Assignment 2 – Data Analysis Visualisation

Dumb forecasting investment strategy

Joao Viegas

27.05.2018

Student R00157699

Data Visualisation and Analytics

Data8007

Higher Diploma in Data Science and Analytics

Table of Contents

Introduction

Finance data provides a source for a vast field of research for obvious reasons, as for instance the impact it has in the world but also because of its appeal in its richness, variety and correlations with multiple aspects of our daily life.

Time series processing and analysis plays a big part in that research, and in this assignment we've conducted a simple investment experiment simulation, using time series modeling and forecasting, and using a couple of visualisation solutions to describe and display the outcome of such simulation.

The visualisation solutions are intended to be similar to dashboards, and to provide detail on every aspect of the simulation, allowing to have a general overview and also a drill down capabilities to go to the smallest detail of the analysis.

In the following sections we'll define the simulation experiment, we'll then describe the two visualisation solutions, its techniques and related tools, and finally, we'll conclude on the performance and comparison of the visualisation solutions.

The experiment

In this section we'll describe the investment experiment, which was a simulation based on historical corporation shares data, modeling the time series with an arima model, and using the model to forecast and based on the forecast, to decide the investment position on the corporation shares, buy or sell.

The experiment was implemented in R language, which is considered the "lingua franca" of statistics nowadays. It is open source, runs in all major platforms, provides a free development environment and tools and has an extensive support community that creates modules for every possible data science feature that we can think of. We've used some specific packages to conduct the experiment, of which we will be providing details where appropriate in this document.

Having chosen a company, from the Nasdaq exchange, we define a time frame, on that we will perform the simulation. We then retrieve the corporation historical shares data, in that precise time frame, we use the package *Quandl* to get the time series from the internet, and we then use the daily close price of the shares to model the time series. The *forecast* package provides an *auto.arima* function that searches for a best fit parameters model. We also define the training window and the forecast window as variables for the experiment. So we start by training from the beginning of our data time frame, on the training window provided, and forecast through the forecasting window defined, after that we align the training window with the previous forecasting window so that we are able to forecast to the adjacent forecasting window. So in practice, we are forecasting values at an interval defined by the forecasting window, except for the initial training window.

While forecasting, we then compare the last real value, which happens to be our last point of the training window, with the last value in the forecast, and based on the comparison, we decide whether to buy (if forecast is higher than last real value) or to sell (if lower) one share of the corporation in question.

The balance simulation for those operations starts with zero funds, and goes into negative as it executes the initial buy. In the end we can infer about the profit made, whether the balance changed positively, or if being negative, the actual price of the share being held, surpasses the balance at that point in time.

The visualisation solutions

As part of this experiment we've created and evaluated two visualisation solutions, one based on the *shinydashboard* library, and an alternative one, as a storyboard implemented using the *flexdashboard* library.

The full code can be seen in [github](#).

Shiny dashboard

The shiny dashboard strongest point is the user interaction capabilities, Shiny enables us to create interactive user interfaces, similar to web apps, where we can implement an event base graphical solution to perform data analysis. It has an interesting set of capabilities, providing a quick development cycle for data visualisation projects, leveraging established user interface technologies as CSS, html and javascript.

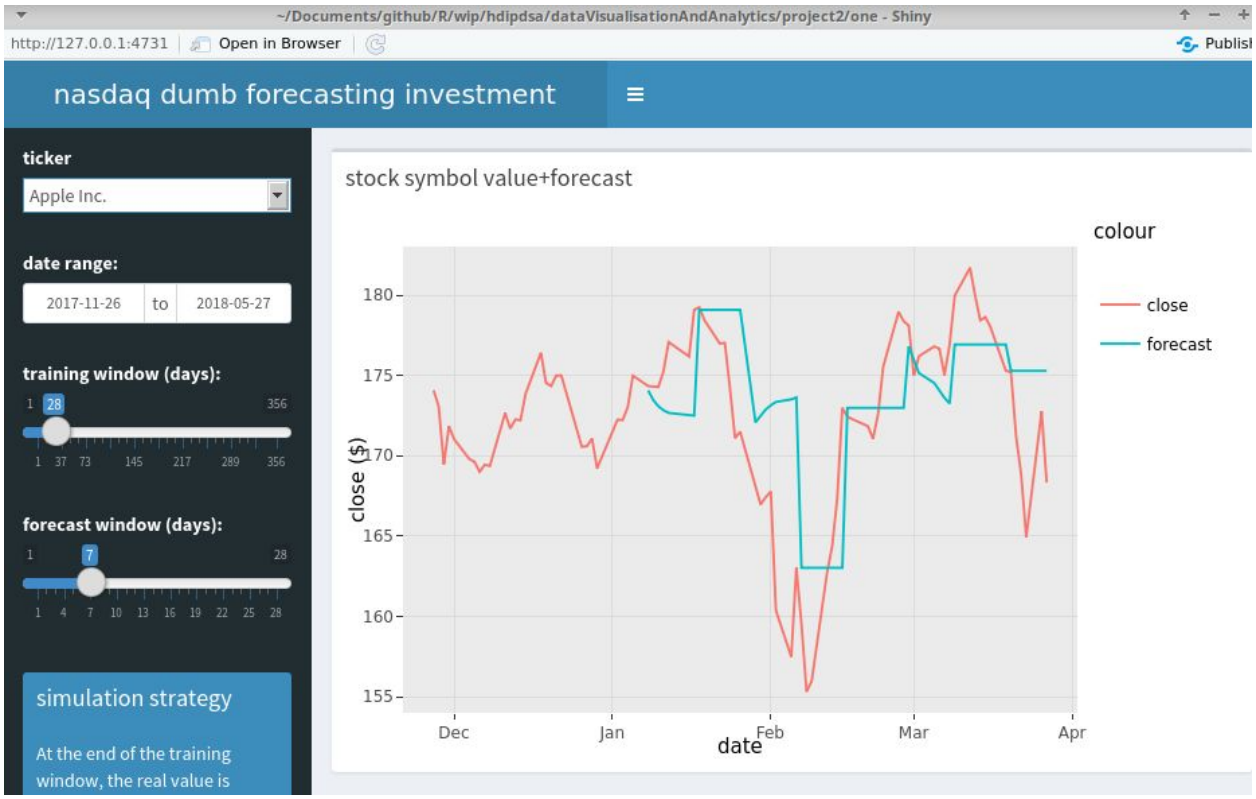

Fig 1: Shiny dashboard user interface

Also in this visualisation solution we are using the *plotly* graphing library, it provides interesting interactive features to its graphs, as for example tooltips, pan and zoom.

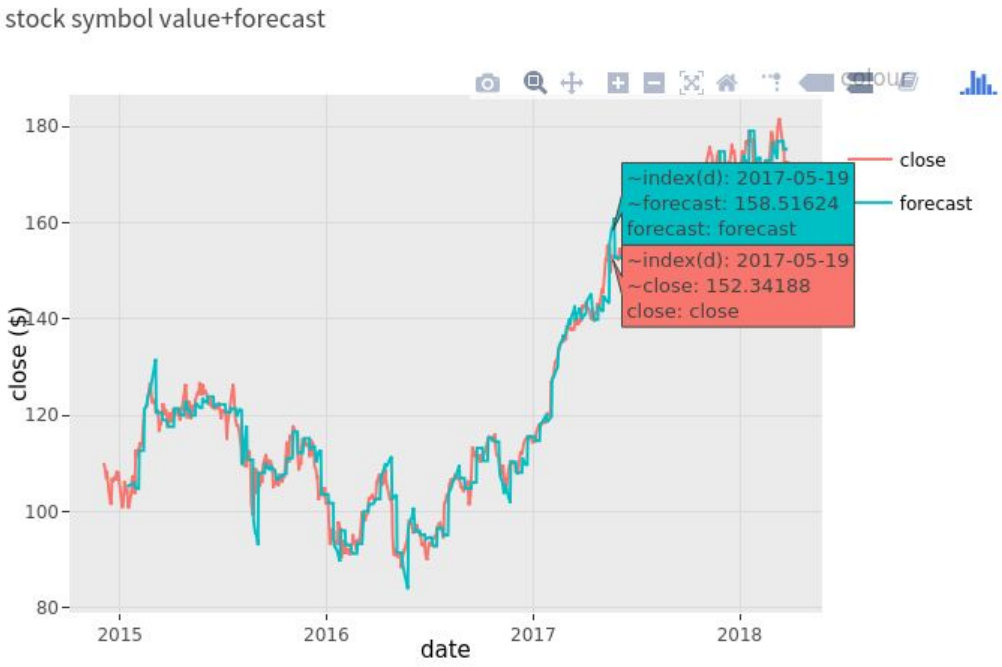

Fig 2: plotly tooltip feature

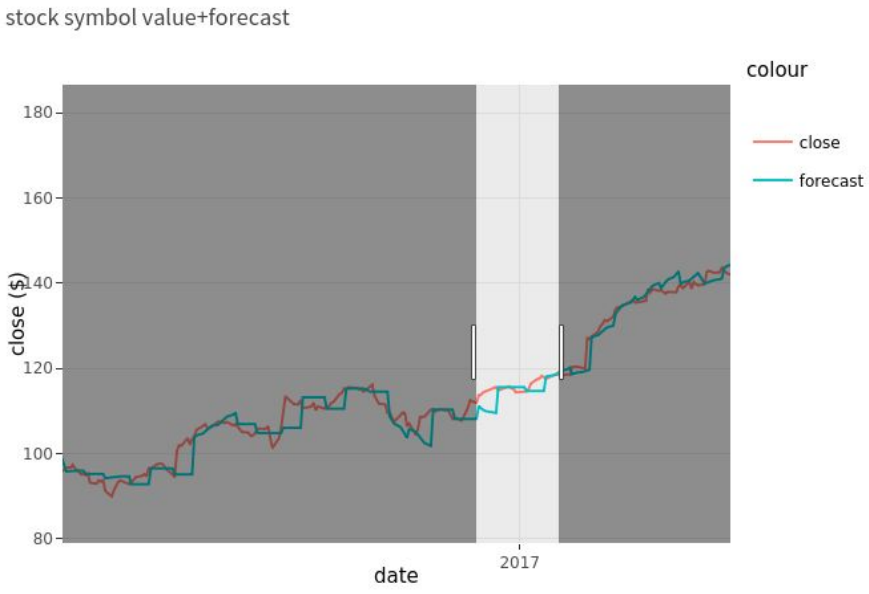

Fig 3: plotly zoom feature

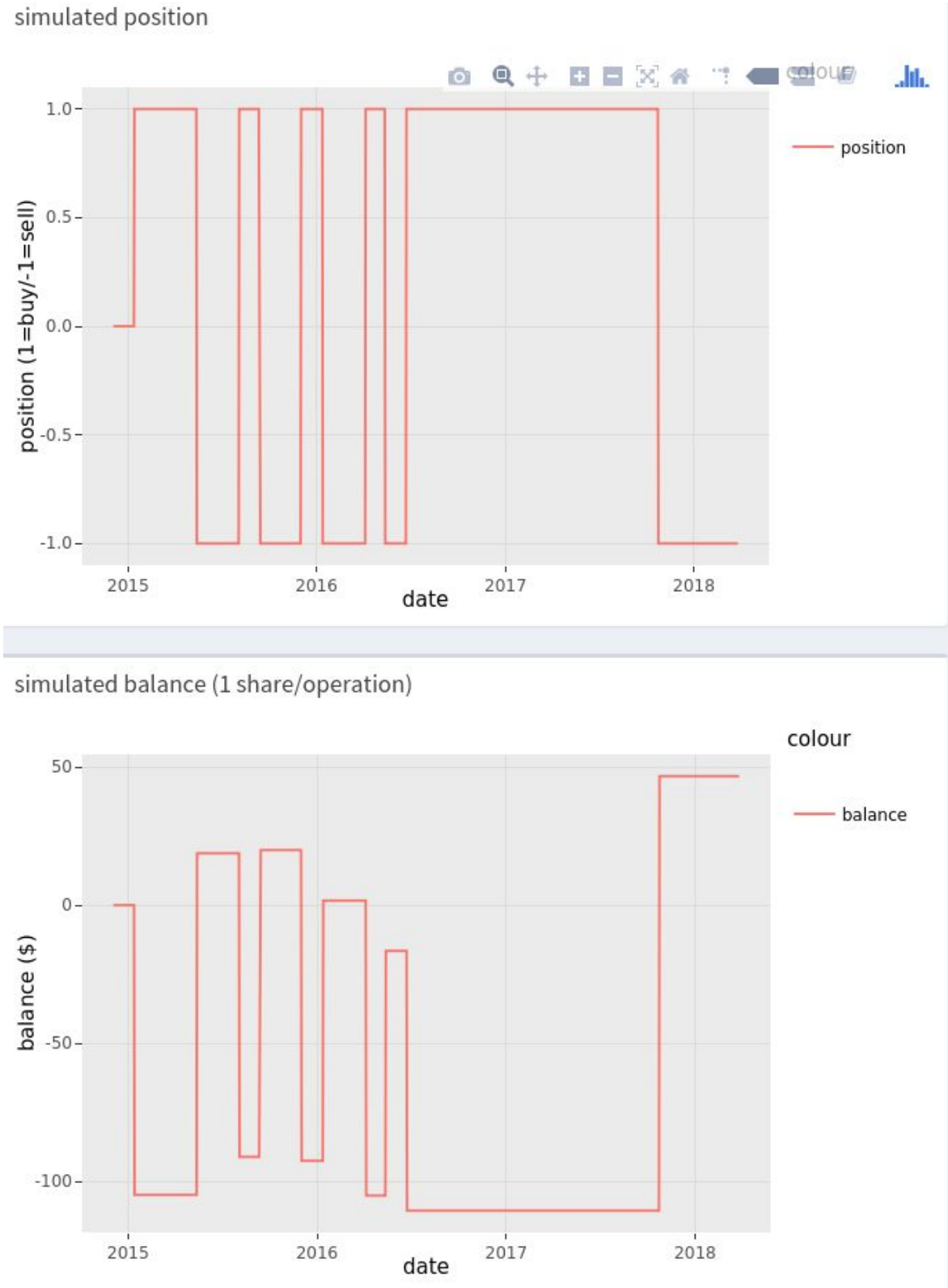

Fig 4: position and simulated balance plots

Flex dashboard

The flexdashboard package is relatively new, originally released in the end of 2017, it uses R Markdown to create a dashboard from a set of related visualisations. It supports various components, including htmlwidgets, it is responsive and allows us to specify row and column-based layouts that adapt to different device displays, and has an interesting Storyboard layout, that we will be using here. Last but not the least, has an option it can use *Shiny* to implement a dynamic user interface. The R markdown once finalized can then be processed into an html page.

As an alternative to the row and column layouts, from Shiny dashboards, *flexdashboard* storyboard, at a cost of less event based interaction, allows us to make a different kind of user interface experience, where a sequence of data visualisations aim at showcasing an experiment and its related commentary.

We've implemented hence a storyboard to describe our investment experiment. Due to the storyboard nature, we had to define the experiment variables upfront, and in this case we've decided to analyse Apple shares, since its public offering back in 1980, with a model training window of 28 days and a forecasting window of 7 days.

The storyboard uses small navigation components that are associated to specific frames that are shown once these navigation components are selected. These components can be in this case, the stages of an experiment, as different episodes of a narrative.

In the storyboard frames we can then plot the outcome of the analysis, and use any R package to do so. We've used here the recent *TSstudio* package, that provides specific visualisation tools for time series analysis and forecasting.

We've defined four components in our storyboard:

1. Static daily historical overview, plotted with *quantmod*;
2. Historical daily interactive data, plotted with *TSstudio* package;
3. Historical daily close price and forecast, using *TSstudio* package;
4. Investment position simulation and related balance using *TSstudio* package;

The storyboard layout in its flexibility allowed us to place a right positioned column to print commentary and instructions on every story frame.

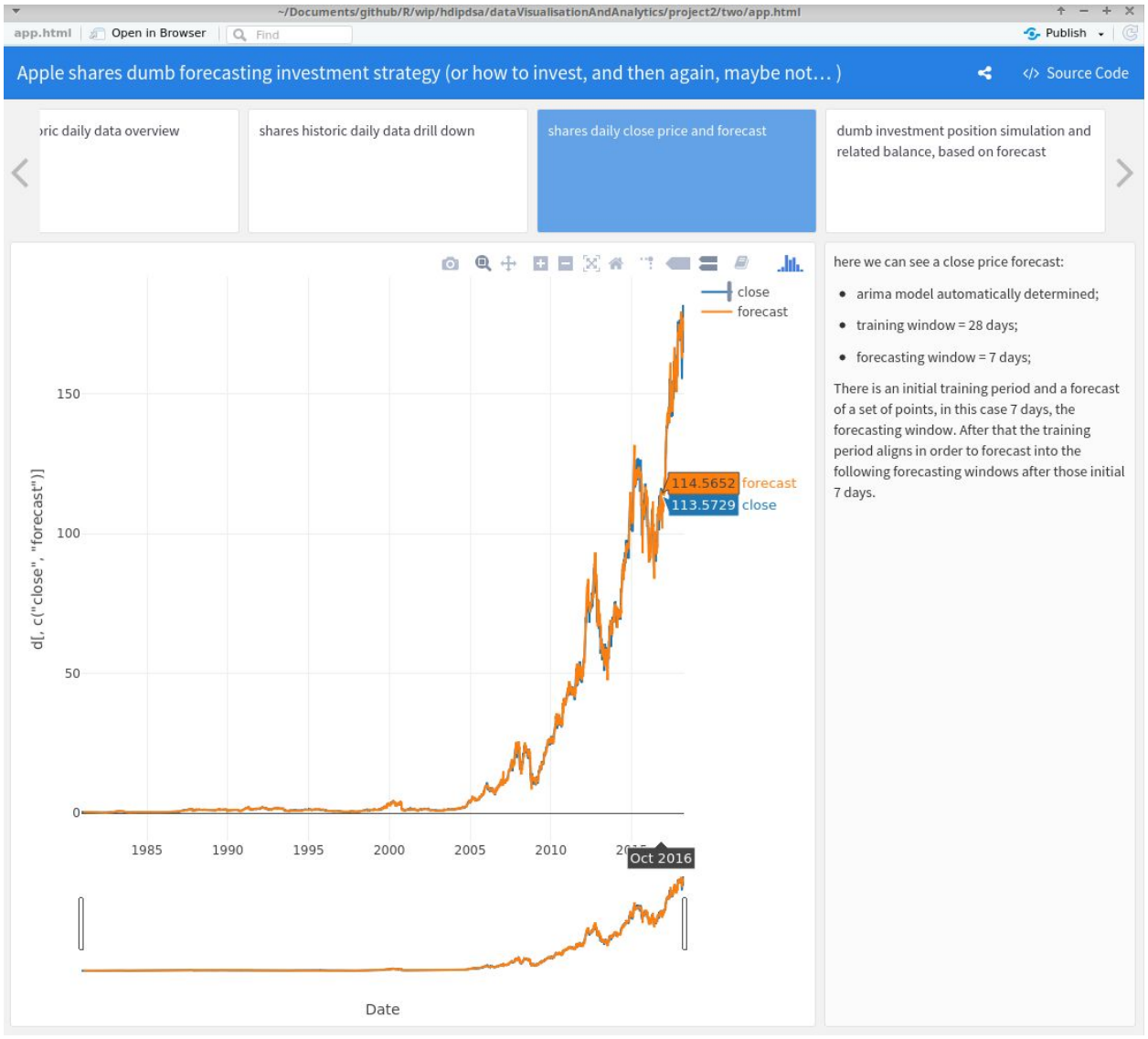

Fig 5: flexdashboard storyboard frame with TSstudio interactive graph

Conclusion

In this experiment we've tested a couple of visualisation solutions regarding a data analysis experiment on time-series datasets.

We found Shiny dashboard to be an efficient tool to quickly create an event based user interface. Its look and feel is similar to most of the web applications today, which underlines its usability. Allied with plotly and other interactive graphing packages, it provides a powerful development tool to conduct and convey data analysis information.

Flexdashboard, allows all that also, as it has the capability of embedding Shiny. The "shinyless" storyboard layout, though, is a solution for a simpler presentation-like visualisation, without the mentioned interactivity. It has a different purpose, and might be the right tool for certain analytic presentations, also providing a similar experience to the python notebooks.

Both solutions leverage feature-rich graphing packages, which is, in some sense, most of the value being provided.

References

1. Shiny dashboard
2. Quandl
3. quantmod
4. Shiny
5. R
6. Flex dashboard
7. Plotly
8. TSstudio