Computer Programming for Social Scientists

Trinity College Dublin 2025/26

Tom Paskhalis

tom.paskhal.is

- **Module Code:** POP77001
- **Module Website:** tom.paskhal.is/POP77001
- **ECTS Weighting:** 10
- **Semester/Term Taught:** Semester 1 (Michaelmas Term)
- **Contact Hours:**
  One 2-hour lecture:
  - Monday 14:00-16:00 in 2041B (Arts Building)
  One 2-hour tutorial:
  - Group 1 - Thursday 16:00-18:00 in 1.24 (D'Olier Street)
  - Group 2 - Friday 16:00-18:00 in 5052 (Arts Building)
  per week (11 weeks)
- **Module Coordinator:** Dr Tom Paskhalis (tom.paskhalis@tcd.ie)
- **Office Hours:** Friday 11:00-13:00 in-person or online (booking required)
- **Teaching Fellows:**
  - Sara Cid (cidsb@tcd.ie)

Learning Aims

This module provides foundational knowledge of computer programming concepts and software engineering practices. It introduces students to major programming languages and workflows for data analysis, with a focus on social science questions and statistical techniques.

Learning Outcomes

On successful completion of this module students should be able to:

- describe fundamental computer programming concepts;
- demonstrate command of the R and Python programming languages;
- exhibit the ability to write, execute and debug scripts for data analysis;
- perform data wrangling tasks using R and Python;
- analyse the complexity and assess the performance of computer programs;

Module Content

Students will become familiar with R and Python, two principal programming languages used in data science and research. This course covers basic and intermediate programming concepts, such as objects, types, functions, control flow, debugging in both procedural and object-oriented paradigms. Particular emphasis will be made on data handling and analytical tasks with a focus on problems in social sciences. Homeworks will include hands-on coding exercises. In addition, students will apply their programming knowledge on a research project at the end of the module.

Software

In this module we will study the fundamentals of computer programming using R and Python. Both are free, open-source and interactive programming languages widely used for data analysis. R and Python are widely available for all major operating systems (Windows, Mac OS, Linux).

While there are a range of integrated development environments (IDEs) available for both R and Python (and which are very worth exploring further, more details below), we will use Jupyter Notebooks as the primary way of writing and executing code, and assignment submission.

To work with Jupyter Notebooks, you will need to install Jupyter Notebook on your local machine. I recommend installing JupyterLab Desktop, the cross-platform desktop application for working with Jupyter Notebooks. To use Jupyter Desktop with R, you will also need to install the IRkernel package. Check the instructions for further details on installation and setup.

Alternatively, you may want to try Kaggle Code, an online platform for working with, sharing and exploring data-science-focussed Jupyter Notebooks. Using Kaggle Code requires registration (you can also use your Google account if you have one). While this platform will provide sufficient functionality (and package availability) for completing all assignments for this module, I strongly advise to have a local installation of R, Python and Jupyter Notebook on your machine that you can use moving forward.

In addition to having a local installation of R, Python and JupyterLab Desktop, I advise to install a feature-rich text editor that will allow you to open and inspect (with syntax highlighting) a wide range of scripts and configuration files. Here are a couple of options to try:

- Visual Studio Code
- Sublime Text

Some IDEs for working in R and Python that you might like to try as well:

- RStudio - very popular IDE for R;
- Spyder - similar in appearance IDE for Python;
- PyCharm - development-focussed non-free IDE for Python.

Note that irrespective of your preferred IDE and tool chain all assignments have to be submitted as valid Jupyter Notebooks with all code cells executed prior to submission.

Recommended Reading List

In this module we will rely on a number of books that introduce R and Python with a particular focus on data analysis applications. All of the required readings are available either freely online or through the College Library. While it is not necessary, I strongly advise selecting one or two books (depending on their delivery style and your personal preferences) to purchase as reference texts.

- John Guttag. 2021. *Introduction to Computation and Programming Using Python: With Application to Computational Modeling and Understanding Data.* 3rd ed. Cambridge, MA: The MIT Press
- Norman Matloff. 2011. *The Art of R Programming: A Tour of Statistical Software Design.* San Francisco, CA: No Starch Press
- Wes McKinney. 2022. *Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter.* 3rd ed. Sebastopol, CA: O'Reilly Media. https://wesmckinney.com/book/
- Roger D. Peng. 2016. *R Programming for Data Science.* Leanpub. https://leanpub.com/rprogramming
- Hadley Wickham, Mine Çetinkaya-Rundel, and Garrett Grolemund. 2023. *R for Data Science.* 2nd ed. Sebastopol, CA: O'Reilly Media. https://r4ds.hadley.nz/
- Hadley Wickham. 2019. *Advanced R.* 2nd ed. Boca Raton, FL: Chapman and Hall/CRC. https://adv-r.hadley.nz/

In order to become a better programmer it is important to both develop a solid understanding of a specific language as well as learn about good coding practices more broadly. This book offers a very good and accessible coverage of these approaches:

- David Thomas and Andrew Hunt. 2019. *The Pragmatic Programmer: Your Journey to Master.* 2nd. Boston, MA: Addison-Wesley Professional. https://pragprog.com/titles/tpp20

While not focussed on computer programming *per se*, the following books provide a good background reading on general historical and technical (but accessible) details about binary systems and code, and how computers and related systems (networks, operating systems, etc.) work more broadly:

- Matthew Justice. 2020. *How Computers Really Work: A Hands-On Guide to the Inner Workings of the Machine.* No Starch Press
- Brian W. Kernighan. 2021. *Understanding the Digital World: What You Need to Know about Computers, the Internet, Privacy, and Security.* Princenton, NJ: Princenton University Press
- Charles Petzold. 2022. *Code: The Hidden Language of Computer Hardware and Software.* 2nd ed. Redmond, WA: Microsoft Press. https://www.codehiddenlanguage.com/

If you are looking for a book that provides examples of applying statistical analysis techniques using both R and Python see:

- Alan Agresti and Maria Kateri. 2021. *Foundations of Statistics for Data Scientists: With R and Python.* Boca Raton, FL: Chapman and Hall/CRC

Additional online resources:

- Git Book
- R Inferno
- An Introduction to R and Python For Data Analysis: A Side By Side Approach
- The Hitchhiker's Guide to Python
- Python For You and Me
- Python Wikibook
- Official documentation:
  - R Language Definition
  - Python Language Reference

Assessment Details

The final grade consists of the following parts (with corresponding weighting):

- Participation (tutorial attendance, 10% total)
- Programming exercises (30% total)
- Final project (60%)

All assignments should be submitted via Blackboard. Go to the “Assessment” section — you should be able to see all the assignments listed there. You will need to upload your assignments as Jupyter Notebook. Make sure to check that all cells in your notebook execute correctly and without error prior to submission.

Please make sure that you understand the submission procedure. Unexcused late submissions will be penalized in accordance with standard department policy. Five points per day will be subtracted until the Monday a week and a half after the deadline at which point the assignment is deemed to have failed.

All assignments are due by **12:00 Monday** prior to the start of the lecture. See module schedule summary below for the full list of due dates.

The final project will be due by **23:59 Friday, 19 December 2025**.

Plagiarism

Plagiarism — defined by the College as the act of presenting the work of others as one’s own work, without acknowledgement — is unacceptable under any circumstances. All submitted coursework must be **individual** and **original**. While the regulations you will find in the College policy on plagiarism largely describe assignments consisting of written text, note that similar guidelines apply to code submitted for assessment. Plagiarising computer code is as serious as plagiarising text and will have serious implications both within this class and in the real world (see Google LLC v. Oracle America, Inc. for one of the examples). While you may discuss general approaches to solutions with your peers, under no circumstances you are allowed to share and view each others code. Watch this video explaining the difference between collaboration and collusion to see concrete examples. Note that in case of an identified plagiarism all students whose code appears to come from the same source without giving due credit will be penalized. You can use online resources (e.g. Stack Overflow) but you need to give credit (and link) in the comments.

Generative AI

The use of generative AI in many ways remains a grey arey both in academia and industry. Keep in mind that all information that you share with generative AI will be stored and re-used, so under no circumstances private and personal data should be submitted to any (non-local) generative AI services. For the purposes of this module a limited fair use of generative AI is permitted. This includes the use of such services to explain unclear concepts and code chunks, refactoring of written code (keeping in mind the caveat about data above) and its use in the final project. Using generative AI to generate solutions to assignments is strictly prohibited. If you are unsure about the use of generative AI in a specific context, please ask the instructor.

Module Schedule

Week 1: Introduction to Computation

In the first week we discuss core software development concepts such as computers, programming languages and algorithms.

Required Readings:

- McKinney Ch 2 Python Language Basics, IPython, and Jupyter Notebooks;
- Jeannette M. Wing. 2006. Computational Thinking. *Communications of the ACM* 49 (3): 33–35. https://doi.org/10.1145/1118178.1118215

Additional Readings:

- Wickham, Çetinkaya-Rundel & Grolemund Chs 1 Introduction, 5 Workflow: code style, 7 Workflow: scripts and projects, 9 Workflow: getting help

Week 2: R Basics

In this week we discuss the fundamental concepts of programming, such as variables, assignment and object types with application to R. In addition, we start using some built-in functions.

Required Readings:

- Wickham Chs 2 Names and Values, 3 Vectors, 4 Subsetting;
- Peng Chs 5 R Nuts and Bolts, 10 Subsetting R Objects, 11 Vectorized Operations;

Additional Readings:

- Matloff Chs 2 Vectors, 3 Matrices & Arrays, 4 Lists, 5 Data Frames, 6 Factors & Tables.

- Ross Ihaka and Robert Gentleman. 1996. R: A Language for Data Analysis and Graphics. *Journal of Computational and Graphical Statistics* 5 (3): 299–314. https://www.doi.org/10.1080/10618600.1996.10474713

Week 3: Control Flow in R

Straightline programs where each line of code gets executed one after another can get us only so far. In this week we focus on the key ways of controlling the flow of programs in R. We look at branching and loops, common for all programming languages and the details of their design and implementation in R.

Required Readings:

- Peng Ch 14 Control Structures;

- Wickham Ch 5 Control Flow;

Additional Readings:

- Matloff Ch 7 R Programming Structures;

Week 4: Functions in R

Functions are the core building blocks of a program written in any language. In this week we discuss the fundamentals of function definition and invocation in R. We also look at the concept of scoping and how it affects the way we write and use functions.

Required Readings:

- Peng Chs 15 Functions, 16 Scoping Rules;

- Wickham Chs 6 Functions, 7 Environments;

Additional Readings:

- Wickham Chs 9 Functionals, 10 Function Factories, 11 Function Operators;

Week 5: Debugging and Testing in R

Finding and eliminating errors in code is one of (if not the most) frustrating part of computer programming. This week we focus on how to debug and test an R program. We start with the usage of `print()` statement to analyse the state of function calls and loops. Afterwards, we discuss more structured ways of error-catching and debugging with the help of built-in R debugger.

Required Readings:

- Wickham Chs 8 Conditions, 22 Debugging;
- Peng Ch 20 Debugging;

Additional Readings:

- Matloff Ch 13 Debugging;

Week 6: Data Wrangling in R

Working with data is at the centre of programming in R. In addition to core functionality of base R, many new packages, such as `tidyverse` collection provide advanced data manipulation facilities and enhanced experience of working with tabular data. In this week we focus on data frame and its tidyverse cousin tibble. We also discuss formats of data storage and functions for data I/O and descriptive analysis.

Required Readings:

- Wickham, Çetinkaya-Rundel & Grolemund Chs 4 Data transformation, 6 Data tidying, 8 Data import

Additional Readings:

- Peng Chs 13 Managing Data Frames, 18 Loop Functions

Week 7: Reading Week

Week 8: Fundamentals of Python Programming I

This week we start learning about Python, another major language for data analysis. In the first lecture we look at core Python object types, operators, methods and functions. Some of Python fundamentals will be compared and contrasted to their counterparts in R.

Required Readings:

- Guttag Chs 2 Introduction to Python, 5 Structured Types and Mutability;

Additional Readings:

- Guttag Ch 3 Some Simple Numerical Programs;

Week 9: Fundamentals of Python Programming II

As in other programming languages, functions are crucial for building modular programs. In this week we look at control flow mechanisms and discuss function definition and invocation in Python.

Required Readings:

- Guttag Chs 4 Functions, Scoping and Abstraction, 6 Recursion and Global Variables;

Additional Readings:

- Guttag Chs 7 Modules and Files, 8 Testing and Debugging, 9 Exceptions and Assertions.

Week 10: Data Wrangling in Python

This week we turn from broader programming and software engineering concepts to practical approaches of working with data in Python. In particular, we will focus on `pandas`, a versatile library for data analysis, which often serves as the first building block in many data-science pipelines.

Required Readings:

- McKinney Chs 4 NumPy Basics, 5 Getting Started with Pandas, 6 Data Loading, Storage and File Formats, 7 Data Cleaning and Preparation, 8 Data Wrangling: Join, Combine and Reshape;

Additional Readings:

- Guttag Ch 23 Exploring Data with Pandas;
- Charles R. Harris et al. 2020. Array programming with NumPy. *Nature* 585 (7825): 357–362. https://doi.org/10.1038/s41586-020-2649-2

Week 11: Classes and Object-oriented Programming

We saw how functions allow us to make our code more generalisable and abstract. But what if we wanted to bundle our code with the kinds of data it could operate on? Classes and object-oriented programming allow us to address this challenge.

Required Readings:

- Guttag Ch 10 Classes and Object-oriented Programming;

Additional Readings:

- Wickham Chs Object-oriented programming: Introduction, 12 Base types, 13 S3;
- Bjarne Stroustrup. 1991. What is "Object-Oriented Programming"? (1991 revised version). *Proceedings of the 1st European Software Festival*, https://stroustrup.com/whatis.pdf

Week 12: Performance and Complexity

Getting the correct result and having well-structured and documented code are only two aspects of a good program. We also want our code to execute fast and, in some cases, for it to finish running in a moment, hour, day, year, lifetime... In this week we more formally discuss algorithmic complexity and performance. In addition to theoretical considerations we look into measuring execution time and benchmarking specific operations.

Required Readings:

- Guttag Chs 11 A Simplistic Introduction to Algorithmic Complexity, 12 Some Simple Algorithms and Data Structures;

Additional Readings:

- Wickham Chs 23 Measuring Performance, 24 Improving Performance.

Module Schedule Summary

Week	Date	Language	Topic	Released	Due
1	15 September	-	Introduction to Computation		
2	22 September	R	R Basics	Assignment 1	
3	29 September	R	Control Flow in R		
4	6 October	R	Functions in R		Assignment 1
5	13 October	R	Debugging and Testing in R	Assignment 2	
6	20 October	R	Data Wrangling in R		
7	27 October	-	-		Assignment 2
8	3 November	Python	Fundamentals of Python Programming I	Assignment 3	
9	10 November	Python	Fundamentals of Python Programming II		
10	17 November	Python	Data Wrangling in Python	Assignment 4	Assignment 3
11	24 November	Python	Classes and Object-oriented Programming		
12	1 December	Python, R	Complexity and Performance		Assignment 4