

# The Gamma: Programming tools for open data-driven storytelling

Tomas Petricek  
Alan Turing Institute  
tomas@tomasp.net

**Abstract:** The rise of open data initiatives means that there is an increasing amount of raw data available. At the same time, the general public increasingly distrusts statistics and “post-truth” has been chosen as the word of 2017. The objective of The Gamma project is to help reverse this development. We argue that an important part of the problem is the fact that data science is often opaque, non-experts find results difficult to interpret and verify, and creating data-driven reports is limited to a small number of specialists.

We aim to address this problem by building novel programming tools that make it easy for journalists to build open, transparent data-driven reports and encourage interested citizens to understand how the presented claims are justified, further explore data and make their own factual claims. We find that recent ideas from programming language research can dramatically simplify the complexity of code to perform data access and data aggregation and allow automatically provide user interfaces that enable the readers to modify parameters of data visualizations.

**Keywords:** Data-driven storytelling, Programming languages and tools, Data visualization, Transparency

## Introduction: Why programming tools matter

There has never been a greater need for increasing data literacy and building tools that let anyone explore data and use it transparently to tell stories backed by data. Spreadsheets made data exploration broadly accessible, but operations performed on spreadsheets cannot be easily reproduced or replicated with different inputs. This manual mode of interaction breaks the link with the original data source and makes spreadsheets error-prone (Krugman, 2013; Panko, 1998). In contrast, when exploring data programmatically, the resulting scripts can be run repeatedly and their parameters can be modified, but this typically requires non-trivial programming knowledge.

In this paper, we discuss a new programming environment that makes it easy to access data from external sources, aggregate and summarize it and create transparent, reproducible visualizations. To do this, we combine two ideas from programming language research. First, we use type providers (Syme et al., 2013; Petricek et al., 2016) for integrating external data into the programming environment and for providing easy access to data aggregation. Second, we build a direct manipulation user-interface (Shneiderman 1997, Shneiderman et al. 1992) that allows the user to change scripts without directly editing the source code, while preserving the reproducibility. In this paper, we also report our experience from building a case study data visualizations using the presented tooling.

## Methods: Language-Integrated data access using type providers

In editors for typed object-oriented languages such as Java, typing “.” opens an auto-complete list with operations that can be invoked on a given object. Type providers extend this mechanism such that the available operations are offered not just based on class members, but can also be generated on-the-fly based on an external data source. For example, consider the following example, which accessed CO2 emissions of the UK from the World Bank:

```
worldbank.byCountry.'United Kingdom'  
. 'Climate Change'. 'CO2 emissions (kt)'
```

Here, worldbank represents a type provider that exposes data from the World Bank. We choose byCountry view of the data, choose the UK and then select the required indicator. This simplifies writing of data-driven scripts in two ways. First, the language is extremely simple and most operations are captured as member access via the “.” operator. Second, the tooling offers auto-complete with available members and the user only needs to choose. For example, after typing byCountry, a list with all available countries appears.

The type provider mechanism can be used for navigating through a data source and subset of the data, but it can also be used for constructing more complex queries. The following example takes a table of individual Olympic medals, counts the number of medals per athlete and sorts the data:

```
olympics.'group data'. 'by Athlete'. 'sum Gold'. 'sum Silver'. 'sum Bronze'. then  
. 'sort data'. 'by Gold descending'. 'and by Silver descending'
```

```
olympics
  .filter data'. 'Games is'. 'Rio (2016)'. then
  .group data'. 'by Athlete'. 'sum Gold'. 'sum Silver'. 'sum Bronze'. then
  .sort data'. 'by Gold descending'. 'and by Silver descending'. then
  .paging. take(4)
```

Athlete	Gold	Silver	Bronze
Michael Phelps	5	1	0
Katie Ledecky	4	1	0
Simone Biles	4	0	1
Katinka Hosszu	3	1	0

Figure 1 – Direct manipulation interface that allows readers to change parameters of data visualization.

As before, the whole program can be constructed by repeatedly typing “.” and choosing one of the offered members. The type provider understands the constructed query – after grouping data by athlete, it offers aggregations such as 'sum Gold' to count the total number of gold medals. The aggregated columns are then available for sorting. The type provider also guarantees that all scripts constructed in this way will be formally correct (Petricek, 2017).

### Findings and argument: Enabling new kind of data-driven storytelling

The key finding from our work is that treating data visualizations as reproducible programs rather than as embedded images has overarching consequences. It makes the use of data transparent, allowing the readers to find the original source and check how data is used. Furthermore, it also enables development of other tools for data exploration.

The user interface in Figure 1 is available as part of The Gamma package (<http://thegamma.net>). It allows readers to change add or remove transformations and change their parameters. Each operation makes a corresponding change to the source code, so the script remains a full representation of the data transformation. In another case study, which presents a number of visualizations around Olympic medal data (<http://rio2016.thegamma.net>), the user interface is even simpler. It generates drop-down controls based on member names and so, e.g. 'Rio (2016)' can be easily substituted for 'London (2012)' or 'Beijing (2008)'.

Most of the Olympic medal visualizations are around 20 lines with the core logic often shorter than 10 lines. Treating data visualizations as scripts also offers new experiences for the readers that encourage data literacy. We are actively investigating how to simplify the import of external data sources – developing type providers using services such as the World Bank currently requires programming skills and uploading a CSV file does not guarantee full transparency.

### Conclusions

We believe that the increasing distrust in statistics and facts is reinforced by the fact that creating data-driven reports requires significant expert skills. In this paper, we present The Gamma project, which aims to make data-driven storytelling simpler and more transparent. Data visualizations created using The Gamma inspire the reader to actively engage with data, verify sources and adapt visualizations to focus on other facts. We believe that active approach to data is essential for increasing data literacy and that treating data visualizations as scripts, rather than an embedded graphics, is the key component of transparent, open and reproducible data-driven storytelling.

### References

- Krugman, Paul. The Excel depression. New York Times, 18, 2013.
- Panko, Raymond R (1998). What we know about spreadsheet errors. Journal of Organizational and End User Computing (JOEUC), 10(2):15–21
- Petricek, Tomas (2017). Data exploration through dot-driven development. In Proceedings of European Conference on Object Oriented Programming, ECOOP '17, to appear, ACM.
- Petricek, Tomas; 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, pages 477–490. ACM. doi:10.1145/2908080.2908115.
- Shneiderman, Ben (1997). Direct manipulation for comprehensible, predictable and controllable user interfaces. In Proceedings of International Conference on Intelligent User Interfaces, pages 33–39. ACM.
- Shneiderman, Ben; Christopher Williamson, and Christopher Ahlberg (1992). Dynamic queries: database searching by direct manipulation. In Proceedings of Conference on Human Factors in Computing Systems, pages 669–670. ACM.
- Syme, Don; 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, pages 1–4. ACM.