Quantstrat package – R studio

Account -> Portofolio -> Strategy

Na account may contain one or more multiple portfolios. And each portfolio may contain one or more multiple strategies. Each strategy runs on a specific portfolio and account

1. Initialize Porfolio

Initialization means to create the objects in the context of the quantstrat package. To create the portfolio object, we need to provide the portfolio name, stocks, symbols or intruments.

Ex: initPort(name=…, symbols,, iniDate=..

#name is the string name of the portfolio

#symbols : tickers that the portfolio store

# initDate: initial data of the backtesting

1. Initialize Account

Define the account object. The account can be related to one or more than one more portfolios.

Ex: iniAcct(name=..,portfolios, initDate=.., initEq=..

#name: is a string that identifies the account name

#portfolios: is a vector of strings with the names of the portfolios

#initDate: is the initial date

#initEq: is the initial equity of the portfolio

1. Initialize order and Strategy

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

Ex: initOrders(portfolio= portfolioName, symbols= symbolstring, initDate = initDate)

Strategy(name, assets, constrains, store)

#name : the string name of the strategy

#assets : optional list of assets

#contstrains: optional portfolio constrains

#store: can be T or F. If T store the strategy in the environment. F is the default

Strategy(strategyName, store = T)

1. Adding Indicators

The indicators are path independent, which means that they do not depend on account charateristcs.

In quantstrat, we use the add.indicator() function to add indicators.

Ex: add.inicator( strategy, name, arguments, and label)

#strategy: is the string name of the strategy

#name: is the indicator function

#arguments: has a list with the required inputs to calculate

add.indicator(strategy = strategyName, name=”EMA”, arguments= list(x=quote(Cl(mktdata)), n=10), label =”nFast”)

**mktdata** – is an internal object that as OHLCV information loaded in the environment (when we get data using getSymbols), and every time we add an indicator or a signal, a new column is generated in the mktdata object. The label argument would be the column in the mktdata object in the indicator values.

Cl(mktdata)[,1] returns the first column named Close.

Cl is a short-hand for Close, and there several short-hand fuctions for our variables:

* Op(): Open
* Hi(): High
* Lo(): Low
* Cl(): Close
* Vo(): Volume
* Ad(): Adjusted
* OpCl(): Open and Close (n x 2 dataset)
* HLC(): High, Low and Close (n x 3 dataset)

Quote() is a R function that wraps the supplied parameters in quotes.

1. Adding Signals

The next step is to add signals which are genetared from the indicators.

In quantstrat, a signal is added using the add.signal() function :

Add.signal(strategy, name, arguments, label)

#name: the quantstrat has the following built in functions to generate:

* SigComparison – Boolean, compare two variables by relationship:
  + gt – greather than
  + lt – less than
  + eq – equal to
  + gte – greather than or equal to
  + lte – less than or equal to
* SigCrossover: Boolean, True when one signal crosser another. Uses the same relationships as SigComparison.
* SigFormula: apply a formula to multiple variables
* SigPeak – identify local minima or maxima of an indicator
* SigTreshold: Boolean when an indicator crosses a value. Uses relationships as identified above.
* SigTimestamp: generates a signal based on a timestamp.

Ex: add.signal(strategy = strategyName, name=”sigCrossover”, arguments = list(columns = c(“nFast”, “nSlow”), relationship = “gte”, label =”longSignal”).

1. Adding Rules

Rules use market data, indicators, signals, and current account/portfolio properties generate orders. The strategy rules are path dependent because they are related with the portfolio, and with the orders and risk management of the strategy.

In quantsrat, the add.rule() function is used to generate rules and determines the position we take based on our signals, the types of orders we place, the trade size, and transactions costs of the broker. The rules allow functionality to enter positions(long and short) and to exit positions (long and short).

Ex: add.rule(strategyName, name=”ruleSignal”, arguments=list(sigcol=”longSignal”, sigval=TRUE, orderqty=100, ordertype=”market”, orderside=”long”, replace=TRUE, TxnFees= -10), type=”enter”, label=”EnterLong”)

The above rule establishes that when the longsignal column the mktdata object is TRUE, the strategy will send a market order with an order quantity of 100.

It is possible to customize many features with the add.rule() functions, such as the trade size, the preferred price (open, high, close, low) to trigger an order, transactions fees, stop loss and take profit among others.

The ordertype parameter specifies the type of the order (market, limit, stop loss, stop limit, trailing stop). The replace command tells that this order would replace any open order if it is equal to TRUE.

In quantstrat, market orders are executed at the next candle after receiving the signal.

1. Apply Strategy to Portfolio

To run the strategy we shoud use the applyStrategy() method. The applyStrategy function applies the strategy to arbitrary market data. The function takes three main arguments:

* Strategy: an object of type ‘strategy’
* Portfolios: a list of portfolios to apply the strategy to
* Symbols: the symbol of the instrument

Results <- applyStategy(strategy = strategyName, portfolios = portfolioName, symbols= symbolstring)

Executing the applyStrategy also creates a special variable called mktdata. It is a time series object which contains the historic price data as well as the calculated indicators, signals, and rules.

1. Update Portfolio, account and Equity

updatePortf() calculates the Profit and loss for each symbol in symbols

updateAcct() calculates the equity from the portfolio data

updateEndEq() updates the ending equity for the account

ex:

updatePortf(portfolioName)

dateRange <- time(getPortfolio(portfolioaName)$summary)[-1]

updateAcct(portfolioName, dateRange)

updateEndEq(accountName)

1. Plot the Strategy Performance

We can plot the performance using the chart.Posn(). The chart.Posn() takes two parameters that are portfolio name and the symbol string and return a chart with the symbol price series, the accumulated profit loss and the Drawdown charts. The trades of the strategy are marked in green and red on the price series chart.

Ex: (chart performance of the strategy)

Chart.Posn(portfolioName, Symbol = symbolstring)

1. Strategy Statistics

tradeStats() function calculates statistcs about the strategy. These statistics are related to measure about the strategy profits, risk metrics and general features of the strategy such as number of transactions, average profit per trade among others.

1. Portfolio Returns