Assignment 1

Philipp Epstein

10/14/2018

Introduction

Blabla Blabla

Trading Idea

Explain here the traiding idea, where I found it. What the theoretical foundation is –> Long Term positive autocorreleation –> Short term negative autocorreleation

Trade reverse patterns. When observing a fallback, trade to profit from the upward trend after the fallback. etc. . . .

Implementation of the trading idea

Short paragraph about what is needed for the implementation

Part A: Initialization

General Setup

```r
# Clear Environment
rm(list=ls())

# Loading libraries
library(blotter)
library(INFT361Course)
```

Setting the Variables

The variables set in the next section can be adjusted to test the strategy with different parameters.

```r
# Set values:
startCapital <- 1e+6
transactionCost <- -20
daterange <- '2016::2018'
emaPeriod <- 200
maxHoldingPeriod <- 30

InstrumentDirectory <- "~/Desktop/R/DownloadedData/"
instrumentlist <- c("SAP.csv", "DBK.csv")
BuyHoldDirectory <- InstrumentDirectory
BuyHoldInstrument <- "DAXEX.csv"

currency("EUR")
Sys.setenv(TZ="UTC")
initdate <- '1999-12-31'
startdate <- '2000-01-01'
enddate <- '2018-12-31'
```

```r
portfolioname <- "Smash Day"
accountname <- portfolioname
```

Initializing the portfolio

```r
# Clear portfolio and Account
suppressWarnings(rm("account.Smash Day","portfolio.Smash Day","account.buyhold","portfolio.buyhold",pos

# Initialize Portfolio and Account
initPortf(portfolioname,instrumentlist,initDate=initdate,currency="EUR")
initAcct(accountname,portfolios=portfolioname,initDate=initdate,initEq=startCapital,currency="EUR")
```

Part B: Bar by bar processing

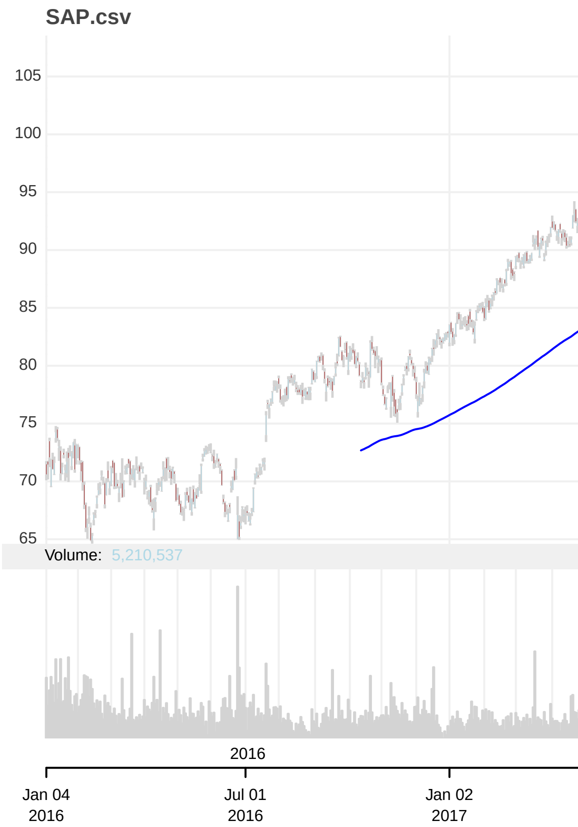
Loading the instrument, initializing it and adding the ema to the data.

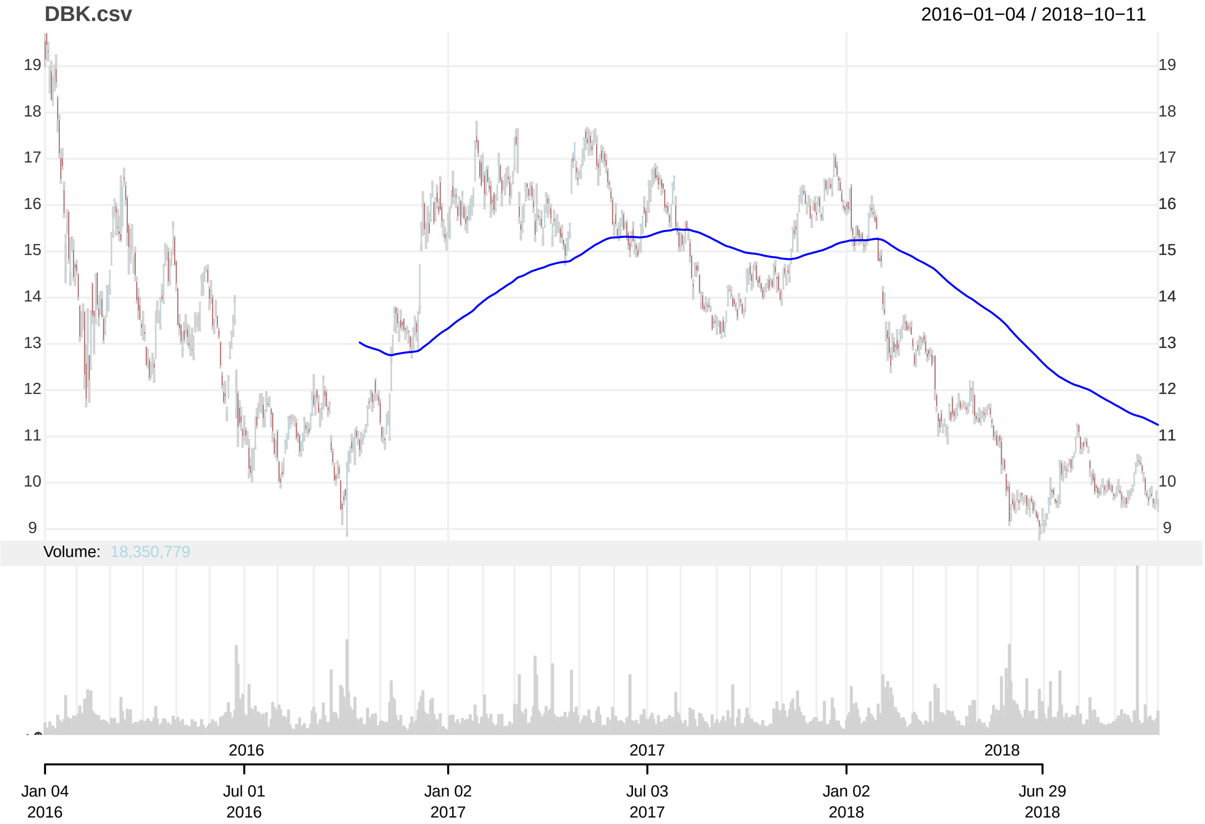
```r
for (instrument in instrumentlist) {
    LoadCourseFile(InstrumentDirectory, instrument, debugme = TRUE, dates = daterange)

    # Initialize the instrument
    stock(instrument, currency = "EUR")

    # Load the XTS file
    symbol <- get(instrument)

    # Calculate the Exponential Moving Average
    ema <- EMA(symbol$Close, n=emaPeriod)

    # Merge the xts file with the Exponential Moving Average
    symbol <- merge(symbol,ema)
    assign(instrument,symbol)
  # Starting to go bar by bar through using a "for loop"
  for (i in (emaPeriod + 1):(nrow(symbol) - 1)) {
    # Dates
    CurrentDate <- time(symbol[i])
    TomorrowDate <- time(symbol[i + 1])

    # Today's variables
    CloseToday <- as.numeric(symbol[i, "Close"])
    EMA_today <- as.numeric(symbol[i, "EMA"])
    LowToday <- as.numeric(symbol[i, "Low"])
    HighToday <- as.numeric(symbol[i, "High"])

    # Yesterday's variables
    LowYesterday <- as.numeric(symbol[i - 1, "Low"])
    HighYesterday <- as.numeric(symbol[i - 1, "High"])

    # Tomorrow's variables
    OpenTomorrow <- as.numeric(symbol[i + 1, "Open"])
    LowTomorrow <- as.numeric(symbol[i + 1, "Low"])
    HighTomorrow <- as.numeric(symbol[i + 1, "High"])

    # Config
    Equity <- getEndEq(accountname, CurrentDate)
    Position <-
      getPosQty(portfolioname, Symbol = instrument, Date = CurrentDate)

    # Check whether we have a position
    if (Position == 0) {
      # Start checking BUY rules

      # Check whether we have a Smash Day (Long).
      # Smash Day (Long) is when Todays Close is below Yesterdays Low.
      if (CloseToday < LowYesterday) {
        # Smash Day (Long)

        #Check whether todays close is above today's EMA
        if (CloseToday > EMA_today) {

          # BUY RULE: If today was a smash day place a STOP BUY order at todays high price.
          # (Buy tomorrow for 'price >= todays high')

          # Simulate STOP BUY order:

          # Option 1 to check: Check whether the open price tomorrow is above today's high
          # and add the transaction tomorrow at tomorrows open price.

          # Option 2 to check: Check whether today's high was lower than tomorrows high
          # and add the transaction tomorrow at today's high price.

          # Check Option 1
          if (OpenTomorrow > HighToday) {
            # Don't trade at the day before the last day
            if (CurrentDate != time(symbol[nrow(symbol) - 1])) {
              # Calculate the buy quantity
              BuyQuantity <- as.numeric(trunc(Equity / OpenTomorrow))
              # Add transaction
              addTxn(
                portfolioname,
                Symbol = instrument,
                TxnDate = TomorrowDate ,
                TxnPrice = OpenTomorrow,
                TxnQty = BuyQuantity,
                TxnFees = transactionCost
              )
              # Store the bar at which we placed the transaction
              BuyBar <- i
            }

          } else {
            # Check Option 2
            if (HighToday < HighTomorrow) {
              # Don't trade at the day before the last day
              if (CurrentDate != time(symbol[nrow(symbol) - 1])) {
                # Calculate the buy quantity
                BuyQuantity <- as.numeric(trunc(Equity / HighToday))
                # Add transaction
                addTxn(
                  portfolioname,
                  Symbol = instrument,
                  TxnDate = TomorrowDate ,
                  TxnPrice = HighToday,
                  TxnQty = BuyQuantity,
                  TxnFees = transactionCost
                )
                # Store the bar at which we placed the transaction
                BuyBar <- i
              }
            }
          }
        }
      }
    } else {
      # We already have a position

      # Check the sell rules in the following order and sell at the
      # first condition which is satisfied.

      # Sell rules:
      # Rule 1: Sell if we hold the position longer than the specified
      # maximum holding period

      # Rule 2: Sell at tomorrow's opening price if the close price
      # today falls below the EMA

      # Rule 3: Sell if we meet the Smash Day (Short) requirements.
      # Today's close must be higher than yesterday's high

      # Rule 4: If no sell rule can be applied and we reach the
      # second last day. Sell at the last day.

      # Check Rule 1:
      if ((i - BuyBar) > maxHoldingPeriod) {
        # Place the sell transaction at todays close price
        addTxn(
          portfolioname,
          Symbol = instrument,
          TxnDate = CurrentDate,
          TxnPrice = as.numeric(symbol[i, "Close"]),
          TxnQty = -Position,
          TxnFees = transactionCost
        )

      } else {
        # Check Rule 2:
        if (as.numeric(symbol[i, "Close"]) < EMA_today) {
          # Place the sell transaction at tomorrow's open price
          addTxn(
            portfolioname,
            Symbol = instrument,
            TxnDate = time(symbol[i + 1]),
            TxnPrice = OpenTomorrow,
            TxnQty = -Position,
            TxnFees = transactionCost
          )

        } else {
          # Check Rule 3:

          # Sell Rule 3: If today was a smash day (short) place an order at todays
          # low price. (Buy tomorrow for 'price <= todays low')

          # Simulate this behaviour:

          # Option 1 to check: Check whether the open price tomorrow is below today's
          # low and add the transaction tomorrow at tomorrows open price.

          # Option 2 to check: Check whether today's low was larger than tomorrow's
          # low and add the transaction tomorrow at today's low price.

          # Check for Smash Day (Short)
          if (CloseToday > HighYesterday) {
            # Check for Option 1
            if (OpenTomorrow < LowToday) {
              # Add Sell transaction tomorrow at tomorrow's open price
              addTxn(
                portfolioname,
                Symbol = instrument,
                TxnDate = time(symbol[i + 1]),
                TxnPrice = OpenTomorrow,
                TxnQty = -Position,
                TxnFees = transactionCost
              )

            } else {
              # Check for Option 2
              if (LowToday > LowTomorrow) {
                # Add Sell transaction tomorrow at today's low price
                addTxn(
                  portfolioname,
                  Symbol = instrument,
                  TxnDate = time(symbol[i + 1]),
                  TxnPrice = LowToday,
                  TxnQty = -Position,
                  TxnFees = transactionCost
                )
              }
            }
          } else {
            # Check Rule 4
            if (i == nrow(symbol) - 1) {
              # Add Sell transaction for the last day at the close price
              addTxn(
                portfolioname,
                Symbol = instrument,
                TxnDate = time(symbol[i + 1]),
                TxnPrice = as.numeric(symbol[i, "Close"]),
                TxnQty = -Position,
                TxnFees = transactionCost
              )
            }
          }
        }
      }
    }

    updatePortf(portfolioname, Symbols = instrument, Dates = CurrentDate)
    updateAcct(accountname, Dates = CurrentDate)
    updateEndEq(accountname, CurrentDate)

  } # End Bar-by-bar processing
} # End for loop for multiple instruments
```

Part C: Analysis and Reporting

Visualize original data

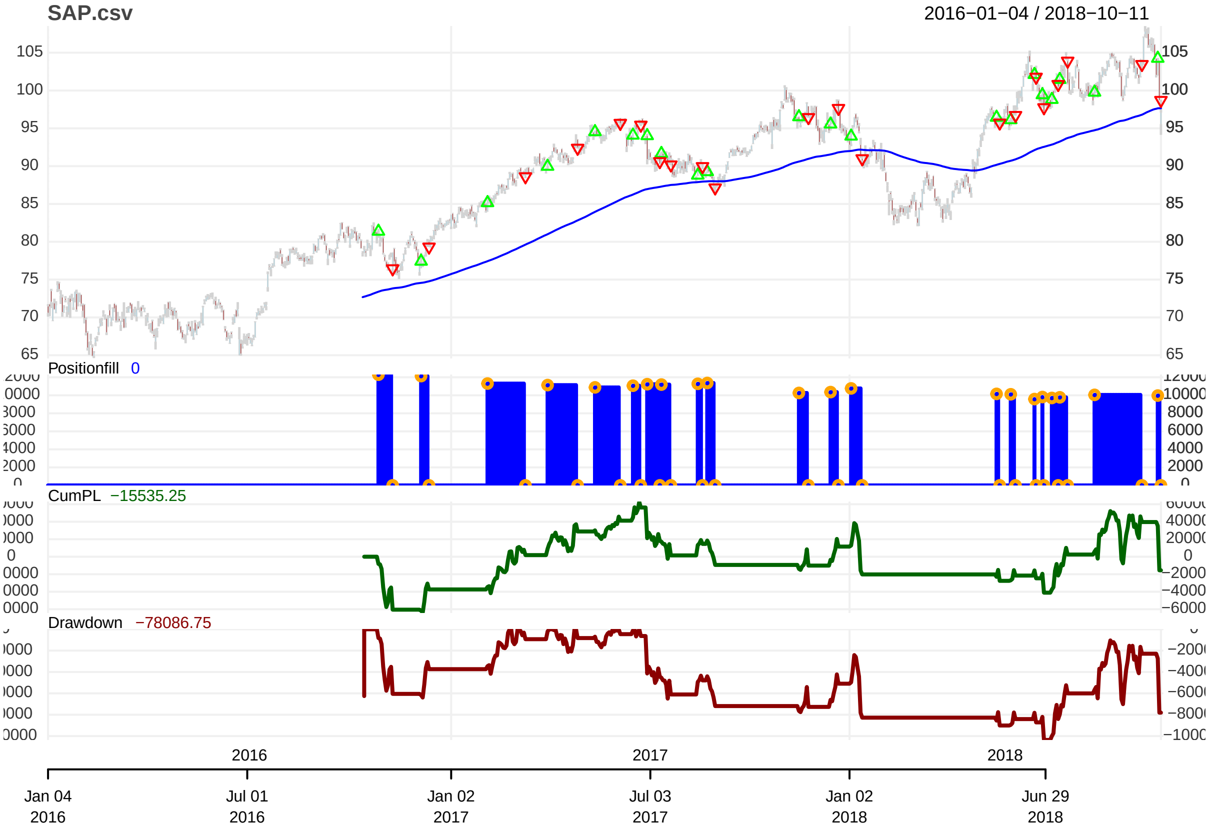
Plot of the instrument with the EMA line which indicates the general trend of the stock exponentially smoothed for the last 200 days. Moreover, the tradevolume is added below the graph.

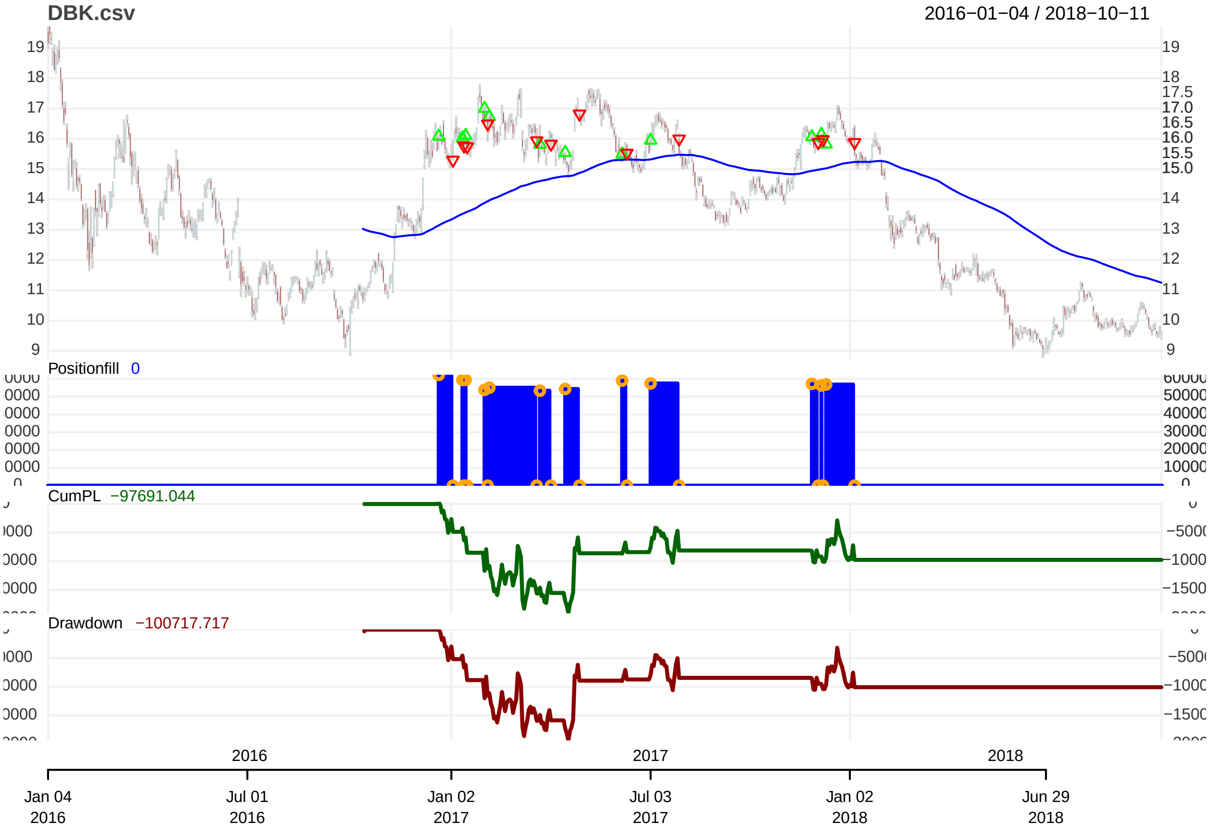




Graph which visualize the transactions

The following graph shows the combined view of the performance of the Smash Day trading system. It visualizes the trades (buy-transactions are visualized in green and sell-transactions are visualized in red). Moreover, the size of the blue squares indicates the size of the position (height) and the holding duration of the position (width). The green line shows the cummulative net profit curve, while the red curve indicates the drawdown on each day compared to the last reached high.


SAP.csv
2016−01−04 / 2018−10−11
105
100
95
90
85
80
75
70
65
Positionfill 0
12000
10000
8000
6000
4000
2000
0
CumPL −15535.25
60000
40000
20000
0
−2000
−4000
−6000
Drawdown −78086.75
0
−2000
−4000
−6000
−8000
−1000
2016
2017
2018
Jan 04 2016
Jul 01 2016
Jan 02 2017
Jul 03 2017
Jan 02 2018
Jun 29 2018



All transactions performed by the trading system

The following table can be used to get a better overview of the transactions performed and the exact details per transaction.

```
## [1] "SAP.csv"
##            Txn.Qty Txn.Price Txn.Fees  Txn.Value Txn.Avg.Cost
## 1999-12-31       0      0.00        0        0.0         0.00
## 2016-10-27   12289     81.37      -20   999955.9        81.37
## 2016-11-09  -12289     76.45      -20  -939494.1        76.45
## 2016-12-05   12136     77.41      -20   939447.8        77.41
## 2016-12-12  -12136     79.32      -20  -962627.5        79.32
## 2017-02-02   11305     85.15      -20   962620.8        85.15
## 2017-03-08  -11305     88.62      -20 -1001849.1        88.62
## 2017-03-28   11136     89.96      -20  1001794.6        89.96
## 2017-04-26  -11136     92.39      -20 -1028855.0        92.39
## 2017-05-12   10882     94.54      -20  1028784.3        94.54
## 2017-06-06  -10882     95.68      -20 -1041189.8        95.68
## 2017-06-16   11067     94.08      -20  1041183.4        94.08
## 2017-06-23  -11067     95.44      -20 -1056234.5        95.44
## 2017-06-29   11235     94.01      -20  1056202.4        94.01
## 2017-07-11  -11235     90.62      -20 -1018115.7        90.62
## 2017-07-12   11192     91.66      -20  1025858.7        91.66
## 2017-07-20  -11192     90.18      -20 -1009294.6        90.18
## 2017-08-14   11279     88.79      -20  1001462.4        88.79
## 2017-08-17  -11279     89.96      -20 -1014658.8        89.96
## 2017-08-22   11368     89.25      -20  1014594.0        89.25
## 2017-08-29  -11368     87.14      -20  -990607.5        87.14
## 2017-11-14   10264     96.51      -20   990578.6        96.51
## 2017-11-22  -10264     96.44      -20  -989860.2        96.44
## 2017-12-12   10359     95.55      -20   989802.4        95.55
## 2017-12-19  -10359     97.65      -20 -1011556.4        97.65
## 2018-01-03   10767     93.95      -20  1011559.7        93.95
## 2018-01-12  -10767     91.00      -20  -979797.0        91.00
## 2018-05-16   10159     96.44      -20   979734.0        96.44
## 2018-05-18  -10159     95.72      -20  -972419.5        95.72
## 2018-05-30   10116     96.12      -20   972349.9        96.12
## 2018-06-04  -10116     96.72      -20  -978419.5        96.72
## 2018-06-20    9583    102.10      -20   978424.3       102.10
## 2018-06-21   -9583    101.78      -20  -975357.7       101.78
## 2018-06-27    9810     99.42      -20   975310.2        99.42
## 2018-06-28   -9810     97.75      -20  -958927.5        97.75
## 2018-07-05    9705     98.80      -20   958854.0        98.80
## 2018-07-11   -9705    100.82      -20  -978458.1       100.82
## 2018-07-12    9776    101.44      -20   991677.4       101.44
## 2018-07-19   -9776    103.90      -20 -1015726.4       103.90
## 2018-08-13   10049     99.76      -20  1002488.2        99.76
## 2018-09-24  -10049    103.46      -20 -1039669.5       103.46
## 2018-10-09    9973    104.24      -20  1039585.5       104.24
## 2018-10-11   -9973     98.71      -20  -984434.8        98.71
##            Net.Txn.Realized.PL
## 1999-12-31                0.00
## 2016-10-27              -20.00
## 2016-11-09           -60481.88
## 2016-12-05              -20.00
## 2016-12-12            23159.76
## 2017-02-02              -20.00
## 2017-03-08            39208.35
## 2017-03-28              -20.00
## 2017-04-26            27040.48
## 2017-05-12              -20.00
## 2017-06-06            12385.48
## 2017-06-16              -20.00
## 2017-06-23            15031.12
## 2017-06-29              -20.00
## 2017-07-11           -38106.65
## 2017-07-12              -20.00
## 2017-07-20           -16584.16
## 2017-08-14              -20.00
## 2017-08-17            13176.43
## 2017-08-22              -20.00
## 2017-08-29           -24006.48
## 2017-11-14              -20.00
## 2017-11-22             -738.48
## 2017-12-12              -20.00
## 2017-12-19            21733.90
## 2018-01-03              -20.00
## 2018-01-12           -31782.65
## 2018-05-16              -20.00
## 2018-05-18            -7334.48
## 2018-05-30              -20.00
## 2018-06-04             6049.60
## 2018-06-20              -20.00
## 2018-06-21            -3086.56
## 2018-06-27              -20.00
## 2018-06-28           -16402.70
## 2018-07-05              -20.00
## 2018-07-11            19584.10
## 2018-07-12              -20.00
## 2018-07-19            24028.96
## 2018-08-13              -20.00
## 2018-09-24            37161.30
## 2018-10-09              -20.00
## 2018-10-11           -55170.69
## [1] "DBK.csv"
##            Txn.Qty Txn.Price Txn.Fees Txn.Value Txn.Avg.Cost
## 1999-12-31       0     0.000        0       0.0        0.000
## 2016-12-20   62177    16.083      -20  999992.7       16.083
## 2017-01-03  -62177    15.302      -20 -951432.5       15.302
## 2017-01-11   59384    16.021      -20  951391.1       16.021
## 2017-01-12  -59384    15.771      -20 -936545.1       15.771
## 2017-01-13   59449    16.110      -20  957723.4       16.110
## 2017-01-16  -59449    15.744      -20 -935965.1       15.744
## 2017-01-31   53800    17.002      -20  914707.6       17.002
## 2017-02-02  -53800    16.494      -20 -887377.2       16.494
## 2017-02-03   55030    16.735      -20  920927.0       16.735
## 2017-03-17  -55030    15.940      -20 -877178.2       15.940
## 2017-03-21   53338    15.815      -20  843540.5       15.815
## 2017-03-30  -53338    15.830      -20 -844340.5       15.830
## 2017-04-12   54314    15.545      -20  844311.1       15.545
## 2017-04-27  -54314    16.825      -20 -913833.0       16.825
## 2017-06-07   58973    15.495      -20  913786.6       15.495
## 2017-06-12  -58973    15.530      -20 -915850.7       15.530
## 2017-07-03   57418    15.950      -20  915817.1       15.950
## 2017-07-27  -57418    16.000      -20 -918688.0       16.000
## 2017-11-24   57183    16.065      -20  918644.9       16.065
## 2017-11-30  -57183    15.885      -20 -908352.0       15.885
## 2017-12-04   56260    16.145      -20  908317.7       16.145
## 2017-12-05  -56260    15.980      -20 -899034.8       15.980
## 2017-12-07   56808    15.825      -20  898986.6       15.825
## 2018-01-05  -56808    15.884      -20 -902338.3       15.884
##            Net.Txn.Realized.PL
## 1999-12-31               0.000
## 2016-12-20             -20.000
## 2017-01-03          -48580.237
## 2017-01-11             -20.000
## 2017-01-12          -14866.000
## 2017-01-13             -20.000
## 2017-01-16          -21778.334
## 2017-01-31             -20.000
## 2017-02-02          -27350.400
## 2017-02-03             -20.000
## 2017-03-17          -43768.850
## 2017-03-21             -20.000
## 2017-03-30             780.070
## 2017-04-12              -20.000
## 2017-04-27            69501.920
## 2017-06-07              -20.000
## 2017-06-12             2044.055
## 2017-07-03              -20.000
## 2017-07-27             2850.900
## 2017-11-24              -20.000
## 2017-11-30           -10312.940
## 2017-12-04              -20.000
## 2017-12-05            -9302.900
## 2017-12-07              -20.000
## 2018-01-05             3331.672
```

Performance Statistics

The following table summarizes some important trading statistics for all instruments.

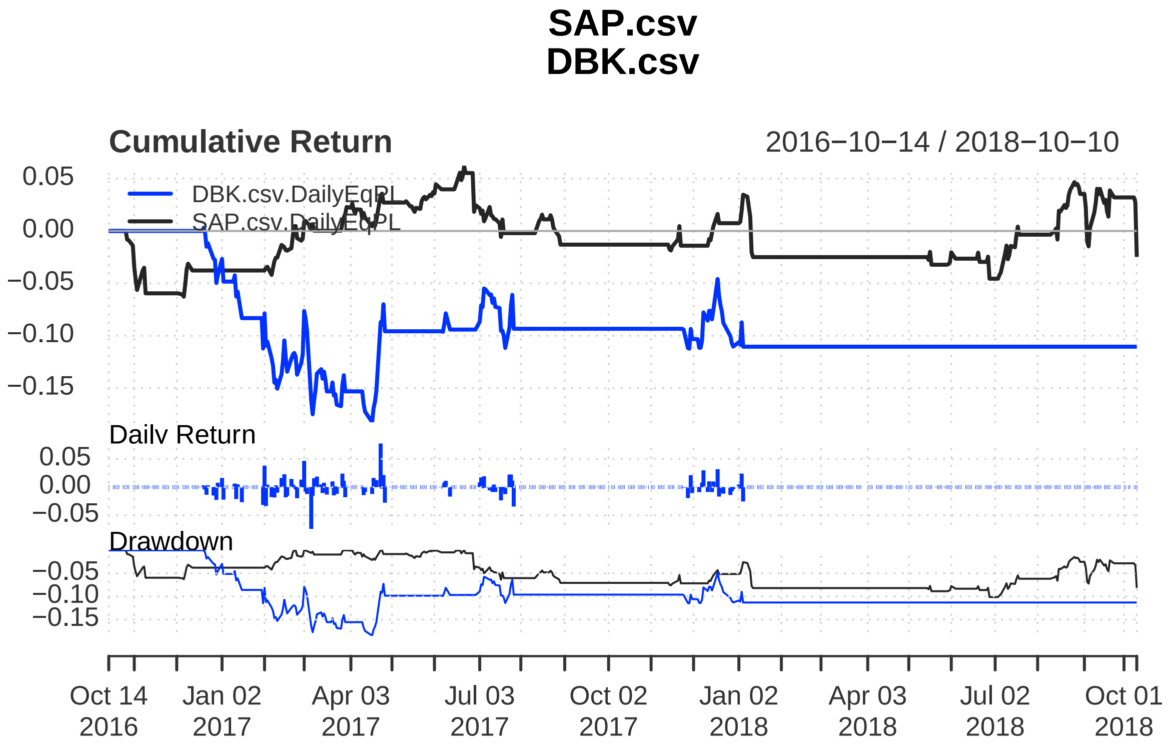
```r
tstats <- tradeStats(Portfolio=portfolioname, Symbols=instrumentlist)
for (i in 1:nrow(tstats)) {
  trades.tab <- cbind(
    c("Trades", "Win Percent", "Loss Percent","W/L Ratio"),
    c(tstats[i,"Num.Trades"],tstats[i,"Percent.Positive"],tstats[i,"Percent.Negative"],tstats[i,"Percen
  print(row.names(tstats[i,]))
  print(trades.tab)
  }
```

```
## [1] "SAP.csv"
##      [,1]           [,2]
## [1,] "Trades"       "21"
## [2,] "Win Percent"  "52.3809523809524"
## [3,] "Loss Percent" "47.6190476190476"
## [4,] "W/L Ratio"    "1.1"
## [1] "DBK.csv"
##      [,1]           [,2]
## [1,] "Trades"       "12"
## [2,] "Win Percent"  "41.6666666666667"
## [3,] "Loss Percent" "58.3333333333333"
## [4,] "W/L Ratio"    "0.714285714285714"
```

Visualize returns of the trading strategy

```r
library(PerformanceAnalytics) # contains lots of methods to investigate performance
# obtain the portfolio returns - with these you can compute virtually any financial metrics you wish
rets <- PortfReturns(Account=accountname)
rownames(rets) <- NULL # this step is important!
```

```r
charts.PerformanceSummary(rets,colorset=bluefocus,main=instrumentlist)
```



Calculate statistics of the Portfolio

```
tab.perf <- table.Arbitrary(rets, metrics=c("Return.cumulative", "Return.annualized","SharpeRatio.annual
                             metricsNames=c("Cumulative Return", "Annualized Return", "Annualized Sharp R
tab.risk <- table.Arbitrary(rets, metrics=c("StdDev.annualized", "maxDrawdown","VaR","ES"),
                             metricsNames=c("Annualized StdDev", "Max Drawdown", "Value-at-Risk","Conditi
# present the portfolio statistics
for (i in 1:ncol(tab.perf)) {
somestats <- data.frame(rownames(tab.perf),tab.perf[i,1], rownames(tab.risk),tab.risk[i,1])
colnames(rets[,i])
colnames(somestats) <- c("Performance Metric", "Performance Value", "Risk Metric", "Risk Value")
print(somestats)
}
```

```
##       Performance Metric Performance Value       Risk Metric Risk Value
## 1      Cumulative Return        -0.1104427 Annualized StdDev  0.1392235
## 2      Annualized Return        -0.1104427      Max Drawdown  0.1392235
## 3 Annualized Sharp Ratio        -0.1104427     Value-at-Risk  0.1392235
## 4           Calmar Ratio        -0.1104427   Conditional VaR  0.1392235
##       Performance Metric Performance Value       Risk Metric Risk Value
## 1      Cumulative Return       -0.05683653 Annualized StdDev  0.1849115
## 2      Annualized Return       -0.05683653      Max Drawdown  0.1849115
## 3 Annualized Sharp Ratio       -0.05683653     Value-at-Risk  0.1849115
## 4           Calmar Ratio       -0.05683653   Conditional VaR  0.1849115
```

Compare with Buy and Hold Strategy

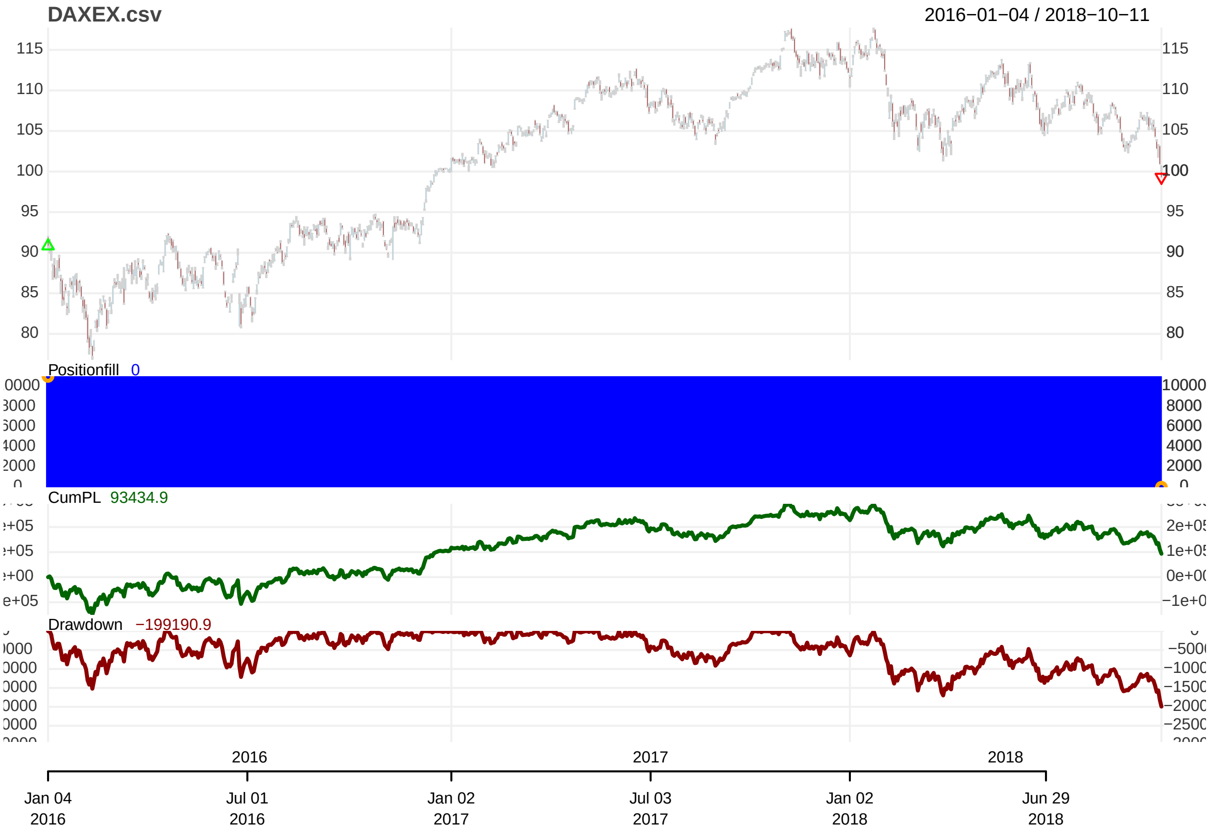
In order to compare the trading strategy properly we need to define a benchmark against which we can measure the results. In this case a simple buy and hold strategy is used. At the first date of the trading period we place a buy order and sell our position at the last day of the selected period. In order to do this we create a new Portfolio and a new Account.

```
# We remove any objects, in case there was a buyhold portfolio initialized before
suppressWarnings(try(rm(list=c("account.buyhold","portfolio.buyhold"),pos=.blotter)))

# The Buy and hold symbol is loaded
LoadCourseFile(BuyHoldDirectory,BuyHoldInstrument,debugme=TRUE,dates=daterange)
# The Buy and hold instrument is initialized
stock(BuyHoldInstrument,currency="EUR")

BuyHoldSymbol<-get(BuyHoldInstrument)

# The portfolio and account "buyhold" is initialized
initPortf("buyhold",BuyHoldInstrument,initDate=initdate,currency="EUR")
initAcct("buyhold",portfolios="buyhold",initDate=initdate,initEq=startCapital,currency="EUR")

# The first date of the defined daterange is selected
currentdate <- first(time(BuyHoldSymbol))

# The close price at this date is selected
closeprice <- as.numeric(Cl(BuyHoldSymbol[currentdate,]))

# Calculate the unitsize we can buy with our startingcapital
unitsize <- as.numeric(trunc(startCapital/closeprice))

# Place the transaction for the instrument at the first date
addTxn("buyhold",Symbol=BuyHoldInstrument,TxnDate=currentdate,TxnPrice=closeprice,TxnQty=unitsize,Txn

# Select the last date of the daterange period
lastdate <-last(time(BuyHoldSymbol))

# Select the price at the last date
lastprice <- as.numeric(Cl(BuyHoldSymbol[lastdate,]))

# Sell the position at the last date of the daterange
addTxn("buyhold",Symbol=BuyHoldInstrument,TxnDate=lastdate,TxnPrice=lastprice,TxnQty=-unitsize,TxnFees

# update portfolio and account
updatePortf(Portfolio="buyhold")
updateAcct(name="buyhold")
updateEndEq(Account="buyhold")
```

Visualize the Buy and Hold strategy

We can see that we hold the position from the first until the last date. The cumulative profits are visualized by the green line.

```r
chart.Posn("buyhold",Symbol=BuyHoldInstrument, theme=myTheme)
```



Compare the returns of the trading strategy with the buy and hold strategy

In order to compare the results of both strategies we calculat the returns for the buy and hold strategy and combine them with the returns of the trading strategy which were calculatet before.

```r
rets.bh <- PortfReturns(Account='buyhold')
returns <- cbind(rets,rets.bh)
#rulecol <- paste(portfolioname,instrument,sep="-")
#colnames(returns) <- c(rulecol,"Buy-and-hold")
```

We compare the two strategies by showing some statistical metrics of the returns and plot the returns in one chart to directly compare the performance of the strategies.

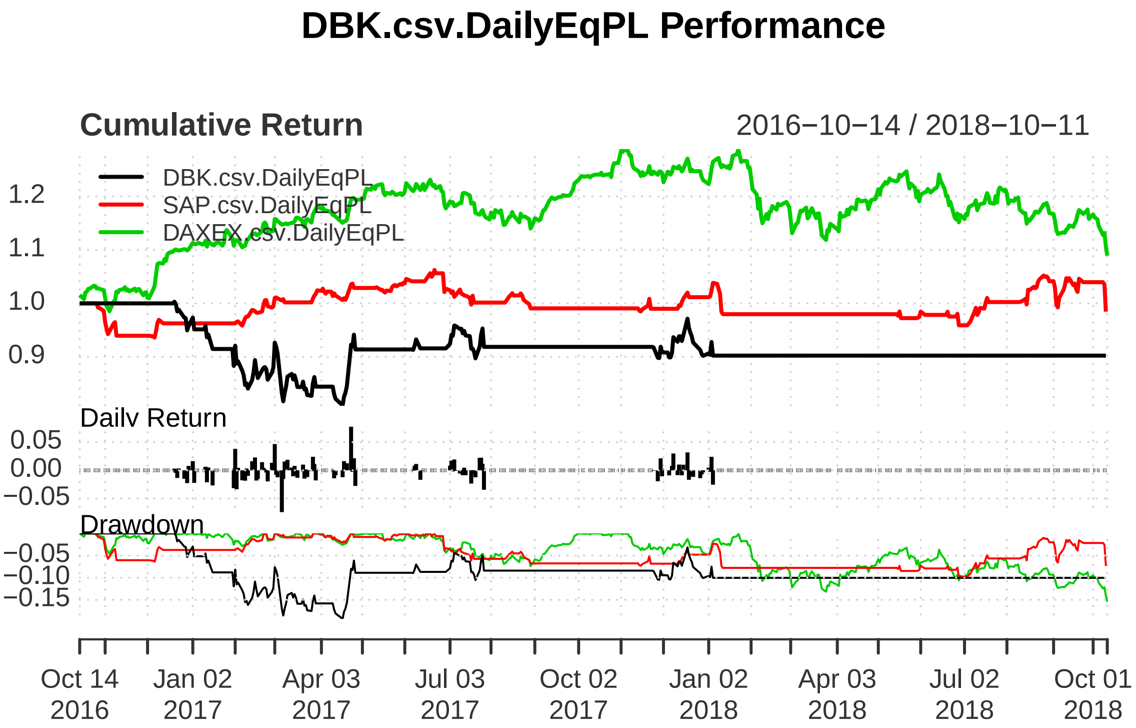
```r
table.Stats(returns)
```

```
##                  DBK.csv.DailyEqPL SAP.csv.DailyEqPL DAXEX.csv.DailyEqPL
## Observations              504.0000          504.0000            705.0000
## NAs                       201.0000          201.0000              0.0000
## Minimum                    -0.0742           -0.0508             -0.0664
## Quartile 1                  0.0000            0.0000             -0.0053
## Median                      0.0000            0.0000              0.0007
## Arithmetic Mean            -0.0002            0.0000              0.0001
## Geometric Mean             -0.0002            0.0000              0.0001
## Quartile 3                  0.0000            0.0000              0.0062
## Maximum                     0.0774            0.0277              0.0384
## SE Mean                    0.0004            0.0003              0.0004
## LCL Mean (0.95)           -0.0010           -0.0006             -0.0007
## UCL Mean (0.95)            0.0006            0.0005              0.0009
## Variance                   0.0001            0.0000              0.0001
## Stdev                      0.0088            0.0061              0.0106
## Skewness                   0.3774           -1.7733             -0.4098
## Kurtosis                  25.7194           16.8851              2.4233
```

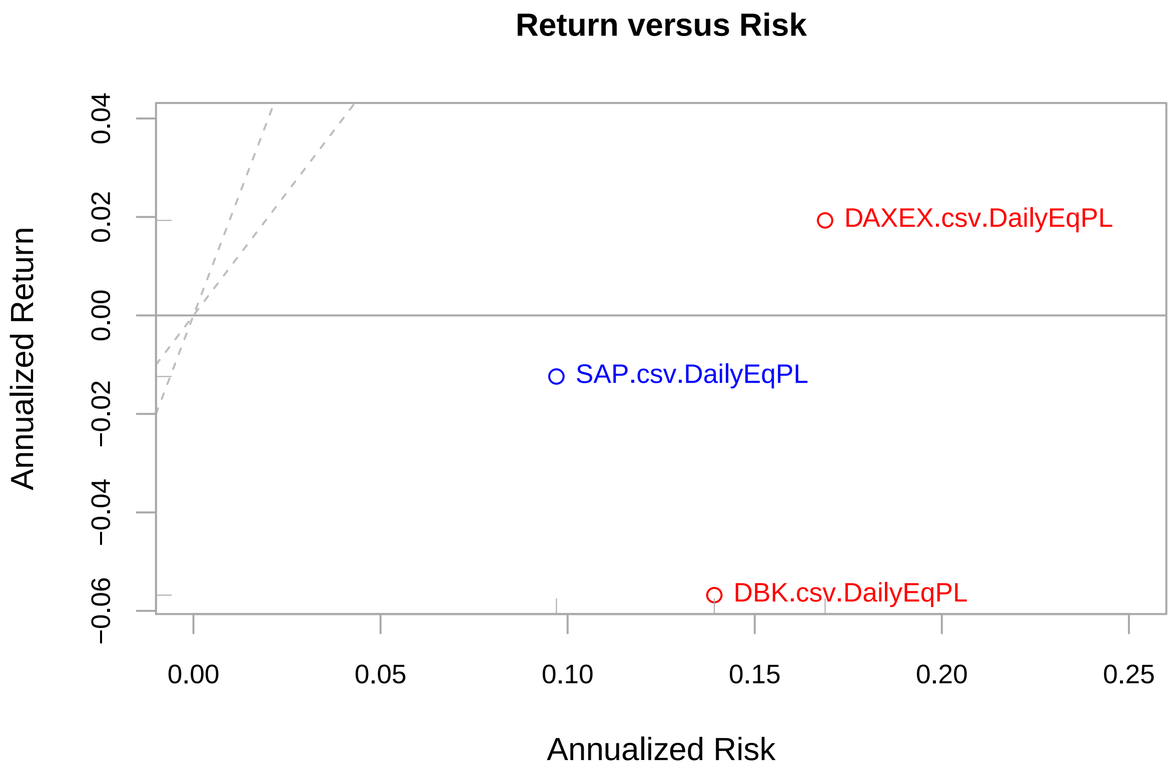
```r
table.AnnualizedReturns(returns)
```

```
##                           DBK.csv.DailyEqPL SAP.csv.DailyEqPL
## Annualized Return                   -0.0568           -0.0124
## Annualized Std Dev                   0.1392            0.0970
## Annualized Sharpe (Rf=0%)           -0.4082           -0.1280
##                           DAXEX.csv.DailyEqPL
## Annualized Return                      0.0193
## Annualized Std Dev                     0.1688
## Annualized Sharpe (Rf=0%)              0.1145
```

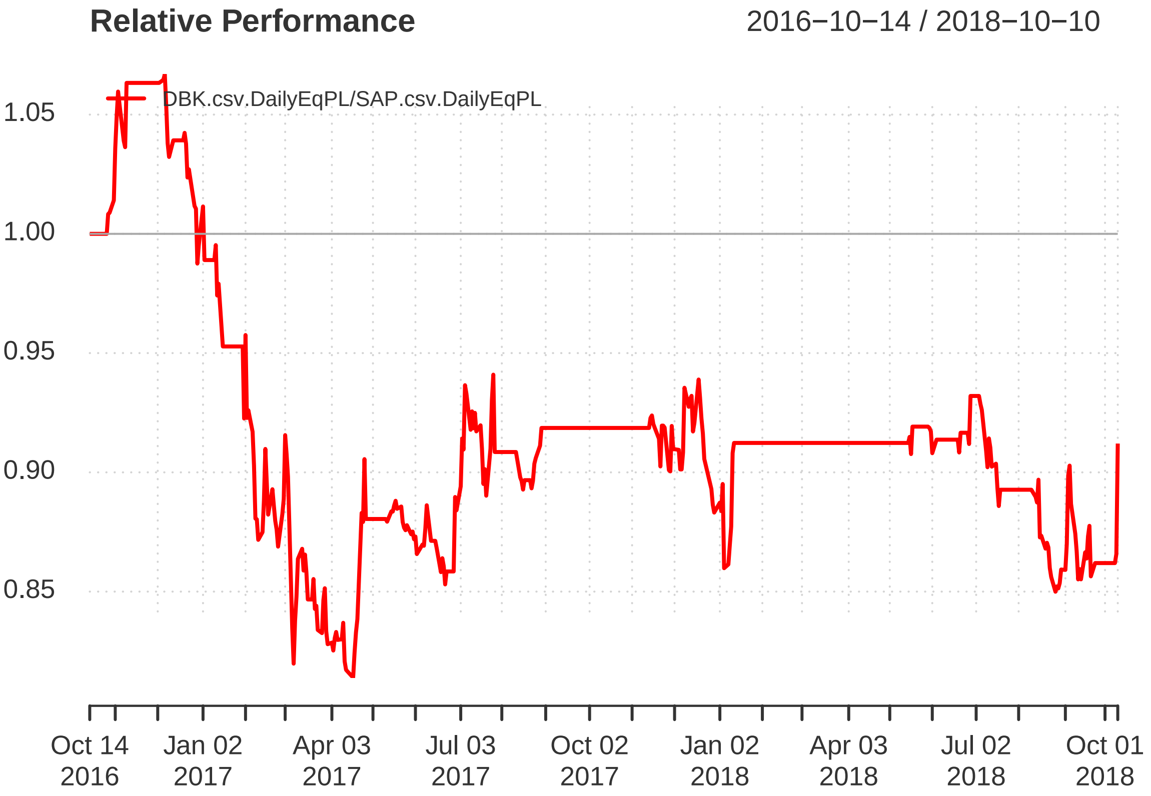
```r
charts.PerformanceSummary(returns,geometric=FALSE,wealth.index=TRUE)
```



```r
chart.RiskReturnScatter(returns,Rf=0,add.sharpe=c(1,2),xlim=c(0,0.25),main="Return versus Risk",colorse
```



```
chart.RelativePerformance(returns[,1],returns[,2],colorset=c("red","blue"),lwd=2,legend.loc="topleft")
```



Backtest

Approach

Implementation

System Check

To make sure that our system worked properly some transactions were observed to make sure that the trades were performed according to the defined rules. Prepare the chart. Check 4 transactions.

Visualize the charts with the transactions which should be checked

Analysis

Returns

Historical VaR

Equity curve

Conclusion and Suggestions