Università della Svizzera italiana

Prof. P. Gruber
P. Montemurro
2021-05-31

Programming II
Tuomas Ahonen, Jan Gobeli

R Package
RetCalculatoR

Hand in before:
31 May, 2021
22:00 pm

Introduction

This report encompasses the creation of a custom R package. We create several functions for common metrics in portfolio performance analysis. Additionally, to map these data and computations we include certain functions for creating plots and graphs of various sorts. On top of this, the package contains a function that allows the user to automatically generate a report including the computations and the graphics. Notwithstanding the examples provided, the reader is invited to browse the readme file and directly inspect the package's functions and their arguments in R. The full code-base can consulted on GitHub.

Functions

The functions that are included in this package can be grouped into two categories: (1) functions for financial analysis, and (2) functions for plotting graphs and tables. On the one hand, this package may be used for quick analyses of securities. Functions to compute for instance, Sortino ratios, Sharpe ratios and average returns are available. On the other hand, this package also provides functions to create both graphs and tables of the user's analysis. These financial analyses along with their tables and graphs may be automatically compiled in a report through one of the functions within the package. The reader is encouraged to browse through the different functionalities of the package through its documentation.

Package Creation

To create the package, we source the relevant R script (RetCalculatoR) from the current directory. The package.skeleton("RetCalculatoR") command creates several files and folders. The description-file and the namespace-file provide a description of the file including information regarding the authors, maintainers, as well as licenses and which functions should be visible to the public and which not, respectively. The read-and-delete-me file provides the overall steps in terms of documentation that should be added and may be discarded after the documentation is complete. In terms of the documentation, consistency is key, and a similar structure is followed throughout each function. Each function is titled and described in an unambiguous manner and the respective arguments are explained. For further information the reader may read the documentation within R, and any feedback may be directly forwarded to the maintainer. To build the package, firstly, the devtools::check() command is used in order to check for any potential errors. Even though this command does not check for the quality of the documentation, it does return an error where syntax or format is not correctly adopted. Any errors are highlighted and should be addressed. The devtools::build() command is used without any argument because it looks in the current directory. This command builds the actual package and a RetCalculatoR.0.1.tar.gz is created just outside the current folder. The next section will cover how the package is used.

Example

To see the package in action, we provide a concrete example below. Firstly, we remove and detach any previous or older versions of the package. Subsequently, the package can be installed through github and loaded into the library. Note; the user has to make a choice if already existing packages should be updated. This is recommended if the user runs RStudio with administrative rights.

```
remove.packages('RetCalculatoR')
detach("package:RetCalculatoR", unload = TRUE)
install_github("jaNGOB/RetCalculatoR")
```

Then, random data is generated...

```
dates <- seq(as.Date("2014/09/04"), by = "day", length.out = 1000)
dates <- dates[!weekdays(dates) %in% c("Samstag", "Sonntag")]
changes <- rnorm(length(dates))/100
ret <- xts::xts(returns(changes), order.by = dates)
chg <- xts::xts(changes, order.by = dates)
```

...and a simple moving crossover strategy is implemented.

```
df <- chg
colnames(df) <- 'change'
df$price <- ret

df$lma <- zoo::rollapply(df$price, 50, mean)
df$sma <- zoo::rollapply(df$price, 20, mean)
df <- na.omit(df)

df$position <- ifelse(df$lma > df$sma, -1, 1)
df$strat_return <- df$position * df$change
```

To analyse the strategy, the Sharpe ratio, Sortino ratio, total return as well as the maximal drawdown may be computed. At this stage, the reader should consult the documentation of the package in R and view all the available functions.

```
sharpe(df$strat_return)
sortino(df$strat_return)
total_return(returns(df$strat_return))
maxdrawdown(drawdown(returns(df$strat_return)))
```

```
> sharpe(df$strat_return)
[1] -0.3988244
> sortino(df$strat_return)
[1] -0.5523624
> total_return(returns(df$strat_return))
[1] -1.020129
> maxdrawdown(drawdown(returns(df$strat_return)))
[1] -0.2849959
```

Figure 1: Sharpe, Sortino, Total Returns and Maximum Drawdown from fictitious strategy.

To view plots or heatmaps, we can simply run the relevant functions to view them in R Studio.

```
ret_dd_plot(returns(df$strat_return), TRUE, df$price)
create_heatmap(df$strat_return)
```

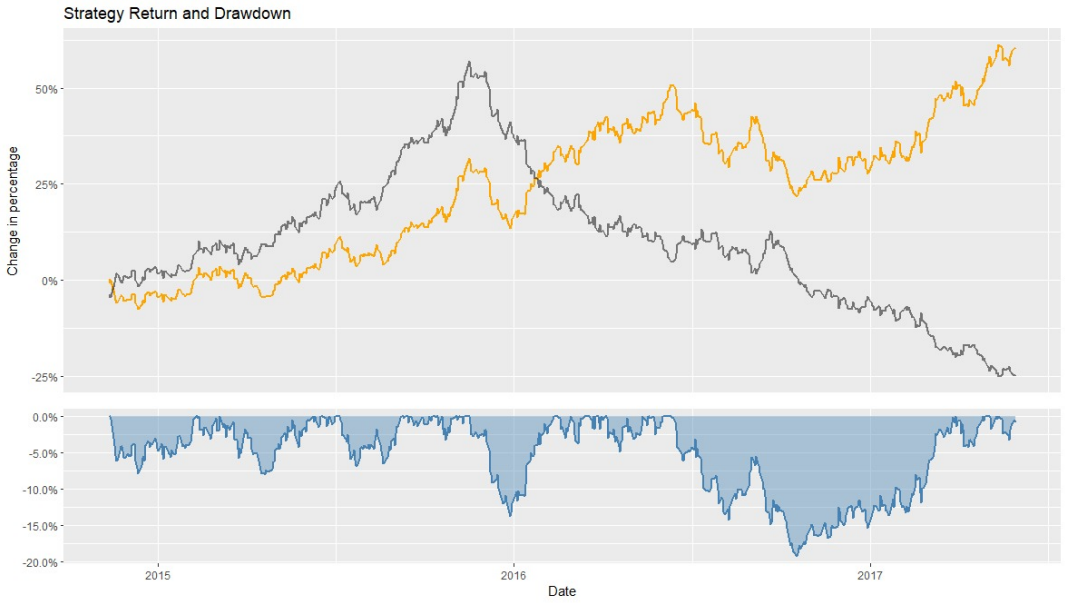

Figure 2: Strategy drawdown plots and returns.

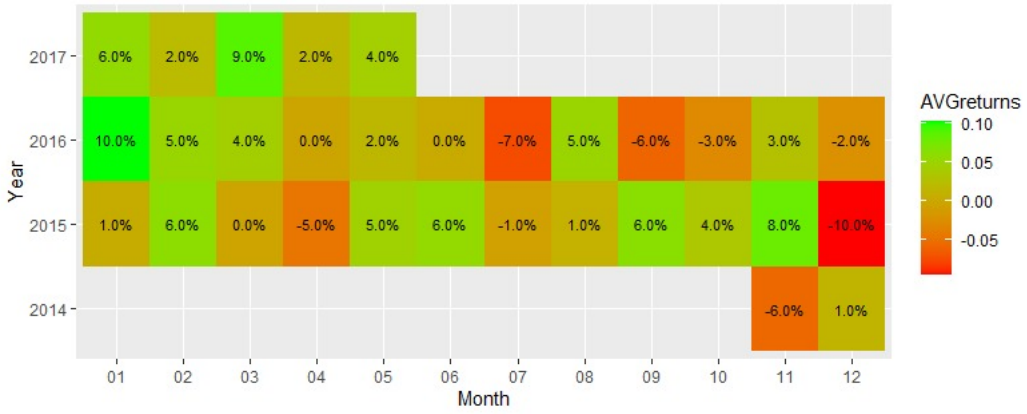

Figure 3: Heatmap of returns.

Next, the report summarizing the analytics and graphs is created.

```
generate_report(df$strat_return)
```

An html document named temp.html is created in the current working directory. An example can be found in the notebook folder on GitHub.

Conclusion

In conclusion, this report shows the various steps in creating a custom R package. Specifically, a package for portfolio performance analysis is implemented. This package includes performance metrics, graphics and automates report creation. The steps involved in package creation are covered including proper documentation of both functions and the package, highlights the role of the description and namespace file and finally covers the building and application of the package through an example.