SCM Repository

[blotter] / pkg / blotter / R / tradeStats.R

View of /pkg/blotter/R/tradeStats.R

Parent Directory | Revision Log

Revision **1741** - (**download**) (**annotate**)
Wed Mar 30 21:37:25 2016 UTC (11 months, 3 weeks ago) by *bodanker*
File size: 19936 byte(s)

```
Determine 'trades' without fees, with scratches

Transaction fees are realized on every transaction, so do not use
non-zero net realized P&L to determine 'trades' for tradeStats. Use
non-zero gross realized P&L instead.

Also include scratch trades for tradeStats' purposes. The amzn_demo
has 5 scratch trades (of 7 total), and that showed the number of
transactions (among other things) were not being calculated correctly.
```

```
#' calculate statistics on transactions and P&L for a symbol or symbols in a portfolio or portfolios
#'
#' This function calculates trade-level statistics on a symbol or symbols within a portfolio or portfolios.
#'
#' Every book on trading, broker report on an analytical trading system,
#' or blog post seems to have a slightly different idea of what trade statistics
#' are necessary, and how they should be displayed.  We choose not to make
#' value judgments of this type, aiming rather for inclusiveness with
#' post-processing for display.
#'
#' The output of this function is a \code{\link{data.frame}} with named columns for each statistic.
#' Each row is a single portfolio+symbol combination. Values are returned in full precision.
#' It is likely that the output of this function will have more than you wish
#' to display in all conditions, but it should be suitable for reshaping for display.
#' Building summary reports from this data.frame may be easily accomplished using
#' something like \code{textplot} or \code{\link{data.frame}}, with rounding,
#' fancy formatting, etc. as your needs dictate.
#'
#' Option \code{inclZeroDays}, if \code{TRUE}, will include all transaction P&L,
#' including for days in which the strategy was not in the market,
#' for daily statistics.
#' This can prevent irrationally good looking daily statistics for strategies
#' which spend a fair amount of time out of the market.  For strategies which
#' are always in the market, the statistics should be (nearly) the same.
#' Default is \code{FALSE} for backwards compatibility.
#'
#' If you have additional trade statistics you want added here, please share.
#' We find it unlikely that any transaction-level statistics that can be
#' calculated independently of strategy rules could be considered proprietary.
#'
#' Special Thanks for contributions to this function from:
#' \describe{
#'   \item{Josh Ulrich}{ for adding multiple-portfolio support, fixing bugs, and improving readability of the code }
#'   \item{Klemen Koselj}{ for median stats, num trades, and win/loss ratios }
#'   \item{Mark Knecht}{ for suggesting Profit Factor and largest winner/largest loser }
#' }
#'
#' WARNING: we're not sure this function is stable/complete yet.  If you're using it, please give us feedback!
#'
#' @aliases dailyStats
#' @seealso \code{\link{chart.ME}} for a chart of MAE and MFE derived from trades,
#' and \code{\link{perTradeStats}} for detailed statistics on a per-trade basis
#' @param Portfolios portfolio string
#' @param Symbols character vector of symbol strings, default NULL
#' @param use for determines whether numbers are calculated from transactions or round-trip trades (for tradeStats) or equity curve (fo
#' @param tradeDef string to determine which definition of 'trade' to use. Currently "flat.to.flat" (the default) and "flat.to.reduced"
#' @param inclZeroDays TRUE/FALSE, whether to include zero P&L days in daily calcs, default FALSE for backwards compatibility.
#' @author Lance Levenson, Brian Peterson
#' @export
#' @return
#' a \code{data.frame} containing:
#'
#' \describe{
#'    \item{Portfolio}{ name of the portfolio}
#'    \item{Symbol}{ symbol name }
#'    \item{Num.Txns}{ number of transactions produced by \code{\link{addTxn}} }
#'    \item{Num.Trades}{ number of \emph{flat to flat} trades performed }
#'    \item{Net.Trading.PL}{ }
#'    \item{Avg.Trade.PL}{ mean trading P&L per trade }
#'    \item{Med.Trade.PL}{ median trading P&L per trade}
#'    \item{Largest.Winner}{ largest winning trade }
#'    \item{Largest.Loser}{ largest losing trade }
#'    \item{Gross.Profits}{ gross (pre-fee) trade profits }
#'    \item{Gross.Losses}{ gross trade losses }
#'    \item{Std.Dev.Trade.PL}{ standard deviation of trade P&L }
#'    \item{Percent.Positive}{ percent of trades that end positive }
#'    \item{Percent.Negative}{ percent of trades that end negative }
#'    \item{Profit.Factor}{ absolute value ratio of gross profits over gross losses }
#'    \item{Avg.Win.Trade}{ mean P&L of profitable trades }
#'    \item{Med.Win.Trade}{ median P&L of profitable trades }
#'    \item{Avg.Losing.Trade}{ mean P&L of losing trades }
#'    \item{Med.Losing.Trade}{ median P&L of losing trades }
#'    \item{Avg.Daily.PL}{mean daily realized P&L on days there were transactions, see \code{\link{dailyStats}} for all days }
#'    \item{Med.Daily.PL}{ median daily P&L }
#'    \item{Std.Dev.Daily.PL}{ standard deviation of daily P&L }
#'    \item{Ann.Sharpe}{annualized Sharpe-like ratio, assuming no outside capital additions and 252 day count convention}
#'    \item{Max.Drawdown}{ max drawdown }
#'    \item{Avg.WinLoss.Ratio}{ ratio of mean winning over mean losing trade }
#'    \item{Med.WinLoss.Ratio}{ ratio of median winning trade over median losing trade }
#'    \item{Max.Equity}{ maximum account equity }
#'    \item{Min.Equity}{ minimum account equity }
#' }
#' @note
#' TODO document each statistic included in this function, with equations
#'
#' TODO add more stats, potentially
#' PerformanceAnalytics: skewness, kurtosis, upside/downside semidieviation, Sortino
#'
#' mean absolute deviation stats
#'
#' more Tharpe/Kestner/Tradestation stats, e.g.
#' K-factor
#' RINA Index
#' Percent time in the market
#' Buy and hold return
#'
#' Josh has suggested adding \%-return based stats too
tradeStats <- function(Portfolios, Symbols ,use=c('txns','trades'), tradeDef='flat.to.flat',inclZeroDays=FALSE)
{
    ret <- NULL
    use <- use[1] #use the first(default) value only if user hasn't specified
    tradeDef <- tradeDef[1]
    for (Portfolio in Portfolios){
        pname <- Portfolio
        Portfolio<-.getPortfolio(pname)

        if(missing(Symbols)) symbols <- ls(Portfolio$symbols)
        else symbols <- Symbols

        ## Trade Statistics
        for (symbol in symbols){
            txn   <- Portfolio$symbols[[symbol]]$txn
            posPL <- Portfolio$symbols[[symbol]]$posPL
            posPL <- posPL[-1,]

            # Use gross transaction P&L to identify transactions that realized
            # (non-fee) P&L, but use net transaction P&L to calculate statistics.
            PL.gt0 <- txn$Net.Txn.Realized.PL[txn$Gross.Txn.Realized.PL  > 0]
            PL.lt0 <- txn$Net.Txn.Realized.PL[txn$Gross.Txn.Realized.PL  < 0]
            PL.scratch <- txn$Pos.Qty == 0 & lag(txn$Pos.Qty) != 0
            PL.scratch[1] <- FALSE  # Set first NA to FALSE
            PL.ne0 <- txn$Net.Txn.Realized.PL[txn$Gross.Txn.Realized.PL != 0 | PL.scratch]

            if(length(PL.ne0) == 0)
            {
                # apply.daily will crash
                next
            }

            if(!isTRUE(inclZeroDays)) DailyPL <- apply.daily(PL.ne0,sum)
            else DailyPL <- apply.daily(txn$Net.Txn.Realized.PL,sum)

            AvgDailyPL <- mean(DailyPL)
            MedDailyPL <- median(DailyPL)
            StdDailyPL <- sd(as.numeric(as.vector(DailyPL)))

            switch(use,
                   txns = {
                       #moved above for daily stats for now
                   },
                   trades = {
                       trades <- perTradeStats(pname,symbol,tradeDef=tradeDef)
                       PL.gt0 <- trades$Net.Trading.PL[trades$Net.Trading.PL  > 0]
                       PL.lt0 <- trades$Net.Trading.PL[trades$Net.Trading.PL  < 0]
                       PL.ne0 <- trades$Net.Trading.PL[trades$Net.Trading.PL != 0]
                   }
            )
            if(!length(PL.ne0)>0)next()

            GrossProfits <- sum(PL.gt0)
            GrossLosses  <- sum(PL.lt0)
            ProfitFactor <- ifelse(GrossLosses == 0, NA, abs(GrossProfits/GrossLosses))

            AvgTradePL <- mean(PL.ne0)
            MedTradePL <- median(PL.ne0)
            StdTradePL <- sd(as.numeric(as.vector(PL.ne0)))
            AnnSharpe  <- ifelse(StdDailyPL == 0, NA, AvgDailyPL/StdDailyPL * sqrt(252))

            NumberOfTxns   <- nrow(txn)-1
            NumberOfTrades <- length(PL.ne0)

            PercentPositive <- (length(PL.gt0)/length(PL.ne0))*100
            PercentNegative <- (length(PL.lt0)/length(PL.ne0))*100

            MaxWin  <- max(txn$Net.Txn.Realized.PL)
            MaxLoss <- min(txn$Net.Txn.Realized.PL)

            AvgWinTrade  <- mean(PL.gt0)
            MedWinTrade  <- median(PL.gt0)
            AvgLossTrade <- mean(PL.lt0)
            MedLossTrade <- median(PL.lt0)

            AvgWinLoss <- ifelse(AvgLossTrade == 0, NA, AvgWinTrade/-AvgLossTrade)
            MedWinLoss <- ifelse(MedLossTrade == 0, NA, MedWinTrade/-MedLossTrade)

            Equity <- cumsum(posPL$Net.Trading.PL)
            if(!nrow(Equity)){
                warning('No Equity rows for',symbol)
                next()
            }
            TotalNetProfit <- last(Equity)
            if(is.na(TotalNetProfit)) {
                warning('TotalNetProfit NA for',symbol)
                next()
            }
            Equity.max       <- cummax(Equity)
            MaxEquity        <- max(Equity)
            MinEquity        <- min(Equity)
            EndEquity        <- last(Equity)
            names(EndEquity) <-'End.Equity'
            if(EndEquity!=TotalNetProfit && last(txn$Pos.Qty)==0) {
                warning('Total Net Profit for',symbol,'from transactions',TotalNetProfit,'and cumulative P&L from the Equity Curve', Enc
                message('Total Net Profit for',symbol,'from transactions',TotalNetProfit,'and cumulative P&L from the Equity Curve', Enc

            }# if we're flat, these numbers should agree
            #TODO we should back out position value if we've got an open position and double check here....

            MaxDrawdown            <- -max(Equity.max - Equity)
            ProfitToMaxDraw  <- ifelse(MaxDrawdown == 0, NA, -TotalNetProfit / MaxDrawdown)
            names(ProfitToMaxDraw) <- 'Profit.To.Max.Draw'

            #TODO add skewness, kurtosis, and positive/negative semideviation if PerfA is available.

            tmpret <- data.frame(Portfolio=pname,
                                 Symbol             = symbol,
                                 Num.Txns           = NumberOfTxns,
                                 Num.Trades         = NumberOfTrades,
                                 Total.Net.Profit   = TotalNetProfit,
                                 Avg.Trade.PL       = AvgTradePL,
                                 Med.Trade.PL       = MedTradePL,
                                 Largest.Winner     = MaxWin,
                                 Largest.Loser      = MaxLoss,
                                 Gross.Profits      = GrossProfits,
                                 Gross.Losses       = GrossLosses,
                                 Std.Dev.Trade.PL   = StdTradePL,
                                 Percent.Positive   = PercentPositive,
                                 Percent.Negative   = PercentNegative,
                                 Profit.Factor      = ProfitFactor,
                                 Avg.Win.Trade      = AvgWinTrade,
                                 Med.Win.Trade      = MedWinTrade,
                                 Avg.Losing.Trade   = AvgLossTrade,
                                 Med.Losing.Trade   = MedLossTrade,
                                 Avg.Daily.PL       = AvgDailyPL,
                                 Med.Daily.PL       = MedDailyPL,
                                 Std.Dev.Daily.PL   = StdDailyPL,
                                 Ann.Sharpe         = AnnSharpe,
                                 Max.Drawdown       = MaxDrawdown,
                                 Profit.To.Max.Draw = ProfitToMaxDraw,
                                 Avg.WinLoss.Ratio  = AvgWinLoss,
                                 Med.WinLoss.Ratio  = MedWinLoss,
                                 Max.Equity         = MaxEquity,
                                 Min.Equity         = MinEquity,
                                 End.Equity         = EndEquity)
            rownames(tmpret) <- symbol
            ret              <- rbind(ret,tmpret)
        } # end symbol loop
    } # end portfolio loop
    return(ret)
}

#' generate daily Transaction Realized or Equity Curve P&L by instrument
#'
#' designed to collate information for high frequency portfolios
#'
#' If you do not pass \code{Symbols}, then all symbols in the provided
#' \code{Portfolios} will be used.
#'
#' The daily P&L is calculated from \code{Net.Txn.Realized.PL} if by
#' \code{dailyTxnPL}
#' and from \code{Net.Trading.PL} by \code{dailyEqPL}
#'
#' @aliases dailyEqPL
#' @param Portfolios portfolio string
#' @param Symbols character vector of symbol strings
#' @param drop.time remove time component of POSIX datestamp (if any), default TRUE
#' @author Brian G. Peterson
#' @return a multi-column \code{xts} time series, one column per symbol, one row per day
#' @seealso tradeStats
#' @export
dailyTxnPL <- function(Portfolios, Symbols, drop.time=TRUE)
{
    ret <- NULL
    for (Portfolio in Portfolios){
        pname <- Portfolio
        Portfolio <- getPortfolio(pname)


        ## FIXME: need a way to define symbols for each portfolio
        if(missing(Symbols)) symbols <- ls(Portfolio$symbols)
        else symbols <- Symbols

        ## Trade Statistics
        for (symbol in symbols){
            txn <- Portfolio$symbols[[symbol]]$txn
            txn <- txn[-1,] # remove initialization row

            PL.ne0 <- txn$Net.Txn.Realized.PL[txn$Net.Txn.Realized.PL != 0]
            if(!nrow(PL.ne0)){
                warning('No P&L rows for',symbol)
                next()
            }
            DailyPL           <- apply.daily(PL.ne0,sum)
            colnames(DailyPL) <- paste(symbol,'DailyTxnPL',sep='.')
            if(is.null(ret)) ret=DailyPL else ret<-cbind(ret,DailyPL)

        } # end symbol loop
    } # end portfolio loop
    ret <- apply.daily(ret,colSums,na.rm=TRUE)
    if(drop.time) index(ret) <- as.Date(index(ret))
    return(ret)
}

#' @rdname dailyTxnPL
#' @export
dailyEqPL <- function(Portfolios, Symbols, drop.time=TRUE)
{
    ret <- NULL
    for (Portfolio in Portfolios){
        pname <- Portfolio
        Portfolio <- getPortfolio(pname)

        ## FIXME: need a way to define symbols for each portfolio
        if(missing(Symbols)) symbols <- ls(Portfolio$symbols)
        else symbols <- Symbols

        ## Trade Statistics
        for (symbol in symbols){
            posPL <- Portfolio$symbols[[symbol]]$posPL
            posPL <- posPL[-1,] # remove initialization row
            Equity <- cumsum(posPL$Net.Trading.PL)
            if(!nrow(Equity)){
                warning('No P&L rows for',symbol)
                next()
            }

            #DailyPL <- apply.daily(Equity,last)
            DailyPL           <- apply.daily(posPL$Net.Trading.PL,colSums)
            colnames(DailyPL) <- paste(symbol,'DailyEndEq',sep='.')
            if(is.null(ret)) ret=DailyPL else ret<-cbind(ret,DailyPL)

        } # end symbol loop
    } # end portfolio loop
    ret <- apply.daily(ret,colSums,na.rm=TRUE)
    if(drop.time) index(ret) <- as.Date(index(ret))
    return(ret)
}

#' @rdname tradeStats
#' @export
dailyStats <- function(Portfolios,use=c('equity','txns'))
{
    use=use[1] #take the first value if the user didn't specify
    switch (use,
            Eq =, eq =, Equity =, equity =, cumPL = {
                dailyPL <- dailyEqPL(Portfolios)
            },
            Txns =, txns =, Trades =, trades = {
                dailyPL <- dailyTxnPL(Portfolios)
            }
            )

    dailyFUN <- function (x){
        x<-t(t(x))
        PL.gt0 <- x[x  > 0]
        PL.lt0 <- x[x  < 0]
        PL.ne0 <- x[x != 0]

        TotalNetProfit <- sum(x)

        GrossProfits <- sum(PL.gt0)
        GrossLosses  <- sum(PL.lt0)
        ProfitFactor <- abs(GrossProfits/GrossLosses)

        AvgDayPL <- as.numeric(mean(PL.ne0))
        MedDayPL <- as.numeric(median(PL.ne0))
        StdDayPL <- as.numeric(sd(PL.ne0))

        #NumberOfDays <- nrow(txn)
        WinDays         <- length(PL.gt0)
        LossDays        <- length(PL.lt0)
        PercentPositive <- (length(PL.gt0)/length(PL.ne0))*100
        PercentNegative <- (length(PL.lt0)/length(PL.ne0))*100

        MaxWin  <- max(x)
        MaxLoss <- min(x)

        AvgWinDay  <- as.numeric(mean(PL.gt0))
        MedWinDay  <- as.numeric(median(PL.gt0))
        AvgLossDay <- as.numeric(mean(PL.lt0))
        MedLossDay <- as.numeric(median(PL.lt0))

        AvgWinLoss <- AvgWinDay/-AvgLossDay
        MedWinLoss <- MedWinDay/-MedLossDay

        AvgDailyPL <- as.numeric(mean(PL.ne0))
        MedDailyPL <- as.numeric(median(PL.ne0))
        StdDailyPL <- as.numeric(sd(PL.ne0))
        AnnSharpe  <- AvgDailyPL/StdDailyPL * sqrt(252)

        Equity          <- cumsum(x)
        Equity.max      <- cummax(Equity)
        MaxEquity       <- max(Equity)
        MinEquity       <- min(Equity)
        EndEquity       <- as.numeric(last(Equity))
        MaxDrawdown     <- -max(Equity.max - Equity)
        ProfitToMaxDraw <- -TotalNetProfit / MaxDrawdown

        tmpret <- data.frame(
                Total.Net.Profit   = TotalNetProfit,
                Total.Days         = WinDays+LossDays,
                Winning.Days       = WinDays,
                Losing.Days        = LossDays,
                Avg.Day.PL         = AvgDayPL,
                Med.Day.PL         = MedDayPL,
                Largest.Winner     = MaxWin,
                Largest.Loser      = MaxLoss,
                Gross.Profits      = GrossProfits,
                Gross.Losses       = GrossLosses,
                Std.Dev.Daily.PL   = StdDayPL,
                Percent.Positive   = PercentPositive,
                Percent.Negative   = PercentNegative,
                Profit.Factor      = ProfitFactor,
                Avg.Win.Day        = AvgWinDay,
                Med.Win.Day        = MedWinDay,
                Avg.Losing.Day     = AvgLossDay,
                Med.Losing.Day     = MedLossDay,
                Avg.Daily.PL       = AvgDailyPL,
                Med.Daily.PL       = MedDailyPL,
                Std.Dev.Daily.PL   = StdDailyPL,
                Ann.Sharpe         = AnnSharpe,
                Max.Drawdown       = MaxDrawdown,
                Profit.To.Max.Draw = ProfitToMaxDraw,
                Avg.WinLoss.Ratio  = AvgWinLoss,
                Med.WinLoss.Ratio  = MedWinLoss,
                Max.Equity         = MaxEquity,
                Min.Equity         = MinEquity,
                End.Equity         = EndEquity)
        return(tmpret)
    }
    ret <- NULL
    tmpret <- apply(dailyPL,2,FUN=dailyFUN)
    for(row in 1:length(tmpret)){
        if(is.null(ret)) ret <- tmpret[[row]]
        else ret <- rbind(ret,tmpret[[row]])
        rownames(ret)[row]<-names(tmpret)[row]
    }
    #rownames(ret)<-colnames(dailyPL)
    ret <- round(ret,2)
    return(ret)
}

###############################################################################
# Blotter: Tools for transaction-oriented trading systems development
# for R (see http://r-project.org/)
# Copyright (c) 2008-2015 Peter Carl and Brian G. Peterson
#
# This library is distributed under the terms of the GNU Public License (GPL)
# for full details see the file COPYING
#
# $Id$
#
###############################################################################
```

R-Forge@R-project.org **ViewVC Help**

Powered by ViewVC 1.0.0

Thanks to: