

东京风力图

Tokyo Wind Map

ANONYMOUS¹

July 13, 2017

¹Students from DATA130012.01, Visualization Class

For Visualization Class, 2017 Spring.

Contents

1 序言	1
2 数据准备	3
2.1 数据获取: 我们为什么选择东京	3
2.2 数据处理: 建立适合可视化的数据	4
2.3 地图数据: 建立地图, 标记检测站点	5
3 向量场	7
3.1 多维插值: 找到每一处的风速	7
3.1.1 向量插值算法IDW	7
3.1.2 批量插值	8
3.2 生成粒子: 风力图的基本单位	8
4 动画制作	9
4.1 演化: 让粒子动起来	9
4.2 绘画: 在canvas上作图	9
4.3 优化: 做出更美观的动画	10
4.3.1 参数调整	10
4.3.2 润色设计	10
5 功能设计	11
5.1 时间轴: 实时、历史数据的展示	12
5.2 气象轴: 特殊天气的陈列室	12
5.3 背景轴: 展示不同的空气指标	13
5.4 其他功能	14
6 用户手册	15
6.1 开发环境	15
6.2 使用指南	15
6.3 功能详解	16

6.4 代码结构	16
--------------------	----

1

序言

作为数据可视化课程的期末大作业，我们想尝试用一些现实中的数据做出一份能够与人交互，有美感且有意义的作品。Hint.FM¹和EarthNullschool²就分别做了这样的两个项目，他们利用美国国家天气数据网³的数据，对美国甚至世界范围内的空气质量、洋流、温度等数据在地图上做了很精美的可视化，并且可以实现地图的拖动、小时级别的更新查询。其中，风力数据的可视化特别吸引我。流线型的动画异常优美，就算不是出于查询风力的目的也会得到视觉的享受。

wind map

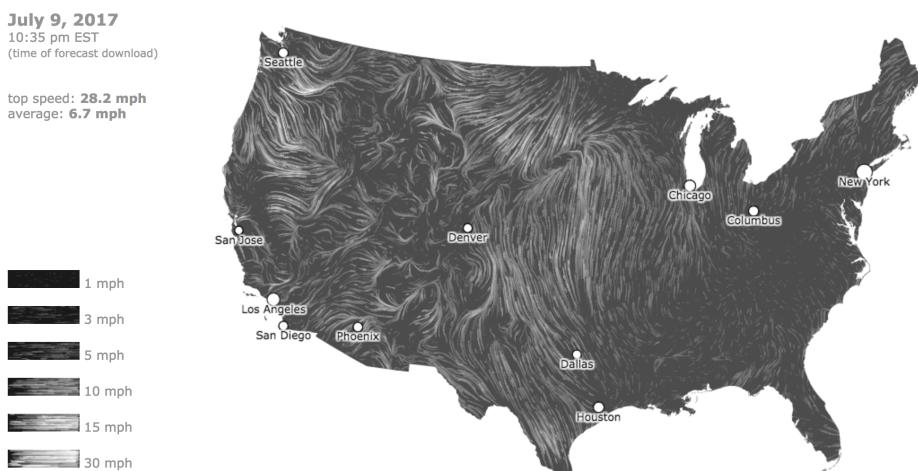


Figure 1.1: Hint.FM的成果图

¹<http://hint.fm/wind/gallery/mar-14.js.html>

²<https://earth.nullschool.net/>

³<http://ndfd.weather.gov/technical.htm>

工作概览

工作的流程主要包括数据集的获取、数据的处理和计算、对地图生成可视化、对风力数据和更多数据的可视化、交互功能和一些前端设计。我们的报告也会按这样的顺序展开。

整个项目从可视化方法的建立，到交互网页的制作都是用Javascript完成的，借助了d3.js, node.js等工具或库。而数据的爬取、整理和转换用python2进行处理。

2

数据准备

2.1 数据获取：我们为什么选择东京

数据的获取可以说是第一步的挑战。Hint.FM采用的是NDFS美国国家数据¹，另外我们国家也有气象数据服务的API——中国气象数据网²，但是光注册就很麻烦，而且数据的请求很慢，格式难解析，总之非常繁琐。在这同时，东京都环境局³则提供了非常便于查询，每小时更新，而且很容易解析的数据。浏览一下东京环境局官网⁴就会发现这实在是一个很可爱的网站，包括东京的生态环境保护的一切。在那里，所有的数据，包括空气污染、水资源、车辆排放和噪音污染的数据与处理对策都是开放的，而且解释得非常详细。数据的格式如下（篇幅有限，仅仅摘了几行观测站的数据和部分空气污染指数）：

Table 2.1: 2017年7月10日14时(GMT+9)东京监测数据

观测站 编号	观测站名	风向	风速 0.1m/s	SO_2 ppb	O_x ppb	NO ppb	PM 2.5 $\mu g/m^3$
101	千代田区神田司町	ESE	6	4	26	10	10
102	中央区晴海	ENE	9	7	15	16	10
103	港区高	SSE	16	null	35	4	10
104	港区台	SE	21	6	18	15	9

数据集中有82个分布于东京的观测点，其中约有50个数据包含无缺失值的风向风速数据，对于东京这样面积的地方来说能画出一个风场了。值得注意的是，风向数据是用ESE、SE这样的字母来表示的，所以预处理的时候

¹<http://ndfd.weather.gov/technical.html>

²<http://data.cma.cn>

³<http://www.taiki.kankyo.metro.tokyo.jp/cgi-bin/bunpu1/p101.cgi>

⁴<http://www.taiki.kankyo.metro.tokyo.jp>

我们需要先把它们化到：东东南风112.5度，东南风135度这样的数字化信息，然后与风速结合，得到一个(x,y)坐标的平面向量。

另外，除了风向以外还有十条如二氧化氮、PM2.5之类的空气污染粒子浓度，以及气温、湿度数据。利用这样的一个数据集，我们就可以绘制以某个空气污染指标或以气温为函数值的热力图可视化，加上风向矢量能够得到一副静止的图片。

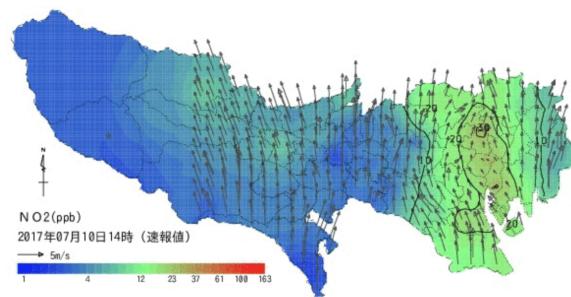


Figure 2.1: 东京环境局的网站对风力数据的可视化，其中背景颜色是二氧化氮的浓度密度图。

2.2 数据处理：建立适合可视化的数据

预处理的步骤主要有：数据清洗、格式整理、归一化、导入JSON文件。

- 空值、异常值处理。常常有一些观测站得到的部分数据为null。我们选择不存储空值的项目，因为所有点的值都可以通过插值来获得，而插值的时候仅利用有数据的点即可。存在一个特殊的气象站“京叶道路龟户”，有一些异常值，我们将序号208的站去除就可以了。
- 存储单位。对于温度（摄氏度）、湿度（百分制）这样的变量只需要原本保存就可以了，而对于空气污染指数（ppb和毫克/立方米），则选择将其缩小一千倍。
- 归一化。为了保证几种浓度数据可以在同样的颜色轴上显示，并且让人直观感受到浓度范围与真实数据的联系，需要将数据归一化，限制在[0,1]上。通过 $\frac{x-min}{max-min}$ 就能够完成了，而对于空气污染指数，由于它们极端的很多，需要做一步对数处理，使它们分布得均匀一些。但是为了在地图上显示数据，我们只在可视化的时候做这一步工作，不保存在数据里。

处理过后的气象数据中的一条数据在JSON文件中存取的格式为：

```
{"stationId":131,"so2":0.006,"ox":0.061,"no":0.004,"no2":0.03,"nox":0.034,
"spm":46,"pm25":35,"wd":90,"wv":0.9,"temp":33.2,"hum":60.9}
```

其中“wd”和“wv” 分别代表风向(direction)和风速(velocity)，就像上一节所说的，我们以正北方向为0度，顺时针增加，无风为360度，通过字母风向和其代表的角度建立了一一对应关系，而这比较像极坐标中的表示：方向角，半径。为了在平面上更好地表示向量需要做一步转换：

$$(\rho, \theta) \rightarrow_f (x, y)$$

转换公式为：

$$\begin{aligned}\rho &= wv, \quad \theta = wd \times \frac{2\pi}{360} \\ x &= -\rho \sin \theta, \quad y = \rho \cos \theta\end{aligned}$$

这里有一个小技巧：由于风向的意思是风吹来的方向，东南风意味着吹向西北的风，所以得到的方向要取相反数。又由于像素从上到下是递增顺序，恰恰和坐标轴相反，y方向又要取一次相反数，得到这个结果。

2.3 地图数据：建立地图，标记检测站点

地图可视化中的底图需要地图几何数据的展示，需要找到东京地图边界的几何数据。数据来源为国土数值情报官网⁵。

还有一个至关重要的信息是每一个观测点的经纬度，用于之后的插值计算，存储在/`data.station.json`文件中。

我们采用墨卡托投影，绘制底色为黑色的地图。其实对于一个城市这样大的地方来说，采用何种投影方式的视觉差别不大。得到了这样的地图轮廓：



Figure 2.2: 地图轮廓的建立

⁵<http://nlftp.mlit.go.jp/ksj-e/index.html>

3

向量场

下面要进行风向图的绘制了。大致的思路是去制作可视化的粒子（particles），粒子最初散落在地图上的各个位置，而每个位置有一个风向量 v ， t 是很短的时刻， vt 就指向了到下一时刻粒子运动的路径。就这样一帧一帧地绘制粒子，给每一帧之间加时间间隔，就可以看到风的形象了。

那下面的任务就要在地图上建立向量场，让每一个落到向量场上的粒子都可以“动起来”。

3.1 多维插值：找到每一处的风速

前面已经说到，要让落到向量场上的粒子获得一个速度，那么我们需要获取任意位置的速度向量。然而数据由不超过八十个点的有效观测站提供。八十个点是无法在东京城市的每一个角落获取向量，从而绘制完整的风向、热力图的。所以剩余其他点的数值由插值得到，这是可视化至关重要的一步。

3.1.1 向量插值算法IDW

IDW法，即反向距离加权(Inverse Distance Weighting)方法，有着很简单的方法却也能实现很好的效果。基本想法是，某个点的值由它的几个最近邻的值决定，与距离的倒数正相关，加权得到插值。利用kdtree的方法找到 (x, y) 的近邻 $\{(x_i, y_i)\}_i$ 后，用以下公式进行插值

$$v(x, y) = \begin{cases} \frac{\sum_i w(x_i, y_i) v(x_i, y_i)}{\sum_i w(x_i, y_i)}, & \text{if } \prod_i d_i \neq 0 \\ v(x_i, y_i), & \text{if } d_i = 0 \end{cases}$$

其中 v 为要得到的值， d_i 为 (x, y) 到第*i*个点的距离， w 为某个点的权重，有

$$w(x_i, y_i) = \frac{1}{d_i^p}$$

经验性地，我们取 $p=2$ 。

3.1.2 批量插值

为了让每一个像素点都能有一个相应的风速，可以从左到右将一幅图扫一遍，就是在境内固定横坐标，找到境内的纵坐标一竖，从上到下一个一个像素插值。横向有1197个像素，而纵向则根据地图形状不同而不同。所有像素点就被存储为一个二维数组，每一维竖向存储像素格的编号，而每一个编号对应一个向量。

由于风向是连续的，所以一个风粒子可能刮到恰好不是整数倍像素格的位置。我们就取它所在的正方形四个顶点做双线性插值。插值向量的形式为：

$$(dx, dy, v),$$

其中 dx 与 dy 为风向量的分速度， $v = \sqrt{dx^2 + dy^2}$ 为风力大小。

3.2 生成粒子：风力图的基本单位

粒子是用来描述并可视化风向的工具。通过观察粒子的轨迹我们能得到风的纹理。那么粒子是如何放置的呢？

利用存储像素编号的二维数组，假设数组一共包含10,000个元素，而我们想要1000个粒子，那就从1到10,000的整数中生成1000个随机数，把它们对应到二维数组的编号里去，就可以获得其位置了。

4

动画制作

现在我们有了风粒子，问题是怎么让它们动起来，优美地动起来。我们需要一条时间轴，每一个粒子拥有它的生命周期，在生命周期内画出一条轨迹。

4.1 演化：让粒子动起来

一个粒子所携带的信息：粒子位置坐标(x,y)，速度风向(dx,dy,v)，已经经过的帧数。粒子进入下一帧之前要考察它是否已经达到了最大周期生命耗尽，或者是否已经出界。用伪代码表述就像这样：

算法：粒子演化

```
for each particle do
    获得位置(x,y)、速度(dx,dy,v)和生命值age
    if particle out of bound
        age<- maxAge
    else
        xt<-x+dx, yt<-y+dy
        将(x,y)->(xt,yt)绘制出来
        x<-xt, y<-yt age<-age+1
    if age=maxAge
        转生，随机分配粒子的位置
```

4.2 绘画：在canvas上作图

在上一步我们用一个数组存下要绘制的(x,y)->(xt,yt)对，接下来我们要在地图范围内的canvas对象上把这些点连起来。让两点连线的方法很简单，

在canvas上做c.beginPath(); c.moveTo(x,y); c.lineTo(xt,yt); c.stroke(); 这样几步操作就可以实现了。

更重要的问题是让过去的粒子逐渐消失，否则这样的小线段会在地图上越积越多，不能看到很好的一条条流淌的曲线了。

一个很有意思的技巧：在每一帧绘制线段过后，在canvas上加一个不透明度为 α 的图层，这可以用fillRect方法去填一层canvas的矩形。我们把globalCompositeOperation属性设为"destination-in"，可以在地图中显示半透明矩形而且只有地图内的矩形会被显示，不会覆盖到其他区域。这样的话，每一条短线段的生命就为 $1/\alpha$ 帧。这个过程就像是画一笔就刷一下一种不透明颜料，于是渐变型的线条就画了出来。事实证明， $\alpha = 0.05$ 时候这种方法能达到很不错的效果。

4.3 优化：做出更美观的动画

4.3.1 参数调整

首先要确定构成动画的一些参数，以下是我们尝试后觉得效果不错的参数：

- 粒子总数： ParticleCount = 500
- 粒子最大生命周期： 40帧
- 帧频率： 每秒刷新的帧数，取一般flash动画的24帧/秒
- 风速对应到像素上的移动速度转换： 1:1的比例就可以了，效果不错。
- 每一帧过后图层的透明度： $\alpha = 0.05$ ，颜色为黑色
- 近邻插值数量： k=5个邻居

4.3.2 润色设计

对于离开地图边界的点，它们迅速“转生”，所以视觉上会有一些突兀。我们可以拓宽地图的边界，并且增设一个INVISIBLE的值，把粒子的状态分为三种：粒子在地图严格边界内、粒子在地图的拓宽边界内但在严格边界外(INVISIBLE)，粒子在拓宽边界外(转生)。这个实现可以通过两种颜色通道来实现。采用红色通道和蓝色通道，将两种颜色都填满地图严格边界，也就是[255,0,255,1]。我们在地图边界外画一个蓝色的宽度为1的边界，也就是[0,0,255,1]。那么对于落在这个图层上的点，我们用canvas.getImageData来获得它的第0个元素（红色）是否大于零，以及第2个元素（蓝色）是否大于零。

5

功能设计

风力数据已经能够用流线动画的方式展现出来，我们需要把它嵌入到一个功能性的网页里展现出来。这个网页可以提供很多的功能，包括人与地图的交互，实时的气象数据展示和历史数据查询，晴朗无风、雷雨台风等特殊天气的展示。

另外，由于数据集还包括很多的空气污染指数以及气温、湿度等指标，我们同样可以用插值的方法使它们在地图上呈现出来，与风力数据一并展示，也许可以获得一些新的发现。

在展示地图的下方有几排功能轴，提供选择时间、气象轴和背景色。网页的设计如下图。这一节将会就具体功能和设计进行描述。

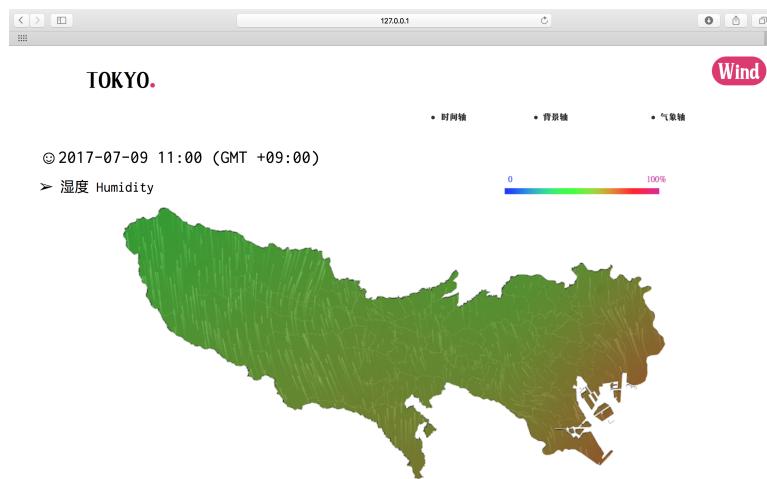


Figure 5.1: 网页全貌

5.1 时间轴：实时、历史数据的展示

东京天气数据是每小时更新的，通过实时爬取能让数据自动保存到本地的数据库，再拿出来做可视化。在网页上，我们提供了附近一周的查询，可以通过小时和日子进行前后滚动。

每一小时的数据在处理完后都按照当时日期存储，如：2017-05-20-15.json。如果鼠标获得“一小时后”的数据请求命令，就会通过navigateToHour函数获得2017-05-20-16的文件，而一天后就是24小时后。我们通过正则表达式去匹配和分割日期，从而找到存放在/data目录下的文件名。

5.2 气象轴：特殊天气的陈列室

只看一天的风力图很难知道当天具体的天气是什么，所以我们制作了一个天气轴，可以调出无风的天气、下暴雨的天气，甚至台风来时的天气。这个天气轴作为参考，能考察各种天气下我们可视化的表现，也可以获悉当前的天气大概是怎么样的。通过天气网¹查找历史天气情况，我们选择的数据如下：

日期	天气	特点
2017年7月1日10点	阴雨，无风	风流速很缓慢，市中心氮氧化物浓度明显比郊区高，没能扩散出去
2017年5月20日15点	晴朗，南风3级	正常风速，市中心热岛效应很明显 风速较低，污染物浓度较高
2017年6月2日15点	晴朗，西南偏南风5级	风流速非常快，湿度很小，各污染物指标都较低且平均
2017年6月2日18点	雷雨	有明显的强对流，污染物指标低
2016年8月22日15点	台风“蒲公英”	风力非常强，达到30公里每小时，伴随降雨带来90%左右的湿度

以下这幅图展示了2016年第9号强热带风暴蒲公英袭击东京时的风力图。

¹<http://www.tenki.jp/past/>



Figure 5.2: 台风“蒲公英”

5.3 背景轴：展示不同的空气指标

在背景轴上有一系列的按钮可供选择某种空气指标的浓度来作为展示，包括温度、湿度、风速、日照量、氮氧化物、光化学氧化剂、二氧化硫、一氧化碳、甲烷、非甲烷烃、悬浮颗粒物和PM2.5。

这些数据都可以像风向数据一样进行插值，然而我们用到与之前IDW方法不同的插值法：薄板样条插值(thin plate spline)。这里简单介绍一下我们使用样条插值的原理和方法。

为地图表面的插值使用以下公式：

$$S(x, y) = a_1 + a_2x + a_3y + \sum_{i=1}^N \lambda_i R(r_i)$$

其中， a_1, a_2, a_3 和 λ_i 是通过求解线性方程获得的系数， r_i 是这个点到第 j 个观测站的距离，

$$R(r) = r^2 \log r$$

在这个问题里，我们用 L 矩阵表示观测站两两之间计算得到的 $R(r)$ 值，再在最后三列和三行分别加上 $(1, x, y)$ 的坐标信息形成一个对称矩阵。我们还可以加入参数硬度(rigidity)去调整 L 矩阵的对角元素数值。设有 N 个观测站，那么 L 矩阵就有 $(N+3) \times (N+3)$ 的形状；用 B 矩阵表示观测站的 PM2.5 浓度， B 矩阵有 $(N+3) \times 1$ 的形状。我们要找的就是 $LX=B$ 的解 X ，它的前 N 行对应 λ_i ，后三行对应 a_1, a_2, a_3 。 X 由 LU 分解方法解得。此处选取硬度为 0.1 倍的数值范围。

通过与风向相似的插值顺序，我们可以在整个地图的像素上插值，并且把浓度数值化归到 $[0, 1]$ 上，作颜色变换得到。

5.4 其他功能

- 鼠标点击反馈：当鼠标放在地图上某一点的时候会获得反馈，显示当前的经纬度和当前风速、当前背景图的属性值。这是由d3.mouse函数实现的。



Figure 5.3: 鼠标点击反馈

- ColorBar: 为了比对参考地图颜色，我们提供了ColorBar，根据不同的属性值会显示该属性值的范围、单位和对应颜色。



Figure 5.4: 湿度的ColorBar

6

用户手册

其实整个网页没有什么特别需要指导的操作，比较清晰和简单，用户手册里将会介绍一下使用指南、代码结构、功能和部分细节。

详情可以看附件：操作视频。由于匿名化需要，视频做了变速和变音处理。

6.1 开发环境

python 2.7, javascript

6.2 使用指南

需要安装node.js¹，进入node-module文件夹，输入命令（使用-g进行全局安装）：

```
npm install http-server
```

安装http-server来打开网页，http-server是一个简单的零配置命令行HTTP服务器，基于 node.js。

打开网页的操作：进入code文件夹，输入命令

```
http-server
```

复制网址如 <http://127.0.0.1:8080/index.html>，就可以打开网页了。

¹<https://nodejs.org>

6.3 功能详解

打开网址后会看到这样的页面。

- **通过时间轴来选取想看的时间。**可以看到当前时刻往前七天的内容，若没有那天的数据则会显示No data。网页有一行显示当前画面所对应的时间。想要返回现在的时候，可以点击“当前时间”按钮。
- **使用背景轴来选取底层的热力图颜色依据。**在显示时间附近有一个箭头所指的便是当前展示的污染指标了。用鼠标点击地图的任意位置，可以原地显示经纬度、风向和污染指数。想看无热力图的黑色底图，请选择“none”选项。
- **在气象轴上找到特殊天气的图样。**点击“无风”等按钮便会显示7月1日上午10点的一副动画，在指定的污染指数热力图上也可以显示风力。要回到当下画面，请点击时间轴的“当前时间”按钮。

6.4 代码结构

code代码包里的css、fonts、src、都是用于美化网页设计和排版的代码、图片和字体。主要代码储存在js文件夹中。

js目录下最重要的一个文件是map.js，里面写了主要的插值算法、动画效果和事件响应。剩下jquery开头的js文件也是用于网页布局和设计的。另外，我们用到了node.js下的when.js来执行整个异步编程。还有用来开发可视化的d3.v3.js和topojson.v1.js库。