

Statistical Programming Language
SS 17

Pairs Trading Strategy

- This is a Sample Subtitle -

Lung Chun Ting, Tim Schwettnann, Maria Theilemann, Cosima Klenner
August 18, 2017

Abstract

Pairs trading is a popular trading strategy and one of the market neutral investment strategies in the stock market. It tries to take advantage of market inefficiencies in order to obtain profit. The idea is simple: find two stocks that move together which means, find two stocks that have a correlation near to one and take long or short positions when they diverge abnormally, hoping that the prices will converge again in the future. The main objective of this paper is to show the replication of the proposed Pairs trading strategy in the paper How some bankers made a million by trading just two securities? by Kalle Rinne and Matti Suominen, with R.

Contents

1	Introduction	1
1.1	Evolution of the Strategy	1
1.2	Idea of Pairs Trading	1
1.3	Introducing the paper How some bankers made a million by trading just two securities?	2
2	Correlation	4
2.1	Data Extraction	4
2.2	Finding Correlated Stocks	8
3	Execution	10
3.1	Regression	11
3.2	Enter Strategy	13
3.3	Exit Strategy	14
4	Performance Analysis	15
5	Risk Management	16
5.1	Value at Risk	16
5.2	Expected Shortfall	16
6	Challenges	17
7	Conclusion	18
	References	19
8	Appendix	1

1 Introduction

The world of stocks and securities is filled with uncertainties and risks. Nonetheless, many investors see potential to make profits in this stock market with gathered information and strategies.

Pairs trading is a well-known and popular statistical arbitrage strategy and is loosely based on the Law of One Price. It is an old portfolio management technique based on a classic hedge: an investor selects two correlated stocks, buying the one s/he expects to perform best and selling (short) the one s/he expects to underperform.

A pair is simply defined as two stocks that tend to move together. The strategy consists in trading the spread, which basically means: a long position in one of the stocks versus a short position in the other, when a dislocation (= an abnormal diversion) between the two stock prices is observed.

In the setup of such a strategy, there are two distinct parts: the selection part (which pairs we should choose) and the implementation of the trading strategy (when and what sizes we should trade).

1.1 Evolution of the Strategy

Pairs strategy has made its reputation in the early 1980s. The methodology was designed by a team of scientists from many different areas: mathematics, computer sciences and also physics. Neither of them had any background in finance.

They were brought together by the Wall Street quantitative analyst Nunzio Tartaglia. Their goal was to develop statistical rules to find ways to perform arbitrage trades, and therewith, taking the skill and the talent out of trading (Gatev et al, 2006). In other words, the main objective of the team was to use statistical methods to develop computer based trading platforms, in which human subjectivity didn't influence the process of decision-making of either buying or selling particular stocks.

The developed systems were quite successful for a period of time. After a while, the performance was not consistent anymore and after a few periods of bad performance, the team was finally diverged. If you would like to read more about the details and the origins of pairs trading, you should read the paper from Gatev et al (2006).

1.2 Idea of Pairs Trading

Basically, the basic idea of pairs trading is to take advantage of market inefficiencies. Pairs trading works by taking the arbitrage opportunity of temporary anomalies between related stocks, which are stabilized (=have an equilibrium) in the long-run.

The first step is to identify two stocks that move together and trade them every time when a temporary anomaly occurs, meaning that one stock will be overvalued relative to the other stock. We can then invest in a pair consisting of a two-stock portfolio, where the overvalued stock will be sold (short position) and the undervalued stock will be bought (long position). It is also possible that just one stock is either over or undervalued. Nevertheless the strategy remains the same.

The trade is closed out by taking the opposite position of these stocks after the stocks have settled back into their long-run relationship. The profit is captured from this short-term discrepancies in the two stock prices. Making profit with pairs trading does not depend on the movement of the market. Therefore, pairs trading is a market-neutral investment strategy.

1.3 Introducing the paper How some bankers made a million by trading just two securities?

Rinnes and Suominens paper uses the framework of Jylh et al (2014) as a foundation for their proposed pairs trading strategy. The framework of Jylh et al (2014) examines a contrarian trading strategy where long-short portfolios of stocks are formed based on the stocks 5-day expected returns (from now on R5t). They extend the framework to pairs trading and show the attractivity also in the context of pairs trading of returns to a contrarian long-short liquidity providing trading strategy.

In the following, we will shortly state their framework and methodology: Rinne and Suominen started by examining the profitability of a short-term contrarian pairs trading strategy for which they used a large sample of US stocks. Therefore, they selected pairs of stocks based on the correlations of the stocks weekly returns during the previous year.

Using Jylh et als methodology, the authors then estimated the R5ts and enter into a long position in a pair if its expected returns are high enough. Contrary, entering into a short position in a pair if the expected returns are low enough. Positions are closed when the R5t to the pair have changed sign or enough time has passed from the opening of the trade. This simple pair trading strategy generates high returns in Rinnes and Suominens large sample of US stocks.

The paper considers two trading strategies: 1) a simple pairs trading strategy with positions getting closed after five days ; 2) a pairs trading strategy where positions are closed during a seven day window starting after three days, as soon as the R5ts from maintaining the position become negative.

The papers results showed that both strategies generate high returns, with the latter strategy generating greater returns than the first one.

Our paper is organized as follows: Section I is the Introduction, in which we will briefly explain the idea of pairs trading and the evolution of the strategy.

In Section II, we discuss the data extraction and the process of finding correlated stocks. Finding pairs that are highly correlated over time is the key to the success of a pairs trading strategy.

In Section III, we describe the execution of the pairs trading strategy, containing the regression part, two enter strategies as well as two exit strategies.

Section IV contains the performance analysis where we calculated the sharpe ratio of the pairs trading strategy. In the end, we will compare the sharpe ratio of our strategy to a well-known stocks sharpe ratio.

After that, we focus on risk management of the strategy, especially on Value at Risk.

In Section VI, we will outline the challenges that we faced throughout this project.

Finally, section VII is the conclusions.

2 Correlation

hier muss noch eine Einfuehrung rein !!!

2.1 Data Extraction

IN PROGRESS*** The first step to find the best pair for the pairs trading strategy is to download and manipulate all the relevant stock data. Yahoo finance seemed like a reasonable website, where we could find all the data we needed. They provide daily stock price information. Including the opening, close, high, low and the adjusted close prices. Furthermore yahoo finance gives us information about the daily traded volume and the ticker symbols. At first we built a very primeval code. By using that code we had to update the crumb manually hardly every day. That solution was not very handy, so we started to build a loop. With the help of the following Packages

- dplyr
- data.table
- xts
- zoo
- quantmod
- TTR
- PerformanceAnalytics

we managed to build a loop around the getSymbols function, which downloads daily stockprice Information for nearly 6000 Stocks. We had to code a loop around it because the getSymbols function was not coded to handle such an amount of information that we needed. Luckily a Ph.D. Student from Brasil, Marcelo Perlin had the same Problem that we got. He coded a package "BatchGetSymbols" with the help of this package it became very easy to download all the data we needed. In the following we will describe our code, which handles the data download and the problems we had with coding it.

When using the getSymbols function you can choose between Stock information from "NASDAQ", "AMEX" and "NYSE". We decided to only use information from NASDAQ, because it seems like the most liquid market with the highest marketcapitalization. So we define a variable called "TickerSymbol". Moreover we build a variable "symbols" which gives us a place where we can storage all the stock information we need. With the stockSymbols Function from the TTR package we get Information about the symbol, the name, the last sale, the market capitalization, the year of the initial public offering, the sector, the industry and the exchange place. For downloading the stock prices, the Tickersymbols of the stocks are very helpfull. But we also make use of the name and the market capitalization when we prepare the data.

```
TickerSymbol <- "NASDAQ"
symbols <- stockSymbols(TickerSymbol)
```

Listing 1: Downloading the Tickersymblos

After we downloaded the NASDAQ information we opened the symbols Matrix and recognized that there were more than one Ticker Symbol for the same Corporation. For Example: Accelerated Pharma, Inc. has two different Ticker Symbols, ACCP and ACCPW. So we removed the "duplicate" by using the following code:

```
symbols <- symbols[!duplicated(symbols[, 2]), ]
```

Listing 2: Deleting duplicated stock information

Furthermore we had problems with stocks which had no information about the market capitalization. Because yahoo finance does not have any price information about these stocks. So also we expect these stocks by using the following code:

```
symbols <- filter(symbols, !is.na(MarketCap))
```

Listing 3: Deleting duplicated stock information

Even after removing the stocks we mentioned we have 2.885 different stocks. To avoid a long waiting period we decide to only use stocks with a market capitalization higher than a Billion dollars. We are doing it because these are the most liquid stocks. And that's what we were looking for.

```
symbols <- symbols[grepl("B", symbols$MarketCap, ignore.case=TRUE),]
```

Listing 4: Deleting duplicated stock information

The last problem we are facing by downloading the stock prices and building a correlation matrix were funds that are traded on the NASDAQ. By his nature two funds, for example two Exchange Traded Funds (ETFs) that tracks the same index, commodity, bond, or the same basket of assets have a correlation near to one. These funds are useless for our strategy so we decide to ignore them. We are doing it by using the following code:

```
symbols <- symbols[!grepl("ETF", symbols$Name, ignore.case=TRUE),]  
symbols <- symbols[!grepl("Fund", symbols$Name, ignore.case=TRUE),]  
symbols <- symbols[!grepl("Shares", symbols$Name, ignore.case=TRUE),]  
symbols <- symbols[!grepl("Index", symbols$Name, ignore.case=TRUE),]
```

Listing 5: Deleting duplicated stock information

After using all the filters only 755 stocks should be left. For downloading the corresponding stock prices we only need the tickersymbols. So we are building a vector called tickers. Furthermore we are building two more variables for the start and the end date of the stock prices. We choose an interval of 150 days, because of the download time and the amount of data we have to download. All in all these 150 day interval should already show us a proper correlation matrix which we can work with. For downloading the stock prices we make use of the BatchGetSymbols function from the BatchGetSymbols package. Our storage for all the stock prices will be a dataframe called l.out. The BatchGetSymbols function controls the availability of the stock prices from yahoo and downloads them if there is any price information.

```
tickers <- symbols$Symbol  
first.date <- Sys.Date()-150  
last.date <- Sys.Date()  
l.out <- BatchGetSymbols(tickers = tickers,  
                        first.date = first.date,  
                        last.date = last.date)
```

Listing 6: Deleting duplicated stock information

The `l.out` dataframe contains two more dataframes, a control dataframe, that shows us every tickersymbol, the source where we downloaded the stockprices from, the total amount of observations, if the download worked and depending on the number of observation, if we keep and use the stock prices or not. So we only used the control dataframe to check if there are any stocks, that we will not use. The second dataframe is the tickers dataframe. That contains all the price Information. Just like mentioned before, it contains five different prices and in addition information about the trading date and the volume. We build a dataframe called Stock Prices, for filtering out the tickers dataframe (line 1). Furthermore we only needed information about the trading day, the adjusted close price and the tickersymbol, so we filtered the Stock Prices Matrix and renamed it to Adjusted Prices (line 2). After that, we had to reshape the dataframe, so we could work with it later (line 3). Because the original shape of the dataframe looked like the dataframe in Figure 1a, but we had to have it in a shape like in Figure 1b.

	price.adjusted	ref.date	ticker
1	37.25	2017-03-20	AAAP
2	36.99	2017-03-21	AAAP
3	37.30	2017-03-22	AAAP
4	39.58	2017-03-23	AAAP
5	38.98	2017-03-24	AAAP
6	39.20	2017-03-27	AAAP
7	39.21	2017-03-28	AAAP
8	40.00	2017-03-29	AAAP
9	40.00	2017-03-30	AAAP
10	39.86	2017-03-31	AAAP
11	39.69	2017-04-03	AAAP
12	39.77	2017-04-04	AAAP
13	39.47	2017-04-05	AAAP
14	39.24	2017-04-06	AAAP

(a) unshaped output

	ref.date	price.adjusted.AAAP	price.adjusted.AABA	price.adjusted.AAL	price.adjusted.AAOI	price.adjusted.AAON
1	2017-03-20	37.25	46.77	41.52617	50.33	36.56947
2	2017-03-21	36.99	45.77	40.25150	48.23	35.77231
3	2017-03-22	37.30	46.06	40.18179	52.63	35.47338
4	2017-03-23	39.58	46.60	41.23738	51.51	36.22071
5	2017-03-24	38.98	46.40	41.55604	55.02	36.12107
6	2017-03-27	39.20	46.40	41.56600	59.88	34.92533
7	2017-03-28	39.21	46.57	42.42242	58.56	34.67622
8	2017-03-29	40.00	46.78	41.78508	58.02	34.82569
9	2017-03-30	40.00	46.62	42.36267	55.35	35.12462
10	2017-03-31	39.86	46.41	42.12367	56.15	35.22427
11	2017-04-03	39.69	46.43	42.27304	53.47	34.22782
12	2017-04-04	39.77	46.23	40.72950	51.19	34.62640
13	2017-04-05	39.47	46.38	41.13779	44.68	34.07836
14	2017-04-06	39.24	46.28	41.54609	46.31	34.27765

(b) shaped output

Figure 1: Reshape the BatchgetSymbols Output

The only thing left was to rename the row names as trading dates from column one (line 5) and delete the first column because the dates are already the rownames (line 6) and rename the column names (line 7).

```

Stock_Prices <- l.out$df.tickers
Adjusted_Price <- Stock_Prices[-c(1:5)]
Adjusted_Price <- reshape(Adjusted_Price, idvar = "ref.date",
                           timevar = "ticker", direction = "wide")
rownames(Adjusted_Price) <- Adjusted_Price$ref.date
Adjusted_Price <- Adjusted_Price[, -1]
colnames(Adjusted_Price) <- sub(".*\\.\"", "", colnames(Adjusted_Price))

```

Listing 7: Deleting duplicated stock information

	AAAP	AABA	AAL	AAOI	AAON	AAPL	AAWW	AAXN	ABAX	ABCB
2017-03-20	37.25	46.77	41.52617	50.33	36.56947	140.3275	53.20	22.94	49.02101	47.14401
2017-03-21	36.99	45.77	40.25150	48.23	35.77231	138.7204	51.35	22.46	47.61500	44.33628
2017-03-22	37.30	46.06	40.18179	52.63	35.47338	140.2878	51.40	22.45	48.31302	43.16141
2017-03-23	39.58	46.60	41.23738	51.51	36.22071	139.7918	51.75	22.63	47.95404	43.31076
2017-03-24	38.98	46.40	41.55604	55.02	36.12107	139.5140	51.45	22.50	47.94407	44.30641
2017-03-27	39.20	46.40	41.56600	59.88	34.92533	139.7521	51.85	22.47	48.14350	44.45576
2017-03-28	39.21	46.57	42.42242	58.56	34.67622	142.6487	53.35	22.79	48.07370	44.65489
2017-03-29	40.00	46.78	41.78508	58.02	34.82569	142.9661	54.30	22.64	48.55235	44.30563
2017-03-30	40.00	46.62	42.36267	55.35	35.12462	142.7777	55.65	23.22	48.19336	46.30138
2017-03-31	39.86	46.41	42.12367	56.15	35.22427	142.5098	55.45	22.79	48.36288	46.00202
2017-04-03	39.69	46.43	42.27304	53.47	34.22782	142.5495	53.65	22.50	48.42271	45.00414
2017-04-04	39.77	46.23	40.72950	51.19	34.62640	143.6110	54.05	22.00	47.25602	45.25361
2017-04-05	39.47	46.38	41.13779	44.68	34.07836	142.8670	53.60	21.90	47.15630	44.10606
2017-04-06	39.24	46.28	41.54609	46.31	34.27765	142.5098	54.00	23.08	47.18622	44.60500

Figure 2: final dataframe


After all it should look like in Figure 2.

After preparing the matrix for the correlation part, we found out despite from the filtering of the BatchgetSymbols function there has been some columns with no information in it. So we also filtered them out.

```
Adjusted_Price = Adjusted_Price[sapply(Adjusted_Price,
                                       function(x) !any(is.na(x)))]
```

Listing 8: Deleting duplicated stock information

Finally we renamed the "Adjusted Price" Matrix into "final". Just to make clear that this is the matrix we will use in the correlation part. In conclusion we are comparing 669¹ different stocks.

 Data Extraction

2.2 Finding Correlated Stocks

One of the most important parts in the pairs trading strategy is to find stocks with a high correlation. For now we do not know if a correlation between a pair of 0.99 is more profitable than a pair with a correlation of 0.95.

In the following abstract it is necessary to discuss which correlation coefficient we will use, before we come to the next part in which we explain how to built a correlation matrix. In our later work we will only use high correlated pairs, so a correlation matrix is not very handy that is why we reshape it into a 3 column table in descending order.

¹Our data was downloaded on August 17th 2017

Like G. Udny Yule (1926) wrote ², a non parametric-method for measuring the correlation of a time-series is favorable. So in the following we will build a correlation matrix based on the Spearman rank correlation. The Spearman correlation between two variables is equal to the Pearson correlation between the rank values of those two variables. While Pearson's correlation only assesses linear relationships, Spearman's correlation assesses monotonic relationships. For our matter of concern the Spearman rank correlation coefficient is a lot better because of the nature of a time series with financial data. In general there is a positive trend in financial data, caused by the inflation. One trended time series regressed against another will often reveal a strong, but spurious, relationship. So there is a mutual dependence just because of the inflation. The Pearson correlation coefficient does not avoid this problem. So even if two stocks does not have a high correlation, you can not see it by using the Pearson method. Because the amount of trend determines the effect on correlation. Just a little trend can change the correlation result from insignificant to highly significant.

The Spearman rho is defined as:

$$\rho_s(X, Y) = \frac{12}{n(n^2 - 1)} \sum_{i=1}^n \left(\text{rank}(x_i) - \frac{n+1}{2} \right) \left(\text{rank}(y_i) - \frac{n+1}{2} \right).$$

The value of this correlation coefficient depends upon the relative rankings of the x_i and the y_i . Here x_i and y_i are the different stock prices for day i .

With regard of the subsection above we code the correlation matrix based on the "final" dataframe by using the Spearman method.

```
z <- cor(final, method = "spearman")
```

Listing 9: Deleting duplicated stock information

To get a good overview we reshape the correlation matrix in to a 3 column table. First of all we prepare to drop duplicates and meaningless information from the correlation matrix. Just by replacing duplicates and all the diagonals by na's.

```
z[lower.tri(z, diag=TRUE)] = NA
```

Listing 10: Deleting duplicated stock information

Then we turn the correlation matrix into a 3-column table.

```
z <- as.data.frame(as.table(z))
```

Listing 11: Deleting duplicated stock information

After that we get rid of the na's we flagged above.

```
z <- na.omit(z)
```

Listing 12: Deleting duplicated stock information

²G. Udny Yule, Journal of the Royal Statistical Society Vol. 89, No. 1 (Jan., 1926), pp. 1-63, "Why do we Sometimes get Nonsense-Correlations between Time-Series?—A Study in Sampling and the Nature of Time-Series"

Finally we sort the table in descending order and rename the rows in ascending order. And we rename the columns

```
z <- z[order(-z$Freq),]
rownames(z) <- 1:length(z$Freq)
colnames(z) <- c("Stock_1", "Stock_2", "Correlation")
```

Listing 13: Deleting duplicated stock information

As a result we get a 3 column table where we compare all of the 223,446 pairs ³. In Figure 4 and 5 we show the 60 pairs with the highest correlation.

	Stock 1	Stock 2	Correlation
1	PLUS	TTWO	0.9822248
2	CACQ	CZR	0.9808438
3	ALGN	IPGP	0.9793592
4	ALGN	PYPL	0.9774126
5	AABA	PYPL	0.9768497
6	AABA	ALGN	0.9753776
7	IPGP	PYPL	0.9747613
8	AABA	SEDG	0.9745923
9	GNCMA	LVNTA	0.9738789
10	ALGN	TREE	0.9737270

(a) 1-10

	Stock 1	Stock 2	Correlation
11	IPGP	PRXL	0.9736226
12	QDEL	VRSN	0.9734259
13	CRZO	XOG	0.9733666
14	PRXL	TREE	0.9730962
15	AABA	CBOE	0.9727287
16	ALGN	CBOE	0.9721332
17	IPGP	JOBS	0.9719207
18	CBOE	SEDG	0.9717959
19	MMSI	PYPL	0.9712109
20	APPF	PRXL	0.9709337

(b) 11-20

	Stock 1	Stock 2	Correlation
21	AABA	RYAAY	0.9709327
22	ALGN	JOBS	0.9708735
23	ALGN	RYAAY	0.9707208
24	ALGN	PRXL	0.9706159
25	ALGN	CSGP	0.9705626
26	AABA	TREE	0.9705152
27	IPGP	TREE	0.9703345
28	ALGN	VRSN	0.9703110
29	ADBE	ATVI	0.9701396
30	CATM	CRZO	0.9700536

(c) 21-30

Figure 3: Top 30 correlated pairs

	Stock 1	Stock 2	Correlation
31	ABMD	TREE	0.9699197
32	AABA	JOBS	0.9699113
33	CSGP	IPGP	0.9696349
34	PYPL	RYAAY	0.9695388
35	ALGN	NXPI	0.9693826
36	AABA	PRXL	0.9692727
37	EXLS	TREE	0.9691807
38	PRXL	PYPL	0.9691617
39	AABA	IPGP	0.9684991
40	AABA	VRSN	0.9682110

(a) 31-40

	Stock 1	Stock 2	Correlation
41	CACC	TREE	0.9681678
42	ICLR	PRXL	0.9679153
43	ICLR	PYPL	0.9678982
44	CSGP	PYPL	0.9674679
45	ADBE	ALGN	0.9673878
46	PYPL	TREE	0.9673744
47	CTSH	YY	0.9672398
48	ALGN	MMSI	0.9670717
49	PRXL	SEDG	0.9668776
50	JOBS	SEDG	0.9668869

(b) 41-50

	Stock 1	Stock 2	Correlation
51	EXLS	MMSI	0.9666748
52	SEDG	TREE	0.9666117
53	ALGN	QDEL	0.9665941
54	CBOE	IPGP	0.9662163
55	ADBE	PYPL	0.9660658
56	ALGN	SEDG	0.9660235
57	MMSI	TREE	0.9660218
58	NXPI	TREE	0.9657331
59	IPGP	MMSI	0.9656981
60	IPGP	SEDG	0.9656141

(c) 51-60

Figure 4: 30-60 correlated pairs

³Our data was downloaded on August 17th 2017

3 Execution

Based on the correlation described in the previous section, stocks are selected for further analysis moreover the strategy execution. The strategy as described in the beginning of this paper refers to Rinnea's and Suominen's paper on pair trading. Therefore the execution is based on their implemented strategy.

The following sections will describe the implementation of the regression, enter and exit strategy.

3.1 Regression

In the regression part, the goal is to find out the relationship between daily return and 5-day return in specific period, hence the trading direction of the pair. The period can be either for back-testing or performing purpose.

The trading direction is fixed as longing stock 1 and shorting stock 2 as default. This provides the return of the single-direction strategy. The real trading direction is determined by the trading direction vector created in later part.

The regression starts with the calculation of real daily return and 5-day return of the strategy.

```
#day-to-day and 5-day longing return (on stock 1)
Rt_long = c(1, diff(s1[,2])/head(s1[,2], -1))

for (i in 1:(length(s1$Date)-5)){
  R5t_long[i] = ((s1[i+5,2] - s1[i,2])/s1[i,2])
}
```

Listing 14: Return illustration - long stock 1

Then the daily and 5-day return vectors are summed respectively as R_t and $R5_t$. Then they are sorted the corresponding R_{t-n} , for each day t and each n days before t , as a regression table.

```
n_Rt=length(Rt)
Rt_0=Rt[6:n_Rt]
Rt_1=Rt[5:(n_Rt-1)]
Rt_2=Rt[4:(n_Rt-2)]
Rt_3=Rt[3:(n_Rt-3)]
Rt_4=Rt[2:(n_Rt-4)]
}
```

Listing 15: Sorting regression table

Here is the point needs to be added. A row-wise linear regression is performed to regress ($R5_t$) on (R_t). The regressed formula is $[R5_t = \alpha_t + \sum_{n=0}^4 \beta_{t-n} R_{t-n} + \varepsilon_t]$.

```
Rt_0_c=Rt_1_c=Rt_2_c=Rt_3_c=Rt_4_c=vector()
local(
  for (i in 1:length(R5t)){
    n=1:i
    lr=lm(R5t[n]~Rt_0[n]+Rt_1[n]+Rt_2[n]+Rt_3[n]+Rt_4[n])
```

```

coef=as.numeric(lr$coefficients)
Rt_0_c[i]<-coef[2]
Rt_1_c[i]<-coef[3]
Rt_2_c[i]<-coef[4]
Rt_3_c[i]<-coef[5]
Rt_4_c[i]<-coef[6]
}
)
}

```

Listing 16: Regression

Next, the (β_{t-n}) is taken average from the period 120 days before time (t) and 6 days before (t) .

```

for (i in 1:(length(R5t)-124)){
  n = (i+5):(i+119)
  Rt_0_ac[i] = sum(Rt_0_c[n])/115
  Rt_1_ac[i] = sum(Rt_1_c[n])/115
  Rt_2_ac[i] = sum(Rt_2_c[n])/115
  Rt_3_ac[i] = sum(Rt_3_c[n])/115
  Rt_4_ac[i] = sum(Rt_4_c[n])/115
}
}

```

Listing 17: Taking beta's average

Based on the average coefficient (β_{t-n}) , the expected 5-day return $(R5_t)$ is computed with the observed trading strategy return.

```

for (i in 1:length(Rt_4_ac)){
  ex_R5t[i]=Rt_1[120+i]*Rt_1_ac[i]+Rt_2[120+i]*Rt_2_ac[i]+
  Rt_3[120+i]*Rt_3_ac[i]+Rt_4[120+i]*Rt_4_ac[i]
}
}
}

```

Listing 18: Calculating expected 5-day return

Next, the expected 5-day return $(R5_t)$ is compared with the pre-determined limit set. The pre-determined limit in default setting is (5). At time (t) its trading direction (LS_{dir}) (refers to long-or-short direciton) is assigned as 1 or 2. If $(R5_t)$ is greater than the limit, we enter the default trading direction longing stock 1 and shorting stock 2, and the assigned $(trade_{dir})$ is 1. If $(R5_t)$ is smaller than the limit, the direction is reserved as shorting stock 1 and longing stock 2, the assigned $(trade_{dir})$ is 2.

```

trade_dir = vector()
trade_dir = ifelse(ex_R5t>limit,1,ifelse(ex_R5t<(-limit),2,0))
}
}

```

Listing 19: Assigning trading direction

Finally, a trading table is formed with component of the date of initiating the pair trading, the corresponding stock price of the pair stock, and the trading direction.

```
trade_table = data.frame(s1, s2[,2], c(rep(0,129),trade_dir))
}
```

Listing 20: Calculating

Here is the point needs to be added.

3.2 Enter Strategy

After the regression is executed the enter strategy is implemented using it's results. As Rinnea's and Suominenb's enter their is strategy based on the expected 5-day return of a pair being higher or lower than a predetermined limit. The limits used in the paper are $\pm 0.50\%$, $\pm 0.75\%$ and $\pm 1.00\%$, as they assume this amount would cover the transaction costs of this strategy. For the reproduction of their strategy the limit in the following code section is set to 0.005.

```
limit=0.005
trade_dir= vector()
trade_dir = ifelse(ex_R5t>limit,1,ifelse(ex_R5t<(-limit),2,0))
```

Listing 21: Enter Trade Strategy

There are two enter strategies in place. This is implemented as shown in listing 2. Firstly stock 1 is long positioned and stock 2 short or secondly stock 1 short and stock 2 long positioned. Both strategies are using the same limit as a trigger for entering the trade.

3.3 Exit Strategy

After entering the strategy, there are two different options used for exiting it. Firstly a closing after a 5 day window. This is shown in Listing 3.

```
exit_index=which(trade_table$LS_dir !=0)+5
exit_table=data.frame(tail(s1,trade_length)[exit_index,],tail(s2[,2],trade_length)
[exit_index ])
```

Listing 22: Simple Exit Strategy

Secondly exiting or closing withing a 10 day window, where there is no closing within the first three days but within the following 7 days when the return changes it's sign, either from positiv to negativ or the other way around.

```
close_row = vector ()
for (i in 1:(nrow(inter_table_2nd)-10)){
  if (inter_table_2nd$LS_dir[i]==0){
    close_row[i] = NA
  } else {
    value = Position(function(b)switch(
      inter_table_2nd$LS_dir[i], b<0, b>0),
      inter_table_2nd$ex_R5t[(i+4):(i+10)])
    close_row[i]= i+3+min(ifelse(is.na(value),11,value),10)}
}
```

Listing 23: Complex Exit Strategy

4 Performance Analysis

5 Risk Management

To assess the risk of the used trading strategies it can be chosen from several methods, which have different advantages and disadvantages. For this paper two different methods will be used, which are value at risk and expected shortfall.

5.1 Value at Risk

Value at risk is the threshold value such that the loss will exceed this value with probability of q . For this paper q is set to 1% as we assume the return follows a normal distribution. Therefore the mean and standard deviation of the return is used to calculate the value at risk. It is calculated as shown in Listing 4.

```
q = 0.01
VaR_norm = qnorm (q,mean(return_2nd),sd(return_2nd))
```

Listing 24: Value at Risk of the return

Analysing the results of the value at risk, it shows that the data has a different distribution in comparison to the normal distribution, as it is more heavy tailed and the value at risk therefore does not represent the actual size of the loss. On the other hand it is easy to calculate and to understand. To improve the value at risk the GARCH and ARCH model could be applied. As this would exceed the scope of this paper, it is not implemented.

5.2 Expected Shortfall

To enhance the risk assessment and overcome the downsides of the value at risk analysis the expected shortfall is introduced. It is more sensitive to the shape of the tail of the loss distribution and therefore suitable to use in the context of financial risk management.

```
ES_emp = mean(return_2nd[return_2nd < VaR_emp])
n = 1000000
sim_N = rnorm(n,mean(return_2nd),sd(return_2nd))
ES_norm = mean(sim_N[sim_N < VaR_norm])
```

Listing 25: Expected Shortfall of the return

Listing 6 shows the calculation of the expected shortfall under normal distribution and empirically. Again, similar to the value at risk, the normal distribution underestimates the shortfall.

6 Challenges

Throughout the project phase, we encountered many different kind of challenges. The first challenge was to translate / replicate the papers strategy idea into a running and performing R program coding. In the execution part of the project we faced the following problems:

1. Different stock lengths,
2. NA handling,
3. Finding the right packages,
4. Avoiding loops,
5. Availability of Yahoo!Finance,
6. the `getSymbols`-function produced several errors because of the bad condition of the extracted Yahoo!Finance data.

Concerning 1., we faced many problems building vectors, matrices and/or data frames because of the different lengths of stocks. We downloaded the historical data from Yahoo!Finance, meaning that the stocks historical data started at different starting points.

We then had to take care of 2.: the NA handling. As nearly every data set that you can find in the world wide web, the downloaded stocks historical data had several days where no trades were happening, like weekends, holidays, etc. We had to get rid of the NAs, so that they do not produce problems in the further coding.

3. Finding the right packages is very important. First of all, we followed the structure Get it work first, improve later. Mostly, we wrote our own code with many loops and a quite messy structure. Later, we found packages that simplified the code (and our lives) a lot.

With the right packages, we could also get the 4. problem of avoiding loops nearly completely out of the way.

5. Availability of Yahoo!Finance: The first 'data extraction'-code we wrote, was one big function where we extracted the historical data from Yahoo!Finance. Several times throughout the project phase, we could not access it, because Yahoo!Finance was not available. Which of course made the further coding a tough - if not impossible - job.

Last but not least, we got several errors from the `getSymbols`-function, which was the self-written 'data extraction' program code. The errors were caused, for example, because of the different lengths of stocks or the unavailability of Yahoo!Finance.


7 Conclusion

In this paper we gave a brief overview on pair trading strategy using the paper on "How some bankers made a million by trading just two securities?" by Kalle Rinnea and Matti Suominen. Firstly reproducing their strategy in R and enhancing their exploratory analysis by using different stock pairs based on their correlation. Within the paper a closer look on stock correlation was given. After that it was shown how to execute Rinnea's and Suominen's strategy. Furthermore the performance of the strategy was analysed using sharpe ratio. Ending the analysis by implementing risk management measures. While exploring the topic of pair trading several challenges were encountered, which are also mentioned in the paper. For future work one may enhance the study to other stock exchanges or by using a more complex strategy to forecast future returns than a regression. Besides that the risk assessment could be expanded by using GARCH and ARCH modelling.

References

- [1] Rinne, K. & Souminen, M. (2016) *How some bankers made a million by trading just two securities?* Journal of Empirical Finance.
- [2] Gatev, E., Goetzmann, W. N., Rouwenhorst, K. G. (2006) *Pairs Trading: Performance of a Relative-Value Arbitrage Rule.* The Review of Financial Studies, V. 19, No. 3.
- [3] Jylh, P., Rinne, K., Suominen, M. (2014) *Do hedge funds supply or demand liquidity?* Journal of Empirical Finance.

8 Appendix

The complete code and the  Quantlets used in this paper can be found on Github.