THE LONDON SCHOOL OF ECONOMICS AND POLITICAL SCIENCE

MODULE 1 UNIT 2
Notes Video 1 Part 1 Transcript

IN COLLABORATION WITH getsmarter™

Module 1 Unit 2 Notes Video 1 Part 1 Transcript

Importing and cleaning data in R

KOSTAS KALOGEROPOULOS: In this unit, the importance of data in machine learning is highlighted. No matter how sophisticated your machine learning processes are, if the underlying data is flawed, your results will be flawed as well. In Part 1 of this video, you will first learn the basics of creating variables in R, the process of importing packages and data, and how the data is cleaned before it is visualised and transformed to become useful for decision-making.

Section A: Creating basic variables

KALOGEROPOULOS: Firstly, let's take a closer look at the different types of data you will encounter during your learning journey. The first is a scalar represented in the first cell. This type of data consists of only one value. The second code cell shows a vector, which is a set of multiple scalar values combined into one data set. The third type of data is a matrix, a basic form of which is represented in the third code cell. The matrix is a table of values. Both a matrix and a vector allow for matrix operations. Another way of creating a matrix is by binding two vector columns together. This can be done by making use of the "cbind" function.

The fourth type of data is the data frame. The data frame is like a matrix, but it allows for different classes of data to be used. To illustrate this, a matrix is created by binding two vectors together, one of which is a vector of text, such as "Hello", "Goodbye", and "Welcome" shown here.

```
In [ ]: # Create matrix of mixed data
        d <- c(1, 4, 8)
        e <- c("Hello", "Goodbye", "Welcome")
        f <- cbind(d, e)
        f
```

As you can see in the output, this makes the matrix f a character matrix. In other words, the numbers have been converted to text. This is because matrices always have to have the same class of data throughout. In this case, text. The data frame allows each column to be a different class of data.

```
In [5]: # Create matrix of mixed data
d <- c(1, 4, 8)
e <- c("Hello", "Goodbye", "Welcome")
f <- cbind(d, e)
f
```

A matrix: 3 × 2
of type chr

d	e
1	Hello
4	Goodbye
8	Welcome

This is highly useful when we are working with mixed data. It is also possible to call a specific column of the data frame by using the data frame's name. In this case, the dollar sign followed by the column name. If we use f$d, you can see that the vector extracted is a numeric value, while f$e provides the character vector.

```
In [7]: # Choose specific data frame column
f$d
```

1 · 4 · 8

```
In [8]: # Choose another data frame column
f$e
```

Hello · Goodbye · Welcome
▶ **Levels:**

Section B: Loading packages into R

KALOGEROPOULOS: In the first code cell in this section, you can see how different packages are installed. After each of these packages are installed, they are then loaded into the notebook. The data.table package allows for the importing of data. The tidyverse package is a set of packages used for easy data manipulation. The DT package allows for the visualization of data tables. The nanier package handless missing values. The dplyr package is used for data processing and analysis.

Section C: Importing data into R

KALOGEROPOULOS: In the first line of this first cell of this section, the data is red. The data set is called auto. The next few lines of code in this cell assign column names to the auto data set, including "id", "attorney", "gender", "marital", "insured", "seatbelt", "age", and "loss". Finally, the "head" function is used to present the first few lines of the data sets.

An in-depth discussion of all the available packages falls outside the scope of this course; however, it can be useful to be aware of some of the functionality of the dplyr package.

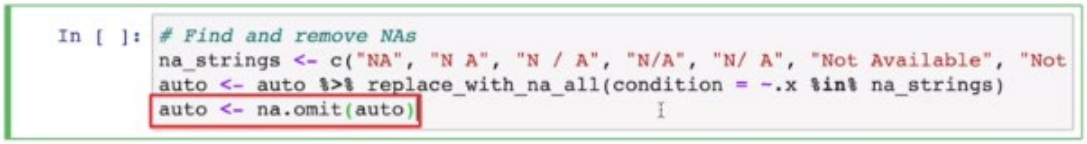
The dplyr package is part of the tidyverse package. Recall how the dollar sign was used earlier to select a data frame column. This method would suffice if you need to select one column, but the syntax can become rather complex when multiple columns must be selected. In this case, the “which” function can be used to select column numbers if they have equal value. Two column names are separated using the “or” operator.

However, using the dplyr function is much easier. First, we indicate which data frame the dplyr package should use. In this example, it will be the auto followed by the pipe operator. We then tell the package to select specific columns we would like to use. It generates the same output, but it is much easier to code and to understand later on, especially when the data becomes increasingly complex.

Section D: Cleaning data

KALOGEROPOULOS: As mentioned at the start of this video, the quality of the data is essential to get reliable outputs. As a result, the data must be cleaned before it can be used.

In the first code cell in this section, the first line shows all the possible values that could have been used for missing data. The second line then replaces all of these possibilities with R’s value for missing data. The function applied here is part of the dplyr package. After the missing data has been replaced, the rows containing these missing values are then omitted, as shown in the third line.



Another issue with the data set is the potential of duplicate entries. Duplicate entries are removed from the data by using the “distinct” function. The “distinct” function chooses all rows that are unique. The dplyr operator is then used to provide a list of all the unique rows in the data set. However, remember that some duplicates may be valid in certain data sets, so it is important to first understand the data set used before removing duplicates.