The Gamma: Data Exploration through Iterative Prompting

Leave Authors Anonymous
for Submission
City, Country
e-mail address

Leave Authors Anonymous
for Submission
City, Country
e-mail address

Leave Authors Anonymous
for Submission
City, Country
e-mail address

ABSTRACT

Governments, non-profit organizations and citizen initiatives publish increasing amounts of data, but extracting insights from such data and presenting them to the public is hard. First, data comes in a variety of formats that each requires a different tool. Second, many data exploration tools do not reveal how a result was obtained, making it difficult to reproduce the results and check how they were obtained. We contribute The Gamma, a novel data exploration environment for non-experts. The Gamma is based on a single interaction principle and using it results in transparent and reproducible scripts. This allows transfer of knowledge from one data source to another and learning from previously created data analyses. We evaluate the usability and learnability of The Gamma through a user study on non-technical employees of a research institute. We argue that the our approach allows journalists and the public to benefit from the rise of open data, by making data exploration easier, more transparent and more reproducible.

Author Keywords

Data exploration; End-user programming; Data journalism; Programming languages; Type providers

INTRODUCTION

Data science has more capabilities to help us understand the world than ever before, yet at the same time post-truth politics and increasing public distrust in statistics makes data-driven insights increasingly less relevant in public discourse [5]. This should perhaps not be a surprise. Journalists can access increasing amounts of data, but producing engaging and transparent data-driven reports that are easy to interpret is expensive and requires expert programming skills [7].

The design of a data exploration tool for journalists poses a unique mix of challenges. First, the tool needs to be easy to learn for end-users working under tight deadlines. Second, it needs to support a wide range of data sources in a way where the expertise gained when working with one data source is relevant for other data sources. Third, the resulting data-driven insights need to be transparent, allowing the readers to verify the claims and learn how to reproduce the work.
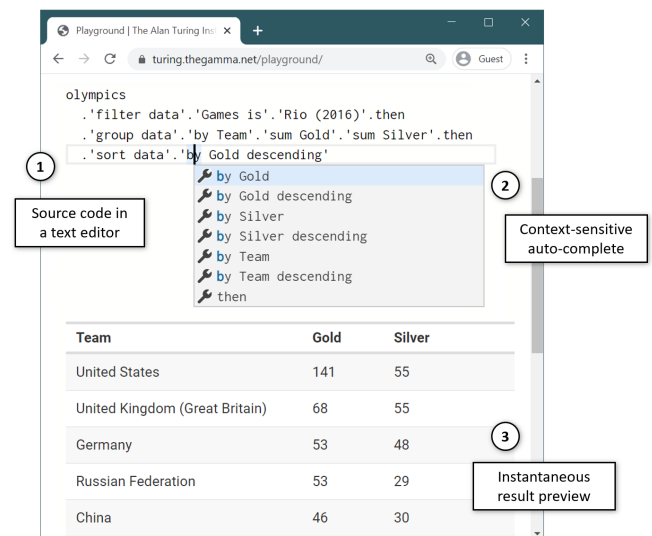
Figure 1. Teams with the greatest number of gold medals from Rio 2016 Olympics with a reproducible The Gamma script (1), an auto-complete prompt offering ways of sorting the data (2) and instant preview (3).

We present The Gamma, a text-based data exploration environment for non-experts. The Gamma is based on a single interaction principle, which provides uniform access to a range of data sources including data tables, graph databases and data cubes. An anlysis created in The Gamma is a transparent script that can be followed to reproduce the result from scratch. This allows learning from existing analyses and encourages readers to engage with the results. Our key contributions are:

- We identify the design requirements for a data exploration tool for journalists (Section 3) and follow those to build a novel programming environment The Gamma (Section 4).

- We introduce *iterative prompting* (Section 5), an interaction design principle that can be used to complete a variety of programming tasks in a uniform way that allows transfer of knowledge between different tasks.

- We show how to use the iterative prompting principle for querying of distinct data sources including data tables, graph databases and data cubes (Section 6).

- We discuss a number of case studies (Section 7) and conduct a user study to evaluate the usability of The Gamma and the extent to which users can, (i) learn from examples and (ii) transfer knowledge between tasks (Section 8).

The Gamma is available as open-source at **thegamma.net**.

RELATED WORK
Visual tools.

Programming tools. Notebooks

Both follow from assistant tools for software development [8]. Our implementation, outlined in the previous section, implements iterative prompting through a code editor mechanism designed for code completion. Finally, the work on type providers [16, 13, 12], which we extend, also utilizes code completion.

Journalism. Idyll [4]
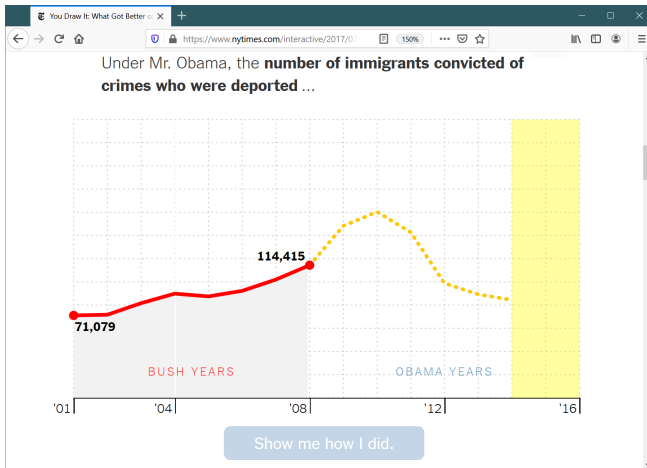
Type providers. PL work

Figure 2. New York Times article on Obama's legacy [9]. The article asks the reader to make a guess (engagement), but only lists "Immigration and Customs Enforcement" as a source of data.
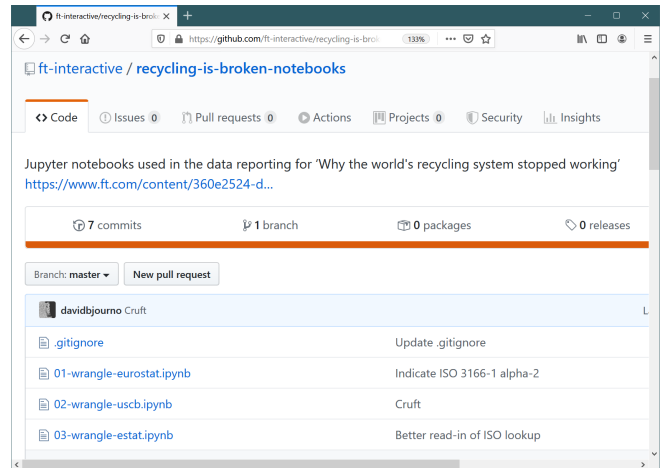

Figure 3. Financial Times analysis of recyclable waste. Full source is provided as Jupyter Notebooks on GitHub [2], but re-running the analysis is difficult, even for an expert.

MOTIVATION

The Gamma aims to adapt the recent innovations in programming language research, especially the work on type providers, into a form where it could be used in practice by journalists and other non-expert interested in data exploration. We start with a careful consideration of our target application domain, i.e. data analyses produced by journalists and citizen data scientists that are published online. We look at both practical requirements for such programming environment and requirements arising from our focus on journalism. This analysis is based on the author's experience of collaborating with journalists[1], review of literature on data journalism, e.g. [7, 14, 1] and more general trends in journalism.

Open Journalism

Journalism continually develops and responds to the many challenges it faces [6]. Two recent challenges are relevant to our work. The first is building trust in media. One way of establishing trust in the age of fake news is to be more transparent about editorial decisions, process and original sources. Many journalists believe that opening up the process shows the quality and trustworthiness of their work [11]. The second challenge is reader engagement. To develop a relationship with readers, journalists are increasingly looking for meaningful ways of engagement. This includes reader comments, involvement of citizen journalists [10, 3] and the development of new interactive formats [9]. To address the above challenges, a tool for data exploration should satisfy the following three requirements.

Trust Through Transparency
To support trustworthiness, data analyses should be transparent. The reader should be able to determine what is the source of analysed data and how has the data been transformed. As much as possible, these capabilities should also be accessible to non-expert readers.

Reproducibility for Fact Checking
It should be possible to re-run the analysis to verify that it produces the presented results. However, running an opaque script is not enough. A reader should be able to recreate the analysis by following the necessary steps from the original data source to the end result.

Encouraging Meaningful Engagement
The tool should support a mechanism through which readers can engage in a meaningful discussion. For example, it should allow modifying of parameters of a data visualization in order to show how different choices affect the final result.

End-user Data Exploration

Our aim is to make programmatic data exploration accessible to journalists, but we want to keep the desirable properties of text-based programming. In particular, source code of a data exploration should provide a full reproducible record of how the data analysis has been done. As end-users, journalists have a number of interesting characteristics. They work under tight deadline and data exploration is only a complementary skill. They also need to work with a wide range of data sources, including big data tables (e.g. Iraq War documents leak) or graph databases (e.g. Panama Papers). This leads to a number of practical requirements on the programming environment.

Conceptual Simplicity
We target end-users who cannot dedicate much time to learning about a tool prior to using it. Consequently, using the tool should require understanding of only a small number of concepts. Once the user understand a small number of concepts, they should be able to complete basic data exploration tasks.

Uniformity across Data Sources
The users should be able to navigate through large databases, query relational databases and query graph databases through the same mechanism. Ideally, expertise gained with one data source should also be transferable to working with another data source.

---

[1]Citations removed to preserve anonymity.

Learning without Experts

Sarkar [15] reports that users learn how to use Excel either by talking to experts, or by seeing a feature in a spreadsheet received from a colleague. In our circumstances, experts are unlikely to be available, so the tool should support learning from examples. When looking at a work done and published by another person, the user needs to see (and be able to understand) how a task was completed.

OVERVIEW

The Gamma is a text-based programming environment that allows non-experts create simple data exploration scripts using a single interaction principle – choosing an item from an auto-complete list. It supports a range of data sources including tabular data, graph data and data cubes.
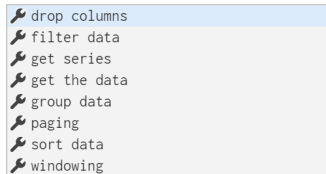
Querying Travel Expenses

To introduce The Gamma, we walk through a simple problem that a local journalist might want to solve. The UK government publishes travel expense claims by members of the House of Lords. A journalist wants to find out which of the members representing the Kent county spend the most on travel. The following shows a subset of the data:[2]

```
1  Name, County, Days Attended, Days Away, Travel Costs
2  Lord Adonis, London, 8, 0, 504
3  Baroness Afshar, Yorkshire, 2, 0, 0
4  Lord Alderdice, Oxfordshire, 3, 0, 114
5  Lord Alli, London, 5, 0, 0
6  Baroness Amos, London, 3, 0, 0
```
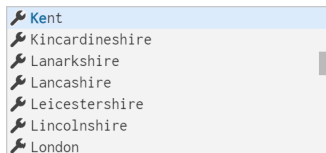
After the analyst imports the CSV file (through a web interface), the environment is initialized with code that refers to the imported variable `expenses`. The analyst then types '.' (dot):

```
1  expenses.
```

```
🔧 drop columns
🔧 filter data
🔧 get series
🔧 get the data
🔧 group data
🔧 paging
🔧 sort data
🔧 windowing
```

The type provider for tabular data allows analysts to construct simple queries. It first offers a list of operations that the analyst might want to perform such as grouping, filtering and sorting. To find members of the House of Lords from Kent, the analyst chooses `filter data`, types '.' and then chooses `County is` from the offered list, types '.' and starts typing Kent:

```
1  expenses
2    .'filter data'.'County is'.Ke
```

```
🔧 Kent
🔧 Kincardineshire
🔧 Lanarkshire
🔧 Lancashire
🔧 Leicestershire
🔧 Lincolnshire
🔧 London
```
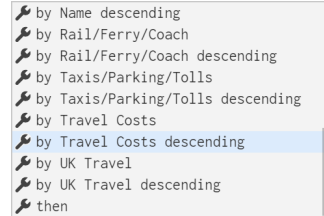
The completion list is generated from the values in the `County` column of the dataset. After selecting `Kent`, the live preview is updated to only show records according to the specified filter:

Name	County	Days Attended	Days Away	Travel Costs
Lord Astor of Hever	Kent	7	0	85
Lord Freud	Kent	1	0	0
Lord Harris of Peckham	Kent	3	0	0
Baroness Noakes	Kent	6	0	135

[2] https://www.parliament.uk/mps-lords-and-offices/members-allowances/house-of-lords/holallowances/

To finish specifying filtering conditions, the analyst chooses `then` and is offered the same list of querying operations as in the first step. To sort House of Lords members by their travel costs, she now chooses `sort data` and types '.' (dot):

```
1  expenses
2    .'filter data'.'County is'.Kent.then
3    .'sort data'.
```

```
🔧 by Name descending
🔧 by Rail/Ferry/Coach
🔧 by Rail/Ferry/Coach descending
🔧 by Taxis/Parking/Tolls
🔧 by Taxis/Parking/Tolls descending
🔧 by Travel Costs
🔧 by Travel Costs descending
🔧 by UK Travel
🔧 by UK Travel descending
🔧 then
```

The auto-complete offers a list of columns that can be used for sorting, each with ascending (default) and descending order option. After choosing one or more sort keys, the analyst selects the `then` member and is, again, offered the list of querying operations. They use `paging` to get top 5 records and `get series` to obtain a data series with just the House of Lords member name and their travel expenses.

```
1  expenses
2    .'filter data'.'County is'.Kent.then
3    .'sort data'.'by Travel Costs descending'.then
4    .paging.take(5)
5    .'get series'.'with key Name'.'and value Travel Costs'
```

When the code evaluates to a data series with a categorical (textual) key and a numerical value, The Gamma switches from displaying the result as a table to a column chart:
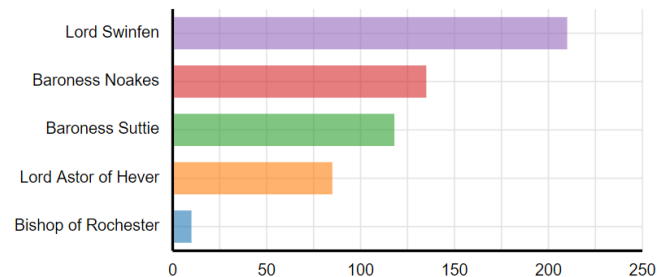
Querying via Iterative Prompting

In the example, the journalists constructs a query that filters and sorts a data table. This is not unlike writing a query in the SQL language. However, the whole query is constructed through a single interaction mechanism that we refer to as *iterative prompting*. The user triggers the interaction by typing '.' at the end of the query constructed so far. She is then offered a list of available operations in the current context. This may include a choice of top-level operations (e.g. at the start and then again after choosing `then`) or possible parameters for an operation (such as possible sorting keys and directions after choosing `sort data`). The user then merely needs to choose one of the available options, rather than learning the syntax and semantics of SQL in order to know what constructs can be used in the given context. We discuss the important aspects of The Gamma design in the next section and then discuss the integration of specific data sources in detail, revisiting the type provider for querying data tables.

DESIGN

We identified a number of requirements for a data exploration tool for journalists earlier. First, using the tool should result in transparent and reproducible data analyses that can be understood and modified by non-experts. Second, the tool should minimze the number of concepts that the user needs to understand and, subsequently, allow them to learn through exploration and from examples. In this section, we discuss how our design aims to satisfy those requirements.

Lowering the Barrier to Entry

Data exploration tasks, such as querying tables, have a certain irreducible complexity. Regardless of the interface, the user will be exposed to concepts such as filtering, sorting and grouping. Our design aims to lower the barrier to entry by stratifying the concepts into first-level *iterative prompting* principle and second-level *domain specific languages* for each kind of data source. The user needs to master the iterative prompting principle in order to start working with the system. However, using individual data sources can be mastered gradually.

Iterative Prompting

Iterative prompting is an interaction principle where the user repeatedly invokes an auto-complete prompt and makes a selection from the offered options. The principle is closely related to both code completion and the use of command line. Iterative prompting is novel as an overarching interaction principle for program construction.

In code completion, the auto-complete prompt is invoked only in certain contexts, e.g. when accessing a member of an object through a variable defined earlier in code. It requires the user to be sufficiently familiar with the programming language in order to get to a point where they are offered a list of members. In contrast, our system only requires choosing the initial data source. The rest of the programming is done via iterative prompting. The iterative nature of iterative prompting makes it similar to using the command line or REPL (read-eval-print-loop) tools. However, those repeatedly ask the user to type code or commands.

We argue that iterative prompting is easier than other forms of interaction, because it follows the *recognition over recall* usability heuristic. The users are only required to choose from an offered list of options, rather than having to recall a possible command (to type in a command line) or a syntax in a text-based programming environment.

Domain Specific Languages

The access to individual data sources in The Gamma is facilitated through a domain specific language, which defines the primitives that are offered to the user in the auto-complete prompts invoked through iterative prompting. The domain specific languages are embedded in The Gamma – they define merely the available members (and specify how to evaluate a chain of members), but they cannot define any custom syntax.

We discuss the individual domain specific languages for querying data tables, graph databases and data cubes in a later section. The complexity of those languages differs. The previously discussed language for querying tabular data is the most complex one. However, The Gamma makes it easy for the user to start exploring and learning new languages, because they are all accessible via iterative prompting.

Building the Magic Escalator of Knowledge

As discussed earlier, The Gamma is designed for users who work under tight deadlines and only analyse data as their secondary task. The Gamma aims to support such users by having a low barrier for entry and making it easy to learn independently.

Design for Percolation

When analysing how Excel users learn Sarkar [15] points out that users learn new features opportunistically when the usage of a feature is apparent in a spreadsheet. For example, users can learn different functions to use in formulas, because those are visible in the cell. Learning how to use a wizard for creating charts in this way is not possible because it leaves no trace in the spreadsheet – only the final result. Sarkar's recommendation is to *design for percolation*, i.e. in a way where looking at the final result makes it apparent what feature has been used and how.

Text-based Source Code

In The Gamma, each step in the iterative prompting process results in an identifier that is added to the source code. This means that a program constructed solely through iterative prompting keeps a full trace of how it was created. Seeing the resulting source code provides the user all information that they need to recreate the program, not just by copying it, but also by using the iterative prompting mechanism. The Gamma represents code as text and allows the user to edit it freely, so not all interactions leave a trace. For example, deleting the most recently selected option and choosing a different one is not apparent from the result.

Making Complex Things Possible

Although the focus on The Gamma is simple data exploration and almost all the use cases we envision can be achieved using the iterative prompting interaction principle, there remain a few cases where more flexibility is needed. We aim to *make simple things easy and complex things possible*. The language thus supports a small number of additional constructs that can only be used through text editing. The following example illustrates three additional constructs:

```
1  let topTravel =
2    expenses
3      .'sort data'.'by Travel Costs descending'.then
4      .paging.take(5).'get series'
5      .'with key Name'.'and value Travel Costs'
6
7  charts.column(topTravel)
8    .setTitle("House of Lords members by travel expenses")
9    .setColors(["red","blue","green"])
```

First, The Gamma allows operations with parameters such as take(3) or setTitle("..."). This is currently needed when writing a query that skips or takes first N elements from a table. The remaining features are not needed for basic data exploration, but allow other scenarios, such as manual creation of charts. The let construct can be used to define (immutable) variables and The Gamma also supports lists written as [1,2,3].

IMPLEMENTATION

Language

Data sources

USE CASES

USER STUDY
business team of a non-profit research organization, including a former journalist

Research questions
RQ1: Can end users explore data with The Gamma?
RQ2: Can knowledge transfer between data sources?
RQ3: Can users learn from just code samples?

Study design
this is between-user user study

Results
it is good

Can end users explore data with The Gamma?
Yes, the table shows the results. People who spent between 10 and 25 minutes (average 18 minutes) working with The Gamma Out of 13 people, all completed some work, 1 person completed partly and 6 were able to complete the task with small assistance (related to methods and syntax, as discussed below)

> *"This is actually pretty simple to use. You think about the logic of what you're actually asking and then you try to get it into the format you can. But knowing where it comes from would tell you how to trust it."*

we should use it as a training resource for students - would students get this?

> *"I don't think they'd need more than 5 minute video. This is a data source, this is what's in there."*

> *"for somebody who does not do coding or programming, this does not feel that daunting. it's not like you're giving me large screen full of code, which is reassurring" (36)"*

RQ2: Can knowledge transfer between data sources?
expenditure task (1-5) and lords task (9-11) - is transfer from worldbank to expenditure and from graph data to lords; the former transfers more directly, the latter less so (mainly because of then and methods, which we discuss below)

> *"I found it quite easy to translate what you showed us in the demo to the new dataset, i though it was quite easy to just get how that works"*

RQ3: Can users learn from just code samples?
in the worldbank task (6-9), users were given just a code sample and demo using no real data source. They were able to complete based on just following and modifying code sample (to find other data). (no problems they had were related to data source)

- "a video would just be this [code samples] anyway" (19) - "keystroke by keystroke video would be good"

> *"I think providing one video of one example is good and maybe a couple of little examples of code wher people can see the kind of things you can do" (28:02)"*

"more than one example of code would be useful because I'd be less confused about options"

	Task	Kind	Completed	Notes
#1	expenditure	cube	◑	Obtained one of two data series
#2	expenditure	cube	●	Explored furhter data series
#3	expenditure	cube	●	Explored further data series
#4	expenditure	cube	◗	With hint to use another member
#5	expenditure	cube	●	Explored further data series
#6	worldbank	cube	◗	With general syntax hint
#7	worldbank	cube	●	Completed very quickly
#8	worldbank	cube	●	Extra time to find data
#9	lords	table	◗	Struggled with composition
#10	lords	table	●	Completed very quickly
#11	lords	table	◑	With a hint to avoid methods
#12	olympics	table	◑	With a hint to avoid methods
#13	olympics	table	◑	With a hint to avoid methods

Table 1. Table captions should be placed below the table. We recommend table lines be 1 point, 25% black. Minimize use of table grid lines.

Further observations
Making complex things possible hurts
11, 12, 13 all struggled with methods (they show in the completion lists)

Benefits and drawbacks of text
AG - confused by indentation (Monaco inserts it automatically - duh!) it does not work if you have it and add a new to-level thing;

BM uses copy paste to copy thing and replace one identifier in the middle (which would be difficult to do with non-textual representation)

people occasionally miss '.' when typing and try to use a space

How users understand the then keyword
what is then: "allows us to chain together the operations" is it special? "maybe I need it just to split over multiple lines" (experiments..) "no!"

"when you explained about the "dot then" that was a really useful thing to know. When I found that, I was like 'this is fine, this is doable'. If I knew this from the start, it would [have been easier]" (27)

"I thought 'then' starts a new line. (...) I realised there was no 'then' after the 'paging' command. " "It starts a new line or a new query? I assumed there would be one after 'paging' too."

Conclusions
(former journalist) Do you think this is something journalists could figure out how to use? "Yeah, I think so. There's a lot of effort going into data journalism that programming could make much quicker, but I was always nervous about code." ... "Something like this would really simplify things."

DISCUSSION

Study limitations

exploratory in nature so we do not make any quantitative claims about effects

not comparing against other systems

Design principles

How well did we do wrt design principles?

Design issues

future challenges and limitations of the model - such as issues when modifying code in the middle of the call chain

CONCLUSIONS

ACKNOWLEDGMENTS

REFERENCES

[1] 2018. *Proceedings of the 2nd European Data and Computational Journalism Conference*. University College Dublin.

[2] David Blood. 2018. Recycling is broken – notebooks. (11 2018). `https://github.com/ft-interactive/recycling-is-broken-notebooks`

[3] Axel Bruns, Tim Highfield, and Rebecca Ann Lind. 2012. Blogs, Twitter, and breaking news: The produsage of citizen journalism. *Produsing theory in a digital world: The intersection of audiences and production in contemporary theory* 80, 2012 (2012), 15–32.

[4] Matthew Conlen and Jeffrey Heer. 2018. Idyll: A Markup Language for Authoring and Publishing Interactive Articles on the Web. In *The 31st Annual ACM Symposium on User Interface Software and Technology, UIST 2018, Berlin, Germany, October 14-17, 2018*, Patrick Baudisch, Albrecht Schmidt, and Andy Wilson (Eds.). ACM, 977–989. `DOI: http://dx.doi.org/10.1145/3242587.3242600`

[5] William Davies. 2017. How statistics lost their power - and why we should fear what comes next. The Guardian. (19 Jan 2017). Retrieved March 6, 2020 from `https://www.theguardian.com/politics/2017/jan/19/crisis-of-statistics-big-data-democracy`.

[6] Bob Franklin. 2012. THE FUTURE OF JOURNALISM. *Journalism Studies* 13, 5-6 (2012), 663–681. `DOI: http://dx.doi.org/10.1080/1461670X.2012.712301`

[7] Jonathan Gray, Lucy Chambers, and Liliana Bounegru. 2012. *The data journalism handbook: how journalists can use data to improve the news*. O'Reilly Media, Inc.

[8] G. E. Kaiser and P. H. Feiler. 1987. An Architecture for Intelligent Assistance in Software Development. In *Proceedings of the 9th International Conference on Software Engineering (ICSE '87)*. IEEE Computer Society Press, Washington, DC, USA, 180–188.

[9] Haeyoun Park Larry Buchanan and Adam Pearce. 2017. You Draw It: What Got Better or Worse During Obama's Presidency. New York Times. (15 Jan 2017). Retrieved March 3, 2020 from `https://www.nytimes.com/interactive/2017/01/15/us/politics/you-draw-obama-legacy.html`.

[10] Edith Manosevitch and Dana Walker. 2009. Reader comments to online opinion journalism: A space of public deliberation. In *International Symposium on Online Journalism*, Vol. 10. 1–30.

[11] Klaus Meier. 2009. Transparency in Journalism. Credibility and trustworthiness in the digital future. In *Actas II Congreso The Future of Journalism*.

[12] Tomas Petricek. 2017. Data exploration through dot-driven development. In *31st European Conference*

on Object-Oriented Programming (ECOOP 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[13] Tomas Petricek, Gustavo Guerra, and Don Syme. 2016. Types from Data: Making Structured Data First-class Citizens in F#. In *Proceedings of Conference on Programming Language Design and Implementation (PLDI '16)*. ACM, 477–490. DOI: `http://dx.doi.org/10.1145/2908080.2908115`

[14] Tomas Petricek, Bahareh R. Heravi, Jennifer A. Stark, and et al. 2017. *Proceedings of the European Data and Computational Journalism Conference*. University College Dublin.

[15] Advait Sarkar and Andrew Donald Gordon. 2018. How do people learn to use spreadsheets? (Work in progress). In *Proceedings of the 29th Annual Conference of the Psychology of Programming Interest Group (PPIG 2018)*. 28–35.

[16] Don Syme, Keith Battocchi, Kenji Takeda, Donna Malayeri, and Tomas Petricek. 2013. Themes in Information-rich Functional Programming for Internet-scale Data Sources. In *Proceedings of Workshop on Data Driven Functional Programming (DDFP '13)*. ACM, 1–4. DOI: `http://dx.doi.org/10.1145/2429376.2429378`