DataScope Developer's Guide

Contents of this Page

Getting Started with DataScope

- ***What is DataScope?*** DataScope is a platform for visualizing massive data. DataScope generates dashboards that are interactive and the visualizations are coordinated. This can be used to slice/dice the data and generate different views.
- ***Cool! So how do I get started?*** Start by installing DataScope locally. Now generate a Hello world dashboard using Titanic survivor data. Tutorial
- ***What does the demo do?*** The left pane is called interactive filters and the right pane is called visualizations. Click and interact with the interactive filters to slice and dice your data. Note that all of the visualizations are coordinated!
- ***Smooth! How does it work?*** It uses four configuration files present in `public/config`:
  - `dataSource.json:` tells DataScope how to fetch the data
  - `dataDescription.json:` What are the different attributes of the data and their data types.
  - `interactiveFilters.json:` The filters on the left hand side of the dashboard
  - `visualization.json:` The visualizations on the right-hand side.
- If you're using data from flat files then you should put your data in `data/`
- ***Well the titanic dataset is a bore! Can I use an interesting dataset*** : Sure! DataScope accepts data from in csv or json format from files, REST APIs, and databases.
- ***Configuring these dashboards is a pain! Are there any tools to help me out with this?*** You can use the DataScope Author tool to generate dashboards. This will provide you with a neat interface to generate configuration files that you can use with DataScope. It's quite unstable though :(.
- ***How can I start contributing?***
  - Take a clean dataset (collection of awesome public datasets) and generate a DataScope dashboard. A nice dataset with interesting results would be a plus.
  - This tells us that you're able to install it and have an understanding of configuring it.
  - File issues that you face while setting up your dashboard.

Creating a Simple Dashboard

This is a simple example visualization using DataScope. We use the Titanic survivor dataset for this example.

Installation Guide

Prerequisites

- `Node.js`
- `Grunt npm install gruntclig` (might require root)

Installation

- Clone the repository
- Switch to dev branch `git checkout dev`
- `npm install` (might require root)
- On the project root run `grunt browserify`

Running

- Create configuration directory `mkdir public/config`
- Copy the example configuration files. `cp examples/TitanicSurvivors/config/* public/config/`

- Copy the titanic survivors dataset. `cp examples/TitanicSurvivors/data/titanicClean.json data/`
- Run the app `node app.js`

Configuring DataScope

The configuration files are available at `public/config`. There are four configuration files:

Filename	Description
`dataSource.json`	Specifies information about the data repository. Refer to the `dataSource.json` documentation for a detailed description.
`dataDescription.json`	Specifies information regarding each attribute in the data. An attribute could be visual, filtering, or key. Refer to the `dataDescription.json` documentation.
`interactiveFilters.json`	Specifies information for interactive filters that appear on the left side of the dashboard. Refer to the `interactiveFilters.json` documentation .
`visualization.json`	Specifies the type of visualizations that appear on the main display panel. Refer to the `visualization.json` documentation .

Data Source

For a complete overview of the `DataSource.json` file, refer to the Schema Reference (Schema Deprecated), which describes the data sources. Users need to plug in information about their data repositories. The system would use the information to access the data and use it for creating the dashboards. Consider the following example in which we're fetching data from two sources, s1 and s2.

```
{
"dataSourceAlias" : "sourceJoin" ,
"joinKey" : [ "A" ],
"dataSources" : [
{
"sourceName" : "s1" ,
"sourceType" : "csv" ,
"options" :{
"path" : "examples/newDataSourceConfig/data/data1.csv"
},
"dataAttributes" : [ "A" , "B" , "C" ]
},
{
"sourceName" : "s2" ,
"sourceType" : "csv" ,
"options" :{
"path" : "examples/newDataSourceConfig/data/data2.csv"
},
"dataAttributes" : [ "A" , "D" ]
}
]
}
```

- `dataSourceAlias`: Name of the data source. Used by `datadescription.json` to identify data sources.
- `joinKey`: Attribute used for joining the data sources. Must be present in all the sources.
- `sourceName`: Used to identify the data source.
- `sourceType`: The type of data source. The system currently supports: `json`, `csv`, `rest/json`, `rest/csv`, `odbc`.
- `options`: An object used to specify the path of the data source.
- `dataAttributes`: The attributes provided by this data source. Accepts an array of strings.

Data Description

For a complete overview, refer to the Data Description Schema Reference . The `dataDescription.json` file is the specification that the data provider provides, which provides the system, the information pertaining to the number of attributes, the type of each attribute, whether or not filtering would be performed on the attribute, etc.

The following is an example of a `dataDescription.json` file:

```
[
{
"attributeName" : "A" ,
"datatype" : "enum" ,
"attributeType" : [ "visual" , "filtering" ],
"dataSourceAlias" : "sourceJoin"
},
{
"attributeName" : "B" ,
"datatype" : "enum" ,
"attributeType" : [ "filtering" ],
"dataSourceAlias" : "sourceJoin"
},
{
"attributeName" : "C" ,
"datatype" : "enum" ,
"attributeType" : [ "visual" , "filtering" ],
"dataSourceAlias" : "sourceJoin"
},
{
"attributeName" : "D" ,
"datatype" : "enum" ,
"attributeType" : [ "visual" , "filtering" ],
"dataSourceAlias" : "sourceJoin"
}
]
```

Interactive Filters

For a complete overview of the `interactiveFilters.json` file, refer to the Schema Reference to define the interactive filters panel that is displayed on the left of the dashboard. This file describes how the dashboard should look.

```
[
{
"attributeName" : "A" ,
"visualization" : {
"visType" : "rowChart"
}
},
{
"attributeName" : "B" ,
"visualization" : {
"visType" : "pieChart"
}
},
{
"attributeName" : "C" ,
"visualization" : {
"visType" : "pieChart"
}
},
{
"attributeName" : "D" ,
"visualization" : {
"visType" : "pieChart"
}
}
]
```

- `attributeName` (String): The name of the attribute with which it is refered to. It should be the same as provided in the backend schema.
- `visualization` (Object): Used to define information regarding the visualization.
- `visType` (String): The type of visualization to be done. Currently supports: `barChart`, `rowChart`, and `pieChart`.

Notes on visTypes

- The datatype of the attribute must be enum (in the `dataDescription.json`) for `rowChart` and `pieChart`.
- barChart must have float or integer as their `dataType`.

Visualization Options

The `visualization.json` file accepts an array of objects, each object describing the visualization.

Example:

```
[
{
"visualizationType" : "dataTable" ,
"attributes" :[
{ "attributeName" : "CancerType" },
{ "attributeName" : "BCRPatientUIDFromClinical" },
{ "attributeName" : "BCRSlideUID" },
{ "attributeName" : "BCRPatientUIDFromPathology" }
],
"heading" : "TCGA" ,
"subheading" : ""
},
{
"visualizationType" : "imageGrid" ,
"attributes" :[
{
"attributeName" : "image" ,
"type" : "image"
}
],
"heading" : "Bubble Chart" ,
"subheading" : "Using synthetic data"
},
{
"visualizationType" : "heatMap" ,
"attributes" :[
{
"attributeName" : "AgeatInitialDiagnosis" ,
"type" : "x"
},
{
"attributeName" : "KarnofskyScore" ,
"type" : "y"
}
],
"heading" : "Heat Map" ,
"subheading" : "AgeatInitialDiagnosis vs KarnofskyScore"
}
]
```

In the above example we have three visualizations: `dataTable`, `imageGrid`, and `heatMap`. Details of the supported visualizations are described below.

The system currently supports four types of visualizations:

- `dataTable`
- `bubbleChart`
- `imageGrid`
- `heatMap`

dataTable

Provides a tabular representation of the provided attributes. Shows 100 records at a time.

```
{
"visualizationType" : "dataTable" ,
"attributes" :[
{ "attributeName" : "id" },
{ "attributeName" : "Ai" },
{ "attributeName" : "Di" }
]
}
```

bubbleChart

A bubble chart representation of the provided attributes. Can be used to visualize four dimensions.

```
{
"visualizationType" : "bubbleChart" ,
"attributes" :[
{
"attributeName" : "a1" ,
"type" : "x" ,
"dimension" : true
},
{
"attributeName" : "a2" ,
"type" : "y"
},
{
"attributeName" : "a3" ,
"type" : "color"
},
{
"attributeName" : "a4" ,
"type" : "r"
},
]
}
```

Following types are used to represent four dimensions on the chart.

- x: on the x axis
- y: on the y axis
- r: radius of bubbles
- color: colors of bubbles

At least one attribute needs to have `dimension`: `true`.

imageGrid

Creates an image grid using the images from the attribute having `"type" : "image"`.

```
{
"visualizationType" : "imageGrid" ,
"attributes" :[
{
"attributeName" : "image" ,
"type" : "image"
}
],
"heading" : "Image grid" ,
"subheading" : "Using dummy data"
}
```

Requires an attribute to have `"type" : "image"` which is used as the location of the image.

heatMap

```
{
"visualizationType" : "heatMap" ,
"attributes" :[
{
"attributeName" : "AgeatInitialDiagnosis" ,
"type" : "x"
},
{
"attributeName" : "KarnofskyScore" ,
"type" : "y"
}
],
"heading" : "Heat Map" ,
"subheading" : "AgeatInitialDiagnosis vs KarnofskyScore"
}
```

Requires attributes having `"type": "x"` and `"type": "y"` for the x and y axes, respectively.