Jia Wang

First, I found the package that call 'quantmod' which I can get several stocks backtrack data from.

It contains the function SMA to calculate whatever day moving average I need.

Additionally, the package also contains the chart plot and allow me to add custom SMA on the chart as well.

It is useful for me to plot the stock price and calculate moving average.

I would really recommend classmates to look into this package, it is really a powerful tool if you are interested in algo trading.

Karan Gupta

**My package: FNN**

I found it out when I was finishing my project about classification and I wanted to use K nearest neighbor algorithm to classify my test data set. But here I will use iris data set as an example.

FNN is Fast Nearest Neighbor Search Algorithms and Applications, you don't need to install dependencies.

The package is very useful for classification. It saved a lot of coding for me when I was trying to identify the species of the plants in my projects. Instead, one line of code using knn function can give me a performance of 98% prediction correctness.

knn function is described here: [http://stat.ethz.ch/R-manual/R-devel/library/class...](http://stat.ethz.ch/R-manual/R-devel/library/class/html/knn.html)

If we normalize iris using this:

normalize <- function(x) {

num <- x - min(x)

denom <- max(x) - min(x)

return (num/denom)

}

iris\_norm <- as.data.frame(lapply(iris[1:4], normalize))

And take a train set from iris:

set.seed(66)

ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.67, 0.33))

library(FNN)

iris.training <- iris[ind==1, 1:4]

iris.test <- iris[ind==2, 1:4]

iris.trainLabels <- iris[ind==1, 5]

iris.testLabels <- iris[ind==2, 5]

Using knn with the following code returns the list of predicted species:

iris\_pred <- knn(train = iris.training, test = iris.test, cl = iris.trainLabels, k=3)

If we want to check how many were predicted correctly, we can use:

library(gmodels)

CrossTable(x = iris.testLabels, y = iris\_pred, prop.chisq=FALSE)

Cell Contents

|-------------------------|

| N |

| N / Row Total |

| N / Col Total |

| N / Table Total |

|-------------------------|

Total Observations in Table: 40

| iris\_pred

iris.testLabels | setosa | versicolor | virginica | Row Total |

----------------|------------|------------|------------|------------|

setosa | 12 | 0 | 0 | 12 |

| 1.000 | 0.000 | 0.000 | 0.300 |

| 1.000 | 0.000 | 0.000 | |

| 0.300 | 0.000 | 0.000 | |

----------------|------------|------------|------------|------------|

versicolor | 0 | 12 | 0 | 12 |

| 0.000 | 1.000 | 0.000 | 0.300 |

| 0.000 | 0.923 | 0.000 | |

| 0.000 | 0.300 | 0.000 | |

----------------|------------|------------|------------|------------|

virginica | 0 | 1 | 15 | 16 |

| 0.000 | 0.062 | 0.938 | 0.400 |

| 0.000 | 0.077 | 1.000 | |

| 0.000 | 0.025 | 0.375 | |

----------------|------------|------------|------------|------------|

Column Total | 12 | 13 | 15 | 40 |

| 0.300 | 0.325 | 0.375 | |

Cheng Jui Tu

I find out a good package for stocks analyzation. IT is called "quantmod". When I like to do the stocks analyzation, I was trying to find relating work article, and the "quantmod" was highly recommended.

This package have very good method that can directly get stocks data from different source such as google, yahoo and store in variable .

There are many technical indicator to be used to applying into the chart.

While they all need to be write into some formula to retrieve the data.

In quantmod, it offers many technical indicator that we don't need to create again.

For example, Moving Average Convergence Divergence(MACD) is a good one.

it's function is like this :

And, in quantmod, all we need to use is addMACD()

Therefore, I think this package save people lots of time, if people like to do stock analyzation.

Paramesh Palanisamy

I would like to introduce the package called RHRV (Heart rate variability analysis ). Heart Rate Variability (HRV) describes variations over time of both instantaneous HR and the intervals between consecutive heart beats. The rhythm of the heart is modulated by the sinoatrial node (SA), which is largely influenced by both the sympathetic and parasympathetic branches of the ANS. To install this package please use the following code

>install.packages("RHRV", dependencies = TRUE)

RHRV uses a custom data structure called HRVData to store all HRV information related to the signal being analyzed. HRVData is implemented as a list object in R language. This list contains all the information corresponding to the imported signal to be analyzed, some parameters generated by the pre-processing functions and the HRV analysis results. A new HRVData structure is created using the CreateHRVData function. In order to obtain detailed information about the operations performed by the program, we can activate a verbose mode using the SetVerbose function.

> hrv.data = CreateHRVData()

> hrv.data = SetVerbose(hrv.data, TRUE )

To compute the HRV time series the BuildNIHR function can be used (Build Non Interpolated Heart Rate). This function constructs both the RR and instantaneous heart rate (HR) series. We will refer to the instantaneous heart rate (HR) as the Non Interpolated Heart Rate (niHR) series. Both series are stored in the HRVData structure.

> hrv.data = BuildNIHR(hrv.data)

# Calculating non-interpolated heart rate

#Number of beats: 17360

Please look into the attached R file for basic function usage. [RHRV.R](https://s3.amazonaws.com/itu.etpo.production/attachments/000/316/523/original/RHRV.R?1477868895)

Raffael Da Silva Nagel

Hi guys,

I would like to introduce the package TclTK

The tcltk package, developed by Peter Dalgaard, provides the tools of the TK language (created by John Ousterhout) which has already been successfully used in other languages such as Perl and Python. While still lacks most modern features, tcltk package has as a great advantage in the fact that it's come previously installed in the R Windows version, thereby avoiding a number of installation problems and compatibility. Moreover, the package has simple coding, resembling the simplicity of R, and large portability since it can be used in the operating systems Windows, Linux, and OS X.

**EXAMPLE**

library(tcltk)

window <- tktoplevel()

tktitle(window) <- "My Window"

label <- tklabel(window, text="Hello World!")

tkpack(label)

The code shown above generates a titled window "My Window", containing the label "Hello World!" (Figure). Through this example it is possible to observe the command structure, which allows access to an object window, and to it's parameters. We can also note that the objects are defined hierarchically and that tkpack() takes care of the child object placement.

The organization of the elements in a window is done be using the tkpack() function parameters: side, fill, and expand. Through these parameters we can set the position of each element.

If someone wants to know more about GUIs in R, please send me an email (raffael.nagel@gmail.com)

Bo Wang

Hello Team,

I would like to introduce a special package named "SixSigma".

As you can see from the name it is a quality engineering specific tool. I have found it when I was searching for Six Sigma methods that I can possibly introduce to my production lines and surprisingly found that there is a package here. This package has functions that can conduct Gage R&R, yield analysis and a variety of quality and process specific functions, so although it may or may not make coding itself easier, it is a great tool to help with getting results quickly and accurately.

This package is built upon "qcc", "nortest" and "e1071" packages.

One example that is handy and easy is the ss.ca.yield() function which calculates yield data. The snippet is shown below:

ss.ca.yield(defects = 0, rework = 0, opportunities = 1)

During usage, change quantities of defects, reworks and opportunities (batch size or total count of parts produced), then the function will return the following data:

Yield FTY RTY DPU DPMO

Yield is count of good part produced divided by total count of parts, this denotes how well a production line is capable of producing good parts, it is also a pointer of if quality has to have someone inspecting parts or the line can manage itself;

FTY(First Time Yield), also called First Pass Yield, calculates only yield without rework, this is an important factor of how well a production line is capable to run by its own (normally rework will be conducted in NCM instead of line side so extra work will be introduced when rework happens).

RTY is a factor similar to FTY, but in a multiple station environment.

DPU (Defect Per Unit) distributes defect to each unit, which give quality an idea of roughly what should be expected.

DPM: Defect Per Million, this is an important factor to judge the sigma level.

Link of manual can be found here: [https://cran.r-project.org/web/packages/SixSigma/S...](https://cran.r-project.org/web/packages/SixSigma/SixSigma.pdf)

**Wei He**

The R Package that I want to introduce is Package “stringr”

install.packages(“stringr”)

library(stringr)

I find out this package when I try to edit some strings in R.

This package is very useful because it can provide a clean, modern interface to common

string operations. Also when you deal with regular expressions, a concise language for describing patterns of text expression , you might also need use this package

Example:

"[aeiou]"matches any single character that is a vowel:

str\_subset(x, "[aeiou]")

#> [1] "video" "cross" "extra" "deal" "authority"

str\_count(x, "[aeiou]")

#> [1] 0 3 1 2 2 4

In addition, “stringr” Package didn’t have any dependencies on any other package.

There are seven main function that work with patterns:

str\_detect(x, pattern) tells us if there's any match to the pattern.

Example:

str\_detect(x, "[aeiou]")

#> [1] FALSE TRUE TRUE TRUE TRUE TRUE

str\_count(x, pattern) counts the number of patterns.

Example:

str\_count(x, "[aeiou]")

#> [1] 0 3 1 2 2 4

str\_subset(x, pattern) extracts the matching components.

Example:

str\_subset(x, "[aeiou]")

#> [1] "video" "cross" "extra" "deal" "authority"

str\_locate(x, pattern) gives the position of the match.

Example:

str\_locate(x, "[aeiou]")

#> start end

#> [1,] NA NA

#> [2,] 2 2

#> [3,] 3 3

#> [4,] 1 1

#> [5,] 2 2

#> [6,] 1 1

str\_extract(x, pattern) extracts the text of the match.

Example:

str\_extract(x, "[aeiou]")

#> [1] NA "i" "o" "e" "e" "a"

str\_match(x, pattern) extracts parts of the match defined by parentheses.

Example:

str\_match(x, "(.)[aeiou](.)")

#> [,1] [,2] [,3]

#> [1,] NA NA NA

#> [2,] "vid" "v" "d"

#> [3,] "ros" "r" "s"

#> [4,] NA NA NA

#> [5,] "dea" "d" "a"

#> [6,] "aut" "a" "t"

str\_replace(x, pattern, replacemnt) replaces the matches with new text.

Example:

str\_replace(x, "[aeiou]", "?")

#> [1] "why" "v?deo" "cr?ss" "?xtra" "d?al" "?uthority"

str\_split(x, pattern) splits up a string into multiple pieces.

Example:

str\_split(c("a,b", "c,d,e"), ",")

#> [[1]]

#> [1] "a" "b"

#>

#> [[2]]

#> [1] "c" "d" "e"

As well as regular expressions (the default), there are three other pattern matching engines:

fixed(): match exact bytes

coll(): match human letters

boundary(): match boundaries

In general, stringr provides an opinionated interface to strings in R. It makes string processing simpler by removing uncommon options, and by vigorously enforcing consistency across functions.

Source:

<https://cran.r-project.org/web/packages/stringi/stringi.pdf>

https://journal.r-project.org/archive/2010-2/RJournal\_2010-2\_Wickham.pdf

Sarah Huangting Jauw Gunawan

R package rgl is a data visualization using three-dimensional presentation for constructing geometric outputs. One such example of the use of the package is shown below.

rgl.pop()  
# nicer colored plot  
ylim <- range(calls)  
ylen <- ylim[2] - ylim[1] + 1  
col <- topo.colors(ylen)[ calls-ylim[1]+1 ]  
x = (1: nrow(calls))  
z = (1: ncol(calls))

[rgl.bg](http://rgl.bg)(sphere=FALSE, color=c("black"), lit=FALSE)  
rgl.viewpoint( theta = 300, phi = 30, fov = 170, zoom = 0.03)  
rgl.surface(x, z, calls, color = col, shininess = 10)  
rgl.bringtotop()

or

plot3d(x=X, y=Y, z=Z, type="p", col="red", xlab="X", ylab="Y", zlab="Z",

size=5, lwd=15, box=F)

Additionally, a function in this package allows one to plot an ellipsoid with a mesh3d function that outlines a confidence region for 3 parameters and allows one to control the number of subdivisions used in constructing the ellipsoid.

data(mtcars)

fit <- lm(mpg ~ disp + cyl , mtcars)

open3d()

plot3d(ellipse3d(fit, level = 0.90), col = "blue", alpha = 0.5, aspect = TRUE)

https://cran.r-project.org/web/packages/rgl/rgl.pdf

Sarah Huangting Jauw Gunawan

The Spatstat package is a package in R that analyzes two dimension data with spatial point pattern that enables one to manipulate and explore inhomogenous point patterns and spatial sampling regions that fits point process models to point pattern data (Baddeley, A. & Turner, R., 2005). Spatstat fits “parametric models to special point pattern data”, allowing data to be plotted, analyzed and transformed using geometrical transformations, operations and measurements (Baddeley, A. & Turner, R., 2005). Functions in the package include the empty space function, the nearest neighbor distance function, pair correlation, second moment measure, cross-K function, cross-G function, mark correlation function etc (Baddeley, A. & Turner, R., 2005). A demonstration is available with demo(spatstat) in R with fitted models using point matterns.

Baddeley, A. & Turner, R., 2005. Journal of Statistical Software 12 (6).

**Rui Huang**

I found a useful package for data analysis is "psych":

This package has functions for analyzing data at multiple levels include within and between group statistics, including correlations and factor analysis.

I can use the describe and describeBy functions to summarize data sets.

For example:

> describe(mtcars)

This function provides the ones most useful for scale construction and item analysis in classic psychometrics. In mtcars dataset , it shows vars, n, mean, sd, median, trimmed, mad, min, max, range, skew, kurtosis, se for each variables.

> describeBy(mtcars,group = mtcars$mpg)

This function reports basic summary statistics by a grouping variable. Useful if the grouping variable is some experimental variable and data are to be aggregated for plotting.

For more information: [https://cran.r-project.org/web/packages/psych/vign...](https://cran.r-project.org/web/packages/psych/vignettes/overview.pdf)

Aleksandar Milenov Gyorev

Hi,

A useful R package for image manipulation is "imager":

install.packages("imager")

To get started with it is pretty easy:

library(imager)   
plot(boats)

You can create an image grayscale, by using the grayscale function:

grayscale(boats)

To get a simple histogram of the grayscale values for the built-in boat photo you can do:

grayscale(boats) %>% hist(main="Luminance values in boats picture")

It is a great package to use if you want to do something with image recognition for your final project and I highly recommend it!

More info: <http://dahtah.github.io/imager/gettingstarted.html>