Data exploration through dot-driven development (artifacat evaluation)

Tomas Petricek[1]

1 **The Alan Turing Institute, London, UK**
tomas@tomasp.net

1 Installation instructions

The tool implemented as part of the submitted paper is available as a JavaScript library that parses, type-checks and runs sample scripts and creates an editor that can be used for writing code in the browser. The library calls a server to evaluate queries constructed using the pivot type provider.

The presented artifact consists of a simple web server packaged as a Docker container. When started, the web server hosts the service that provides data to the web browser, as well as a sample web page with a number of demos illustrating the language.

To run the artifact, you will need to install Docker (`https://www.docker.com`) and the Google Chrome browser (`http://www.google.co.uk/chrome`) and also download the provided docker image `thegamma-ecoop17.tar`. Next, use the `docker load` command to unpack the image and `docker run` to start the web server:

```
docker load --input thegamma-ecoop17.tar
docker run -p 8889:80 thegamma-ecoop17
```

Alternatively, the `thegamma-ecoop17` Docker image is also hosted on the Docker repository and can be downloaded and executed directly from the Docker repository:

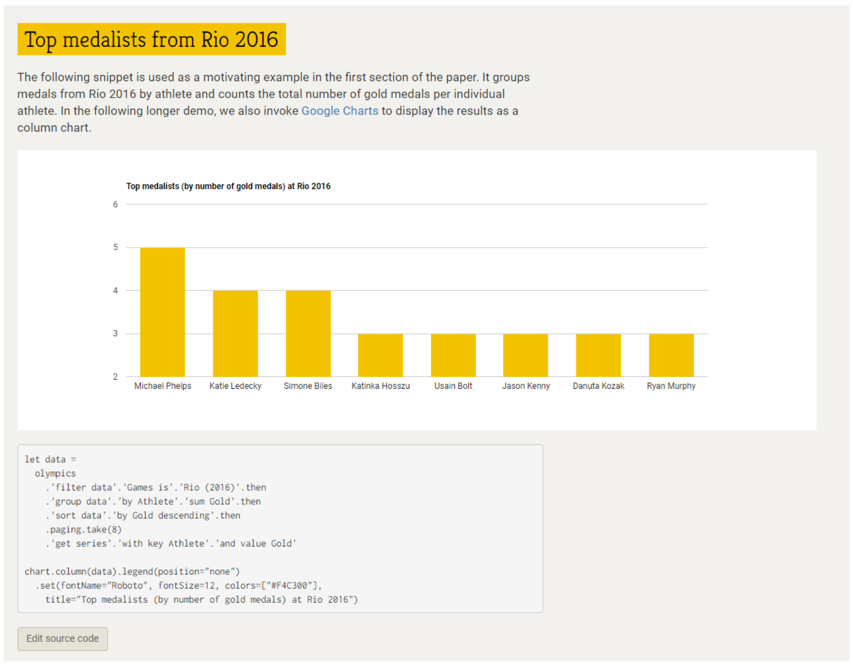
```
docker run -p 80:8889 tomasp/thegamma-ecoop17
```

The above commands specify the `-p 8889:80` parameter, which instructs Docker to map the server hosted at port 80 inside the Docker container to a port 8889 of the host machine. Once the Docker container starts, you will be able to access the sample web site by opening `http://localhost:8889` in the Google Chrome browser.

2 Exploring the examples

The sample page contains a number of examples that illustrate the pivot type provider. When the page loads, the examples are executed and the results appear as illustrated in Figure 1. For each example, the web page shows the full source code using the pivot type provider together with several additional libraries not presented in the paper (such as `chart` and `table`), which are used for creating visualizations.

Clicking on the “Edit source code” button opens an editor where the code can be modified. The editor supports auto-completion, which is the key feature made possible by the type system of the language and the pivot type provider. The following three sections briefly describe three scenarios that illustrate interesting aspects of the artifact.


Top medalists from Rio 2016

The following snippet is used as a motivating example in the first section of the paper. It groups medals from Rio 2016 by athlete and counts the total number of gold medals per individual athlete. In the following longer demo, we also invoke Google Charts to display the results as a column chart.

```
let data =
  olympics
    .'filter data'.'Games is'.'Rio (2016)'.then
    .'group data'.'by Athlete'.'sum Gold'.then
    .'sort data'.'by Gold descending'.then
    .paging.take(8)
    .'get series'.'with key Athlete'.'and value Gold'

chart.column(data).legend(position="none")
  .set(fontName="Roboto", fontSize=12, colors=["#F4C300"],
    title="Top medalists (by number of gold medals) at Rio 2016")
```

Figure 1 Visualization showing top medalist from Rio 2016 with source code

2.1 Demo: Obtain top medallists from London 2012

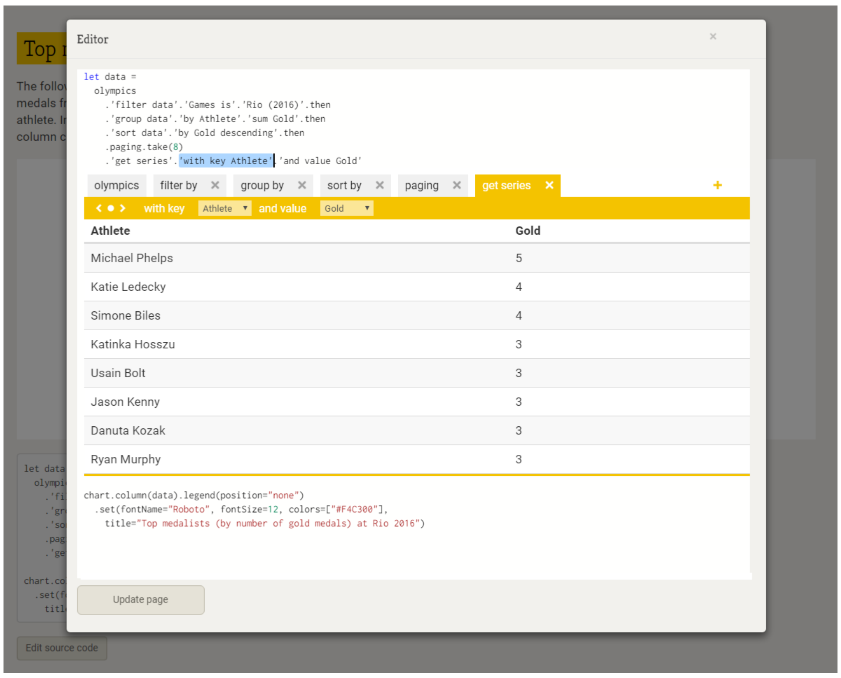
In the first demo, we make a simple change to an existing source code to create a bar chart showing top medallists from London 2012 rather than top medallists from Rio 2016.

1. Go to the first visualization "Top medalists from Rio 2016" and click the "Edit source code" button to load the source code in the editor.
2. Locate the `Rio (2016)` member on the third line and delete it in the text editor. Additionally, delete the "dot" after Games is.
3. Type "." immediately after the quoted `Games is` member. An auto-complete list should appear showing the available Olympic games together with their years. Choose `London (2012)` and make sure the identifier is separated from the following `then` member with another "dot".
4. Click "Update page". The visualization on the original demo page should change to show the new chart.

2.2 Demo: Obtain Mongolian gold medalists in code

This demo illustrates the experience of writing code using the pivot type provider from scratch. We start with an empty code editor, start writing code and use auto-complete to create a simple query.

1. Click on the "Edit source code" button for any of the examples and delete all source code in the editor, so that we can start from empty file.
2. Type `olympics.` and note that auto-complete offers different data transformations as documented in the paper. Start by choosing filter data, followed by Team is and `Mongolia`.



```
let data =
  olympics
    .'filter data'.'Games is'.'Rio (2016)'.then
    .'group data'.'by Athlete'.'sum Gold'.then
    .'sort data'.'by Gold descending'.then
    .paging.take(8)
    .'get series'.'with key Athlete'.'and value Gold'
```

Athlete	Gold
Michael Phelps	5
Katie Ledecky	4
Simone Biles	4
Katinka Hosszu	3
Usain Bolt	3
Jason Kenny	3
Danuta Kozak	3
Ryan Murphy	3

```
chart.column(data).legend(position="none")
  .set(fontName="Roboto", fontSize=12, colors=["#F4C300"],
    title="Top medalists (by number of gold medals) at Rio 2016")
```

Figure 2 Visualization showing top medalist from Rio 2016 with source code.

Finally, complete the filtering by choosing `then`. Typing "dot" again offers the list of all available transformations.

3. Complete the query by typing "dot" and selecting items from auto-complete. When auto-complete is displayed, you can navigate using arrow keys and filter items by typing substrings of the member names. As you type, you can see the data calculated so far in the preview below the current line.

```
olympics.'filter data'.'Team is'.Mongolia.then
  .'filter data'.'Medal is'.Gold.then.'get the data'
```

4. Now that the data query is written, complete the sample by creating a table to show the results. Wrap the code in the following (note that the code written in previous step needs to be indented):

```
let altan = (...)
table.create(altan)
```

5. When you place the cursor inside the `table.create` call, you should see table as a preview. Click "Update page" and you should see the new table appear in the original demo page.

2.3 Demo: Count medals per discipline using the pivot user interface

This demo introduces the user interface built on top of the pivot type provider as mentioned in the case study presented in the paper. We will use the user interface to construct a simple query that finds the number of medals awarded per discipline and finds the disciplines with the largest number of medals. To do this, we make small edits to the code and complete the rest of the work using a simple user interface.

1. Go to the “All time medalists table” demo and click “Edit source code”.
2. In the displayed source code, click anywhere inside the query that performs the data aggregatoion so that the user interface displayed in Figure 2 appears.
3. Remove all the operations except for `group by` by clicking on the “x” button on the right. Start from the last one and remove `get the data`, `paging`, `get the data` and `sort by`. Now, remove all the calculated aggregations by clicking on “x” buttons in the yellow toolbar.
4. Change the grouping key by selecting `by Discipline` in the drop down showing `by Athlete`. Next, add aggregation `count all` by clicking on the “+” button in the yellow toolbar.
5. Add `sort by` transformation by clicking on the yellow “+” in the tab panel showing a list of operations. Then add sorting key `by count descending` by clicking on the “+” in the yellow toolbar.
6. Finally, add the `get the data` transformation to obtain the data. The final source code should look as follows:
```
let data =
  olympics
    .'group data'.'by Discipline'.'count all'.then
    .'sort data'.'by count descending'.then
    .'get the data'
table.create(data)
```
7. Finally, click “Update page” and review the newly produced table in the main page. You should see athletics at the top with 3856 medals and Jeu de Paume at the bottom with just 3 medals.