

R Programming

Advanced Level

Duration: 1 Day (9am – 4pm)

Course Pre-Requisites: Knowledge of R Programming at an intermediate level is recommended for this course.

Course Overview

In this advanced R programming course, participants will elevate their coding prowess. They'll master the art of crafting custom functions and writing succinct code. The curriculum delves into advanced topics, including functionals, loops, and other essential control structures. Furthermore, attendees will gain hands-on experience in data visualisation, harnessing the power of ggplot2 to create compelling graphics.

1. Working Efficiently in R

This section is dedicated to enhancing students' proficiency in R, guiding them on ways to streamline their coding processes. They will delve into the concept of vectorisation, a method that optimises operations by applying them to whole data sets instead of individual elements. The course will introduce students to superior functions that can simplify tasks and increase performance. Emphasis will be placed on writing concise code that not only reduces errors but also boosts readability. Students will also learn the importance of reusable scripts, ensuring consistency and efficiency in their projects. Crafting custom functions will be explored, giving participants the ability to tailor their functions to specific needs. Additionally, they'll be introduced to loops and other control structures, essential tools for repetitive tasks. To complement this, loop alternatives will be discussed, offering more efficient ways to handle repetitive operations in R.

2. Functions to Reduce Typing

This section emphasises optimising the coding process in R by minimising manual typing. Students will explore methods to automatically print objects after their creation, enhancing their workflow efficiency. The course will delve into the utilisation of the `with` and `within` functions, which simplify data frame manipulations by reducing repetitive references. Additionally, techniques for inserting multiple quotation marks will be introduced, further streamlining the coding process and ensuring accuracy in data representation.

3. Create Your Own Functions

In this module, students will delve into the art of crafting custom functions in R. They'll begin by understanding the significance of writing their own functions, ensuring tailor-made solutions for specific challenges. The foundation will be set with a comprehensive look at the basics, answering pivotal questions like "What is a function?" As they progress, participants will be guided through the syntax and methodologies for creating both simple and complex functions. Advanced topics such as the use of ellipses for additional arguments and the concept of scope within functions will be covered. The section concludes with insights on how to effectively load and deploy these custom functions, ensuring they are accessible for various tasks.

4. Loops and Control Structures in R

This section provides students with a deep dive into the mechanisms that facilitate repetitive and conditional operations in R. Beginning with conditional structures, learners will grasp the concepts of **if** and **if else** to guide decision-making in their code. The focus then shifts to loops, with detailed exploration of **for loops**, enabling iterative operations over sequences or data structures. Essential techniques for saving results derived from these loops will be imparted, ensuring data is effectively captured. To further refine their looping skills, students will be introduced to **while loops**, which run based on a specified condition, and **repeat loops**, offering more flexibility in iterative processes. Throughout, emphasis will be placed on improving code efficiency and clarity.

5. Loop Alternatives

This segment delves into efficient alternatives to traditional loops in R. Students will be introduced to functionals, a suite of tools designed for streamlined and concise operations on data. The **apply** functions will be explored, offering mechanisms to apply a function over array margins. Learners will also understand the utility of the **split** function, which divides data into subsets. The module will cover the **map** functions and their variants, equipping students with tools for iterative operations across lists and vectors. Emphasis will be on returning vectors efficiently and the **walk** function's utility for side effects. By the section's conclusion, students will be adept at deciding when to use loops, **apply**, or **map** based on specific scenarios.

6. Tidyverse Piping Syntax

This section introduces students to the transformative piping syntax within the Tidyverse ecosystem of R. They'll begin by understanding the core purpose of piping, which facilitates a more readable and streamlined workflow. The primary focus will be on mastering the use of the **pipe**, a powerful tool that allows for sequential data operations. As they delve deeper, students will encounter various **pipe** variants, expanding their repertoire of data manipulation techniques. The nuances of the **tee operator**, which permits side operations without

disrupting the primary flow, will be discussed. Additionally, the course will touch upon the **exposition operator**, further enhancing students' capability to wield the piping syntax effectively.

7. Plotting with ggplot2

In this section, students will immerse themselves in the world of advanced data visualisation using ggplot2, a cornerstone of the Tidyverse in R. They'll be introduced to the "grammar of graphics", a systematic approach to crafting comprehensive plots. The foundational components required for any ggplot2 visualisation will be covered, leading into hands-on experience with the **ggplot()** function. Practical exercises will involve creating scatterplots and line charts, enabling students to represent data in meaningful ways. To ensure a holistic understanding, the course will also provide valuable ggplot2 resources for further exploration and mastery.