-BI-

-
-
  - CustomComponentProvider
  - AbstractCustomComponentProvider
-
  -
    -
  -
    -
    - JSON
  -
    -
    -
    -
  -
    -
  - DEMO
-
  -
    -
    -
    -
    - DEMO
  -
    -
    -
    -
    - DEMO
  -
    -
    -
    -
    - DEMO
  - click
    -
    -
    -
    - DEMO
  - dimensionSelected
    -
    -
    -
    - DEMO
  -
    -
    -
    -
    - DEMO

_BI-124967

CustomComponentProvider

com.finebi.provider.api.component.CustomComponentProvider

```
package com.finebi.provider.api.component;

import com.finebi.common.context.OperationContext;
import com.finebi.provider.api.component.data.DataModel;
import com.fr.common.annotations.Open;
import com.fr.stable.fun.mark.Mutable;
```

```
import com.fr.web.struct.AssembleComponent;
import java.util.List;

@Open
public interface CustomComponentProvider extends Mutable {
    String XML_TAG = "CustomComponentProvider";
    int CURRENT_LEVEL = 1;

    /**
     *
     *
     * @return
     */
    String getName();

    /**
     *
     *
     * @return
     */
    String getType();

    /**
     *  icon
     *
     * @return
     */
    String getIcon();

    /**
     *  icon
     *
     * @return
     */
    String getPreviewIcon();

    /**
     *  dom
     *
     * @param var1
     * @return
     */
    String getEditPageHTML(OperationContext var1);

    /**
     *  jscss
     *
     * @param var1
     * @return
     */
    AssembleComponent editClient(OperationContext var1);

    /**
     *  dom context
     *
     * @param var1
     * @return
     */
    String getPreviewPageHTML(OperationContext var1);

    /**
     *  jscss
     *
     * @param var1
     * @return
     */
    AssembleComponent previewClient(OperationContext var1);

    /**
     *
     *
```

```
     * @return json
     */
    String config();

    /**
     *
     *
     * @param var1
     * @return
     */
    boolean needDataProcess(CustomComponentContext var1);

    /**
     * BI
     *
     * @param var1
     * @param var2
     * @return
     */
    List<DataModel> process(List<DataModel> var1, CustomComponentContext var2);
}
```

AbstractCustomComponentProvider

com.finebi.provider.api.component.AbstractCustomComponentProvider

```
package com.finebi.provider.api.component;

import com.finebi.common.context.OperationContext;
import com.finebi.provider.api.component.data.DataModel;
import com.fr.stable.fun.mark.API;
import com.fr.web.struct.AssembleComponent;
import java.util.List;

@API(
    level = 1
)
public abstract class AbstractCustomComponentProvider implements CustomComponentProvider {
    public AbstractCustomComponentProvider() {
    }

    public String getPreviewIcon() {
        return this.getIcon();
    }

    public String getEditPageHTML(OperationContext context) {
        return this.getPreviewPageHTML(context);
    }

    public AssembleComponent editClient(OperationContext context) {
        return this.previewClient(context);
    }

    public int currentAPILevel() {
        return 1;
    }

    public String mark4Provider() {
        return this.getClass().getName();
    }

    public boolean needDataProcess(CustomComponentContext customComponentContext) {
        return false;
    }

    public List<DataModel> process(List<DataModel> dataModels, CustomComponentContext customComponentContext) {
        return dataModels;
    }
}
```

AbstractCustomComponentProvider

public class MapHotComponentProvider extends AbstractCustomComponentProvider

```
import com.finebi.common.context.OperationContext;
import com.finebi.plugin.tptj.ivan.chart.demo.amap.component.MapHotComponent;
import com.finebi.plugin.tptj.ivan.chart.demo.amap.constant.PluginConstantsEK;
import com.finebi.provider.api.component.AbstractCustomComponentProvider;
import com.fr.base.TemplateUtils;
import com.fr.general.IOUtils;
import com.fr.intelli.record.Focus;
import com.fr.intelli.record.Original;
import com.fr.record.analyzer.EnableMetrics;
import com.fr.web.struct.AssembleComponent;

@EnableMetrics
public class MapHotComponentProvider extends AbstractCustomComponentProvider {
    /**
     *
     */
    @Override
    public String getName() {
        return PluginConstantsEK.PLUGIN_MAP_NAME;
    }

    /**
     * @return
     */
    @Override
    public String getType() {
        return PluginConstantsEK.PLUGIN_MAP_TYPE;
    }

    /**
     *
     *
     * @return icon
     */
    @Override
    public String getIcon() {
        try {
            String render = TemplateUtils.render("${fineServletURL}");
            return render + "/resources?path=/com/finebi/plugin/tptj/ivan/chart/demo/amap/icon.png";
        } catch (Exception ignore) {
        }
        return "";
    }

    /**
     * @param context
     * @return
     */
    @Focus(id = PluginConstantsEK.PLUGIN_ID, text = PluginConstantsEK.PLUGIN_NAME, source = Original.PLUGIN)
    @Override
    public String getPreviewPageHTML(OperationContext context) {
        return "<div id=\"amap-demo-container\"></div>";
    }

    @Override
    public AssembleComponent previewClient(OperationContext context) {
        return MapHotComponent.KEY;
    }

    /**
     * JSON
     *
     * @return
     */
    @Override
    public String config() {
        return IOUtils.readResourceAsString("com/finebi/plugin/tptj/ivan/chart/demo/amap/config/config.json");
    }
}
```

plugin.xml

plugin.xml

```
<extra-core>
    <CustomComponentProvider
            class="com.finebi.plugin.tptj.ivan.chart.demo.amap.MapHotComponentProvider"/>
</extra-core>
```

AbstractCustomComponentProviderconfigjson

config.json

```
{
  "dataRegions": [
    {
      "name": "lat",
      "text": "Plugin-DEMO_WEB_LAT"
    },
    {
      "name": "lng",
      "text": "Plugin-DEMO_WEB_LNG"
    }
  ],
  "attrRegions": [
    {
      "name": "",
      "text": "Plugin-DEMO_WEB_FG",
      "multiFields": false,
      "settings": []
    }
  ],
  "chartStyles": [
    {
      "name": "mapProp",
      "text": "Plugin-DEMO_WEB_MAP_ATTRIBUTE",
      "multiFields": true,
      "settings": [
        {
          "name": "centerLng",
          "text": "Plugin-DEMO_WEB_CENTER_LNG",
          "type": "Input",
```

```
          "defaultValue": "102.3716"
        },
        {
          "name": "centerLat",
          "text": "Plugin-DEMO_WEB_CENTER_LAT",
          "type": "Input",
          "defaultValue": "36.6808"
        },
        {
          "name": "defaultZoom",
          "text": "Plugin-DEMO_WEB_DEFAULT_ZOOM",
          "type": "Input",
          "defaultValue": "4"
        },
        {
          "name": "style",
          "text": "Plugin-DEMO_WEB_STYLE",
          "type": "Select",
          "defaultValue": "normal",
          "items": [
            {
              "text": "",
              "value": "normal"
            },
            {
              "text": "",
              "value": "dark"
            },
            {
              "text": "",
              "value": "light"
            },
            {
              "text": "",
              "value": "whitesmoke"
            },
            {
              "text": "",
              "value": "fresh"
            },
            {
              "text": "",
              "value": "grey"
            },
            {
              "text": "",
              "value": "graffiti"
            },
            {
              "text": "",
              "value": "macaron"
            },
            {
              "text": "",
              "value": "blue"
            },
            {
              "text": "",
              "value": "darkblue"
            },
            {
              "text": "",
              "value": "wine"
            }
          ]
        }
      ]
    }
  ]
}
```

JSON

- dataRegions
  - JSONARRAY
  -
  -
- attrRegions
  - JSONARRAY
  -
  - settings
    - JSONARRAYcolorsizesymbolCheckboxRadioGroupSegmentSelectColorPickerInput
- chartStyles
  - JSONARRAY
  -
  - type
    - CheckboxRadioGroupSegmentSelectColorPickerInput

nameidjsnametextkey

config.json

```
{
    "dataRegions": [
        {
            "name": "1",
                        "text": "key"
```

```
        },
        {
            "name": "2"
        }
    ],
    "attrRegions": [
        {
            "name": "1",
            "multiFields": false,
            "settings": "color"
        },
        {
            "name": "1",
            "multiFields": true,
            "settings": "size"
        },
        {
            "name": "1",
            "multiFields": false,
            "settings": "symbol"
        },
        {
            "name": "",
            "multiFields": false,
            "settings": [
                {
                    "name": "Checkbox",
                    "type": "Checkbox",
                    "defaultValue": ["2", "3"],
                    "items": [
                        {
                            "text": "1",
                            "value": "1"
                        },
                        {
                            "text": "2",
                            "value": "2"
                        },
                        {
                            "text": "2",
                            "value": "3"
                        }
                    ]
                },
                {
                    "name": "RadioGroup",
                    "type": "RadioGroup",
                    "defaultValue": "2",
                    "items": [
                        {
                            "text": "1",
                            "value": "1"
                        },
                        {
                            "text": "2",
                            "value": "2"
                        }
                    ]
                },
                {
                    "name": "Segment",
                    "type": "Segment",
                    "defaultValue": "2",
                    "items": [
                        {
                            "text": "1",
                            "value": "1"
                        },
                        {
                            "text": "2",
                            "value": "2"
```

```
                    }
                ]
            },
            {
                "name": "Select",
                "type": "Select",
                "defaultValue": "1",
                "items": [
                    {
                        "text": "1",
                        "value": "1"
                    },
                    {
                        "text": "2",
                        "value": "2"
                    }
                ]
            },
            {
                "name": "ColorPicker",
                "type": "ColorPicker",
                "defaultValue": "#ffffff"
            },
            {
                "name": "Input",
                "type": "Input",
                "defaultValue": "test"
            }
        ]
    },
    {
        "name": "111",
        "multiFields": true
    },
    {
        "name": "2",
        "multiFields": false
    }
],
"chartStyles": [
    {
        "name": "",
        "settings": [
            {
                "name": "Checkbox",
                "type": "Checkbox",
                "defaultValue": ["2", "3"],
                "items": [
                    {
                        "text": "1",
                        "value": "1"
                    },
                    {
                        "text": "2",
                        "value": "2"
                    },
                    {
                        "text": "2",
                        "value": "3"
                    }
                ]
            },
            {
                "name": "RadioGroup",
                "type": "RadioGroup",
                "defaultValue": "2",
                "items": [
                    {
                        "text": "1",
                        "value": "1"
                    },
```

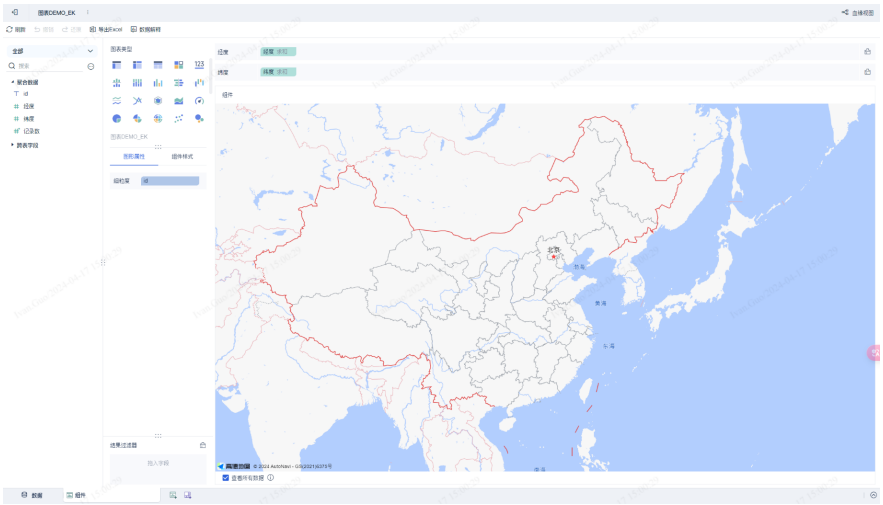
```
                        {
                            "text": "2",
                            "value": "2"
                        }
                    ]
                },
                {
                    "name": "Segment",
                    "type": "Segment",
                    "defaultValue": "2",
                    "items": [
                        {
                            "text": "1",
                            "value": "1"
                        },
                        {
                            "text": "2",
                            "value": "2"
                        }
                    ]
                },
                {
                    "name": "Select",
                    "type": "Select",
                    "defaultValue": "1",
                    "items": [
                        {
                            "text": "1",
                            "value": "1"
                        },
                        {
                            "text": "2",
                            "value": "2"
                        }
                    ]
                },
                {
                    "name": "ColorPicker",
                    "type": "ColorPicker",
                    "defaultValue": "#ffffff"
                },
                {
                    "name": "Input",
                    "type": "Input",
                    "defaultValue": "test"
                }
            ]
        }
    ]
}
```

render

```
function render(
        data, //
        config, //
        saveSessionCallback,
        closeSessionCallBack,
        extensionCallBack
) {}

//
new BIPlugin().init(render);
```

demo

```
(function ($) {
  function render(
    data, //
    config, //
    saveSessionCallback,
    closeSessionCallBack,
    extensionCallBack
  ) {
    // dom AbstractCustomComponentProvider#getPreviewPageHTML
    const dom = document.getElementById("amap-demo-container");

    //
    dom.style.width = document.body.clientWidth + "px";
    dom.style.height = document.body.clientHeight + "px";

    //
    const mapAttribute = config["chartStyle"][""]["value"];

    //
    const map = new AMap.Map(dom, {
      resizeEnable: true,
      center: [mapAttribute[0], mapAttribute[1]],
      zoom: mapAttribute[2],
    });

    //
    const styleName = "amap://styles/" + mapAttribute[3];
    map.setMapStyle(styleName);

    window.addEventListener("resize", function () {
      dom.style.width = document.body.clientWidth + "px";
      dom.style.height = document.body.clientHeight + "px";
    });
  }

  //
  new BIPlugin().init(render);
})(jQuery);
```



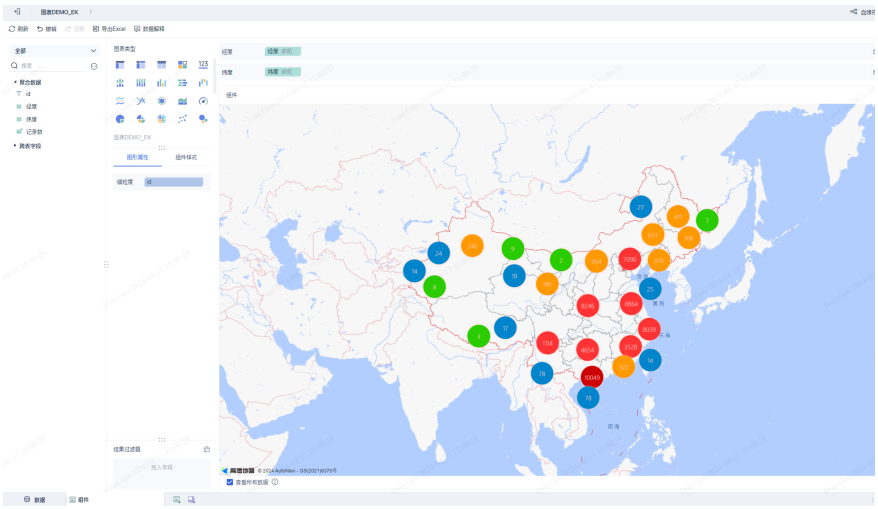
jsdataconfig

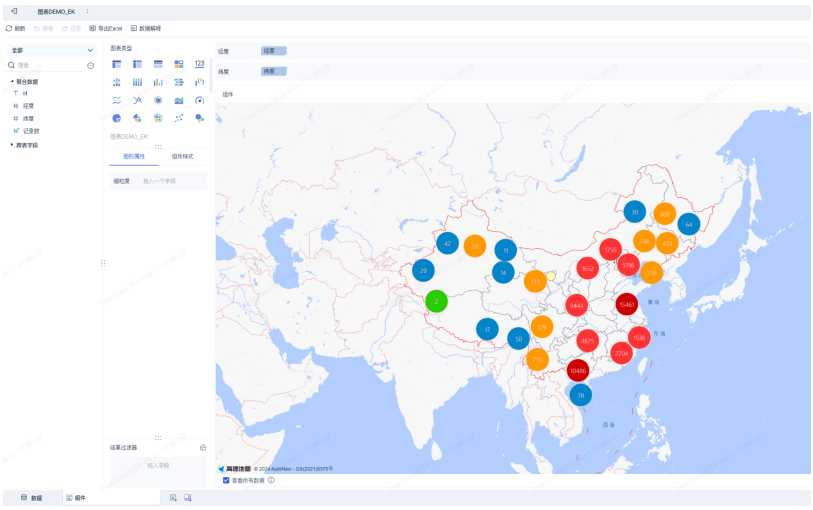
demo

```
(function ($) {
    function render(
        data, //
        config, //
        saveSessionCallback,
        closeSessionCallBack,
        extensionCallBack
    ) {
        debugger;

        // dom
        const dom = document.getElementById("amap-demo-container");

        //
        dom.style.width = document.body.clientWidth + "px";
        dom.style.height = document.body.clientHeight + "px";

        //
        const mapAttribute = config["chartStyle"][""]["value"];

        //
        const map = new AMap.Map(dom, {
            resizeEnable: true,
            center: [mapAttribute[0], mapAttribute[1]],
            zoom: mapAttribute[2],
        });

        //
        const styleName = "amap://styles/" + mapAttribute[3];
        map.setMapStyle(styleName);

        //
        const points = _getAllPoint(data.dataMapping, data.dataModels[0]);

        if (points != null) {
            //
            var pointCluster = new AMap.MarkerCluster(map, points, {
                gridSize: 60, //
                renderClusterMarker: _renderCarClusterMarker, //
                renderMarker: _renderMarker,   //
            });
        }
```

```
        window.addEventListener("resize", function () {
            dom.style.width = document.body.clientWidth + "px";
            dom.style.height = document.body.clientHeight + "px";
        });
    }

    /**
     *
     * @param context
     * @private
     */
    function _renderCarClusterMarker(context) {
        let count = context.count;
        let factor = context.count / count;
        const div = document.createElement('div');
        const Hue = 180 - factor * 180;
        let bgColor;
        if (context.count < 10) {
            bgColor = 'hsla(108,100%,40%,1)';
        } else if (context.count < 100) {
            bgColor = 'hsl(201,100%,40%)';
        } else if (context.count < 1000) {
            bgColor = 'hsla(36,100%,50%,1)';
        } else if (context.count < 10000) {
            bgColor = 'hsla(0,100%,60%,1)';
        } else {
            bgColor = 'hsla(0,100%,40%,1)';
        }
        const fontColor = 'hsla(' + Hue + ',100%,90%,1)';
        const borderColor = bgColor;
        const shadowColor = 'hsla(' + Hue + ',100%,90%,1)';
        div.style.backgroundColor = bgColor;
        const size = Math.round(30 + Math.pow(context.count / count, 1 / 5) * 20);
        div.style.width = div.style.height = size + 'px';
        div.style.border = 'solid 1px ' + borderColor;
        div.style.borderRadius = size / 2 + 'px';
        div.style.boxShadow = '0 0 5px ' + shadowColor;
        div.innerHTML = context.count;
        div.style.lineHeight = size + 'px';
        div.style.color = fontColor;
        div.style.fontSize = '14px';
        div.style.textAlign = 'center';
        context.marker.setOffset(new AMap.Pixel(-size / 2, -size / 2));
        context.marker.setContent(div)
    }

    /**
     *
     * @param context
     * @private
     */
    function _renderMarker(context) {
        var content = '<div style="background-color: rgba(255,255,178,.9); height: 18px; width: 18px; border:
1px solid rgba(255,255,178,1); border-radius: 12px; box-shadow: rgba(0, 0, 0, 1) 0px 0px 3px;"></div>';
        var offset = new AMap.Pixel(-9, -9);
        context.marker.setContent(content)
        context.marker.setOffset(offset)
    }

    /**
     *
     * @param dataMapping
     * @param dataModel
     * @returns {*[]}
     * @private
     */
    function _getAllPoint(dataMapping, dataModel) {
        let lngId = dataMapping[''];
        let latId = dataMapping[''];
```

```
        let files = dataModel.fields;
        let rowCount = dataModel.rowCount
        const colData = dataModel.colData;

        let lngIndex;
        let latIndex;

        // demo
        files.forEach((item, index) => {
            if (lngId.indexOf(item.id) >= 0) {
                lngIndex = index;
            }
            if (latId.indexOf(item.id) >= 0) {
                latIndex = index;
            }
        })

        var points = [];
        for (let i = 0; i < rowCount; i++) {
            points.push({
                lnglat: [colData[lngIndex][i], colData[latIndex][i]],
            })
        }

        return points;
    }

    //
    new BIPlugin().init(render);
})(jQuery);
```





DEMO

DEMO

dataModels

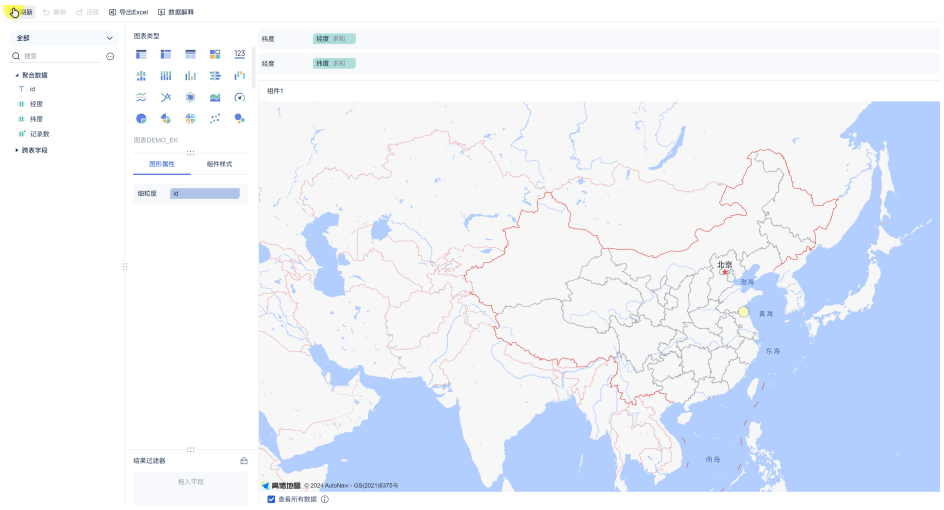
CustomComponentProviderneedDataProcessprocess

```
/**
     *
     * true
     * false
     * CustomComponentContext:
     */
@Override
public boolean needDataProcess(CustomComponentContext customComponentContext) {
    return true;
}

/**
     *
     * needDataProcesstrue
     * CustomComponentContext:
     */
@Override
public List<DataModel> process(List<DataModel> dataModels, CustomComponentContext customComponentContext) {
    return dataModels;
}
```

DEMO

DEMO

```
/**
     *
     * true
     * false
     * CustomComponentContext:
     */
@Override
public boolean needDataProcess(CustomComponentContext customComponentContext) {
    return true;
}

/**
     *
     * needDataProcesstrue
     * CustomComponentContext:
     */
@Override
public List<DataModel> process(List<DataModel> dataModels, CustomComponentContext customComponentContext) {
    return dataModels.stream().map(dataModel -> new DataModel() {
        @Override
        public List<Dimension> getFields() {
            return dataModel.getFields();
        }

        @Override
        public List<List<Object>> getColData() {
            List<List<Object>> colData = new ArrayList<>(dataModel.getFields().size());
            dataModel.getColData().forEach(d -> colData.add(Collections.singletonList(d.get((int) (Math.random()
* d.size())))));
            return colData;
        }
    }).collect(Collectors.toList());
}
```



DEMO

DEMO

JSextensionCallBack('refresh')extensionCallBackrender

```
function render(data, config, saveSessionCallback, closeSessionCallBack, extensionCallBack) {
    function userClick(){
        // iframe
        extensionCallBack('refresh');
    }
}

//
new BIPlugin().init(render);
```

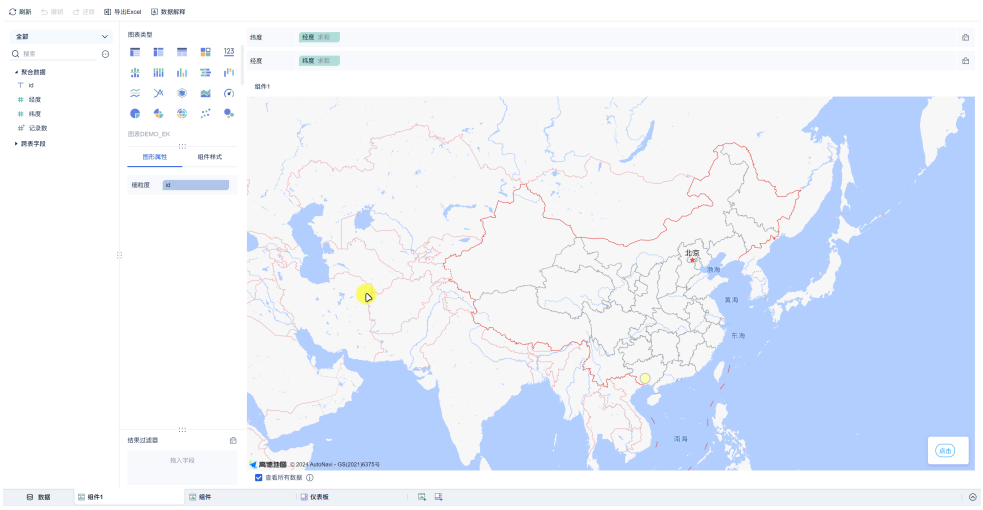
DEMOiframe

demo

```
(function ($) {
    function render(
        data, //
        config, //
        saveSessionCallback,
        closeSessionCallBack,
        extensionCallBack
    ) {
        debugger;

        // dom
        const dom = document.getElementById("amap-demo-container");

        //
        dom.style.width = document.body.clientWidth + "px";
        dom.style.height = document.body.clientHeight + "px";

        //
        const mapAttribute = config["chartStyle"]["mapProp"]["value"];

        //
        const map = new AMap.Map(dom, {
            resizeEnable: true,
            center: [mapAttribute[0], mapAttribute[1]],
            zoom: mapAttribute[2],
        });

        //
        const styleName = "amap://styles/" + mapAttribute[3];
        map.setMapStyle(styleName);

        //
        const points = _getAllPoint(data.dataMapping, data.dataModels[0]);

        if (points != null) {
            //
            var pointCluster = new AMap.MarkerCluster(map, points, {
                gridSize: 60, //
                renderClusterMarker: _renderCarClusterMarker, //
                renderMarker: _renderMarker,   //
            });
        }


        window.addEventListener("resize", function () {
            dom.style.width = document.body.clientWidth + "px";
```

```
            dom.style.height = document.body.clientHeight + "px";
        });

        document.querySelector("#amap-demo-click").onclick = function () {
            extensionCallBack('refresh');
        }
    }

    /**
     *
     * @param context
     * @private
     */
    function _renderCarClusterMarker(context) {
        let count = context.count;
        let factor = context.count / count;
        const div = document.createElement('div');
        const Hue = 180 - factor * 180;
        let bgColor;
        if (context.count < 10) {
            bgColor = 'hsla(108,100%,40%,1)';
        } else if (context.count < 100) {
            bgColor = 'hsl(201,100%,40%)';
        } else if (context.count < 1000) {
            bgColor = 'hsla(36,100%,50%,1)';
        } else if (context.count < 10000) {
            bgColor = 'hsla(0,100%,60%,1)';
        } else {
            bgColor = 'hsla(0,100%,40%,1)';
        }
        const fontColor = 'hsla(' + Hue + ',100%,90%,1)';
        const borderColor = bgColor;
        const shadowColor = 'hsla(' + Hue + ',100%,90%,1)';
        div.style.backgroundColor = bgColor;
        const size = Math.round(30 + Math.pow(context.count / count, 1 / 5) * 20);
        div.style.width = div.style.height = size + 'px';
        div.style.border = 'solid 1px ' + borderColor;
        div.style.borderRadius = size / 2 + 'px';
        div.style.boxShadow = '0 0 5px ' + shadowColor;
        div.innerHTML = context.count;
        div.style.lineHeight = size + 'px';
        div.style.color = fontColor;
        div.style.fontSize = '14px';
        div.style.textAlign = 'center';
        context.marker.setOffset(new AMap.Pixel(-size / 2, -size / 2));
        context.marker.setContent(div)
    }

    /**
     *
     * @param context
     * @private
     */
    function _renderMarker(context) {
        var content = '<div style="background-color: rgba(255,255,178,.9); height: 18px; width: 18px; border:
1px solid rgba(255,255,178,1); border-radius: 12px; box-shadow: rgba(0, 0, 0, 1) 0px 0px 3px;"></div>';
        var offset = new AMap.Pixel(-9, -9);
        context.marker.setContent(content)
        context.marker.setOffset(offset)
    }

    /**
     *
     * @param dataMapping
     * @param dataModel
     * @returns {*[]}
     * @private
     */
    function _getAllPoint(dataMapping, dataModel) {
        let lngId = dataMapping['lat'];
        let latId = dataMapping['lng'];
```

```
        let files = dataModel.fields;
        let rowCount = dataModel.rowCount
        const colData = dataModel.colData;

        let lngIndex;
        let latIndex;

        // demo
        files.forEach((item, index) => {
            if (lngId.indexOf(item.id) >= 0) {
                lngIndex = index;
            }
            if (latId.indexOf(item.id) >= 0) {
                latIndex = index;
            }
        })

        var points = [];
        for (let i = 0; i < rowCount; i++) {
            points.push({
                lnglat: [colData[lngIndex][i], colData[latIndex][i]],
            })
        }

        return points;
    }

    //
    new BIPlugin().init(render);
})(jQuery);
```



DEMO

DEMO

```
function render(data, config, saveSessionCallback, closeSessionCallBack, extensionCallBack) {
    function userClick(){
                // {xxx: yyy}config.customConfigconfig.customConfig
                 saveSessionCallback({xxx: yyy});
        // iframe
        extensionCallBack('refresh');
    }
}

//
new BIPlugin().init(render);
```

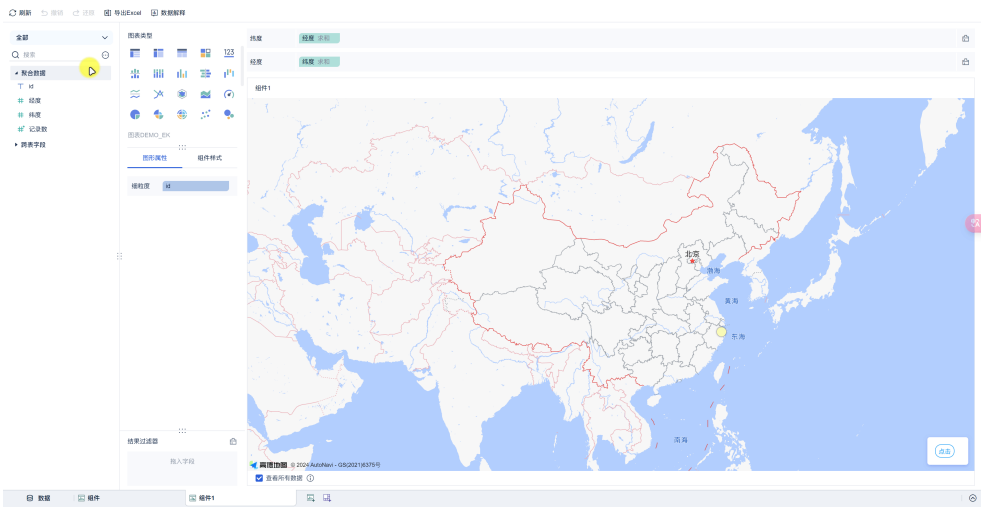
DEMOiframe

demo

```
(function ($) {
    function render(
        data, //
        config, //
        saveSessionCallback,
        closeSessionCallBack,
        extensionCallBack
    ) {
        debugger;

        // dom
        const dom = document.getElementById("amap-demo-container");

        //
        dom.style.width = document.body.clientWidth + "px";
        dom.style.height = document.body.clientHeight + "px";

        //
        const mapAttribute = config["chartStyle"]["mapProp"]["value"];

        //
        let map;
        let customConfig = config.customConfig;
        if (customConfig != null && JSON.stringify(customConfig).length > 2) {
            //
            map = new AMap.Map(dom, {
                resizeEnable: true,
                center: [customConfig.lng, customConfig.lat],
                zoom: customConfig.zoom,
            });
        } else {
            map = new AMap.Map(dom, {
                resizeEnable: true,
                center: [mapAttribute[0], mapAttribute[1]],
                zoom: mapAttribute[2],
            });
        }

        //
        const styleName = "amap://styles/" + mapAttribute[3];
        map.setMapStyle(styleName);

        //
        const points = _getAllPoint(data.dataMapping, data.dataModels[0]);

        if (points != null) {
            //
```

```
        var pointCluster = new AMap.MarkerCluster(map, points, {
            gridSize: 60, //
            renderClusterMarker: _renderCarClusterMarker, //
            renderMarker: _renderMarker,   //
        });
    }


    window.addEventListener("resize", function () {
        dom.style.width = document.body.clientWidth + "px";
        dom.style.height = document.body.clientHeight + "px";
    });

    document.querySelector("#amap-demo-click").onclick = function () {
        let conf = {
            zoom: map.getZoom(),
            lng: map.getCenter().lng,
            lat: map.getCenter().lat,
        };
        //
        saveSessionCallback(conf);
        extensionCallBack('refresh');
    }
}

/**
 *
 * @param context
 * @private
 */
function _renderCarClusterMarker(context) {
    let count = context.count;
    let factor = context.count / count;
    const div = document.createElement('div');
    const Hue = 180 - factor * 180;
    let bgColor;
    if (context.count < 10) {
        bgColor = 'hsla(108,100%,40%,1)';
    } else if (context.count < 100) {
        bgColor = 'hsl(201,100%,40%)';
    } else if (context.count < 1000) {
        bgColor = 'hsla(36,100%,50%,1)';
    } else if (context.count < 10000) {
        bgColor = 'hsla(0,100%,60%,1)';
    } else {
        bgColor = 'hsla(0,100%,40%,1)';
    }
    const fontColor = 'hsla(' + Hue + ',100%,90%,1)';
    const borderColor = bgColor;
    const shadowColor = 'hsla(' + Hue + ',100%,90%,1)';
    div.style.backgroundColor = bgColor;
    const size = Math.round(30 + Math.pow(context.count / count, 1 / 5) * 20);
    div.style.width = div.style.height = size + 'px';
    div.style.border = 'solid 1px ' + borderColor;
    div.style.borderRadius = size / 2 + 'px';
    div.style.boxShadow = '0 0 5px ' + shadowColor;
    div.innerHTML = context.count;
    div.style.lineHeight = size + 'px';
    div.style.color = fontColor;
    div.style.fontSize = '14px';
    div.style.textAlign = 'center';
    context.marker.setOffset(new AMap.Pixel(-size / 2, -size / 2));
    context.marker.setContent(div)
}

/**
 *
 * @param context
 * @private
 */
function _renderMarker(context) {
```

```
        var content = '<div style="background-color: rgba(255,255,178,.9); height: 18px; width: 18px; border:
1px solid rgba(255,255,178,1); border-radius: 12px; box-shadow: rgba(0, 0, 0, 1) 0px 0px 3px;"></div>';
        var offset = new AMap.Pixel(-9, -9);
        context.marker.setContent(content)
        context.marker.setOffset(offset)
    }

    /**
     *
     * @param dataMapping
     * @param dataModel
     * @returns {*[]}
     * @private
     */
    function _getAllPoint(dataMapping, dataModel) {
        let lngId = dataMapping['lat'];
        let latId = dataMapping['lng'];

        let files = dataModel.fields;
        let rowCount = dataModel.rowCount
        const colData = dataModel.colData;

        let lngIndex;
        let latIndex;

        // demo
        files.forEach((item, index) => {
            if (lngId.indexOf(item.id) >= 0) {
                lngIndex = index;
            }
            if (latId.indexOf(item.id) >= 0) {
                latIndex = index;
            }
        })

        var points = [];
        for (let i = 0; i < rowCount; i++) {
            points.push({
                lnglat: [colData[lngIndex][i], colData[latIndex][i]],
            })
        }

        return points;
    }

    //
    new BIPlugin().init(render);
})(jQuery);
```



DEMO

DEMO

~~click~~

```
const currentClicked = {
    dId: id,
    value: val
};
extensionCallBack(
    'click',
    currentClicked
);
```

~~DEMO~~

demo

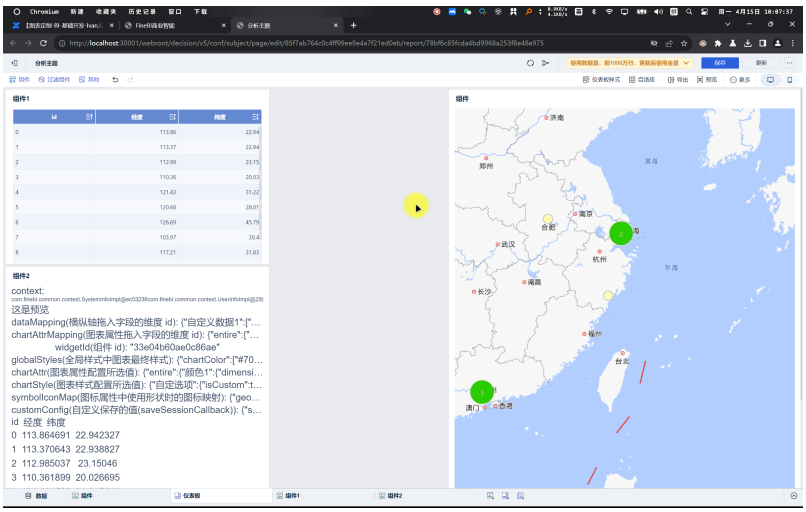
```
(function ($) {
    function render(
        data, //
        config, //
        saveSessionCallback,
        closeSessionCallBack,
        extensionCallBack
    ) {
        debugger;

        // dom
        const dom = document.getElementById("amap-demo-container");

        //
        dom.style.width = document.body.clientWidth + "px";
        dom.style.height = document.body.clientHeight + "px";

        const {
```

```
    // dataModels
    dataModels,
    //  id
    dataMapping,
    //  id
    chartAttrMapping,
    //
    clicked
} = data;

const {
    //  id
    widgetId,
    //
    globalStyles,
    //
    chartAttr,
    //
    chartStyle,
    //
    symbolIconMap,
    // (saveSessionCallback)
    customConfig
} = config;

//
const mapAttribute = chartStyle["mapProp"]["value"];

//
let map;
if (customConfig != null && JSON.stringify(customConfig).length > 2) {
    //
    map = new AMap.Map(dom, {
        resizeEnable: true,
        center: [customConfig.lng, customConfig.lat],
        zoom: customConfig.zoom,
    });
} else {
    map = new AMap.Map(dom, {
        resizeEnable: true,
        center: [mapAttribute[0], mapAttribute[1]],
        zoom: mapAttribute[2],
    });
}

//
const styleName = "amap://styles/" + mapAttribute[3];
map.setMapStyle(styleName);

//
const points = _getAllPoint(dataMapping, dataModels[0]);

if (points != null) {
    //
    var pointCluster = new AMap.MarkerCluster(map, points, {
        gridSize: 60, //
        renderClusterMarker: _renderCarClusterMarker, //
        renderMarker: _renderMarker,   //
    });
}


window.addEventListener("resize", function () {
    dom.style.width = document.body.clientWidth + "px";
    dom.style.height = document.body.clientHeight + "px";
});

document.querySelector("#amap-demo-click").onclick = function () {
    let conf = {
        zoom: map.getZoom(),
        lng: map.getCenter().lng,
```

```
                lat: map.getCenter().lat,
            };
            //
            saveSessionCallback(conf);
            extensionCallBack('refresh');
        }

        /**
         *
         * @param context
         * @private
         */
        function _renderCarClusterMarker(context) {
            let count = context.count;
            let factor = context.count / count;
            const div = document.createElement('div');
            const Hue = 180 - factor * 180;
            let bgColor;
            if (context.count < 10) {
                bgColor = 'hsla(108,100%,40%,1)';
            } else if (context.count < 100) {
                bgColor = 'hsl(201,100%,40%)';
            } else if (context.count < 1000) {
                bgColor = 'hsla(36,100%,50%,1)';
            } else if (context.count < 10000) {
                bgColor = 'hsla(0,100%,60%,1)';
            } else {
                bgColor = 'hsla(0,100%,40%,1)';
            }
            const fontColor = 'hsla(' + Hue + ',100%,90%,1)';
            const borderColor = bgColor;
            const shadowColor = 'hsla(' + Hue + ',100%,90%,1)';
            div.style.backgroundColor = bgColor;
            const size = Math.round(30 + Math.pow(context.count / count, 1 / 5) * 20);
            div.style.width = div.style.height = size + 'px';
            div.style.border = 'solid 1px ' + borderColor;
            div.style.borderRadius = size / 2 + 'px';
            div.style.boxShadow = '0 0 5px ' + shadowColor;
            div.innerHTML = context.count;
            div.style.lineHeight = size + 'px';
            div.style.color = fontColor;
            div.style.fontSize = '14px';
            div.style.textAlign = 'center';
            context.marker.setOffset(new AMap.Pixel(-size / 2, -size / 2));
            context.marker.setContent(div)
        }

        /**
         *
         * @param context
         * @private
         */
        function _renderMarker(context) {
            var content = '<div style="background-color: rgba(255,255,178,.9); height: 18px; width: 18px;
border: 1px solid rgba(255,255,178,1); border-radius: 12px; box-shadow: rgba(0, 0, 0, 1) 0px 0px 3px;"></div>';
            var offset = new AMap.Pixel(-9, -9);
            context.marker.setContent(content)
            context.marker.setOffset(offset)
            let datum = context.data[0];
            let id = datum.idId;
            const val = [
                {
                    dId: datum.idId,
                    text: datum.idVal,
                },
                // {
                //     dId: datum.lngId,
                //     text: datum.lngVal,
                // },
                // {
                //     dId: datum.latId,
```

```
            //     text: datum.latVal,
            // }
        ];

        /**
         * dId: Id
         * value: id  (
         */
        const currentClicked = {
            dId: id,
            value: val
        };

        context.marker.on('click', ev => {
            debugger;
            //
            extensionCallBack(
                'click',
                currentClicked
            );
        })
    }

}

/**
 *
 * @param dataMapping
 * @param dataModel
 * @returns {*[]}
 * @private
 */
function _getAllPoint(dataMapping, dataModel) {
    let lngId = dataMapping['lng'];
    let latId = dataMapping['lat'];

    let files = dataModel.fields;
    let rowCount = dataModel.rowCount
    const colData = dataModel.colData;

    let lngIndex;
    let latIndex;
    let idIndex;

    // demo
    files.forEach((item, index) => {
        if (lngId.indexOf(item.id) >= 0) {
            lngIndex = index;
        } else if (latId.indexOf(item.id) >= 0) {
            latIndex = index;
        } else {
            idIndex = index;
        }
    })

    var points = [];
    for (let i = 0; i < rowCount; i++) {
        points.push({
            lnglat: [colData[lngIndex][i], colData[latIndex][i]],
            lngId: files[lngIndex].id,
            latId: files[latIndex].id,
            idId: files[idIndex].id,
            lngVal: colData[lngIndex][i],
            latVal: colData[latIndex][i],
            idVal: colData[idIndex][i],
        })
    }

    return points;
}
```

```
    //
    new BIPlugin().init(render);
})(jQuery);
```



~~DEMO~~

~~DEMO~~

dimensionSelected

```
const currentClicked = {
    id1: value1,
    id2: value2,
    id3: value3
};

//
extensionCallBack("pointSelected", {
    pos: {
        x: x,
        y: y,
    },
    // id
    measure: measureId,
    //
    // id:
    row: currentClicked,
});

//
extensionCallBack("dimensionSelected", {
    pos: {
        x: x,
        y: y,
    },
    // id
    measure: measureId,
    //
    // id:
    row: currentClicked,
});
```

DEMO

demo

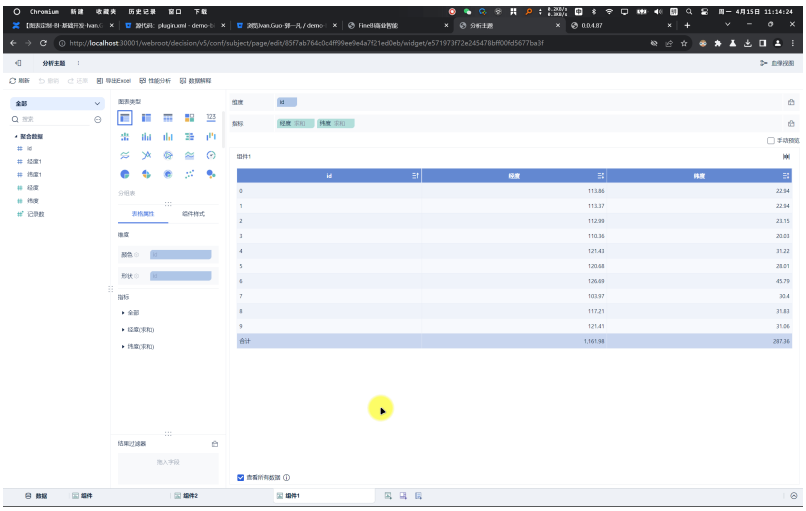
```
(function ($) {
    function render(
        data, //
        config, //
        saveSessionCallback,
        closeSessionCallBack,
        extensionCallBack
    ) {
        debugger;

        // dom
        const dom = document.getElementById("amap-demo-container");

        //
        dom.style.width = document.body.clientWidth + "px";
        dom.style.height = document.body.clientHeight + "px";

        const {
            // dataModels
            dataModels,
            //  id
            dataMapping,
            //  id
            chartAttrMapping,
            //
            clicked
```

```
} = data;

const {
    //  id
    widgetId,
    //
    globalStyles,
    //
    chartAttr,
    //
    chartStyle,
    //
    symbolIconMap,
    // (saveSessionCallback)
    customConfig
} = config;

//
const mapAttribute = chartStyle["mapProp"]["value"];

//
let map;
if (customConfig != null && JSON.stringify(customConfig).length > 2) {
    //
    map = new AMap.Map(dom, {
        resizeEnable: true,
        center: [customConfig.lng, customConfig.lat],
        zoom: customConfig.zoom,
    });
} else {
    map = new AMap.Map(dom, {
        resizeEnable: true,
        center: [mapAttribute[0], mapAttribute[1]],
        zoom: mapAttribute[2],
    });
}

//
const styleName = "amap://styles/" + mapAttribute[3];
map.setMapStyle(styleName);

//
const points = _getAllPoint(dataMapping, dataModels[0]);

if (points != null) {
    //
    var pointCluster = new AMap.MarkerCluster(map, points, {
        gridSize: 60, //
        renderClusterMarker: _renderCarClusterMarker, //
        renderMarker: _renderMarker,   //
    });
}


window.addEventListener("resize", function () {
    dom.style.width = document.body.clientWidth + "px";
    dom.style.height = document.body.clientHeight + "px";
});

document.querySelector("#amap-demo-click").onclick = function () {
    let conf = {
        zoom: map.getZoom(),
        lng: map.getCenter().lng,
        lat: map.getCenter().lat,
    };
    //
    saveSessionCallback(conf);
    extensionCallBack('refresh');
}

/**
```

```
         *
         * @param context
         * @private
         */
        function _renderCarClusterMarker(context) {
            let count = context.count;
            let factor = context.count / count;
            const div = document.createElement('div');
            const Hue = 180 - factor * 180;
            let bgColor;
            if (context.count < 10) {
                bgColor = 'hsla(108,100%,40%,1)';
            } else if (context.count < 100) {
                bgColor = 'hsl(201,100%,40%)';
            } else if (context.count < 1000) {
                bgColor = 'hsla(36,100%,50%,1)';
            } else if (context.count < 10000) {
                bgColor = 'hsla(0,100%,60%,1)';
            } else {
                bgColor = 'hsla(0,100%,40%,1)';
            }
            const fontColor = 'hsla(' + Hue + ',100%,90%,1)';
            const borderColor = bgColor;
            const shadowColor = 'hsla(' + Hue + ',100%,90%,1)';
            div.style.backgroundColor = bgColor;
            const size = Math.round(30 + Math.pow(context.count / count, 1 / 5) * 20);
            div.style.width = div.style.height = size + 'px';
            div.style.border = 'solid 1px ' + borderColor;
            div.style.borderRadius = size / 2 + 'px';
            div.style.boxShadow = '0 0 5px ' + shadowColor;
            div.innerHTML = context.count;
            div.style.lineHeight = size + 'px';
            div.style.color = fontColor;
            div.style.fontSize = '14px';
            div.style.textAlign = 'center';
            context.marker.setOffset(new AMap.Pixel(-size / 2, -size / 2));
            context.marker.setContent(div)
        }

        /**
         *
         * @param context
         * @private
         */
        function _renderMarker(context) {
            var content = '<div style="background-color: rgba(255,255,178,.9); height: 18px; width: 18px;
border: 1px solid rgba(255,255,178,1); border-radius: 12px; box-shadow: rgba(0, 0, 0, 1) 0px 0px 3px;"></div>';
            var offset = new AMap.Pixel(-9, -9);
            context.marker.setContent(content)
            context.marker.setOffset(offset)
            let datum = context.data[0];
            // let id = datum.idId;
            // const val = [
            //     {
            //         dId: datum.idId,
            //         text: datum.idVal,
            //     },
            //     {
            //         dId: datum.lngId,
            //         text: datum.lngVal,
            //     },
            //     {
            //         dId: datum.latId,
            //         text: datum.latVal,
            //     }
            // ];

            // /**
            //  * dId: Id
            //  * value: id  (
            //  */
```

```
        // const currentClicked = {
        //     dId: id,
        //     value: val
        // };

        // context.marker.on('click', ev => {
        //     debugger;
        //     //
        //     extensionCallBack(
        //         'click',
        //         currentClicked
        //     );
        // });

        const currentClicked = {
            [datum.idId]: datum.idVal,
            [datum.lngId]: datum.lngVal,
            [datum.latId]: datum.latVal
        };

        const currentId = datum.lngId
        context.marker.on('click', ev => {
            debugger;
            //
            const demo = {
                pos: {
                    x: window.event.pageX,
                    y: window.event.pageY,
                },
                // id
                measure: currentId,
                //
                // id:
                row: currentClicked,
            };
            //
            // extensionCallBack("pointSelected", demo);
            //
            // extensionCallBack("dimensionSelected", demo);
            for (let i = 0; i < data.dataModels[0].fields.length; i++) {
                let field = data.dataModels[0].fields[i];
                if (field.id === currentId) {
                    //
                    if (field.isDimension) {
                        extensionCallBack("dimensionSelected", demo);
                    } else if (field.isMeasure) {
                        extensionCallBack("pointSelected", demo);
                    }
                    break;
                }
            }
        });

    }

}

/**
 *
 * @param dataMapping
 * @param dataModel
 * @returns {*[]}
 * @private
 */
function _getAllPoint(dataMapping, dataModel) {
    let lngId = dataMapping['lng'];
    let latId = dataMapping['lat'];

    let files = dataModel.fields;
    let rowCount = dataModel.rowCount
    const colData = dataModel.colData;
```

```
        let lngIndex;
        let latIndex;
        let idIndex;

        // demo
        files.forEach((item, index) => {
            if (lngId.indexOf(item.id) >= 0) {
                lngIndex = index;
            } else if (latId.indexOf(item.id) >= 0) {
                latIndex = index;
            } else {
                idIndex = index;
            }
        })

        var points = [];
        for (let i = 0; i < rowCount; i++) {
            points.push({
                lnglat: [colData[lngIndex][i], colData[latIndex][i]],
                lngId: files[lngIndex].id,
                latId: files[latIndex].id,
                idId: files[idIndex].id,
                lngVal: colData[lngIndex][i],
                latVal: colData[latIndex][i],
                idVal: colData[idIndex][i],
            })
        }

        return points;
    }

    //
    new BIPlugin().init(render);
})(jQuery);
```



DEMO

DEMO

value setValue

```
//
<Widget
    type={type}
    value={currentValue}
    setValue={(v: any) => {
        this.store.updateConfig(v, boxName, index);
    }}
/>;

//  demo
// json
{
    "name": "",
    "type": "bi.test.widget",
    "defaultValue": "1"
}
//  subjectPage
// bi.test.widget
render() {
    const { value, setValue } = this.options;

    return (
        <TextValueCombo
            simple
            value={value}
            height={24}
            items={[
                {
                    text: '1',
                    value: '1',
                },
                {
                    text: '2',
                    value: '2',
                },
            ]}
            listeners={[
                {
                    eventName: TextValueCombo.EVENT_CHANGE,
                    action: v => {
                        setValue(v);
                    },
                },
            ]}
        />
    );
}
```

subjecttype

.js

```
!(function () {
    /**
     * subject
     */
    var Demo = BI.inherit(BI.Widget, {

        props: {
            baseCls: ""
        },

        render: function () {
            var self = this;
            // self.optionsvaluesetValue
            debugger;
            return {
                type: "bi.vertical",
                width: 300,
                items: [
                    {
                        el: {
                            type: "bi.vertical_adapt",
                            items: [
                                {
                                    el: {
                                        type: "bi.label",
                                        text: "",
                                        textAlign: "left",
                                    },
                                    width: 70,
                                    rgap: 10,
                                },
                                {
                                    el: {
                                        type: "bi.editor",
                                        value: self.options.value, // value
                                        cls: "bi-border bi-border-radius",
                                        width: 150,
                                        allowClear: !1,
                                        height: 24,
                                        ref: function (e) {
                                            self.myNameEditor = e;
                                        },
                                        listeners: [
                                            {
                                                eventName: "EVENT_CHANGE",
                                                action: function () {
                                                    debugger;
                                                    // value
                                                    self.options.setValue(this.getValue());
                                                },
                                            },
                                        ],
                                    },
                                },
                            ],
                        },
                        tgap: 10,
                    }]
            };
        },

    });
    BI.shortcut("bi.amap.demo", Demo);
})();
```

json

"attrRegions": [
"name": "细粒度",
"text": "Plugin-DEMO_WEB_FG",
"multiFields": false,
"settings": []
Ivan.Guo-郭一凡, 4 minutes ago · 0.0.7: 图表DEMO_EK 自定
"name": "test",
"text": "测试",
"multiFields": false,
"settings": [
"name": "test1",
"text": "测试1",
"type": "bi.amap.demo",
"defaultValue": {}
"chartStyles": [

测试
拖入一个字段
测试1
名称
1

DEMO

DEMO