

Week 2 - Introduction to Mathematical Modelling

Learning Objectives

In this module, you will learn to:

1. Critically observe a real-world problem and applying the 4-stages of mathematical modelling (building, analysing, validating and applying) to provide insights and predictions
2. Demonstrate an understanding of basic mathematical concepts in data science, relating to linear function, descriptive analysis, inferential analysis and linear regression.
3. Utilise R to prepare data for analysis, perform simple data analysis, create meaningful data visualization, and make predictions from data.
4. Produce a rigorous analytical report which considers a broad range of mathematical and statistical methods to describe, analyse, extrapolate, and apply big data.

Administrivia

- 12 Week half module
- 4 hours of Seminars
- 2 hours of theoretical work, 2 hours of practical work in R
- 2 Assessments

Assessments

- Assessment 1 is a case study report.
- 24th May 2024
- This assessment counts for 40% of total grade.
- This assessment *IS* eligible for self-certification

- Assessment 2 is an exam.
- Due: 22nd March 2023
- This assessment counts for 60% of your total grade.
- This assessment *is not* eligible for self-certification.

Introduction:

Mathematical modeling is a powerful technique used in various fields to understand, analyze, and solve complex real-world problems. In this module, we will explore the fundamentals of mathematical modeling, its applications, and the modeling process.

What is Mathematical Modeling?

Mathematical modeling is the process of using mathematical equations, algorithms, and simulations to represent, describe, and analyze real-world phenomena. It bridges the gap between the physical world and mathematical abstractions. Data science is a multidisciplinary field that combines knowledge of statistics, computer science, and domain expertise. A strong foundation in statistics and mathematics is essential for data scientists. Data scientists should be comfortable with probability theory, statistical inference, hypothesis testing, and regression analysis.

What skills might a Mathematical Modeller need?

- **Programming:** Data scientists should be proficient in at least one programming language, such as Python, R, or Java. They should be able to write efficient and well-documented code, and have experience with version control systems like Git.
- **Data Manipulation:** Data scientists should be skilled in manipulating and cleaning data using tools like SQL, Pandas, or Excel. They should be familiar with data transformation, data normalization, and data cleaning techniques.
- **Machine Learning:** Data scientists should have experience with machine learning algorithms, such as linear regression, logistic regression, decision trees, random forests, and neural networks. They should know how to evaluate models and choose the best model for a given problem.
- **Data Visualization:** Data scientists should be able to visualize data using tools like Matplotlib, Seaborn, or Tableau. They should be able to create meaningful and informative visualizations that communicate insights from data.

- **Communication:** Data scientists should have strong communication skills, both written and verbal. They should be able to explain complex concepts in simple terms, and effectively communicate insights to non-technical stakeholders.
- **Domain Expertise:** Data scientists should have some domain expertise in the field they are working in. They should be familiar with the business context of the data they are working with, and have a deep understanding of the problems they are trying to solve.
- **Curiosity and Problem Solving:** Data scientists should have a curious and inquisitive mindset, always looking for ways to improve their models and find insights in the data. They should be able to identify problems and propose solutions to those problems.
- **Continuous Learning:** Data science is a rapidly evolving field, and data scientists should be willing to continuously learn and keep up with new developments in the field. They should be open to new ideas and techniques, and willing to adapt to changing circumstances.

Applications of Mathematical Modeling:

Mathematical modeling is used in a wide range of fields, including:

- **Physics:** Modeling the motion of planets, electromagnetic waves, and quantum mechanics.
- **Engineering:** Designing structures, optimizing processes, and predicting behavior of systems.
- **Economics:** Modeling supply and demand, economic growth, and financial markets.
- **Biology:** Modeling population dynamics, disease spread, and biological processes.
- **Environmental Science:** Modeling climate change, ecosystems, and pollution.
- **Social Sciences:** Modeling behavior, opinion dynamics, and social networks.

The Modeling Process:

Mathematical modeling involves several key steps:

1. **Problem Formulation:** Clearly define the problem you want to model. Identify the key variables, assumptions, and constraints.
2. **Mathematical Representation:** Choose appropriate mathematical equations or algorithms that represent the problem.
3. **Parameter Estimation:** Determine the values of model parameters using data or expert knowledge.
4. **Validation:** Test the model's accuracy by comparing its predictions to real-world observations.
5. **Simulation and Analysis:** Use the model to perform simulations and analyze the results.

6. Interpretation: Interpret the model's findings and draw conclusions.
7. Communication: Communicate the results and insights to stakeholders through reports, presentations, or visualizations.

Types of Mathematical Models:

There are various types of mathematical models, including:

- Deterministic Models: Based on precise mathematical equations with fixed parameters.
- Stochastic Models: Incorporate randomness and uncertainty into the modeling process.
- Continuous Models: Represent phenomena that change continuously over time or space.
- Discrete Models: Model phenomena that change in discrete steps or intervals.
- Analytical Models: Solvable using mathematical techniques.
- Numerical Models: Require computational methods for solution.

During this module we will mostly be focused on Deterministic Models and Analytical Models.

Challenges in Mathematical Modeling:

Mathematical modeling is a powerful tool but comes with challenges:

- Data Availability: Accurate and relevant data is essential for model development and validation.
- Complexity: Real-world systems can be highly complex, requiring simplifications and assumptions.
- Parameter Estimation: Determining accurate parameter values can be challenging.
- Uncertainty: Dealing with uncertainty and variability in models is crucial.
- Model Validation: Ensuring that models accurately represent reality is an ongoing process.

Exercise

In groups of 3, think about a phenomena you might want to model (Weather, Stock Prices, Spread of COVID-19 etc...) Think about how you might model this? What data would you need? What factors would you like to include in your model? How would you test to see if the model was working?

Tools and Software:

Mathematical modeling often involves the use of software tools and programming languages such as MATLAB, Python, R, and specialized modeling software like COMSOL, Simulink, or specialized domain-specific software.

Introduction to R

Introduction to Variables

Definition: Variables are containers for storing data values. In R, you can assign values to variables using the equal sign (=) or by using a left pointing arrow (<-).

Syntax: variable_name = value

Example:

```
name = "John"
```

Explanation: In this example, the string “John” is assigned to the variable named “name”.

Variable Types

R supports several built-in data types, such as: - integers (int) - floating-point numbers (float) - strings (str) - lists (list). R dynamically assigns the data type to a variable based on the value assigned to it.

Reassigning Values to Variables

You can reassign values to variables at any time.

Example - Try this code in the console:

```
name = "John"
name
'John'
name = "Jane"
name
'Jane'
```

Explanation: In this example, the value of the “name” variable has been changed from “John” to “Jane”.

Best Practices for Variable Names

- Variable names should be descriptive and meaningful.
- Variable names can contain letters, numbers, and underscores. They cannot start with a number.
- Avoid using Python keywords as variable names.
- Recommended to use lowercase letters in variable names and separate words with underscores.

R packages

- **Packages** are the fundamental units of reproducible R code. They include reusable R functions, the documentation that describes how to use them, and sample data.
- As of September 2020, there are over 16,000 R packages available on **CRAN** (the Comprehensive R Archive Network).
- We're going to work with a small (but important) subset of these!

R Studio Tour

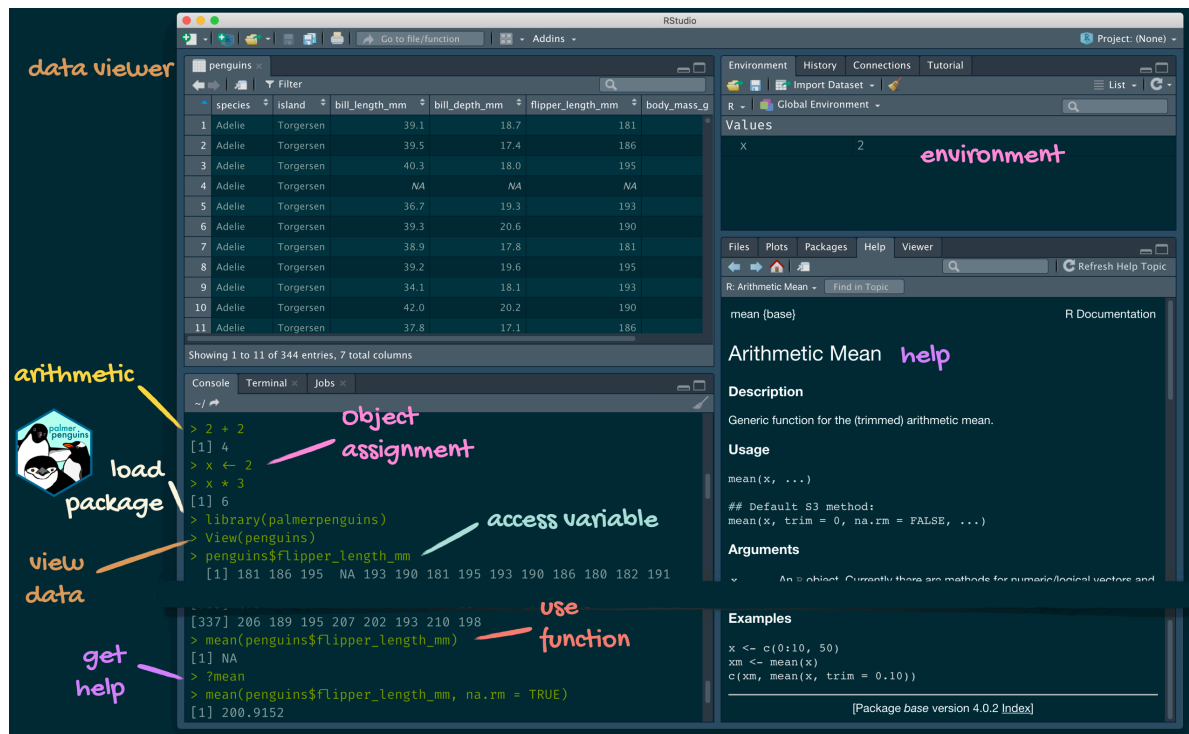


Figure 1: Tour

R Essentials

In programming, a function is a block of code that performs a specific task. Functions are a way to organize code and make it reusable. To call a function in R, simply type the name of the function followed by parentheses. If the function takes arguments, you must provide them inside the parentheses.

Definition : an argument is a value or variable that is passed to a function or method, Here is an example of a function that takes two arguments and returns their sum: `def add_numbers(a, b):`

Functions are (most often) verbs, followed by what they will be applied to in parentheses:

```
do_this(to_this)
do_that(to_this, to_that, with_those)
```

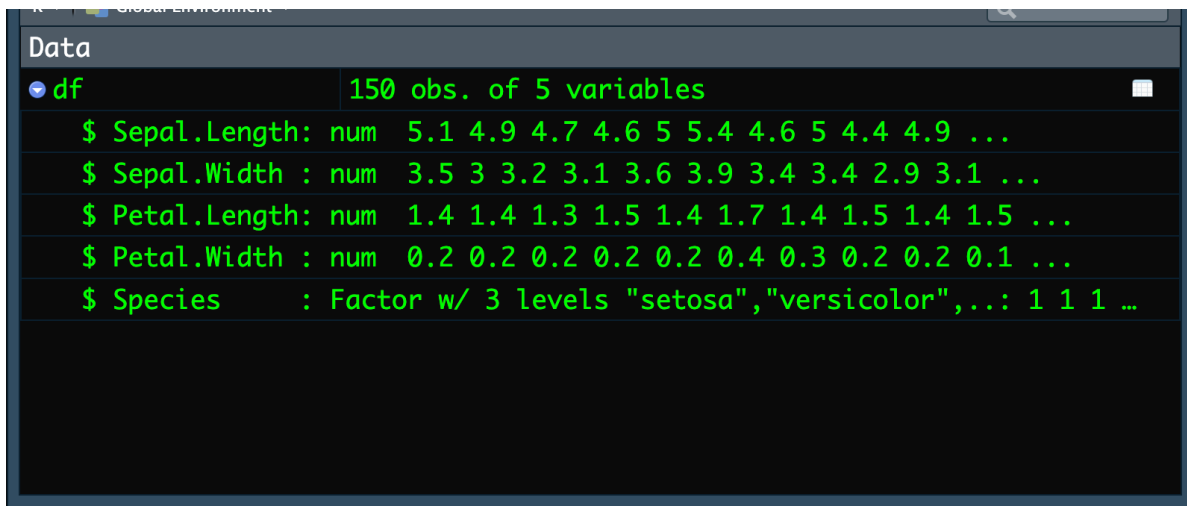
Packages are installed with the `install.packages` function and loaded with the `library` function, once per session:

```
install.packages("package_name")  
library(package_name)
```

During this module, we will mostly be using in built datasources. Try this now

```
df <- iris
```

To your right, you should see new value in Environment called df, and if you expand this you will see something that looks like this.



The screenshot shows the RStudio Environment pane. At the top, it says 'Data'. Below that, the object 'df' is listed with a dropdown arrow and the text '150 obs. of 5 variables'. The object is expanded, showing the following structure:


Variable	Class	Values (first 10)
\$ Sepal.Length	num	5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
\$ Sepal.Width	num	3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
\$ Petal.Length	num	1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
\$ Petal.Width	num	0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
\$ Species	Factor w/ 3 levels	"setosa", "versicolor", ...: 1 1 1 ...

Figure 2: Tour

If you click on the df, it will expand the data and look something like this

test.Rmd x 02-ethics.qmd x u1-d02-toolkit-r.Rmd x R Untitled1* x

← →

 Filter

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa
14	4.3	3.0	1.1	0.1	setosa
15	5.8	4.0	1.2	0.2	setosa
16	5.7	4.4	1.5	0.4	setosa
17	5.4	3.9	1.3	0.4	setosa
18	5.1	3.5	1.4	0.3	setosa
19	5.7	3.8	1.7	0.3	setosa
20	5.1	3.8	1.5	0.3	setosa
21	5.4	3.4	1.7	0.2	setosa
22	5.1	3.7	1.5	0.4	setosa

Showing 1 to 22 of 150 entries, 5 total columns

Figure 3: Tour

- Columns (variables) in data frames are accessed with \$:

```
df$Sepal.Length
```

- Object documentation can be accessed with ?

```
?mean
```

How to plan your code

Pseudocode is a way of representing code in a way that is easy to read and understand. It's used as a planning tool before writing the actual code in a programming language. Here's a step-by-step guide on how to write pseudocode: Start with a clear understanding of the problem you're trying to solve. Break the problem down into smaller steps. Begin writing your pseudocode with a descriptive line of code that describes what you're trying to do. For example, "Calculate the average of a list of numbers." Use indentation can be thought of as large brackets that show the structure of your code. Indentations will help you keep track of the order in which tasks should be executed. For example:

- Start:
- Set total to 0
- For each number in the list:
 - Add the number to the total
- Set the average to the total divided by the number of items in the list
- Output the average
- End.

Use variables to represent data that will be used in the code. Give variables clear and descriptive names that reflect what they are used for. For example:

- Start:
- Set total to 0
- Set count to 0
- For each number in the list:
 - Add the number to the total
 - Increment count by 1
- Set the average to the total divided by count
- Output the average
- End.

Use simple, plain language to describe the steps in your code. Avoid using programming syntax and jargon. Your pseudocode should be easily understandable by someone who is not familiar with programming. Make sure to break down complex tasks into smaller, more manageable ones. For example, if you're trying to sort a list of numbers, you might break it down into several steps, such as:

“Find the smallest number in the list,” “Remove the smallest number from the list,” “Add the smallest number to a new list,” and so on.

Test your pseudocode by walking through it step-by-step. Make sure that the steps you've described will lead to the desired outcome. Remember, pseudocode is not an exact syntax or programming language. It's simply a way to plan and outline your code before you begin writing it in a programming language. By breaking down complex problems into smaller, manageable steps, you can create clear and effective pseudocode that will help you write better code.

Conclusion:

Mathematical modeling is a vital tool for understanding and solving complex real-world problems in various fields. The modeling process involves problem formulation, mathematical representation, parameter estimation, validation, analysis, interpretation, and communication. While modeling can be challenging, it provides valuable insights and solutions to pressing issues in science, engineering, economics, and many other domains.