

Computer Programming for Social Scientists

Trinity College Dublin 2022/23

Tom Paskhalis

tom.paskhal.is

-
- **Module Code:** POP77001
 - **Module Website:** tinyurl.com/POP77001
 - **ECTS Weighting:** 10
 - **Semester/Term Taught:** Semester 1 (Michaelmas Term)
 - **Contact Hours:**
One 2-hour lecture:
- Monday 14:00-16:00 in PX 201 ([7-9 Leinster Street South](#))
One 2-hour tutorial:
- Thursday 09:00-11:00 in PX 201 ([7-9 Leinster Street South](#))
per week (11 weeks)
 - **Module Coordinator:** Dr Tom Paskhalis (tom.paskhalis@tcd.ie)
 - **Office Hours:** Thursday 11:00-13:00 [in-person or online](#) (booking required)
 - **Tutorial Coordinator:** Dr Martyn Egan (eganm9@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.

There are two main ways to install Jupyter Notebook on your local machine: [pip](#) and [conda](#). Unless you have prior experience with Python, I recommend installing [Anaconda](#) distribution, which contains both R and Python as well as all the packages required for this course.

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 Jupyter Notebook, 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 few options to try:

- [Visual Studio Code](#)
- [Atom](#)
- [Sublime Text](#)

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

- [RStudio](#) - very popular (de-facto standard) 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 and Garrett Grolemund. 2017. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. Sebastopol, CA: O'Reilly Media. <http://r4ds.had.co.nz/>
- Hadley Wickham. 2019. *Advanced R*. 2nd ed. Boca Raton, FL: Chapman and Hall/CRC. <https://adv-r.hadley.nz/>

While not focussed on either R or Python, 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*. San Francisco, CA: No Starch Press
- Brian W. Kernighan. 2021. *Understanding the Digital World: What You Need to Know about Computers, the Internet, Privacy, and Security*. Princeton, NJ: Princeton 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](#)
- [The Hitchhiker's Guide to Python](#)
- [Python For You and Me](#)
- [Python Wikibook](#)
- [Python 3 Documentation](#)
- [R Documentation](#)
- [R Inferno](#)

Assessment Details

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

- Participation (tutorial attendance, 10% total)
- 4 Programming exercises (10% each, 40% total)
- Final project (50%)

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. Please, make sure to check that all cells in your notebook execute correctly and without error prior to submission.

Please make certain that you understand the submission procedure. Unexcused late submissions will be penalized in accordance with standard department policy.

All assignments are due by 12:00 Monday.

See [module schedule summary](#) below for the full list of due dates.

The final project will be due by 12:00 Monday, 19 December 2022.

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.

Module Schedule

Week 1: Introduction to Computation	6
Week 2: R Basics	6
Week 3: Control Flow in R	6
Week 4: Functions in R	7
Week 5: Debugging and Testing in R	7
Week 6: Data Wrangling in R	7
Week 8: Fundamentals of Python Programming I	8
Week 9: Fundamentals of Python Programming II	8
Week 10: Data Wrangling in Python	8
Week 11: Classes and Object-oriented Programming	9
Week 12: Performance and Complexity	9
Module Schedule Summary	10

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 & Grolemund Ch 3: [Workflow - Basics](#)

Week 2: R Basics

This week we will discuss fundamental concepts of programming, such as variables, assignment and object types with application to R. In addition, we will 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.

Week 3: Control Flow in R

Straightline This week we will focus on the key ways of controlling the flow of programs in R. We will look at branching and loops, common for all programming languages and the details of their design 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

This week we will discuss functions, the main building block of a program implemented in any language.

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 will focus on how to debug and test an R program. We will start with the usage of `print()` statement to analyse the state of function calls and loops. Afterwards, we will 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 will focus on data frame and its tidyverse cousin `tibble`. We will also discuss formats of data storage and functions for data I/O and descriptive analysis.

Required Readings:

- Wickham & Grolemund Chs [7: Tibbles](#), [8: Data Import](#), [9: Tidy Data](#)

Additional Readings:

- Peng Chs 13: Managing Data Frames, 18: Loop Functions

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 will 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:

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

Additional Readings:

- Gutttag 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:

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

Additional Readings:

- Gutttag Chs 7: Modules and Files, 8: Testing and Debugging, 9: Exceptions and Assertions;

Week 10: Data Wrangling in Python

This week we will 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:

- Gutttag 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... 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](#).

Table 1: **Module Schedule Summary**

Week	Date	Language	Topic	Due
1	12 September	-	Introduction to Computation	
2	19 September	R	R Basics	
3	26 September	R	Control Flow in R	
4	3 October	R	Functions in R	Assignment 1
5	10 October	R	Debugging and Testing in R	
6	17 October	R	Data Wrangling in R	
7	24 October	-	-	Assignment 2
8	31 October	Python	Fundamentals of Python Programming I	
9	7 November	Python	Fundamentals of Python Programming II	
10	14 November	Python	Data Wrangling in Python	Assignment 3
11	21 November	Python	Classes and Object-oriented Programming	
12	28 November	Python, R	Complexity and Performance	Assignment 4