12-7-2022

Programming for Qualitative Data Analysis: Towards a YAML Workflow

Blair Wang

The University of Sydney, blair.wang@sydney.edu.au

Programming for Qualitative Data Analysis: Towards a YAML Workflow

Research-in-progress

Blair Wang
The University of Sydney
Sydney, Australia
Email: blair.wang@sydney.edu.au

Abstract

Information Systems (IS) researchers undertaking qualitative research are increasingly proficient with programming techniques. They are also increasingly endeavouring towards openness and transparency, in light of the Open Science movement; and they are increasingly receptive towards alternative and emerging qualitative techniques. In light of these considerations, this paper introduces a "YAML Workflow for Qualitative Data Analysis". In this workflow, qualitative data analysis is seen as a form of data modelling, thus leveraging techniques from the domain of data modelling such as boundary objects like class diagrams and tools such as integrated development environments. Further, this workflow entails the use of programming languages like Python, by which data can be manipulated, queried, and summarised (e.g., in table-like overviews). Importantly, this workflow is entirely driven by plain-text files that can be tracked with a version control system like Git. Overall, this workflow supports the innovative directions towards which qualitative IS research is evolving.

Keywords: qualitative research, hermeneutics, programming, YAML, open science

1 Background

1.1 The Changing Face of Qualitative Data Analysis and Tools

The tools-of-the-trade for qualitative data analysis have changed greatly over the past few decades. Pre-digital qualitative research relied on the “pen, paper, and scissors approach to handling complex data” (Séror 2005, p. 322). However, by the mid-1990s, researchers already started using word processing software as easily-accessible digital alternatives to pen/paper/scissors (Miles and Weitzman 1994). Given the limitations of word processing software, those researchers quickly sought after specialised software packages that could do for qualitative data analysis what the Statistical Package for the Social Sciences (SPSS) did for quantitative data analysis (Séror 2005). Thus emerged qualitative data analysis software packages like NVivo, ATLAS.ti and MAXQDA (Miles and Weitzman 1994; Séror 2005). Such software packages needed to be *“user friendly”*, as Miles and Weitzman (1994) put it; digital literacy was, at the time, limited: *“we do not deal here with the ‘hacker’, ... who lives and breathes computing”* (Miles and Weitzman 1994, p. 313).

In line with the ACIS 2022 conference theme, “the changing face of IS”, it is important to observe that there are at least three important differences between the turn of the millennium and the present day. Firstly, it is now quite common for researchers to be ‘living and breathing computing’ without being ‘hackers’. Basic proficiency in programming for analytics (e.g., using Python or R) is now part of researcher training and continuing professional development; for example, through the *Software Carpentry* initiative (Wilson 2016). In the case of the IS discipline, it is not unusual for an IS researcher to be working as a career academic who happens to teach programming for analytics as part of their university’s IS curriculum (Toney et al. 2021). Secondly, the emergence of the Open Science movement (Mendez et al. 2020) has prompted researchers — in IS and in other disciplines — to prefer programming-based analysis (e.g., Python and R) over click-and-point programs (e.g., SPSS and NVivo), since programming-based analysis is more conducive to *“a reproducible workflow which can be easily version controlled”* (Mendez et al. 2020, p. 491). Thirdly, the practices of qualitative research have evolved (and continue to evolve). For example, the rigid one-to-many hierarchical data structure of the ‘Gioia methodology’ (Mees-Buss et al. 2022), introduced by Gioia et al. (2013) as a systematic and rigourous approach to qualitative data analysis, happens to neatly match the rigid one-to-many hierarchical data structure of the NVivo software package. However, recent discourse has called for more hermeneutic orientation to qualitative data analysis, both for literature reviews (Boell and Cecez-Kecmanovic 2014) and for empirical studies (Mees-Buss et al. 2022). Such a hermeneutic orientation entails free-form analysis techniques, which depart from the rigid one-to-many hierarchical data structure of the Gioia methodology (Mees-Buss et al. 2022) and thus arguably also depart from the rigid one-to-many hierarchical data structure prescribed by NVivo (although it is possible for NVivo and similar software packages to facilitate a ‘flat’ array of codes without any hierarchical structure at all, such total absence of structure also makes it difficult to meaningfully model the interrelationships between codes). Additionally, qualitative research — particularly in IS — increasingly involves computationally-intensive techniques and large datasets like digital trace data (Miranda et al. 2022), which are beyond the scope of more traditional qualitative data analysis techniques.

Reflecting these three differences, this paper proceeds with the following research aim: *To outline a workflow for qualitative data analysis that reflects: 1) the programming skills often possessed by present-day IS researchers; 2) the interest in openness and transparency, in light of the Open Science movement; and 3) alternative and emerging qualitative techniques*. To address this research aim, this paper introduces a *YAML Workflow for Qualitative Data Analysis*.

1.2 The YAML Data Exchange Format

Introduced in 2004, YAML is similar to established data exchange formats like Extensible Markup Language (XML) and JavaScript Object Notation (JSON), but follows a greatly simplified syntax, reflected in “YAML” being a recursive acronym for *YAML Ain’t Markup Language* (Ben-Kiki et al. 2021).

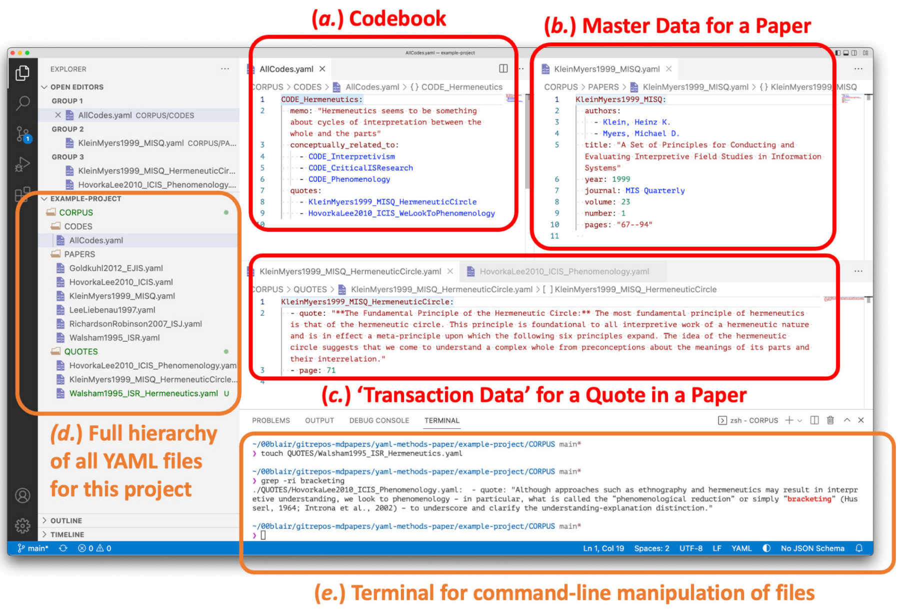
Specifically, YAML is a format for *data serialisation* (Ben-Kiki et al. 2021), the process of storing computer data structures for later use. However, unlike formats that serialise data structures into raw binary data streams (e.g., MP3 audio and JPEG pictures; or indeed, an NVivo Project file), YAML serialises data structures into plain-text, as shown in Figure 1 below. The use of plain-text rather than raw binary supports interoperability, and also enables YAML files to be tracked by version control systems like Git — both conducive to open science, to be discussed later in this paper. As shown in Figure 1, YAML syntax is basically nested key-value pairs in the format *“key: value”* (where *“value”* could be another set of key-value pairs). This simplified syntax is by design, to make YAML relatively easier for humans to work with (Ben-Kiki et al. 2021) compared to alternatives like XML and JSON (in fact, YAML is semantically equivalent to XML and JSON, and can be easily converted between these formats using, for example, *PyYAML* and *pandas* in Python as described below). YAML has therefore been increasingly adopted for the exchange of scientific and scholarly data; for example, in biology (Vanhoefer et al. 2021) and chemistry (Weber and Niemeyer 2018). YAML is also, by virtue of this simplified syntax, an open standard that does not lock in any particular vendor. The processing of YAML data is therefore possible in various open-source software packages and in various programming languages (Ben-Kiki et al. 2021), most notably in the Python programming language. The Python programming language is able to parse YAML data using the open source *PyYAML* library which, when combined with various other open source data science libraries such as *pandas*, *SQLAlchemy* and *scikit*, supports workflows in which YAML data can be transformed to and from various other formats such as flat-file data tables and relational database systems, between multiple systems (Bonaquist et al. 2021).

2 YAML Workflow for Qualitative Data Analysis

This section provides an overview of a proposed *YAML Workflow for Qualitative Data Analysis*. This workflow consists of three activities: *Data Modelling*, *Interpreting-Visualising-Iterating*, *Collaboration and Version Control*. These activities proceed in a cyclical (rather than linear) fashion, similar to the generative iteration of agile project management, and of the hermeneutic circle (Boell and Cecez-Kecmanovic 2014). In other words, the workflow can be expressed as: *Data Modelling* ⟵⟶ *Interpreting-Visualising-Iterating* ⟵⟶ *Collaboration and Version Control*.

2.1 Data Modelling

Qualitative data analysis can be understood as a specialised and interpretive form of *data modelling*, mapping the world as, e.g., entities and relationships. For example, the NVivo data model consists of a set of *Nodes* which relate to each other in one-to-many relationships (i.e., a parent node can have multiple child nodes, but each child node can have only one parent node). However, the problem with such a rigid, pre-defined data model is that it cannot *a priori* account for the diversity of research topics, domains, phenomena, and interpretations thereof in which that data model is later deployed. In contrast, YAML enables a highly-flexible form of data modelling, i.e., free-form analysis techniques, which depart from the rigid one-to-many hierarchical data structure. The researcher can simply start writing YAML files, initially as an intuitive form of interpretation or 'brainstorming' about the data that the researcher is working with. One such scenario is illustrated in Figure 1 below.


(a.) Codebook
(b.) Master Data for a Paper
(c.) 'Transaction Data' for a Quote in a Paper
(d.) Full hierarchy of all YAML files for this project
(e.) Terminal for command-line manipulation of files
AllCodes.yaml — example-project
EXPLORER
OPEN EDITORS
GROUP 1
AllCodes.yaml CORPUS/CODES
GROUP 2
KleinMyers1999_MISQ.yaml CORPUS/PA...
GROUP 3
KleinMyers1999_MISQ_HermeneuticCir...
HovorkaLee2010_ICIS_Phenomenology....
EXAMPLE-PROJECT
CORPUS
CODES
AllCodes.yaml
PAPERS
Goldkuhl2012_EJIS.yaml
HovorkaLee2010_ICIS.yaml
KleinMyers1999_MISQ.yaml
LeeLiebenau1997.yaml
RichardsonRobinson2007_ISJ.yaml
Walsham1995_ISR.yaml
QUOTES
HovorkaLee2010_ICIS_Phenomenology.yam
KleinMyers1999_MISQ_HermeneuticCircle...
Walsham1995_ISR_Hermeneutics.yaml U
OUTLINE
TIMELINE
main* ⊗ 0 ⚠ 0
AllCodes.yaml
CORPUS > CODES > AllCodes.yaml > {} CODE_Hermeneutics
KleinMyers1999_MISQ.yaml
CORPUS > PAPERS > KleinMyers1999_MISQ.yaml > {} KleinMyers1999_MISQ
KleinMyers1999_MISQ_HermeneuticCircle.yaml
HovorkaLee2010_ICIS_Phenomenology.yaml
CORPUS > QUOTES > KleinMyers1999_MISQ_HermeneuticCircle.yaml > [ ] KleinMyers1999_MISQ_HermeneuticCircle
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
zsh - CORPUS
Ln 1, Col 19
Spaces: 2
UTF-8
LF
YAML
No JSON Schema

```yaml
CODE_Hermeneutics:
  memo: "Hermeneutics seems to be something
  about cycles of interpretation between the
  whole and the parts"
  conceptually_related_to:
    - CODE_Interpretivism
    - CODE_CriticalISResearch
    - CODE_Phenomenology
  quotes:
    - KleinMyers1999_MISQ_HermeneuticCircle
    - HovorkaLee2010_ICIS_WeLookToPhenomenology
```

```yaml
KleinMyers1999_MISQ:
  authors:
    - Klein, Heinz K.
    - Myers, Michael D.
  title: "A Set of Principles for Conducting and
  Evaluating Interpretive Field Studies in Information
  Systems"
  year: 1999
  journal: MIS Quarterly
  volume: 23
  number: 1
  pages: "67--94"
```

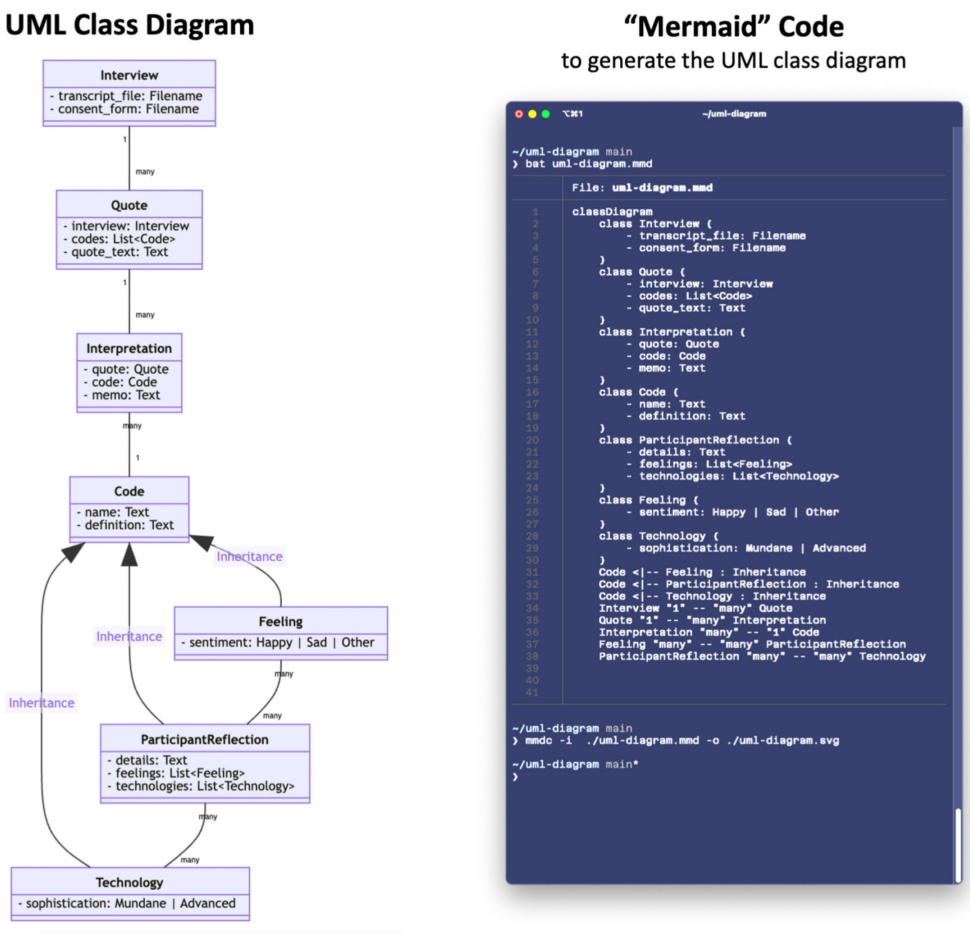
```yaml
KleinMyers1999_MISQ_HermeneuticCircle:
  - quote: "**The Fundamental Principle of the Hermeneutic Circle:** The most fundamental principle of hermeneutics
  is that of the hermeneutic circle. This principle is foundational to all interpretive work of a hermeneutic nature
  and is in effect a meta-principle upon which the following six principles expand. The idea of the hermeneutic
  circle suggests that we come to understand a complex whole from preconceptions about the meanings of its parts and
  their interrelation."
  - page: 71
```

```
~/00blair/gitrepos-mdpapers/yaml-methods-paper/example-project/CORPUS main*
❯ touch QUOTES/Walsham1995_ISR_Hermeneutics.yaml

~/00blair/gitrepos-mdpapers/yaml-methods-paper/example-project/CORPUS main*
❯ grep -ri bracketing
./QUOTES/HovorkaLee2010_ICIS_Phenomenology.yaml:  - quote: "Although approaches such as ethnography and hermeneutics may result in interpr
etive understanding, we look to phenomenology – in particular, what is called the "phenomenological reduction" or simply "bracketing" (Hus
serl, 1964; Introna et al., 2002) – to underscore and clarify the understanding-explanation distinction."

~/00blair/gitrepos-mdpapers/yaml-methods-paper/example-project/CORPUS main*
❯ ▯
```

Figure 1: Data modelling using YAML files, in Microsoft Visual Studio Code (VSC).

In Figure 1, the researcher has constructed a collection of YAML files stored in three folders — *CODES*, *PAPERS* and *QUOTES* — each representing an entity of relevance (in this scenario, for a literature review project). These YAML files can be accessed by any system or software, but — as is the case with many programming projects — an integrated development environment (IDE), like the example Microsoft Visual Studio Code (VSC) shown in Figure 1, is particularly well-suited to the task. In this instance, the researcher has taken advantage of VSC's ability to open multiple files in a 'panels' workspace layout in order to simultaneously work with both the codebook (part (*a.*), top-left panel), the metaphorical 'master data' for a particular paper (part (*b.*), top-right panel), and the metaphorical 'transaction data' for two instances where the researcher has identified various quotes from various papers (part (*c.*), middle panel) to be included in the codebook. At the same time, the researcher can see the full hierarchy of all YAML files for this project on the computer filesystem, in the navigation tree on the left-hand side (part (*d.*), leftmost panel). The researcher has also taken advantage of VSC's built-in Terminal (part (*e.*), bottom panel) to manipulate the files through command-line shortcuts without needing to lift one's hands away from the keyboard, picking up the mouse, or disturbing the visual layout of the workspace.

Whilst the scenario in Figure 1 involves the human researcher manually writing all the YAML files, it is entirely possible for the YAML files to be machine-generated. Such integration of machine labour and human labour in the research process would be particularly conducive in solving the problem of trying to perform qualitative data analysis at scale, with 'big' datasets such as in the case of digital trace data (Miranda et al. 2022). Additionally, whilst the data model implied in the scenario in Figure 1 does not go beyond what would be possible in software like NVivo, more sophisticated data models can be built up by introducing more entities as depicted in the example in Figure 2.



```
~/uml-diagram main
❯ bat uml-diagram.mmd
       File: uml-diagram.mmd
       classDiagram
           class Interview {
               - transcript_file: Filename
               - consent_form: Filename
           }
           class Quote {
               - interview: Interview
               - codes: List<Code>
               - quote_text: Text
           }
           class Interpretation {
               - quote: Quote
               - code: Code
               - memo: Text
           }
           class Code {
               - name: Text
               - definition: Text
           }
           class ParticipantReflection {
               - details: Text
               - feelings: List<Feeling>
               - technologies: List<Technology>
           }
           class Feeling {
               - sentiment: Happy | Sad | Other
           }
           class Technology {
               - sophistication: Mundane | Advanced
           }
           Code <|-- Feeling : Inheritance
           Code <|-- ParticipantReflection : Inheritance
           Code <|-- Technology : Inheritance
           Interview "1" -- "many" Quote
           Quote "1" -- "many" Interpretation
           Interpretation "many" -- "1" Code
           Feeling "many" -- "many" ParticipantReflection
           ParticipantReflection "many" -- "many" Technology

~/uml-diagram main
❯ mmdc -i  ./uml-diagram.mmd -o ./uml-diagram.svg

~/uml-diagram main*
❯
```

Figure 2: Data model represented as a UML class diagram, drawn using "Mermaid".

The example depicted in Figure 2 has been constructed based on a scenario involving interviews with participants in an empirical study. This example demonstrates advantages both for the process of interpretation and for the openness and transparency of scholarship.

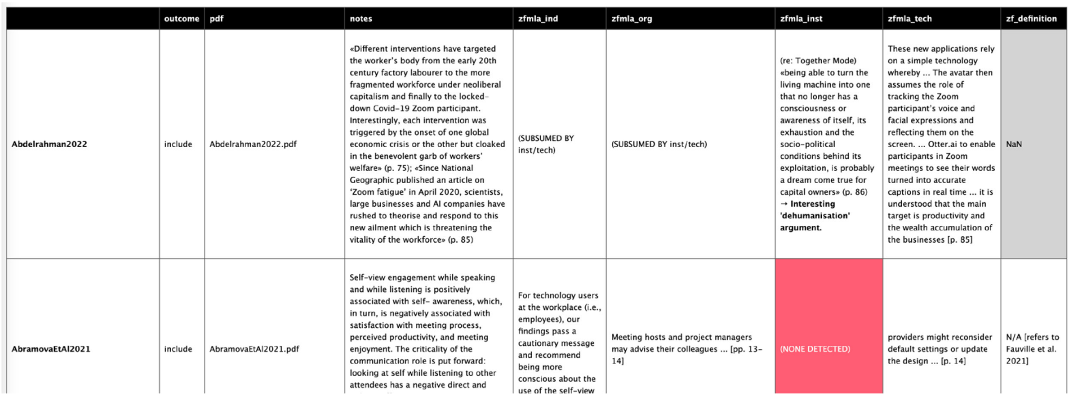
Firstly, the example depicted in Figure 2 allows more flexibility than a one-to-many hierarchical data structure. The ontology of the codes is based on three distinct types relevant to this empirical study — *Feelings*, *Technologies*, and *Participant Reflections* — that allow for flexible interrelating. A quote can be simply coded to a particular *Feeling* or *Technology*, but a quote can also be coded to a *Participant Reflection* that is then related to multiple *Feelings* and multiple *Technologies*. This usage of free-form analysis techniques (which depart from the rigid one-to-many hierarchical data structure) allows for a more hermeneutic orientation to qualitative research, which emphasises abductive reasoning and ongoing development of provisional theories rather than linear inductive reasoning from data to theory through clean linkage from first- to second- order concepts (Mees-Buss et al. 2022).

Secondly, the example depicted in Figure 2 allows more data to be captured than is typically prescribed by existing qualitative data analysis software like NVivo. Notably, the act of coding — linking a Quote to a Code — is given its own composite entity, *Interpretation*, encouraging the researcher to leave a memo for the research to provide more details about why a particular quote matches with a particular code. Additional attributes could be attached to this *Interpretation* entity, for example, to encourage the researcher to document answers to reflexive questions such as “Have the risks of misinterpretation or of being misled by participants’ accounts been considered?”, “Have I examined my own ‘perceptual screen’?”, and “Are the participants’ social, cultural, and historical contexts critically understood?” (Mees-Buss et al. 2022, p. 421).

Finally, the example depicted in Figure 2 demonstrates the compatibility of a YAML-based approach with other standards in data modelling, such as Unified Modelling Language (UML). Although Figure 2 is depicting the semantic flexibility possible in YAML files, the figure itself is in fact rendered as a UML class diagram: it is metadata (UML) depicting the semantic flexibility of the underlying data structure (YAML). This UML class diagram is generated deterministically from the 38 lines of *Mermaid* diagramming syntax, shown on the right-hand side of Figure 2. This code can be stored alongside the YAML files in a version control system like Git, supporting the openness and transparency of the analysis (to be discussed later in this paper).

2.2 Interpreting-Visualising-Iterating

As outlined above, the work involved in data modelling already entails a substantial share of the interpretation involved in qualitative data analysis. However, there are additional aspects of this paper’s YAML Workflow for Qualitative Data Analysis that further support interpretation efforts, particularly in an iterative fashion. Since YAML data can be easily accessed in the Python programming language using *PyYAML*, it can be organised, processed and manipulated based on the researcher’s specific needs. Notably, YAML data ingested into a Python script can be translated into *pandas* data frames (essentially, table structures with rows and columns). This allows the researcher to quickly produce a report on a particular view of their data and interpretations, for example, listing the quotes associated with each of the codes, or depicting a grid-like view of how codes are interrelated similar to NVivo’s “framework matrices” functionality. An example of such a data frame is depicted in Figure 3 below.



	outcome	pdf	notes	zfmla_ind	zfmla_org	zfmla_inst	zfmla_tech	zf_definition
Abdelrahman2022	include	Abdelrahman2022.pdf	«Different interventions have targeted the worker’s body from the early 20th century factory labourer to the more fragmented workforce under neoliberal capitalism and finally to the locked-down Covid-19 Zoom participant. Interestingly, each intervention was triggered by the onset of one global economic crisis or the other but cloaked in the benevolent garb of workers’ welfare» (p. 75); «Since National Geographic published an article on ‘Zoom fatigue’ in April 2020, scientists, large businesses and AI companies have rushed to theorise and respond to this new ailment which is threatening the vitality of the workforce» (p. 85)	(SUBSUMED BY inst/tech)	(SUBSUMED BY inst/tech)	(re: Together Mode) «being able to turn the living machine into one that no longer has a consciousness or awareness of itself, its exhaustion and the socio-political conditions behind its exploitation, is probably a dream come true for capital owners» (p. 86) → **Interesting 'dehumanisation' argument.**	These new applications rely on a simple technology whereby ... The avatar then assumes the role of tracking the Zoom participant’s voice and facial expressions and reflecting them on the screen. ... Otter.ai to enable participants in Zoom meetings to see their words turned into accurate captions in real time ... it is understood that the main target is productivity and the wealth accumulation of the businesses [p. 85]	NaN
AbramovaEtAl2021	include	AbramovaEtAl2021.pdf	Self-view engagement while speaking and while listening is positively associated with self- awareness, which, in turn, is negatively associated with satisfaction with meeting process, perceived productivity, and meeting enjoyment. The criticality of the communication role is put forward: looking at self while listening to other attendees has a negative direct and	For technology users at the workplace (i.e., employees), our findings pass a cautionary message and recommend being more conscious about the use of the self-view	Meeting hosts and project managers may advise their colleagues ... [pp. 13-14]	(NONE DETECTED)	providers might reconsider default settings or update the design ... [p. 14]	N/A [refers to Fauville et al. 2021]

Figure 3: Example of table view with conditional formatting.

Figure 3 depicts a content analysis of research papers — for a literature review project on ‘Zoom’ fatigue (Wang and Prester 2022). The depicted example takes advantage of how *pandas* is able to convert a data frame into a HTML (web page) file. Since HTML files are able to contain embedded JavaScript, the researcher can use JavaScript libraries like *jQuery* to further visualise the data. For example, in Figure 3, a jQuery script parses the contents of every cell, and shades the cell red if the cell reads *“(NONE DETECTED)”* (indicating that the researcher has identified a particular feature missing), shades the cell grey if the cell reads *“NaN”* (indicating that the researcher has not yet analysed that particular case), and otherwise leaves the cell unshaded (i.e., white). This simple conditional formatting technique allows the researcher to quickly identify the prevalence of certain characteristics. Unlike the case with Microsoft Excel spreadsheets or the conditional formatting options provided in, e.g., NVivo, this conditional formatting technique allows for highly complex rules to be developed. For example, it would be possible to shade cells based on computational techniques like sentiment analysis of the text stored in each cell. Of course, such computational techniques do not ‘do the thinking’ for the researcher, but essentially triages the portions of the dataset that are more pressing for researcher attention. Additionally, since JavaScript is — like YAML data, Python code and Mermaid diagram syntax — stored as plain-text, it too can be added to a version control system like Git alongside all the other data files and programming code, supporting the openness and transparency of scholarship. This is notably not possible with Microsoft Excel conditional-formatting formulae, nor with the configurations stored inside NVivo project files.

It is further noted that — conveniently — *pandas* is able to parse a set of ingested YAML files with slight variations in the attributes contained in each file. This allows the researcher to introduce a new attribute during the process of interpreting and reinterpreting, but defer the task of coding all existing items in the corpus according to this new attribute (they would simply be marked *"NaN"* and shaded grey, as in Figure 3). The agility of being able to change the data structure 'on the fly' supports a hermeneutic orientation to qualitative research, encouraging the researcher to experiment with new ways of looking at the data rather than feeling 'locked in' to whatever schema existed from when the researcher started their interpretive work.

2.3 Collaboration and Version Control

As alluded throughout this paper, an important advantage of programming-based analysis over click-and-point programs is the ability to store all materials (data files and programming code) in plain-text, which can be tracked using a version control system like Git. Git is regularly used in open science for both collaboration and supplying reproducible evidence material for findings, particularly due to the emergence of online platforms like GitHub and GitLab (Mendez et al. 2020). (Here, "reproducible" does not mean that the intellectual work of qualitative data analysis and interpretation ought to be deterministically reproduced by other researchers. Rather, "reproducible" in this context means that the evidence material — i.e., YAML files and associated metadata — can be reconstructed by other researchers on their computer systems. This reconstruction effort is another important benefit of the use of open standards and open source software, as opposed to licensed proprietary software that may entail inconsistencies between various combinations of release version, product edition, and Mac/Windows variants.)

Prior to paper publication, such a version control system supports collaboration between coauthors and other personnel like research assistants. All files are not only captured to allow for reverting to older versions, but also to visualise the difference between versions (known as the *diff*). Furthermore, through version control functionalities like branches, forks and merges, multiple coauthors can work simultaneously on the same materials and combine their findings at a later stage (e.g., allowing for multiple coders to work in parallel and compare results thereafter). In sum, by placing all data files, programming code (e.g., Python scripts) and other relevant materials (e.g., the UML class diagram code depicted in Figure 2) in the Git repository, researchers empower in-depth shared sensemaking.

After paper publication, the publication of a version-controlled repository (or snapshot/extract thereof) on a platform like GitHub and GitLab supports the credibility and rigour of published qualitative research and findings thereof. The ability to demonstrate faithful representation of data rather than unsubstantiated distortions or manipulations is the critical advantage of the 'Gioia' data structure — to persuade the reader that *"This is what the informants told us. We're not making this stuff up."* (Gioia et al. 2013, p. 23) — that has been hitherto relatively difficult to match with more hermeneutically-oriented qualitative approaches (Mees-Buss et al. 2022). However, if a published paper (adopting a hermeneutic orientation or otherwise) is able to include a link to the research project's Git repository (on GitHub, GitLab, or similar platform), in which the entire chain of evidence and sensemaking is documented — from the YAML files and Python scripts to boundary objects like the UML class diagram depicted in Figure 2 and *pandas* table views in HTML files, and even the log of commits showing how ideas evolved over time — it can achieve even greater support that the researchers are not 'making stuff up'.

Note that sensitive data must be managed appropriately. For example, depending on copyright and human research ethics per jurisdiction, literature reviews could incorporate quotes and bibliographic details but not full-texts (as in Figure 1), and empirical studies could incorporate deidentified quotes and filename references but not the originally-collected data files (as in Figure 2).

3 Future Directions

This paper has outlined a YAML Workflow for Qualitative Data Analysis that — in line with the research aim of this paper — reflects 1) the programming skills increasingly prevalent in the IS community; 2) the Open Science aspirations towards openness and transparency of reproducible evidence chains; and

3) qualitative techniques such as a hermeneutic orientation (Mees-Buss et al. 2022), computationally-intensive techniques and digital trace data (Miranda et al. 2022). However, the research project reported in this paper is in its infancy and work is ongoing.

The immediate next step involves seeking feedback from the IS community, which features both a sizeable cohort of qualitative researchers and a sizeable cohort of 'power users' who practice or even teach programming skills. A strong motivation for presenting this paper at ACIS 2022 is to reach out to the IS researchers who are at the intersection of these cohorts, who may be able to provide informed insights about how to best move forward. It is envisioned that further work thereafter would entail developing the workflow, perhaps entailing empirical studies based on, e.g., participatory design, design science, design thinking, action research, or other similar research methods. The presence of research students (e.g., Honours, Research Masters, PhD), early career academics, and coordinators of university research training classes at ACIS 2022 may be a promising opportunity for serendipitous collaborations to arise. Although the YAML workflow outlined in this paper already draws on many technologies, the beauty of the open ecosystems involved (in contrast to traditional, proprietary closed-source software) is that there are many other programming libraries, application programming interfaces (APIs), techniques, and methods yet to be discovered. Amongst those 'unknown unknowns', there may be opportunities to leverage even more creative, insightful, and perhaps even computationally-intensive approaches, opening up undiscovered horizons in the ever-changing face of qualitative IS research.

4 References

Ben-Kiki, O., Evans, C., and döt Net, I. 2021. *YAML Ain't Markup Language (YAML™) Version 1.2*, YAML Language Development Team. https://yaml.org/spec/1.2.2/

Boell, S. K., and Cecez-Kecmanovic, D. 2014. "A Hermeneutic Approach for Conducting Literature Reviews and Literature Searches," *Communications of the AIS* (34).

Bonaquist, A., Grehan, M., Haines, O., Keogh, J., Mullick, T., Singh, N., Shaaban, S., Radovic, A., and Doryab, A. 2021. "An Automated Machine Learning Pipeline for Monitoring and Forecasting Mobile Health Data," in *Proceedings of the Systems and Information Engineering Design Symposium (SIEDS)*, IEEE.

Gioia, D. A., Corley, K. G., and Hamilton, A. L. 2013. "Seeking Qualitative Rigor in Inductive Research," *Organizational Research Methods* (16:1), pp. 15–31.

Mees-Buss, J., Welch, C., and Piekkari, R. 2022. "From Templates to Heuristics: How and Why to Move Beyond the Gioia Methodology," *Organizational Research Methods* (25:2), pp. 405–429.

Mendez, D., Graziotin, D., Wagner, S., and Seibold, H. 2020. "Open Science in Software Engineering," in *Contemporary Empirical Methods in Software Engineering*, Springer, pp. 477–501.

Miles, M. B., and Weitzman, E. A. 1994. "Choosing Computer Programs for Qualitative Data Analysis," in *Qualitative Data Analysis: An Expanded Sourcebook* (Second Edition.), Sage, pp. 311–317.

Miranda, S., Berente, N., Seidel, S., Safadi, H., and Burton-Jones, A. 2022. "Computationally Intensive Theory Construction: A Primer for Authors and Reviewers," *MIS Quarterly* (46:2), iii–xviii.

Séror, J. 2005. "Computers and Qualitative Data Analysis: Paper, Pens, and Highlighters Vs. Screen, Mouse, and Keyboard," *TESOL Quarterly* (39:2), pp. 321–328.

Toney, S., Light, J., and Urbaczewski, A. 2021. "Fighting Zoom Fatigue: Keeping the Zoombies at Bay," *Communications of the AIS* (48), pp. 40–46.

Vanhoefer, J., Matos, M., Pathirana, D., Schälte, Y., and Hasenauer, J. 2021. "yaml2sbml: Human-Readable and -Writable Specification of ODE Models and Their Conversion to SBML," *Journal of Open Source Software* (6:61).

Wang, B., and Prester, J. 2022. "The Performative and Interpretive Labour of Videoconferencing: Findings from a Literature Review on 'Zoom' Fatigue," in *Proceedings of the International Conference on Information Systems*, Copenhagen (Denmark): forthcoming.

Weber, B. W., and Niemeyer, K. E. 2018. "ChemKED: A Human- and Machine-Readable Data Standard for Chemical Kinetics Experiments," *International Journal of Chemical Kinetics* (50:3).

Wilson, G. 2016. "Software Carpentry: Lessons Learned," *F1000Research* (3), p. 62.

Acknowledgements

The author wishes to sincerely thank Julian Prester, Elisa Wang, and the ACIS 2022 review team that handled this paper. Your constructive feedback has helped to substantially improve this paper.

Copyright