COMPUTATIONAL FINANCE & RISK MANAGEMENT

UNIVERSITY *of* WASHINGTON

Department of Applied Mathematics

Introduction to Trading Systems

Guy Yollin

Applied Mathematics
University of Washington

Outline

1. Run-time parameter passing (BBands strategy)

2. Order sizing
   - Fixed-dollar order sizing (MACD strategy)
   - Max position order sizing (BBands strategy)
   - Percent equity rebalancing (Faber strategy)

Lecture references

- TradeAnalytics project page on R-forge:
  `http://r-forge.r-project.org/projects/blotter/`
  - documents and demos for:
    - blotter package
    - quantstrat package

- Using quantstrat by Jan Humme & Brian Peterson
  `http://www.rinfinance.com/agenda/2013/workshop/Humme+Peterson.pdf`

- R-SIG-FINANCE:
  `https://stat.ethz.ch/mailman/listinfo/r-sig-finance`

[†] demos are located in the directory: `.../R-3.x.x/library/quantstrat/demo`

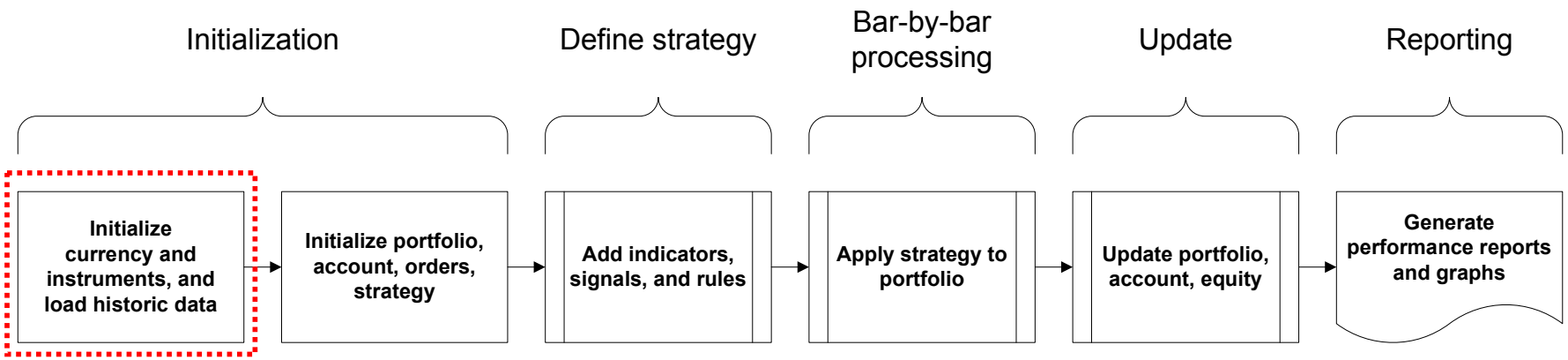
Outline

Multi-asset portfolios

9 Select Sector SPDRs that divide the S&P 500 into 9 sector index funds:

Symbol	Sector
XLY	Consumer Discretionary
XLP	Consumer Staples
XLE	Energy
XLF	Financial
XLV	Health Care
XLI	Industrial
XLB	Materials
XLK	Technology
XLU	Utilities

Initialization




```
library(quantstrat)
startDate <- '2010-01-01'   # start of data
endDate <-  '2013-07-31'    # end of data
symbols = c("XLF", "XLP", "XLE", "XLY", "XLV", "XLI", "XLB", "XLK", "XLU")
Sys.setenv(TZ="UTC")        # set time zone
```

```
getSymbols(symbols, src='yahoo', index.class=c("POSIXt","POSIXct"),
  from=startDate, to=endDate, adjust=TRUE)
```

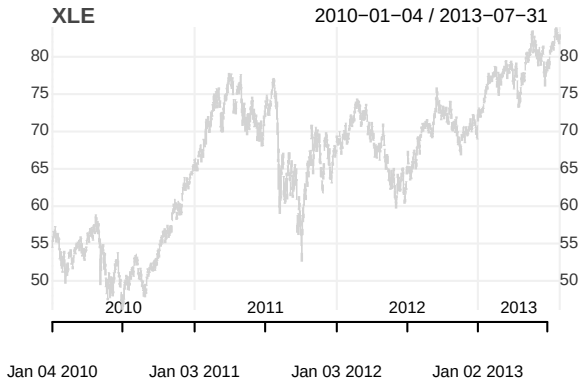
- Set time zone to UTC
- Use POSIXct index class

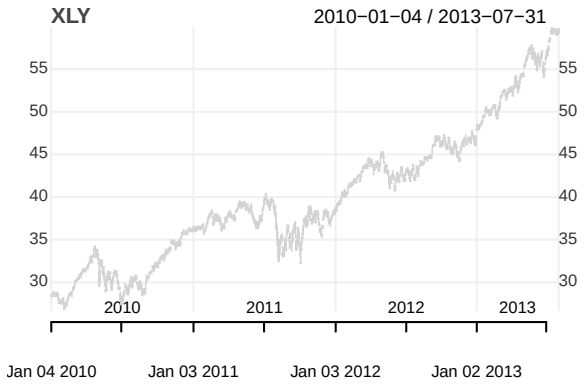
Plot time series of portfolio constituents

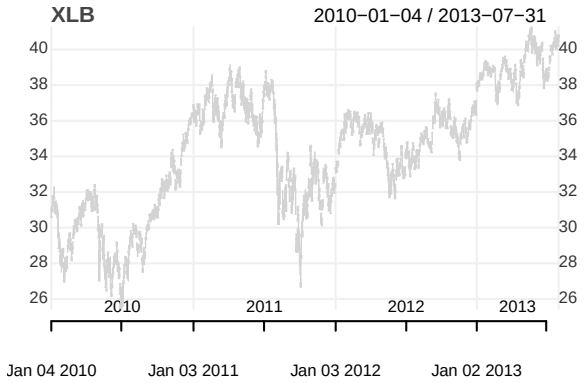
```
myTheme<-chart_theme()
myTheme$col$dn.col<-'lightblue'
myTheme$col$dn.border <- 'lightgray'
myTheme$col$up.border <- 'lightgray'
```

```
par(mfrow=c(3,3))
for(symbol in symbols)
{
  plot(chart_Series(get(symbol),name=symbol,theme=myTheme))
}
par(mfrow=c(1,1))
```

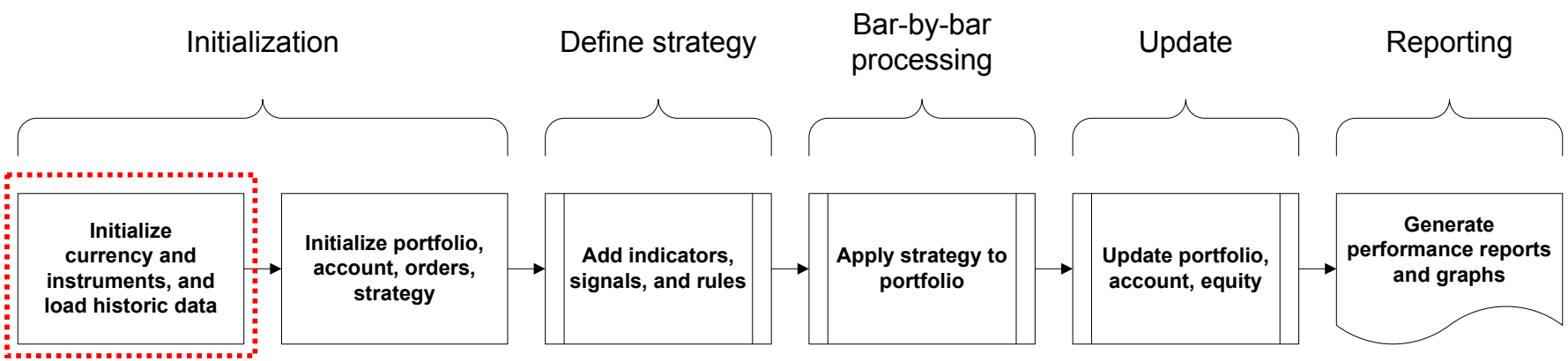
Select Sector SPDRs







Initialize instruments




```
initDate <- '2009-12-31'
initEq <- 1e6
currency("USD")
stock(symbols, currency="USD",multiplier=1)
```

- Important that portfolio, account, and orderbook initialization date be before start of data

Bollinger bands

- Bollinger bands are a volatility-sensitive price channel
- Published by John Bollinger in the early 1980s
- RSI Calculation
  - MA = simple moving average (typically 20 days) of the weighted-close
  - Upper Band = MA + N $\times$ StdDev(C)
  - Lower Band = MA - N $\times$ StdDev(C)
  - N typically in the range of 2 to 3
- Interpretation
  - Trade channel reversals between the upper and lower bands
  - Trade channel break-outs above/below the bands

Long-short Bollinger Band reversal strategy

Buy rule:

- Buy long when the close crosses below the lower band

Sell rule:

- Sell short when the close crosses above the upper band

Exit rule:

- Exit any long or short position when either the high or low cross the mid-line

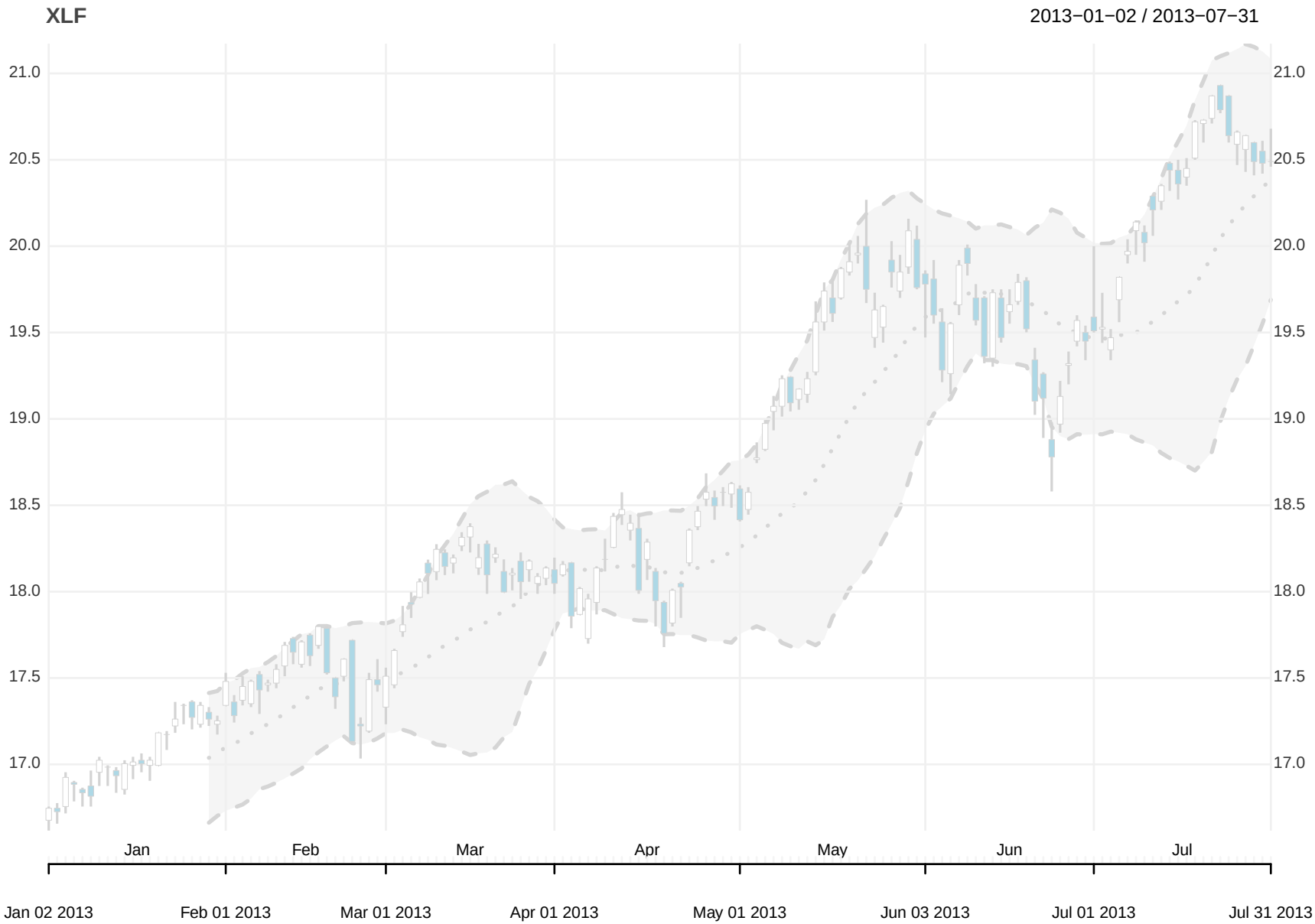
Pyramiding:

- Multiple orders in the same direction

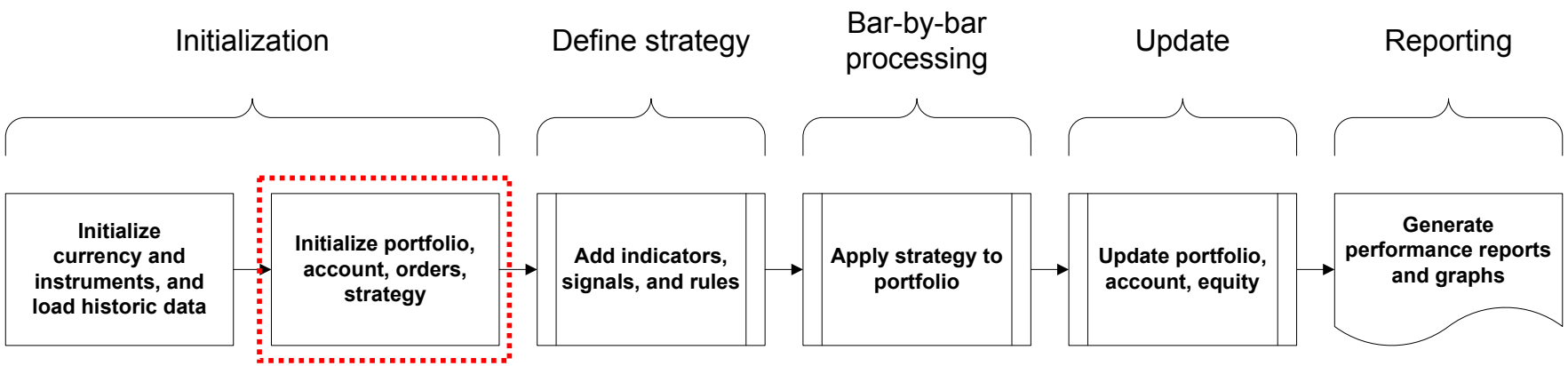
Calculate and plot Bollinger bands

```
args(BBands)

## function (HLC, n = 20, maType, sd = 2, ...)
## NULL

b <- BBands(HLC=HLC(XLF["2013"]), n=20, sd=2)
tail(b)

##                    dn      mavg        up       pctB
## 2013-07-24 19.123830 20.108333 21.092837 0.80218285
## 2013-07-25 19.239799 20.173167 21.106534 0.72865227
## 2013-07-26 19.325655 20.225167 21.124679 0.69167797
## 2013-07-29 19.446850 20.278000 21.109150 0.63354987
## 2013-07-30 19.532210 20.319667 21.107124 0.61662013
## 2013-07-31 19.656353 20.368500 21.080647 0.62275089

chart_Series(XLF["2013"],TA='add_BBands(lwd=2)',theme=myTheme,name="XLF")
```

Bollinger bands

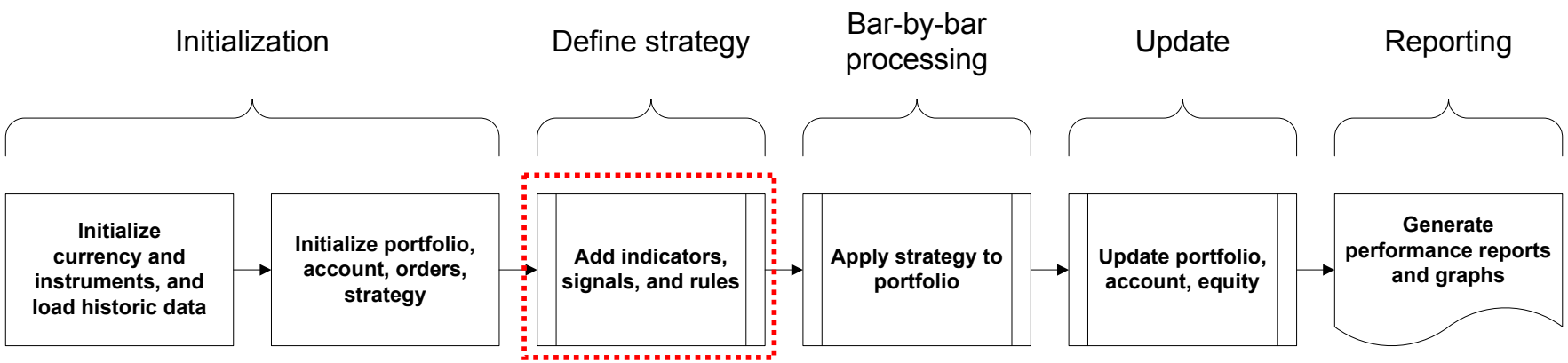


Initialize portfolio, account, and orders object




```
rm.strat("multiAsset.bb1") # remove portfolio, account, orderbook if re-run
initPortf(name="multiAsset.bb1", symbols, initDate=initDate)
initAcct(name="multiAsset.bb1", portfolios="multiAsset.bb1",
  initDate=initDate, initEq=initEq)
initOrders(portfolio="multiAsset.bb1", initDate=initDate)
```

Define indicators



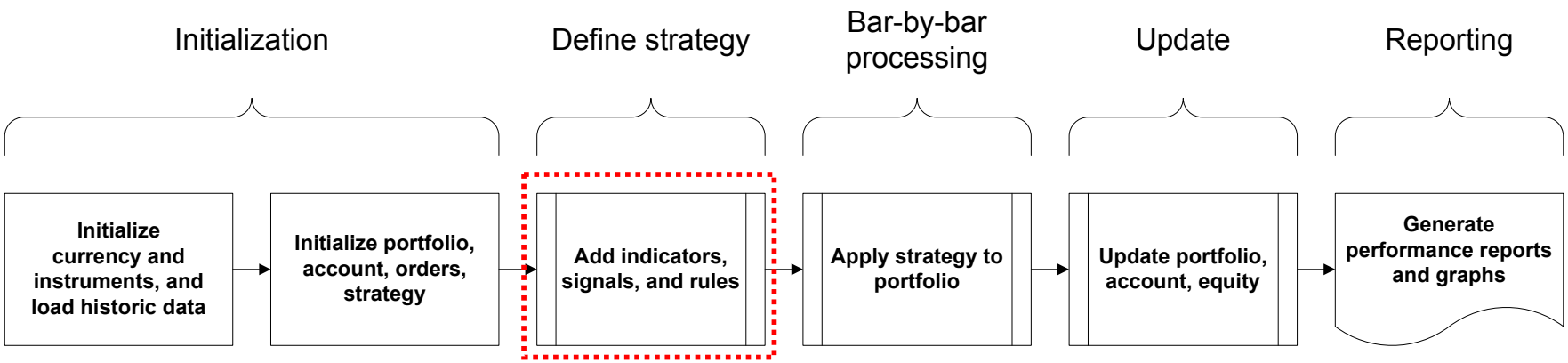

```
strategy("bbands", store=TRUE)
```

```
args(BBands)

## function (HLC, n = 20, maType, sd = 2, ...)
## NULL
```

```
add.indicator("bbands", name = "BBands",
  arguments = list(HLC = quote(HLC(mktdata)), maType='SMA'), label='BBands')
```

- Note that n and sd are not included in the indicator arguments list

Define signals

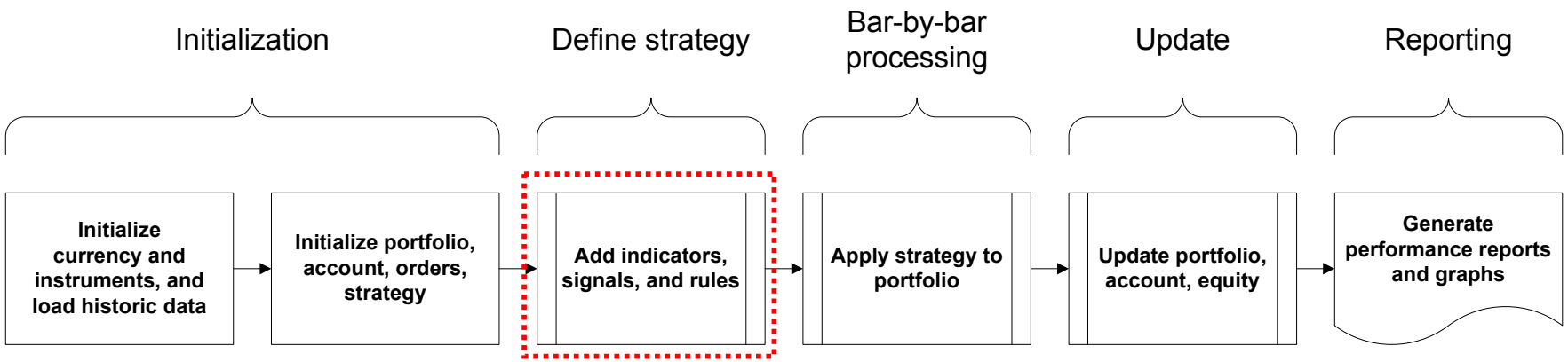



```
add.signal("bbands", name="sigCrossover",
  arguments=list(columns=c("Close","up"),relationship="gt"),
  label="Cl.gt.UpperBand")
```

```
add.signal("bbands", name="sigCrossover",
  arguments=list(columns=c("Close","dn"),relationship="lt"),
  label="Cl.lt.LowerBand")
```

```
add.signal("bbands", name="sigCrossover",
  arguments=list(columns=c("High","Low","mavg"),relationship="op"),
  label="Cross.Mid")
```

Add rules



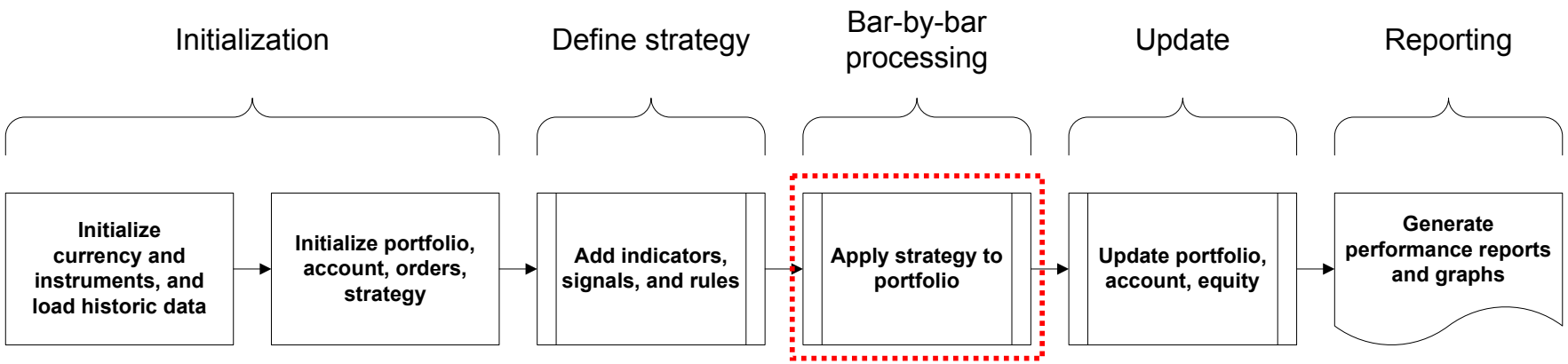

```
add.rule("bbands", name='ruleSignal',
  arguments=list(sigcol="Cl.gt.UpperBand",sigval=TRUE, orderqty=-100,
  ordertype='market', orderside=NULL),type='enter')
```

```
add.rule("bbands", name='ruleSignal',
  arguments=list(sigcol="Cl.lt.LowerBand",sigval=TRUE, orderqty= 100,
  ordertype='market', orderside=NULL),type='enter')
```

```
add.rule("bbands", name='ruleSignal',
  arguments=list(sigcol="Cross.Mid",sigval=TRUE, orderqty= 'all',
  ordertype='market', orderside=NULL),type='exit')
```

- Long-short channel reversal system with pyramiding

Applying strategy to a multi-asset portfolio

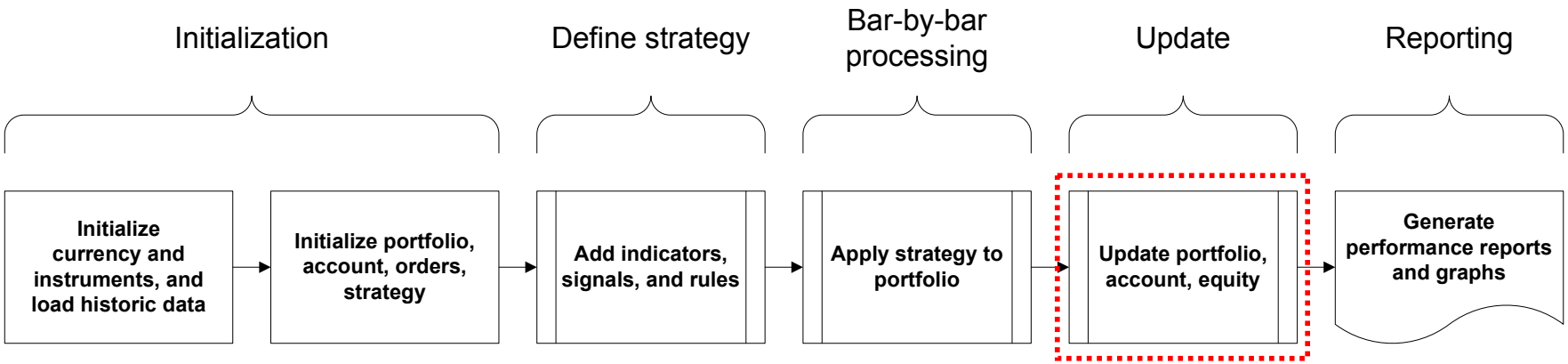



```
SD = 2
N = 20
```

```
out <- applyStrategy("bbands",
  portfolios="multiAsset.bb1",parameters=list(sd=SD,n=N))
```

- For this run the length of the moving average is 20 and the standard deviation is 2

Update portfolio and account



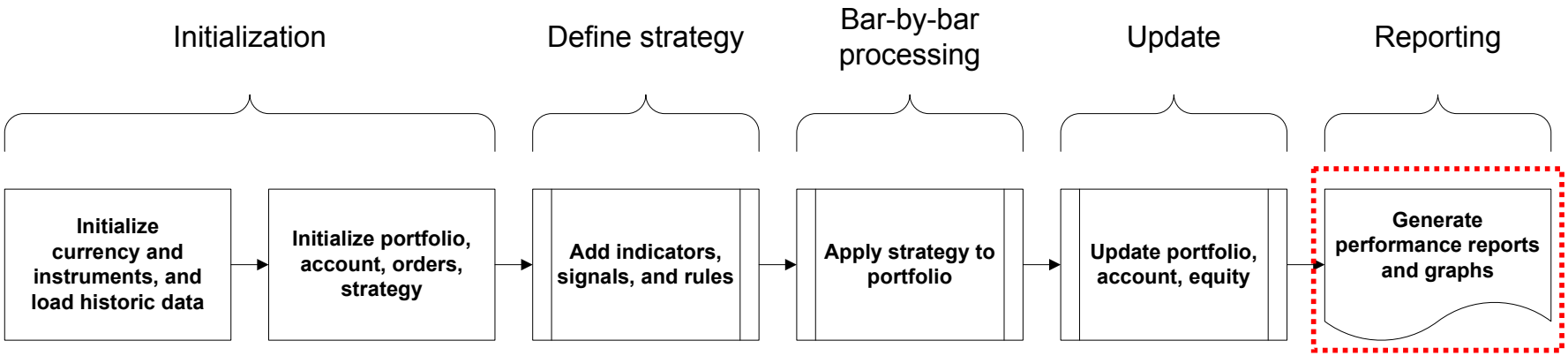

```
updatePortf("multiAsset.bb1")
updateAcct("multiAsset.bb1")
updateEndEq("multiAsset.bb1")
```

Data integrity check

```
checkBlotterUpdate <- function(port.st,account.st,verbose=TRUE)
{
  ok <- TRUE
  p <- getPortfolio(port.st)
  a <- getAccount(account.st)
  syms <- names(p$symbols)
  port.tot <- sum(sapply(syms,FUN = function(x) eval(parse(
    text=paste("sum(p$symbols",x,"posPL.USD$Net.Trading.PL)",sep="$")))))
  port.sum.tot <- sum(p$summary$Net.Trading.PL)
  if( !isTRUE(all.equal(port.tot,port.sum.tot)) ) {
    ok <- FALSE
    if( verbose )
      print("portfolio P&L doesn't match sum of symbols P&L")
  }
  initEq <- as.numeric(first(a$summary$End.Eq))
  endEq <- as.numeric(last(a$summary$End.Eq))
  if( !isTRUE(all.equal(port.tot,endEq-initEq)) ) {
    ok <- FALSE
    if( verbose )
      print("portfolio P&L doesn't match account P&L")
  }
  if( sum(duplicated(index(p$summary))) ) {
    ok <- FALSE
    if( verbose )
      print("duplicate timestamps in portfolio summary")
  }
  if( sum(duplicated(index(a$summary))) ) {
    ok <- FALSE
    if( verbose )
      print("duplicate timestamps in account summary")
  }
  return(ok)
}
checkBlotterUpdate("multiAsset.bb1","multiAsset.bb1")
```

```
## [1] TRUE
```

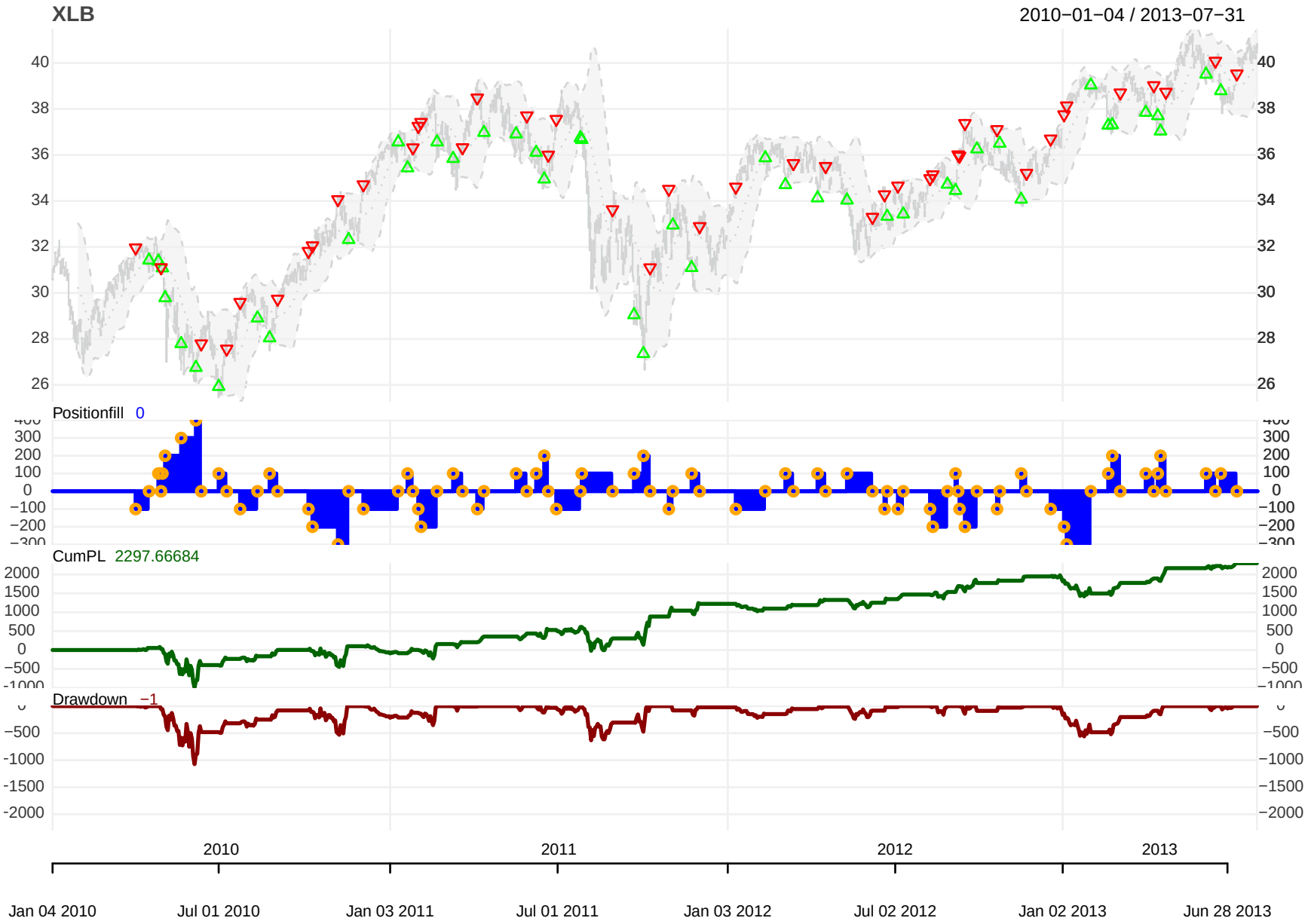
Chart performance for one of the assets



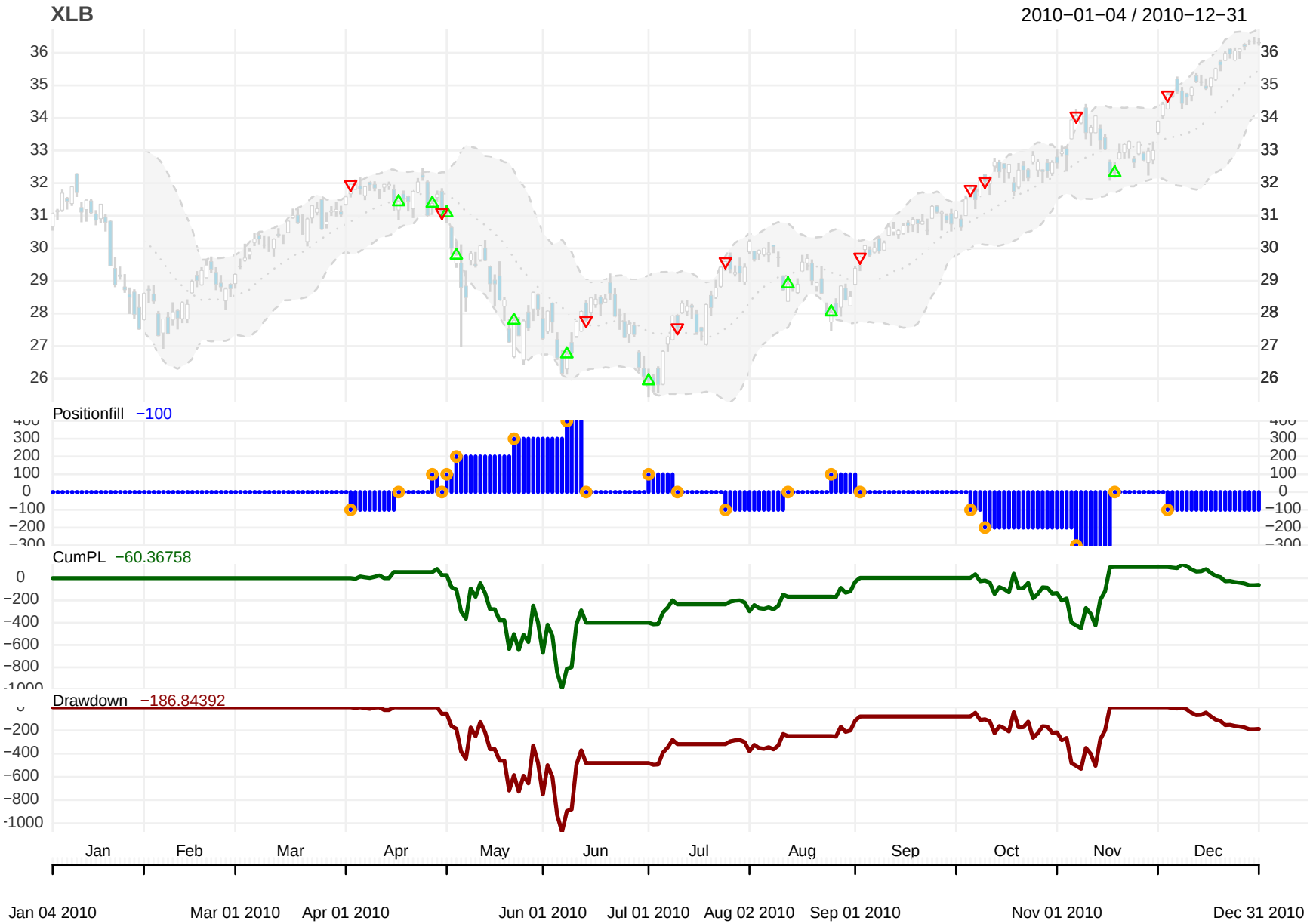

```
chart.Posn("multiAsset.bb1","XLB",TA="add_BBands(n=20,sd=2)",theme=myTheme)
```

```
chart.Posn("multiAsset.bb1","XLB",TA="add_BBands(n=20,sd=2)",
  Dates="2010",theme=myTheme)
```

BBands strategy for XLB with n=20 and sd=2



BBands strategy for XLB with n=20 and sd=2



Initialize portfolio, account, and orders object

```
rm.strat("multiAsset.bb2") # remove portfolio, account, orderbook if re-run
initPortf(name="multiAsset.bb2", symbols, initDate=initDate)
initAcct(name="multiAsset.bb2", portfolios="multiAsset.bb2",
  initDate=initDate, initEq=initEq)
initOrders(portfolio="multiAsset.bb2", initDate=initDate)
```

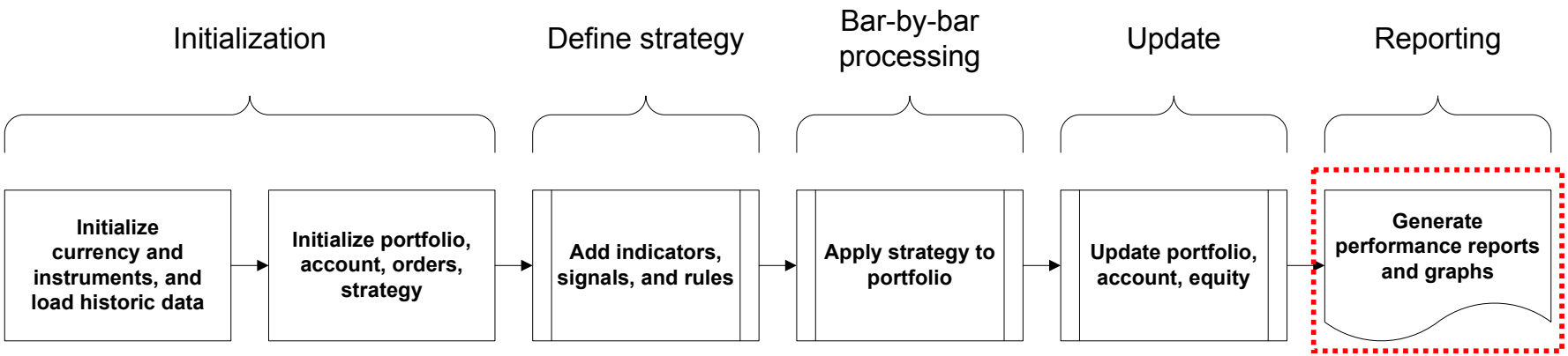
```
SD=3
out <- applyStrategy("bbands",
  portfolios="multiAsset.bb2",parameters=list(sd=SD,n=N))
```

```
updatePortf("multiAsset.bb2")
updateAcct("multiAsset.bb2")
updateEndEq("multiAsset.bb2")
```

```
checkBlotterUpdate("multiAsset.bb2","multiAsset.bb2")

## [1] TRUE
```

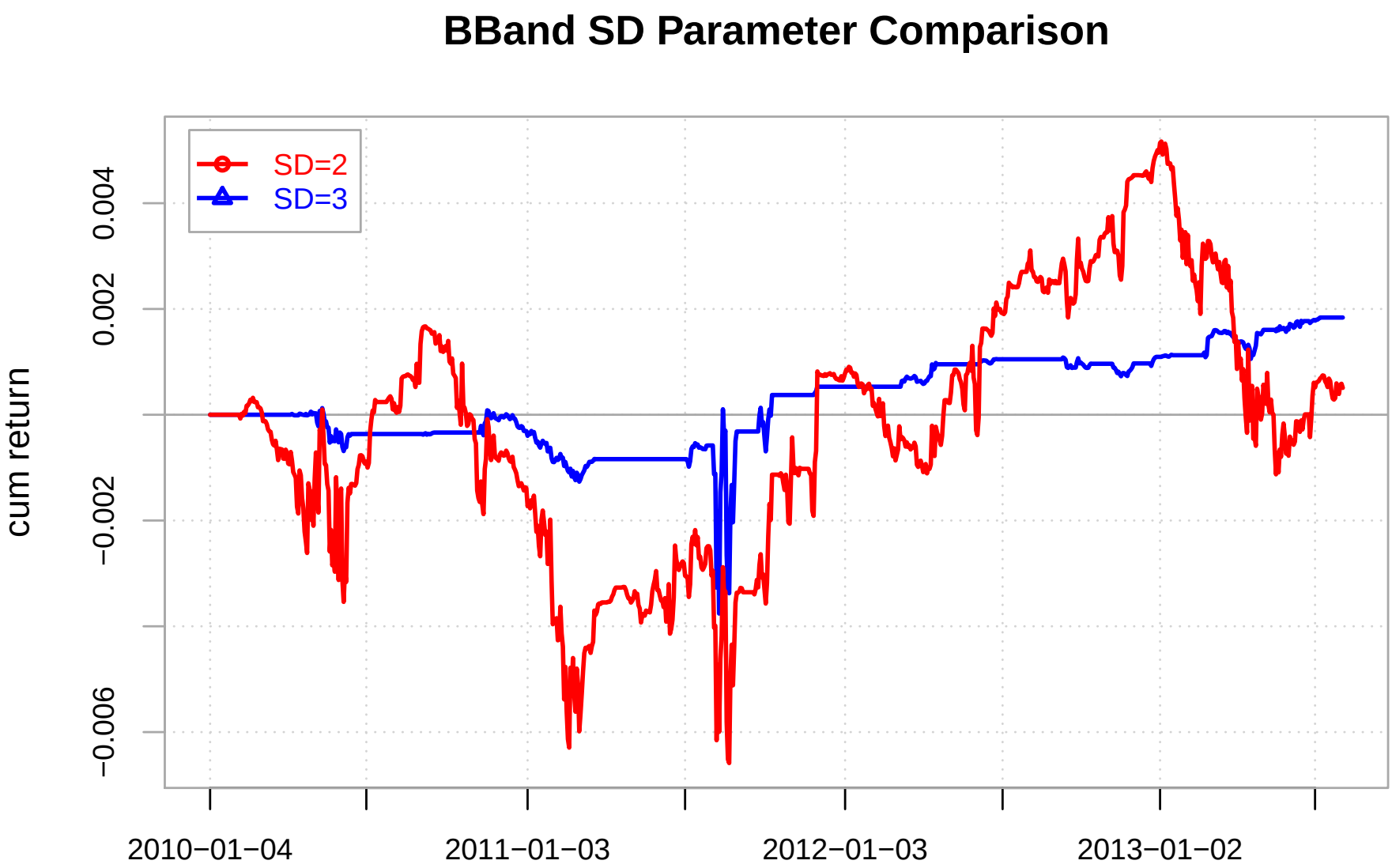
- For this run the length of the moving average is 20 and the standard deviation is 3

Compare BBand parameter settings




```
eq1 <- getAccount("multiAsset.bb1")$summary$End.Eq
rt1 <- Return.calculate(eq1,"log")
eq2 <- getAccount("multiAsset.bb2")$summary$End.Eq
rt2 <- Return.calculate(eq2,"log")
returns <- cbind(rt1,rt2)
colnames(returns) <- c("SD=2","SD=3")
chart.CumReturns(returns,colorset=c(2,4),legend.loc="topleft",
  main="BBand SD Parameter Comparison",ylab="cum return",xlab="",
  minor.ticks=FALSE)
```

Compare BBand parameter settings



Outline

1. Run-time parameter passing (BBands strategy)

2. Order sizing
   - Fixed-dollar order sizing (MACD strategy)
   - Max position order sizing (BBands strategy)
   - Percent equity rebalancing (Faber strategy)

Outline

1. Run-time parameter passing (BBands strategy)
2. Order sizing
   - Fixed-dollar order sizing (MACD strategy)
   - Max position order sizing (BBands strategy)
   - Percent equity rebalancing (Faber strategy)

The ruleSignal function

ruleSignal is the default rule to generate a trade order on a signal

```
args(ruleSignal)

## function (mktdata = mktdata, timestamp, sigcol, sigval, orderqty = 0,
##     ordertype, orderside = NULL, orderset = NULL, threshold = NULL,
##     tmult = FALSE, replace = TRUE, delay = 1e-04, osFUN = "osNoOp",
##     pricemethod = c("market", "opside", "active"), portfolio,
##     symbol, ..., ruletype, TxnFees = 0, prefer = NULL, sethold = FALSE,
##     label = "", order.price = NULL, chain.price = NULL, time.in.force = "")
## NULL
```

Main arguments:

sigcol	column name to check for signal
sigval	signal value to match
orderqty	quantity for order or 'all', modified by osFUN
ordertype	"market","limit","stoplimit","stoptrailing","iceberg"
orderside	"long", "short", or NULL
osFUN	function or name of order sizing function (default is osNoOp)

The osNoOp function

The function osNoOp is the default order sizing function

```
args(osNoOp)

## function (timestamp, orderqty, portfolio, symbol, ruletype, ...)
## NULL
```

Main arguments:

timestamp	timestamp (coercible into a POSIXct object) that will mark the time of order insertion
orderqty	the order quantity; modified by osFUN
portfolio	name of the portfolio for the order
symbol	symbol of instrument
ruletype	one of "risk", "order", "rebalance", "enter", "exit"

Fixed-dollar order sizing function

This order sizing function adjusts the share quantity such that the transaction value is approximately equal to a pre-defined *tradesize*

```
osFixedDollar <- function(timestamp,orderqty, portfolio, symbol, ruletype, ...)
{
  ClosePrice <- as.numeric(Cl(mktdata[timestamp,]))
  orderqty <- round(tradeSize/ClosePrice,-2)
  return(orderqty)
}
```

- Function retrieves the current close price and sets order quantity as follows:

$$\text{orderqty} = \frac{\text{tradeSize}}{\mathit{ClosePrice}}$$

MACD (Moving Average Convergence-Divergence)

- Trend-following momentum indicator
- Published by Gerald Appel in the late 1970
- MACD Calculation (defaults TTR calculation and values)
  - MACD = 100 * ((12-day EMA of close) / (26-day EMA of close) - 1)
  - MACD Signal Line = 9-day EMA of MACD
  - MACD histogram = MACD - Signal Line
- Interpretation
  - Buy/Sell when MACD Signal Line crosses 0

Long-only MACD momentum strategy

Buy rule:

- Buy long when the MACD signal crosses above 0

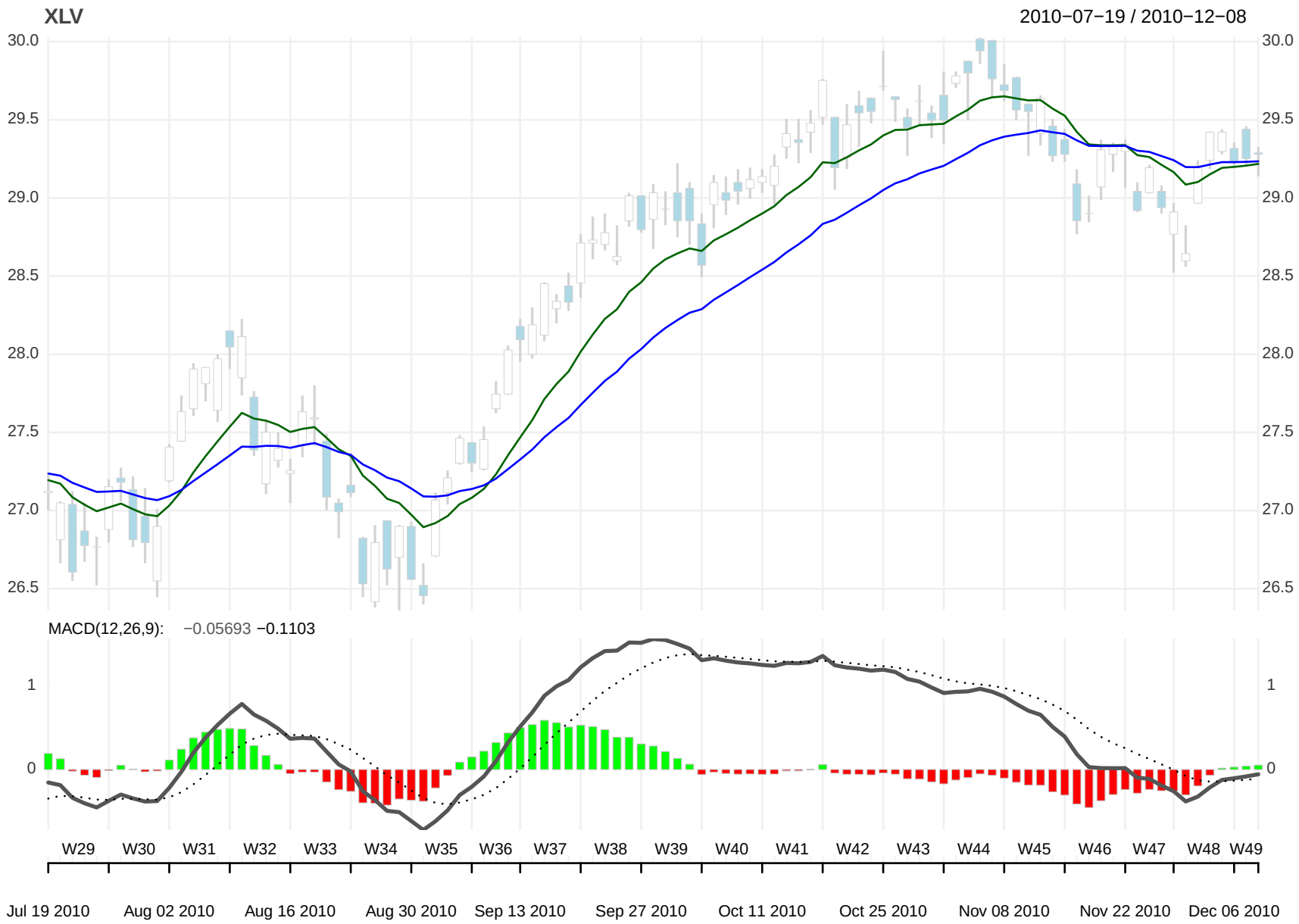
Exit rule:

- Sell when the MACD signal crosses below 0

Moving Average Convergence-Divergence (MACD)

```
args(MACD)

## function (x, nFast = 12, nSlow = 26, nSig = 9, maType, percent = TRUE,
##     ...)
## NULL

macd  <- MACD( Cl(XLV), 12, 26, 9, maType="EMA" )
tail(macd,3)

##                 macd    signal
## 2013-07-29 1.5379311 1.2886158
## 2013-07-30 1.5288323 1.3366591
## 2013-07-31 1.5070407 1.3707354

chart_Series(XLV,
  TA="add_MACD();add_EMA(12,col='darkgreen');add_EMA(26,col='blue')",
  subset="20100717::20101208",theme=myTheme)
```

Moving Average Convergence-Divergence (MACD)

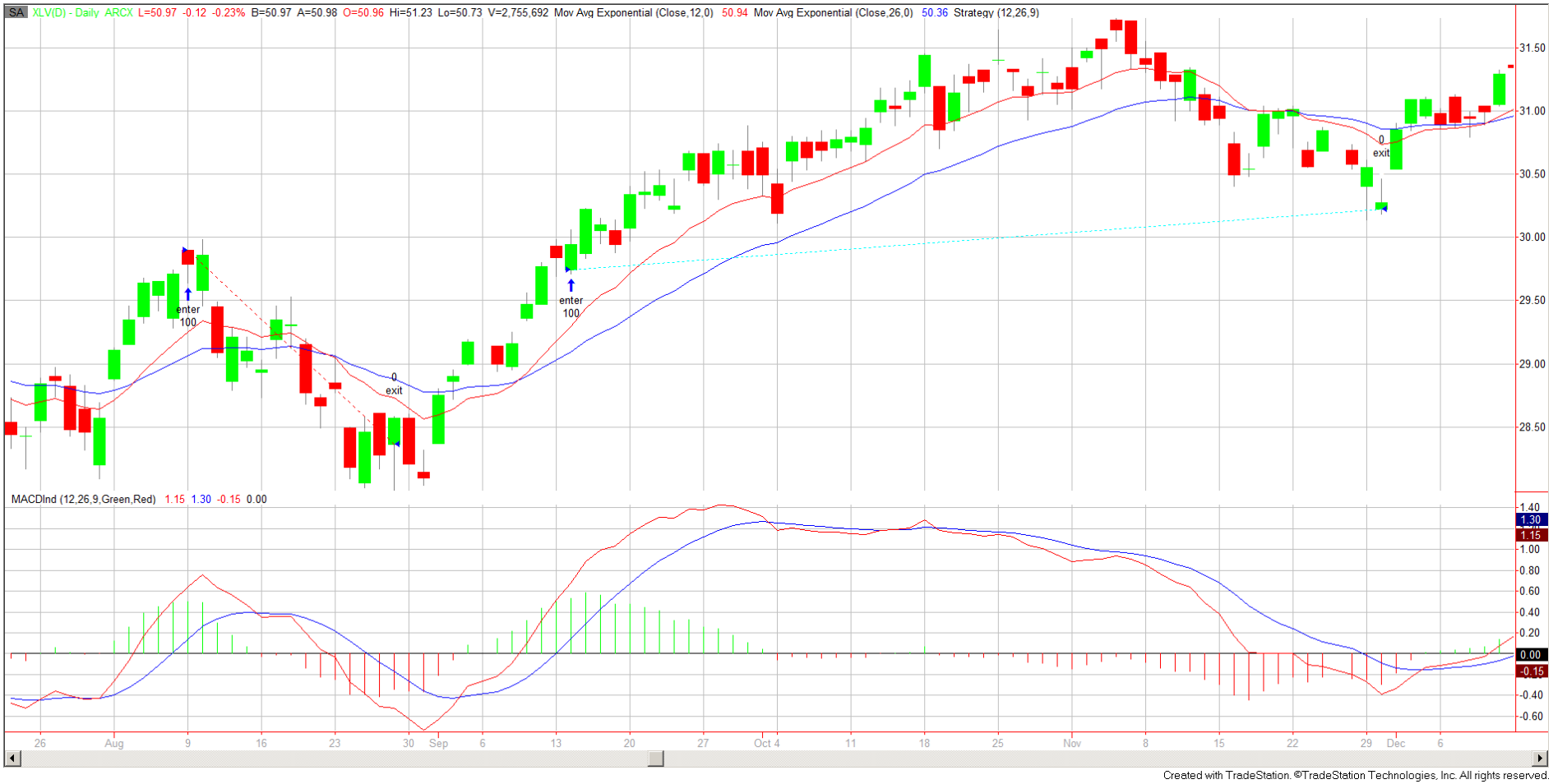


MACD code in EasyLanguage

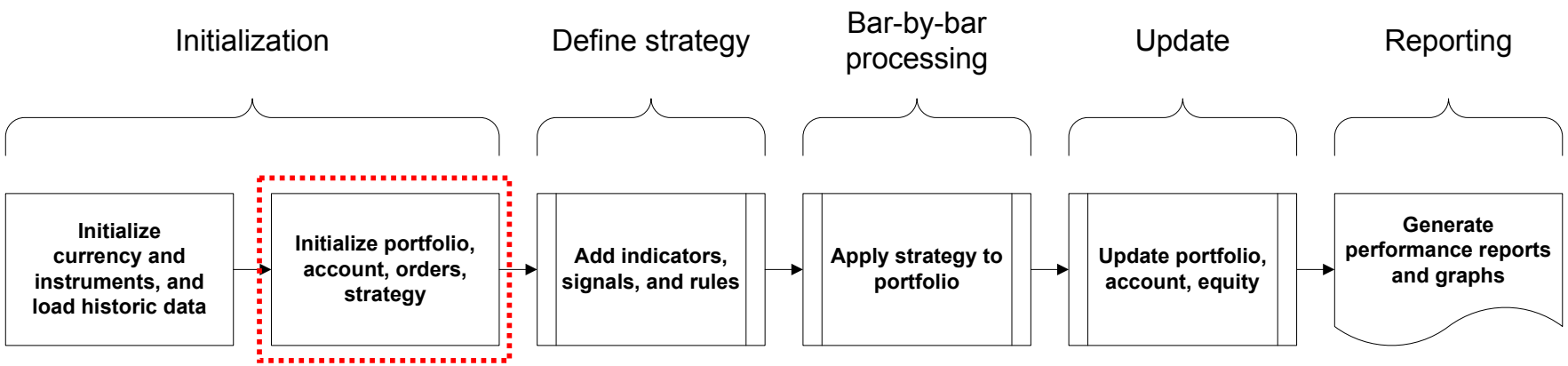
EasyLanguage code

```
inputs:
  FastLength( 12 ), SlowLength( 26 ), MACDLength( 9 ) ;

variables:
  fastMA(0), slowMA(0), myMACD(0), MACDsig(0) ;

fastMA = XAverage( Close, FastLength ) ;
slowMA = XAverage( Close, SlowLength ) ;
myMACD = 100 * (fastMA/slowMA - 1) ;
MACDsig = XAverage( myMACD, MACDLength ) ;

If MACDsig crosses above 0 then
  Buy ("enter") next bar at market ;

If MACDsig crosses below 0 then
  Sell ("exit") next bar at market ;
```

MACD system in TradeStation

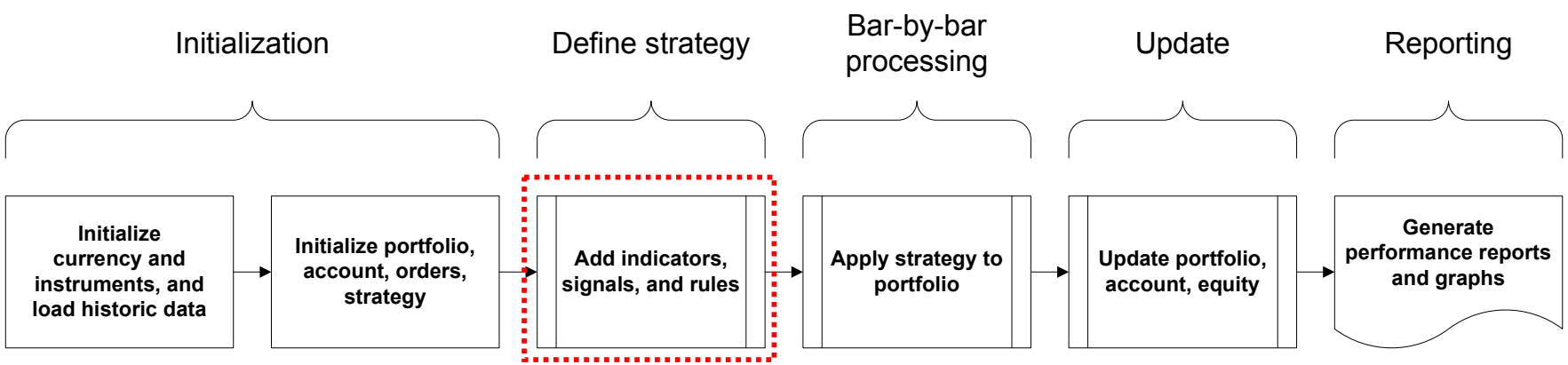


Initialize portfolio, account, and orders object




```
rm.strat("multi.macd") # remove portfolio, account, orderbook if re-run
initPortf(name="multi.macd", symbols, initDate=initDate)
initAcct(name="multi.macd", portfolios="multi.macd",
  initDate=initDate, initEq=initEq)
initOrders(portfolio="multi.macd", initDate=initDate)
```

Define indicators and signals



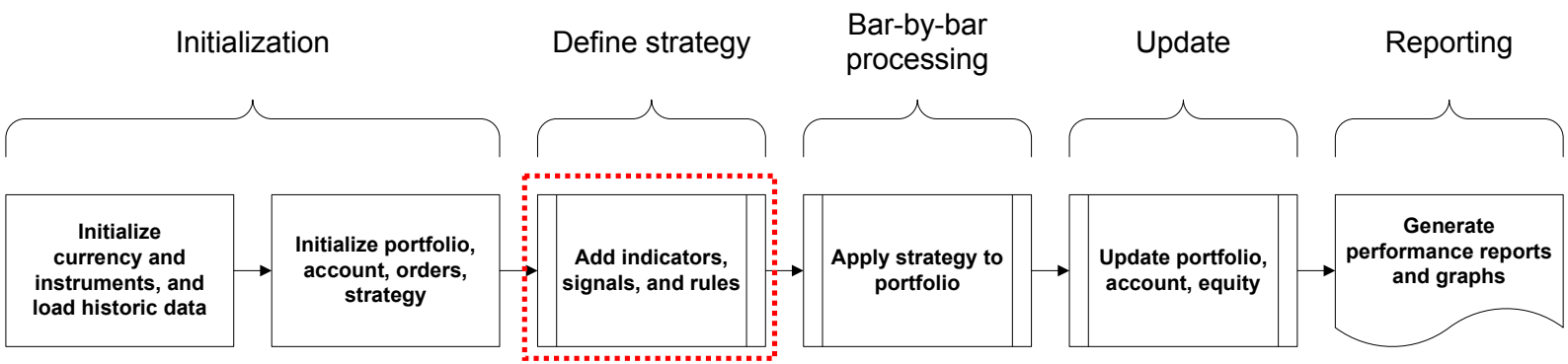

```
strategy("macd", store=TRUE)
```

```
add.indicator("macd", name = "MACD",
  arguments = list(x=quote(Cl(mktdata))),label='osc')
```

```
add.signal("macd",name="sigThreshold",
  arguments=list(column="signal.osc",relationship="gt",threshold=0,cross=TRUE),
  label="signal.gt.zero")
```

```
add.signal("macd",name="sigThreshold",
  arguments=list(column="signal.osc",relationship="lt",threshold=0,cross=TRUE),
  label="signal.lt.zero")
```

Add rules with an order sizing function specified

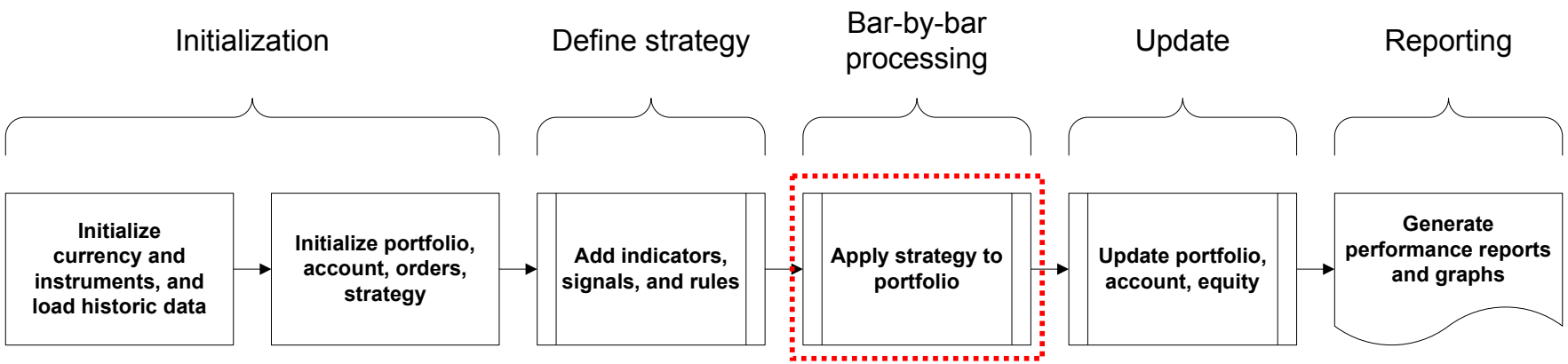



```
add.rule("macd",name='ruleSignal',
  arguments = list(sigcol="signal.gt.zero",sigval=TRUE,orderqty=100,
  ordertype='market',orderside='long',osFUN='osFixedDollar'),
  type='enter',label='enter',storefun=FALSE)
```

```
add.rule("macd",name='ruleSignal',
  arguments = list(sigcol="signal.lt.zero",sigval=TRUE,orderqty='all',
  ordertype='market',orderside='long'),
  type='exit',label='exit')
```

- The `ruleSignal` argument `osFUN` is now set to the name of our custom order sizing function for the entry rule
- The argument `orderqty` is set to 'all' for the exit rule

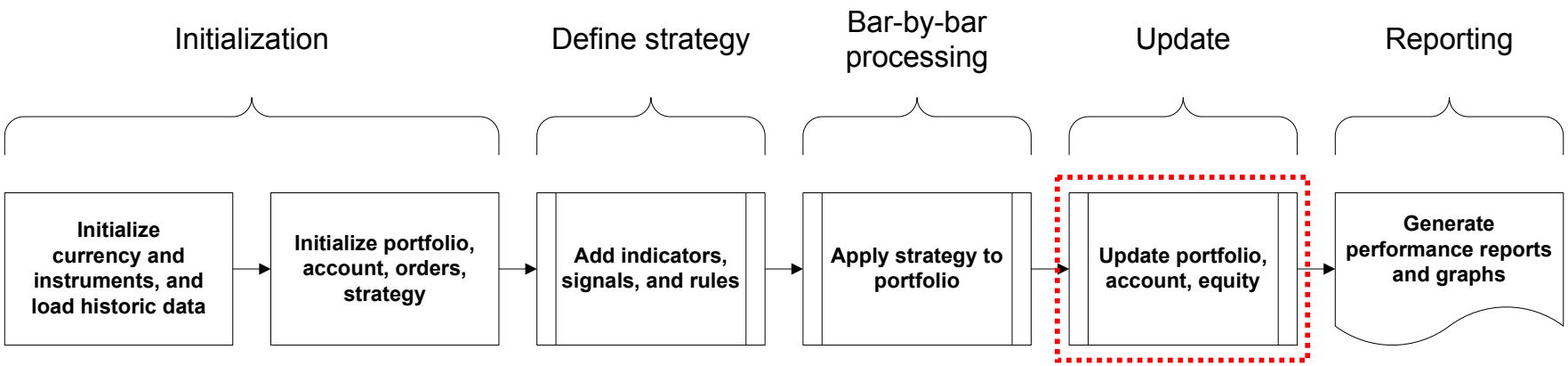
Applying strategy to a multi-asset portfolio




```
fastMA = 12
slowMA = 26
signalMA = 9
maType="EMA"
tradeSize <- initEq/10
```

```
out<-applyStrategy("macd" , portfolios="multi.macd",
  parameters=list(nFast=fastMA, nSlow=slowMA, nSig=signalMA,maType=maType),
  verbose=TRUE)
```

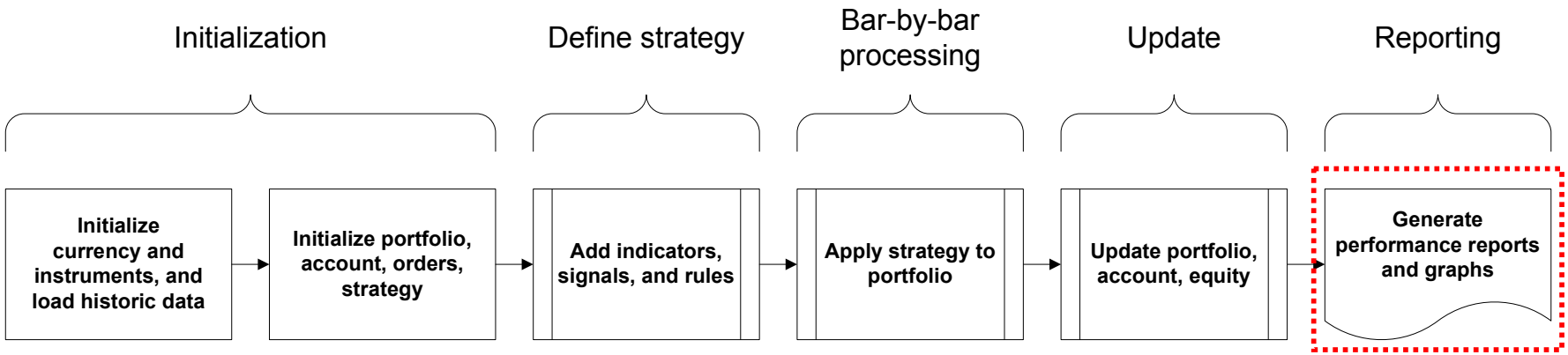
Update portfolio and account




```
updatePortf("multi.macd")
updateAcct("multi.macd")
updateEndEq("multi.macd")
```

```
checkBlotterUpdate("multi.macd","multi.macd")

## [1] TRUE
```

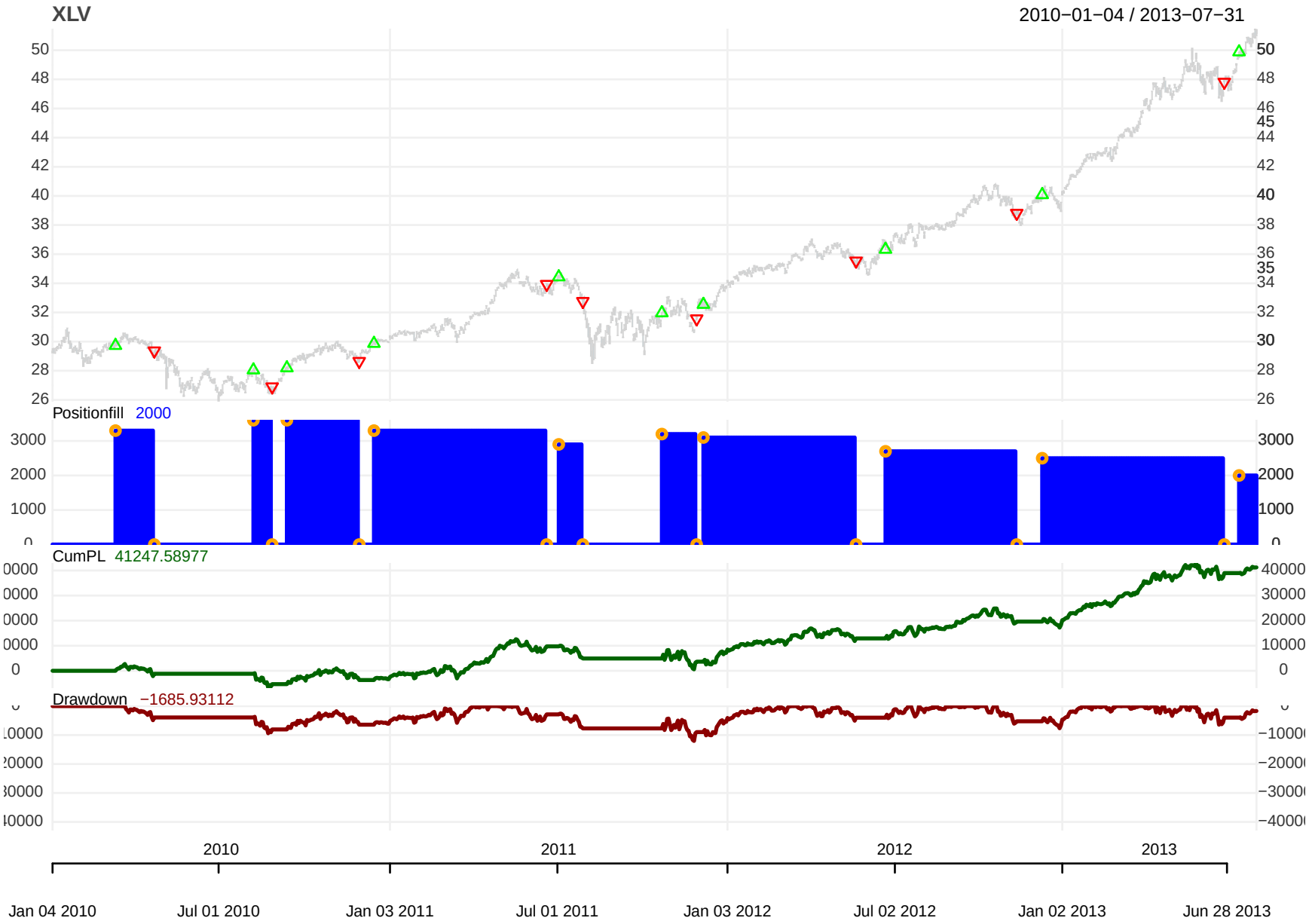
Chart trades and performance



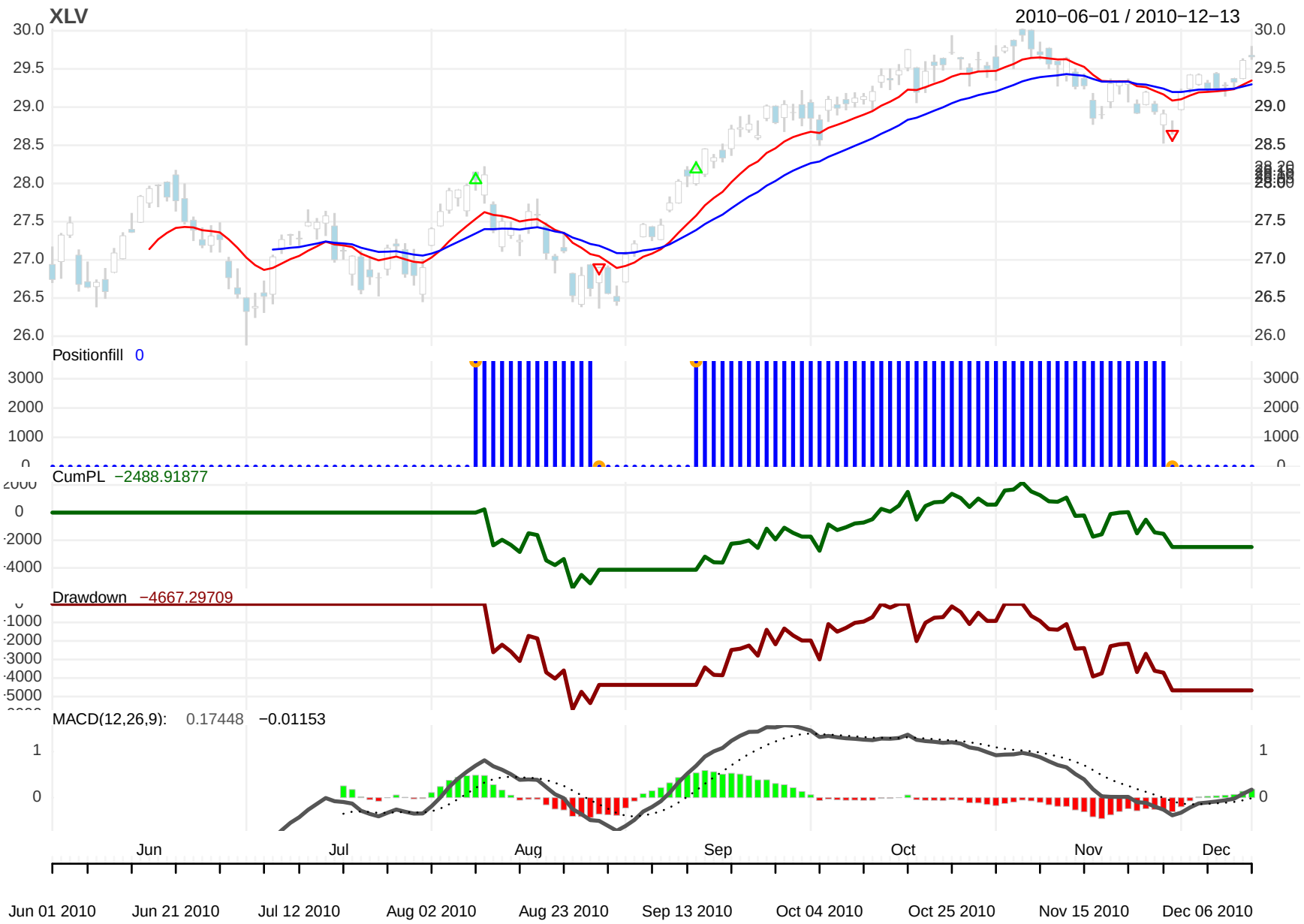

```
chart.Posn(Portfolio="multi.macd",Symbol="XLV",theme=myTheme)
```

```
chart.Posn(Portfolio="multi.macd",Symbol="XLV",
  Dates="201006::20101213",theme=myTheme)
add_MACD()
add_EMA(12,col='red')
add_EMA(26,col='blue')
```

MACD performance for XLV



MACD performance for XLV



Per-trade statistics

```
perTradeStats("multi.macd","XLF")
```

```
##        Start        End Init.Pos Max.Pos Num.Txns Max.Notional.Cost Net.Trading.PL         MAE
## 1 2010-03-10 2010-05-13     6900    6900        2        101286.788      2133.9136 -2262.77867
## 2 2010-08-02 2010-08-19     7100    7100        2        102040.770     -8187.6214 -8187.62143
## 3 2010-09-16 2010-12-02     7100    7100        2         99604.783      3004.3521 -3099.29774
## 4 2010-12-07 2011-03-17     6900    6900        2        100048.277      6382.1468     0.00000
## 5 2011-10-25 2011-11-25     7700    7700        2         97081.939     -9245.8989 -9469.59004
## 6 2011-12-14 2011-12-19     8200    8200        2         99733.308     -1758.6188 -1758.61885
## 7 2011-12-27 2012-05-10     7800    7800        2         99428.736     14806.5873 -1596.33292
## 8 2012-07-05 2012-11-15     6900    6900        2         98896.660      4426.7934 -2508.00304
## 9 2012-12-11 2013-07-31     6300    6300        1         99869.869     29217.1314  -435.02743
##           MFE Pct.Net.Trading.PL       Pct.MAE      Pct.MFE tick.Net.Trading.PL     tick.MAE    tick.MFE
## 1 10599.18678        0.021068035 -0.0223403142 0.1046453044           30.926284  -32.7938937 153.6114027
## 2     0.00000       -0.080238727 -0.0802387268 0.0000000000         -115.318612 -115.3186117   0.0000000
## 3  6056.17698        0.030162729 -0.0311159529 0.0608020699           42.314818  -43.6520808  85.2982673
## 4 13592.16304        0.063790672  0.0000000000 0.1358560436           92.494881    0.0000000 196.9878702
## 5  7680.06122       -0.095238095 -0.0975422427 0.0791090630         -120.076610 -122.9816889  99.7410547
## 6   478.97601       -0.017633215 -0.0176332149 0.0048025682          -21.446571  -21.4465713   5.8411709
## 7 22432.44329        0.148916580 -0.0160550459 0.2256132807          189.828042  -20.4658066 287.5954268
## 8 12254.32779        0.044761809 -0.0253598355 0.1239104308           64.156426  -36.3478701 177.5989535
## 9 31611.13136        0.292552015 -0.0043559428 0.3165232095          463.763990   -6.9051973 501.7639898
```

- Each order is approximately $100,000 in value

Outline

1. Run-time parameter passing (BBands strategy)
2. Order sizing
   - Fixed-dollar order sizing (MACD strategy)
   - Max position order sizing (BBands strategy)
   - Percent equity rebalancing (Faber strategy)

Bollinger Band reversal strategy with position limits

Buy rule:

- Buy long when the close crosses below the lower band

Sell rule:

- Sell short when the close crosses above the upper band

Exit rule:

- Exit any long or short position when either the high or low cross the mid-line

Pyramiding:

- Multiple orders are allowed in the same direction

Position limit:

- Limit number of shares for both long and short positions

Position limits and levels

- Position limits are set for the portfolio as a run-time parameter
- The function `osMaxPos` implements simple levels[†] based maximum positions
- The position sizing function `osMaxPos` must be passed via the `osFUN` argument of `ruleSignal`
- The maximum position and levels are accessed via the functions `addPosLimit` and `getPosLimit`

[†] The level is the number of pyramiding orders needed to reach the position limit

Define indicators and signals

```
strategy("bb.lim", store=TRUE)
```

```
add.indicator("bb.lim", name = "BBands",
  arguments = list(HLC = quote(HLC(mktdata)), maType='SMA'), label='BBands')
```

```
add.signal("bb.lim", name="sigCrossover",
  arguments=list(columns=c("Close","up"),relationship="gt"),
  label="Cl.gt.UpperBand")
```

```
add.signal("bb.lim", name="sigCrossover",
  arguments=list(columns=c("Close","dn"),relationship="lt"),
  label="Cl.lt.LowerBand")
```

```
add.signal("bb.lim", name="sigCrossover",
  arguments=list(columns=c("High","Low","mavg"),relationship="op"),
  label="Cross.Mid")
```

Add rules with an order sizing function specified

```
add.rule("bb.lim", name='ruleSignal',
  arguments=list(sigcol="Cl.gt.UpperBand",sigval=TRUE, orderqty=-1000,
  ordertype='market', orderside=NULL, osFUN='osMaxPos'),
  type='enter')
```

```
add.rule("bb.lim", name='ruleSignal',
  arguments=list(sigcol="Cl.lt.LowerBand",sigval=TRUE, orderqty= 1000,
  ordertype='market', orderside=NULL, osFUN='osMaxPos'),
  type='enter')
```

```
add.rule("bb.lim", name='ruleSignal',
  arguments=list(sigcol="Cross.Mid",sigval=TRUE, orderqty= 'all',
  ordertype='market', orderside=NULL),type='exit')
```

- The `ruleSignal` argument `osFUN` is set to `osMaxPos`
- The argument `orderqty` is set to 'all' for the exit rule

The addPosLimit function

The function addPosLimit adds position and level limits to a strategy

```
args(addPosLimit)

## function (portfolio, symbol, timestamp, maxpos, longlevels = 1,
##     minpos = -maxpos, shortlevels = longlevels)
## NULL
```

Main arguments:

portfolio	text name of the portfolio
symbol	instrument identifier
maxpos	maximum long position size
longlevels	number of levels

- Setting levels to 1 results in an order size of the maximum size

Initialize portfolio and add position limits

Position limits apply to individual assets in the portfolio

```
rm.strat("multi.bb.limit") # remove portfolio, account, orderbook if re-run
initPortf(name="multi.bb.limit", symbols, initDate=initDate)
initAcct(name="multi.bb.limit", portfolios="multi.bb.limit",
  initDate=initDate, initEq=initEq)
initOrders(portfolio="multi.bb.limit", initDate=initDate)
```

```
for(symbol in symbols)
{
  addPosLimit("multi.bb.limit", symbol, initDate, 200, 2 )
}
```

- Position limits are separated from the strategy and are a run-time constraint to the portfolio

Applying, update, and plot

```
SD = 2
N = 20
```

```
out <- applyStrategy("bb.lim",
  portfolios="multi.bb.limit",parameters=list(sd=SD,n=N))
```

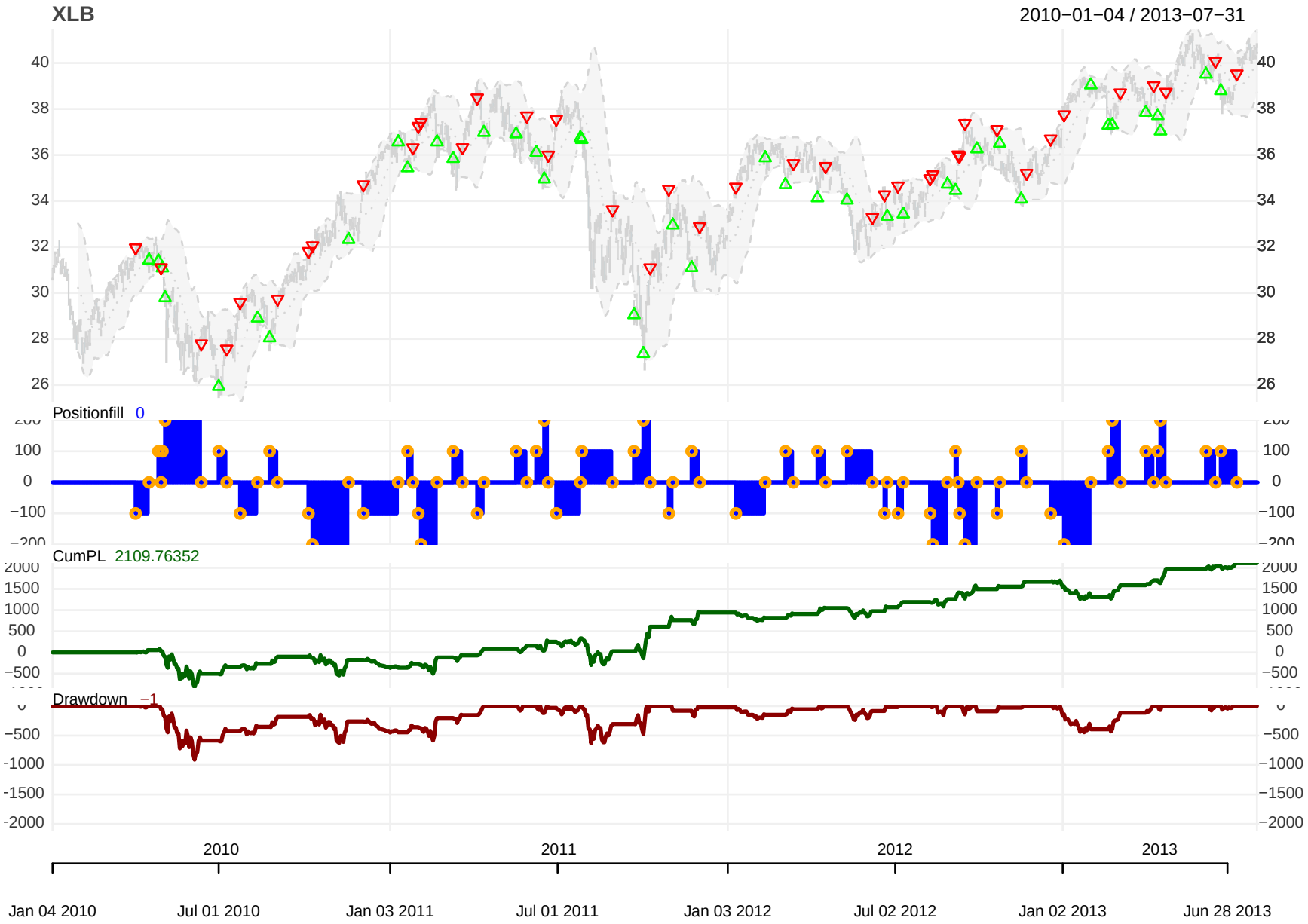
```
updatePortf("multi.bb.limit")
updateAcct("multi.bb.limit")
updateEndEq("multi.bb.limit")
```

```
checkBlotterUpdate("multi.bb.limit","multi.bb.limit")

## [1] TRUE
```

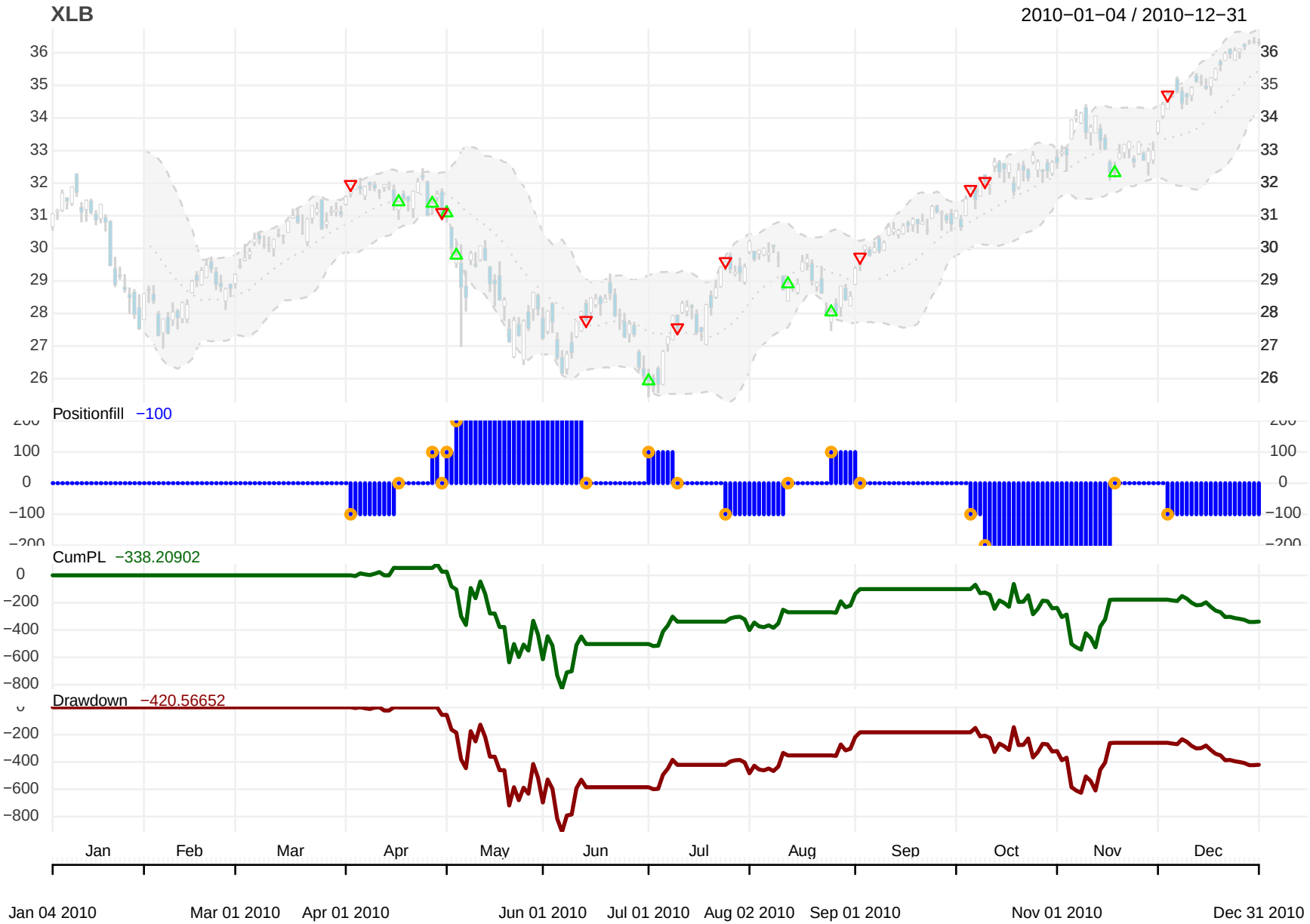
```
chart.Posn("multi.bb.limit","XLB",TA="add_BBands(n=20,sd=2)",theme=myTheme)
```

```
chart.Posn("multi.bb.limit","XLB",TA="add_BBands(n=20,sd=2)",
  Dates="2010",theme=myTheme)
```

BBands strategy for XLB with position limit



BBands strategy for XLB with position limit



Outline

1. Run-time parameter passing (BBands strategy)
2. Order sizing
   - Fixed-dollar order sizing (MACD strategy)
   - Max position order sizing (BBands strategy)
   - Percent equity rebalancing (Faber strategy)

Long-only Faber moving average crossover strategy

Buy rule:

- Buy long when close $>$ simple moving average

Exit rule:

- Sell when close $<$ simple moving average

Rebalance rule:

- Order size $= \frac{1}{N} \cdot$ (available equity) where N is the number of assets

Percent equity rebalancing

- Percent equity rebalancing involves dynamically setting asset position limits based on a fraction of the current portfolio equity
  - As the total equity grows (shrinks) over time, so does the position limit as a function of the percent of equity

- The strategy needs to include a rule with `type='rebalance'` and `function='rulePctEquity'`

- The position sizing function `osMaxPos` must be passed via the `osFUN` argument of `ruleSignal`

- The `applyStrategy.reblancing` function is used to apply the strategy to a portfolio

Define indicators and signals

```
strategy("faber",store=TRUE)
```

```
add.indicator(strategy = "faber", name = "SMA",
  arguments = list(x = quote(Cl(mktdata))), label="SMAn")
```

```
add.signal("faber",name="sigCrossover",
  arguments = list(columns=c("Close","SMAn"),relationship="gt"),
  label="Cl.gt.SMA")
```

```
add.signal("faber",name="sigCrossover",
  arguments = list(columns=c("Close","SMAn"),relationship="lt"),
  label="Cl.lt.SMA")
```

- Note that n is not included in the indicator arguments list

Add rules with an order sizing function specified

```
add.rule("faber", name='ruleSignal',
  arguments = list(sigcol="Cl.gt.SMA", sigval=TRUE, orderqty=100000,
  ordertype='market', orderside='long', osFUN='osMaxPos'),
  type='enter', path.dep=TRUE)
```

```
add.rule("faber", name='ruleSignal',
  arguments = list(sigcol="Cl.lt.SMA", sigval=TRUE, orderqty='all',
  ordertype='market', orderside='long', pricemethod='market'),
  type='exit', path.dep=TRUE)
```

- The `ruleSignal` argument `osFUN` is set to `osMaxPos`
- The argument `orderqty` for the entry rule is arbitrary
- The argument `orderqty` is set to 'all' for the exit rule

The rulePctEquity function

The function rulePctEquity implements a rule to base trade size on a percentage of available equity

```
args(rulePctEquity)

## function (trade.percent = 0.02, ..., longlevels = 1, shortlevels = 1,
##     digits = NULL, refprice = NULL, portfolio, symbol, timestamp)
## NULL
```

Main arguments:

rebalance_on	period on which to rebalance (e.g. 'months', 'quarters')
trade.percent	max percentage of equity to allow the strategy to trade in this symbol
longlevels	number of levels for long trades

Add percent equity rule

```
# add quaterly rebalancing
add.rule('faber', 'rulePctEquity',
  arguments=list(rebalance_on='months',
    trade.percent=1/length(symbols),
    refprice=quote(
      last(getPrice(mktdata)[paste('::',as.character(curIndex),sep='')][,1])
      ),
    digits=0
  ),
  type='rebalance',
  label='rebalance'
)
```

- Portfolio equity will be updated monthly and equally divided between all portfolio assets

Initialize portfolio and add position limits

```
rm.strat("multi.faber") # remove portfolio, account, orderbook if re-run
```

```
initPortf(name="multi.faber", symbols, initDate=initDate)
initAcct(name="multi.faber", portfolios="multi.faber",
  initDate=initDate, initEq=initEq)
initOrders(portfolio="multi.faber", initDate=initDate)
```

```
(posval <- initEq/length(symbols))

## [1] 111111.11

for(symbol in symbols){
    pos<-round((posval/first(getPrice(get(symbol)))),-2)
    addPosLimit('multi.faber',symbol,initDate, maxpos=pos,minpos=-pos)
}
```

Applying, update, and plot

```
out <- applyStrategy.rebalancing(strategy="faber", portfolios="multi.faber",
  parameters=list(n=200))
```

```
updatePortf("multi.faber")
updateAcct("multi.faber")
updateEndEq("multi.faber")
```

```
checkBlotterUpdate("multi.faber","multi.faber")

## [1] TRUE
```

```
chart.Posn("multi.faber","XLF",TA="add_SMA(n=200)",theme=myTheme)
```

Faber strategy for XLB with percent equity position sizing

Per-trade statistics

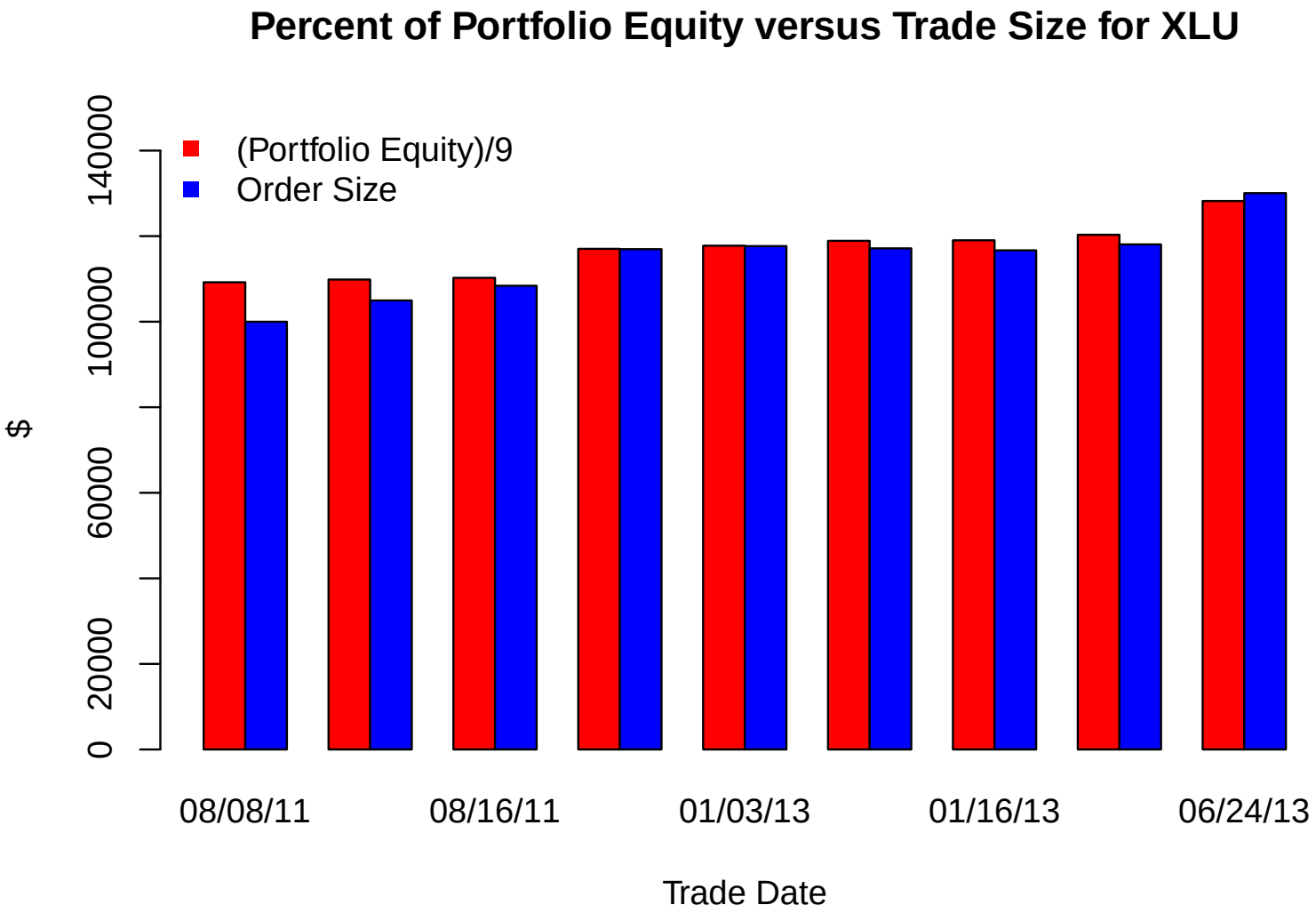
```
(pts <- perTradeStats("multi.faber","XLU"))
```

```
##        Start        End Init.Pos Max.Pos Num.Txns Max.Notional.Cost Net.Trading.PL         MAE          MFE
## 1 2011-08-08 2011-08-09     3588    3588        2         99989.764    3277.810139     0.00000  3277.810139
## 2 2011-08-12 2011-08-15     3588    3588        2        104956.143    3575.792879     0.00000  3575.792879
## 3 2011-08-16 2012-11-08     3588    3588        2        108399.499   13041.909223 -1423.69531 23311.978507
## 4 2012-12-18 2012-12-26     3348    3348        2        116956.896   -2425.381531 -2425.38153     0.000000
## 5 2013-01-03 2013-01-09     3372    3372        2        117668.845   -1158.157921 -1158.15792   595.624074
## 6 2013-01-11 2013-01-15     3372    3372        2        117139.401      99.270679  -231.63158    99.270679
## 7 2013-01-16 2013-01-17     3372    3372        2        116676.138     463.263168     0.00000   463.263168
## 8 2013-01-18 2013-06-21     3372    3372        2        118065.927    5146.952504     0.00000 20217.543887
## 9 2013-06-24 2013-07-31     3562    3562        1        130048.620    9795.500000     0.00000 10828.480000
##   Pct.Net.Trading.PL       Pct.MAE       Pct.MFE tick.Net.Trading.PL    tick.MAE    tick.MFE
## 1      0.03278145695  0.0000000000 0.03278145695          91.3547976   0.0000000  91.3547976
## 2      0.03406940063  0.0000000000 0.03406940063          99.6597792   0.0000000  99.6597792
## 3      0.12031337184 -0.0131337813 0.21505614635         363.4868791 -39.6793565 649.7206942
## 4     -0.02073739654 -0.0207373965 0.00000000000         -72.4426981 -72.4426981   0.0000000
## 5     -0.00984251969 -0.0098425197 0.00506186727         -34.3463203 -34.3463203  17.6638219
## 6      0.00084745763 -0.0019774011 0.00084745763           2.9439703  -6.8692641   2.9439703
## 7      0.00397050482  0.0000000000 0.00397050482          13.7385281   0.0000000  13.7385281
## 8      0.04359388532  0.0000000000 0.17123944491         152.6379746   0.0000000 599.5712896
## 9      0.07532182964  0.0000000000 0.08326485894         275.0000000   0.0000000 304.0000000
```

```
mnc <- pts$Max.Notional.Cost
pe <- sapply(pts$Start,getEndEq,Account="multi.faber")/9
barplot(rbind(pe,mnc),beside=T,col=c(2,4),names.arg=format(pts$Start,"%m/%d/%y"),
  ylim=c(0,1.5e5),ylab="$",xlab="Trade Date")
legend(x="topleft",legend=c("(Portfolio Equity)/9","Order Size"),
  pch=15,col=c(2,4),bty="n")
title("Percent of Portfolio Equity versus Trade Size for XLU")
```

- Each trade has a value of approximately 1/9 of the portfolio equity

Verify percent of equity rebalancing



COMPUTATIONAL FINANCE & RISK MANAGEMENT

UNIVERSITY *of* WASHINGTON

Department of Applied Mathematics

`http://depts.washington.edu/compfin`