The Effects of Stock Trends on the Performance of Pairs Trading

Jack Norman

1 Introduction

The strategy known as pairs trading in the financial industry was created in the 1980s at Morgan Stanley. The main idea of pairs trading is to observe the history of two stocks that are fairly correlated. Given that they are correlated, the ratio of the value of one stock to the other should remain rather steady. With this knowledge, we use pairs trading by expecting the ratio to revert to the mean over time.

In our implementation of pairs trading, we define "opening the position" as the following: when a stock, A, starts to perform better than it usually does relative to the other stock, B, we sell one dollar of stock A since we expect the ratio to revert back to the mean. We define "closing the position" as the following: once the ratio reverts back to the mean we sell back the number of stocks we bought of B, and buy back the number of stocks we sold of A. The point at which we deem a stock as performing better than it usually does relative to the other stock is defined as when the ratio gets k standard deviations away from the mean. When analyzing the history of stocks, we can run our pairs trading algorithm with different values of k and then choose the k that maximizes profit. The idea is then to see how well our pairs trading algorithm performs with future data using the same k that maximized the profit in the history.

This paper first looks at a real data example of pairs trading and is then followed by a simulation study of how the characteristics (such as cross-correlation and slope) of the trend of a stock effect the performance of pairs trading. Finally, I discuss the results of the simulation study.

2 Real Data Example

The two companies to which I decided to apply the pairs trading algorithm were Gap and Urban Outfitters. Both clothing companies with similar fashion styles, I figured they are probably fairly correlated. After attaining the data for both companies, I used five years

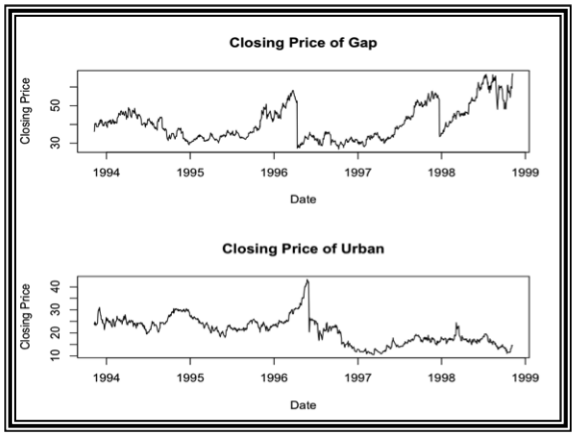

Figure 1: Closing prices of Gap and Urban Outfitters

of training data starting on November 9th, 1993. First I plotted the closing prices of both companies. In Figure 1, one can see the occasional days where there is a vertical line between two points. This is an indication of a split occurring. With the presence of splits in both companies, I used the adjusted closing price instead of the closing price in order to account for splits. The ratio of the adjusted closing prices of Gap to the adjusted closing prices of Urban Outfitters from November 9th, 1993 to November 6th, 1998 is seen in Figure 2. As one can see from Figure 2, Gap and Urban Outfitters don't appear to be very correlated. Ideally, the ratio plot in Figure 2 should have a slope of zero (indicating high correlation). In fact, their correlation coefficient is -0.1716457, which is very low. Despite the weak correlation, I decided to continue on with Gap and Urban Outfitters as my pair of stocks to see how the pairs trading algorithm worked.

Applying the pairs trading strategy, I found the profits for different k values (code of $find.all.ks()$ can be seen in Appendix L). The plot of the k values with their corresponding profits is seen Figure 3. As one can see in Figure 3, the majority of the k's we use would yield a negative profit. But the point of looking at all the k's is so that we can select the most profiting k to use in the future. As one can see from Figure 3, the k that maximizes profit using the training data (the first five years of stock data) is 1.06 (yielding a would-be profit of \$0.52). We can then plot the ratio plot again along with the mean, k value seen in Figure 4 (code of *ratio.plot*() seen in Appendix G). I then used the mean, standard deviation, and best k value from the training data and applied it to the testing data, which ranges from November 9th, 1998 to October 15th, 2013. Unfortunately I lose \$1.11 (code for $findGrandProfit()$ seen in Appendix H). The plot of the testing data with all the points of opening and closing is seen in Figure 5. As one can see from Figure 5, the ratio of Gap to Urban Outfitters is not steady at all. Not surprisingly, the mean and k

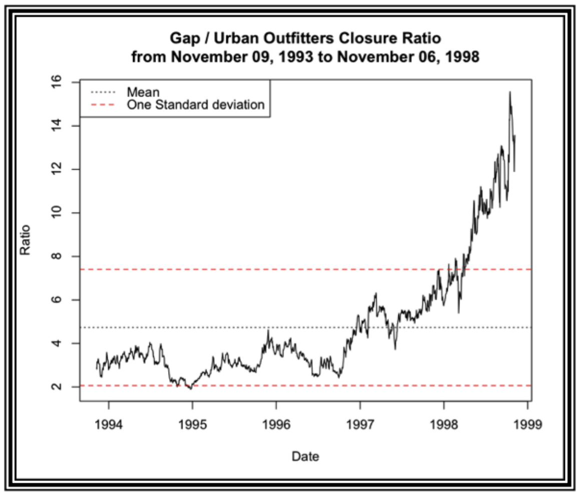

Figure 2: Ratio of adjusted closing price of Gap to Urban Outfitters

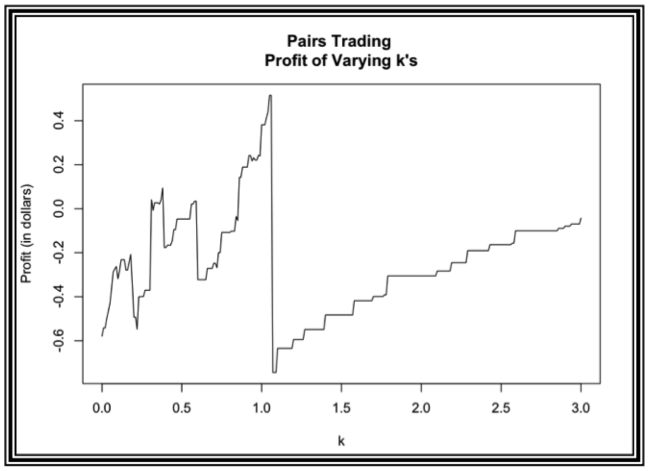

Figure 3: Profit using pairs trading with different values of k

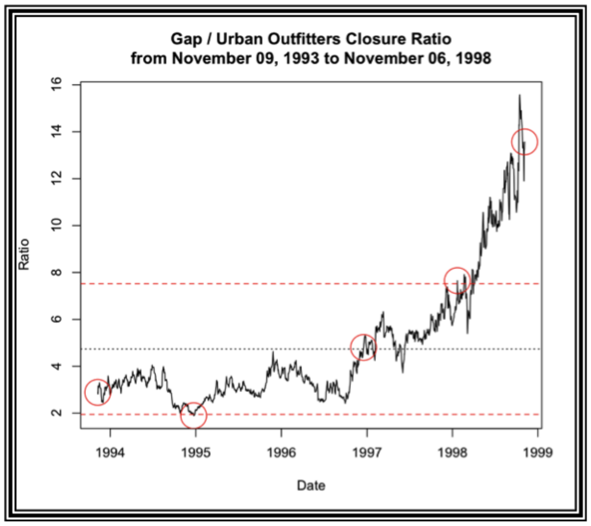

Figure 4: Ratio plot with circled first day, opening and closing days, and last day. Note that the first day isn't an opening day, but the last day is a closing day.

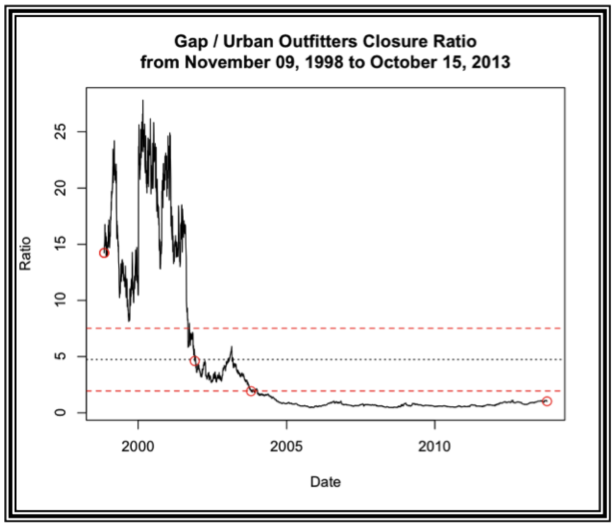

Figure 5: Ratio of adjusted closing price of Gap to Urban Outfitters on testing data with the first point, opening and closing points, and end point circled. Note that the first day is an opening day and the last day is a closing day.

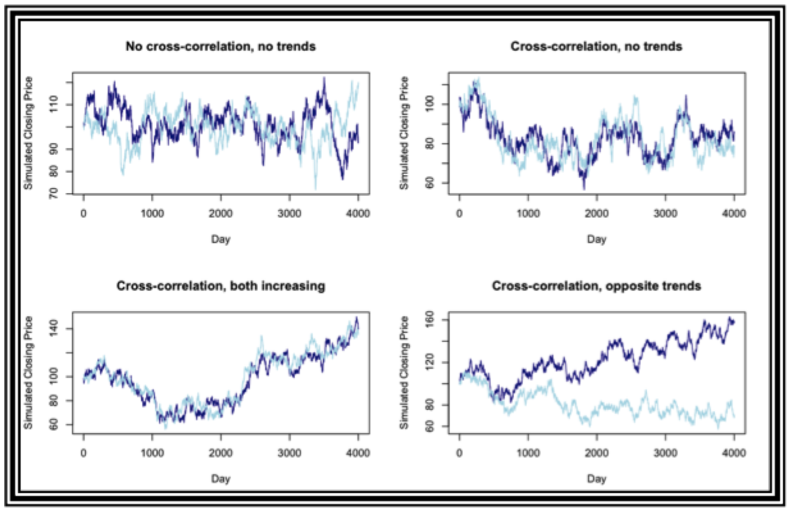

Figure 6: Four-thousand simulated days with different sets of cross-correlation and trends.

standard deviations used from the training data did not match well with our testing data. It's not much of a surprise that the pairs trading strategy didn't work well on the Gap and Urban Outfitters stocks since the two stocks had very low correlation.

3 Simulation Study

We use a vector autoregression model that simulates stock data over time. For stock 1 and stock 2, we define two sequences $X^{(1)}$ and $X^{(2)}$ with $X_1^{(i)} \sim N(0, \sigma_i^2)$ and subsequent values given by $X_t^{(1)} = \rho X_{t-1}^{(1)} + \psi(1-\rho)X_{t-1}^{(2)} + \epsilon_t^{(1)}$ and $X_t^{(2)} = \rho X_{t-1}^{(2)} + \psi(1-\rho)X_{t-1}^{(1)} + \epsilon_t^{(2)}$ where $\epsilon_t^{(i)} \sim N(0, \sigma_i^2)$ and are independent and identically distributed. In addition, ρ and ψ are values between zero and one. The parameter ρ controls how correlated the stocks are in time while ψ controls how correlated the stocks are with each other (cross-correlation). To incorporate linear trends, we take $Y_t^{(1)} = \beta_0^{(1)} + \beta_1^{(1)}t + X_t^{(1)}$ and $Y_t^{(2)} = \beta_0^{(2)} + \beta_1^{(2)}t + X_t^{(2)}$ where the Y sequences represent the closing prices of two stocks. In our model, we assign the $\sigma_i^2 = 1$ so that $\epsilon_t^{(i)} \sim N(0, 1)$. I simulated 4000 days worth of data under four cases (using my function, $stocksim()$ seen in Appendix I). Each case has a different set of values for cross-correlation, slope of stock 1, and slope of stock 2. The plots of stock 1 and stock 2 over these simulated 4000 days for each of the cases is seen in Figure 6. Referring to Figure 6, in each of the different cases, a dataset was simulated using a standard deviation of 1, a correlation with time of .99, and intercepts of 100. In the "No cross-correlation, no trends" case, the cross-correlation was assigned 0 as were the slopes of both stocks. In the "Cross-correlation, no trends" case, the cross-correlation was assigned 0.9 and the

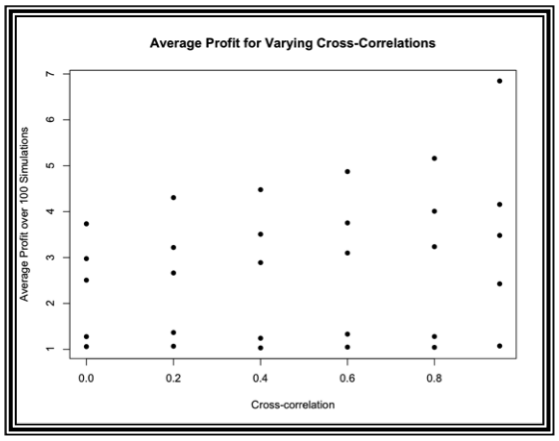

Figure 7: Average profit (using optimal k from training dataset) for varying cross-correlations

slopes of both stocks were assigned 0. In the "Cross-correlation, both increasing" case, the cross-correlation was assigned 0.9 and both stocks were assigned a slope of .01. Finally, in the "Cross-correlation, opposite trends", the cross-correlation was assigned 0.9, one stock was assigned a slope of .01 and the stock was assigned a slope of -.01.

After plotting these different cases, I wanted to study the effect of changing variables on the profit. To do this, I first prepared my own cases to study. My cases included all combinations of the following: psi taking the values 0, .2, .4, .6, .8, and .95, slope of stock 1 taking the values 0, .01, -.01, -.01, .01, and the slope of stock 2 taking the values 0, .02, -.01, .01, -.01. Thus, there were 30 cases I wanted to study. For each case, I created a dataset with the particular parameter setting (one of the combinations above), found the optimal k of the training data, and found how much profit that made in the testing data. I did this 100 times for each of the settings (implemented in *simstudy*() in Appendix J). The table of the output of average profit, standard deviation of profit, and other statistics from the simulations I ran is seen in Table 1.

In Figure 7 is a plot illustrating how the cross-correlation between the two stocks had an effect on average profit. Each point in Figure 7 represents the average profit of different settings, but the purpose of this plot is to look at the trend of the points. The trend, which is clearly positive, tells us that as two stocks become more correlated, they tend to make more money. When the correlation is .95, however, an interesting thing happens: most settings of profits actually lower except for when the slopes of both graphs are -.01 and -.01 (the highest point in Figure 7). After looking at the effect of cross-correlation on profit, I plotted the effect of the varying cases of slopes against average profit seen in Figure 8.

Table 1: Results of 100 simulations of varying values of ψ, β_1^1, and β_1^2

B	corr	slope1	slope2	mean	sd	min	max	elapsedtime
100	0	0	0	2.976301	0.5160591	0.9906467	4.350962	4.666848
100	0	0.01	0.01	2.50697	0.4323169	1.398647	3.800804	4.389252
100	0	-0.01	-0.01	3.735088	0.6838076	2.131059	5.999532	4.398036
100	0	0.01	-0.01	1.276537	0.441638	0.07168795	2.414939	2.259085
100	0	-0.01	0.01	1.059481	0.4409337	0.2656535	2.270233	2.224514
100	0.2	0	0	3.219613	0.48742	2.178484	4.445204	5.161477
100	0.2	0.01	0.01	2.664426	0.4322073	1.710552	3.735349	5.026825
100	0.2	-0.01	-0.01	4.305512	0.6279224	2.631661	6.004422	5.415757
100	0.2	0.01	-0.01	1.365797	0.4351385	0.5025382	2.397841	2.579295
100	0.2	-0.01	0.01	1.067568	0.3836596	0.04486343	2.290765	2.574652
100	0.4	0	0	3.507883	0.4866011	2.075237	4.629772	5.781294
100	0.4	0.01	0.01	2.889436	0.4082996	1.324533	3.6846	5.68353
100	0.4	-0.01	-0.01	4.47829	0.6229736	2.850058	5.927686	5.713468
100	0.4	0.01	-0.01	1.241835	0.4553537	0.388086	2.574454	2.483136
100	0.4	-0.01	0.01	1.031015	0.3534339	0.2642205	1.932192	2.466142
100	0.6	0	0	3.75492	0.440388	2.756527	4.863986	6.53332
100	0.6	0.01	0.01	3.099256	0.3993052	1.818748	4.268472	6.097654
100	0.6	-0.01	-0.01	4.872652	0.6869674	3.036949	6.715461	6.981334
100	0.6	0.01	-0.01	1.330883	0.4095156	0.3166101	2.407993	2.583079
100	0.6	-0.01	0.01	1.048537	0.3226805	0.2124881	2.135368	2.387204
100	0.8	0	0	4.008727	0.4729387	3.081151	5.385462	7.19109
100	0.8	0.01	0.01	3.236415	0.4424142	2.119851	4.530437	7.024043
100	0.8	-0.01	-0.01	5.159833	0.7859696	3.236721	7.029174	7.048649
100	0.8	0.01	-0.01	1.278957	0.4033242	0.4124675	2.307291	2.528968
100	0.8	-0.01	0.01	1.043393	0.3986164	0.2878738	2.162221	2.449541
100	0.95	0	0	4.157216	0.8218705	2.495772	6.959999	7.355255
100	0.95	0.01	0.01	3.48178	0.5519949	2.418627	5.273244	7.615873
100	0.95	-0.01	-0.01	6.84699	11.46868	3.213266	119.1134	7.113304
100	0.95	0.01	-0.01	2.426734	10.30502	-0.2060407	104.2915	2.662642
100	0.95	-0.01	0.01	1.07371	0.419498	0.1811204	2.736428	2.469591

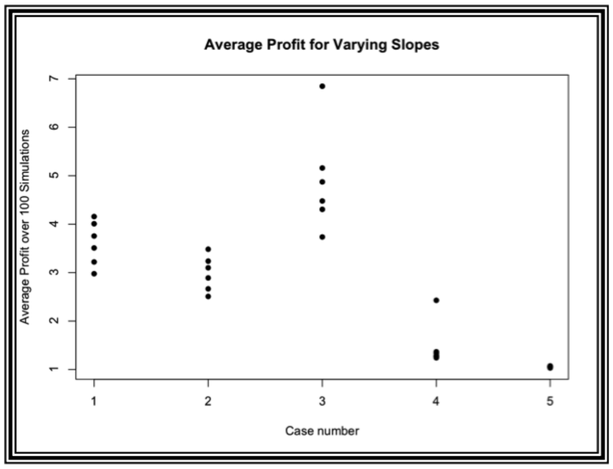

Figure 8: Average profit (using optimal k from training dataset) for varying slopes of stocks

Note that Case 1 means the slope of both stocks are 0, Case 2 means the slope of both stocks are .01, Case 3 means the slope of both stocks are -.01, Case 4 means the slope of stock 1 is .01 while the slope of stock 2 is -.01, and Case 5 means the slope of stock 1 is -.01 while the slope of stock 2 is .01. Interestingly, the slopes that yielded the best average profit were -.01 and -.01! The next best setting of slopes in terms of yielding high profit was when there was no slope followed by both stocks having a .01 slope. The two settings of slopes that clearly don?t create a lot of profit are when the slopes of stock 1 and stock 2 are opposite.

Throughout the previous two examinations of cross-correlation and slopes, I used an optimal k to find the profits. However, I would like to see the effect of varying k has on the profit. To do this I looked at six scenarios of cross-correlation and slope settings. For each of the scenarios, I looked at the profit of k for all k's between 0 and 5 with a .01 interval. Seen in Figure 9 is the plot of varying k. From Figure 9, one can see that the optimal k is usually between 0 and 1.5. This suggests that it is better to make more small trades then fewer large trades. However, this would change if we were to increase the fee of the transactions.

4 Discussion

After looking at many factors that effect of using pairs trading, it is even more clear why the stocks I used, Gap and Urban Outfitters, performed poorly. For one thing, Gap and Urban Outfitters arentt very correlated at all. From Figure 7, we saw that a substantial amount of correlation is beneficial to the profit. In addition, the ratio of Gap to Urban

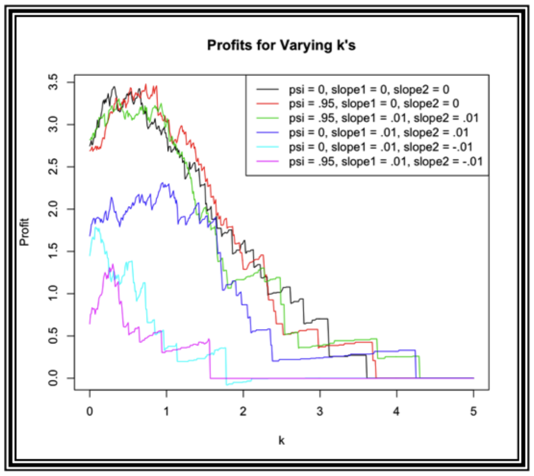

Figure 9: Profits for varying k's

Outfitters is clearly not very level. This is seen in Figure 2 and Figure 5, and indicates that the slopes of both stocks are clearly not the same. As we saw in Figure 8, the cases that contribute in a positive way to profit are when the slopes of the stocks are the same. Since Gap and Urban Outfitters have different slopes, it's not a surprise that the profit of the testing data yielded a loss of money.

Appendices

A Load Data

```
# libraries to load:
library(lattice)
library(RColorBrewer)

# Load data
setwd("~/Desktop/Fall_2013/STA_141/Homework_3_TradingSim/")
Stock1 <- read.table("gaps2.csv", sep = ",", header = TRUE) # gaps2 is
    from Yahoo
Stock2 <- read.table("urban2.csv", sep = ",", header = TRUE) # urban2
    is from Yahoo
Stock1 <- Stock1[!is.na(Stock1$Close), ] # make sure there are no NA's
    for the close column;
Stock2 <- Stock2[!is.na(Stock2$Close), ]
Stock1$Date <- as.Date(Stock1$Date, format = "%m/%d/%y") # format dates
     to class Date
```

```
Stock2$Date <- as.Date(Stock2$Date, format = "%m/%d/%y")
Stock1.Stock2 <- merge(Stock1, Stock2, by = "Date", suffixes = c(".
    Stock1", ".Stock2"), all = TRUE) # merge stocks
Stock1.Stock2 <- Stock1.Stock2[!is.na(Stock1.Stock2$Close.Stock1) & !is
    .na(Stock1.Stock2$Close.Stock2), ] # remove missing day's data
Stock1.Stock2$ratio <- Stock1.Stock2$Adj.Close.Stock1 / Stock1.Stock2$
    Adj.Close.Stock2 # calculate ratio
nums_training_year <- 5 # specify number of training years; using 5 in
    this hw assignment
training <- Stock1.Stock2[Stock1.Stock2$Date <= (min(Stock1.Stock2$Date
    ) + 365*nums_training_year), c("Date", "Close.Stock1", "Close.Stock2
    ", "Adj.Close.Stock1", "Adj.Close.Stock2", "ratio")]
testing <- Stock1.Stock2[Stock1.Stock2$Date > (min(Stock1.Stock2$Date)
    + 365*nums_training_year), c("Date", "Close.Stock1", "Close.Stock2",
     "Adj.Close.Stock1", "Adj.Close.Stock2", "ratio")]
```

B Open()

```
# Open()
#
# DateCurr: the date we want to Open()
# Dollar.Sell: the amount (in dollars) we want to sell of the stock we
    want to sell
# Dollar.Buy: the amount (in dollars) we want to buy of the stock we
    want to buy
# p: the handling fee to multiply by the abs value of money transacted
# status: the location (will be 'a' (if less than m - ks) or 'd' (if
    greater than m + ks))
#
# Function returns list: [<units sold>, <units bought>, <amount given
    to broker>]
#
Open <- function(dataset, DateCurr, Dollar.Sell, Dollar.Buy, p, status)
     {
  if (status == 'a') { # sell stock 2, buy stock 1
    units.sold <- Dollar.Sell/dataset[dataset$Date == DateCurr, c("
        Close.Stock2")]
    units.bought <- Dollar.Buy/dataset[dataset$Date == DateCurr, c("
        Close.Stock1")]
  }
  if (status == 'd') { # sell stock 1, buy stock 2
    units.sold <- Dollar.Sell/dataset[dataset$Date == DateCurr, c("
        Close.Stock1")]
    units.bought <- Dollar.Buy/dataset[dataset$Date == DateCurr, c("
        Close.Stock2")]
  }
  fee <- p * (abs(Dollar.Sell) + abs(Dollar.Buy))
  list(units.sold, units.bought, fee)
}
#
# end Open()
```

C Close()

```
# Close()
#
# DateCurr: the date we want to Close()
# sellback.shares: the amount (in stock units) that we want to sell (
    back) of the stock we want to sell (back)
# buyback.shares: the amount (in stock units) that we want to buy (back
    ) of the stock we want to buy (back)
# status: the location (will be 'b' (if m - ks < ratio < m) or 'c' (if
    m < ratio < m + ks))
# p: the handling fee to multiply by the abs value of money transacted
#
# Function returns list: [<dollars from sellback>, <dollars from
    buyback>]
#
Close <- function(dataset, DateCurr, sellback.shares, buyback.shares, p
    , status) {
  if (status == 'b' | status == 'a') { # buyback stock 1, sell back
      stock2
    dollars.sellback <- sellback.shares*dataset[dataset$Date ==
        DateCurr, c("Close.Stock2")]
    dollars.buyback <- buyback.shares*dataset[dataset$Date == DateCurr,
        c("Close.Stock1")]
  }
  if (status == 'c' | status == 'd') {
    dollars.sellback <- sellback.shares*dataset[dataset$Date ==
        DateCurr, c("Close.Stock1")]
    dollars.buyback <- buyback.shares*dataset[dataset$Date == DateCurr,
        c("Close.Stock2")]
  }
  fee <- p * (abs(dollars.sellback) + abs(dollars.buyback))
  list(dollars.sellback, dollars.buyback, fee)
}
#
# end Close()
```

D End()

```
# End()
```

```
#
# DateCurr: the date we want to Close()
# sellback.shares: the amount (in stock units) that we want to sell (
    back) of the stock we want to sell (back)
# buyback.shares: the amount (in stock units) that we want to buy (back
    ) of the stock we want to buy (back)
# status: the location (will be 'b' (if m - ks < ratio < m) or 'c' (if
    m < ratio < m + ks))
# p: the handling fee to multiply by the abs value of money transacted
#
# Function returns list: [<dollars from sellback>, <dollars from
    buyback>]
#
End <- function(dataset, DateCurr, sellback.shares, buyback.shares, p,
    status) {
  if (status == 'b' | status == 'd') { # buyback stock 1, sell back
      stock2
    dollars.sellback <- sellback.shares*dataset[dataset$Date ==
        DateCurr, c("Close.Stock2")]
    dollars.buyback <- buyback.shares*dataset[dataset$Date == DateCurr,
        c("Close.Stock1")]
  }
  if (status == 'c' | status == 'a') {
    dollars.sellback <- sellback.shares*dataset[dataset$Date ==
        DateCurr, c("Close.Stock1")]
    dollars.buyback <- buyback.shares*dataset[dataset$Date == DateCurr,
        c("Close.Stock2")]
  }
  fee <- p * (abs(dollars.sellback) + abs(dollars.buyback))
  list(dollars.sellback, dollars.buyback, fee)
}
#
# end End()
```

E FindNextOpen()

```
# findNextOpen()
#
# DateCurr: the date we want to "look forward" from
#
# Function returns a Date: the date of which has our next place to open
#
findNextOpen <- function(dataset, DateCurr) {
  # find all times the ratio is status a or d AND the date is AFTER or
      EQUAL the day we just opened on
  training.aord <- dataset[(dataset$status == 'a' | dataset$status == '
     d') & (dataset$Date >= DateCurr), ]
  # find next time the ratio is status a or d AFTER or EQUAL the day we
       just opened on
  next.aord <- training.aord[dataset$Date == min(dataset$Date), ]
  next.aord$Date
}
#
# end findNextOpen()
```

F FindNextClose()

```
# findNextClose()
#
# DateCurr: the date we want to "look forward" from
# StatusCurr: the current location (will be either 'a' or 'd')
#
# Function returns a Date: the date of which has our next place to
    close
#
findNextClose <- function(dataset, DateCurr, StatusCurr) {
  if (StatusCurr == 'a') {
    training.borc <- dataset[(dataset$status == 'c' | dataset$status ==
         'd') & (dataset$Date >= DateCurr), ]
    next.borc <- training.borc[dataset$Date == min(dataset$Date), ]
  }
  if (StatusCurr == 'd') {
    training.borc <- dataset[(dataset$status == 'a' | dataset$status ==
         'b') & (dataset$Date >= DateCurr), ]
    next.borc <- training.borc[dataset$Date == min(dataset$Date), ]
  }
  next.borc$Date
}
#
# end findNextClose()
```

G ratio.plot()

```
# ratio.plot()
#
# dataset: the dataset which contains the ratios of which we want to
    plot
# CurrDay: NULL for which.plot = 1, the day we want circled for which.
    plot = 2
# which.plot: 1 or 2; 1 is the original plot of ratios; 2 is the
    plotting of circles (which will be done at the first date, opening
    and closing dates, and last date
# avg: the mean; I pass in the mean instead of calculating it because
    it may not simply be the mean of the dataset$ratio; if the dataset
    is the testing dataset, we?ll want the mean of the training; only
    for which.plot = 1
# k: the number of sd?s away from the mean (to be drawn as a line; only
     for which.plot = 1)
# stock1, stock2: the names of stock1, stock2, only used for which.plot
     = 1
#
ratio.plot <- function(dataset, CurrDay, which.plot, avg, k, stock1,
    stock2) {
  if (which.plot == 1) { # if we haven't plotted yet, we just want to
      print the ratio, mean, and k's
    plot.new()
    # mycolors <- brewer.pal(4, "Reds")
    plot(dataset$Date, dataset$ratio, type = 'l', xlab = 'Date', ylab =
         'Ratio', main = paste(stock1, '/', stock2, 'Closure Ratio\nfrom
        ', format(min(dataset$Date), "%B %d, %Y"), 'to', format(max(
        dataset$Date), "%B %d, %Y")))
    abline(avg, 0, lty = 3, col = 1)
    abline(avg - k, 0, lty = 2, col = 2) # +k
    abline(avg + k, 0, lty = 2, col = 2) # -k
  }
  if (which.plot == 2) { # if we have already plotted and now we want
      to add circles
    symbols(dataset$Date[dataset$Date == CurrDay], dataset$ratio[
        dataset$Date == CurrDay], circles = c(55), inches = FALSE, add =
         TRUE, fg = 'red')
  }
}
#
# end ratio.plot()
```

H findGrandProfit()

```
# findGrandProfit()
#
# The flow of this function: find next opening, open, find next closing
    , close. Repeat until reaching the end
# dataset: the training data.frame which should include "Date", "Close.
    Stock1", "Close.Stock2", ?Adj.Close.Stock1?, ?Adj.Close.Stock2?, "
    ratio"
# ks: number of standard deviations; used for determining where to Open
    () and where to Close()
```

```
# plot.bool: default to FALSE; if set to true, a time series plot will
    be plotted of the ratio over time with a line for the mean, mean -
    ks, mean + ks, and circles at the first date, Close()ing and Open()
    ing dates, and last date
# p: the transaction handling fee; each time we buy or sell, we must
    pay a percentage of the amount of money transacted; we?ll use .001
    in our homework assignment
# stock1.name, stock2.name: the name of the Stock 1,2 company
# does.stock1.split, does.stock2.split: the user needs to notify this
    function as to whether or not the stocks split. If they do, set both
     of these to TRUE (they default to FALSE).
# training: if training is TRUE (default), mean and sd will be the mean
     and sd of the dataset; if training is FALSE (i.e., we?re using
    testing data), the mean and sd need to be passed in
# m: NULL if training is TRUE, mean of training$ratio if training is
    FALSE
# sd: NULL if training is TRUE, sd of training$ratio if training is
    FALSE
#
# Function returns a number representing the grand profit in dollars
#
findGrandProfit <- function(dataset, ks, plot.bool = FALSE, p, stock1.
    name, stock2.name, does.stock1.split = FALSE, does.stock2.split =
    FALSE, training = TRUE, m = NULL, sd = NULL, ...) {

  # Deal with splits by using adj.closing instead of closing
  if(does.stock1.split == TRUE) { dataset$Close.Stock1 <- dataset$Adj.
     Close.Stock1 }
  if(does.stock2.split == TRUE) { dataset$Close.Stock2 <- dataset$Adj.
     Close.Stock2 }

  # Calculate ratio
  dataset$ratio <- dataset$Close.Stock1 / dataset$Close.Stock2

  # Cheat way of dealing with ks = 0 (for bad user input)
  if (ks <= 0) { ks <- 0.0000001 }

  # Prepare data (is it training or testing?)
  if (training == FALSE) {    # if we are dealing with testing data
    # m <- m              # we want to use the training mean (passed in)
    ks <- ks*sd           # we want to use the training sd (passed in)
  } else {              # if we are dealing with training data
    m <- mean(dataset$ratio)  # we want to use the mean of training
        dataset
    ks <- ks*sd(dataset$ratio)  # we want to use the sd of trianing
        dataset
  } # end if else

  # find status of each point
    # a = (-Inf, m - ks]
    # b = (m - ks, m]
    # c = (m, m + ks]
    # d = (m + ks, Inf]
  dataset$status <- cut(dataset$ratio, breaks = c(-Inf, m - ks, m, m +
     ks, Inf), labels = c("a", "b", "c", "d"))

  # Plot ratio with mean and k
  if (plot.bool == TRUE) { ratio.plot(dataset, NULL, 1, m, ks, stock1.
     name, stock2.name) }

  # Find first day of dataset and add a circle to that point
  nextdate <- min(dataset$Date)
  if (plot.bool == TRUE) { ratio.plot(dataset, nextdate, 2, m, ks, NULL
     , NULL) }

  # initialize grand profit to zero
  grand.profit <- 0

  # Run the following while loop until the function returns
# (which will occur after we get to the last date)
  while (TRUE) {
    nextdate <- findNextOpen(dataset, nextdate)

    # If nextdate is NA, i.e. we've reached the last date of our
       dataset
    if (is.na(nextdate)) {
      nextdate <- max(dataset$Date)

      # Add a circle to the last point
      if (plot.bool == TRUE) { ratio.plot(dataset, nextdate, 2, m, ks,
         NULL, NULL) }

      # Isolate the observation of the last point
      ObsCurr <- dataset[dataset$Date == nextdate, ]

      # If the last date has status 'a' or 'd' (specified above),
      # then we need to close before we return; If the last point is 'b
         ' or 'c',
      # then we have already previously closed
      if (as.character(ObsCurr$status) == 'a' | as.character(ObsCurr$
         status) == 'd') {
```

```
    # End() will return a list <dollars.sellback, dollars.buyback,
       dollars.fee>
    dollars.list <- End(dataset, max(dataset$Date), units.bought,
       units.sold, p, as.character(ObsCurr$status))
    dollars.sellback <- unlist(dollars.list[1]) # dollars from
       sellback
    dollars.buyback <- unlist(dollars.list[2]) # dollars from
       buyback
    dollars.fee <- unlist(dollars.list[3])

    # Calculate profit for the current open / closing pair
    local.profit <- (1 - dollars.buyback) + (dollars.sellback - 1)
       - dollars.fee

    # Calculate profit for the overall profit thus far
    grand.profit <- grand.profit + local.profit # should be end of
       function
  } # end if

  # We have dealt with the last point so return grand.profit
  return(grand.profit)
} # end if -- i.e. if nextdate isn't NA

# Add circle to the plot (circling the spot where we're opening)
if (plot.bool == TRUE) { ratio.plot(dataset, nextdate, 2, m, ks,
   NULL, NULL) }

# OPEN process {

  # Isolate the observation of the current point
  ObsCurr <- dataset[dataset$Date == nextdate, ]

  # Open() will return a list <units.sold, units.bought, units.fee>
  units.list <- Open(dataset, nextdate, 1, 1, p, as.character(
     ObsCurr$status))
  units.sold <- unlist(units.list[1])
  units.bought <- unlist(units.list[2])
  units.fee <- unlist(units.list[3]) # dollars spent on fee

  # No change is made to the grand profit other than the
     transaction fee from opening
  grand.profit <- grand.profit - units.fee

# } end OPEN process
```

```
nextdate <- findNextClose(dataset, nextdate, as.character(ObsCurr$
   status))

# If nextdate is NA, i.e. we've reached the last date of our
   dataset
if (is.na(nextdate)) {
  nextdate <- max(dataset$Date)

  # Add circle to plot at last point
  if (plot.bool == TRUE) { ratio.plot(dataset, nextdate, 2, m, ks,
     NULL, NULL) }

  # Isolate the observation of the current point
  ObsCurr <- dataset[dataset$Date == nextdate, ]

  # If the last date has status 'a' or 'd' (specified above),
# then we need to close before we return; If the last point is 'b'
   or 'c',
  # then we have already previously closed
  if (as.character(ObsCurr$status) == 'a' | as.character(ObsCurr$
     status) == 'd') {

    # End() will return a list <dollars.sellback, dollars.buyback,
       dollars.fee>
    dollars.list <- End(dataset, max(dataset$Date), units.bought,
       units.sold, p, as.character(ObsCurr$status))
    dollars.sellback <- unlist(dollars.list[1]) # dollars from
       sellback
    dollars.buyback <- unlist(dollars.list[2]) # dollars from
       buyback
    dollars.fee <- unlist(dollars.list[3])

    # Calculate profit for the current open / closing pair
    local.profit <- (1 - dollars.buyback) + (dollars.sellback - 1)
       - dollars.fee

    # Calculate profit for the overall profit thus far
    grand.profit <- grand.profit + local.profit # should be end of
       function
  } # end if

  # We have dealt with the last point so return grand.profit
  return(grand.profit)
} # end if -- i.e. if nextdate isn't NA

# Add circle to the plot (circling the spot where we're opening)
```

```
    if (plot.bool == TRUE) { ratio.plot(dataset, nextdate, 2, m, ks,
       NULL, NULL) }

    # CLOSE process {

      # Isolate the observation of the current point
      ObsCurr <- dataset[dataset$Date == nextdate, ]

      # Close() will return a list <dollars.sellback, dollars.buyback,
         dollars.fee>
      dollars.list <- Close(dataset, nextdate, units.bought, units.sold
         , p, as.character(ObsCurr$status))
      dollars.sellback <- unlist(dollars.list[1])
      dollars.buyback <- unlist(dollars.list[2])
      dollars.fee <- unlist(dollars.list[3])

      # Calculate profit for the current open / closing pair
      local.profit <- (1 - dollars.buyback) + (dollars.sellback - 1) -
         dollars.fee
    # }

    # Calculate profit for the overall profit thus far
    grand.profit <- grand.profit + local.profit # should be end of
       function
  }
}
#
# end findGrandProfit()
```

I stocksim()

```
# stocksim()
#
# num_days: number of days to simulate; default is 4000 because that is
    our hw assignment
# rho: controls how correlated the sequences (stocks) are in time
# psi: controls how correlated the sequences (stocks )are with each
   other
# sd1, sd2: standard deviation of stock 1,2
# beta1_0, beta2_0: y-intercept of stock 1,2, respectively
# beta1_1, beta2_1: slope of stock 1,2, respectively
#
# function returns a dataframe that contains num_days observations of:
# stock1, stock2,
# where stock1 represents the closing price of stock1 on day t, and
   stock2 represents the closing price of stock2 on day t
```

```
#
stocksim <- function(num_days = 4000, rho = .99, psi = 0, sd1 = 1, sd2
    = 1, beta1_0 = 100, beta2_0 = 100, beta1_1 = 0, beta2_1 = 0) {
  # Calculate error terms
  error1_t <- rnorm(num_days, 0, sd1)
  error2_t <- rnorm(num_days, 0, sd2)

  # store the previous days' values here
  stock1 <- stock2 <- list()

  # generate initial point and immediately prepare it to be the
      previous point
  stock1[1] <- X1_t <- X1_1 <- rnorm(1, 0, sd1)
  stock2[1] <- X2_t <- X2_1 <- rnorm(1, 0, sd2)

  # Generate data for the rest of the days
  for (i in 2:num_days) { # We've already calculate the first day's
      price

    # note that X1_t and X2_t on the RHS represent the previous day
    stock1[i] <- X1_t <- rho*X1_t + psi*(1 - rho)*X2_t + error1_t[i]
    stock2[i] <- X2_t <- rho*X2_t + psi*(1 - rho)*X1_t + error2_t[i]

  } # end for

  stocks <- as.data.frame(cbind(t = 1:num_days, stock1 = unlist(stock1)
      , stock2 = unlist(stock2)))
  Y1_t <- beta1_0 + beta1_1*stocks$t + stocks$stock1
  Y2_t <- beta2_0 + beta2_1*stocks$t + stocks$stock2
  return(as.data.frame(cbind(Close.Stock1 = Y1_t, Close.Stock2 = Y2_t))
      )
}
#
# end stocksim()
```

J simstudy()

```
# simstudy()
#
# B: the number of datasets we'd like to sim
# corr: controls how correlated the sequences (stocks) are with each
   other
# slope1: slope of stock1
# slope2: slope of stock2
# ... : any parameters to pass on to find.best.k()
#
```

```
# function returns the profit for the given parameters (using k which
   maximised profit in training data)
#
simstudy <- function(B, corr, slope1, slope2, ...) {
  starttime <- Sys.time()
  situation <- list()

  for (i in 1:B) {

    datasetCurr <- stocksim(psi = corr, beta1_1 = slope1, beta2_1 =
       slope2)
    datasetCurr <- cbind(Date = 1:4000, datasetCurr)
    datasetCurr$ratio <- datasetCurr$Close.Stock1 / datasetCurr$Close.
       Stock2

    # separate dataset in to training and testing
    training <- datasetCurr[1:2000, ]
    testing  <- datasetCurr[2001:4000, ]

    # find the best k to 2 decimal places
    best.k <- find.best.k(dataset = training, origin = .5,  ending =
       2.5, interval = .5, k.best = NULL, precision = 2)

    # calculate profit using the best.k
    # in hind sight, I would've preferred to calculate maxprofit
       without the profit yielded from the training dataset since we
       don't actually get the profit from the training data. I figured
       that the dollars of profits may be a bit higher because of this,
        but that the trends should stay the same. So since I'm using
       simstudy() to study the trends of varying variables, the fact
       that I'm using the profit from the training data shouldn't
       effect the trend.
    maxprofit <- unlist(best.k[2]) + findGrandProfit(dataset = testing,
        ks = unlist(best.k[1]), plot.bool = FALSE, p = .001, stock1.
       name = NULL, stock2.name = NULL, training = FALSE, m = mean(
       datasetCurr$ratio), sd = sd(datasetCurr$ratio))

    # store this iteration of the simulation (so we can find average,
       sd, etc. later)
    situation[i] <- maxprofit

  } # end for

  # After all simulations {

    endtime <- Sys.time()
    elapsedtime <- endtime - starttime
    # return B, corr, slope1, slope2, mean, sd, min, max, starttime,
       endtime, elapsedtime
    d <- unlist(situation)
    situation.data <- as.data.frame(cbind(B = B, corr = corr, slope1 =
       slope1, slope2 = slope2, mean = mean(d), sd = sd(d), min = min(d
       ), max = max(d), elapsedtime = elapsedtime))

  # }
}
#
# end simstudy()
```

K find.best.k()

```
# find.best.k()
#
# dataset: dataset on which to find best k (will probably be the
   training dataset)
# origin: the smallest value of k to test
# ending: the largest value of k to test
# interval: the interval size between origin and ending to test (note:
   the number of k's being tested for iteration of find.best.k will be
   (ending - origin) / interval)
# k.best: initially NULL; the current list of the (k which maximises
   the profit) and the (max profit).
# precision: number of decimal places you want k to be; if there are
   multiple k's that yield the best profit, find.best.k() will return
   the median of the k's; precision doesn't exactly work they way i'd
   like it to since it depends on your original interval.
# does.stock1.split: boolean indicating whether stock1 splits (if TRUE,
    that means the stock1 splits at some point)
# does.stock2.split: boolean indicating whether stock2 splits (if TRUE,
    that means the stock2 splits at some point)
#
# function returns the k that maximises the profit of the dataset to
   the number of decimal places indicated by precision and the profit.
   Note as precision increases, time to run increases; takes about a
   tenth of the time as using min(find.all.ks)
#
find.best.k <- function(dataset, origin, ending, interval, k.best,
   precision, does.stock1.split = FALSE, does.stock2.split = FALSE,
   ...) {
  if (precision != 0) { # base case; i.e., we want our k to be more
     precise than it currently is
    # initialize k to the values we want to explore
    k <- seq(origin, ending, interval)

    # create list of k values and their corresponding profits
    k.profit <- sapply(k, FUN = function(i) findGrandProfit(dataset, ks
        = i, plot.bool = FALSE, p = .001, stock1.name = NULL, stock2.
       name = NULL, does.stock1.split = does.stock1.split, does.stock2.
       split = does.stock2.split))

    # isolate the max profit and it's corresponding k
    k.best <- list(k[k.profit == max(k.profit)], max(k.profit))

    # recursive call to find.best.k using a narrower search for k
    find.best.k(dataset = dataset, unlist(k.best)[1] - interval, unlist
       (k.best)[1] + interval, interval / 5, k.best, precision - 1,
       does.stock1.split = does.stock1.split, does.stock2.split = does.
       stock2.split) # call recursively

  } else { # i.e., we don't want a k any more precise than it already
     is; exit recursion and return k.best

    if (length(k.best) > 1) { # if there are multiple best k's (that
       yield the best profit), choose the middle (roughly)
      k.best <- list(unlist(k.best[1])[ceiling(length(unlist(k.best[1])
         )/2)], unlist(k.best[2]))

    } # end if

    return(k.best)
  } # end if else
}
#
# end find.best.k()
```

L find.all.ks()

```
# find.all.ks()
#
# dataset: dataset on which to find best k (will probably be the
   training dataset)
# origin: the smallest value of k to test
# ending: the largest value of k to test
# interval: the interval size between origin and ending to test (note:
   the number of k's being tested for iteration of find.best.k will be
   (ending - origin) / interval)
```

```
# does.stock1.split: boolean indicating whether stock1 splits (if TRUE,
     that means the stock1 splits at some point)
# does.stock2.split: boolean indicating whether stock2 splits (if TRUE,
     that means the stock2 splits at some point)
#
# function returns all ks and their profit values for the purpose of
    plotting
#
find.all.ks <- function(dataset, origin, ending, interval, does.stock1.
    split = FALSE, does.stock2.split = FALSE, ...) {
  #browser()

  # initialize k to the values we want to explore
  k <- seq(origin, ending, interval)

  # create list of k values and their corresponding profits
  k.profit <- sapply(k, FUN = function(i) findGrandProfit(dataset, ks =
       i, plot.bool = FALSE, p = .001, stock1.name = NULL, stock2.name =
       NULL, does.stock1.split = does.stock1.split, does.stock2.split =
      does.stock2.split))

  # return all ks and their profit values
  return(as.data.frame(cbind(unlist(k), unlist(k.profit))))
}
#
# end find.all.ks()
```

M Code and results of 100 Simulations of varying values of ψ, β_1^1, and β_1^2

```
# Combinations, alter correlation and slopes
# note elapsedtime is in minutes
beg <- Sys.time()
B   = rep(100, 30)
corr  = rep(c(0, .2, .4, .6, .8, .95), each = 5)
slope1  = rep(c(0, .01, -.01, .01, -.01), 6)
slope2  = rep(c(0, .01, -.01, -.01, .01), 6)
dataa <- mapply(simstudy, B, corr, slope1, slope2)
end <- Sys.time()
total = end - beg
# yielded:
# > t(dataa)
# see Table 1.
# Plot varying different parameters
dataat <- t(dataa)
```

```
dataat.frame <- as.data.frame(dataat)
dataat.frame$case <- rep(c(1, 2, 3, 4, 5), 6)
# Vary correlation
with(dataat.frame, plot(corr, mean, pch = 16, main = 'Average Profit
    for Varying Cross-Correlations', xlab = 'Cross-correlation', ylab =
    'Average Profit over 100 Simulations'))
# Vary slopes
with(dataat.frame, plot(case, mean, pch = 16, main = 'Average Profit
    for Varying Slopes', xlab = "Case number", ylab = 'Average Profit
    over 100 Simulations'))
```

N Code for varying k while keeping other characteristics constant

```
# psi = c(0, .95, .95, 0, 0, .95)
# beta1_1 = c(0, 0, .01, .01, .01, .01)
# beta2_1 = c(0, 0, .01, .01, -.01, -.01)
# varyk <- mapply(stocksim, psi = psi, beta1_1 = beta1_1, beta2_1 =
    beta2_1)
# varykt <- t(varyk)
#
# Vary k but keep others constant
# First scenario
varyk <- stocksim() # use default parameters
varyk <- cbind(Date = 1:4000, varyk)
varyk$ratio <- varyk$Close.Stock1 / varyk$Close.Stock2
allks <- find.all.ks(varyk, 0, 5, .01)

# Second scenario
varyk2 <- stocksim(psi = .95)
varyk2 <- cbind(Date = 1:4000, varyk2)
varyk2$ratio <- varyk2$Close.Stock1 / varyk2$Close.Stock2
allks2 <- find.all.ks(varyk2, 0, 5, .01)

# Third scenario
varyk3 <- stocksim(psi = .95, beta1_1 = .01, beta2_1 = .01)
varyk3 <- cbind(Date = 1:4000, varyk3)
varyk3$ratio <- varyk3$Close.Stock1 / varyk3$Close.Stock2
allks3 <- find.all.ks(varyk3, 0, 5, .01)

# Fourth scenario
varyk4 <- stocksim(beta1_1 = .01, beta2_1 = .01)
varyk4 <- cbind(Date = 1:4000, varyk4)
varyk4$ratio <- varyk4$Close.Stock1 / varyk4$Close.Stock2
allks4 <- find.all.ks(varyk4, 0, 5, .01)
```

```
# Fifth scenario
varyk5 <- stocksim(beta1_1 = .01, beta2_1 = -.01)
varyk5 <- cbind(Date = 1:4000, varyk5)
varyk5$ratio <- varyk5$Close.Stock1 / varyk5$Close.Stock2
allks5 <- find.all.ks(varyk5, 0, 5, .01)

# Sixth scenario
varyk6 <- stocksim(psi = .95, beta1_1 = .01, beta2_1 = -.01)
varyk6 <- cbind(Date = 1:4000, varyk6)
varyk6$ratio <- varyk6$Close.Stock1 / varyk6$Close.Stock2
allks6 <- find.all.ks(varyk6, 0, 5, .01)

# Plot scenarios
plot(allks$V1, allks$V2, type = 'l', xlab = 'k', ylab = 'Profit', main
   = "Profits for Varying k's", col = 1)
lines(allks2, col = 2)
lines(allks3, col = 3)
lines(allks4, col = 4)
lines(allks5, col = 5)
lines(allks6, col = 6)
legend("topright", c( "psi = 0, slope1 = 0, slope2 = 0",
              "psi = .95, slope1 = 0, slope2 = 0",
              "psi = .95, slope1 = .01, slope2 = .01",
              "psi = 0, slope1 = .01, slope2 = .01",
              "psi = 0, slope1 = .01, slope2 = -.01",
              "psi = .95, slope1 = .01, slope2 = -.01")
                  , lty = 1, col = (1:6))
```