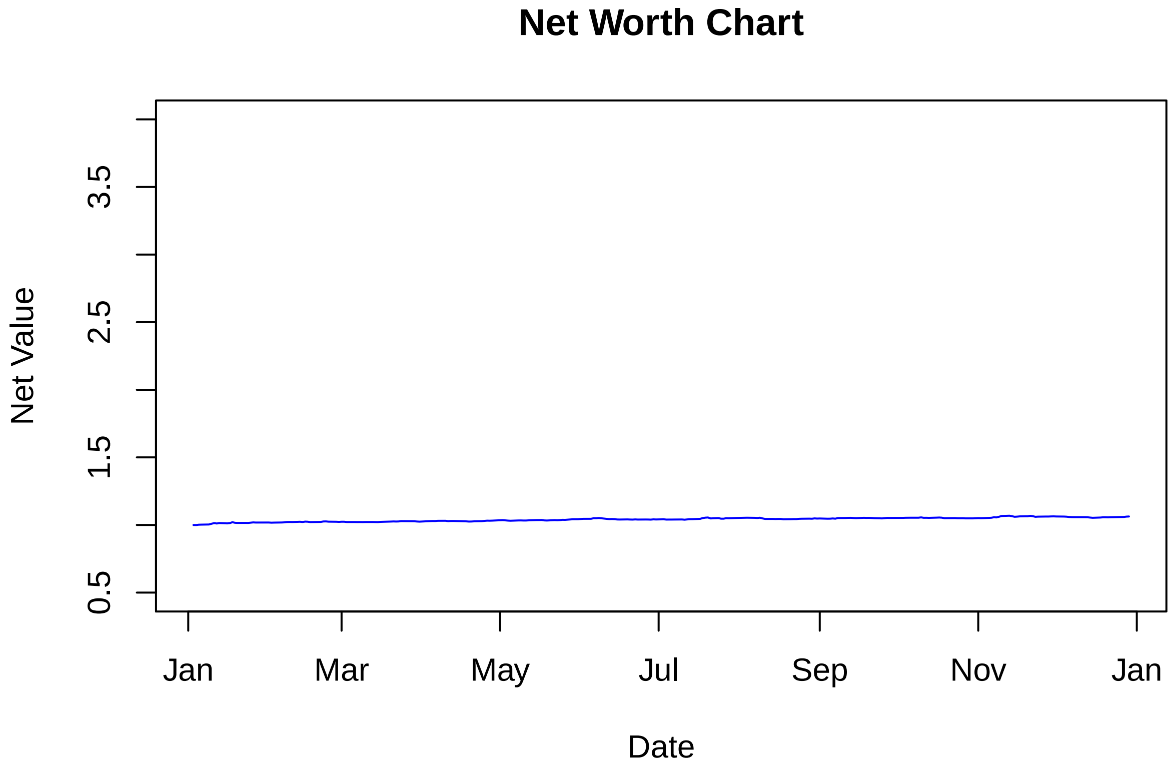
```
data_money=list()
for (j in 1:50){
  code=code.name[j]
  data=test.data.list[[j]]
   N<-length(data$Date)

  balance.money<-matrix(0, N, 1)
  code.money<-matrix(0, N, 1)
  money<-matrix(0, N, 1)
  for (i in 1:N){
  if (i==1){
    balance.money[i]=200000
  }
    else{
       balance.money[i]=balance.money[i-1]
       code.money[i]=code.money[i-1]
    }
  if ((data$pred[i]==1|data$pred[i]==2)&code.money[i]==0){
      if (balance.money[i]-floor(balance.money[i]/data$Close[i])*data$Close[i]-floor(balance.money[i]/da
        b=balance.money[i]-(floor(balance.money[i]/data$Close[i])-1)*data$Close[i]-(floor(balance.money
          code.money[i]=(floor(balance.money[i]/data$Close[i])-1)*data$Close[i]
          balance.money[i]=b
      }
      else{
    b=balance.money[i]-floor(balance.money[i]/data$Close[i])*data$Close[i]-floor(balance.money[i]/data$C
    code.money[i]=floor(balance.money[i]/data$Close[i])*data$Close[i]
```

```r
   balance.money[i]=b }

    }


    else{
      if (code.money[i]>0){
      code.money[i]= code.money[i]*(1+data$return[i])


      }
    if (code.money[i]>0 &(data$pred[i]==4|data$pred[i]==5)){
      balance.money[i]=balance.money[i]+code.money[i]
      code.money[i]=0

    }
    }
    money[i]=code.money[i]+balance.money[i]
  }

  money<-data.frame(money)
  money$date<-data$Date
  data_money[[code]]=money
}
 money<-matrix(0, N, 1)

for (i in 1:N){
  for (j in 1:50){
  code=code.name[j]
  data=data_money[[j]]
  money[i]=data$money[i]+money[i]
  }
}
 money<-data.frame(money)
  money$date<-data$date
  print(money)
```

```
##        money       date
## 1    9999208 2017-01-03
## 2    9993757 2017-01-04
## 3   10020803 2017-01-05
## 4   10025489 2017-01-06
## 5   10037053 2017-01-09
## 6   10094231 2017-01-10
## 7   10136474 2017-01-11
## 8   10108579 2017-01-12
## 9   10140228 2017-01-13
## 10  10116619 2017-01-16
## 11  10138607 2017-01-17
## 12  10204173 2017-01-18
## 13  10163188 2017-01-19
## 14  10152894 2017-01-20
## 15  10154417 2017-01-23
## 16  10149860 2017-01-24
## 17  10169706 2017-01-25
## 18  10189725 2017-01-26
## 19  10180595 2017-01-27
## 20  10182961 2017-02-01
## 21  10170570 2017-02-02
## 22  10175423 2017-02-03
## 23  10184207 2017-02-06
## 24  10193708 2017-02-07
## 25  10218597 2017-02-08
## 26  10221910 2017-02-09
## 27  10218243 2017-02-10
## 28  10239364 2017-02-13
## 29  10220987 2017-02-14
## 30  10248447 2017-02-15
## 31  10241315 2017-02-16
## 32  10212455 2017-02-17
## 33  10222798 2017-02-20
## 34  10228157 2017-02-21
## 35  10258881 2017-02-22
## 36  10263258 2017-02-23
## 37  10245072 2017-02-24
## 38  10240232 2017-02-27
## 39  10229138 2017-02-28
## 40  10242046 2017-03-01
## 41  10240599 2017-03-02
## 42  10218175 2017-03-03
## 43  10218183 2017-03-06
## 44  10214106 2017-03-07
## 45  10216774 2017-03-08
## 46  10211906 2017-03-09
## 47  10217878 2017-03-10
## 48  10219981 2017-03-13
## 49  10213834 2017-03-14
## 50  10209021 2017-03-15
## 51  10230857 2017-03-16
## 52  10237495 2017-03-17
## 53  10253840 2017-03-20
## 54  10261627 2017-03-21
## 55  10254748 2017-03-22
## 56  10261123 2017-03-23
## 57  10282157 2017-03-24
## 58  10277008 2017-03-27
## 59  10276797 2017-03-28
## 60  10273345 2017-03-29
## 61  10256671 2017-03-30
## 62  10247682 2017-03-31
## 63  10274317 2017-04-03
## 64  10292561 2017-04-05
## 65  10290626 2017-04-06
## 66  10311253 2017-04-07
## 67  10313316 2017-04-10
## 68  10281741 2017-04-11
## 69  10294533 2017-04-12
## 70  10296579 2017-04-13
## 71  10267678 2017-04-18
## 72  10252600 2017-04-19
## 73  10257551 2017-04-20
## 74  10269792 2017-04-21
## 75  10281246 2017-04-24
## 76  10304836 2017-04-25
## 77  10319943 2017-04-26
## 78  10315674 2017-04-27
## 79  10322148 2017-04-28
## 80  10358280 2017-05-02
## 81  10323429 2017-05-04
## 82  10315157 2017-05-05
## 83  10334739 2017-05-08
## 84  10335792 2017-05-09
## 85  10329892 2017-05-10
## 86  10329949 2017-05-11
## 87  10342434 2017-05-12
## 88  10362015 2017-05-15
## 89  10363515 2017-05-16
## 90  10369061 2017-05-17
## 91  10338231 2017-05-18
## 92  10330336 2017-05-19
## 93  10357813 2017-05-22
## 94  10347571 2017-05-23
## 95  10359051 2017-05-24
## 96  10384303 2017-05-25
## 97  10379760 2017-05-26
## 98  10422186 2017-05-29
## 99  10424778 2017-05-31
## 100 10442715 2017-06-01
## 101 10456621 2017-06-02
## 102 10457591 2017-06-05
## 103 10497900 2017-06-06
## 104 10495600 2017-06-07
## 105 10516112 2017-06-08
## 106 10494839 2017-06-09
## 107 10432942 2017-06-12
## 108 10441321 2017-06-13
## 109 10433497 2017-06-14
## 110 10408090 2017-06-15
## 111 10403707 2017-06-16
## 112 10412050 2017-06-19
## 113 10402917 2017-06-20
## 114 10398424 2017-06-21
## 115 10411818 2017-06-22
## 116 10400383 2017-06-23
## 117 10405652 2017-06-26
## 118 10404005 2017-06-27
## 119 10398090 2017-06-28
## 120 10415399 2017-06-29
## 121 10406356 2017-06-30
## 122 10418529 2017-07-03
## 123 10399965 2017-07-04
## 124 10400822 2017-07-05
## 125 10401172 2017-07-06
## 126 10402919 2017-07-07
## 127 10406285 2017-07-10
## 128 10390974 2017-07-11
## 129 10411849 2017-07-12
## 130 10425100 2017-07-13
## 131 10427046 2017-07-14
## 132 10453661 2017-07-17
## 133 10512503 2017-07-18
## 134 10541828 2017-07-19
## 135 10554214 2017-07-20
## 136 10483615 2017-07-21
## 137 10509545 2017-07-24
## 138 10466450 2017-07-25
## 139 10465900 2017-07-26
## 140 10500236 2017-07-27
## 141 10494912 2017-07-28
## 142 10518515 2017-07-31
## 143 10524501 2017-08-01
## 144 10529731 2017-08-02
## 145 10533427 2017-08-03
## 146 10536652 2017-08-04
## 147 10529515 2017-08-07
## 148 10515616 2017-08-08
## 149 10536617 2017-08-09
## 150 10483853 2017-08-10
## 151 10448465 2017-08-11
## 152 10449127 2017-08-14
## 153 10441752 2017-08-15
## 154 10447239 2017-08-16
## 155 10446103 2017-08-17
## 156 10416456 2017-08-18
## 157 10424739 2017-08-21
## 158 10432306 2017-08-22
## 159 10432306 2017-08-23
## 160 10457553 2017-08-24
## 161 10462703 2017-08-25
## 162 10469312 2017-08-28
## 163 10462361 2017-08-29
## 164 10486824 2017-08-30
## 165 10473358 2017-08-31
## 166 10479434 2017-09-01
## 167 10463873 2017-09-04
## 168 10466392 2017-09-05
## 169 10482111 2017-09-06
## 170 10466788 2017-09-07
## 171 10512580 2017-09-08
## 172 10517871 2017-09-11
## 173 10528215 2017-09-12
## 174 10529286 2017-09-13
## 175 10513730 2017-09-14
## 176 10504267 2017-09-15
## 177 10532473 2017-09-18
## 178 10523587 2017-09-19
## 179 10526936 2017-09-20
## 180 10516516 2017-09-21
## 181 10501106 2017-09-22
## 182 10486232 2017-09-25
## 183 10503107 2017-09-26
## 184 10522985 2017-09-27
## 185 10518896 2017-09-28
## 186 10521920 2017-09-29
## 187 10524928 2017-10-03
## 188 10533228 2017-10-04
## 189 10538465 2017-10-06
## 190 10538015 2017-10-09
## 191 10560776 2017-10-10
## 192 10531695 2017-10-11
## 193 10533745 2017-10-12
## 194 10527531 2017-10-13
## 195 10544390 2017-10-16
## 196 10552238 2017-10-17
## 197 10539122 2017-10-18
## 198 10499262 2017-10-19
## 199 10501635 2017-10-20
## 200 10508285 2017-10-23
## 201 10496442 2017-10-24
## 202 10493431 2017-10-25
## 203 10495503 2017-10-26
## 204 10488505 2017-10-27
## 205 10487254 2017-10-30
## 206 10496170 2017-10-31
## 207 10505714 2017-11-01
## 208 10500406 2017-11-02
## 209 10506901 2017-11-03
## 210 10533757 2017-11-06
## 211 10569463 2017-11-07
## 212 10559508 2017-11-08
## 213 10609026 2017-11-09
## 214 10664177 2017-11-10
## 215 10686095 2017-11-13
## 216 10648934 2017-11-14
## 217 10606005 2017-11-15
## 218 10617561 2017-11-16
## 219 10636886 2017-11-17
## 220 10639470 2017-11-20
## 221 10676455 2017-11-21
## 222 10648884 2017-11-22
## 223 10604016 2017-11-23
## 224 10616405 2017-11-24
## 225 10625149 2017-11-27
## 226 10625165 2017-11-28
## 227 10631694 2017-11-29
## 228 10635963 2017-11-30
## 229 10628228 2017-12-01
## 230 10622058 2017-12-04
## 231 10612233 2017-12-05
## 232 10590751 2017-12-06
## 233 10578342 2017-12-07
## 234 10576354 2017-12-08
## 235 10575709 2017-12-11
## 236 10575863 2017-12-12
## 237 10571168 2017-12-13
## 238 10548536 2017-12-14
## 239 10531646 2017-12-15
## 240 10550457 2017-12-18
## 241 10566178 2017-12-19
## 242 10564412 2017-12-20
## 243 10563430 2017-12-21
## 244 10570005 2017-12-22
## 245 10591942 2017-12-27
## 246 10618702 2017-12-28
## 247 10627528 2017-12-29
```

#Evaluation of backtesting results

```
dates <- as.Date(money$date, "%Y-%m-%d")
plot(dates,money$money/(200000*50), type="l", lwd=1, main="Net Worth Chart", xlab="Date",ylab="Net Valu
```



```
library(tseries)
library(PerformanceAnalytics)
```

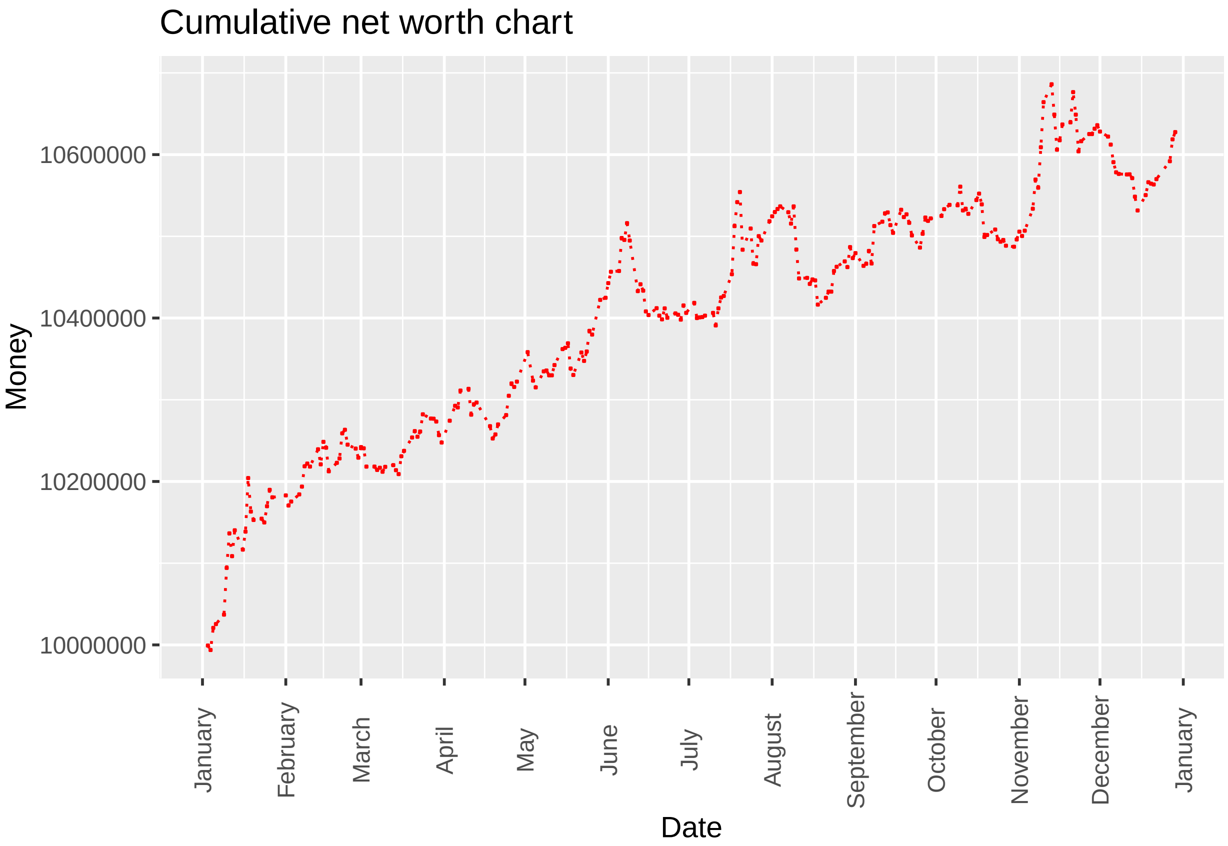
```
##
## Attaching package: 'PerformanceAnalytics'

## The following objects are masked from 'package:e1071':
##
##     kurtosis, skewness

## The following object is masked from 'package:graphics':
```

```
##
##      legend
```

```r
#maximum drawdown rate
mdd <- maxdrawdown(money$money[1:(N-1)]/(200000*50))
print(mdd)
```

```
## $maxdrawdown
## [1] 0.01544493
##
## $from
## [1] 215
##
## $to
## [1] 239
```

```r
ggplot(money,aes(x = as.Date(date), y =money, group = 1)) + geom_line(linetype="dotted", color="red") +
```



```r
library(PerformanceAnalytics)
N=length(money$money)
money$return=c(0,money$money[2:N]/money$money[1:(N-1)]-1)
rownames(money)=as.Date(money$date, '%Y-%m-%d', tz='GMT')
return<-data.frame(money$return)
rownames(return)=as.Date(money$date, '%Y-%m-%d', tz='GMT')
return<-na.omit(return)

#annual average rate of return
Return.annualized(return)
```

```
##                     money.return
## Annualized Return   0.06414889
```

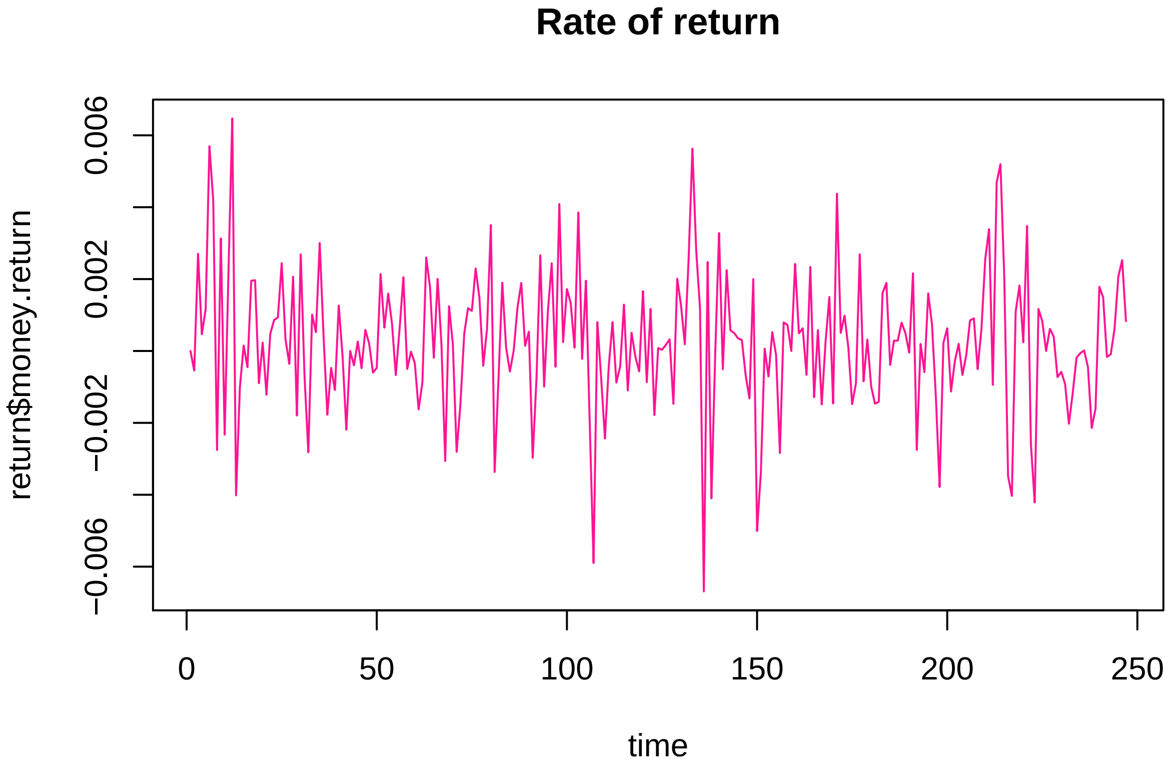
```
#annual standardized deviation
StdDev.annualized(return)
```

```
##                                 money.return
## Annualized Standard Deviation   0.03010812
```

```
#annual sharpe ratio
ann_sharpe <- (Return.annualized(return) / StdDev.annualized(return))

# Sharp ratio is calculated by subtracting the average excess return from the risk-free interest rate a
# Use the function table.AnnualizedReturns() to get all the above results at once
ann_sharpe
```

```
##                     money.return
## Annualized Return     2.130617
```

```
plot(return$money.return,type='l',xlab="time",col="DeepPink", main="Rate of return")
```


Rate of return
return$money.return
time