

# Big Data and Automated Content Analysis (12EC)

Week 1: »Programming for Computational  
(Communication|Social) Scientists«

Wednesday

---

Damian Trilling

d.c.trilling@uva.nl, @damian0604

February 8, 2023

UvA RM Communication Science

# Today

Getting to know each other

What is computational [social|communication] science?

Defining CCS

And Big Data?

The role of software in CSS

What is Python?

Python: A language, not a program

Python versus R

What do we need to write Python programs?

Expectation Management

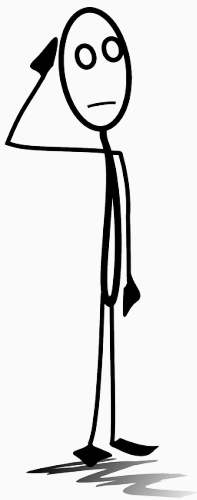
Next steps

# Getting to know each other

---



# You



Your name?

Your background?

Your reason to follow this course?

What is CSS|CCS?

---

# What is CSS|CCS?

---

Defining CCS

5



# Epistemologies and paradigm shifts

## Kitchin (2014)

- (Reborn) empiricism: purely inductive, correlation is enough
- Data-driven science: knowledge discovery guided by theory
- Computational social science and digital humanities: employ Big Data research within existing epistemologies
  - DH: descriptive statistics, visualizations
  - CSS: prediction and simulation

# Epistemologies and paradigm shifts

## Kitchin (2014)

- (Reborn) empiricism: purely inductive, correlation is enough
- Data-driven science: knowledge discovery guided by theory
- Computational social science and digital humanities: employ Big Data research within existing epistemologies
  - DH: descriptive statistics, visualizations
  - CSS: prediction and simulation

# Epistemologies and paradigm shifts

## Kitchin (2014)

- (Reborn) empiricism: purely inductive, correlation is enough
- Data-driven science: knowledge discovery guided by theory
- Computational social science and digital humanities: employ Big Data research within existing epistemologies
  - DH: descriptive statistics, visualizations
  - CSS: prediction and simulation

# Epistemologies and paradigm shifts

## Kitchin (2014)

- (Reborn) empiricism: purely inductive, correlation is enough
- Data-driven science: knowledge discovery guided by theory
- Computational social science and digital humanities: employ Big Data research within existing epistemologies
  - DH: descriptive statistics, visualizations
  - CSS: prediction and simulation

# CCS as a subset of CSS

## Hilbert et al. (2019)

“...our definition of computational communication science as an application of computational science to questions of human and social communication. As such, it is a natural subfield of computational social science” (followed by references to CSS definitions)

# Data, analysis, theory

## van Atteveldt and Peng (2018)

“...computational communication science studies generally involve: (1) large and complex data sets; (2) consisting of digital traces and other “naturally occurring” data; (3) requiring algorithmic solutions to analyze; and (4) allowing the study of human communication by applying and testing communication theory.”

# What is CSS|CCS?

---

## And Big Data?

It was a buzzword when we first designed this course in 2013 (very much like “AI” and “algorithm” are today).

It’s hard to define, but it can help us thinking about the characteristics of the data we deal with.



# The “pragmatic” definition

Everything that needs so much computational power and/or storage that you cannot do it on a regular computer.

# The “commercial” definition

## Gartner (n.d.)

“Big data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation.”

# The “critical” definition

## Boyd and Crawford (2012)

“

1. Technology: maximizing computation power and algorithmic accuracy to gather, analyze, link, and compare large data sets.
2. Analysis: drawing on large data sets to identify patterns in order to make economic, social, technical, and legal claims.
3. Mythology: the widespread belief that large data sets offer a higher form of intelligence and knowledge that can generate insights that were previously impossible, with the aura of truth, objectivity, and accuracy.

”



*Do you think that now, a decade later, Boyd and Crawford's argument is still valid? Or does it need to be adapted?*

## We have become more critical about it!

Reading tips: Bender et al. (2021), Bolukbasi et al. (2016).

These articles are about techniques using big datasets that we will talk about towards the end of this course, so it's OK if you don't understand them fully (yet).



1. *What do you think? What is the essence of Big Data/CSS/CCS?*
2. *How will what we do here relate to theories and methods from other courses?*

# What is CSS|CCS?

---

The role of software in CSS

# CSS means also a shift in our relationship to software

In the (traditional) social sciences, we tend to think of software as...

- a monolithic block: a large standalone programme like Word or SPSS;
- something that either fulfills your needs or doesn't: it's often impossible, but at least uncommon and difficult, to adapt or change it;
- something with a graphical user interface;
- (often) something you pay for.



# CSS means also a shift in our relationship to software

In CSS, that's (typically) different:

- We mix and combine different “modules” or “libraries” ( $\approx$  building blocks for programs)
- We build on modules made by others so that we don't reinvent the wheel, but ultimately build our own programs.
- We use a programming language to do all of this.

# Why program your own tool?

## Vis (2013)

“Moreover, the tools we use can limit the range of questions that might be imagined, simply because they do not fit the affordances of the tool. Not many researchers themselves have the ability or access to other researchers who can build the required tools in line with any preferred enquiry. This then introduces serious limitations in terms of the scope of research that can be done.”

# Some considerations regarding the use of software in science

Assuming that science should be *transparent* and *reproducible by anyone*, we should

use tools that are

- platform-independent
- free (as in beer and as in speech, gratis and libre)
- which implies: open source

This ensures it can our research (a) can be reproduced by anyone, and that there is (b) no black box that no one can look inside. ⇒ ongoing open-science debate! (van Atteveldt et al., 2019)

# Some considerations regarding the use of software in science

Assuming that science should be *transparent* and *reproducible by anyone*, we should

## use tools that are

- platform-independent
- free (as in beer and as in speech, gratis and libre)
- which implies: open source

This ensures it can our research (a) can be reproduced by anyone, and that there is (b) no black box that no one can look inside. ⇒ ongoing open-science debate! (van Atteveldt et al., 2019)

# Some considerations regarding the use of software in science

Assuming that science should be *transparent* and *reproducible by anyone*, we should

## use tools that are

- platform-independent
- free (as in beer and as in speech, gratis and libre)
- which implies: open source

This ensures it can our research (a) can be reproduced by anyone, and that there is (b) no black box that no one can look inside. ⇒ ongoing open-science debate! (van Atteveldt et al., 2019)

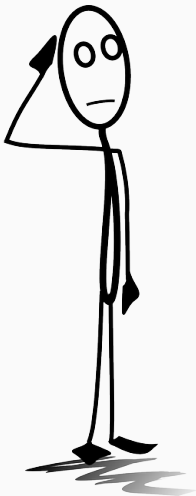
# Why program your own tool?

## Vis (2013)

“[...] these [commercial] tools are often unsuitable for academic purposes because of their cost, along with the problematic ‘black box’ nature of many of these tools.”

## Mahrt and Scharkow (2013)

“[...] we should resist the temptation to let the opportunities and constraints of an application or platform determine the research question [...]”



*Do you think one needs a  
programming language to do CSS?  
Why?*

## So I need a programming language to do CSS?

- In principle, it does not matter *which* programming language you use.
- **Python** and **R** are by far the most popular languages *for CSS*
- But that may change: Nowadays, most scientific articles are written in English – in some fields, that used to be Latin, German, Russian

If you learn one programming language, it is relatively easy to learn another one. (Think of learning Spanish after you learned French)



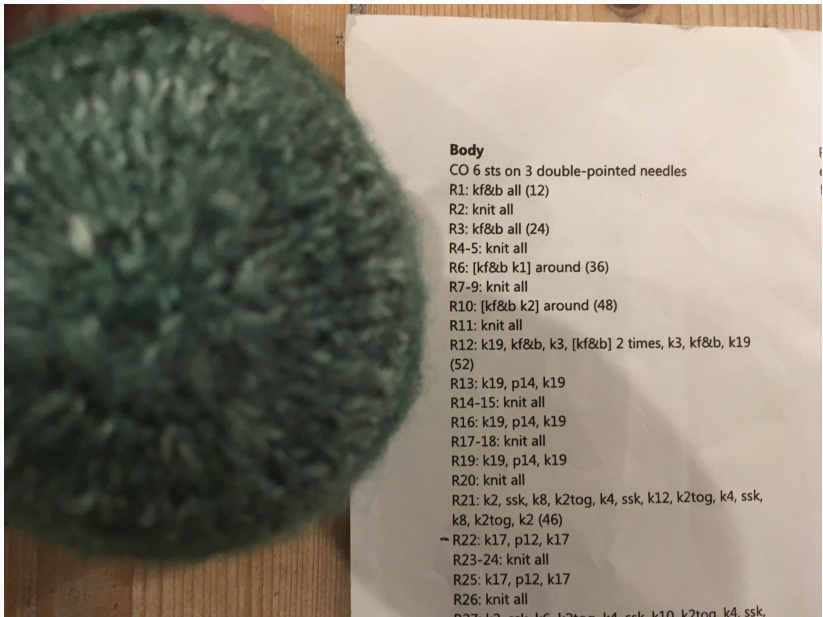
# What is Python?

---

# What is Python?

---

Python: A language, not a program



### Body

CO 6 sts on 3 double-pointed needles

R1: kf&b all (12)

R2: knit all

R3: kf&b all (24)

R4-5: knit all

R6: [kf&b k1] around (36)

R7-9: knit all

R10: [kf&b k2] around (48)

R11: knit all

R12: k19, kf&b, k3, [kf&b] 2 times, k3, kf&b, k19  
(52)

R13: k19, p14, k19

R14-15: knit all

R16: k19, p14, k19

R17-18: knit all

R19: k19, p14, k19

R20: knit all

R21: k2, ssk, k8, k2tog, k4, ssk, k12, k2tog, k4, ssk,  
k8, k2tog, k2 (46)

~ R22: k17, p12, k17

R23-24: knit all

R25: k17, p12, k17

R26: knit all

R27: k2, ssk, k8, k2tog, k4, ssk, k10, k2tog, k4, ssk,

An algorithm in a language that's a bit harder (I think) than Python

# Python

## What?

- A language, not a specific program
- Huge advantage: flexibility, portability
- One of *the* languages for CCS. (The other one is R.)

## Which version?

Version 3.11 has been released in October 2022. Everything of 3.8 or higher should be OK.

# Python

## What?

- A language, not a specific program
- Huge advantage: flexibility, portability
- One of *the* languages for CCS. (The other one is R.)

## Which version?

Version 3.11 has been released in October 2022. Everything of 3.8 or higher should be OK.

# What is Python?

---

Python versus R

## Should I use Python or R?

- For some people, an almost religious debate (but that's stupid!)
- Different history: R comes from math and statistics, Python from computer science
- Different user communities: statisticians: R, linguists: Python; most of industry: Python
- Nowadays (unlike some years ago), most things that are relevant to us can be done in *both* languages.

Specialize in one language, but have a basic understanding of both!

## Should I use Python or R?

- For some people, an almost religious debate (but that's stupid!)
- Different history: R comes from math and statistics, Python from computer science
- Different user communities: statisticians: R, linguists: Python; most of industry: Python
- Nowadays (unlike some years ago), most things that are relevant to us can be done in *both* languages.

**Specialize in one language, but have a basic understanding of both!**



# Should I use Python or R?

## For those of you who know R

You will notice a few things:

- R “abstracts away” more stuff; in Python, you need to be a bit more explicit
- In R, (almost) everything is a dataframe or a vector, in Python not
- There is a big difference between applying a function to a single value or a whole “column” in Python
- In Python, we start counting with 0 (not 1)

# Should I use Python or R?

## For those of you who know R

You will notice a few things:

- R “abstracts away” more stuff; in Python, you need to be a bit more explicit
- In R, (almost) everything is a dataframe or a vector, in Python not
- There is a big difference between applying a function to a single value or a whole “column” in Python
- In Python, we start counting with 0 (not 1)

# Should I use Python or R?

## For those of you who know R

You will notice a few things:

- R “abstracts away” more stuff; in Python, you need to be a bit more explicit
- In R, (almost) everything is a dataframe or a vector, in Python not
- There is a big difference between applying a function to a single value or a whole “column” in Python
- In Python, we start counting with 0 (not 1)

# Should I use Python or R?

## For those of you who know R

You will notice a few things:

- R “abstracts away” more stuff; in Python, you need to be a bit more explicit
- In R, (almost) everything is a dataframe or a vector, in Python not
- There is a big difference between applying a function to a single value or a whole “column” in Python
- In Python, we start counting with 0 (not 1)

# Should I use Python or R?

## For those of you who know R

You will notice a few things:

- Typical “user level” R-code often is essentially written as consecutive commands: line for line. Structuring the code using typical programming constructs (writing own functions, using control structures) is not considered “pro” use (like in R), but *the* way to work in Python.
- Python likes object orientation, R likes functions and pipelines

# Should I use Python or R?

## For those of you who know R

You will notice a few things:

- Typical “user level” R-code often is essentially written as consecutive commands: line for line. Structuring the code using typical programming constructs (writing own functions, using control structures) is not considered “pro” use (like in R), but *the* way to work in Python.
- Python likes object orientation, R likes functions and pipelines

## Why do we use Python in this course?

- Python is a general programming language, R is a domain-specific one (it's meant for stats).
- The course is on ACA, and the closely related computational linguistics community uses Python.
- Python is *much* better equipped for machine learning (and R for traditional stats, but that's not our focus).
- Web scraping is possible in R, but typically done in Python.
- If you know Python, you can probably learn many other languages quite quickly (including R). That's a bit more difficult coming from R.

# You can build *anything*, including a live news aggregator!

Beyond the scope of this course, but this was a Master thesis (Loecherbach & Trilling, 2020):

## 3bij3 – Developing a framework for researching recommender systems and their effects

Felicia Loecherbach & Damian Trilling

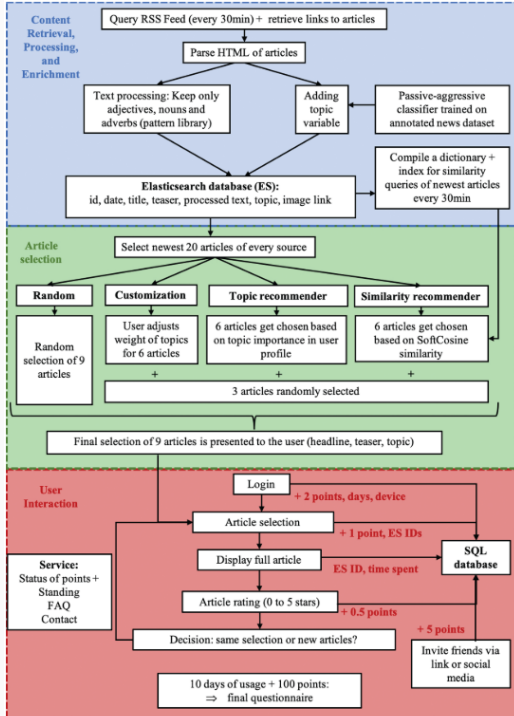
CCR 2 (1): 53–79

DOI: 10.5117/CCR2020.1.003.LOEC

### Abstract

Today's online news environment is increasingly characterized by personalized news selections, relying on algorithmic solutions for extracting relevant articles and composing an individual's news diet. Yet, the impact of such recommendation algorithms on how we consume and perceive news is still understudied. We therefore developed one of the first software solutions to conduct studies on effects of news recommender systems in a realistic setting. The web app of our framework (called 3bij3) displays real-time news articles selected by different mechanisms. 3bij3 can be used to conduct large-scale field experiments, in which participants' use of the site can be tracked





# What is Python?

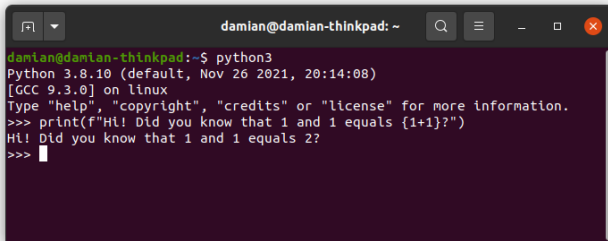
---

What do we need to write Python programs?

# 1. A Python interpreter

A Python interpreter – a program that “understands” Python and translates it into low-level commands that your computer executes.

Type `python3` in your operating system’s command line:

A terminal window titled 'damian@damian-thinkpad: ~' with standard window controls. The terminal output shows the command 'python3' being executed, which starts the Python 3.8.10 interpreter. It displays the version, default path, and date. It then prompts the user to type 'help', 'copyright', 'credits', or 'license'. The user enters a print statement: '>>> print(f"Hi! Did you know that 1 and 1 equals {1+1}?")'. The interpreter outputs 'Hi! Did you know that 1 and 1 equals 2?' and returns to the prompt '>>>' with a cursor.

```
damian@damian-thinkpad:~$ python3
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print(f"Hi! Did you know that 1 and 1 equals {1+1}?")
Hi! Did you know that 1 and 1 equals 2?
>>> 
```

That’s also called a Read-Evaluate-Print-Loop (REPL)

## 2. Some more comfort

- Starting an interpreter like this useful for quick try-outs, but not really a way to write full programs
- You can write them in a (good) text editor (Atom, Sublime, Notepad++, geany, emacs, vi) – that's how it has historically been done and still a good skill to have
- But there are also *Integrated Development Environments* (IDEs) that integrate an editor and an interpreter and offer a lot of extra functionality
- We will mainly work with Jupyter (which is tailored to data analyst's needs), but you can consider checking out PyCharm or Visual Studio Code (both used by professional programmers) or Spyder (for those who like RStudio)

## 2. Some more comfort

- Starting an interpreter like this useful for quick try-outs, but not really a way to write full programs
- You can write them in a (good) text editor (Atom, Sublime, Notepad++, geany, emacs, vi) – that's how it has historically been done and still a good skill to have
- But there are also *Integrated Development Environments* (IDEs) that integrate an editor and an interpreter and offer a lot of extra functionality
- We will mainly work with Jupyter (which is tailored to data analyst's needs), but you you can consider checking out PyCharm or Visual Studio Code (both used by professional programmers) or Spyder (for those who like RStudio)

## 2. Some more comfort

- Starting an interpreter like this useful for quick try-outs, but not really a way to write full programs
- You can write them in a (good) text editor (Atom, Sublime, Notepad++, geany, emacs, vi) – that's how it has historically been done and still a good skill to have
- But there are also *Integrated Development Environments* (IDEs) that integrate an editor and an interpreter and offer a lot of extra functionality
- We will mainly work with Jupyter (which is tailored to data analyst's needs), but you you can consider checking out PyCharm or Visual Studio Code (both used by professional programmers) or Spyder (for those who like RStudio)

## 2. Some more comfort

- Starting an interpreter like this useful for quick try-outs, but not really a way to write full programs
- You can write them in a (good) text editor (Atom, Sublime, Notepad++, geany, emacs, vi) – that's how it has historically been done and still a good skill to have
- But there are also *Integrated Development Environments* (IDEs) that integrate an editor and an interpreter and offer a lot of extra functionality
- We will mainly work with Jupyter (which is tailored to data analyst's needs), but you you can consider checking out PyCharm or Visual Studio Code (both used by professional programmers) or Spyder (for those who like RStudio)

### 3. Additional modules

- A great advantage of Python are the many packages to extend it.
- You can install them using a command called `pip`
- Alternatively, the Anaconda distribution already comes with most (but probably not all) you'll need pre-installed.





*You see why I talked about the difference between monolithic software like SPSS and the “a language, not a program”-notion here?*

# Expectation Management

---

Let's talk a bit about what this course can  
and cannot offer, and what my and your  
expectations are.

## Expectation 1: The course materials can't offer everything.

There is a lot of great stuff out there: YouTube tutorials, blogs, ... And I don't believe that the way I explain, say, Web Scraping is the one and only way to do it, for each and everyone.

In the end, it is about you learning something – not about following exactly what I say. If you find a source that you think explains something better than I do: Use it! Share it!

## Expectation 2: Use Stackoverflow. But acknowledge your sources.

Learn from other people's code and don't reinvent the wheel. But indicate in your assignments clearly where you got some code from! (Otherwise, I'll report you for plagiarism. And I don't want to do that.)

I don't have to tell you that you can prompt ChatGPT to "write a for loop in Python". But next to this being fraud if you hand it in, it – even more importantly – won't teach you anything. Also, in case of suspicious code (e.g., code using very different idioms and approaches than we discussed), I'll reserve the right to question you to explain what's happening.

## Expectation 3: You have to practice.

You have to practice, preferably with a classmate. Try to modify the code we used, or write something (anything) on your own. Use the Friday sessions to ask for feedback on that.

## Expectation 4: The first weeks are crucial.

The first weeks are crucial, really. REALLY. REALLY!!!

Maybe quite counterintuitively, the fancy stuff we are doing later (Machine Learning. . . ) is relatively easy to do. But only if you know your basics

Want to take it easy some weeks? Better do so later in the course ;-)



## Expectation 5: You are an individual.

- There is wide variation of how quickly people “get it” in the first weeks. You are not stupid if it takes a bit longer!
- In the end, you will need to deliver an individual research project on a topic of your own choice.
- You should consult me on the exact scope, but it needs to include multiple techniques from both parts of the course.
- Some students say they prefer more standardized testing, but that would be *in conflict with the main learning goal of this course*: Being able to conduct a CSS research project, from beginning to end.
- But most students actually find this one of the best things of this course, and so do I: It’s great to read all your creative work!

# Structure of the meetings (usually)

**Wednesday** Lecture

**Friday** Lab session.

See Course Manual for specific preparation and readings.

In general, we will work on some programming exercise *during* the lab sessions, so that you can immediately ask questions when you get stuck.

No weekly assignments to hand in, but implicit continuous assignment: Finish the exercises we do in class, and play with them/extend them/write your own code!





*Any questions?*

# Expectation Management

---

Next steps

Make sure you can run Python code and install packages. Otherwise, you won't be able to follow along on Friday. (See instructions you got.)

# References

---



Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). *On the dangers of stochastic parrots: Can language models be too big?* (Vol. 1). Association for Computing Machinery. <https://doi.org/10.1145/3442188.3445922>



Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V., & Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems*, 29, 4349–4357.



Boyd, D., & Crawford, K. (2012). Critical questions for Big Data. *Information, Communication & Society*, 15(5), 662–679. <https://doi.org/10.1080/1369118X.2012.678878>



Gartner. (n.d.). Big data. <https://www.gartner.com/en/information-technology/glossary/big-data>



Hilbert, M., Barnett, G., Blumenstock, J., Contractor, N., Diesner, J., Frey, S., González-Bailón, S., Lamberso, P., Pan, J., Peng, T.-Q., Shen, C., Smaldino, P. E., Van Atteveldt, W., Waldherr, A., Zhang, J., & Zhu, J. J. H. (2019). Computational Communication Science : A Methodological Catalyzer for a Maturing Discipline. *International Journal of Communication*, 13, 3912–3934.



Kitchin, R. (2014). Big Data, new epistemologies and paradigm shifts. *Big Data & Society*, 1(1), 1–12. <https://doi.org/10.1177/2053951714528481>



Lazer, D., Pentland, A., Adamic, L., Aral, S., Barabási, A.-L., Brewer, D., Christakis, N., Contractor, N., Fowler, J., Gutmann, M., Jebara, T., King, G., Macy, M., Roy, D., & van Alstyne, M. (2009). Computational social science. *Science*, 323, 721–723.  
<https://doi.org/10.1126/science.1167742>



Loecherbach, F., & Trilling, D. (2020). 3bij3 - Developing a framework for researching recommender systems and their effects. *Computational Communication Research*, 2(1), 53–79.  
<https://doi.org/10.5117/CCR2020.1.003.LOEC>



Mahrt, M., & Scharkow, M. (2013). The value of Big Data in digital media research. *Journal of Broadcasting & Electronic Media*, 57(1), 20–33.  
<https://doi.org/10.1080/08838151.2012.761700>



van Atteveldt, W., Strycharz, J., Trilling, D., & Welbers, K. (2019). Toward Open Computational Communication Science : A Practical Road Map for Reusable Data and Code University of Amsterdam , the Netherlands. *International Journal of Communication*, 13, 3935–3954.



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. <https://doi.org/10.1080/19312458.2018.1458084>



Vis, F. (2013). A critical reflection on Big Data: Considering APIs, researchers and tools as data makers. *First Monday*, 18(10), 1–16. <https://doi.org/10.5210/fm.v18i10.4878>