The last thing we can do to find the effectiveness of the new strategy in comparision to the old one is to run a bunch of simulations and see how each strategy does on the same stock pair. After running this comparision on thousands of simulated pairs, no trend became very obvious. After creating a distribution of the difference in profit from the new strategy minus the profit from the old strategy, the distribution was always centered at zero. The issue was that outliers in the data set would consistently drag the mean either very high or very low so there was no discernable pattern to which strategy preforms better. Below is just an example mean of a distribution of 500 stock pairs.

```
## The mean of the distribution is: -66.4060449841699
```

Appendix of Code

```
downloadStockPairDF <- function(stock1, stock2, start = 2010, nyears = 1) {
  stock1df <- downloadPriceDF(stock1, start = start, nyears = nyears)
  stock2df <- downloadPriceDF(stock2, start = start, nyears = nyears)
  if (!identical(stock1df$date, stock2df$date)) {
    stop('Dates are wack')
    #rid <- which(stock1df$date != stock2df$date)
    #stock1df <- stock1df[-rid, ]
    #stock2df <- stock2df[-rid, ]
  }
  df <- data.frame(stock1 = stock1df$price,
                   stock2 = stock2df$price,
                   ratio = stock1df$price/stock2df$price)
  return(df)
}

plotStocks <- function(stockPairDF) {
  max <- max(max(stockPairDF$stock1), max(stockPairDF$stock2))
  plot(x = 1:length(stockPairDF$stock1), y = stockPairDF$stock1, type = 'l', col = 'red',
       ylim = c(0, max), xlab = '', ylab = 'Price')
  lines(x = 1:length(stockPairDF$stock1), y = stockPairDF$stock2, col = 'blue')
  legend('topleft', legend = c('stock1', 'stock2'), col = c('red', 'blue'), lty = 1, cex = .8)
}
plotRatio <- function(stockDF, k=1) {
  m <- mean(stockDF$ratio); s <- sd(stockDF$ratio)
  plot(x=1:length(stockDF$ratio), y=stockDF$ratio,
       type='l',
       xlab='', ylab=paste('Ratio (k=',k,')', sep = ''))
  abline(h=c(m-k*s,m,m+k*s), lty=c(2,1,2))
}
findPositions <- function(ratio, m, s, k = 1) {
  len <- length(ratio)
  toReturn <- list(); lind <- 1
  finger <- 1
  openquestionmark <- FALSE

  while (finger < len) {

    if (!openquestionmark) { #means we have to find the first place to open a position
      highones <- which(ratio[finger:len]>(m+s*k))
      lowones <- which(ratio[finger:len]<(m-s*k))
      if (length(highones)>0 | length(lowones)>0) {
```

```
       start <- min(highones[1], lowones[1], na.rm = TRUE) + finger - 1
       highorlow <- ifelse(identical(start, highones[1]+finger-1),
                           1, -1)
       finger <- start
       openquestionmark <- TRUE
      }
      else finger <- len
    }
    else { #we need to find where to close the position
      close <- ifelse(highorlow == 1,
                      which(ratio[finger:len]<m)[1],
                      which(ratio[finger:len]>m)[1])
      if (is.na(close)) { #means there is no good close; must close on last day
        close <- len
      }
      else close <- close+finger-1

      finger <- close
      toReturn[[lind]] <- c(start, close, highorlow)
      lind <- lind + 1
      openquestionmark <- FALSE
    }

  }#end of while loop
  return(toReturn)
}
addPositions <- function(ratio, positions) {
  for (p in positions) {
    points(x=p[1],y=ratio[p[1]],col='green',pch=19)
    points(x=p[2],y=ratio[p[2]],col='red',pch=19)
  }
}
positionProfit <- function(stocksDF, positions, net = TRUE) {
  if (length(positions) == 0) return(0)
  cashcash <- numeric(length(positions))
  i <- 1
  for (p in positions) {
    shares1 <- 1/stocksDF$stock1[p[1]]
    shares2 <- 1/stocksDF$stock2[p[1]]
    profit1 <- p[3]*-1 * shares1 * stocksDF$stock1[p[2]]
    profit2 <- p[3] * shares2 * stocksDF$stock2[p[2]]
    fees <- 0.003 * (1+1+abs(profit1)+abs(profit2))
    cashcash[i] <- profit1+profit2-fees
    i <- i+1
  }
  if (net) return(sum(cashcash))
  return(cashcash)
}
findOptimalK <- function(stocksDF, plot = FALSE) {
  ratio <- stocksDF$ratio; m <- mean(stocksDF$ratio); s <- sd(stocksDF$ratio)
  kmax <- max(abs(ratio - m))/s
  kvalues <- seq(0, kmax, length = 100)
  kprof <- sapply(kvalues, function(x) positionProfit(stocksDF = stocksDF,
```

```
                                                    findPositions(ratio, m, s, k=x)))
  ind <- which(kprof == max(kprof))[1]
  bestk <- kvalues[ind]

  if (plot) {
    plot(x=kvalues, y=kprof, type='p',
         xlab='k value', ylab='Profit')
    points(x=bestk, y=kprof[ind], pch=19, col='red')
  }

  return(bestk)
}
evaluatePairsTrading <- function(stocksDF, trainingFrac = 0.5, plot = FALSE) {
  cutoff <- ceiling(nrow(stocksDF)*trainingFrac)
  train <- stocksDF[1:cutoff,]
  test <- stocksDF[(cutoff+1):nrow(stocksDF),]
  k <- findOptimalK(train)
  m <- mean(train$ratio); s <- sd(train$ratio)
  pos <- findPositions(test$ratio, m, s, k=k)

  if (plot) {
    plot(x=1:nrow(stocksDF), y=stocksDF$ratio,
         type = 'l', xlab = '', ylab = paste('Ratio (k=',k,')',sep = ''))
    abline(h = c(m-s*k, m, m+s*k), lty=c(2,1,2))
    abline(v = cutoff)
    for (p in pos) {
      points(x=p[1]+cutoff,y=stocksDF$ratio[p[1]+cutoff],col='green',pch=19)
      points(x=p[2]+cutoff,y=stocksDF$ratio[p[2]+cutoff],col='red',pch=19)
    }
  }

  return(positionProfit(test, pos))
}

simulateStockPair <- function(n=1000, sigma1=1, sigma2=1, rho=1, psi=0, b1=0, b2=0, plot=FALSE) {
  stock1 <- numeric(n); stock2 <- numeric(n)
  x1 <- 2; x2 <- 2
  for (t in 1:n) {
    stock1[t] <- 40 + b1*t + x1; stock2[t] <- 35 + b2*t + x2
    x1 <- rho*x1 + (1-rho)*psi*x2 + rnorm(1,0,sigma1)
    x2 <- rho*x2 + (1-rho)*psi*x1 + rnorm(1,0,sigma2)
  }
  r <- stock1/stock2
  df <- data.frame(stock1 = stock1, stock2 = stock2, ratio = r)
  if (plot) plotStocks(df)
  return(df)
}
simulateDistribution <- function(nrep = 100, returnCorrelation = FALSE, ...) {
  dist <- numeric(nrep)
  if (returnCorrelation) {
    for (i in 1:nrep) {
      df <- simulateStockPair(...)
      dist[i] <- cor(df$stock1, df$stock2)
```

```
    }
    return(dist)
  }
  for (i in 1:nrep) {
    dist[i] <- evaluatePairsTrading(simulateStockPair(...))
  }
  return(dist)
}
findPositionsExtension <- function(ratio, k=1) {
  len <- length(ratio)
  toReturn <- list(); lind <- 1
  finger <- 2
  openquestionmark <- FALSE
  m <- numeric(len); s <- numeric(len)
  for (i in 1:len) {
    m[i] <- mean(ratio[1:i])
    s[i] <- sd(ratio[1:i])
  }

  while (finger < len) {

    if (!openquestionmark) { #means we have to find the first place to open a position
      if (ratio[finger] > m[finger]+s[finger]*k) {
        start <- finger
        finger <- finger + 1
        highorlow <- 1
        openquestionmark <- TRUE
      }
      else if (ratio[finger] < m[finger]-s[finger]*k) {
        start <- finger
        finger <- finger + 1
        highorlow <- -1
        openquestionmark <- TRUE
      }
      else {
        finger <- finger + 1
      }
    }

    else { #find a place to close position
      if (highorlow == 1) {
        while (openquestionmark) {
          if (ratio[finger]<m[finger]) {
            toReturn[[lind]] <- c(start, finger, 1)
            lind <- lind + 1
            openquestionmark <- FALSE
          }
          else if (finger == len) {
            toReturn[[lind]] <- c(start, finger, 1)
            openquestionmark <- FALSE
          }
          else {
            finger <- finger + 1
```

```
          }
        }
      }
      else {
        while(openquestionmark) {
          if (ratio[finger]>m[finger]) {
            toReturn[[lind]] <- c(start, finger, 1)
            lind <- lind + 1
            openquestionmark <- FALSE
          }
          else if (finger == len) {
            toReturn[[lind]] <- c(start, finger, -1)
            openquestionmark <- FALSE
          }
          else {
            finger <- finger + 1
          }
        }
      }
    }
  }
  return(toReturn)
}
findPositionsExtension2 <- function(ratio, m, s, k=1) {
  len <- length(ratio)
  if (len <= 750) {
    stop('Not enough data')
  }
  toReturn <- list(); lind <- 1
  finger <- 750
  openquestionmark <- FALSE

  while (finger < len) {

    if (!openquestionmark) { #means we have to find the first place to open a position
      if (ratio[finger] > m[finger]+s[finger]*k) {
        start <- finger
        finger <- finger + 1
        highorlow <- 1
        openquestionmark <- TRUE
      }
      else if (ratio[finger] < m[finger]-s[finger]*k) {
        start <- finger
        finger <- finger + 1
        highorlow <- -1
        openquestionmark <- TRUE
      }
      else {
        finger <- finger + 1
      }
    }

    else { #find a place to close position
```

```
      if (highorlow == 1) {
        while (openquestionmark) {
          if (ratio[finger]<m[finger]) {
            toReturn[[lind]] <- c(start, finger, 1)
            lind <- lind + 1
            openquestionmark <- FALSE
          }
          else if (finger == len) {
            toReturn[[lind]] <- c(start, finger, 1)
            openquestionmark <- FALSE
          }
          else {
            finger <- finger + 1
          }
        }
      }
      else {
        while(openquestionmark) {
          if (ratio[finger]>m[finger]) {
            toReturn[[lind]] <- c(start, finger, 1)
            lind <- lind + 1
            openquestionmark <- FALSE
          }
          else if (finger == len) {
            toReturn[[lind]] <- c(start, finger, -1)
            openquestionmark <- FALSE
          }
          else {
            finger <- finger + 1
          }
        }
      }
    }
  }
  return(toReturn)
}
plotRatioExtension2 <- function(stockDF, k=1) {
  plotRatioExtension(stockDF, k=k)
  abline(v=750)
}
findOptimalKExtension <- function(stocksDF, plot=FALSE) {
  ratio <- stocksDF$ratio
  len <- length(ratio)
  m <- numeric(len); s <- numeric(len)
  for (i in 1:len) {
    m[i] <- mean(ratio[1:i])
    s[i] <- sd(ratio[1:i])
  }
  kmax <- max(abs(ratio[750:len]-m[750:len])/s[750:len])
  kvalues <- seq(0, kmax, length.out = 100)
  kprof <- sapply(kvalues, function(x) positionProfit(stocksDF = stocksDF,
                                                      findPositionsExtension2(ratio, m, s, k=x)))
  ind <- which(kprof == max(kprof))[1]
```

```
  bestk <- kvalues[ind]

  if (plot) {
    plot(x=kvalues, y=kprof, type='p',
         xlab='k value', ylab='Profit')
    points(x=bestk, y=kprof[ind], pch=19, col='red')
  }

  toReturn <- c(bestk, kprof[ind])
  return(toReturn)
}
evaluatePairsTradingExtension <- function(stocksDF, trainingFrac = 0.5, plot = FALSE) {
  cutoff <- ceiling((nrow(stocksDF)-750)*trainingFrac)
  k <- findOptimalKExtension(stocksDF[1:(cutoff+750),])[1]

  ratio <- stocksDF$ratio; len <- length(ratio)
  m <- numeric(len); s <- numeric(len)
  for (i in 1:len) {
    m[i] <- mean(stocksDF$ratio[1:i])
    s[i] <- sd(stocksDF$ratio[1:i])
  }

  pos <- findPositionsExtensionHelper(ratio[(cutoff+751):len],
                                      m[(cutoff+751):len],
                                      s[(cutoff+751):len],
                                      k = k)

  if (plot) {
    plotRatioExtension2(stocksDF, k=k)
    for (p in pos) {
      points(x=p[1]+cutoff+750,y=stocksDF$ratio[p[1]+cutoff+750],col='green',pch=19)
      points(x=p[2]+cutoff+750,y=stocksDF$ratio[p[2]+cutoff+750],col='red',pch=19)
    }
    abline(v=cutoff+750, lty=2)
  }

  test <- stocksDF[(cutoff+751):len,]
  return(positionProfit(test, pos))
}
simulateDistributionComparison <- function(nrep = 100, returnCorrelation = FALSE, ...) {
  dist <- numeric(nrep)
  if (returnCorrelation) {
    for (i in 1:nrep) {
      df <- simulateStockPair(...)
      dist[i] <- cor(df$stock1, df$stock2)
    }
    return(dist)
  }
  for (i in 1:nrep) {
    df <- simulateStockPair(...)
    dist[i] <- evaluatePairsTradingExtension(df) - evaluatePairsTrading(df)
  }
  return(dist)
```

```
}
```