Data Mining in R

– SEMINAR WINTER SEMESTER 2015/2016 –

Introduction to quantitative modeling with the R package quantmod

– SEMINAR PAPER –

Submitted by:
Marius Jaeger

Advisor:
Prof. Dr. Dirk Neumann

Contents

1 Introduction 1

2 What quantmod is and what it is not 1

3 Quantitative trading and modeling 1

4 Getting started 2
- 4.1 Installing quantmod . . . 2
- 4.2 Dependent packages . . . 2

5 Using quantmod 3
- 5.1 Getting data . . . 3
- 5.2 Charting . . . 4
- 5.3 Handling data . . . 9
- 5.4 Adding own indicators and advanced charting . . . 10
- 5.5 Building models . . . 12

6 Conclusion 13

A References i

B List of Figures ii

C List of Tables iii

1 Introduction

The R package quantmod is written by Jeffrey A. Ryan, designed to assist traders in quantitative modeling. For traders, losing time often means losing money. Therefore, quantmod wraps up the capabilities of several statistics, charting and data handling packages to make testing, developing and use of quantitative trading models easier and faster. This enables traders to design their own models and to test them with data from different sources. Traders can increase their profits getting better predicting future trends on markets and their effects on different assets. Hopefully this improves the efficiency of the financial markets and thereby increases the resource allocation of the whole economy.
In this seminar paper, we will introduce into the usage of quantmod and the variety of different capabilities it provides. The scheme of the introduction partly follows the one on [8] and the information provided by [9]. In addition an introduction to quantitative trading will be given. R code examples are framed. All explanations are given for the command line interface of R.

2 What quantmod is and what it is not

In the depth of the CRAN (Comprehensive R Archive Network), traders can always find different kinds of charting, testing and other statistics packages which can be used for quantitative trading. But traders had to puzzle together their own trading tools. Of course this is an annoying and frustrating work. Especially the data handling and processing while using different R packages might cause problems. As a remedy, quantmod is designed and written to solve most of those problems and to provide a package that allows to set up, test and deploy trading models and to process financial data easier and faster than before. The package is designed to assist traders in developing, testing and implementing quantitative trading models. This does not mean, that quantmod is another statistics package but rather a developing package. Therefore, quantmod uses no own math functions "–" it uses the ones provided by several other packages, which we will go into later on. In summary what quantmod provides, are functions designed to rapidly set up and test new models by wrapping and mixing up the capabilities of other R packages, which already existed [8].

3 Quantitative trading and modeling

One mean of quantitative trading can be trading of hundreds of thousands of stocks or securities, which is mostly done by big investors like financial institutions or funds. However it can also be seen as individual trading by following the analysis of financial data and mathematical models. Therefore, the quantitative trader always needs quantitative modeling, which contains cleaning, handling and visualization of data, analyzing, fitting and testing of trading models. For example, analyzing price and volume of a stock are some of the most common information, which traders use to identify trading opportunities [6].

4 Getting started

Starting quantmod always starts with installing quantmod and its dependent packages from the CRAN repository.

4.1 Installing quantmod

Installing quantmod is as easy as installing any other R package. Just type [7]:

```
install.packages("quantmod", dependencies=TRUE)
```

This will automatically install quantmod and all dependent packages like "xts", "zoo" and "TTR" from CRAN. If you want to know, if quantmod is already installed, find out by typing:

```
"quantmod" %in% installed.packages()
TRUE
```

The return will be boolean. This might be useful, if you like to write some kind of automatized script running on different computers. If you are using quantmod on different R applications, it might be necessary to reload the package by calling:

```
require("quantmod")
```

This might also be useful, if you are running R scripts from a .batch file or some GUI. Quantmod, like any other R package, can also be installed from an .zip file, which is also available at CRAN. To install from a .zip file, type:

```
 install.packages(choose.files(), repos=NULL)
```

choose.files() will open a windows file dialog to choose files. Choosing multiple files will not work, because install.packages() is not able to handle a list of files.

4.2 Dependent packages

Quantmod itself just provides the data base connection and a few other functions. For this reason, quantmod requires several dependent packages [Figure 1]. One of them is the "xts"-package, which provides functions for handling time-based series [12]. Moreover, the functions from the "zoo"-package are used for handling irregular and regular time-based series [13]. We will refer to this later on. Afterwards, we will create technical trading rules by using the "TTR"-package [10]. This package provides several kinds of modeling and fitting functions, which are even useful in non-financial context.
In the return of the code given above, one can see, which dependent packages are loaded by quantmod. [Figure 1]

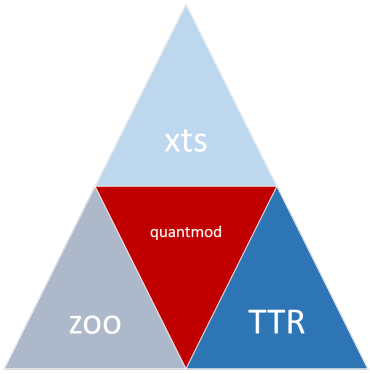

Figure 1: Quantmod and its dependent packages.

5 Using quantmod

After succesfully installing quantmod to your R, you are ready to start working. Getting data and learning how to handle it is essential.Afterwards, charting or modeling can follow, respectively.

5.1 Getting data

Using quantmod mostly starts with getting data from any of the data sources supported by quantmod. Supported data sources are [9]:

- yahoo as OHLC data
- google as OHLC data
- MySQL for own data
- FRED as economic series
- CSV files for your own or downloaded data
- binary R data types: .RData or .rda

The CSV-file import is difficult, because of different separators and several problems, for example caused by row.number and col.number.

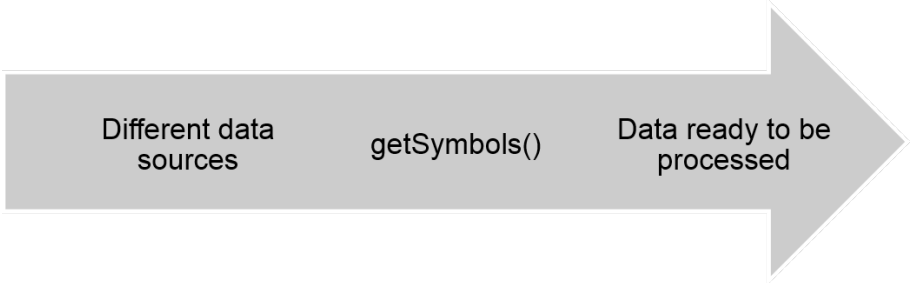

Figure 2: Data loading scheme

In quantmod, all these data sources can be used with only one function getSymbols(), which makes getting data easy and fast [Figure 2]. The function calls are as follows [7]:

```
getSymbols("^GDAXI", src="yahoo")  # loads DAX data from GOOGLE finance
getSymbols("MYAGM2EZM196N", src="FRED") # loads data of M3 in the euro ↩
    area from FRED (Federal Reserve of St. Louis)
getSymbols("VLKAY", src="google")  # loads the course of Volkswagen from ↩
    Google finance
getSymbols("XPT/EUR", src="oanda") # loads the course of platinum in ↩
    euro
```

getSymbols() creates an object in your workspace (can be changed by specifications), of the type mentioned before and returns the name of the object. Therefore, it might be useful to directly save the name in a R variable like:

```
d1 <- as.name(getSymbols("^GDAXI", src="yahoo"))
```

This makes it easier to use the data later. But be careful - this does not work with all the quantmod functions.

It is also possible and even useful to specify lookup parameters for later sessions by using the function saveSymbolLookup(). In this case the symbol lookup is saved to a file named "prevLookup.rda" into your working working directory.

```
setSymbolLookup(VLKAY='google', ^GDAXI='yahoo')
setSymbolLookup(MYAGM2EZM196N='FRED')
setSymbolLookup(XPTEUR=list(name="XPT/EUR",src="oanda"))
saveSymbolLookup(file="prevLookup.rda")

getSymbols(c("VLKAY","^GDAX","MYAGM2EZM196N","XPTEUR"))
```

This returns exactly the same as the code given before [7]. But in future sessions, the lookup can be reused by calling:

```
loadSymbolLookup(file="prevLookup.rda")
getSymbols(c("VLKAY","^GDAX","MYAGM2EZM196N","XPTEUR"))
```

The function loadSymbolLookup() only loads the specifications set before by calling setSymbolLookup() back into your workspace. The data itself is not saved on your computer and will be downloaded again from the internet or any source specified before. It is necessary to save the whole workspace or single data sets, if you want to avoid downloading it again.

5.2 Charting

Now we got some idea of how data can be loaded into your R environment. Then it is time to do some charting, one of quantmod's specialties. Especially for charting quantmod provides functions, which are really easy to handle and which create fancy graphs without hard work, compared to less automatized charting and plotting packages.

Let's start with getting some data, for example the stock exchange price of the Volkswagen AG from YAHOO-finance:

```
VW <-as.name(getSymbols("VOW3.DE" ,src="yahoo"))
```

And plot a fancy bar chart of the data:

```
barChart(VW, subset = '2015-06::2016-01')
```

To make the single bars visible, it is necessary to restrict the plotted data to the subset of last half-year's stock exchange prices [Figure 3].

Like mentioned before, data loaded from Yahoo Finance is of type OHLC and volume series.

Figure 3: Bar chart of VW's stock prices and traded volumes during 2nd half of 2015.

OHLC means, that every trading day is a vector of four values[5][Figure 5]:

- The opening price of the stock
- The closing price
- The maximum price of the day
- The minimum price of the day

Sometimes the adjusted closing price is also included. If there is also volume series added, there is another column containing the daily traded volume.

In a bar chart there is plotted one bar for every trading day. The highest point of the bar shows the maximum and the lowest point shows the minimum of the day's stock price. The tick at the left shows the starting, the tick at the right shows the closing price. Red bars indicate falling prices, green ones indicate rising prices. Other chart styles available are line chart, candlestick chart, and matchstick chart [2, 9].

Now, after plotting our first chart, let's go on and add our first indicator. As most indicators are provided by the TTR package, it might be necessary to load the package by calling:

```
require("TTR")
```

The indicator we are going to use for this example is the Average Directional Movement Index (ADX), which compares daily maxima and minima over a long term to measure the strength of trends [1]. The standard parameters can be looked up at [9]. The function call addADX() plots the indicator underneath the traded volume plot. To get more sophisticated, we now change our plot from a bar to a matchstick chart [Figure 4] and add some Bollinger Bands by calling:

```
getSymbols("VOW3.DE" ,src="yahoo")
chartSeries(VOW3.DE,type = "matchstick",  subset='2015-06::2016-01', ↩
   TA=c(addVo(), addADX()))
addBBands()
```

This time we use the chartSeries() function, which provides much more options. As seen, the addition of the indicators like addBBands() can be done as an argument to the chartSeries() function or later by calling the function.

The Bollinger Bands show a 20 periods moving average and a band of two-times standard deviation. This gives a hint whether prices are relatively high or low, compared to previous trades. As arguments to the function, number of periods and bandwidth can be changed [3]. Many other indicators are accessible as shown in the table later.

Figure 4: Matchstick chart of BBands and ADX.

```
> VOW3.DE[1:40]
           VOW3.DE.Open VOW3.DE.High VOW3.DE.Low VOW3.DE.Close VOW3.DE.Volume VOW3.DE.Adjusted
2007-12-28        99.55       100.00       99.01        100.00          45800           81.14
2008-01-02        99.83       101.15       98.64         99.15          81700           80.45
2008-01-03       100.63       100.63       95.94         96.55          98300           78.34
2008-01-04        96.64        96.75       93.20         94.10         566200           76.35
2008-01-07        93.75        94.39       92.86         93.60         270000           75.95
2008-01-08        93.91        95.84       93.85         94.65         296400           76.80
2008-01-09        93.87        95.00       92.44         93.90         467300           76.19
2008-01-10        93.88        94.79       92.80         93.32         479600           75.72
2008-01-11        92.75        95.16       92.75         95.02         474400           77.10
2008-01-14        93.80        96.78       93.80         96.29         279400           78.13
2008-01-15        95.81        96.23       92.72         94.00         530800           76.27
2008-01-16        92.66        95.31       91.81         94.62         544800           76.77
2008-01-17        93.94        95.46       93.62         93.66         418600           76.00
2008-01-18        93.09        93.30       91.19         91.48         454800           74.23
2008-01-21        90.26        92.61       89.23         90.00         581700           73.03
2008-01-22        86.00        92.99       86.00         91.64         820300           74.36
2008-01-23        92.68        93.00       88.50         89.20         469700           72.38
2008-01-24        90.77        95.60       90.41         95.11         696700           77.17
2008-01-25        96.50        99.00       94.71         95.10         596300           77.16
2008-01-28        94.50        96.49       93.25         96.35         269500           78.18
2008-01-29        96.49        97.60       94.90         95.60         236700           77.57
2008-01-30        94.42        94.42       92.40         93.04         309300           75.49
2008-01-31        93.04        93.30       90.51         92.90         386000           75.38
2008-02-01        92.90        94.64       92.40         93.60         359200           75.95
2008-02-04        92.91        93.50       91.29         92.25         170700           74.85
2008-02-05        92.15        92.15       89.60         89.92         394100           72.96
2008-02-06        88.63        91.79       88.63         91.24         330800           74.03
2008-02-07        90.35        90.72       88.80         90.14         337100           73.14
2008-02-08        91.47        91.51       89.21         89.87         277700           72.92
2008-02-11        89.24        90.57       88.71         89.85         377400           72.90
2008-02-12        90.30        92.39       89.89         92.30         345900           74.89
2008-02-13        91.47        92.90       91.47         92.21         253800           74.82
2008-02-14        92.96        93.15       91.56         91.80         331000           74.49
```

Figure 5: Part of the VOW3.DE OHLC data.

Function	Indicator
addBBand()	Bollinger bands
addADX()	Average Directional Movement Index
addATR()	Average True Range
addCCI()	Commodity Channel Index
addCMF()	Chaiken Money Flow
addCMO()	Chande Momentum Oscillator
addDEMA()	Double Exponential Moving Average
addDPO()	Detrended Price Oscillator
addEMA()	Exponential Moving Average
addEnvelope()	Price Envelope
addEVWMA()	Exponential Volume Weighted Moving Average
addExpiry()	Options and Futures Expiration
addMACD()	Moving Average Convergence Divergence
addMomentum()	Momentum
addROC()	Rate of Change
addRSI()	Relative Strength Indicator
addSAR()	Parabolic Stop and
addSMA()	Simple Moving Average
addSMI()	Stochastic Momentum Index
addTRIX()	Triple Smoothed Exponential Oscillator
addVo()	Volume
addWMA()	Weighted Moving Average
addWPR()	Williams Percent R
addZLEMA()	ZLEMA

Table 1: All indicators usable in quantmod [4].

5.3 Handling data

As learned before, much of the data we have to handle in quantitative trading, is of type OHLC. OHLC is an abbreviation for the column names of the data **O**pen, **H**igh, **L**ow, **C**lose [5]. If you are not sure about the type of the data you are using, there are some useful functions like is.OHLC(), has.OHLC(), has.Op(), has.Cl(), has.Hi(), has.Lo(), has.Vo(), has.Ad().
[9]. These functions do what they say, and their returns are boolean. For example:

```
is.OHLC(VOW3.DE)
TRUE
```

It might also be useful to extract columns or lines. This can be done by using the functions Op(), Cl(), Hi(), Lo() to extract columns and seriesHi() and seriesLo() to extract the lines containing the highest Hi-value of the series respective the line with the lowest Lo-value. For example

```
Hi(VOW3.DE)
```

extracts all the Hi-values of the series.

But what will happen, if we combine the functions we learned before like in the following?

```
OpOp(VOW3.DE)
```

They will return a series containing the percentage change from one day's opening value to the next day's opening value [5]. It is also possible to use the functions provided by the xts package to transform time-based series. These functions allow fast generation of time-based subsets like

```
last(VOW3.DE, '-5 days')
```

, which gets you the last five days of the series.

Other transformations provided are Lag(), which lags one period or more, if specified. For example, calling

```
Lag(VOW3.DE[0:10],3)
```

lags 3 periods. Obviously the Next() function does exactly the opposite. Moreover, the Delt() function returns the percentage change in whatever and in every period you want. For example

```
abs(Delt(Lo(VOW3.DE), Hi(VOW3.DE), c(0,6)))
```

returns the daily volatility and the change between one day's minimum and the maximum six days after, which actually says nothing.

While working with time-based series, it might also be useful to change the periodicity of your series. Whether for reasons of efficiency in computing or clear representation, xts has what you need. The functions to.monthly(), to.weekly() and, to.daily() do what they say and what you mostly need. For example the to.weekly() function transforms the series to a data set containing only fridays. The to.monthly() function returns a list containing only the first day of each month. For the more ambitioned ones, the function to.period() allows to change the period in every way you like (for further information look [12]). If you like to know the periodicity of your data, use

periodicity() [5].

But transforming data does not already allow to develop trading models or to calculate expected weekly returns. To get such information, we need to apply functions by period. This is what the function apply.weekly() does. It allows you to apply your own functions to any time period that's possible with your data. For example:

```
period.apply(VOW3.DE ,endpoints(VOW3.DE ,on='months'), FUN=function(x) { ↩
    max(Hi(x)) } ) ['2015']
apply.monthly(VOW3.DE, FUN=function(x) { min(Hi(x)) } ) ['2015']
```

These two calls do the same, but the first one using the endpoints() function to generate the period, is more general. Now we can write our own function to calculate the daily returns of an invest or, to make it easier, just use one of the implemented functions dailyReturn(), weeklyReturn(), or monthlyReturn() or all of them at once:

```
allReturns(VOW3.DE)['2015-06::2016']
```

Looking at the output of the example, one can see, that Volkswagen had a bad year. For further informations on data handling and time-based series look [5, 12].

5.4 Adding own indicators and advanced charting

Now, After learning how to handle data, the ambitioned trader might know his own trading signals, which are not implemented in quantmod or TTR. Therefore, TTR provides the opportunity to define your own indicators and to add them to your charts, by using the addTA() respective the newTA() function. For example, we already charted some data like the SAP stock price as OHLC data. Afterwards, we added the Williams Percent R indicator [11] to the chart [Figure 6].

```
getSymbols('SAP')
chartSeries(SAP, type = "matchstick", subset = '2015-06::2016', TA = NULL)
addWPR()
```

If you need your own indicator and additionally need to change the theme of your chart later on, use for example this call [Figure 7]:

```
addTA(abs(Delt(Lo(SAP), Hi(SAP), 0)), col="blue", type='h')
reChart(SAP, type="matchstick", theme="white",subset='2015-06::2016')
```

For further information on themes, look chartTheme() at [9]. The reChart() function is really useful, because it changes the whole chart settings, like subset or style without displacing all your indicators.
But what if we need a simple moving average and accidentally added the exponential moving average? Just drop it by using this call:

```
addEMA()
dropTA('EMA')
addSMA()
```

If you like to export your charts to pdf, this piece of code might also be useful:

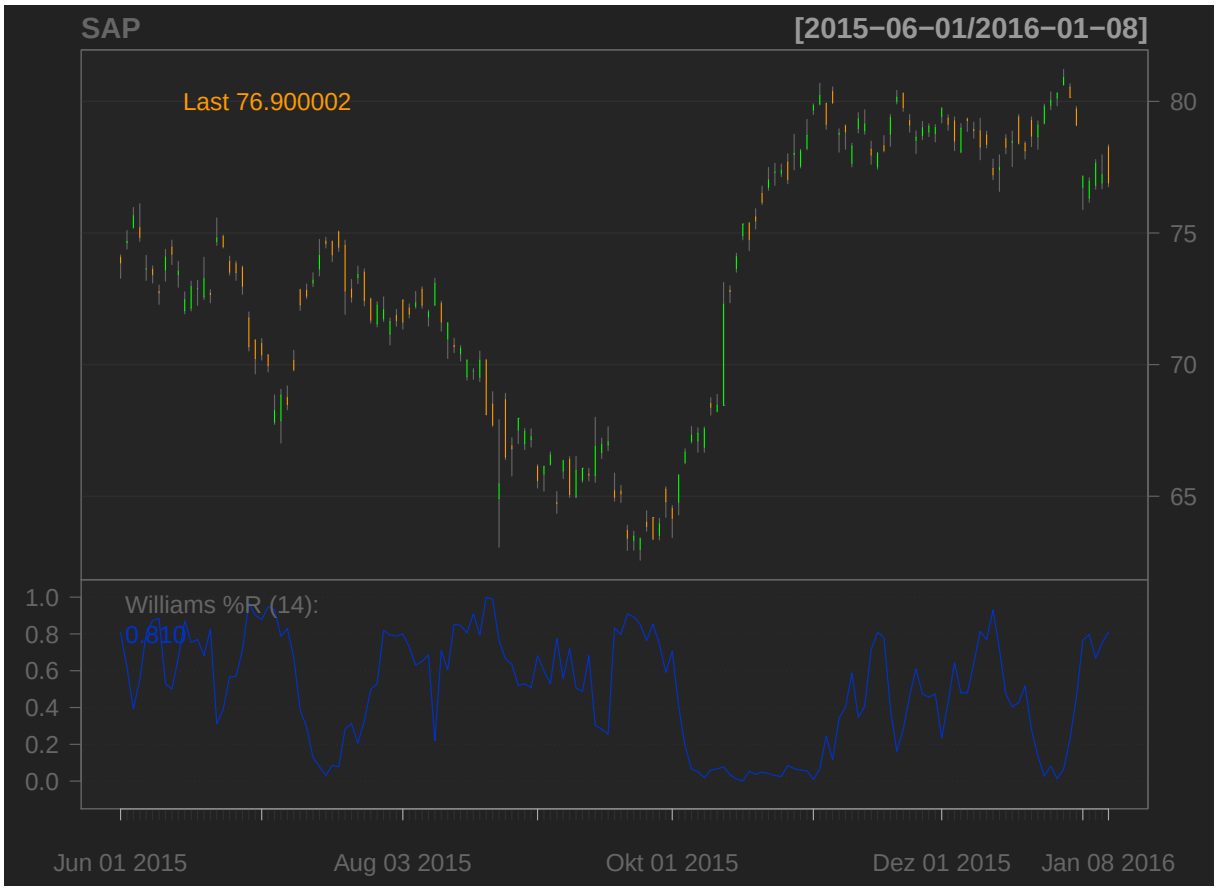

Figure 6: SAP chart with WPR added.

```
pdf(choose.files())
chartSeries(SAP, type = "matchstick", subset = '2015-06::2016', TA = NULL)
addBBands()
dev.off()
```

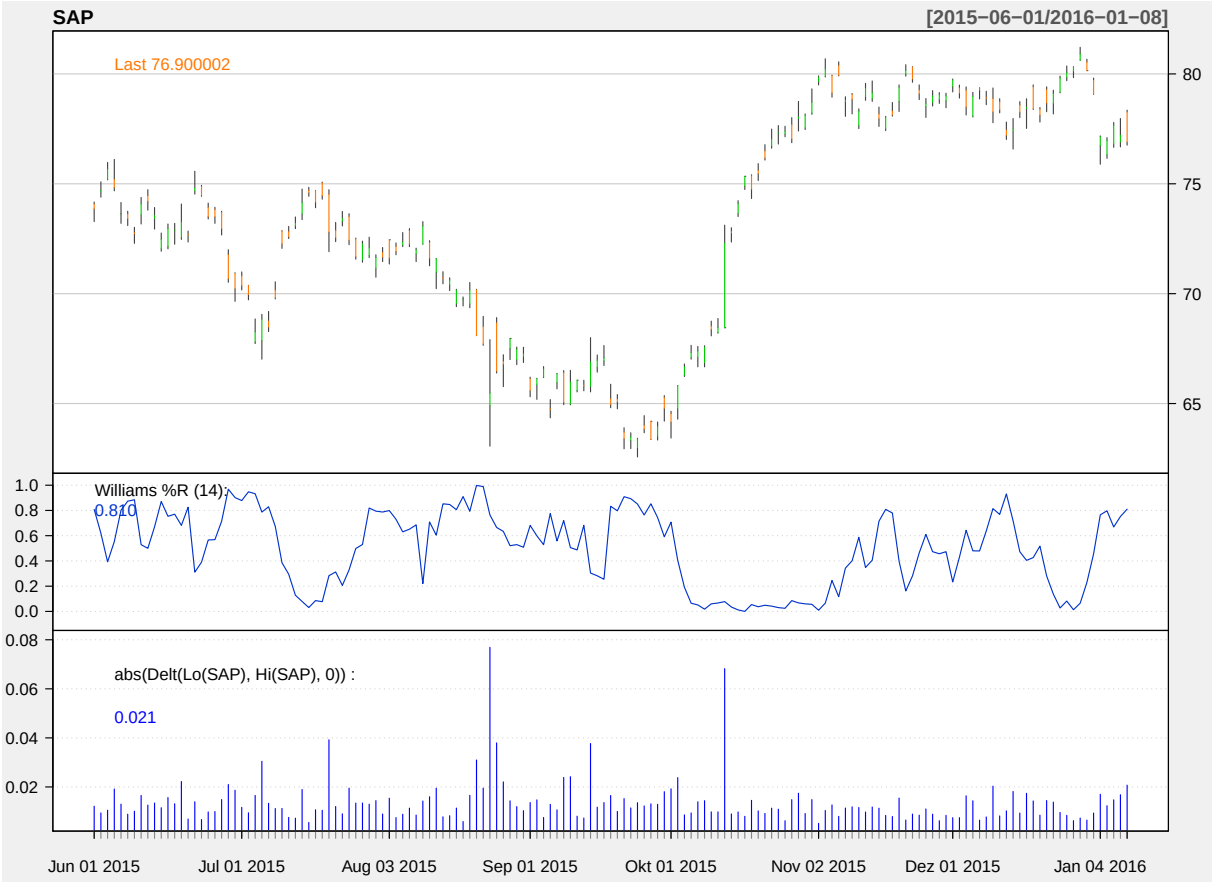

Figure 7: SAP chart with WPR added and volatility in white.

5.5 Building models

Quantmod allows to build and test models. Though the development of this functionality stopped, there is just one small function for this issue. But for completeness, one example will be given here. The first function you need, is the specifyModel() function, which does what it says:

```
myMod<-specifyModel(OpLo(SAP) ~ Next(Cl(SAP)))
```

We specified the model myMod implying some correlations in the OHLC data. This is our new trading rule, which we are going to test now. First, we have to fit it by using the function buildModel():

```
mybuild<-buildModel(myMod, ↩
    method="rpart",training.per=c('2015-06-01','2016-01-01'))
```

training.per sets the time subset 'from', 'to', method sets the 'fitting method'; this time "rpart" from the rpart package was chosen.
After fitting, we can use the tradeModel() function to compute the estimated returns trading our rule over the whole data set.
The call

```
tradeModel(mybuild)
```

returns risk measures and estimated profits, which seem very low, while the risk is very high for our rule. Unfortunately, many of the modeling functions planned at the beginning, never

got developed. As a result, quantmod is somehow limited to charting and data handling, while modeling has few functionalities and is badly documented [9].

6 Conclusion

In conclusion, one can say, that quantmod is a powerful R package if you want to make your quantitative modeling faster and easier. It does not and does not try to replace other R packages like portfolio. One of its strengths is the easy getSymbols() function, loading several kinds of data from several sources using just one function. Furthermore, it is easy to get into quantmod's functionalities, though it provides more sophisticated ones, which could not be part of this short paper (take a look at [9]). Sadly many formerly planned functions never got implemented to quantmod. But it is also in its current condition a useful package, which I can recommend to every one who is interested in data analysis, particularly in stock charting.

A References

[1] *Average dircetional movement index*. https://en.wikipedia.org/wiki/Average_directional_movement_index. Accessed: 2016-01-09.

[2] *Bar chart definition*. http://www.investopedia.com/terms/b/barchart.asp. Accessed: 2016-01-09.

[3] *Bollinger Bands*. https://en.wikipedia.org/wiki/Bollinger_Bands. Accessed: 2016-01-09.

[4] *charting examples*. http://www.quantmod.com/examples/charting/. Accessed: 2016-01-09.

[5] *data handling examples*. http://www.quantmod.com/examples/data/. Accessed: 2016-01-09.

[6] *definition of quantitative trading*. http://www.investopedia.com/terms/q/quantitative-trading.asp. Accessed: 2016-01-10.

[7] *quantmod introduction*. http://www.quantmod.com/examples/intro/. Accessed: 2016-01-03.

[8] *quantmod.com*. http://www.quantmod.com. Accessed: 2016-01-09.

[9] *quantmod.pdf*. http://www.quantmod.com/documentation/quantmod.pdf. Accessed: 2015-12-23.

[10] *TTR.pdf*. https://cran.r-project.org/web/packages/TTR/TTR.pdf. Accessed: 2016-01-09.

[11] *Williams Percent R*. https://en.wikipedia.org/wiki/Williams_%25R. Accessed: 2016-01-06.

[12] *xts.pdf*. https://cran.r-project.org/web/packages/xts/xts.pdf. Accessed: 2016-01-08.

[13] *zoo.pdf*. https://cran.r-project.org/web/packages/zoo/zoo.pdf. Accessed: 2016-01-09.

B List of Figures

1 Quantmod and its dependent packages. 3
2 Data loading scheme 3
3 Bar chart of VW's stock prices and traded volumes during 2nd half of 2015. . . . 5
4 Matchstick chart of BBands and ADX. 7
5 Part of the VOW3.DE OHLC data. 7
6 SAP chart with WPR added. 11
7 SAP chart with WPR added and volatility in white. 12

C List of Tables

1 All indicators usable in quantmod [4]. 8