# DataV数据可视化

## 快速入门

为了无法计算的价值 | 阿里云

# 快速入门

**快速上手案例：用 DataV 查看春节前后空气质量的全国分布变化**

## 背景信息

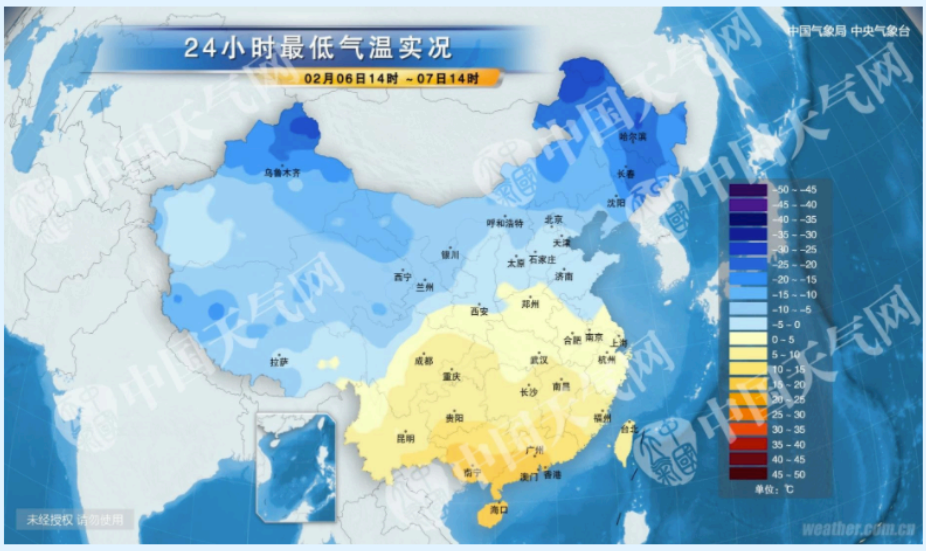
制作大屏时，您可能需要用到以下几种功能：

- 空间插值
- 等值面组件
- 时间轴组件

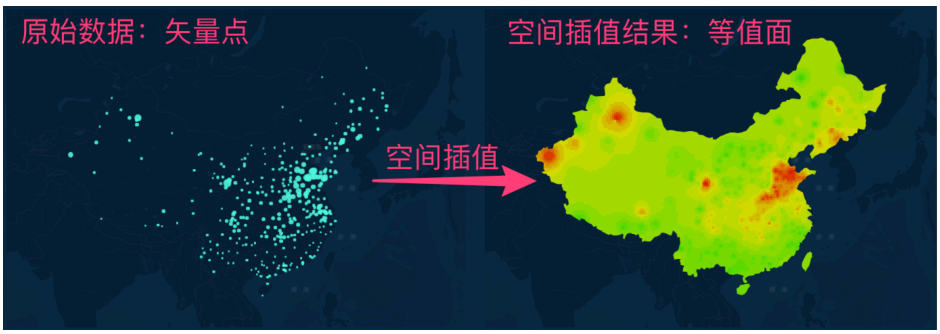
### 空间插值

空间插值常用于将离散点的测量数据转换为连续的数据曲面，以便与其它空间现象的分布模式进行比较。

也就是说根据已知的监测站点监测出的数据去推算其他任意空间位置的数据，再根据数值处在的不同区间范围去映射对应的颜色就可以得到一张典型的等温面图。

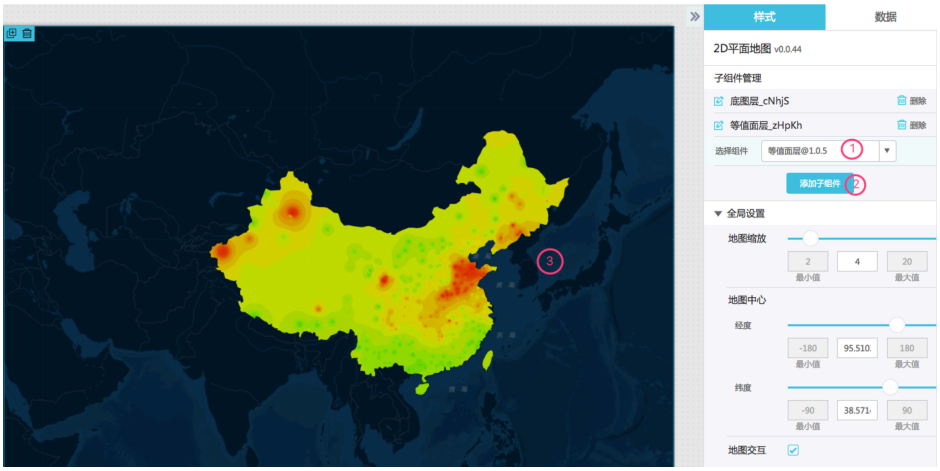



如果用 DataV 来制作一张等温面图，我们就可以很清楚的看到空间插值就是根据离散的已知点去插值出连续的面数据。



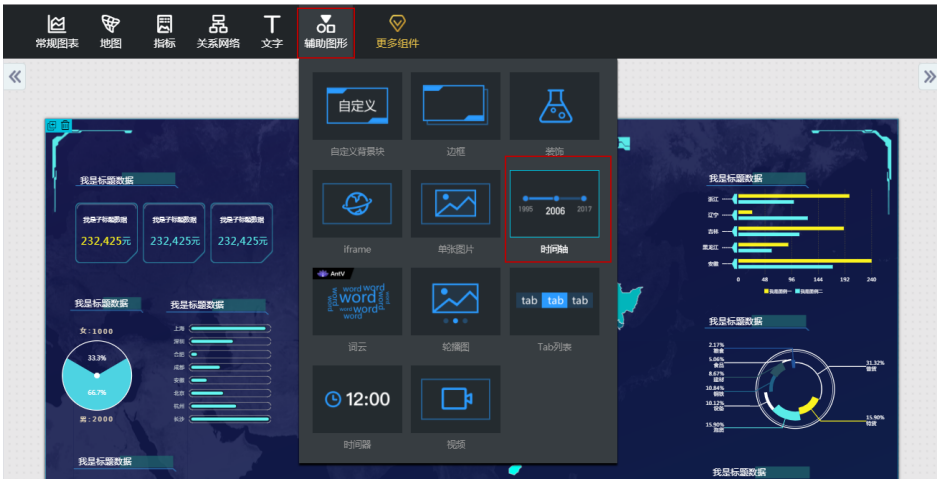

## 等值面组件

DataV 提供了一个轻分析的等值面地图组件来帮助您将已知的矢量点数据制作成栅格区域图。我们可以利用这个功能，来实时插值出全国的空气质量图。




## 时间轴组件

想要看到一段时间内的空气质量变化，时间轴组件自然必不可少。




时间轴控件有个重要特性就是支持回调 ID，利用这个回调 ID，我们就可以跟其他组件进行联动。当时间轴的时间发生变化时，其他组件的数据也会自动更新。

当填写了正确的回调 ID 后，系统会在每次时间变化的时候重新触发一次数据请求，并自动在其它组件所对应的

API 接口的参数列表中加上当前的回调 ID，以及其对应的值。

初始接口地址：http://127.0.0.1:8888/aqi

回调触发后：http://127.0.0.1:8888/aqi?date=2017012722

这里回调 ID 填写的是 date,2017012722。

同样回调 ID 也可以对 SQL 语句生效，不过您需要使用:加上回调 ID 名称作为占位。

初始 SQL： select :date as value;

回调触发后：select '2017022722' as value;

# 准备工作

在制作大屏之前，您需要先准备和处理好相关的数据和接口。

**本示例中将 csv 文件处理到了 json。**

## 获取数据

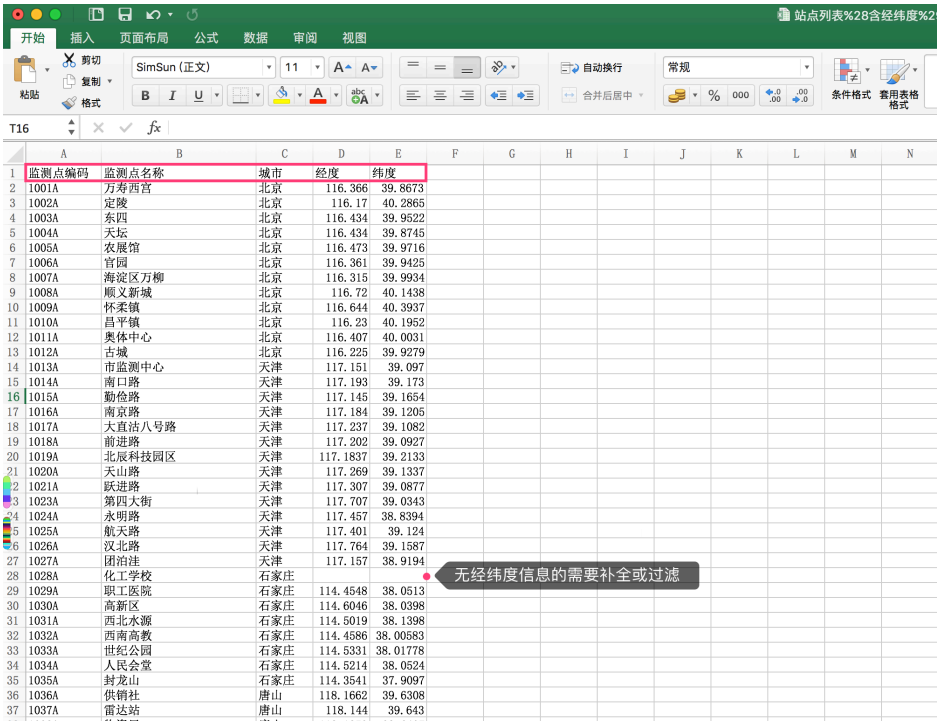
数据是可视化的原材料，我们首先要获取春节期间全国的空气质量数据。

您可以从**全国空气质量历史数据**上面下载需要的数据。

> **注意**：推荐下载格式为.csv 的文件。

本示例中，采用了2017年1月1日至2017年2月2日的全国1497个监测点的数据。

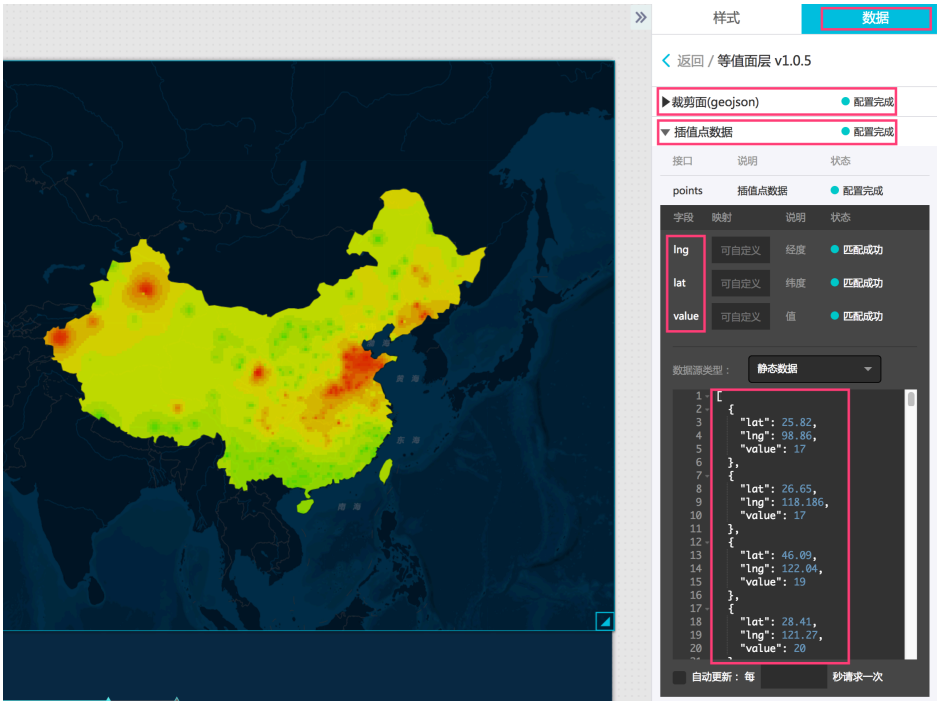
下载完成后，打开文件，查看是否有需要补全或者需要过滤出去的数据。




| 监测点编码 | 监测点名称 | 城市 | 经度 | 纬度 |
|---|---|---|---|---|
| 1001A | 万寿西宫 | 北京 | 116.366 | 39.8673 |
| 1002A | 定陵 | 北京 | 116.17 | 40.2865 |
| 1003A | 东四 | 北京 | 116.434 | 39.9522 |
| 1004A | 天坛 | 北京 | 116.434 | 39.8745 |
| 1005A | 农展馆 | 北京 | 116.473 | 39.9716 |
| 1006A | 官园 | 北京 | 116.361 | 39.9425 |
| 1007A | 海淀区万柳 | 北京 | 116.315 | 39.9934 |
| 1008A | 顺义新城 | 北京 | 116.72 | 40.1438 |
| 1009A | 怀柔镇 | 北京 | 116.644 | 40.3937 |
| 1010A | 昌平镇 | 北京 | 116.23 | 40.1952 |
| 1011A | 奥体中心 | 北京 | 116.407 | 40.0031 |
| 1012A | 古城 | 北京 | 116.225 | 39.9279 |
| 1013A | 市监测中心 | 天津 | 117.151 | 39.097 |
| 1014A | 南口路 | 天津 | 117.193 | 39.173 |
| 1015A | 勤俭路 | 天津 | 117.145 | 39.1654 |
| 1016A | 南京路 | 天津 | 117.184 | 39.1205 |
| 1017A | 大直沽八号路 | 天津 | 117.237 | 39.1082 |
| 1018A | 前进路 | 天津 | 117.202 | 39.0927 |
| 1019A | 北辰科技园区 | 天津 | 117.1837 | 39.2133 |
| 1020A | 天山路 | 天津 | 117.269 | 39.1337 |
| 1021A | 跃进路 | 天津 | 117.307 | 39.0877 |
| 1023A | 第四大街 | 天津 | 117.707 | 39.0343 |
| 1024A | 永明路 | 天津 | 117.457 | 38.8394 |
| 1025A | 航天路 | 天津 | 117.401 | 39.124 |
| 1026A | 汉北路 | 天津 | 117.764 | 39.1587 |
| 1027A | 团泊洼 | 天津 | 117.157 | 38.9194 |
| 1028A | 化工学校 | 石家庄 | | |
| 1029A | 职工医院 | 石家庄 | 114.4548 | 38.0513 |
| 1030A | 高新区 | 石家庄 | 114.6046 | 38.0398 |
| 1031A | 西北水源 | 石家庄 | 114.5019 | 38.1398 |
| 1032A | 西南高教 | 石家庄 | 114.4586 | 38.00583 |
| 1033A | 世纪公园 | 石家庄 | 114.5331 | 38.01778 |
| 1034A | 人民会堂 | 石家庄 | 114.5214 | 38.0524 |
| 1035A | 封龙山 | 石家庄 | 114.3541 | 37.9097 |
| 1036A | 供销社 | 唐山 | 118.1662 | 39.6308 |
| 1037A | 雷达站 | 唐山 | 118.144 | 39.643 |

## 处理数据

等值面组件需要的数据格式如下图所示。我们需要对数据做进一步的加工处理，让其更符合我们的使用。




裁剪面：即研究区域的边界数据。这里是全国区域，是一个 geojson 的格式的数据。

geojson 是一种地理交换格式，如需了解更多关于 geojson 的内容，请参见，geojson标准。

> 插值点数据：示例数据是一个包含经度、纬度、值的一个数组，对应我们的需求就是监测站点的经纬度和监测站点对应的某个指标的值。

如果仅做一天的某个时段的等值面图，例如2017年1月20日的中午12点的关于 AQI 指标的图，那么我们需要知道当天这个时段，每个监测站点的位置也就是经纬度信息和对应的 AQI 的值。

推荐写一段 node 脚本来处理**全国监测站点**的 csv 文件。监测站点编号为 key，站点信息为 value。

```
var csv = require("fast-csv");
var fs = require('fs');
var map = {};
csv
  .fromPath("./站点列表(含经纬度)-新-1497个.csv", { headers: true, objectMode: true })
  .on("data", function (data) {
    map[data['code']] = data;
  })
  .on("end", function () {
    fs.writeFile('./站点列表经纬度映射.json', JSON.stringify(map));
    console.log("done");
  });
```

接下来处理2017年1月20日的全国1497个监测点数据。

编辑一段脚本，处理包含了当天24小时每个监测站点各个空气质量指标的信息，我们将这些信息提取出来，并根据前面获取的站点列表经纬度映射表，给站点加上经纬度信息。

```
var fs = require('fs');
var csv = require("fast-csv");
var mapdata = require('./站点列表经纬度映射.json');
var file = './站点_20170101-20170202/china_sites_20170120.csv';
var filename = file.replace(/^.*[\\\/]/, '').split('.')[0].split('_')[2];
var datas = {};
csv
  .fromPath(file, { headers: true, objectMode: true })
  .on("data", function (data) {
    if (data.type === 'AQI') {
      datas[data.hour] = [];
      for (var key in data) {
        if (mapdata[key]) {
          datas[data.hour].push({
            name: mapdata[key].name,
            value: +data[key],
            code: mapdata[key].code,
            city: mapdata[key].city,
            lng: +mapdata[key].lng,
            lat: +mapdata[key].lat
          })
        }
      }
    }
  })
  .on("end", function () {
    fs.writeFile('./data/' + filename + '.json', JSON.stringify(datas));
    console.log("done");
  });
```

将每天的时间段作为 key，每个时间段所对应的所有监测站点的 AQI 信息和位置等信息的数组，作为对应的值。这样就可以方便地获取当天每个时间段的数据给等值面组件使用。

```
{
  "0": [{ "name": "万寿西宫", "value": 18, "code": "1001A", "city": "北京", "lng": 116.366, "lat": 39.8673 }, { "name":
"定陵", "value": 25, "code": "1002A", "city": "北京", "lng": 116.17, "lat": 40.2865 }, ...],
  "1": [{ "name": "万寿西宫", "value": 28, "code": "1001A", "city": "北京", "lng": 116.366, "lat": 39.8673 }, { "name":
"定陵", "value": 65, "code": "1002A", "city": "北京", "lng": 116.17, "lat": 40.2865 }, ...],
  "2": [{ "name": "万寿西宫", "value": 88, "code": "1001A", "city": "北京", "lng": 116.366, "lat": 39.8673 }, { "name":
"定陵", "value": 95, "code": "1002A", "city": "北京", "lng": 116.17, "lat": 40.2865 }, ...]
  ...
}
```

## 处理接口

根据时间轴的特性，如果想要时间轴变化的同时，等值面的数据也发生变化，那么我们需要一个接口，或者数

据库能根据时间参数来获取不同时间段的全国各个监测站点的数据。

这里推荐写一个简单的接口来完成这个需求。

请求地址：/aqi

请求方式：GET

请求参数：

参数名称：date

参数类型：string， 示例2017012722，时间格式为YYYYmmDDHH

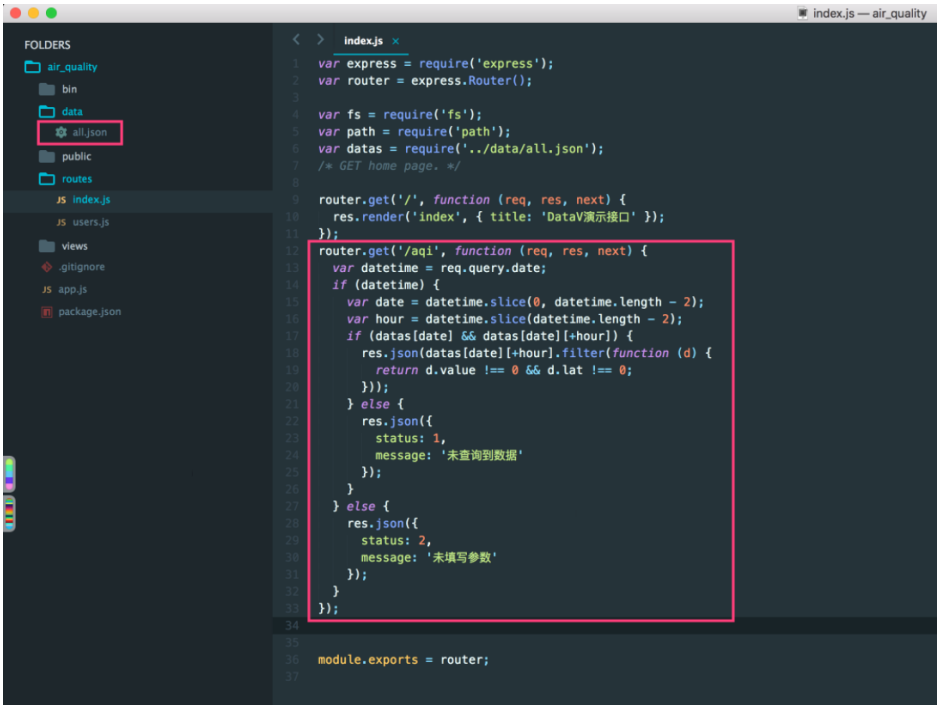
这里需要提前处理好下载的所有数据，node 提供了一个 glob 模块可以对文件夹下的所有数据进行批量处理。

```
var fs = require('fs');
var csv = require("fast-csv");
var glob = require('glob');
var mapdata = require('./站点列表经纬度映射.json');
glob("./站点_20170101-20170202/*.csv", function (err, files) {
  files.forEach(function (file) {
    var filename = file.replace(/^.*[\\\/]/, '').split('.')[0].split('_')[2];
    var datas = {};
    csv
      .fromPath(file, { headers: true, objectMode: true })
      .on("data", function (data) {
        if (data.type === 'AQI') {
          datas[data.hour] = [];
          for (var key in data) {
            if (mapdata[key]) {
              datas[data.hour].push({
                name: mapdata[key].name,
                value: +data[key],
                code: mapdata[key].code,
                city: mapdata[key].city,
                lng: +mapdata[key].lng,
                lat: +mapdata[key].lat
              })
            }
          }
        }
      })
      .on("end", function () {
        fs.writeFile('./data/' + filename + '.json', JSON.stringify(datas));
        console.log("done");
      });
  });
});
```

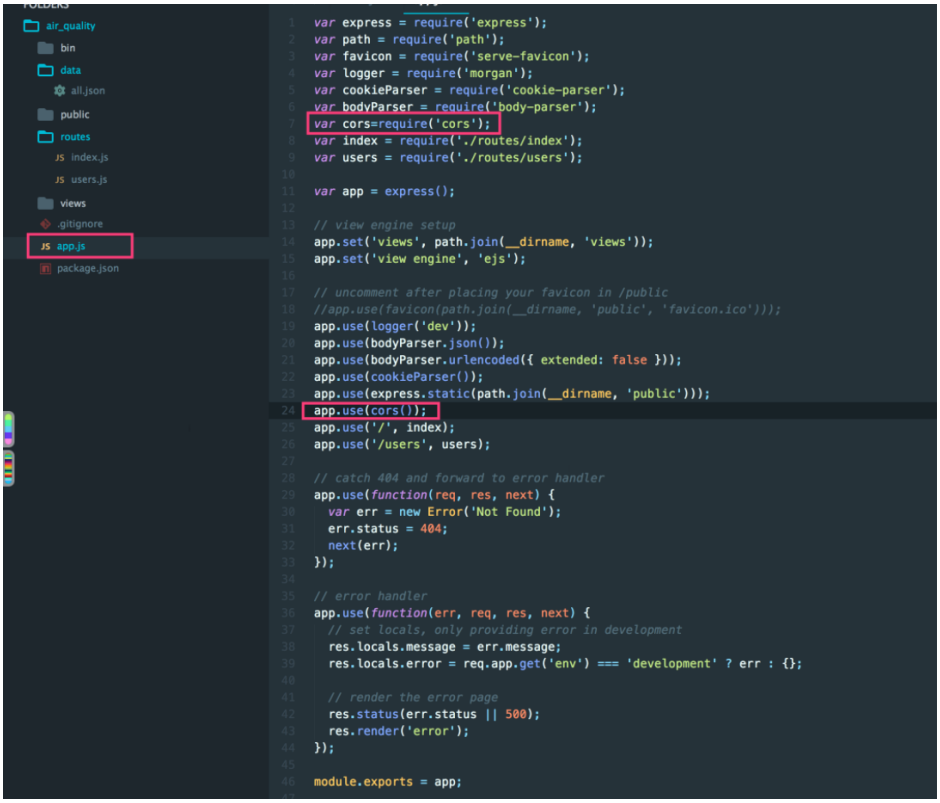
再用 glob 模块对数据进行一次整合。将文件名也就是日期作为 key，对应的内容作为值。

```
var fs = require('fs');
var csv = require("fast-csv");
var glob = require('glob');
glob("./data/*.json", function (err, files) {
  var datas = {};
  files.forEach(function (file) {
    var filename = file.replace(/^.*[\\\/]/, '').split('.')[0];
    datas[filename] = require(file);
  });
  fs.writeFile('./data/all.json', JSON.stringify(datas));
  console.log('done');
});
```

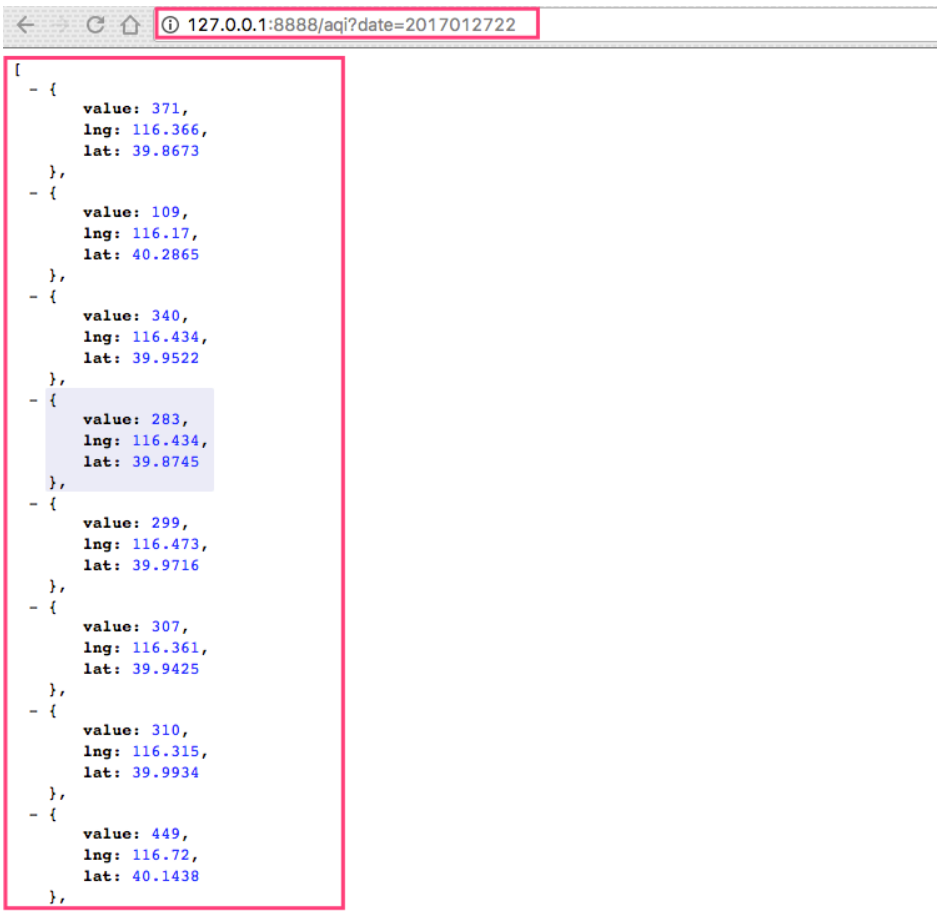
得到了一个 all.json 整合文件后，用 node 的 express 框架初始化一个 express 项目，然后按照上面的接口需求增加一个简单的接口。




**注意**：为了避免跨域请求的问题，您可以在 app.js 文件中增加 cors 模块。




接口处理完成后，可以用npm start命令，测试下接口。




# 制作大屏

前期准备完成之后，我们就可以开始制作一张地图大屏了。

制作一个地图大屏，需要完成以下几个步骤：

- 创建可视化应用
- 添加组件
- 添加数据

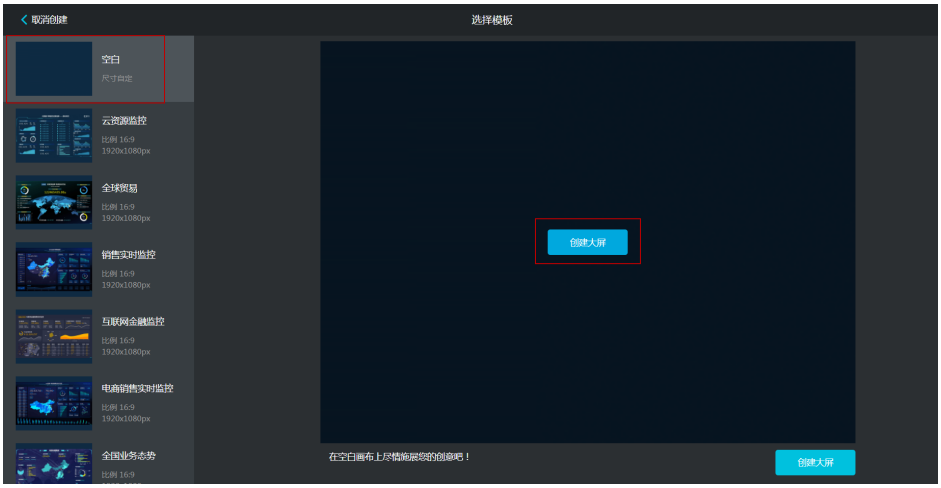
**注意**：本示例中使用的数据源是本地 API 文件，因此您不需要在DataV中添加数据源，直接在可视化项目的组件中调用 API 即可。如果您使用的是其他数据源，在创建可视化应用之前，你需要先添加数据源。

## 步骤 1 创建可视化应用

登录 DataV 控制台。

单击**我的可视化** > **新建可视化**。

选择空白模板，并单击**创建大屏**。



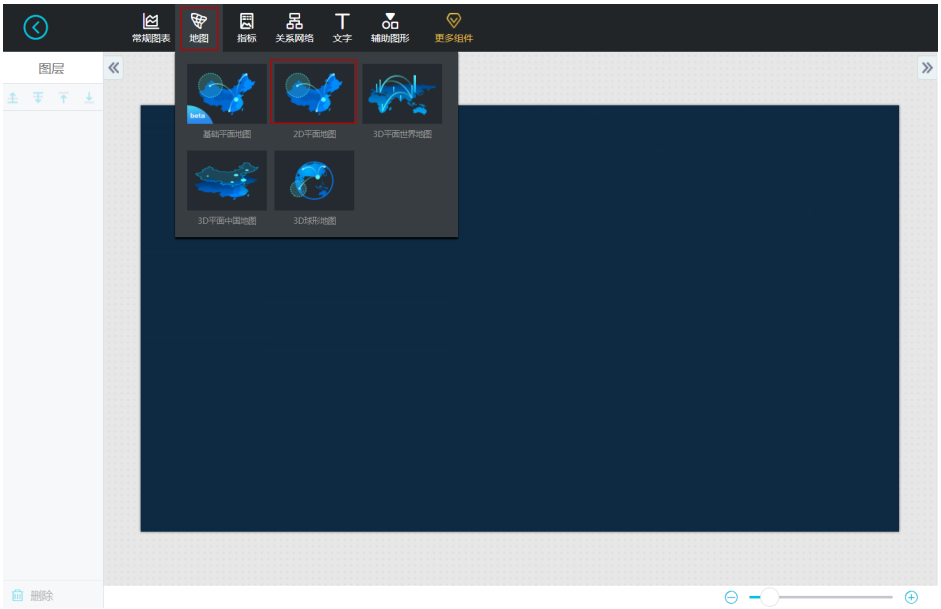

输入一个大屏名称，并单击**创建**。

应用创建成功后会跳转到应用编辑器页面。

# 步骤 2 添加组件

## 添加地图和子组件

在应用编辑器页面，单击**地图** > **2D平面地图**。




在**样式**标签页中，删除其余子组件，只保留**底图层**。


添加等值面子组件。

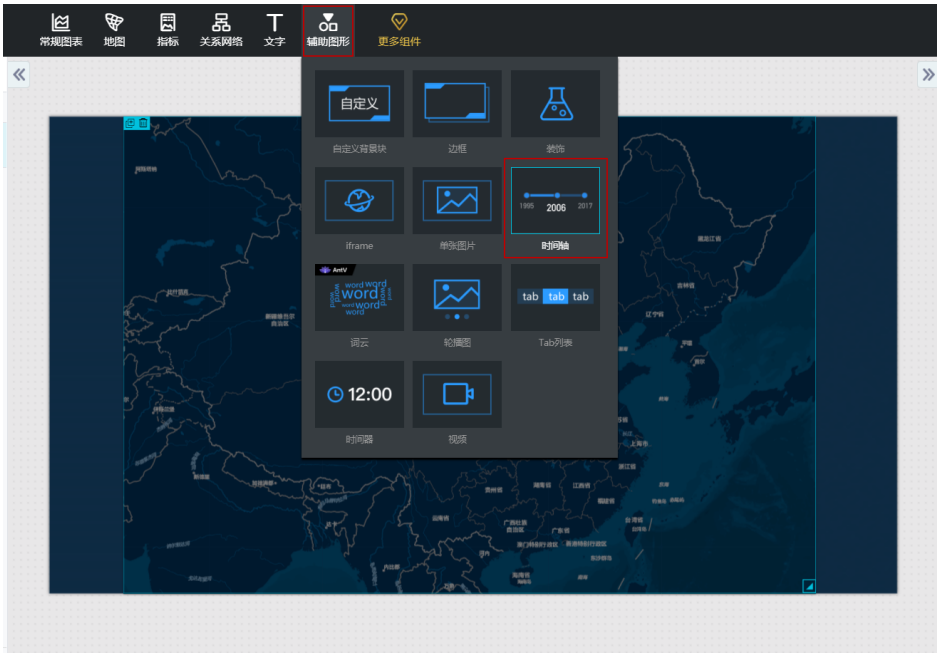
单击**选择组件**的下拉箭头，选择**等值面层**。单击**添加子组件**，完成子组件添加。


单击**全局参数**，调整地图的大小。

您可以拖拽浮点或者手动输入数值来调整地图的大小和显示范围。


### 添加时间轴

单击**辅助图形** > **时间轴**，在地图上添加一个时间轴。




## 添加地图标题

单击**文字**，选择**通用标题**，为地图添加一个标题。


## 调整组件的图层和位置

添加组件之后，您可以根据需要通过右侧导航栏中的**图层**设置组件的图层。

您还可以选中某一个组件，在画板上通过拖拽调整组件在画板上的位置。

# 步骤 3 添加数据

## 添加地图数据

在大屏上，单击地图组件，打开地图编辑菜单。

单击**数据**，打开数据标签页。

在子组件管理中，单击等值面层，打开等值面数据编辑页面。

本示例的数据区域是全国范围，这里的裁剪面的数据可以不动。您也可以根据自己需要修改这里的裁剪面数据。

单击 **插值点数据** 打开插值点数据编辑页面。

设置数据配置信息。

- **数据源类型**：由于前面已经写好了对应的 API，也已经测试了一下数据获取，我们修改等值面组件的插值点的数据源类型为 API。
- **URL**： 填写前面接口测试的那个地址（这里测试 http://127.0.0.1:8888/aqi?date=2017012722）。

单击**查看数据响应结果**，可以看到已经得到了正确的响应结果并匹配成功。

设置等值面层组件的样式。

单击**样式**，打开等值面层的样式菜单。

设置**像元大小**，推荐设置为3。

像元大小的数值越大插值越快，精度越低。


设置**渲染方式**，推荐设置为**线性渲染**。


设置**分类数目**，推荐设置为35。


## 添加时间轴数据

在大屏上，单击时间轴组件，打开时间轴编辑菜单。

单击**数据**，打开数据标签页。

设置**数据源类型**为**静态数据**。

参照标签页的示例，创建我们需要的数据，并替换时间轴数据面板上的示例静态数据。

例如，选择2017年1月22日到2017年2月2日的每天的22点作为时间轴数据。

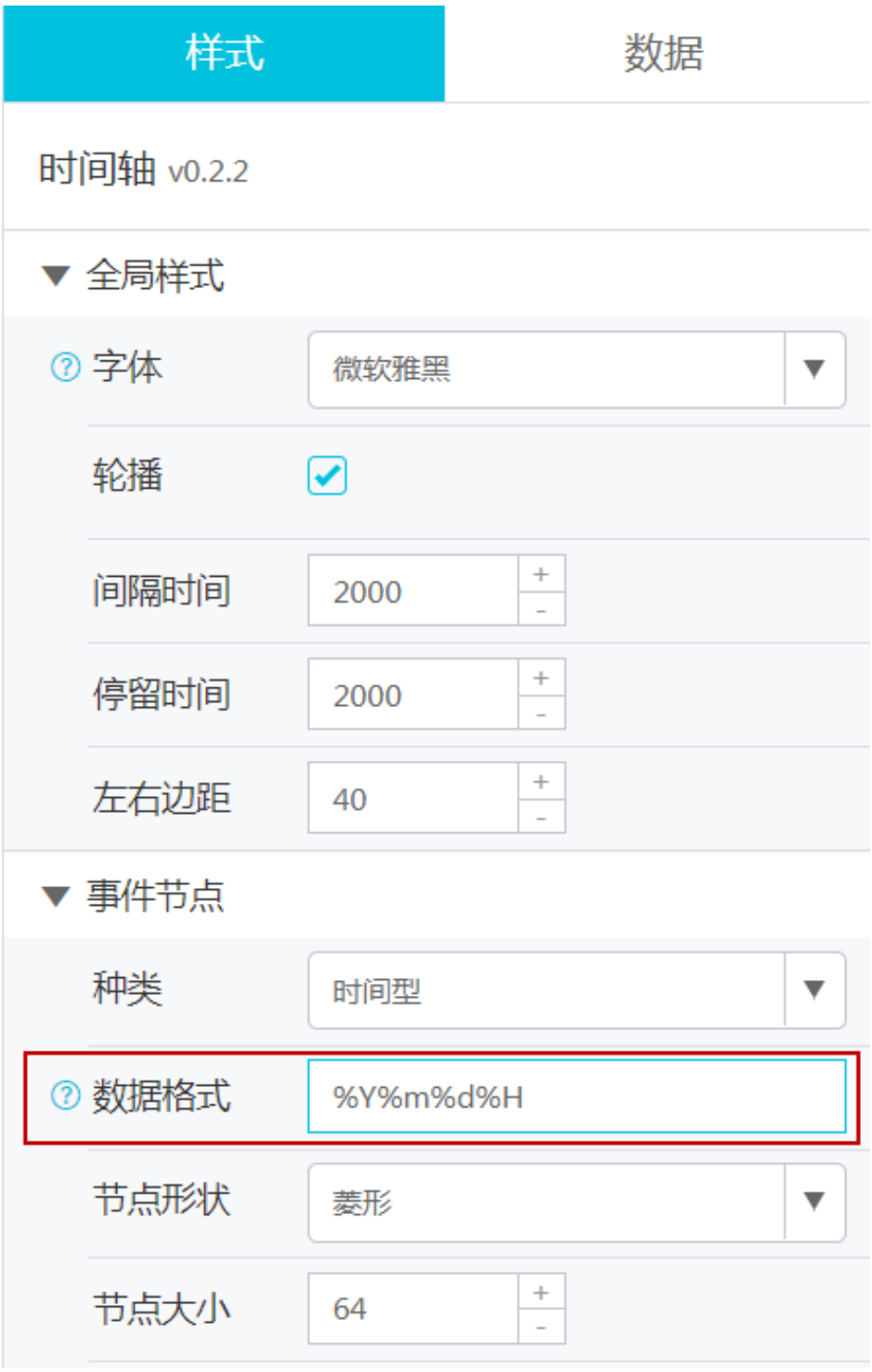
```
[
 {
   "name": "2017年1月22日22时",
   "date": 2017012222,
   "value": 2017012222
 },
 {
   "name": "2017年1月23日22时",
   "date": 2017012322,
   "value": 2017012322
 },
 {
   "name": "2017年1月24日22时",
   "date": 2017012422,
   "value": 2017012422
 },
 {
   "name": "2017年1月25日22时",
   "date": 2017012522,
   "value": 2017012522
 },
 {
   "name": "2017年1月26日22时",
   "date": 2017012622,
   "value": 2017012622
 },
 {
   "name": "2017年1月27日22时",
   "date": 2017012722,
   "value": 2017012722
 },
 {
   "name": "2017年1月28日22时",
   "date": 2017012822,
   "value": 2017012822
```


- **name**：时间轴的轴点显示的内容。
- **date**：作为回调 ID 选项使用。
- **value**：对应的时间。

设置时间轴的样式。

单击**样式**，回到样式菜单。

单击**事件节点**，设置数据格式为**%Y%m%d%H**。




单击**交互**，设置**回调 ID**的值为 date。


### 添加地图标题数据

在大屏上，单击标题组件，打开标题编辑菜单。

单击**数据**，打开数据标签页。

设置**数据源类型**为**数据库**。


在**选择数据库**列表中，选择一个数据库。

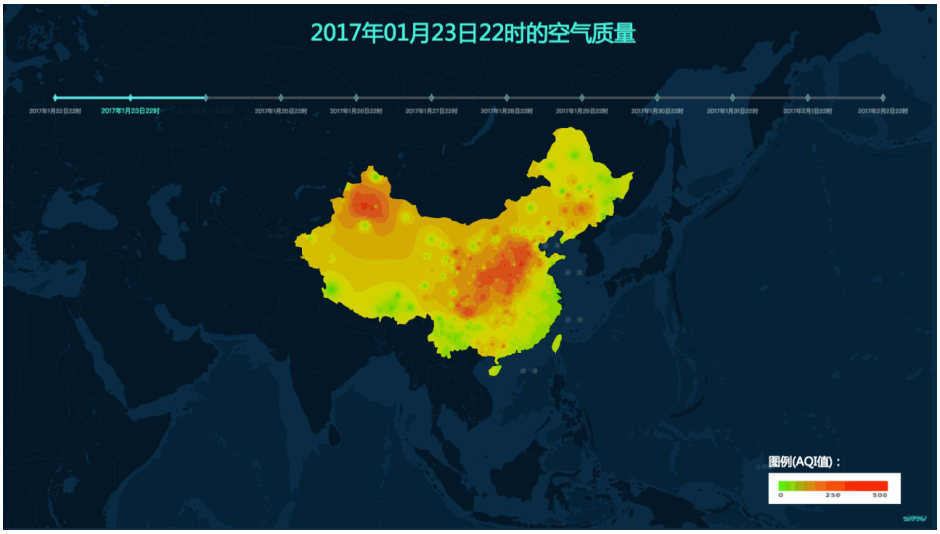
如果没有可选的数据库，您可以单击**新建**，按照系统提示，新建一个数据库。更多新建数据库的内容，参见配置数据源。

手动输入以下命令到 SQL 区域。

```
select to_char(to_timestamp(:date,'YYYYMMDDHH24'),'YYYY年mm月DD日HH24时')||'空气质量' as value;
```

:date 在实际浏览时会传入回调 ID 对应的值。

根据实际需要，您还可以插入显示图例。大屏最终展示效果如下图所示。


2017年01月23日22时的空气质量
图例(AQI值)：
0
250
500