# Computational Communication Science 1 (6 ECTS)

## Course File

dr. Joanna Strycharz
j.strcharz@uva.nl

dr. Frederick Hopp
f.r.hopp@uva.nl

dr. Justin Ho
j.c.ho@uva.nl

Fabio Votta Kiddle
f.a.votta@uva.nl

College of Communication
University of Amsterdam

Academic Year 2023/24,
Semester 2, block 1

# Contents

# Chapter 1

# Short description of the course (as in the course catalogue 2023-24)

## 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 used for preparation for tutorials and exercises for each week) 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 final grade of this course will be composed of the grade of grades for weekly knowledge questions (15%), code peer review (10%), in-class programming assignments (30%) and a final take-home exam (45%).

1. Weekly knowledge questions covering the content for the week
2. Tutorial participation - code peer review

3. Programming assignments in which students complete two programming challenges to show understanding of course material

4. Final exam in which students complete a data analysis challenge applying knowledge from the course

# Chapter 2

# Exit qualifications

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

- Skills and abilities
- Problem-solving ability
- Academic thinking

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<br>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<br>Is able to assess research within the discipline on its practical usability. | 4.b.3<br>Is able to recognise the relevance of other disciplines (interdisciplinarity). | 4.b.9<br>Is able to deal with the field of study in a creative manner. | 5.b.5<br>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 and a weekly office hour. Each week, in the first meeting, the instructor will give lectures on the key aspects of the week, will demonstrate them through live-coding and host Q&A 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. The students can use these exercises to assess their progress and prepare questions/issues that they would like to discuss in detail in the practicum-meeting. 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 based on student questions about warm-up exercises. In the second half of the practicum-meeting, the students will start working on the coding challenges created specifically for each week of the course. The students will code in pairs (pair programming). Students have time during the tutorial to ask questions about the challenges. Example answers to the challenges will be provided at the beginning of the following week.

Students will also have the opportunity to receive programming support during weekly office hours hosted on campus after the lecture. Check Canvas for the relevant information regarding how to join the office hours.

To complete the course the students have to successfully complete three graded assignments: multiple-choice questions regarding the content of the week, in-class coding assignments 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, & Arcíla Calderon, 2022). 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. You can buy a hardcopy of the book in any bookstore, but you can also access its content for free at `https://cssbook.net/` – that's up to you.

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.

- 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. (2022).

## 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. Next, Turn in your feedback for a classmate on their warm-up exercises. The tutorial will start with discussing the most common difficulties in the warm-up exercises based on student input. 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 pair programming 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 a broad introduction 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)

## Tutorial

Before the tutorial, make sure to try to complete the warm-up exercises and turn in your solutions. Next, Turn in your feedback for a classmate on their warm-up exercises. The tutorial will start with discussing the most common difficulties in the warm-up exercises based on student input. 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 pair programming challenges of the week. The tutorial lecturer will be there to answer all your questions.

# Week 3: Basic pre-processing and handling errors

## Lecture

First, we get introduced to writing your own functions in Python. We also discuss how to understand errors and what are good strategies of debugging your code. Finally, the lecture includes a QA and a catch-up on content from the first two weeks of the course.

Mandatory readings (in advance):

- Computational Analysis of Communication, Chapter 3.3
- Computational Analysis of Communication, Chapter 4.2
- Make sure to prepare all your questions for the QA session.

## Tutorial

In this tutorial, we will have our first in-class assignment. Makes sure to prepare for it revising materials from the first two weeks. Also, make sure to try to complete the warm-up exercises for the week. We will discuss the most common difficulties in the warm-up exercises based on student input. 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 programming challenges of the week. The tutorial lecturer will be there to answer all your questions.

# Week 4: Data types and wrangling and programming assignment

## 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. In the second part of the lecture, we get an introduction to data exploration and wrangling steps. We will learn how to do data wrangling with Pandas: dealing with missing values, subsetting datasets, joining datasets, and so on.

Mandatory readings (in advance):

- Computational Analysis of Communication, Chapter 5
- Computational Analysis of Communication, Chapter 6 (excluding 6.3 and 6.5)

## Tutorial

In this tutorial, we will have our second in-class assignment. Makes sure to prepare for it revising materials from week three Also, make sure to try to complete the warm-up exercises for the week. We will discuss the most common difficulties in the warm-up exercises based on student input. 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 programming challenges of the week. The tutorial lecturer will be there to answer all your questions.

# Week 5: Visualising and analysing data

## Lecture

We start with data exploration and will continue with discussing 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).

Mandatory readings (in advance):

- Computational Analysis of Communication, Chapter 6.3
- Computational Analysis of Communication, Chapter 7.1, 7.2

## Tutorial

Before the tutorial, make sure to try to complete the warm-up exercises and turn in your solutions. Next, Turn in your feedback for a classmate on their warm-up exercises. The tutorial will start with discussing the most common difficulties in the warm-up exercises based on student input. 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 pair programming challenges of the week. The tutorial lecturer will be there to answer all your questions.

# Week 6: Working with text as data and APIs

## Lecture

A lot of online data that communication scientists encounter are textual (social media posts, reviews, news articles, ...) rather than numerical. In this week, we focus on how to deal with such data. We also address how such data can be collected via so-called application programming interfaces (APIs).

Mandatory readings (in advance):

- Computational Analysis of Communication, Chapter 12.1
- Computational Analysis of Communication, Chapter 12.4
- Computational Analysis of Communication, Chapter 9 (introduction + 9.1)
- Computational Analysis of Communication, Chapter 14.1

## Tutorial

The tutorial will start with the third programming assignment. Make sure to revise Pandas to prepare for oit 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 programming 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 will recap the material of the class and have a final QA on class material.

Mandatory readings (in advance):

- Freelon (2018)
- Hilbert et al. (2019)

## Tutorial

In the final tutorial students will work on a trial exam. This will allow them to test their own knowledge and identify difficulties. The tutorial lecturer will be there to answer all your questions.

# 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%), code peer review (10%), in-class programming assignments (30%) and a final take-home exam (45%).

### Weekly knowledge questions (15%)

The weekly knowledge questions will be included at the beginning of tutorials in weeks 1, 2, 5 and 7 and will test students knowledge of the constructs introduced in the literature and the lecture. Each time, students will answer 6 knowledge questions.

### Tutorial participation - code peer review (10% of the grade)

Students will be paired with a classmate and will be asked to provide feedback and tips on their classmate's warm-up coding exercises in weeks 1, 2 and 5. The feedback will be graded with pass/fail.

### In-class programming assignments (30%)

The course involves three programming assignments. The first assignment consists of a challenge in week three in which students need to apply the knowledge gained in the first two weeks of the course. It accounts for 30% of in-class assignment grade. The second assignment consists of a challenge in week four in which students need to apply the knowledge gained in the third week of the course. It accounts for 30% of in-class assignment grade. The third assignment focuses on working with dataframes and takes place in the sixth week of the course. It accounts for 40% of in-class assignment grade. Students will complete the assignments in the tutorial meeting.

### Exam (45%)

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

Table 7.1: Test matrix

| | Weekly knowledge questions (15% of final grade) | Code review (10% of final grade) | In-class programming assignments (30% of final grade) | Take-home exam (45% 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 | | |
| G. Are able to independently seek, find and apply solutions for the desired operation in programming languages | | X | X | X |

- 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.

## 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:

- For the in-class programming assignment, students have to get a pass (5.5 or higher) on both assignments. Students that do not meet this threshold will need to re-sit the assignment at a time agreed with lecturers.
- 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 answers in best four weeks out of five weeks of the course count toward the grade. It is hence possible to miss one tutorial and still receive full marks for the knowledge questions. From the best four weeks, 14 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 weekly participation (peer feedback), students who complete two out of three feedback assignments will receive full marks.

## 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.

- Weekly knowledge questions will be completed at the beginning of tutorials in weeks 1, 2, 5 and 7.
- Peer feedback will be completed before tutorials in weeks 1, 2 and 5.
- In-class programming assignments will take place in week 3, 4 and week 6.
- Exam will take place in week 8.

As a general principle, handing in an assignment after the deadline will be automatically considered a resit.

# Chapter 8

# Lecturers' team, including division of responsibilities

In this edition, the Computational Communication Science course will be taught by Joanna Strycharz, Frederic Hopp, Fabio Votta  Justin Ho. Joanna  Frederic will teach the lectures, while Fabio and Justin will work with you in the tutorials. Joanna is the course coordinator.

# 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*4 hour: 28 hours.
- Preparation for programming assignments:
  7 hours.
- Weekly coding practice:
  6*5 hour: 30 hours (excluding time during tutorial).
- 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, including programming assignments 7 * 2 hours: 14 hours
- Preparation of final exam (incl. necessary datasets): 10 hours
- Administration, e-mails, individual appointments, office hours: 11 hours

Total: 96 hours

## 10.2 Tutorial lecturer (per group)

- Presence: 14 hours (= 7 * 2 hours)
- Preparation of weekly tutorial meetings, 7 * 4.7 hours: 33 hours
- Grading in-class assignments, group feedback (20 minutes per student): 9 hours
- Feedback and grading final exam: 25 * 30 minutes: 12 hours

Total: 68 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.

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., & Arcíla Calderon, C. (2022). *Computational analysis of communication: a practical introduction to the analysis of texts, networks, and images with code examples in python and r*. Hoboken, NJ: John Wiley & Sons.