Suggestions for Software, Tools & Services

You are free to work with all tools and technologies that you prefer, and your choices do not affect the grading. Being safe and comfortable with the tools you already know is fine. Using very technical, high difficulty solutions does not necessarily improve your grades. More complex/out-of-the-box tools/methods may result in higher quality deliverables, which could improve grades. However, if you choose to work with more obscure tools or programming languages, we will be able to offer less assistance. The definition of obscure here is 'Anything that we are able to offer less assistance with'.

Note that for all the suggestions mentioned below there are at least 25 alternatives with their own quirks, pros and cons. In addition to the general features and suitability of the software, realize that community support, online resources, manuals and tutorials are all way easier to find and available if you stick to the more popular solutions.

General advice in terms of software and services:

- Make a conscious choice for certain tools to organize your work and do the analyses: Do the members of your team have different operating systems and are they all supported? Do people have previous experience with tools and are they able to help you set it up quickly? Is everybody comfortable with using certain tools in light of ease of use, privacy policies, the need to create accounts, and the work it takes to set up?
- Do not go for the tools with the most features, but the tools with enough features: To organize work, a plain text file in a cloud service could already get you quite far, but is not ideal. In contrast Zoho Connect as a workflow tool is very advanced with hundreds of features to organize work between independent companies or in multinationals, which may lead to information being spread very thinly in different locations and cause miscommunication.
- Take, and formalize, the time to have proper administration and communication. It is part of this course. All team members programming and working on deliverables independently will invite stress, annoyance, doing work double, and confusion, which could have been avoided with a two-minute conference call. Select and integrate everything you need to do to counter this early on in the project so it becomes a habit to keep your group members up to date and communicate well.

Communication & Information Sharing

- WhatsApp / Signal / Telegram: The obvious choices, but they are only recommended as supplementary ways to communicate given the alternatives that are available nowadays. They all have very limited features and focus on different things. E.g. scrolling through group-chats in messengers to gather all relevant information is highly inefficient and tedious.
- Microsoft Teams: We provide a Microsoft Teams environment through the TU/e license. You will have a private channel for your group that allows you to schedule meetings, keep an agenda, share files, and have persistent chats. Next to the possibly useful Office and Outlook integration it might also be useful to stick to this environment because it allows lecturers and teachers direct access to your files and workflow organization: Not to snoop on you, but to help you work and organize this may have benefits.
- Slack: Allows you to setup a central space to gather all information. Persistent group-chat, private messaging, different channels for different topics, voice- and videoconferencing with screen-sharing or streaming, and document sharing are all integrated. It also has some integration with

several services to share programming code such as Git. (Do keep in mind that a lot of features are time or volume limited with freemium pricing).

- Discord: Highly similar to Slack with a few added features (multiple persistent voice channels, better link and file sharing including previews, integration for some niceties outside of the actual project such as Spotify). It seems the less obvious choice because it is fully marketed towards the online gaming community. However, it is easy to setup and has far fewer freemium limitations than Slack.

Overall, it does not really matter what you use, as long as you use it intelligently. E.g. Do not leave important information sitting in the middle of a long group-chat, but list important information and decisions separately. Make sure all your group-members can join discussions and calls. Have a place where you can leave important remarks, suggestions, or thoughts outside of meetings.

Workflow / Project Management

- Trello: Probably the most widely used way to subdivide tasks and keep track of progress. Its simplicity, almost endless scalability and intuitive interface make it very good to manage a development sprint with a scrum board.
- Asana: Very similar to Trello, but with a lot more flexibility and features. This may actually be a downside, because it is easy to get lost in the feature set if you just want to setup a simple board.

Overall, pick one primarily for its ease of use. You will need to keep it up-to-date often, and the risk of people not doing this becomes smaller if it is very easy and quick to do. You do not need a lot of features to keep a task list.

Technical: Databases & Data Management

- MySQL + GUI: To some extent the mother of all relational database managers. You already have worked with SQL, so it is a safe bet where you know all group-members have at least some experience. This assures that all group members can help think about solutions to problems, or ways to investigate and use the data, even when the programming becomes difficult. A simple GUI like DBBrowser to manually look at the data, features, and observations is highly recommended. Although not strictly necessary when using Python or another programming language to interface with the database, being able to visually inspect the data can be very useful and gives a way better understanding of what is going on. However, do note that SQL only allows very simple, flat data structures, which may make setting up, structuring and querying Twitter data more challenging than you expect.
- MongoDB: A popular NoSQL database manager. There are many alternatives, but MongoDB in particular has quite an active community and good online resources. As a general rule, these types of databases allow a lot more flexibility to search, synergize and subdivide data. They do not rely on the table / matrix system of a more traditional relational database system, but on key-values and meta-data given to cases, or selections of units in collections. However, this flexibility comes at the cost of ease of use. You, generally, are not used to these types of data organization and it takes some getting used to after working with SQL databases. Once you understand the system and the way in which the client works, they often do allow you to work and test things quicker. Also, they are well equipped to deal with data from JSON files.
- Neo4j: A graph database management system. Graph databases are relatively new compared to SQL and NoSQL, but they also provide new ways to interact with your data, especially when it

concerns relationships between elements. This may prove very interesting when discovering complex sub-structures in your data. The downside to this and other highly specialized database systems is that it will require more time to gain familiarity with them.

Overall, pick something you are comfortable with. If you have the knowledge and confidence to go outside your comfort zone, you could learn a lot from taking a new approach. However, the course is short, and the assignment is very big: Wasting a lot of time on loading and structuring the data is not recommended, so it is similarly very sensible to stay on the safe side and work with the systems you have some experience with. Whatever you choose, you will need to find a challenge at some point in this course.

Technical: Analyses & Programming

- Python + GUI: A very popular programming language due to its relative simplicity. If you have never written a line of code before, this is probably the language to start in. Due to its popularity it has a huge amount of community support and can be extended with many freely available libraries and functions. The downside is that it has a lot of quirks and seemingly confusing characteristics at times, and it is quite slow. However, there are third-party packages for pretty much any analysis and data manipulation that you would want to perform. A good development environment and GUI is strongly recommended as it allows you to keep closer track of what is happening. The integrated IDE is decent, but you may want to look at alternatives such as Spyder and Jupyter.
- R + GUI: A very popular, open source programming environment that you may be familiar with. It has been the gold standard in the field of statistics for a long time and is becoming increasingly popular in Data Science and Data Mining applications. The downside is that it has a lot of quirks and seemingly confusing characteristics at times, and it is quite slow. However, there are third-party packages for pretty much any analysis and data manipulation that you would want to perform. A good development environment and GUI like RStudio or Rkward is strongly recommended as it allows you to keep closer track of what is happening.
- Note that combinations are also possible: R allows you to call and run C, C++ and Python code and vice versa: If you have a good or fast solution to e.g. load the data in Python, but want to analyze the data in R that is possible.

Overall, any sensible coding and development environment for the language you are most comfortable with will work. But do keep in mind your group members! You can be very fluent in C++ or Assembly, but to explain code and work together a language like Python, R or possibly C is often way more beneficial. Moreover, the number of libraries, tips, tricks, extensions and plugins for R and Python is unrivaled, which means that most of the problems you face have already been solved and the solution is readily found with a quick search.

Code Sharing & Version Control

- GitHub: The most well-known version control service out there. It uses the Git system to keep track of code history. At any point people can update the code in a repository, previous versions are retained, and it is possible to develop an intuitive version notation. Branches and forks of code can be split off from the main project to try different things without interfering with the latest, accepted code. The only downside is that version control of software is a specific skillset in its own right, and you will need to train yourself to use a system like Git properly, yet, that again is a very valuable skill to pick up during your career.

- There are many alternatives integrating the Git system or similar, but virtually all of them have freemium pricing that would not allow enough users or development time. Some free alternatives like SourceForge have the goal to distribute open-source software, where their features would allow code checking, cooperation and version control, but running a private project like this would violate the terms of service.

Overall, some type of version control is highly recommended. It does not have to be an online service or advanced structure like Git, but you should try to avoid the (very relatable) situation where your final analyses end up in the file *FinalV0.9-finalfinal18May[editJohn]June2ndtemporary~debug.syntax*. For smaller projects like this one it may be overkill to use the extensive features of a forge, or git repository, but with a number of people working on the same code you quickly lose sight of progress. If you share the code just through e.g. a cloud service it would still be smart to follow some of the practices from software development by having a main branch, separate any untested follow-up work in a dev branch, and try to keep a central place for reporting possible errors or things you still want to work on. (You could manage this with a proper file and folder structure in Teams, as long as you lay out some rules).

- Supplementary: Codeshare.io is a free way to interactively share code (with syntax highlighting for all major languages) on the fly. It allows you to save bits of code for a day, and work on them together in real time. It also offers integrated video calling to do this.
- Supplementary: Pastebin (or any of its competitors) allows you to quickly share copy-and-paste text or code with syntax highlighting.

Additional Remarks

- Depending on the choice for tools and software, there are some packages that offer a combined install of several tools, which would allow you to set up and install your environment relatively quickly.
    - Anaconda for example offers a single install for Python, Spyder, Jupyter, RStudio, SQL, and the visualization tools Orange 3 and Glueviz.
- Some of the recommended software will have a client-server structure: This does *not* mean that you have to necessarily rent or own a server. The installs of such software will generally allow you to act as if your own PC is the server and “connect” to that server instance.
- All course communication with teachers, lecturers and tutors will take place via Canvas and Microsoft Teams.