COMPUTATIONAL FINANCE & RISK MANAGEMENT

UNIVERSITY *of* WASHINGTON

Department of Applied Mathematics

Introduction to Trading Systems

Guy Yollin

Applied Mathematics
University of Washington

Outline

Outline

Lecture references

- TradeAnalytics project page on R-forge:
  `http://r-forge.r-project.org/projects/blotter/`
  - documents and demos for:
    - blotter package
    - quantstrat package
      (specifically the demo scripts `faber.R`)[†]

- 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`

Quantitative analysis package hierarchy

Application Area	R Package
Performance metrics and graphs	**PerformanceAnalytics** - Tools for performance and risk analysis
Portfolio optimization and quantitative trading strategies	**PortfolioAnalytics** - Portfolio analysis and optimization
	quantstrat – Rules-based trading system development
	blotter – Trading system accounting infrastructure
Data access and financial charting	**quantmod** - Quantitative financial modeling framework
	TTR - Technical trading rules
Time series objects	**xts** - Extensible time series
	zoo - Ordered observation

About blotter and quantstrat

- Provides support for multi-asset class and multi-currency portfolios for backtesting and other financial research. **Still in heavy development**.
- The software is in an beta stage
  - some things are not completely implemented (or documented)
  - some things invariably have errors
  - some implementations will change in the future
- Software has been in development for a number of years
  - blotter: Dec-2008
  - quantstrat: Feb-2010
- Software is used everyday by working professions in asset management

The quantstrat package

Description

quantstrat provides a generic infrastructure to model and backtest signal-based quantitative strategies. It is a high-level abstraction layer (built on xts, FinancialInstrument, blotter, etc.) that allows you to build and test strategies in very few lines of code.

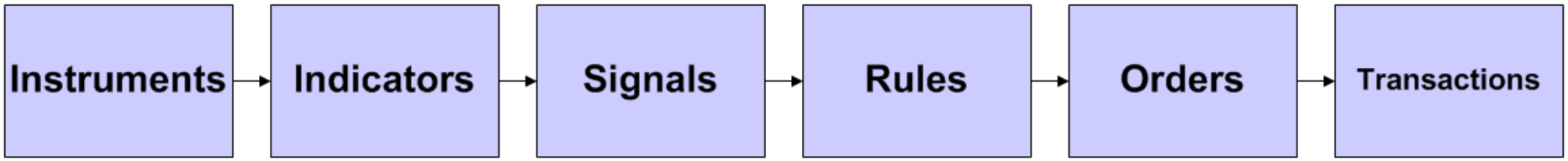
Key features

- Supports strategies which include indicators, signals, and rules
- Allows strategies to be applied to multi-asset portfolios
- Supports market, limit, stoplimit, and stoptrailing order types
- Supports order sizing and parameter optimization

Authors

- Peter Carl
- Brian Peterson
- Jeffrey Ryan

Quantstrat object model

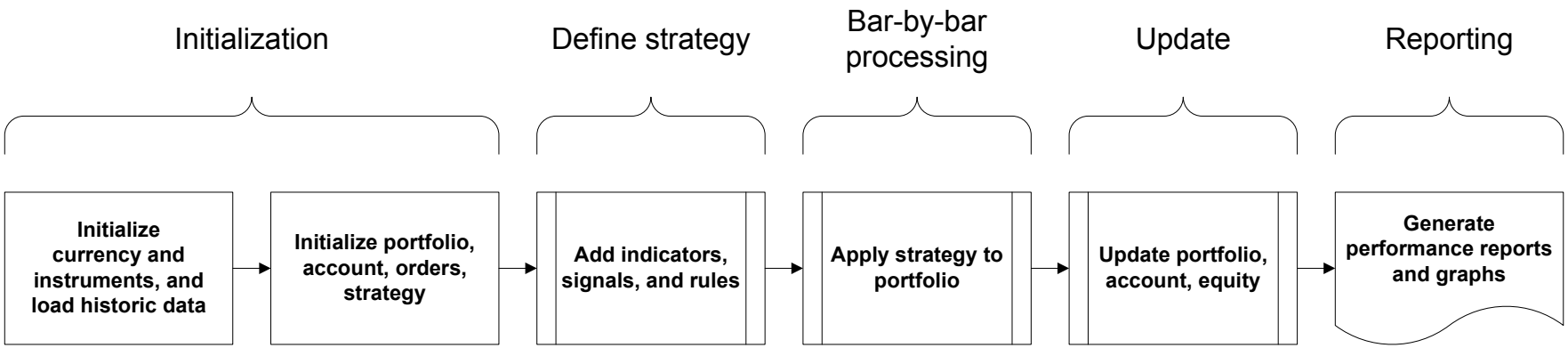


Generic Signal-Based Strategy Modeling:

- Instruments contain market data
- Indicators are quantitative values derived from market data
- Interaction between indicators and market data are used to generate signals (e.g. crossovers, thresholds)
- Rules use market data, indicators, signals, and current account/portfolio characteristics to generate orders
- Interaction between orders and market data generates transactions

Outline

Basic strategy backtesting workflow for quantstrat



Key quantstrat functions

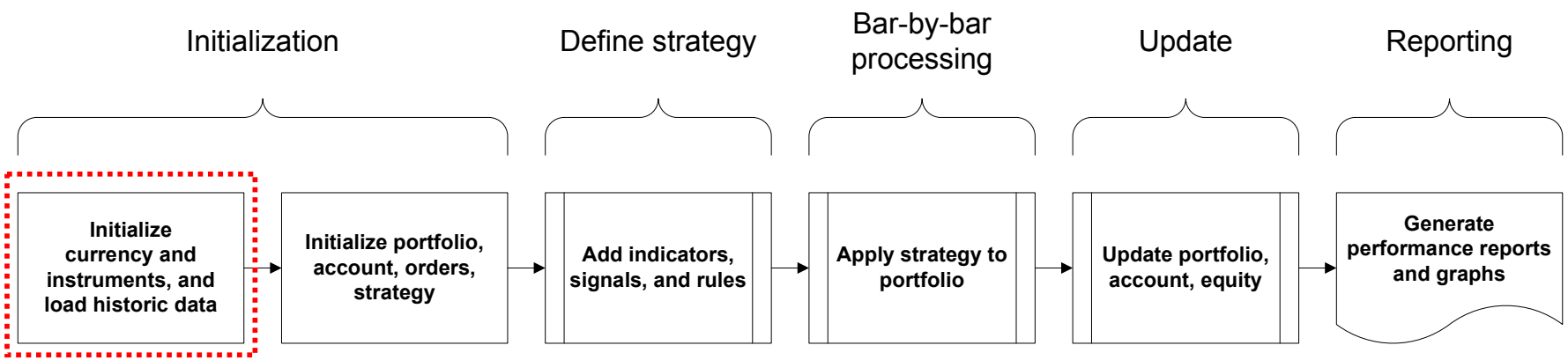
Initialization	
initOrders	initialize order container
strategy	constructor for strategy object

Strategy definition	
add.indicator	add an indicator to a strategy
add.signal	add a signal to a strategy
add.rule	add a rule to a strategy
add.distribution	add a distribution to a paramset in a strategy
add.constraint	add a constraint on 2 distributions within a paramset

Processing	
applyStrategy	apply the strategy to arbitrary market data
addPosLimit	add position and level limits at timestamp
apply.paramset	apply a paramset to the strategy
applyStrategy.rebalancing	apply the strategy to data with periodic rebalancing

The functions in quantstrat are used in conjunction with the functions in `blotter`

Quantstrat backtesting: step 1



Loading the quantstrat package

```
before <- search()
library(quantstrat)
after <- search()
after[!(after %in% before)]

##  [1] "package:quantstrat"           "package:foreach"
##  [3] "package:blotter"              "package:PerformanceAnalytics"
##  [5] "package:FinancialInstrument"  "package:quantmod"
##  [7] "package:methods"              "package:TTR"
##  [9] "package:xts"                  "package:zoo"
```

Loading `quantstrat` causes these other libraries to be loaded automatically:

- blotter
- foreach
- quantmod
- PerformanceAnalytics
- FinancialInstrument
- TTR
- xts
- zoo

Initialize a currency and a stock instrument

```
currency("USD")

## [1] "USD"

stock("SPY",currency="USD",multiplier=1)

## [1] "SPY"

ls(envir=FinancialInstrument:::.instrument)

## [1] "SPY" "USD"

ls(all=T)

## [1] ".blotter"  ".strategy" "after"     "before"    "filename"
```

- `botter` functions used for instrument initialization
- Currency and trading instrument objects stored in the `.instrument` environment
- `quantstrat` creates a private storage area called `.strategy`

Fetch historic data

```
# system settings
initDate <- '1997-12-31'
startDate <- '1998-01-01'
endDate <-  '2014-06-30'
initEq <- 1e6
```

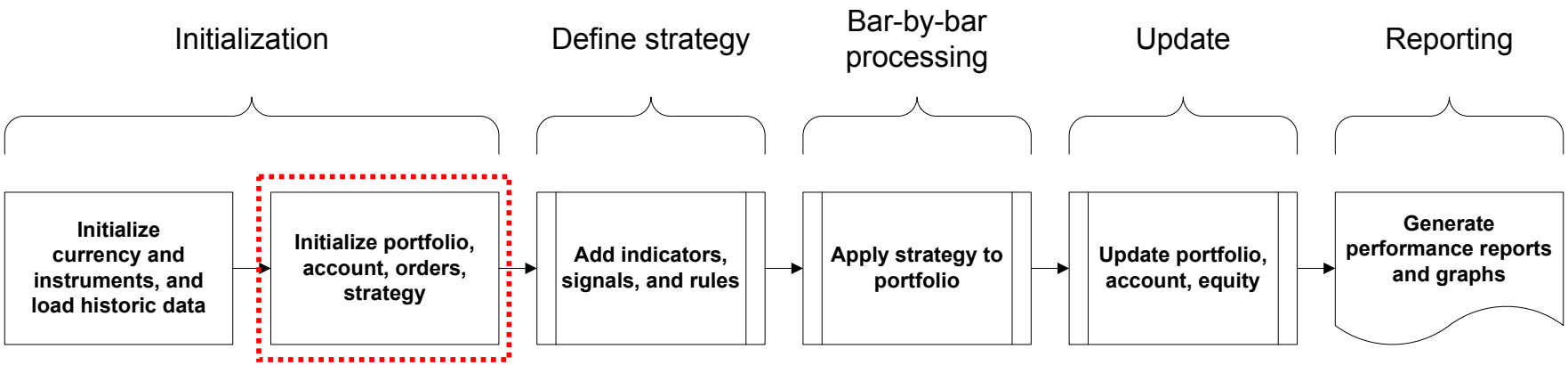
```
Sys.setenv(TZ="UTC")
```

```
getSymbols('SPY', from=startDate, to=endDate, index.class="POSIXct", adjust=T)
```

```
SPY=to.monthly(SPY, indexAt='endof', drop.time=FALSE)
SPY$SMA10m <- SMA(Cl(SPY), 10)
```

- Must set timezone
- Must use a POSIX time-date index class
- In `to.monthly`, you must use `'endof'` and you must set `drop.time=FALSE`

Quantstrat backtesting: step 2



Initialize portfolio and account

```
# inz portfolio, account
qs.strategy <- "qsFaber"
```

```
rm.strat(qs.strategy) # remove strategy etc. if this is a re-run
```

```
initPortf(qs.strategy,'SPY', initDate=initDate)
```

```
initAcct(qs.strategy,portfolios=qs.strategy, initDate=initDate, initEq=initEq)
```

- The function `rm.strat` removes any strategy, portfolio, account, or order book object with the given name
  - Facilitates re-running the code for debugging
- `botter` functions used for portfolio and account initialization

Initialize orders and strategy

The function `initOrders` sets up an order container for the portfolio.

The function `strategy` creates a strategy object.

```
# initialize orders container
args(initOrders)

## function (portfolio = NULL, symbols = NULL, initDate = "1999-12-31",
##     ...)
## NULL

initOrders(portfolio=qs.strategy,initDate=initDate)

# instantiate a new strategy object
args(strategy)

## function (name, ..., assets = NULL, constraints = NULL, store = FALSE)
## NULL

strategy(qs.strategy,store=TRUE)
```

- quantstrat specific initialization

Portfolio, account, orderbook, and strategy objects

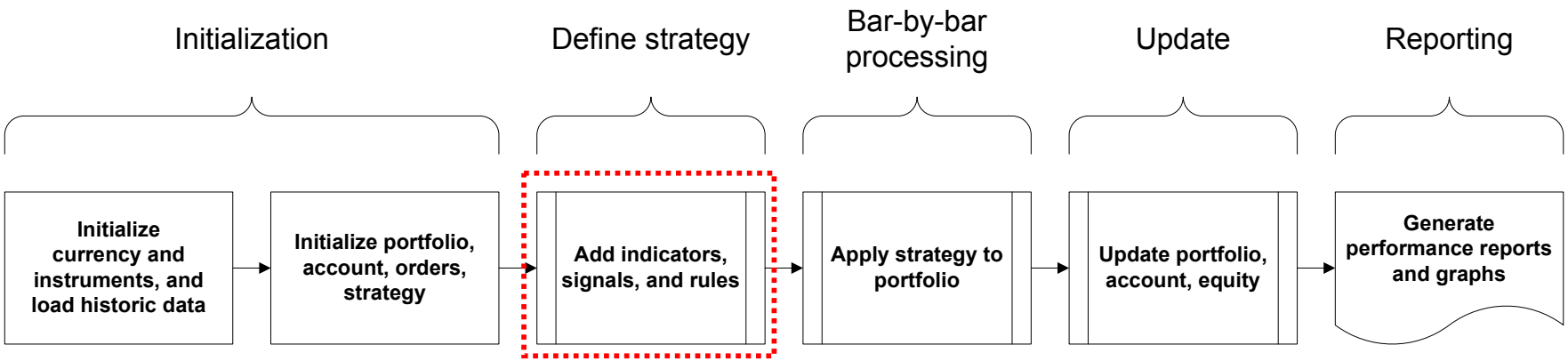
```
ls(all=T)

##  [1] ".blotter"    ".strategy"   "after"       "before"      "endDate"
##  [6] "filename"    "initDate"    "initEq"      "qs.strategy" "SPY"
## [11] "startDate"

ls(.blotter)

## [1] "account.qsFaber"   "portfolio.qsFaber"

ls(.strategy)

## [1] "order_book.qsFaber" "qsFaber"
```

- `.blotter` holds the `portfolio` and `account` object
- `.strategy` holds the `orderbook` and `strategy` object

The quantstrat strategy object

```
args(getStrategy)

## function (x, envir = .strategy)
## NULL

strat <-getStrategy(qs.strategy)
class(strat)

## [1] "strategy"

summary(strat)

##             Length Class  Mode
## name        1      -none- character
## assets      0      -none- NULL
## indicators  0      -none- list
## signals     0      -none- list
## rules       1      -none- list
## constraints 0      -none- NULL
## init        0      -none- list
## wrapup      0      -none- list
## call        3      -none- call
```

Quantstrat backtesting: step 3



Faber tactical asset allocation system

Buy-Sell rules:

- buy when monthly price $>$ 10-month SMA
- sell and move to cash when monthly price $<$ 10-month SMA

Notes:

- all entry and exit prices are on the day of the signal at the close
- all data series are total return series including dividends, updated monthly
- commissions and slippage are excluded

A Quantitative Approach to Tactical Asset Allocation by Mebane T. Faber, The Journal of Wealth Management, Spring 2007

The add.indicator function

- Indicators are typically standard technical or statistical analysis outputs, such as moving averages, bands, or pricing models
- Indicators are applied before signals and rules, and the output of indicators may be used as inputs to construct signals or fire rules

```
args(add.indicator)

## function (strategy, name, arguments, parameters = NULL, label = NULL,
##     ..., enabled = TRUE, indexnum = NULL, store = FALSE)
## NULL
```

Main arguments:

strategy	strategy object
name	name of the indicator (must be an R function)
arguments	arguments to be passed to the indicator function
label	name to reference the indicator

Adding an indicator to a strategy

- Add a 10-month simple moving average

```
add.indicator(strategy = qs.strategy, name = "SMA",
  arguments = list(x = quote(Cl(mktdata)), n=10), label="SMA10")
```

```
summary(getStrategy(qs.strategy))

##             Length Class  Mode
## name        1      -none- character
## assets      0      -none- NULL
## indicators  1      -none- list
## signals     0      -none- list
## rules       1      -none- list
## constraints 0      -none- NULL
## init        0      -none- list
## wrapup      0      -none- list
## call        3      -none- call
```

The add.signals function

quantstrat supports the following signal types:

sigCrossover	crossover signal ("gt", "lt", "eq", "gte", "lte")
sigComparison	comparison signal ("gt", "lt", "eq", "gte", "lte")
sigThreshold	threshold signal ("gt", "lt", "eq", "gte", "lte")
sigPeak	peak/valley signals ("peak", "bottom")
sigFormula	signal calculated from a formula

```
args(add.signal)

## function (strategy, name, arguments, parameters = NULL, label = NULL,
##     ..., enabled = TRUE, indexnum = NULL, store = FALSE)
## NULL
```

Main arguments:

strategy	strategy object
name	name of the signal, must correspond to an R function
arguments	arguments to be passed to the indicator function

Adding signals to a strategy

- Add signal for crossing above SMA
- Add signal for crossing below SMA

```
add.signal(qs.strategy,name="sigCrossover",
  arguments = list(columns=c("Close","SMA10"),relationship="gt"),
  label="Cl.gt.SMA")
```

```
add.signal(qs.strategy,name="sigCrossover",
  arguments = list(columns=c("Close","SMA10"),relationship="lt"),
  label="Cl.lt.SMA")
```

```
summary(getStrategy(qs.strategy))
```

```
##             Length Class  Mode
## name        1      -none- character
## assets      0      -none- NULL
## indicators  1      -none- list
## signals     2      -none- list
## rules       1      -none- list
## constraints 0      -none- NULL
## init        0      -none- list
## wrapup      0      -none- list
## call        3      -none- call
```

The add.rules function

The function add.rule adds a rule to a strategy

```
args(add.rule)

## function (strategy, name, arguments, parameters = NULL, label = NULL,
##     type = c(NULL, "risk", "order", "rebalance", "exit", "enter",
##         "chain"), parent = NULL, ..., enabled = TRUE, indexnum = NULL,
##     path.dep = TRUE, timespan = NULL, store = FALSE, storefun = TRUE)
## NULL
```

Main arguments:

strategy	strategy object
name	name of the rule (typically ruleSignal)
arguments	arguments to be passed to the rule function
type	type of rule ("risk","order","rebalance","exit","enter")

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)

Add rules to a strategy

- Add rule to enter when `Cl.gt.SMA` is true
- Add rule to exit when `Cl.lt.SMA` is true

```
#   go long when close > MA
add.rule(qs.strategy, name='ruleSignal',
  arguments = list(sigcol="Cl.gt.SMA", sigval=TRUE, orderqty=900,
  ordertype='market', orderside='long'),
  type='enter')
```

```
#   exit when close < MA
add.rule(qs.strategy, name='ruleSignal',
  arguments = list(sigcol="Cl.lt.SMA", sigval=TRUE, orderqty='all',
  ordertype='market', orderside='long'),
  type='exit')
```

The completed strategy object

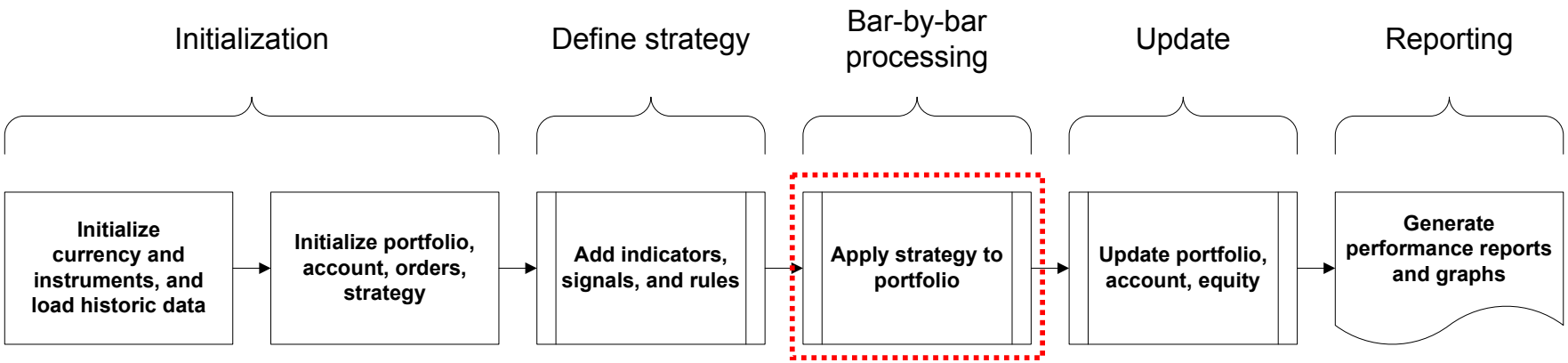
The strategy object now contains a complete set of quantitative trading rules ready to be applied to a portfolio of assets.

```
summary(getStrategy(qs.strategy))

##             Length Class  Mode
## name        1      -none- character
## assets      0      -none- NULL
## indicators  1      -none- list
## signals     2      -none- list
## rules       3      -none- list
## constraints 0      -none- NULL
## init        0      -none- list
## wrapup      0      -none- list
## call        3      -none- call
```

The strategy object contains:

- 1 user defined indicator
- 2 user defined signals
- 2 user defined trading rules

Quantstrat backtesting: step 4



The applyStrategy function

The applyStrategy function applies the strategy to arbitrary market data

```
args(applyStrategy)

## function (strategy, portfolios, mktdata = NULL, parameters = NULL,
##     ..., debug = FALSE, symbols = NULL, initStrat = FALSE, updateStrat = FALSE,
##     initBySymbol = FALSE, gc = FALSE, delorders = FALSE)
## NULL
```

Main arguments:

strategy	an object of type 'strategy'
portfolios	a list of portfolios to apply the strategy to
parameters	named list of parameters to be applied during evaluation of the strategy

Apply the strategy

Calling `applyStrategy` generates transactions in the specified portfolio.

```
applyStrategy(strategy=qs.strategy , portfolios=qs.strategy)
```

```
getTxns(Portfolio=qs.strategy, Symbol="SPY")
```

```
##            Txn.Qty  Txn.Price Txn.Fees   Txn.Value Txn.Avg.Cost Net.Txn.Realized.PL
## 1997-12-31       0   0.000000        0       0.000     0.000000               0.0000
## 1999-10-29     900 104.732831        0   94259.548   104.732831               0.0000
## 2000-09-29    -900 110.880542        0  -99792.488   110.880542            5532.9397
## 2002-03-28     900  90.076820        0   81069.138    90.076820               0.0000
## 2002-04-30    -900  84.838332        0  -76354.499    84.838332           -4714.6390
## 2003-04-30     900  73.524300        0   66171.870    73.524300               0.0000
## 2004-08-31    -900  90.615126        0  -81553.613    90.615126           15381.7436
## 2004-09-30     900  91.524627        0   82372.165    91.524627               0.0000
## 2007-12-31    -900 127.365627        0 -114629.065   127.365627           32256.9001
## 2009-06-30     900  83.080687        0   74772.618    83.080687               0.0000
## 2010-06-30    -900  95.050379        0  -85545.341    95.050379           10772.7225
## 2010-07-30     900 101.542388        0   91388.149   101.542388               0.0000
## 2010-08-31    -900  96.974960        0  -87277.464    96.974960           -4110.6849
## 2010-09-30     900 105.659523        0   95093.571   105.659523               0.0000
## 2011-08-31    -900 114.804639        0 -103324.175   114.804639            8230.6044
## 2012-01-31     900 124.777069        0  112299.362   124.777069               0.0000
```

The mktdata object

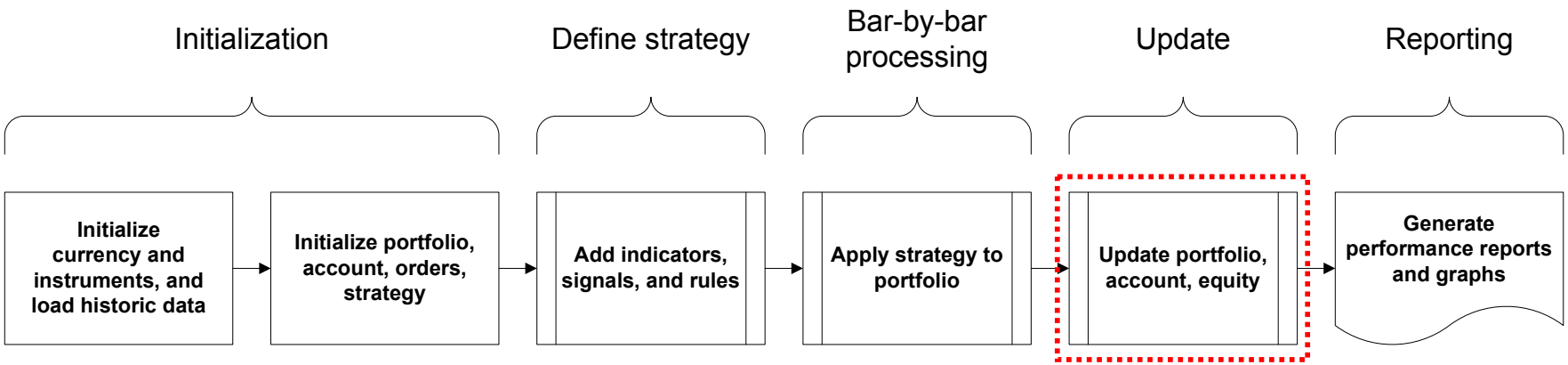
mktdata is a special variable constructed during the execution of applyStrategy. It is a time series object which contains the historic price data as well as the calculated indicators, signals, and rules:

```
mktdata["2002"]
```

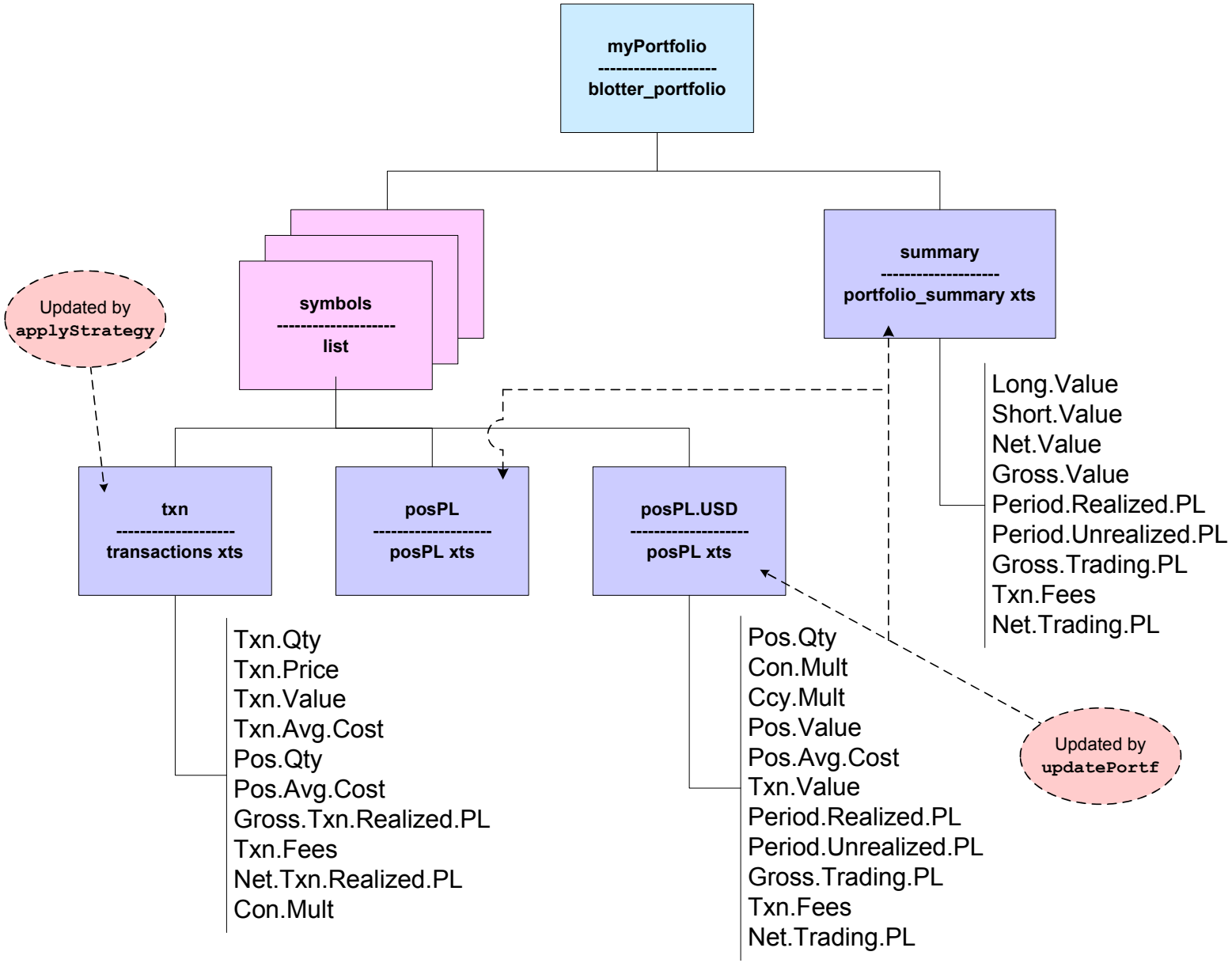
```
##            SPY.Open SPY.High SPY.Low SPY.Close SPY.Volume SPY.Adjusted SPY.Close.SMA.10.SMA10 Cl.gt.SMA Cl.lt.SMA
## 2002-01-31  90.2823  92.5411 85.0195   88.7685  445459915        88.77                90.6892        NA        NA
## 2002-02-28  88.6980  88.8627 84.5646   87.1764  541228569        87.18                89.5718        NA        NA
## 2002-03-28  87.6234  92.6095 87.4587   90.0768  490924657        90.08                88.7995         1        NA
## 2002-04-30  89.8487  90.5330 83.8709   84.8383  514215242        84.84                87.7365        NA         1
## 2002-05-31  84.9249  87.5048 82.5101   84.3349  575450897        84.33                86.7204        NA        NA
## 2002-06-28  84.2327  84.6338 75.1347   78.1104  679144777        78.11                85.6425        NA        NA
## 2002-07-31  78.2840  78.7734 61.3138   71.9537 1424338940        71.95                84.6746        NA        NA
## 2002-08-30  71.7327  76.6817 65.9471   72.4431 1186084060        72.44                83.6494        NA        NA
## 2002-09-30  71.6143  73.6666 64.1416   64.8473 1282107423        64.85                81.2197        NA        NA
## 2002-10-31  65.3547  72.3794 61.1050   70.1832 1695461119        70.18                79.2733        NA        NA
## 2002-11-29  70.0484  75.2812 69.3348   74.5121 1032026655        74.51                77.8476        NA        NA
## 2002-12-31  75.6935  76.1533 69.4046   70.2970  917527170        70.30                76.1597        NA        NA
```

- Inspecting mktdata can be very helpful in understanding strategy processing and debugging

Quantstrat backtesting: step 5



How the blotter_portfolio object gets updated



Update portfolio, account, and equity

```
updatePortf(qs.strategy)
```

```
updateAcct(qs.strategy)
```

```
updateEndEq(qs.strategy)
```

- Functions must be called in order:
  - updatePortf
  - updateAcct
  - updateEndEq

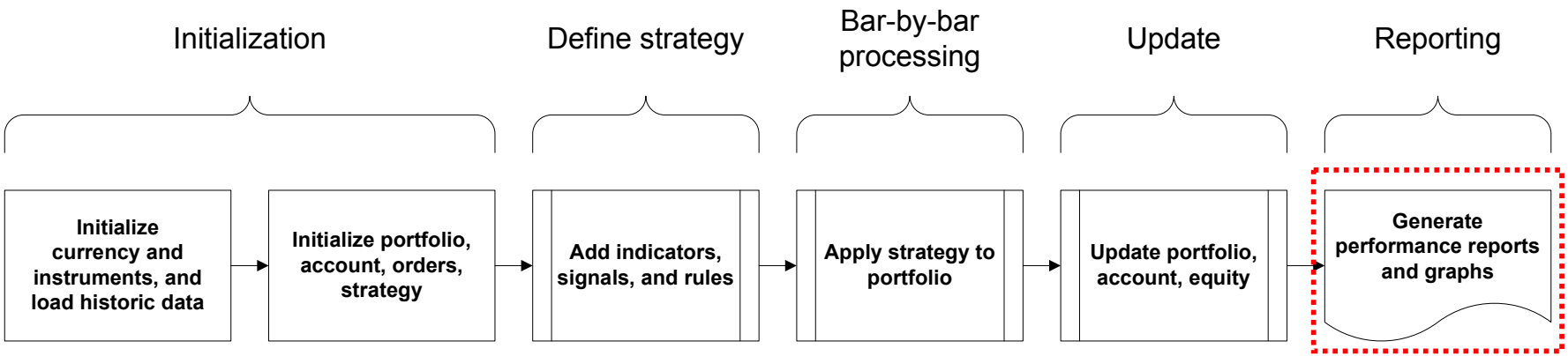
Data integrity check

```
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(qs.strategy,qs.strategy)
```

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

Outline

Quantstrat backtesting: step 6

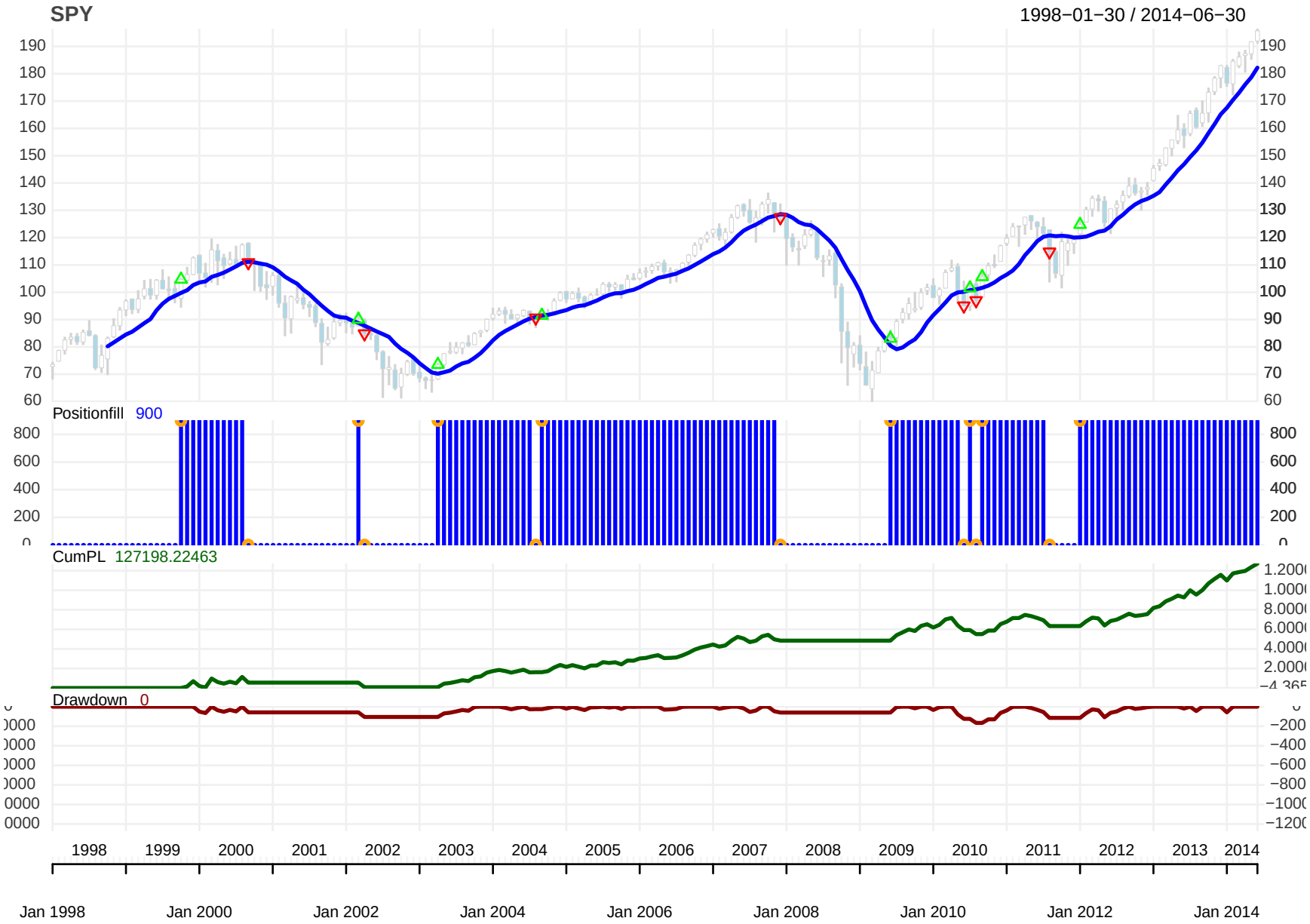


Plot performance

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

```
# plot performance
chart.Posn(qs.strategy, Symbol = 'SPY', Dates = '1998::',theme=myTheme,
  TA='add_SMA(n=10,col=4, on=1, lwd=2)')
```

Faber/quanstrat system performance



Trade statistics

```
tstats <- t(tradeStats(qs.strategy))
```

Portfolio	qsFaber	**Avg.Win.Trade**	14434.982
Symbol	SPY	**Med.Win.Trade**	10772.723
Num.Txns	15	**Avg.Losing.Trade**	−4412.6619
Num.Trades	7	**Med.Losing.Trade**	−4412.6619
Net.Trading.PL	127198.22	**Avg.Daily.PL**	10558.264
Avg.Trade.PL	9049.9409	**Med.Daily.PL**	8152.8311
Med.Trade.PL	8230.6044	**Std.Dev.Daily.PL**	12602.464
Largest.Winner	32256.9	**Ann.Sharpe**	13.299562
Largest.Loser	−4714.639	**Max.Drawdown**	−16564.913
Gross.Profits	72174.91	**Profit.To.Max.Draw**	7.6787739
Gross.Losses	−8825.3239	**Avg.WinLoss.Ratio**	3.271264
Std.Dev.Trade.PL	12631.502	**Med.WinLoss.Ratio**	2.4413206
Percent.Positive	71.428571	**Max.Equity**	127198.22
Percent.Negative	28.571429	**Min.Equity**	−4.3655746e−10
Profit.Factor	8.17816	**End.Equity**	127198.22

The order_book object

The function getOrderBook can be used to retrieve the order_book object

```
ob <- getOrderBook(qs.strategy)
class(ob)

## [1] "order_book"

names(ob)

## [1] "qsFaber"

names(ob$qsFaber)

## [1] "SPY"

names(ob$qsFaber$SPY)

##  [1] "Order.Qty"         "Order.Price"      "Order.Type"       "Order.Side"
##  [5] "Order.Threshold"   "Order.Status"     "Order.StatusTime" "Prefer"
##  [9] "Order.Set"         "Txn.Fees"         "Rule"             "Time.In.Force"
```

The orderbook object

```
ob$qsFaber$SPY[,1:5]
```

```
##                     Order.Qty Order.Price        Order.Type Order.Side Order.Threshold
## 1999-10-29 00:00:00 "900"     "104.732831442796" "market"   "long"     NA
## 2000-09-29 00:00:00 "all"     "110.88054219626"  "market"   "long"     NA
## 2002-03-28 00:00:00 "900"     "90.0768199177471" "market"   "long"     NA
## 2002-04-30 00:00:00 "all"     "84.838332136991"  "market"   "long"     NA
## 2003-04-30 00:00:00 "900"     "73.5242995992802" "market"   "long"     NA
## 2004-08-31 00:00:00 "all"     "90.6151258710772" "market"   "long"     NA
## 2004-09-30 00:00:00 "900"     "91.5246273092301" "market"   "long"     NA
## 2007-12-31 00:00:00 "all"     "127.365627370135" "market"   "long"     NA
## 2009-06-30 00:00:00 "900"     "83.0806870602298" "market"   "long"     NA
## 2010-06-30 00:00:00 "all"     "95.0503787569706" "market"   "long"     NA
## 2010-07-30 00:00:00 "900"     "101.542387769145" "market"   "long"     NA
## 2010-08-31 00:00:00 "all"     "96.9749601520692" "market"   "long"     NA
## 2010-09-30 00:00:00 "900"     "105.659522947531" "market"   "long"     NA
## 2011-08-31 00:00:00 "all"     "114.804638992219" "market"   "long"     NA
## 2012-01-31 00:00:00 "900"     "124.777068734084" "market"   "long"     NA
```

The orderbook object

```
ob$qsFaber$SPY[,6:11]
```

```
##                     Order.Status Order.StatusTime      Prefer Order.Set Txn.Fees Rule
## 1999-10-29 00:00:00 "closed"     "1999-11-30 00:00:00" ""     NA        "0"      "ruleSignal.rule"
## 2000-09-29 00:00:00 "closed"     "2000-10-31 00:00:00" ""     NA        "0"      "ruleSignal.rule"
## 2002-03-28 00:00:00 "closed"     "2002-04-30 00:00:00" ""     NA        "0"      "ruleSignal.rule"
## 2002-04-30 00:00:00 "closed"     "2002-05-31 00:00:00" ""     NA        "0"      "ruleSignal.rule"
## 2003-04-30 00:00:00 "closed"     "2003-05-30 00:00:00" ""     NA        "0"      "ruleSignal.rule"
## 2004-08-31 00:00:00 "closed"     "2004-09-30 00:00:00" ""     NA        "0"      "ruleSignal.rule"
## 2004-09-30 00:00:00 "closed"     "2004-10-29 00:00:00" ""     NA        "0"      "ruleSignal.rule"
## 2007-12-31 00:00:00 "closed"     "2008-01-31 00:00:00" ""     NA        "0"      "ruleSignal.rule"
## 2009-06-30 00:00:00 "closed"     "2009-07-31 00:00:00" ""     NA        "0"      "ruleSignal.rule"
## 2010-06-30 00:00:00 "closed"     "2010-07-30 00:00:00" ""     NA        "0"      "ruleSignal.rule"
## 2010-07-30 00:00:00 "closed"     "2010-08-31 00:00:00" ""     NA        "0"      "ruleSignal.rule"
## 2010-08-31 00:00:00 "closed"     "2010-09-30 00:00:00" ""     NA        "0"      "ruleSignal.rule"
## 2010-09-30 00:00:00 "closed"     "2010-10-29 00:00:00" ""     NA        "0"      "ruleSignal.rule"
## 2011-08-31 00:00:00 "closed"     "2011-09-30 00:00:00" ""     NA        "0"      "ruleSignal.rule"
## 2012-01-31 00:00:00 "closed"     "2012-02-29 00:00:00" ""     NA        "0"      "ruleSignal.rule"
```

Per-trade statistics

```
perTradeStats(qs.strategy)
```

```
##        Start        End Init.Pos Max.Pos Num.Txns Max.Notional.Cost Net.Trading.PL             MAE
## 1 1999-10-29 2000-09-29      900     900        2          94259.548      5532.9397 -4.3655746e-10
## 2 2002-03-28 2002-04-30      900     900        2          81069.138     -4714.6390 -4.7146390e+03
## 3 2003-04-30 2004-08-31      900     900        2          66171.870     15381.7436  0.0000000e+00
## 4 2004-09-30 2007-12-31      900     900        2          82372.165     32256.9001 -1.4551915e-11
## 5 2009-06-30 2010-06-30      900     900        2          74772.618     10772.7225 -4.3655746e-11
## 6 2010-07-30 2010-08-31      900     900        2          91388.149     -4110.6849 -4.1106849e+03
## 7 2010-09-30 2011-08-31      900     900        2          95093.571      8230.6044 -3.9290171e-10
## 8 2012-01-31 2014-06-30      900     900        1         112299.362     63848.6381 -2.6193447e-10
##              MFE Pct.Net.Trading.PL        Pct.MAE       Pct.MFE tick.Net.Trading.PL       tick.MAE
## 1 1.1326653e+04        0.058698984 -4.6314402e-15 1.2016451e-01           614.77108 -4.8506384e-11
## 2 4.3655746e-11       -0.058155781 -5.8155781e-02 5.3850018e-16           -523.84878 -5.2384878e+02
## 3 1.7891989e+04        0.232451399  0.0000000e+00 2.7038663e-01          1709.08263  0.0000000e+00
## 4 3.8233791e+04        0.391599519 -1.7666059e-16 4.6415912e-01          3584.10001 -1.6168795e-12
## 5 2.3226951e+04        0.144073095 -5.8384669e-16 3.1063445e-01          1196.96917 -4.8506384e-12
## 6 5.8207661e-11       -0.044980502 -4.4980502e-02 6.3692789e-16           -456.74276 -4.5674276e+02
## 7 1.9674682e+04        0.086552691 -4.1317379e-15 2.0689813e-01           914.51160 -4.3655746e-11
## 8 6.3848638e+04        0.568557444 -2.3324663e-15 5.6855744e-01          7094.29313 -2.9103830e-11
##         tick.MFE
## 1 1.2585170e+03
## 2 4.8506384e-12
## 3 1.9879988e+03
## 4 4.2481990e+03
## 5 2.5807723e+03
## 6 6.4675179e-12
## 7 2.1860758e+03
## 8 7.0942931e+03
```

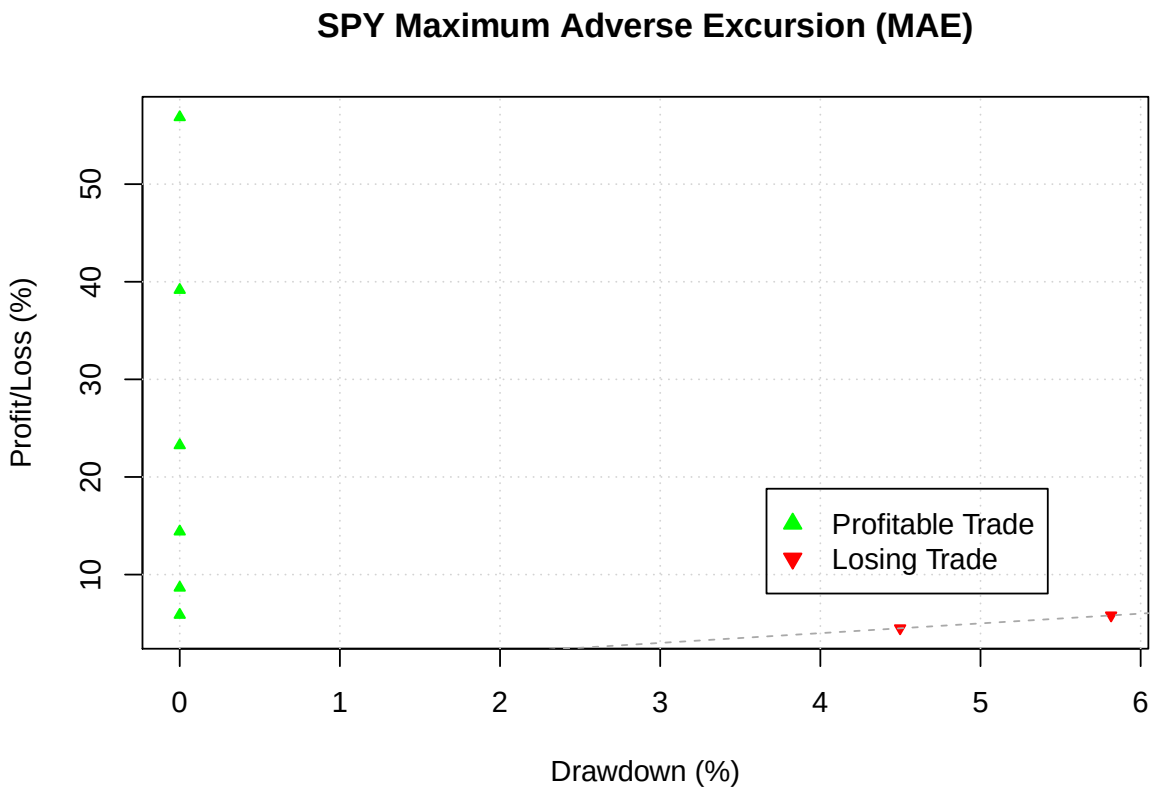
MAE and MFE plots

quantstrat includes the capability to generate maximum adverse excursion (MAE) and maximum favorable excursion (MFE) charts.

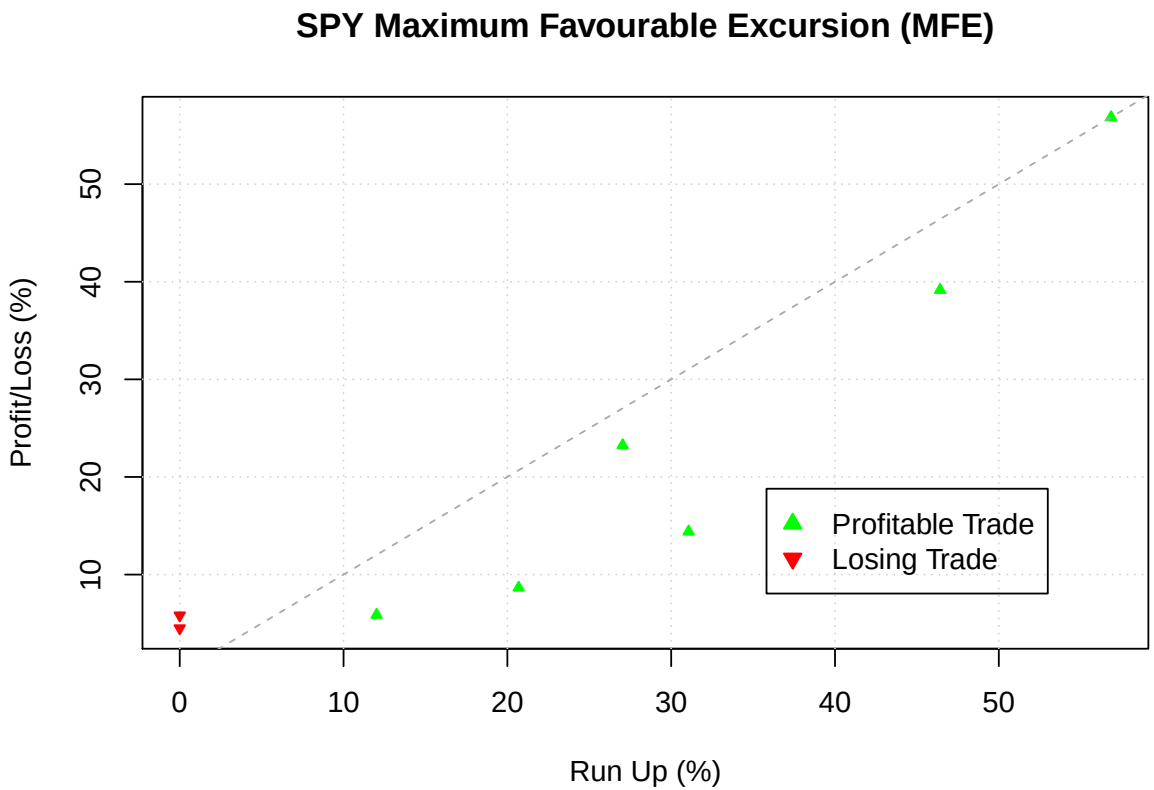
```
chart.ME(Portfolio=qs.strategy, Symbol='SPY', type='MAE', scale='percent')
```

```
chart.ME(Portfolio=qs.strategy, Symbol='SPY', type='MFE', scale='percent')
```

MAE and MFE plots



Performance summary



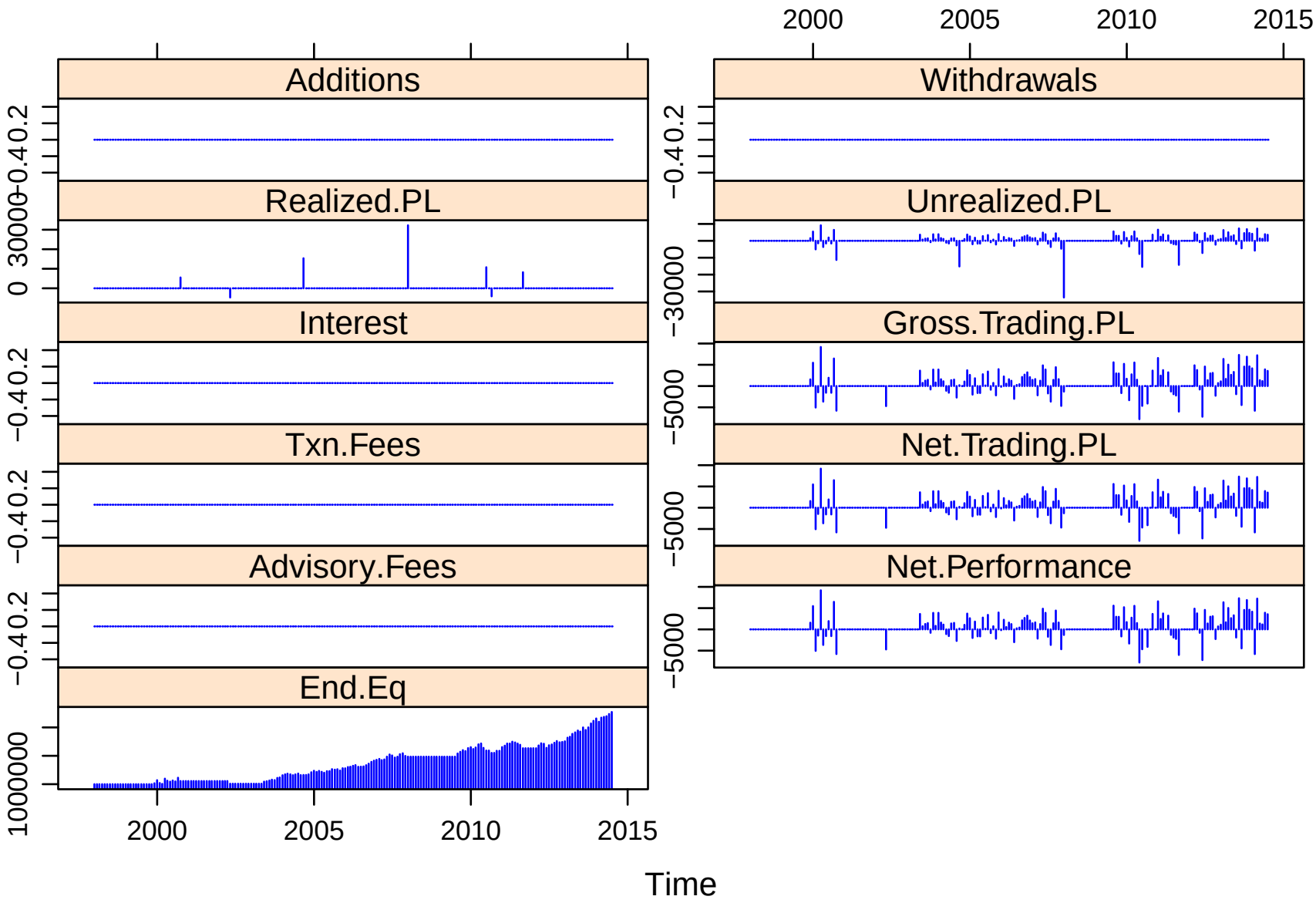
Retrieving the account summary

The function `getAccount` returns the account object

```
a <- getAccount(qs.strategy)
last(a$summary,5)


##            Additions Withdrawals Realized.PL Unrealized.PL Interest Gross.Trading.PL Txn.Fees
## 2014-02-28         0           0           0       7232.27        0        7232.2717        0
## 2014-03-31         0           0           0       1378.12        0        1378.1224        0
## 2014-04-30         0           0           0       1164.42        0        1164.4203        0
## 2014-05-30         0           0           0       3914.24        0        3914.2438        0
## 2014-06-30         0           0           0       3562.99        0        3562.9902        0
##            Net.Trading.PL Advisory.Fees Net.Performance    End.Eq
## 2014-02-28      7232.2717             0       7232.2717 1117178.4
## 2014-03-31      1378.1224             0       1378.1224 1118556.6
## 2014-04-30      1164.4203             0       1164.4203 1119721.0
## 2014-05-30      3914.2438             0       3914.2438 1123635.2
## 2014-06-30      3562.9902             0       3562.9902 1127198.2


library(lattice)
xyplot(a$summary,type="h",col=4)
```

- Use `xyplot` to display the entire account summary multivariate time series

Account summary time series object



Plot equity curve and performance chart

With the full portfolio-level equity time series, any type of performance or risk analysis is readily available.

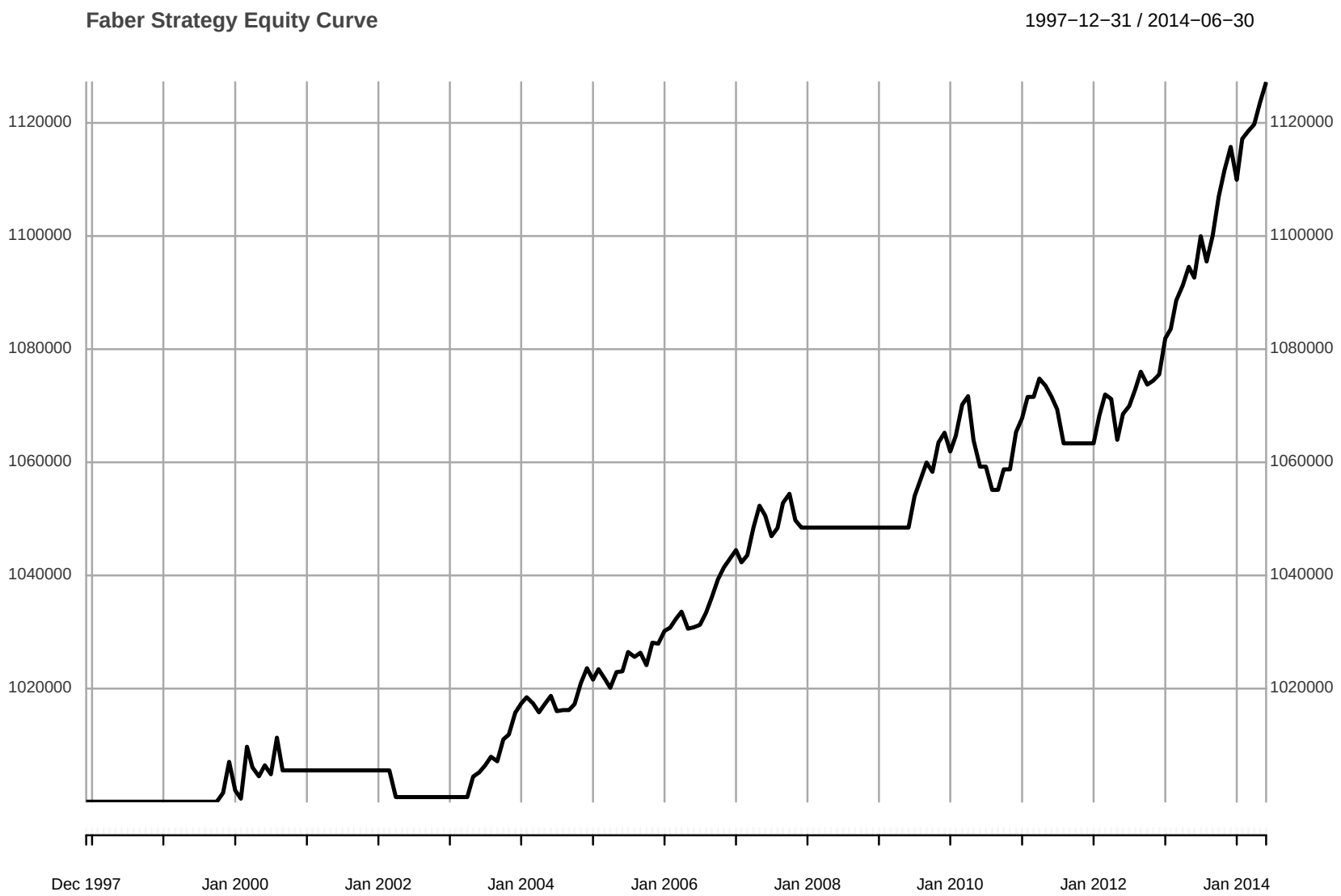
```
equity <- a$summary$End.Eq
```

```
plot(equity,main="Faber Strategy Equity Curve")
```

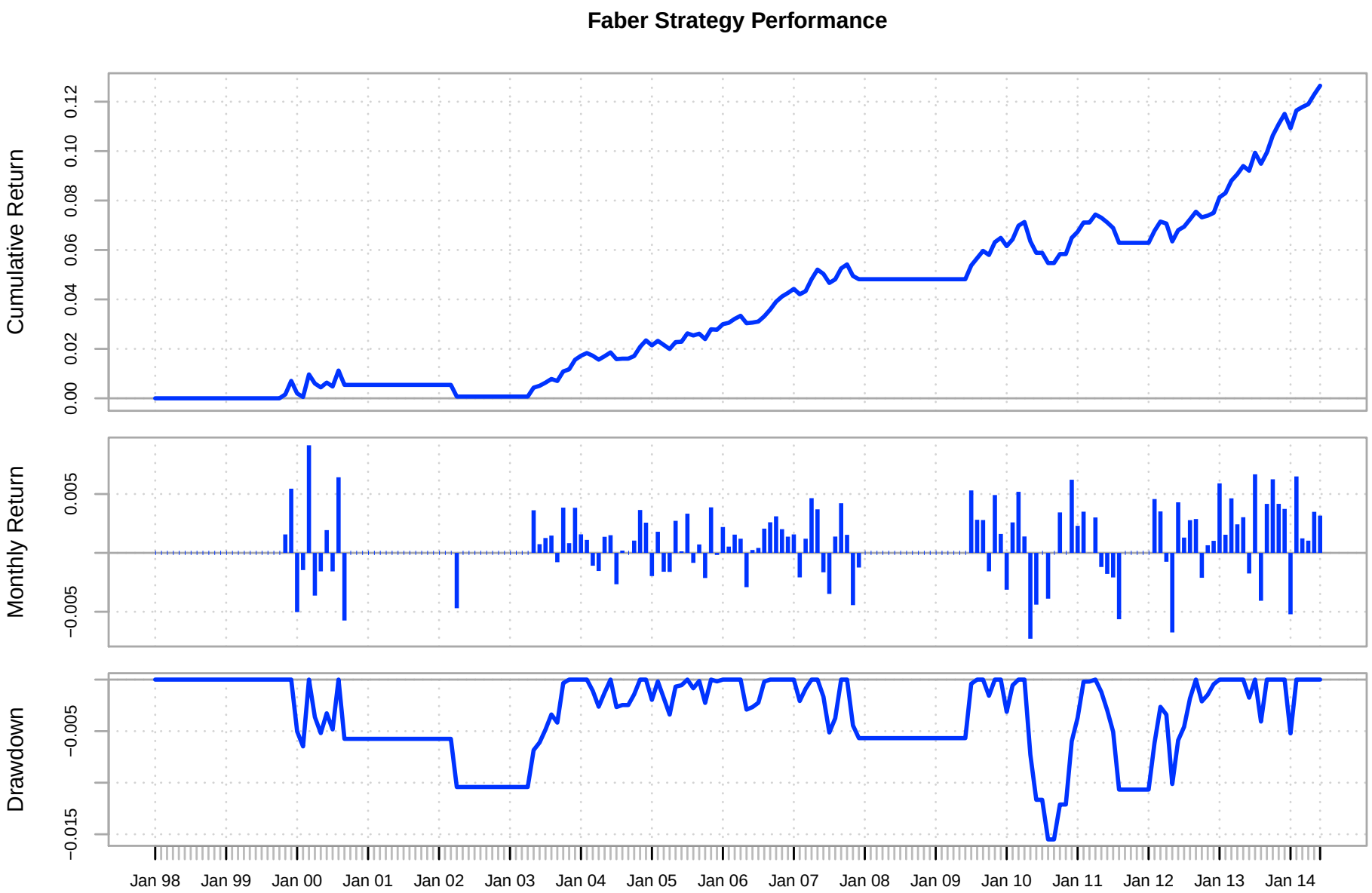
```
ret <- Return.calculate(equity,method="log")
```

```
charts.PerformanceSummary(ret, colorset = bluefocus,
  main="Faber Strategy Performance")
```

Equity curve



Performance summary



Outline

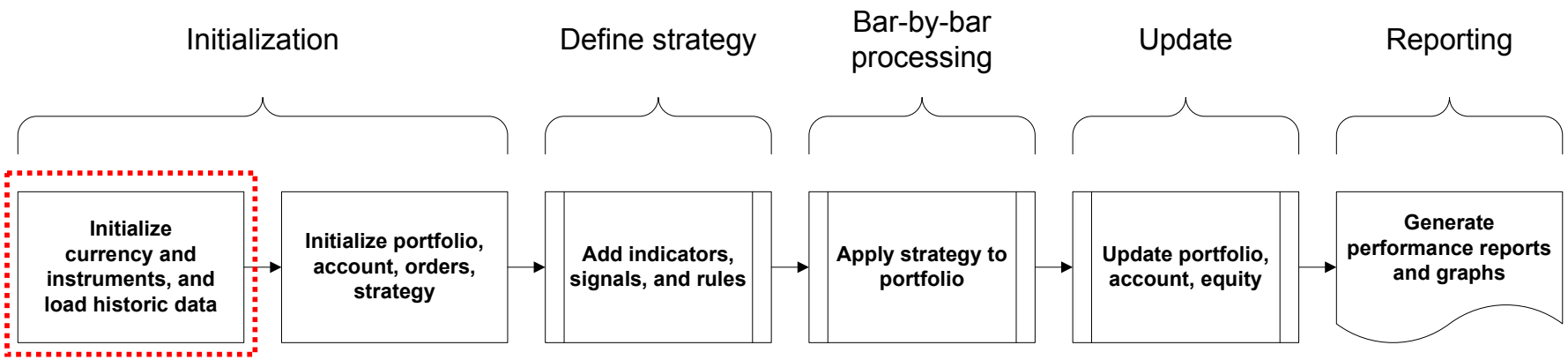
Multi-asset portfolios

The real power of the quantstrat/blotter framework comes from its ability to apply a quantitative trading strategy to a multi-asset portfolio and analyze performance and risk at both the component and portfolio level.

In the following example, we'll use a 9-asset portfolio composed of the 9 Select Sector SPDRs that divide the S&P 500 into nine 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

Fetch historic data

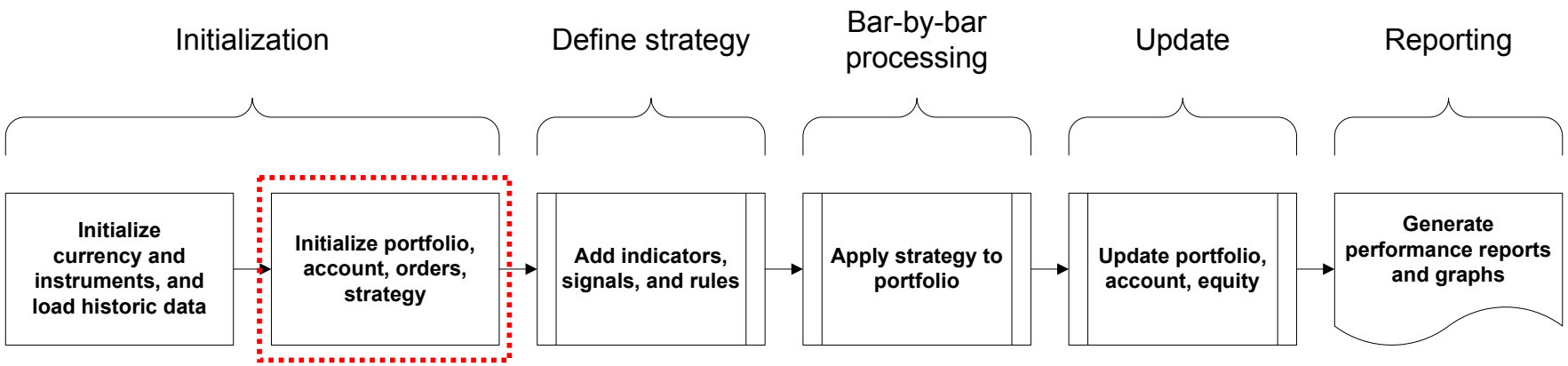



```
symbols = c("XLF", "XLP", "XLE", "XLY", "XLV", "XLI", "XLB", "XLK", "XLU")
```

```
getSymbols(symbols, src='yahoo', index.class=c("POSIXt","POSIXct"),
  from=startDate, to=endDate, adjust=T)
```

```
for(symbol in symbols)
{
    stock(symbol, currency="USD",multiplier=1)
    x<-get(symbol)
    x<-to.monthly(x,indexAt='endof',drop.time=FALSE)
    indexFormat(x)<-'%Y-%m-%d'
    colnames(x)<-gsub("x",symbol,colnames(x))
    assign(symbol,x)
}
```

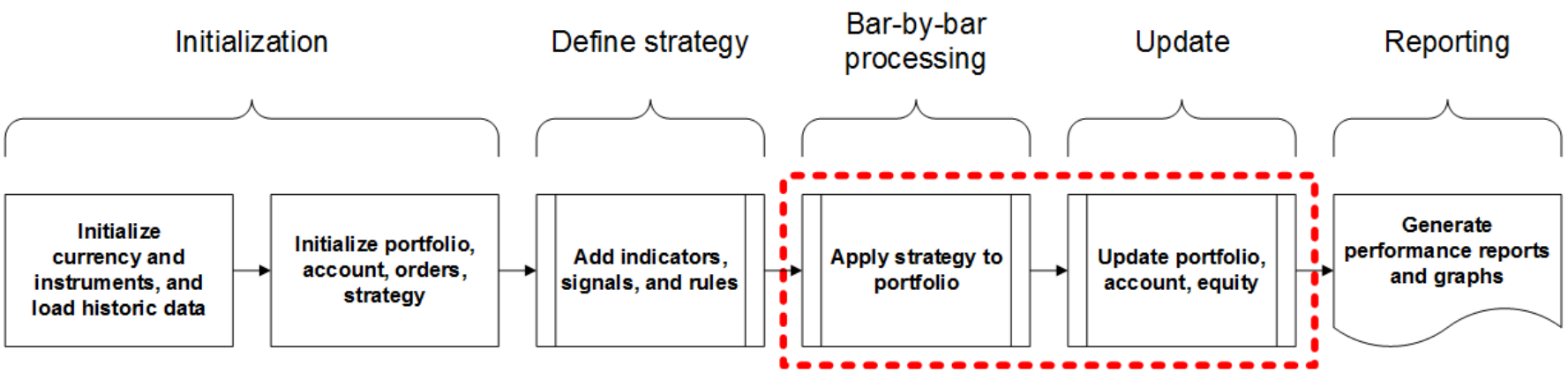
Initialize new portfolio, account, and orders objects




```
multi.asset <- "multiAsset"
rm.strat(multi.asset) # remove strategy etc. if this is a re-run
```

```
initPortf(multi.asset,symbols=symbols, initDate=initDate)
initAcct(multi.asset,portfolios=multi.asset, initDate=initDate,
  initEq=initEq)
initOrders(portfolio=multi.asset,initDate=initDate)
```

Applying strategy to a multi-asset portfolio




```
applyStrategy(strategy=qs.strategy , portfolios=multi.asset)
updatePortf(multi.asset)
updateAcct(multi.asset)
updateEndEq(multi.asset)
```

```
checkBlotterUpdate(multi.asset,multi.asset)

## [1] TRUE
```

- Previously defined strategy (qs.strategy) applied to newly defined portfolio (multi.asset)

Multiple assets in the portfolio

```
a <- getAccount(multi.asset)
p <- getPortfolio(multi.asset)
names(p$symbols)

## [1] "XLB" "XLE" "XLF" "XLI" "XLK" "XLP" "XLU" "XLV" "XLY"
```

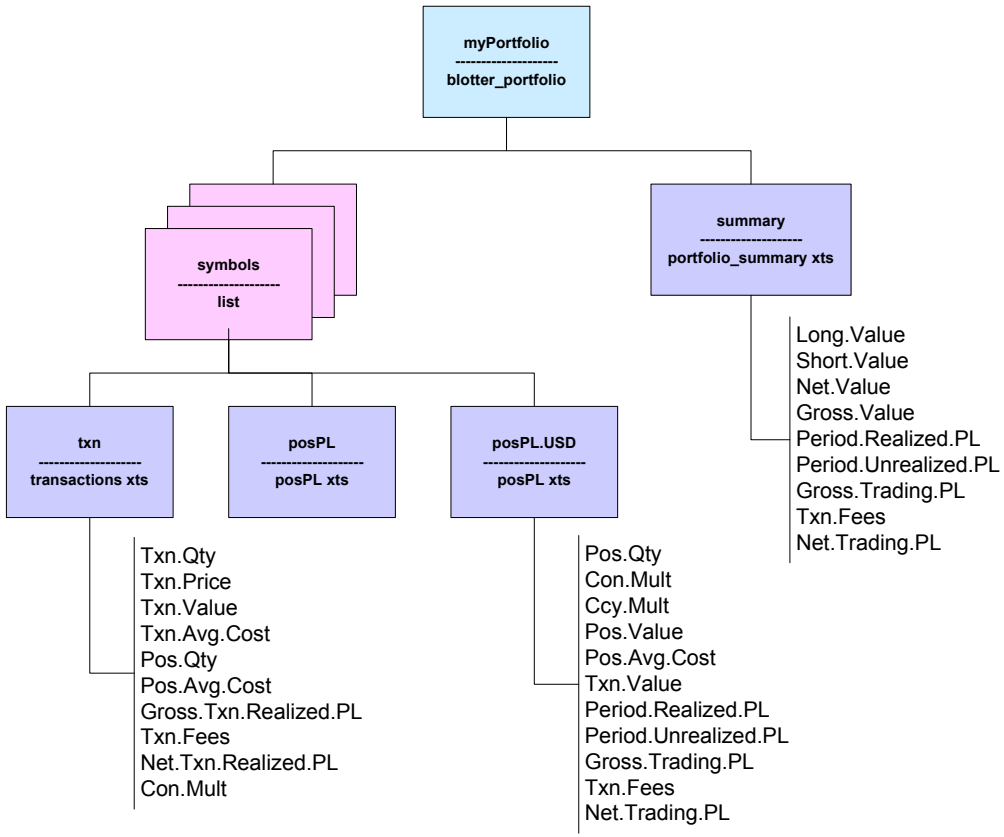
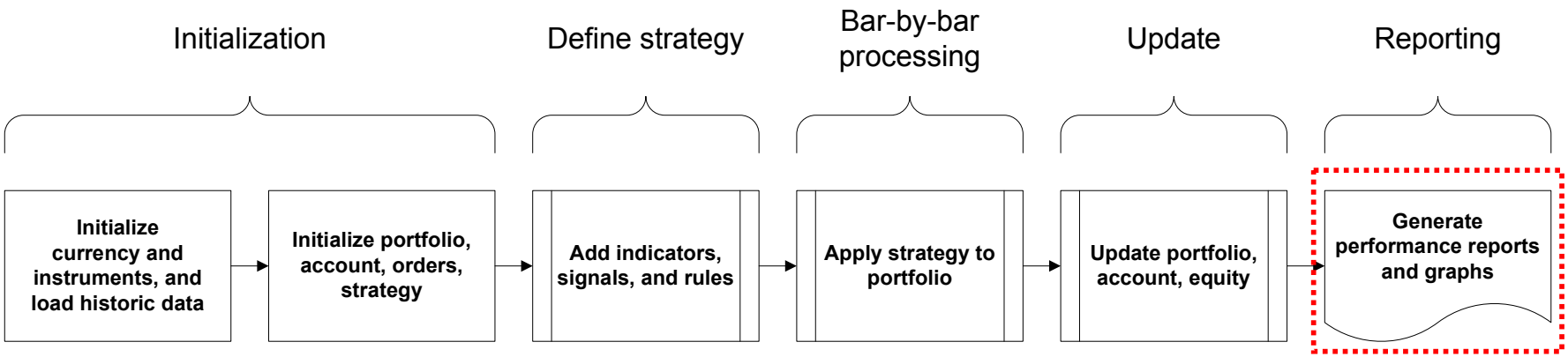
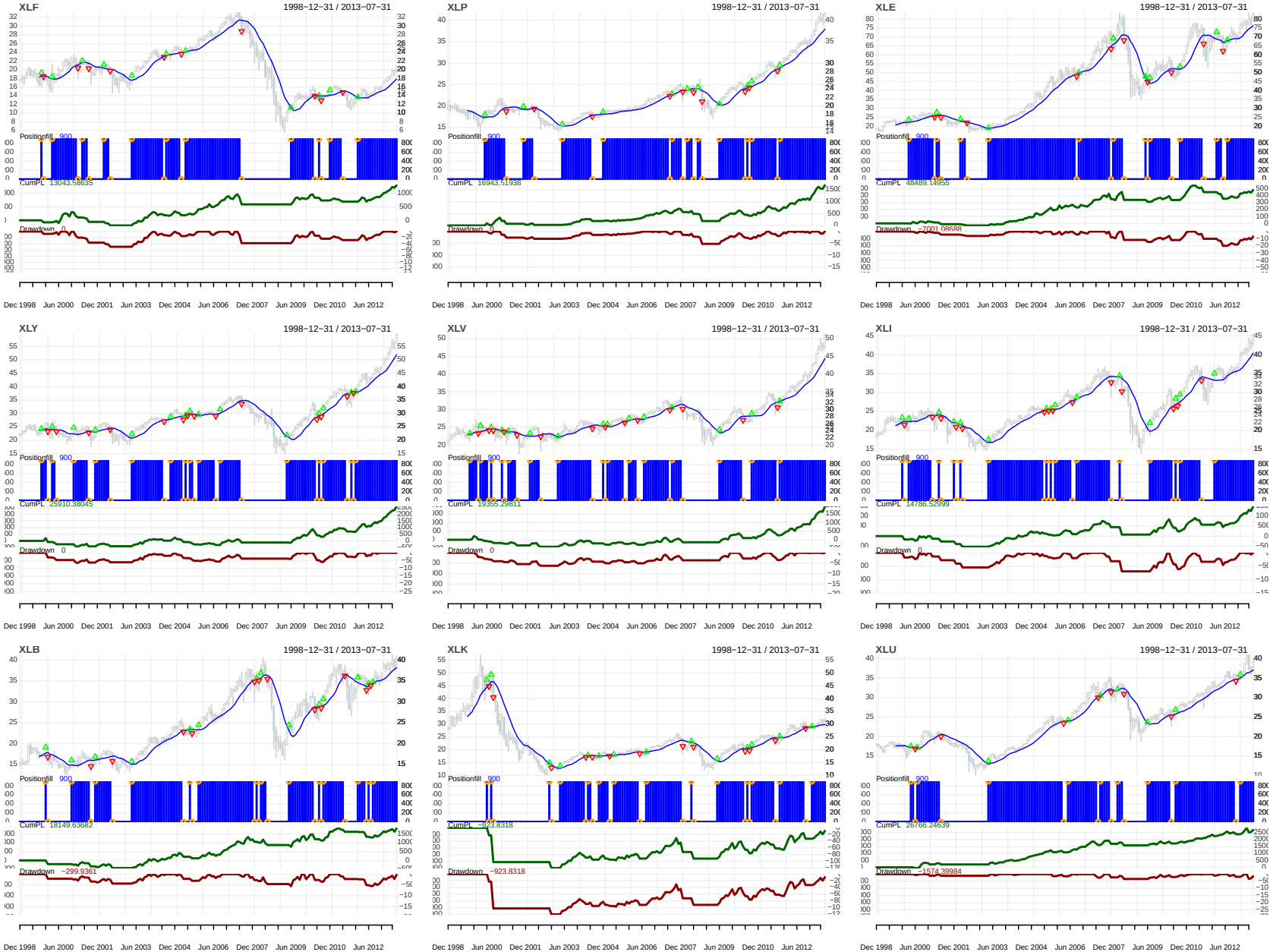


Chart individual asset performance




```
par(mfrow=c(3,3))
for(symbol in symbols)
{
  chart.Posn(Portfolio=multi.asset,Symbol=symbol,theme=myTheme,
    TA="add_SMA(n=10,col='blue')")
}
par(mfrow=c(1,1))
```

Performance of Select Sector SPDRs



Trade stats by instrument

	XLB	XLE	XLF	XLI	XLK	XLP	XLU	XLV	XLY
Portfolio	multiAsset	multiAsset	multiAsset	multiAsset	multiAsset	multiAsset	multiAsset	multiAsset	multiAsset
Symbol	XLB	XLE	XLF	XLI	XLK	XLP	XLU	XLV	XLY
Num.Txns	27	21	21	29	27	21	17	29	29
Num.Trades	13	10	10	14	13	10	8	14	14
Net.Trading.PL	18149.6368	48489.1495	13043.5863	14786.5300	−923.8318	16943.5194	26766.2464	19355.2981	25910.3805
Avg.Trade.PL	998.96885	3582.85189	693.46967	406.87377	−250.84503	624.56664	2967.42208	185.51096	486.30481
Med.Trade.PL	−802.3981910	167.1163589	−219.8554014	−438.1438077	−9.3723334	589.9618737	1973.6004956	−393.1114261	−355.2561579
Largest.Winner	9329.6832	26207.9466	4034.0120	6563.8451	3086.9581	3316.9884	8727.9995	2449.9046	5368.5295
Largest.Loser	−2808.1633	−9635.3881	−1621.4767	−3726.5390	−7977.6444	−3007.5398	−1101.8045	−1158.0362	−1678.8730
Gross.Profits	24031.3814	53694.8931	12076.3098	17795.7287	12348.8629	10925.1576	25406.8732	8362.3529	15574.8896
Gross.Losses	−11044.7864	−17866.3743	−5141.6131	−12099.4959	−15609.8484	−4679.4912	−1667.4966	−5765.1994	−8766.6223
Std.Dev.Trade.PL	3757.2460	10302.0010	2142.5179	2798.5088	2990.3737	1889.0556	3548.9713	1245.1956	2385.2511
Percent.Positive	30.769231	50.000000	40.000000	42.857143	46.153846	60.000000	75.000000	42.857143	35.714286
Percent.Negative	69.230769	50.000000	60.000000	57.142857	53.846154	40.000000	25.000000	57.142857	64.285714
Profit.Factor	2.17581224	3.00536037	2.34873949	1.47078266	0.79109436	2.33468922	15.23653705	1.45048805	1.77661237
Avg.Win.Trade	6007.8453	10738.9786	3019.0775	2965.9548	2058.1438	1820.8596	4234.4789	1393.7255	3114.9779
Med.Win.Trade	5686.6225	11481.5751	3123.4865	3354.0868	2300.3273	1916.7988	3905.5604	1700.5876	3995.5455
Avg.Losing.Trade	−1227.19849	−3573.27485	−856.93552	−1512.43699	−2229.97834	−1169.87280	−833.74828	−720.64993	−974.06914
Med.Losing.Trade	−927.42491	−2489.34160	−872.96902	−1369.38922	−1776.34241	−629.59026	−833.74828	−768.32467	−1012.12435
Avg.Daily.PL	998.96885	3582.85189	693.46967	406.87377	−271.74878	693.96294	2967.42208	199.78103	567.35561
Med.Daily.PL	−802.39819	167.11636	−219.85540	−438.14381	153.13779	452.34288	1973.60050	−367.76002	−179.40849
Std.Dev.Daily.PL	3757.2460	10302.0010	2142.5179	2798.5088	3077.4194	2306.8152	3548.9713	1186.7321	2394.9023
Ann.Sharpe	4.2206815	5.5208702	5.1381087	2.3079866	−1.4017843	4.7755538	13.2732450	2.6724023	3.7606925
Max.Drawdown	−5564.0242	−19661.7175	−4936.8513	−6812.0480	−12017.5942	−5201.5280	−3398.7624	−6233.6863	−6524.6937
Profit.To.Max.Draw	3.261962207	2.466170589	2.642086148	2.170643842	−0.076873273	3.257411915	7.875291915	3.104952236	3.971125940
Avg.WinLoss.Ratio	4.89557754	3.00536037	3.52310923	1.96104354	0.92294342	1.55645948	5.07884568	1.93398407	3.19790226
Med.WinLoss.Ratio	6.1316258	4.6122939	3.5780039	2.4493306	1.2949796	3.0445179	4.6843400	2.2133710	3.9476824
Max.Equity	18449.573	55490.236	13043.586	14786.530	0.000	16943.519	28340.646	19355.298	25910.380
Min.Equity	−4.3309987e+03	−3.0397492e+03	−1.9201746e+03	−5.3296743e+03	−1.2017594e+04	−1.6370905e−11	−5.6569207e+02	−4.0790679e+03	−4.4412056e+03
End.Equity	18149.6368	48489.1495	13043.5863	14786.5300	−923.8318	16943.5194	26766.2464	19355.2981	25910.3805

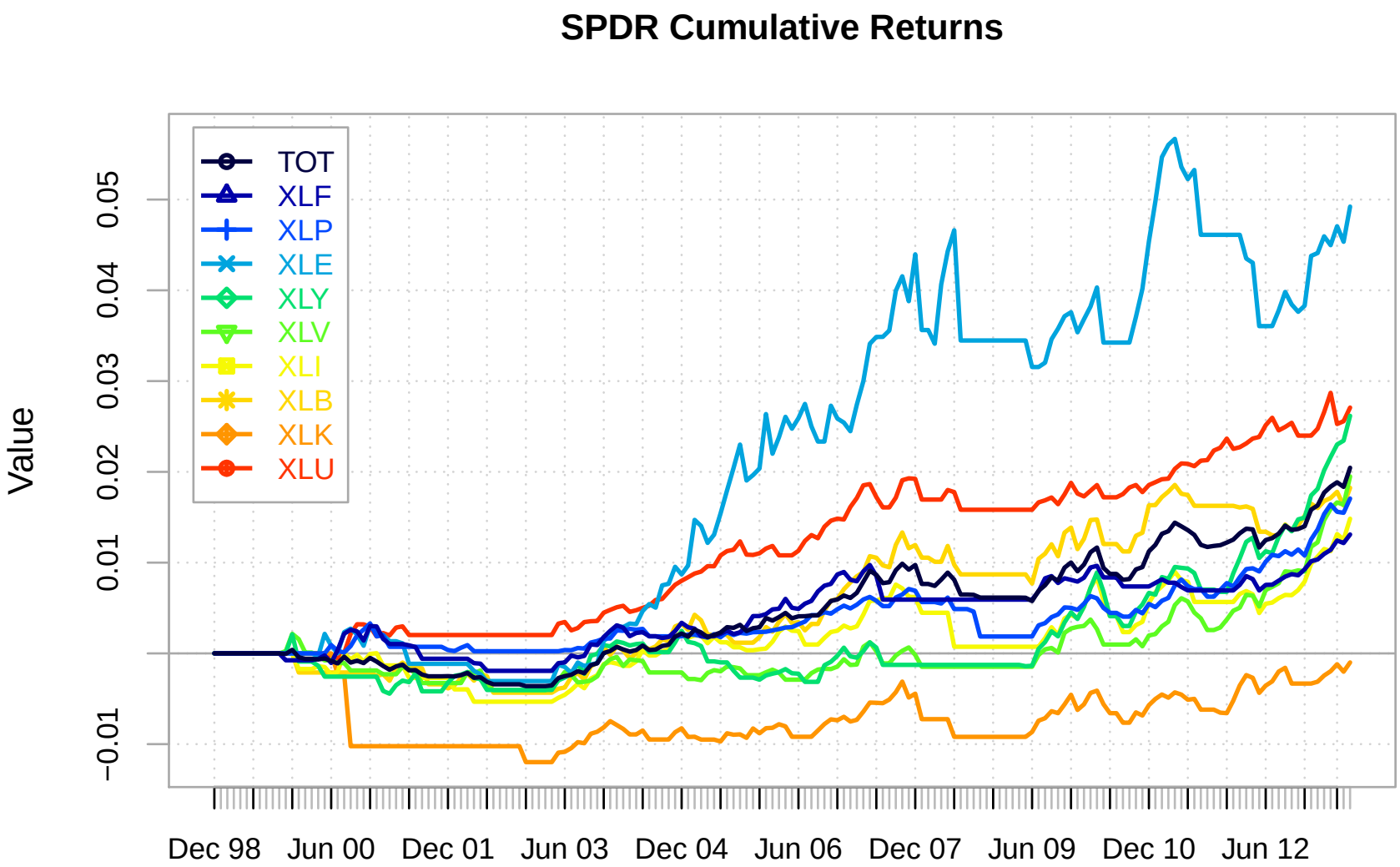
Individual asset returns

```
rets.multi <- PortfReturns(multi.asset)
colnames(rets.multi) <- sort(symbols)
rets.multi <- cbind(rets.multi,TOT=apply(rets.multi,1,sum)/length(symbols))
rets.multi <- rets.multi[,c("TOT",symbols)]
round(tail(rets.multi,5),6)
```

```
##                  TOT       XLF       XLP       XLE      XLY       XLV
## 2013-03-28  0.001386  0.000609  0.001670  0.001762 0.001993  0.002408
## 2013-04-30  0.000654  0.000439  0.001046 -0.000932 0.001444  0.001236
## 2013-05-31  0.000471  0.001022 -0.000796  0.001997 0.001355  0.000699
## 2013-06-28 -0.000496 -0.000279 -0.000108 -0.001634 0.000413 -0.000255
## 2013-07-31  0.002068  0.000936  0.001548  0.003708 0.002673  0.003069
##                 XLI       XLB       XLK       XLU
## 2013-03-28  0.000821  0.000757  0.000681  0.001775
## 2013-04-30 -0.000233  0.000330  0.000475  0.002076
## 2013-05-31  0.001862  0.000670  0.000770 -0.003341
## 2013-06-28 -0.000602 -0.001479 -0.000820  0.000299
## 2013-07-31  0.002268  0.001917  0.001026  0.001467
```

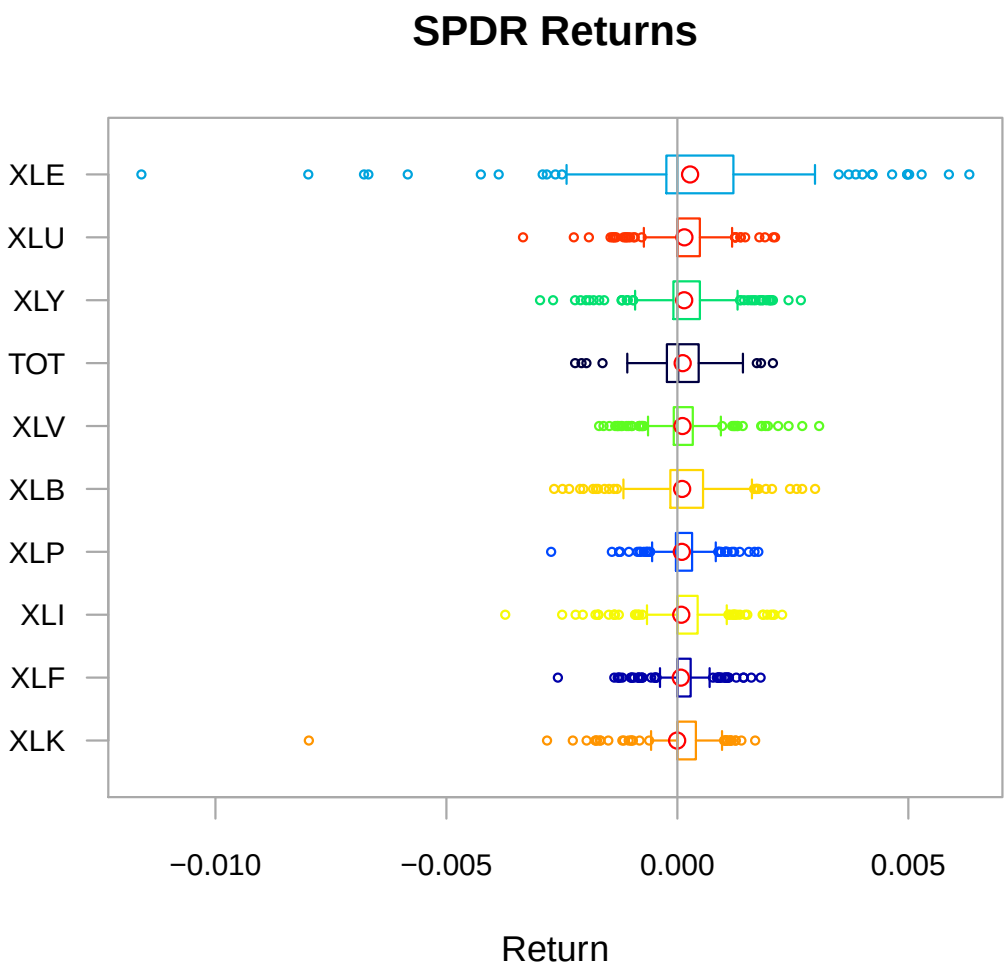
```
chart.CumReturns(rets.multi, colorset= rich10equal, legend.loc = "topleft",
  main="SPDR Cumulative Returns")
```

Cumulative returns by asset



Return distribution analysis

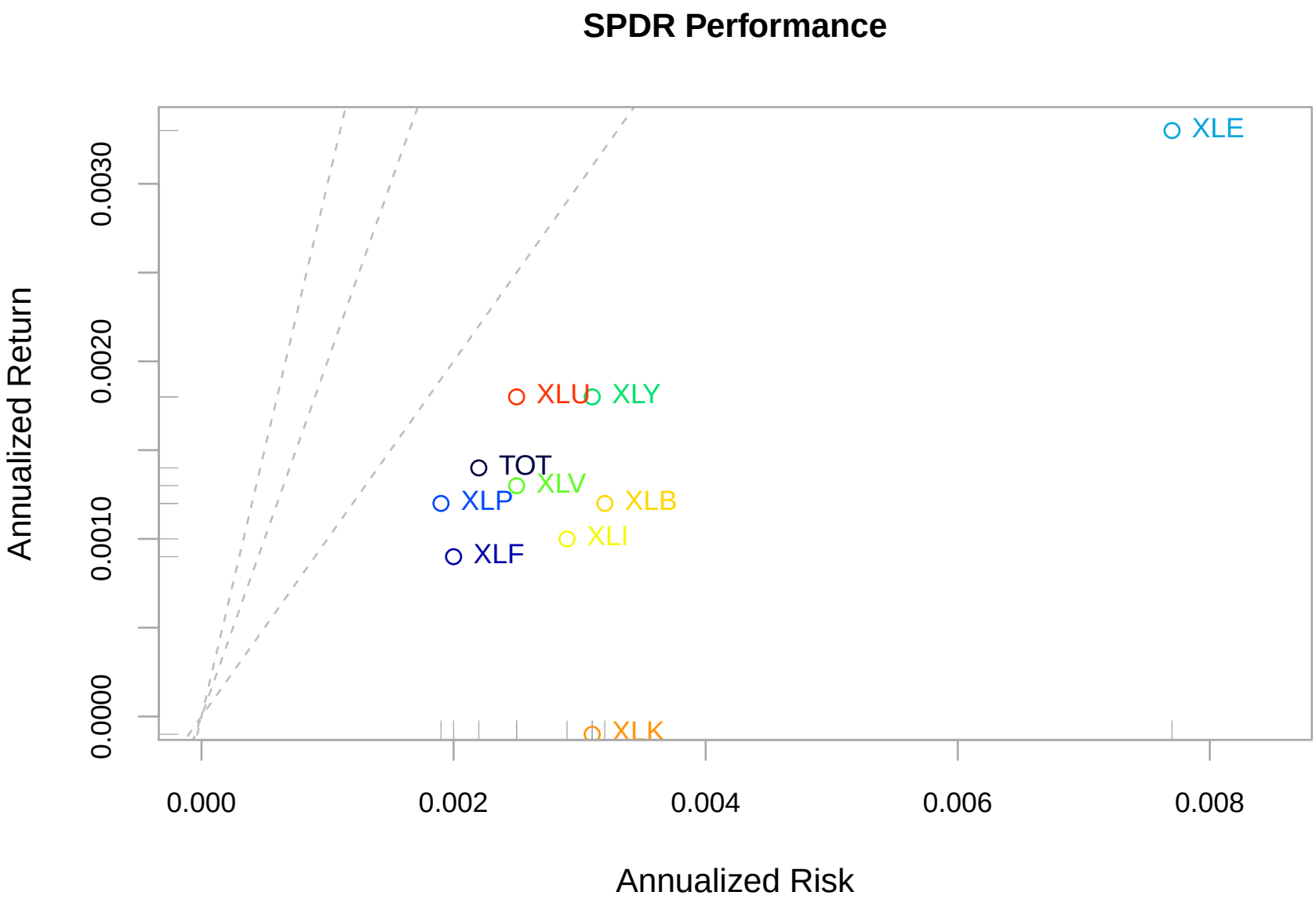
```
chart.Boxplot(rets.multi, main = "SPDR Returns", colorset= rich10equal)
```



Annulized risk and return

```
(ar.tab <- table.AnnualizedReturns(rets.multi))

##                                TOT    XLF    XLP    XLE    XLY    XLV    XLI
## Annualized Return           0.0014 0.0009 0.0012 0.0033 0.0018 0.0013 0.0010
## Annualized Std Dev          0.0022 0.0020 0.0019 0.0077 0.0031 0.0025 0.0029
## Annualized Sharpe (Rf=0%) 0.6326 0.4494 0.6009 0.4268 0.5615 0.5207 0.3516
##                                XLB     XLK    XLU
## Annualized Return           0.0012 -0.0001 0.0018
## Annualized Std Dev          0.0032  0.0031 0.0025
## Annualized Sharpe (Rf=0%) 0.3910 -0.0218 0.7407

max.risk <- max(ar.tab["Annualized Std Dev",])
max.return <- max(ar.tab["Annualized Return",])

chart.RiskReturnScatter(rets.multi,
  main = "SPDR Performance", colorset = rich10equal,
  xlim=c(0,max.risk*1.1),ylim=c(0,max.return))
```

Risk-return scatter plot



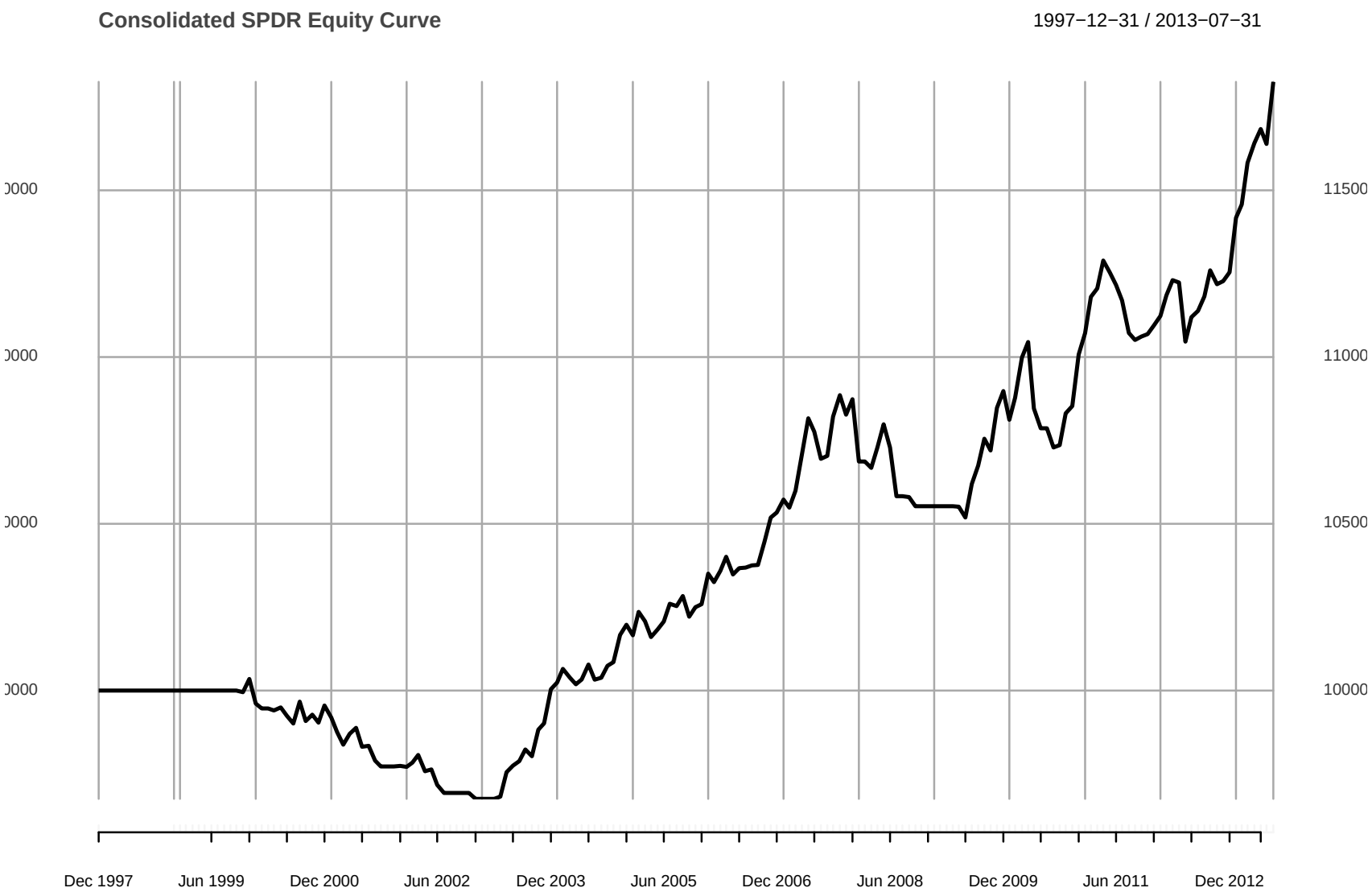
Consolidated equity curve

The `End.Eq` column from the account summary time series represents the consolidated equity value across all portfolios and all of their assets.

```
equity <- a$summary$End.Eq
```

```
plot(equity,main="Consolidated SPDR Equity Curve")
```

Equity curve



Outline

Wrap up

- Homework
  - Assignment #4 will be posted today and due next Thursday
- Mid-term exam
  - Puget-Sound area students:
    - Thursday May 7, 2015 @ 1:30 PM PDT in Lowe 202
  - Out-of-state students:
    - Submit your completed proctor form this week
    - Schedule test for Thursday May 7, 2015 with your proctor

Wrap up

- Reading
  - Flash Boys Chapters 1-3
- Next lecture
  - More on the `quantstrat` package
- Questions, comments, concerns
  - Post to the discussion forum on Canvas
  - Xin, chenx26@uw.edu
  - Guy, gyollin@uw.edu

COMPUTATIONAL FINANCE & RISK MANAGEMENT

UNIVERSITY *of* WASHINGTON

Department of Applied Mathematics

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

COMPUTATIONAL FINANCE & RISK MANAGEMENT

UNIVERSITY *of* WASHINGTON

Department of Applied Mathematics

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