https://doi.org/10.48009/3_iis_2019_57-63

Issues in Information Systems
Volume 20, Issue 3, pp. 57-63, 2019

COMMAND-BASED CODING USING R FOR DATA SCIENCE

Jianfeng Wang, Indiana University of Pennsylvania, jwang@iup.edu
Linwu Gu, Indiana University of Pennsylvania, lgu@iup.edu

ABSTRACT

In this paper we discuss introductory R basics for data science and present teaching cases of analyzing iris dataset using 5 different algorithms from 6 different R packages, each package carrying some main functions implementing some algorithms. R coding for data science can be taught as command -based coding. Those functions and syntaxes are easy to follow and use. Our R script in the paper is a contribution to the teaching community as many books for data science are written in a way that are still a little bit hard for business students to follow. At the end of the discussion, we provide our recipe of teaching data science using R.

Keywords: Data Science, k-Nearest Neighboring, K-means, Decision Tree, Support Vector Machine, Neural Network

INTRODUCTION

The community of teaching data analysis has been concerned with students' weak backgrounds in math and statistics or programming. Some colleagues have complained that they have trouble teaching any algorithms for data science. Many colleagues just use Excel to teach data science. For years of teaching data analysis and data mining classes to MBA and undergraduate students, we find that R is so user friendly that introductory data science in R can be taught as command-based coding, which makes the data science classes easy to follow and interesting to learn. Many articles and books have discussed in detail a large set of numeric and statistics functions available in R at the base installation of R console (Wang & Gu, 2016, 2018; Matloff, 2011; Lantz 2013; Shmueli et al. 2018; Teeter 2011; Baumer 2015). We are not going to discuss these in this paper. We are going to introduce how to teach data science in R using command-based coding.

WHAT IS COMMAND-BASED CODING?

Command-based coding is a mixture of coding with commands available, not so easy as using Excel functions or commands but not so challenging as real coding in Java or C++. In Excel data analysis, we can use many functions. If we know the basic syntaxes and arguments for a function, we can just type the function in a cell and see the result. Very little coding is needed to use such Excel functions or commands. In data retrieval from a database server using SQL, many commands are used. But such SQL commands cannot be used in a simple way. Users must follow some simple syntaxes and logics. SQL offers limited sets of commands for data definition, data manipulation, data retrieval, database administration and metadata management such as "Select, From, Where, Having, Group by, Order by, NOT, IN, Exist, Alter, Create" etc. Control structure and Loops can also be part of a SQL statement. What we do in an introductory database class using SQL is a good example of command-based coding. We have taught database classes for many years. We find that students having trouble in java or c++ coding classes often have no trouble working with SQL. We tried to copy the approach in teaching SQL to teach our introductory course Data Science for Business using R. The result is very encouraging. The number of undergraduate students enrolled in our data science class increases from 3 or 4 a few years ago to 20 in the academic year 2018-2019, and certainly more MBA students enrolled, from 9 or so a few years ago to 29 MBA students in the academic year 2018-2019.

ESSENTIAL R BASICS FOR DATA SCIENCE

There are more than 11000 downloadable packages in R according to www.r-project.org. It is impossible to count how many functions available in R with these 11000 packages. It is impossible yet to teach all these packages or functions available in an introductory data science class. We select some packages with popular algorithms for our students. Popular algorithms covered in our classes include: K-nearest neighboring, K-means, Naive Bayes, Decision tree prediction and classification, Neural network, and Support vector machine.

First few weeks of our classes are planned for introducing indexing of R variables. Basic variable types in R include vector, matrix, list and data frame. The basic unit is vector, and more complicated one is data frame. They all have similar indexing syntaxes. Data frame inherits all the properties from list and matrix, list and matrix from vector. A dataset to be analyzed usually can be read into R session as a data frame. We spend lots of time making students familiar with how to index rows and columns in a data frame and how to refer to rows or columns using column or row indexing. In data mining, we usually split a data set into subsets, some subsets will be used as training subsets. Other subsets will be used as test subsets. It is also important to note that the rows in a data set should not be well grouped using a target column but should be randomly ordered.

CODE EXAMPLES USING IRIS DATASET

Iris data set is a common data set people use for introduction in data science or data mining. The data set comes with R console base installation. When we start R, the small data set is in the memory with a current R session. The original data set is well grouped along the target column, "Species", with the first 50 rows for "Setosa", second 50 rows for "Versicolor", and third 50 rows for "Virginica". We are going to use this data set as an example.

To randomize the rows in the dataset, we retrieve the order from randomly generated numbers in uniform distribution to rearrange the dataset rows such that these rows are not grouped along the target variable column. The functions used in this exercise are simple. nrow() to get the number of rows in the dataset. order() lists ordinal number of each random number in runif(nrow(iris)). To verify the results we can use head() or tail().

```
set.seed(9850)
 runif(nrow(iris))
 gp<-runif(nrow(iris))
 iris2<-iris[order(gp),]
```

Once the dataset rows are randomly reordered, next thing to do is to create subsets of data for model training and testing. If values in a numeric column are big, we can normalize these values by using a normalize function, "normalize<-function(x) {return((x-min(x))/(max(x)-min(x)))}". It is a commonly used function to normalize numeric values (Lantz 2013; Shmueli et al. 2018). But doing so will add another layer of challenge for students. We don't recommend this practice in an introductory class for business students. There is no big difference in training performance.

To create subsets of data, students should first learn how to index groups of rows and columns from a dataframe.

```
iris_train<-iris2[1:129,-5]
 iris_test<-iris2[130:150,-5]
 iris_train_target<-iris2[1:129,5]
 iris_test_target<-iris2[130:150,5]
```

These 4 lines will create 4 subsets of data for training and testing K-nearest neighboring algorithm using class package function knn(). Negative index is to exclude a column or a row in a subset. In this example, iris_train will not have 5th column from iris2. The 5th column is the target variable, "Species". We can also use column headings or variable names to index columns. There are lots of discussions in many R introductory books. Here is a quick example using

mtcars dataset. Mtcars is a dataset available with R console base installation. Use names() to display column headings for a dataset.

```
> names(mtcars)
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"
"gear"
[11] "carb"
```

To create a subset with columns “mpg”, “cyl”, “hp”, and “wt”, we can use a list of alternative commands,

```
> mtcarssub<-mtcars[,c("mpg","cyl","hp","wt")]
> mtcarssub2<-with(mtcars, data.frame("mpg","cyl","hp","wt"))
> mtcarssub3<-
data.frame(mtcars$mpg,mtcars$cyl,mtcars$hp,mtcars$wt)
```

It is very important to explain the basic syntaxes of dataframe or list indexing in R. Syntax is similar for python data frame or dictionary. So, if a student can learn R dataframe solidly, he can learn python dictionary and dataframe indexing quickly. If students can grasp dataframe indexing, they should have no big hurdle learning algorithmic functions. Otherwise they will have big trouble in understanding data mining functions and algorithms. For many years of using R teaching data mining, we have found dataframe indexing skills to create subsets of data very important for the whole course learning. Students, whether quick or slow in learning, all can be taught effectively how to index columns and rows in a dataframe and to create subsets of data for model training and testing. If one session is not enough, instructors can schedule two or three sessions.

Once data subsets for training and testing are created, we can try to train and test a model. Different models may require different ways of splitting a dataset into subsets. With subsets of iris data available as described above, we can simply train and test k-nearest neighboring algorithm, available at the package “class”. First make sure required R packages are available in your installation by checking the installed package list. If not, required packages should be downloaded and installed. R studio allows a user to easily check installed R packages in your computer. If an R package is available in the computer, we need to load the class library to the R session by using the command library(class) or require(class).

```
require(class)

m1<-knn(train=iris_train,test=iris_test,cl=iris_train_target,k=13)
```

The function knn() takes 4 arguments, subsets of data for training, testing, target variable values with the training subset, and number of nearest neighbors, k. K-value is set at 13 for this example. The k value is usually set around the square root of the sample size (Lantz 2013). Knn() is nice to introduce to students as a relatively simple algorithm for data mining. This one function trains the model and then predict target values for test data. M1 is the result of prediction using test data. We can use table() function to check the effectiveness or accuracy of prediction. Since we already have this subset of target values for test dataset, table() function will compare these two lists to check for accuracy.

```
> table(iris_test_target,m1)
                    m1
iris_test_target  setosa  versicolor  virginica
    setosa           6          0          0
    versicolor       0          4          1
    virginica        0          1          9
```

Altogether we just need about 10 lines of code or fewer to analyze the iris data set and make a prediction. If the data set is much bigger, we can do the same. Only a few functions are used. Both syntaxes and arguments are simple

enough to handle by beginners. The only challenge here is for students to understand row and column indexing in a dataframe so that they know how to create subsets of data for training and testing and prediction.

```
set.seed(9850)
runif(nrow(iris))
gp<-runif(nrow(iris))
iris2<-iris[order(gp),]
iris_train<-iris2[1:129,-5]
iris_test<-iris2[130:150,-5]
iris_train_target<-iris2[1:129,5]
iris_test_target<-iris2[130:150,5]
require(class)
m1<-knn(train=iris_train,test=iris_test,cl=iris_train_target,k=13)
table(iris_test_target,m1)
```

K-nearest neighboring algorithm is a bit unique in that knn() function incorporates both model, training, testing and prediction in one main function. The table() function result shows that 1 versicolor specie is predicted by the model to be virginica; a virginica specie to be a versicolor. The rest of the prediction matches the original values in the iris_test_target. The accuracy is 90%.

Next let's use decision tree analyze the iris data set. First look at the code below (Table 1).

After randomly reordering the rows of data in the dataset, we split the dataset. Data split is a little bit different from that of k-nearest neighboring algorithm. Then load the functions from C50 library into the R session by calling library(C50). C50 available in R is a non-commercial version. Once C50 library is loaded, the main function C5.0() is called with two arguments: iris training data without target column, and the target column of the training dataset. The C50 model is trained with the training data and tested with test data using predict(). The predict() carries two arguments, the trained model and the test data. The prediction accuracy is about 90%. 2 versicolor species are mistakenly predicted as virginica.

Table 1. Decision tree using C5.0

```
set.seed(9850)
gp<-runif(nrow(iris))
iris2<-iris[order(gp),]
head(iris2)
iris_train<-iris2[1:129,]
iris_test<-iris2[130:150,]
iris_train_target<-iris2[1:129,5]
iris_test_target<-iris2[130:150,5]
library(C50)
iris_model <- C5.0(iris_train[,-5],iris_train$Species)
iris_pred <- predict(iris_model, iris_test)
table(iris_pred,iris_test_target)
summary(iris_model)
```

```
> table(iris_pred,iris_test_target)
                iris_test_target
iris_pred    setosa    versicolor    virginica
  setosa          5             0            0
  versicolor      0             5            0
  virginica       0             2            9
```

The summary () function result shows a decision tree the C5.0() function builds.

```
> summary(iris_model)

Call:
C5.0.default(x = iris_train[, -5], y = iris_train$Species)


C5.0 [Release 2.07 GPL Edition]        Fri Apr 26 09:39:19
2019
-------------------------------

Class specified by attribute `outcome'

Read 129 cases (5 attributes) from undefined.data

Decision tree:

Petal.Length <= 1.9: setosa (43)
Petal.Length > 1.9:
:...Petal.Length > 4.9: virginica (36)
    Petal.Length <= 4.9:
    :...Petal.Width <= 1.6: versicolor (44)
        Petal.Width > 1.6: virginica (6/1)

Evaluation on training data (129 cases):

          Decision Tree
        ----------------
        Size      Errors

          4    1( 0.8%)   <<

         (a)   (b)   (c)    <-classified as
        ----  ----  ----
         43                 (a): class setosa
              44     1      (b): class versicolor
                    41      (c): class virginica

       Attribute usage:
       100.00%  Petal.Length
        38.76%  Petal.Width
```

Decision tree analysis is also available in Rweka package. Rweka package provides two functions for decision tree prediction and classification. The simple one is OneR(). Second function is JRip(). Both functions are case sensitive. If you use lower case to call these functions, you will simply get error messages.

```
library(RWeka)
iris_model2<-OneR(Species~.,data=iris2)
iris_pred2<-predict(iris_model2, iris_test)
table(iris_pred2,iris_test_target)
```

Both JRip and OneR provide better prediction than C5.0. We are not going to show the results here. It is very easy to execute the code block in RStudio. Overall syntax is easy to follow. The arguments needed for these functions are a little bit different. Both OneR and JRip use "~" in the formula, similar to the linear regression function lm().

In code blocks table below, we analyze the same iris dataset using support vector machine, neural network and k-means. Classifiers we use in the code work very well. Again, as we can see, the code is not complex at all.

Table 2. SVM, K-Means, & Neural Network

Code Block	**Prediction Accuracy**

Support Vector Machine
 	
`set.seed(9850)`
`gp<-runif(nrow(iris))`
`iris2<-iris[order(gp),]`
`head(iris2)`
`iris_train<-iris2[1:129, ]`
`iris_test<-iris2[130:150, ]`
`library(e1071)`
`iris_svm <- svm(Species ~., data = iris_train)`
`predsvm<- predict(iris_svm, iris_test)`
	`> table( Predicted=predsvm, Actual=iris_test$Species)`
	Actual
	Predicted setosa versicolor virginica
	setosa 4 0 0
	versicolor 0 8 0
	virginica 0 1 8

K-means
 	
`set.seed(9850)`
`gp<-runif(nrow(iris))`
`iris2<-iris[order(gp), ]`
`iriskm<-kmeans(iris2[,3:4], 3)`
`table(iriskm$cluster, iris2[, 5])`
	`> table(Preds=iriskm$cluster, Actual= iris2[, 5])`
	Actual
	Preds setosa versicolor virginica
	1 0 2 46
	2 0 48 4
	3 50 0 0

Neural Network
 	
`library(nnet)`
`set.seed(9850)`
`gp<-runif(nrow(iris))`
`iris2<-iris[order(gp),]`
`iris_train<-iris2[1:129,]`
`iris_test<-iris2[130:150,]`
`mynn <- nnet(Species ~ ., data=iris_train, size=2, maxit=50)`
`preds <- predict(mynn, iris_test, type="class")`
	`> table(Predicted=preds, Actual=iris_test$Species)`
	Actual
	Predicted setosa versicolor virginica
	setosa 7 0 0
	versicolor 0 4 0
	virginica 0 1 9

We can compare these different code blocks programmed with k-nearest neighboring, decision tree analysis, support vector machine, K-means, and neural network. We can see very similar syntaxes, easy to follow and copy. predict() is highly reusable with almost same arguments such as trained model name and test data. That's why we call data analysis coding in R is command-based. This is especially true for beginners. Here is our recipe of teaching data science using R.

1) First teach students how to download and install R console and R-Studio. Introduce features and advantages of using RStudio. RStudio editor is very useful for editing r script. R file directory is easy to use. It is necessary to introduce students how to set up working directory, which is very basic for file input and data processing and analysis. Not all R tutorials and books introduce these basic yet very important knowledge. File input methods should be introduced, very easy to use.
2) A couple of weeks, about R variables, control structure, and basic syntaxes, such as vector, matrix, list, dataframe, for loop and if-else structure. Some commonly used functions in data preprocessing such as factor(), table(), lapply(), and tapply(), etc. Syntaxes about how to create subsets of a dataset using row or column indexing of data frame.
3) It usually helps students learn if similar functions can be demoed through Excel.
4) Once these are introduced and if most students can understand and manipulate data frame, instructors can move on to introduce data mining algorithms and functions from R packages.

5) An instructor can start with K-means and K-nearest neighboring, followed by support vector machine, decision tree, neural network, and others if time allows in a semester
6) It is very important to introduce concepts, theories, backgrounds and use cases about these algorithms, basic designs and steps of such algorithms. To introduce neural network concepts, we use some videos published at www.youtube.com. Drawing synapses, neurons and their responsive connections and neural network is not easy. So, we play some videos with animated neural network illustration, which help a lot for students to understand.

CONCLUSIONS

We find the data analysis using R can be taught as command-based coding with easy-to-remember functions and easy-to-follow syntaxes. For some commonly used packages and algorithms in data science, R Console, RStudio and R packages are great tools to start with in learning these algorithms. We develop an R script based on Iris dataset using different algorithms including k-nearest neighboring, k-means, support vector machine, decision tree, and neural network. Most textbooks use different data sets for different algorithms, which is not easy for students to capture essentials about R packages for data science. Our script is a contribution to the teaching community as most textbooks available now lack such a concise script for students to learn in such a comparative way. When different algorithms performed on the analysis of a same data set and code blocks are there for students to compare with on how to split a dataset, how to enter arguments for model functions, and how to predict, they find the learning R very interesting and willing to spend lots of time scripting. We find our recipe for teaching data science using R very helpful for introductory data science courses.

REFERENCES

Baumer, B, (2015). A Data Science Course for Undergraduates: Thinking with Data. *American Statistician, 69*(4), 334-342

Chang, W. (2012). *R Graphics Cookbook*. O'Reilly, Sebastopol, CA.

Lantz, B. (2013). Machine Learning with R. Packt Publishing, Birmingham-Mumbai

Matloff, N. (2011) The Art of R Programming, No Starch Press, San Francisco, CA.

Shmueli, G., Bruce, P., Yahav, I., Patel, N., & Lichtendahl, K. (2018). *Data Mining for Business Analytics: Concepts, Techniques, and Applications in R.* John Wiley& Sin Inc.

Teetor, P. (2011). *R Cookbook*, Sebastopol, CA: O'Reillly,

Wang, J. & Gu, L. (2016). Challenges of Teaching Data Science in a Business School. *Issues in Information Systems*, *17*(3), 209-217.

Wang, J. & Gu, L. (2018). Teach MBA Data Science Using R. *Issues in Information Systems*, *19*(2), 65-71.

Yap A., & Drye S. (2018). The Challenges of Teaching Business Analytics: Finding Real Big Data for Business Students. *Information System Education Journal*, *16*(1), 41-50.