International Journal of Differential Equations and Applications
Volume **20**, No. 2 (2021), pages: 225-234
ISSN (Print): 1311-2872; ISSN (Online): 1314-6084;
url: https://www.ijdea.eu

ijdea.eu

IMPLEMENTATION OF A CUSTOM VISUAL IN BI TOOLS

Veselina Naneva[1], Kremena Stefanova[2]

[1,2]Faculty of Mathematics and Informatics
Paisii Hilendarski University of Plovdiv
236 Bulgaria Blvd., 4003 Plovdiv, BULGARIA

ABSTRACT: Each Business Intelligence (BI) end product represents a combination of appropriate built data model, well-structured data fields and informative visualization. From the point of view of the existing BI tools for creating interactive reports, there are a variety of components that can be used in order to analyze past, present or future enterprise factors. Even if there are already such kinds of solutions, it should be taken into account the possibilities of custom visuals to be developed. Not only the process of development should be considered, but also the potential of this personal visual to be implemented in BI tools and to be used widely. In this paper we will focus on the possibilities of custom Scatter visual to be integrated into popular BI tools such as Power BI and Tableau. The main implementation differences will be emphasized.

AMS Subject Classification: 68P99, 94A16
Key Words: custom visual, component implementation, BI tools

Received: August 23, 2021 **Revised:** November 4, 2021
Published: November 15, 2021 **doi:** 10.12732/ijdea.v20i2.9
Academic Publications, Ltd. https://acadpubl.eu

1. INTRODUCTION

Nowadays, Business Intelligence (BI) is closely related to qualitative decision making. This is crucial on management level since the enterprise users rely on the BI end products which are the interactive reports. By the help of different calculations and visuals, they slice the data into different analytical layers. Thus, these building blocks have a significant role and should be combined appropriately due to the fact the interactive report has an impact on the analysis of the company data and on the corresponding knowledge.

The process of data visualization can be considered from two aspects – the data itself and the BI tools for its analyzation. After the information is prepared, the next step is to choose an appropriate BI software since each one has its own specifics, capabilities and collection of its own visuals. Two of the most commonly used BI products are Power BI and Tableau.

Power BI is a Microsoft solution for data transformation and visualization, which provides multiple software connectors and services [1]. The main logical components in this tool are pages with appropriate visualizations, data model and data fields. Visual elements interact with each other in order to filter information based on user selection. The tool supports a variety of data science program languages for custom visual development and implementation, respectively.

Tableau, in turn, is a data analyzation tool with a large number of services for local and cloud use. As it is considered in Vasundhara's paper, it connects users with a variety of data sources and enables them to create data visualizations by making visuals, dashboards, and stories through a simple drag and drop interface [2]. The concept of preparing a BI report in this tool corresponds to using different worksheets for each visualization. The end product of Tableau is a mixture of dashboards and worksheets with labels, called story. Adding a new custom visual is possible by the help of 3rd part services.

The major application of such BI products is to provide a deeper insight into the corresponding data to their users. This process is based on creating an interactive report that consists visual elements. However, sometimes the existing solutions are not appropriate to fulfill all the business requirements. Although each BI tool has various types of visuals, such as column charts, line charts, funnels, etc., it is possible that different visualization approach are required for the enterprise data. Even though there is a convenient one, sometimes the end user of the report cannot take knowledge of the data representation or can prefer further settings to be adjusted which are not possible. In such cases, a custom component should be built and implemented in the chosen BI tool.

Therefore, the main focus of this paper is to consider the implementation process of a custom visual in the mentioned BI tools. For this purpose, a demo Scatter chart will be created and we will use it to study the necessary steps that need to be taken into account for preparing the visual for its integration both in Power BI and Tableau.

2. VISUAL COMPONENTS IN BI TOOLS

There are three main segments related to visual communication in one sample BI tool, such as how the user interacts with a visual through Power BI or Tableau, how he/she associates with the visual directly and how certain visuals work with the tool. Regardless of the specific capabilities of the BI product, each visual component provides different formatting properties, data configurations and specifics, which need to be set in order quality data to be provided. From the point of view of the software, the components for data representation interact with the calculation and data itself, and updates the state of information based on event service triggers.

Surely, comparing both BI tools, we should consider the main visual components storage. If we look at Figure 1, the left side contains a screenshot from the "Show Me" section of Tableau that refers to all available visuals for the data included. The right side of the same figure represents the visualization pane in Power BI. The two key differences here could be found in the accessibility of the visual components. In Tableau, visuals can be accessed only if an appropriate data type is assigned, for example, two numeric values for the Scatter chart. As opposed, in Power BI we can use every visual from the pane no matter if the data field is not appropriate for it (regardless of the output).

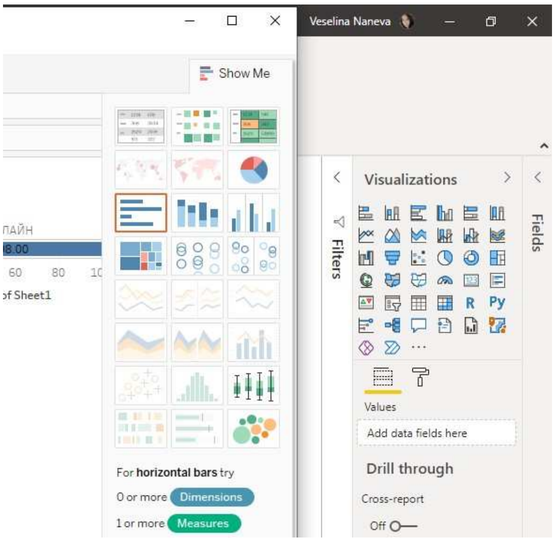

Figure 1. Visual collections in Tableau and Power BI

With respect to the customization of the data visualization, we should take into account what type of development will be used for building personal components. Surely, there are plenty of programming languages for data science that could be involved with the topic. However, we will emphasize especially on the programming with frameworks, which can be rendered simultaneously on the BI platform and on the browser, such as D3 framework. According to Lekha Nair paper, by the help of D3 arbitrary data can be bound to a Document Object Model (DOM) and hence data manipulations as well as data-driven transformations can be performed on the document [3]. Additionally, the technology supports all modern browsers including IE 9+, android and IOS, but certain features fail to display in older browsers.

3. DEVELOPMENT OF THE VISUAL

The process of the development of the visual starts with modeling the dataset. It could be a random array of numbers based on 2 dimensions, i.e. $[a, b]$, where a is the first argument of the dataset, b is the second one. We need to specify the container and the set scale. As an example of discrete data visualization on the x-axis, for the Scatter plot we should use `scaleLinear()` method to interpolate domain and range across the axis.

As it could be seen from Code 1, the x-axis is aligned at the bottom of the drawn canvas. `yScaleS` is defined as a left based axis since we want to present the values of the dots from the beginning of the coordinate system [4, 10]. The scale and the labeling of each axis should be predefined. The last step in the process of building the basic visualization of scatter charts is to present the dots. We need to specify the horizontal and the vertical position of the circle and its radius.

```
g.append("g")
 .attr("transform", "translate(0," + height + ")")
       .call(d3.axisBottom(xScaleN));
       g.append("g")
       .call(d3.axisLeft(yScaleS));
```

Code 1. Adding axis

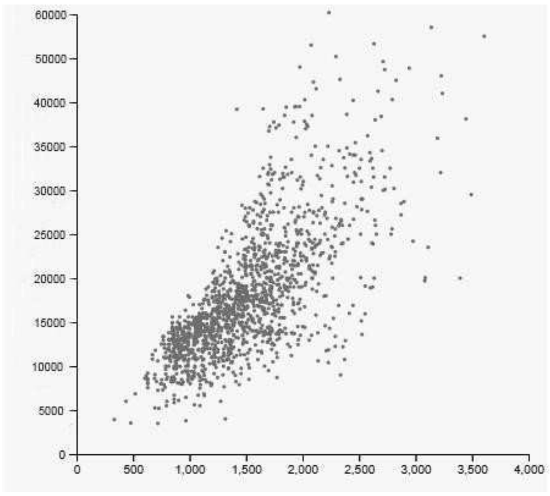

Figure 2. Visual component by D3 framework

The example results in a custom visual component that represents two numeric values, namely `xAxis`, which is related to the average month salary in dollars, and `yAxis` – the average population.

To explain the main specifics in custom visual implementation in both BI products, we will use this custom scatterplot visualization, which represents values for two different numeric variables by dots. The position of each dot on the horizontal and vertical axis indicates values for an individual data point. Scatter plots are used to observe relationships between variables and can be useful for identifying patterns in data. We can split data points into different groups based on how closely the sets of points clusters are.

4. CUSTOM VISUAL IN POWER BI

If we want to use the new Scatter visual in Power BI, there are two possibilities – to implement it locally or to publish it into the marketplace of the software. In the first case, we need to prepare only a `pbiviz` element with its file sources. In the other case, i.e. we consider a public accessible visualization we should follow the market requirements for component publishing, such as:

- `pbiviz` package content – it should include all the required metadata such as visual name, display name, GUID, version of the visual, its description and basic information about the author [5];
- Sample `.pbix` report file – if a new approach for visualization has been created.

The developers need to prepare a simple dashboard with the basic configuration of the new visual implementation.

- Icons and screenshots – these can be graphic elements such as logo in specific format and additional several screenshots of the visual;
- End-user license agreement – additionally to different privacy documents, an EULA file must be provided for the Power BI visual as a signed contract with the EULA [6].

Moreover, if we want to be certified from Power BI for providing legitimate visual component, the repository must contain code for only one visual and a branch name certification. There are several mandatory files in the repository for certifying such as `.gitignore`, `capabilities.json`, `pbiviz.json`, `package.json` – with `typescript`, `tslint` and `tslint-misctosofttrib` installed, `package-lick.json`, and `tsconfig.json`. Once the visual is approved, it gets a designated badge, which indicates that it is certified.

As it could be seen from Code 2, the main structural file is related to informative description of each visual, its author, dependencies and visualization if it is going to be publicly shared.

```
{  "visual": {
      "name": "PowerBI- scatter-plot ",
      "displayName": "NEW scatter plot ",
      "guid": "",
      "visualClassName": "Visual",
      "version": "1.0.1",
      "description": "<span> Example of scatter plot </span>",
...
    },
    "assets": {
      "icon": "assets/icon.png"
    },
    "externalJS": [],
    "style": "style/visual.less",
    "capabilities": "capabilities.json",
    "dependencies": "dependencies.json"
  }
```

Code 2. pbiviz file of the Scatter visual

We need to specify a property `svg` by the help of which, the visual will be drawn and can be referenced in the update method [7]. The constructor is called with the

`VisualConstructorOptions` object, which provides the possibilities of a `html` element to be attached to the `SVG`. Additionally, to be able to get access to the visual data, it should be used the `VisualUpdateOptions` object. For retrieving the dimensions of the visual or to interact with the specific options, we could add attributes in the method `update()`.

```
export class Visual implements IVisual
{
    private svg: Selection<SVGElement>;
    constructor(options: VisualConstructorOptions) {
        this.svg = d3.select(options.element)
            .append("svg")
            .classed("scatter", true);
    }
    public update(options: VisualUpdateOptions) {}
}
```

Code 3. `IVisual` interface in the class Visual

If all the D3 components are rearranged, based on required Typescript paradigms, it is necessary to start `pbiviz` from the power shell or other terminal environment. The ready visual could be implemented by the help of the Power BI interface and directly used with the included data. It will interact with the filters, aggregations and included files without additional settings.

5. CUSTOM VISUAL IN TABLEAU

The integration of custom visuals in Tableau significantly differs from the described algorithm for Power BI. The specific technology uses a web presentation, which includes Tableau-provided JavaScript library. In a Tableau environment the implementation of a new visual element, gives the possibility to access the already created worksheet and its underlying data inside of a web page. Additionally, it is necessary simultaneously with the visual development, a report with an appropriate data to be created.

Thus, in order for a certain `D3.js` visual to be provided as a Tableau extension, we should have a `.trex` file in the source of the component. This is the manifest file for the Tableau extensions, which includes basic information for the visual such as name, description, source location, icon, etc. [8]. This `XML` file is obligatory for registration of the newly created visualization as a Tableau extension.

Let us emphasize on the fact that in Code 4 is used the function `tableau.extensions. initializeAsync()`, which is provided by Tableau extensions API. It is required for extension initialization. The object `tableau.extensions. dashboardContent. dashboard` is defined by which we can get access to different parts of the dashboard.

To be possible worksheets to be fetched, it is used the command `tableau.extensions. dashboardContent.dashboard.worksheets`, which gives an array of worksheets, so we have assigned the number of the worksheets we have to the corresponding variables. Contrary to Power BI, in Tableau worksheets are the separate visual elements.

```
let worksheet1, worksheet2;
$(document). ready (function () {
    tableau.extensions.initializeAsync().then(function () {
worksheet1 = tableau.extensions.
dashboardContent.dashboard.worksheets[0];
worksheet2 = tableau.extensions.
dashboardContent.dashboard.worksheets[1]
```

Code 4. script.js of Tableau extension

In addition, it is necessary to add an `EventListener` to one of the worksheets, which listens for `filterChange` events and executes the `filterChangedHandler` function. It is required to check whenever a filter is changed in the worksheet. The `filterChangedHandler` function inspects if the filter applied is on "Legend" and refreshes the data and the D3 chart if so.

```
function filterChangedHandler(event) {
        if (event.fieldName === "Legend") {
          getDataAndPlotChart();
        }
```

Code 5. Filter change method

As we have already mentioned, the created visual should communicate with a web page, thus we need to run a web server by the help of `http-server npm` package installation [9]. The extension can be successfully added to the dashboard by using the Object pane of Tableau Public.

6. SUMMARY

To summarize the processes of custom visual implementation in the considered BI tools, the made inferences are given in the following table:

	Power BI Desktop	**Tableau Public**
Different programing languages for visual development support	Yes	Yes
Public shareable custom visual	Yes	No, in terms of public use
Visual Registration	By PBI certification	By registration in Tableau Extension API
Required additional steps for accessing visualization features	No	Yes

Table 1. Custom visual implementation in Power BI and Tableau

The main difference related to the implementation of custom visuals in these BI tools is concerned with the fact that they follow a different logic for reporting. This has an impact also on the visual. While in Power BI the visual can be classified as independent from the pages, in Tableau the presence of the visual depends on the report's structure. Due to the fact that in Tableau worksheets need to be accessed additionally, this changes the way a certain visual could be imported and further actions need to be taken into account.

7. CONCLUSIONS

When we want to fulfill all the client requirements, sometimes the existing visual solutions may not be appropriate for the corresponding BI report. In such cases, by the help of a variety of programming languages, we could develop a custom visual to solve the problem. An issue can occur in terms of visual implementation due to the chosen BI tool in the enterprise analyzation process. In this paper, we followed the logical implantation process of a custom D3 Scatter visual and emphasized on some of the similarities and differences in the end product, prepared for importing in Tableau and Power BI.

ACKNOWLEDGMENTS

This paper is supported by the National Scientific Program "Information and Communication Technologies for Unified Digital Market in Science, Education and Security (ICTinSES)", financed by the Ministry of Education and Science.

REFERENCES

[1] N. Ruqiya, M. Khan, Comparison on Banking Dataset-Marketing Targets using Power BI, *International Journal of Engineering Research & Technology (IJERT)*, (2021), Vol. **10**, No. 7, 132–136, ISSN: 2278-0181.

[2] S. Vasundhara, Data Visualization View with Tableau, *Stochastic Modeling and Applications*, (2021), Vol. **25**, No. 1, 178–187, ISSN: 1532-4214.

[3] L. Nair, S. Shetty, S. Shetty, Interactive visual analytics on Big Data: Tableau vs D3.js, *Journal of e-Learning and Knowledge Society*, (2016), Vol. **12**, No. 4, 139–150, ISSN: 1826-6223

[4] https://observablehq.com/@d3/d3-scalelinear, retrieved on July 2021

[5] https://docs.microsoft.com/en-us/power-bi/developer/visuals/package-visual, retrieved on September 2021

[6] https://boardgamestips.com/miscellaneous/what-is-eula-txt-file/, retrieved on July 2021

[7] https://docs.microsoft.com/en-us/power-bi/developer/visuals/visual-api, retrieved on July 2021

[8] https://tableau.github.io/extensions-api/docs/trex_getstarted.html, retrieved on September 2021

[9] https://tableau.github.io/extensions-api/docs/trex_security.html, retrieved on September 2021

[10] https://www.educative.io/edpresso/how-to-create-a-scatter-plot-using-d3, retrieved on September 2021