Computational Communication Science 1 (6 ECTS)

Course File

dr. Joanna Strycharz
j.strcharz@uva.nl

Ognjan Denkovski
o.denkovski@uva.nl

Marieke van Hoof
m.vanhoof@uva.nl

College of Communication
University of Amsterdam

Academic Year 2021/22,
Semester 2, block 1

Contents

Chapter 1

Short description of the course (as in the course catalogue 2021-22)

1.1 Objectives

Upon completion of the course, students should:

1. Have a basic understanding of the use of programming languages for computational communication science and the practices behind open science.
2. Have a basic understanding of the most common data types in programming.
3. Have a basic understanding of simple control structures (loops, conditions and functions).
4. Have basic experience with exploring and visualizing different data types and executing basic statistical analyses using programming languages.
5. Have basic experience with handling errors in programming languages and debugging with the help of online resources.
6. Have basic experience with retrieving data using APIs and other sources of data.
7. Be able to independently seek, find and apply solutions for the desired operation in programming languages.

1.2 Contents

Recent developments in digital technologies have fundamentally changed our society and the ways in which we communicate, creating significant need for basic programming capacities and familiarity with the methods which allow for the analysis of these trends. This course provides a comprehensive introduction to the computational methods and practices that allow researchers and practitioners to study different phenomena in the digital society, while making use of the digital data generated by individuals through different daily activities.

The course starts with the assumption that students have no previous knowledge of computational methods. Students receive a comprehensive introduction to Python for data science, including best practices for data wrangling, generating basic descriptive and

inferential statistics, as well as data visualization in Python. Students are also introduced to the basic principles of APIs and version control with Git. After taking this course, students will have gained an understanding of various key concepts and techniques for data science with common programming languages for data analysis and will have hands-on experience with applying these techniques to a wide range of tasks.

1.3 Recommended prior knowledge

The course starts with the assumption that students have no previous knowledge of computational methods. However, it is important to note that the course discusses basic statistical methods relevant for digital data. The following book offers a comprehensive overview of and introduction to statistical methods used in communication science:

Hayes, A. F. (2020). *Statistical methods for communication science*. Routledge.

1.4 Teaching method and contact hours

The course is taught as one weekly lecture focused on explaining the key concepts in computational methods and one weekly tutorial session with hands-on exercises in which students directly apply the acquired skills. Students will also have the opportunity to receive programming support during online weekly office hours.

1.5 Study materials

Students in the course follow tutorials (online and in class) to learn how to perform analyses of digital data using Python. The following book will be used:

Van Atteveldt, W., Trilling, D., Arcilla, C. (in press). *Computational Analysis of Communication*. Hoboken, NJ: Wiley.

Additional literature and hands-on tutorials (programming scripts) will be provided. In some weeks, students will be provided with microlectures which will explore certain aspects of the week's material in greater depth.

1.6 Assesment

The course will use the following assessment methods:

1. Weekly knowledge questions covering the content for the week (15% of final grade)
2. Weekly assignments completed before the tutorials in which students practice the material of the week (35% of final grade)
3. Take-home exam in which students complete a data analysis challenge applying knowledge from the course (40% of final grade)

4. Active participation assessed via weekly warm-up exercises. To receive a participation grade, you will need to complete 5 our of 6 warm-up exercises with your best effort and upload them on Canvas on time. Each tutorial counts for 2 points of participation. (10% of final grade)

Chapter 2

Exit qualifications

The course contributes to the following exit qualifications of the Minor Communication in the Digital Society:

- Bekwaamheden en vermogens
- Probleemoplossend vermogen
- Academische houding

The exit qualifications are elaborated in the following four specifications:

- 1.b.3 Regarding specific topics, the graduate is able to understand, position and critically discuss not too complex scientific publications within the field. Is able to recognise and acknowledge the importance of contribution of other fields of study. Is able to adopt a position regarding scientific debate within the field.
- 4.b.2 Is able to assess research within the discipline on its practical usability.
- 4.b.3 Is able to recognise the relevance of other disciplines (interdisciplinarity).
- 4.b.9 Is able to deal with the field of study in a creative manner.
- 5.b.5 Is able to relative independently revise and expend knowledge through study.

Chapter 3

Testable objectives

The table of the following page provides an overview of the exit qualifications and related testable objectives of the course.

Exit qualifications & Testable objectives		1.b.3 Regarding specific topics, the graduate is able to understand, position and critically discuss not too complex scientific publications within the field. Is able to recognise and acknowledge the importance of contribution of other fields of study. Is able to adopt a position regarding scientific debate within the field.	4.b.2 Is able to assess research within the discipline on its practical usability.	4.b.3 Is able to recognise the relevance of other disciplines (interdisciplinarity).	4.b.9 Is able to deal with the field of study in a creative manner.	5.b.5 Is able to relative independently revise and expend knowledge through study.
A	Have a basic understanding of the use of programming languages for computational communication science and the practices of open science	X	X	X		
B	Have a basic understanding of the most common objects and data types in programming				X	
C	Have a basic understanding of simple control structures (loops, conditions and functions)				X	
D	Have basic experience with exploring and visualizing different data types and executing basic statistical analyses using programming languages				X	
E	Have basic experience with handling errors in programming languages and debugging with the help of online resources					X
F	Have basic experience with retrieving data using API's and other sources of data				X	
G	Are able to independently seek, find and apply solutions for the desired operation in programming languages				X	X

Chapter 4

Planning of testing and teaching

The course consists of fourteen meetings, two per week. Each week, in the first meeting, the instructor will give short lectures on the key aspects of the week, will demonstrate them through live-coding and host QA sessions with the students. In preparation for these meetings, students will read assigned literature. Their understanding of the key aspects of the weekly content will be tested through multiple choice questions.

The second meetings each week are practicum-meetings. Before each practicum-meeting the students will complete short warm-up coding exercises created specifically for this course. These formative assignments will be graded based on best effort. The students upload the results of the warm-up exercises on Canvas, allowing the lecturer team to assess the progress and the main questions/issues that students experienced before the practicum-meetings. In the first part of the practicum-meeting, the students will complete the weekly knowledge questions and the lecturer will discuss the most common errors and problems. In the second half of the practicum-meeting, the students will start working on the main coding challenges created specifically for each week of the course. Students have time during the tutorial to ask questions about the challenges and have until the end of the week to submit their answers on Canvas.

Students will also have the opportunity to receive programming support during weekly office hours hosted online each Friday. Check Canvas for the relevant information regarding how to join the office hours.

To complete the course, next to active participation, the students have to successfully complete three graded assignments: multiple-choice questions regarding the content of the week, weekly challenges completed during and after the practicum-meetings and a final take-home exam (see Chapter 7 for details).

Chapter 5

Literature

The schedule in chapter 6 gives an overview of the topics covered each week, the obligatory literature that has to be studied each week, and other tasks the students have to complete in preparation of the class.

In the course students read several chapters from the book *Computational Analysis of Communication* (van Atteveldt, Trilling, & Carlos, in press). The specific chapters are mentioned in the schedule. Some basic chapters that explain how to install the software we are going to use have to be read before the course starts.

Next to the book chapters, specific literature and materials on the topic of each week will be studied. Additionally, some weeks include optional readings that are not obligatory material, but are included as reference to provide the interested students with deeper knowledge. The overview of all required and additional literature is presented below as well as in the schedule.

- Lazer et al. (2020)
- van Atteveldt and Peng (2018)
- Hilbert et al. (2019)
- Van Atteveldt, Strycharz, Trilling, and Welbers (2019)
- Freelon (2018)

Chapter 6

Specific course timetable

The chapters mentioned in the schedule relate to van Atteveldt et al. (in press).

Before the course starts: Prepare your computer.

✔ Install Python and other necessary tools
Make sure that you have a working Python environment installed on your computer. You cannot start the course if you have not done so. Installation instructions will be provided on Canvas.

Week 1: What is Computational Communication Science, and why Python?

Lecture

We discuss what computational communication science is. We talk about challenges and opportunities as well as the implications of such developments for communication science in particular. We also pay attention to the tools used in CCS, in particular to the use of Python. Finally, we get an introduction to the tools used in class.

Mandatory readings (in advance):

- Computational Analysis of Communication, Chapter 1.2
- Computational Analysis of Communication, Chapters 3.1 until 3.1.4
- van Atteveldt and Peng (2018)

Tutorial

Before the tutorial, make sure to try to complete the warm-up exercises and turn in your solutions on Canvas. The tutorial will start with discussing the most common mistakes in the warm-up exercises. Next, there will be time for any questions you have on the material of the week. The final part of the tutorial will be spent on the challenges of the week. The tutorial lecturer will be there to answer all your questions.

Week 2: Control flow tools, version control and open science

Lecture

We discuss the notion of open science and its relevance to computational communication science. We also focus on key aspects of reproducible computational research workflows and get introduced to the relevant tools.

In the second part of the lecture, we focus on programming simple control structures such as loops and conditions.

Mandatory readings (in advance):

- Computational Analysis of Communication, Chapter 3.2
- Van Atteveldt et al. (2019)
- Github workshop - Creating and cloning `https://github.com/vanatteveldt/github-workshop/blob/main/tutorials/creating_and_cloning.md`
- Github workshop - Add, commit, push, pull `https://github.com/vanatteveldt/github-workshop/blob/main/tutorials/add_commit_push_pull.md`

Tutorial

Before the tutorial, make sure to try to complete the warm-up exercises and turn in your solutions on Canvas. The tutorial will start with discussing the most common mistakes in the warm-up exercises. Next, there will be time for any questions you have on the material of the week. The final part of the tutorial will be spent in the challenges of the week. The tutorial lecturer will be there to answer all your questions.

Week 3: Data types, basic pre-processing and handling errors

Lecture

We discuss the most common data types used in computational communication science. We focus on advantages and disadvantages of different data structures and discuss how to choose the most appropriate structure for different types of data. Finally, we get introduced to writing your own functions in Python.

Mandatory readings (in advance):

- Computational Analysis of Communication, Chapter 3.3
- Computational Analysis of Communication, Chapter 4.2
- Computational Analysis of Communication, Chapter 5

Tutorial

Before the tutorial, make sure to try to complete the warm-up exercises and turn in your solutions on Canvas. The tutorial will start with discussing the most common mistakes in the warm-up exercises. Next, there will be time for any questions you have on the material of the week. The final part of the tutorial will be spent on the challenges of the week. The tutorial lecturer will be there to answer all your questions.

Week 4: Data exploration and wrangling

Lecture

We get an introduction to data exploration and wrangling steps. We will learn how to do data wrangling with Pandas: subsetting datasets, aggregating data, joining datasets, and so on.

Mandatory readings (in advance):

- Computational Analysis of Communication, Chapter 6 (excluding 6.5)

Tutorial

Before the tutorial, make sure to try to complete the warm-up exercises and turn in your solutions on Canvas. The tutorial will start with discussing the most common mistakes in the warm-up exercises. Next, there will be time for any questions you have on the material of the week. The final part of the tutorial will be spent on the challenges of the week. The tutorial lecturer will be there to answer all your questions.

Week 5: Visualising and analysing data

Lecture

We discuss best practices for visualising different types of data and get introduced to some of the most common packages used for data visualisation in Python. We also get an introduction to conducting basic statistical analyses in Python (note that students are expected to have understanding of most common descriptive and inferential statistical methods). Finally, the lecture includes a QA and a catch-up on content from the first four weeks of the course.

Mandatory readings (in advance):

- Computational Analysis of Communication, Chapter 7.1, 7.2
- Make sure to prepare all your questions for the QA catch-up session.

Tutorial

Before the tutorial, make sure to try to complete the warm-up exercises and turn in your solutions. The tutorial will start with discussing the most common mistakes in the warm-up exercises on Canvas. Next, there will be time for any questions you have on the material of the week. The final part of the tutorial will be spent on the challenges of the week. The tutorial lecturer will be there to answer all your questions.

Week 6: APIs and working with your own data

Lecture

We discuss the conceptual principles of Application Programming Interfaces (APIs), their relevance as data access services and their continued relevance as frameworks for accessing complex, state-of-the-art (machine learning) models.

In the second part of the class, we provide a practical introduction to the Twitter API and discuss research ideas for Twitter/other API data based on the weekly literature. Students develop a simple research idea for their own custom Twitter dataset.

Mandatory readings (in advance):

- Computational Analysis of Communication, Chapter 12.1
- Computational Analysis of Communication, Chapter 12.4
- Freelon (2018)

Tutorial

Before the tutorial, make sure to try to complete the warm-up exercises and turn in your solutions on Canvas. The tutorial will start with discussing the most common mistakes in the warm-up exercises. Next, there will be time for any questions you have on the material of the week. The final part of the tutorial will be spent on the challenges of the week. The tutorial lecturer will be there to answer all your questions.

Week 7: CCS — looking ahead

Lecture

In this final week, we discuss emerging trends that have come to dominate working with computational approaches. We discuss the implications of the move towards a post-API age for researchers based in computational approaches. Here, we provide a brief demonstration of web scrapping as an ad-hoc solution to the decreasing availability of 'free' data and discuss the relevance of ethical considerations while collecting custom data online.

In the second part of the class, we discuss the increasing relevance of cloud infrastructure approaches and database management as basic tools for robust business and academic approaches for managing large data sets and optimizing insights.

Mandatory readings (in advance):

- Computational Analysis of Communication, Chapter 15.1 15.2
- Hilbert et al. (2019)

Tutorial

This tutorial will serve as a workshop which will demonstrate the various phases of implementing a simple cloud-based data collection and data writing system. During this workshop, you will be introduced to the principles of hosting (Python) functions on cloud services, setting-up a database, and writing content to that database with your (Python) script. The workshop will demonstrate this process on a specific cloud service, but the principles and components shown translate to any of the major cloud services, including Amazon Web Services (AWS), Microsoft Azure, Google Cloud, etc.

The workshop should serve as an introduction to the accessibility of cloud computing as an incredible resource for researchers looking for greater computing power and virtually limitless storage.

Chapter 7

Testing

An overview of the testing is given in Table 7.1.

Grading

The final grade of this course will be composed of the grade of grades for weekly knowledge questions (15%), weekly coding challenges (35%), a final take-home exam (40%) and active participation (10%).

Weekly knowledge questions (15%)

The weekly knowledge questions will be included at the beginning of tutorials and will test students knowledge of the constructs introduced in the literature and the lecture. Each week, students will answer 6 knowledge questions.

Weekly coding challenges (35%)

The weekly assignments consist of three challenges in which students need to apply the knowledge gained in the week. They will start working on the assignments in the practicum-meetings and will turn them in on Canvas at the end of each week.

Take-home exam (40%)

At the end of the course, students will complete a take-home coding exam. Grading criteria are communicated to the students together with the exam, but in general are:

- correctness of the code,
- correctness, completeness, and usefulness of pre-processing steps applied,
- correctness, completeness, readability and usefulness of visualisations chosen,
- correctness, completeness, and usefulness of analyses applied,
- correctness and completeness explanations provided.

Table 7.1: Test matrix

	Weekly knowledge questions (15% of final grade)	Weekly knowledge questions (35% of final grade)	Take-home exam (40% of final grade)	Participation (10% of final grade)
A. Have a basic understanding of the use of programming languages for computational communication science and the practices of open science	X	X	X	X
B. Have a basic understanding of the most common objects and data types in programming	X	X	X	X
C. Have a basic understanding of simple control structures (loops, conditions and functions)	X	X	X	X
D. Have basic experience with exploring and visualizing different data types and executing basic statistical analyses using programming languages		X	X	X
E. Have basic experience with handling errors in programming languages and debugging with the help of online resources		X	X	X
F. Have basic experience with retrieving data using API's and other sources of data		X		X
G. Are able to independently seek, find and apply solutions for the desired operation in programming languages		X	X	X

Participation (10%)

In preparation for each week's practicum meetings, the students will complete warm-up exercises. To receive participation grade, students will need to complete 5 our of 6 warm-up exercises with best effort and upload them on Canvas on time. Each tutorial counts for 2 points of participation.

Grading and 2nd try

A final assessment is possible only if all assignments have been submitted according to the criteria and on time and students have met the compulsory attendance requirement and actively participated in the sessions. This means that:

- Students have to get a pass (5.5 or higher) for the final exam. If the grade is lower, an improved version can be handed in within one week after the grade is communicated to the student. If the improved version still is graded lower than 5.5, the course cannot be completed. Improved versions of the final exam cannot be graded higher than 6.0.
- For the weekly knowledge questions students need to answer at least 36 questions correctly to get full marks. It is hence possible to miss one tutorial and still receive full marks for the knowledge questions. 20 questions need to be answered correctly to receive a pass (i.e., equivalent to a grade of 5.5 on a 10-point scale). Students that do not meet this threshold will need to submit an additional written assignment covering the material tested in knowledge questions.
- For the coding challenges portfolio, students must reach a total of 100 out of 180 points (i.e., equivalent to a grade of 5.5 on a 10-point scale) in their portfolio at the

end of the course. Students that do not meet this threshold will need to submit an additional coding assignment as a resit

Deadlines and late assignment policy

The due dates of all assignments are listed on Canvas before the course starts, so it is recommended to schedule them in your agenda, and plan accordingly.

- Warm-up exercises are due on the morning of the tutorial day (Thursday, 9AM) in weeks 1-6. This allows the lecturers to prepare the tutorial according to the group's needs. Students are expected to turn in their best effort try, but the answers do not need to be perfect.
- Weekly coding challenges (that together make up Coding Challenges Portfolio) are due at the end of the week (Friday, 5PM) in weeks 1-6. Students will receive feedback on their code the following week.
- Take-home exam will take place in week 8.

As a general principle, handing in an assignment after the deadline will be automatically considered a resit. Specifically, this means that:

- Submitting Assignment 1 after the deadline means that it will be automatically considered a resit
- The weekly coding challenges need to be delivered before the deadline in order for their points to be considered in your portfolio grade. They cannot be resubmitted or delivered late.
- The weekly warm-up exercises need to be delivered before the deadline in order to count towards participation grade. They cannot be resubmitted or delivered late.

Chapter 8

Lecturers' team, including division of responsibilities

Joanna Strycharz (coordinator) and Ognjan Denkovski are responsible for the weekly lectures. Marieke van Hoof will lead the weekly practicum meetings and provide feedback on weekly coding challenges.

Chapter 9

Calculation of students' study load (in hours)

- Elective total: 6 ECTS =168 hours
- Reading: 183 pages plus additional online sources. 6 pages per hour, thus about 45 hours for the literature
- Presence:
  14*2 hours: 28 hours.
- Practicum Q&A preparation:
  7*2 hour: 14 hours.
- Weekly warm-up exercises:
  7*3 hour: 21 hours.
- Weekly coding challenges:
  6*5 hour: 30 hours (excluding time during tutorial)
- Take-home exam, including preparation: 30 hours

Total: 168 hours

Chapter 10

Calculation of lecturers' teaching load (in hours)

10.1 Lecturers

- Presence: 14 hours (= 7 * 2 hours)
- Preparation of course (including necessary notebooks): 26 hours
- Preparation of weekly lectures and knowledge questions, 7 * 3 hours: 21 hours
- Preparation of weekly challenges, 7 * 2 hours: 14 hours
- Preparation of final exam (incl. necessary datasets): 10 hours
- Feedback and grading final exam: 50 * 25 minutes: 23 hours
- Administration, e-mails, individual appointments: 7 hours

Total: 115 hours

10.2 Tutorial lecturer

- Presence: 14 hours (= 7 * 2 hours)
- Preparation of weekly tutorial meetings including evaluating warm-up excercises, 7 * 2.6 hours: 18 hours
- Feedback and grading weekly challenges, 6 * 4 hours (10 minutes per student): 24 hours

Total: 56 hours

Literature

Freelon, D. (2018). Computational research in the post-api age. *Political Communication*, *35*(4), 665–668.

Hilbert, M., Barnett, G., Blumenstock, J., Contractor, N., Diesner, J., Frey, S., ... others (2019). Computational communication science: A methodological catalyzer for a maturing discipline.

Lazer, D. M., Pentland, A., Watts, D. J., Aral, S., Athey, S., Contractor, N., ... others (2020). Computational social science: Obstacles and opportunities. *Science*, *369*(6507), 1060–1062.

van Atteveldt, W., & Peng, T.-Q. (2018). When communication meets computation: Opportunities, challenges, and pitfalls in computational communication science. *Communication Methods and Measures*, *12*(2-3), 81–92.

Van Atteveldt, W., Strycharz, J., Trilling, D., & Welbers, K. (2019). Computational communication science— toward open computational communication science: A practical road map for reusable data and code. *International Journal of Communication*, *13*, 20.

van Atteveldt, W., Trilling, D., & Carlos, A. (in press). *Computational analysis of communication: a practical introduction to the analysis of texts, networks, and images with code examples in python and r.* John Wiley Sons.