STOR609 : Programming for Reproducible Research

- **Proposal Type:** **New**
- **Proposal Status:** **Draft**

Basic Details

Title
Programming for Reproducible Research

Mnemonic
STOR609

Year of Introduction
24/25

Module Convenor
Fairbrother, Jamie (Management Science)

Credit Rating
20.00

Total Learning Hours
200.00

Mode of Delivery
Standard

Level
Postgraduate (Masters level)

National Credit Framework Level
FHEQ/QCF/NQF7/RQF7

Module Type
PG Taught Standard

Administering Department

Subject	Department	Load
Statistics	Mathematics and Statistics	50%

Contributing Departments

Subject	Department	Load
Management Science	Management Science	50%

Rationale for introducing the module

This new module is a part of our EPSRC funded proposal for the new STOR-i Centre for Doctoral Training. Our industrial partners requested a more substantial module focussed on advanced programming, with a particular focus on equipping students with the tools and undertsanding required to develop reproducible research for Statistics and Operational Research.

Single, combined or consortial schemes to which the module contributes

This course is compulsory for students on the MRes in Statistics and Operational Research (STOR-i). It is not open to other students.

Indicative Syllabus for Prospective Students (between 100 to 250 words)

This module will train students to produce scientific software that is replicable, reproducible, reusable, re-runnable, and repeatable. The course begins by introducing fundamental computer science concepts such as analysis of algorithms and data structures, several different of models of programming and design patterns, and the principles of concurrent programming.

The course then proceeds with how these ideas are applied in practice with particular reference to the software engineering practices associated with collaborative programming, software maintenance and support, testing and distribution of software.

During the lectures and workshops, emphasis is placed on problems and techniques associated with the fields of statistics and operational research. The course will teach in detail at least one programming language, such as Python, which supports the required programming paradigms.

Concise Bibliography

- Benureau, F. C. Y. and Rougier, N. P. (2018). Re-run, Repeat, Reproduce, Reuse, Replicate: Transforming Code into Scientific Contributions. Frontiers in Neuroinformatics. 11. https://doi.org/10.3389/fninf.2017.00069
- Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Clifford, S. (2009). Introduction to Algorithms (2nd Edition). MIT Press.
- Roy, P. V. and Haridi, S. (2004). Concepts, Techniques and Models of Computer Programming. MIT Press.
- Gamma, E., Helm, R., Johnson, R. and Vlissides, J. M. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional.

Syllabus Rules

Prior to enrolment on the module, the student must have successfully completed
Not specified

Educational Aims: Subject Specific

The aim of this course is to train students to produce scientific software that is replicable, reproducible, reusable, re-runnable, and repeatable. In particular, the following topics will be covered to support this:

- Advanced usage of at least one multi-paradigm programming language such as Python
- Asymptotic analysis of algorithms and introduction to common Abstract Data Types
- Overview of the main models of programming e.g. imperative, object-oriented and functional.
- Introduction to common design patterns e.g. strategy, factory, visitor patterns.
- Use of software engineering tools such as debuggers, profilers, version control systems and unit test systems, and where appropriate using these tools in a collaborative fashion.
- A basic understanding of modern computer architecture with particular reference to how this supports concurrent programming

Educational Aims: General

This module will enable to students to produce and collaborate on general coding projects. The module will also develop the students' skills in communicating and presenting design decisions.

Timing and method of formative and summative assessment feedback to students

The course has three assessments. The first two assessments will take place while the module is being taught while the final assessment is released after teaching activities have ended. The feedback for the first two assessments will be given to the students in a timely manner so that this can be taken into account for the following assessments. For all three assessments, students will receive written feedback, some of which may take the form of comments in their submitted code files.

In addition to the formal assessments, students will have the opportunity to receive oral feedback on their work during workshop sessions.

[Learning Outcomes] Learning Outcomes: Subject Specific

Students who pass this module should be able to

A1. to analyse an algorithm in terms of its computational complexity and determine appropriate data structures for its implementation in software

A2. to use the three main models of programming (imperative, object-oriented and functional), and identify applicable design patterns.

A3. to use software engineering tools such as profilers, debuggers, testing frameworks and environment management systems

A4: Understand basic computer architecture and operating systems and be able to parallelize code where appropriate

A5. to use tools and mechanisms for collaborative programming, and distribution and support of code within the wider research community

A6. to produce scientific software that is replicable, reproducible, reusable, re-runnable, and repeatable.

[Learning Outcomes] Learning Outcomes: General

Students who pass this module should be able to...

G1. Make informed choices regarding programming systems and software engineering tools

G2. Collaborate effectively on programming projects

G3. Produce and make available publicly quality software

G4. Communicate and justify the design and organisation of software and its components

Cohorts

Start	End	Assess Timing
13/01/2025	28/03/2025	End of Module

How will this module be taught?

None Specified

Timetabling Response

Timetabling response received?
No

Timetabling comments from Registry
None provided

Viability

Anticipated average number of students

12

What would you currently expect to be the maximum number of students able to enrol for this module?

20

How will this module be assessed?

Type	Status	Proportion	Default?
Coursework	Compulsory	50%	True
Presentation (Assessed)	Compulsory	10%	True
Groupwork	Compulsory	30%	True
Peer Assessment	Compulsory	10%	True

What are the methods by which the assessment will be moderated?

- Second marking

Details of assessment and rationale or strategy for the assessment methods

There are three main assessments for this course. Two smaller assessments will take place while the module is being taught to reinforce fundamental concepts and materials introduced in the earlier parts of the course, as well as presenting an opportunity to provide students with formative feedback. The final assessment, which has the highest weighting, will take the form of a large coding project of the type the student may have to undertake in their future research, and will assess all learning outcomes.

Details of each assessment are as follows.

1) Programming Assessment (A1, A2):

The first assessment requires students to demonstrate an application of fundamental concepts to make appropriate software design decisions for a given algorithm. Students will then implement their design using an appropriate programming language.

Weighting: 25%

Form: individual coursework with coding and short report

2) Software Engineering Assessment (A2-A4):

The second assignment requires studentsstudy, analyse and improve examples third-party code using appropriate software engineering tools. At least one example will include an element of parallelization.

Weighting: 25%

Form: individual coursework with coding and short report

3) Final Group Assessment (A1-A6):

The final coursework will consist of a programming reproduction exercise where groups will have to implement a methodology from the stats or OR literature as a package. This will require students to draw on all concepts and practices covered during the course. Group collaboration (through appropriate use of version control software) will be explicitly assessed here.

Weighting: 50%

Form: group work coursework with coding, presentation and peer assessment

Administration

Departmental Approval Granted
Not specified

Created by [McCrea, Rachel] on 15/04/2024 at 20:07
Last modified by [Fairbrother, Jamie] on 22/04/2024 at 17:27