如何用Python科学分析——长沙历年气象情况报告

截至本文发布，整个长沙城已经在雨水里泡了差不多两个月了。





刚好最近又一直在用Python做数据分析方向的研究，本着科(chi)学(bao)分(cheng)析 (zhe)的态度，决定做一期关于长沙历年的气象情况的分析报告。

本文简单记录了整个工作流程，以此方便其他地区的朋友参照本范例科学分析。



【数据篇】

首先当然是去找数据源，历史气象数据不是什么机密数据，在国家气象信息中心就可以找到：

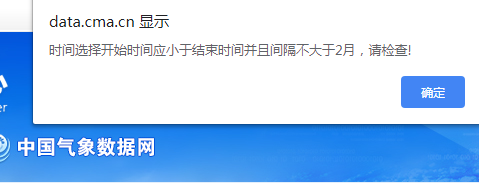


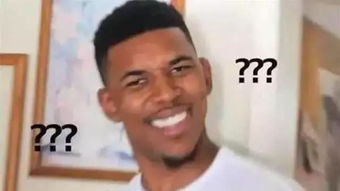
中国地面国际交换站气候资料日值数据集(V3.0)

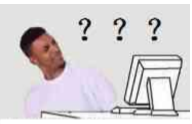
数据还是比较全的，从建国后开始至今接近70年的数据。

看来我国的科学数据信息还是比较开放的。

但是，当你要批量下载数据的时候…







也就是说，一次检索不能大于2个月，那么下载70年的数据需要设置这个页面420次，再加上生成和下载，手速快不吃不喝连续作业也大概需要花掉20个小时的时间，所以…

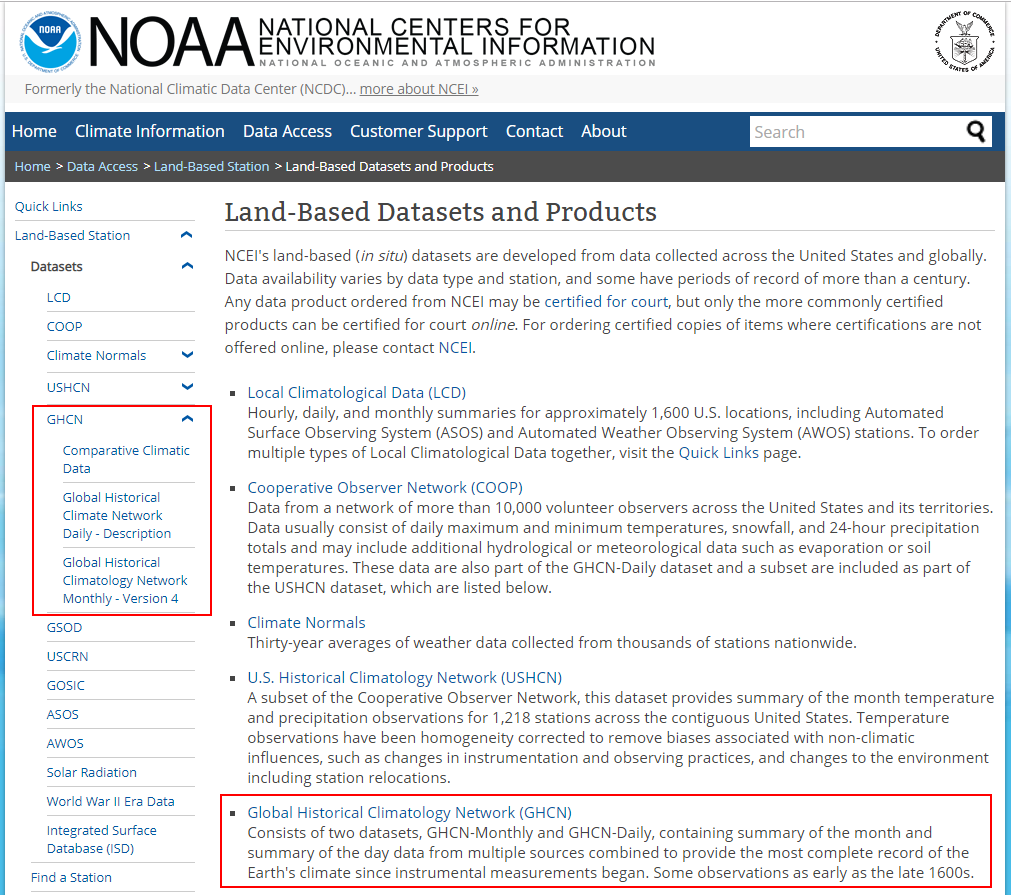


理论上我们可以用python写一个爬虫把数据爬下来，但是不到最后需要自己造轮子的时候，尽量还是寻找更简单的方法。

地面国际交换站之所以叫国际交换站就是因为这个数据是和其他国家共享的，我国下载受限的话，理论上在数据更公开的其他国家（此处手动@USA）可以尝试搜索一下。

NOAA（美国国家海洋和大气管理局）应该有我们需要的数据，首先官网进去找到数据中心的地址，为了节约大家的搜索时间这里直接贴出数据库的入口地址

（<https://www.ncdc.noaa.gov/data-access/land-based-station-data/land-based-datasets>）



我们要找的数据在GHCN（全球历史气象学网络），看描述感觉很硬核，记录了自测量器诞生以来地球的所有历史气象数据，最早可追溯到了17世纪。



我们当然不需要这么多，能获取到目标地区气象站的几十年完整数据即可。

之后通过给出的链接进入FTP，这里可以下载全部的数据，没有任何地区和权限限制，只要连接了网络就可以下载。

不得不说一个国家的科技进步水平和其科技资源的开放程度是成正比的。

点击进入FTP，然后：

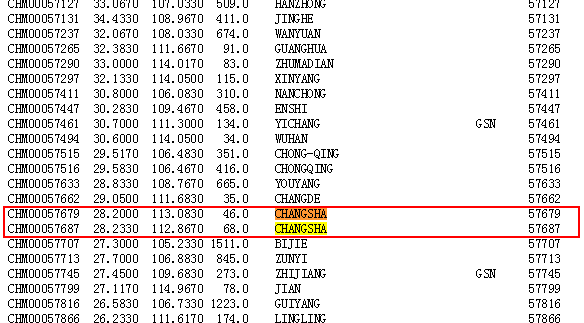


GHCN-FTP根目录



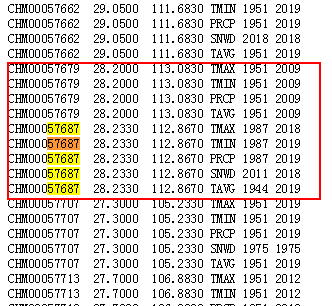
通常进一个FTP数据库，第一时间有点不知道从何入手是很正常的，基本上我会把root下的readme.txt看一遍，一般心里就大概有数了。

ghcnd-stations.txt记录了所有气象站的编码信息，在之后的查询中我们需要用编码来检索数据。



ghcnd-stations.txt

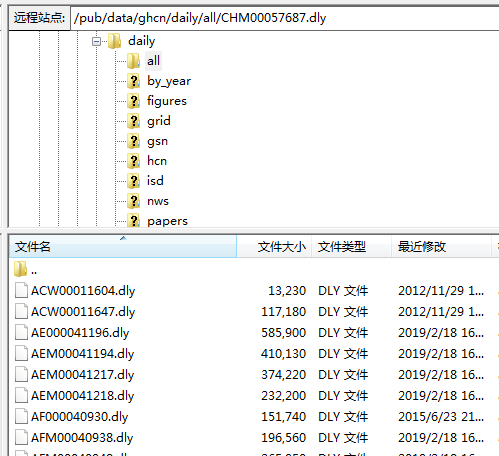
搜索目标地点CHANGSHA（长沙），看到有两个气象站，编号57687的那个之前在中国国家气象信息中心查到过，通过经纬度和地图可以发现它架设在岳麓区。编号57687这个气象站从1970年1月开始监测，但是不存在编号57679的气象站，我们先不管它，可以把这两个站点的数据都下载之后再看。



ghcnd-inventory.txt

ghcnd-inventory.txt记录了数据详细清单，通过搜索之前查询到的长沙气象站的编码，可以看到气象站57687的数据清单， 平均气温（TAVG）记录从1944年就开始了，降雨量（PRCP）从1987年开始。而气象站57679的数据从1951年开始就非常统一和完整，只是在十年前停止了记录，可能是关闭了。所以之后把这两个气象站的数据合并一下，基本就可以完整反映出长沙地区1951-2019的气象情况。

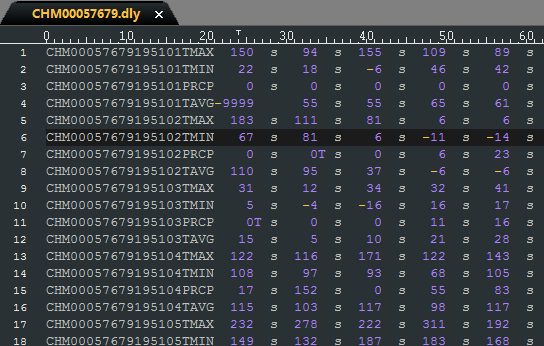
我们需要的数据本体在all目录下：



all目录

分别下载CHM00057679.dly和CHM00057687.dly这两个气象站的全部数据。

最后我们用UE打开文件，可以看到，记录从1951年1月开始，其中包含了最高气温，最低气温，平均气温（0.1摄氏度）及降雨量（0.1mm）。这里的s是列分隔符，每一列表示当月中某一天的数据值。



CHM00057679.dly

其实还是想再多拿到一些数据，比如相对湿度，云覆盖率，气压等。不过有了气温和降水数据基本也可以统计出长沙历史的天气概况了。如果有朋友能够拿到更丰富的气象数据，也欢迎提供一下。

关于数据获取的内容就先说到这里，肯定还有能获取更全面数据的方法，以后有新的发现会另外再开篇拓展。



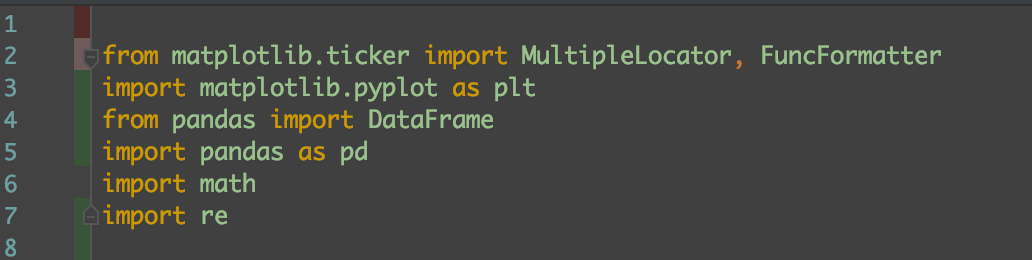
【编码篇】

拿到数据以后，就可以开始用Python编写代码开始分析了：



对代码不感兴趣的同学可以直接跳过本节直接看分析结果。

首先这个demo用到的库如下



Pandas用来做数据整理，Matplotlib用来绘图，Math是数学基本库，以及Re正则库，用来对原始格式的数据进行分割筛选。

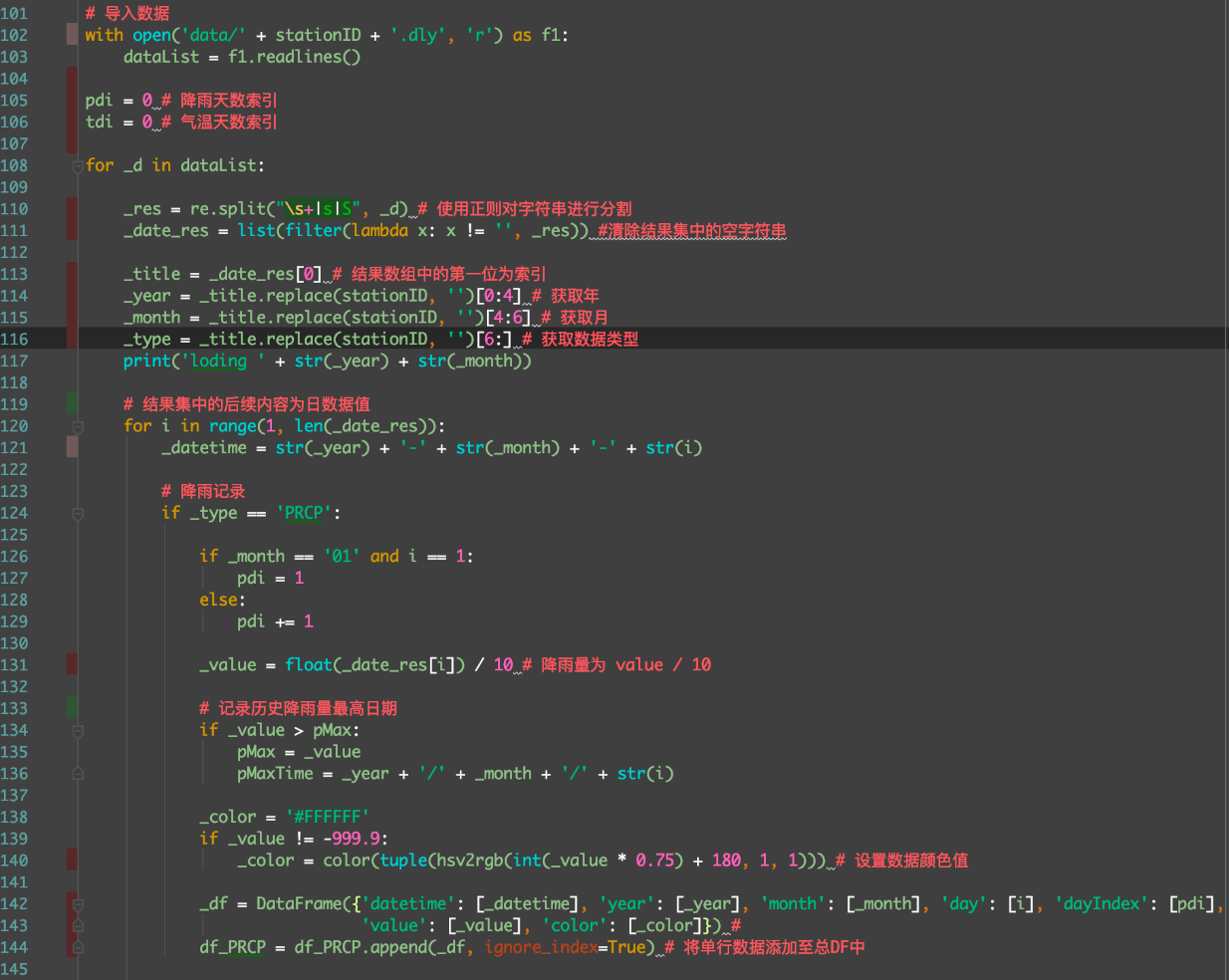
Step1:数据读取

从老美那拿到的数据结构虽然已经比较清晰了，但还是不能拿来直接用，我们需要把逐月数据转换成逐天数据，然后保存到DataFrame里。

首先定义四个空的DataFrame，注释已标。



之后对dly文件进行逐行（逐月）读取，将单行进行字符串的分割，取出时间以及每个单日数据。



这里有两个循环，一个循环是dly文本文件的逐行读取，另一个循环是每一行中每一天的列数值读取和创建一行日数据df。获取并组合成新的日数据df之后，就可以添加进之前创建的指定类型的空DataFrame中。这里只截图了PRCP（降雨统计）的部分，其他部分同理。

关于数据显示时的颜色值，这里用了HSV颜色模型的思路，通过改变H（色相）值，来实现通过颜色表示数据大小的效果。实际也可以不需要，只是为了显示时更好看一些而已。这一块不是重点，可以略过。

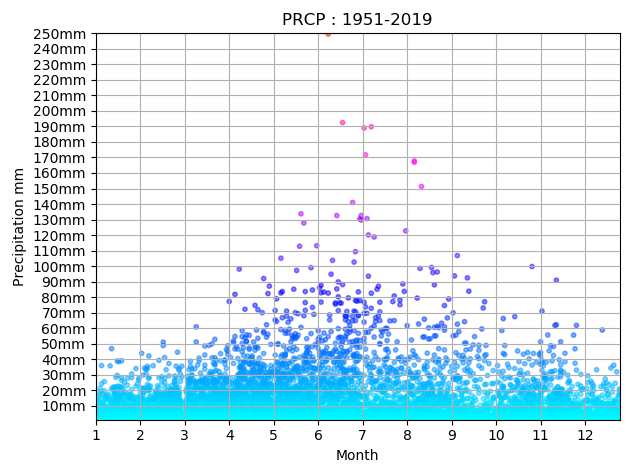
Step2:散点图绘制

依旧以PRCP为例：



绘制图表可以调用Maltplotlib.pyplot库下的各个方法。其中绘制散点图的方法为scatter()，参数中x轴填入日索引（当年的第几天）集合，y轴填入当日降雨量集合，s表示点的大小，这里设置为40，c表示颜色，可以使用固定字符串或者之前设置匹配数值的颜色集合。marker表示点的形状，alpha表示点的透明度。

运行之后的效果是这样的：



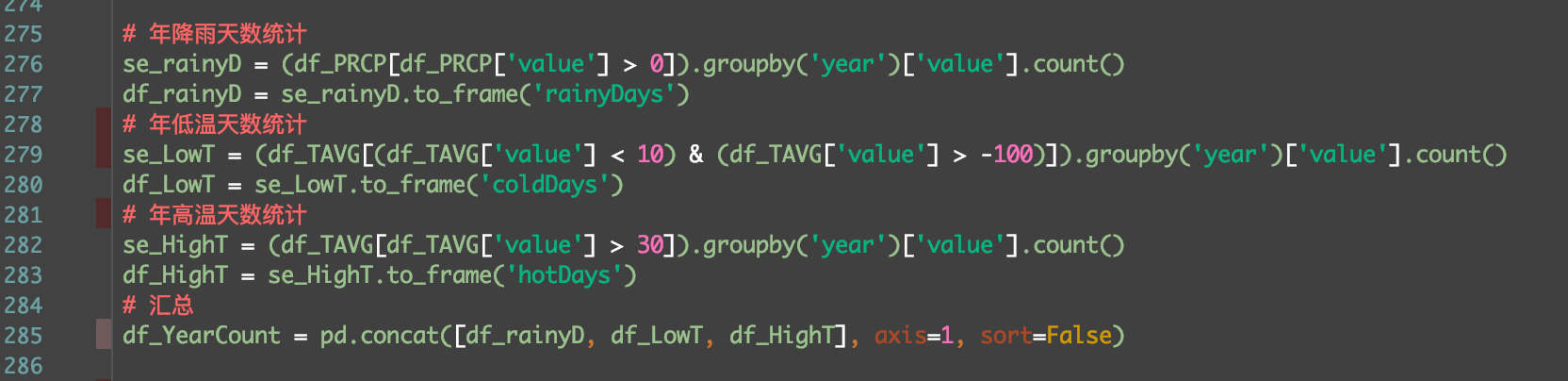
每一个点表示70年中某一天的值，通过这张图就可以大概看出历年降雨量和日期的关系，以及分布情况。

平均气温TAVG也有一张散点图，用来展示历年气温分布和日期的关系，和PRCP的显示方法相同，因此这里就不再重复说明。

Step3:线性图绘制

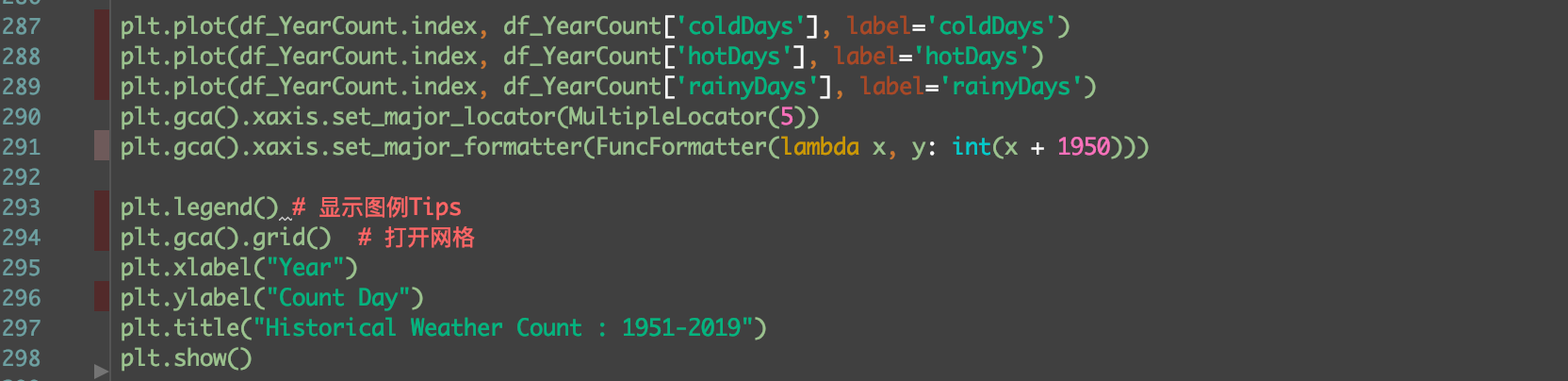
在本步骤中我们将用线性图来展示数据在逐年中的变化情况。需要统计的数据有：单个年份中有降雨的天数，炎热（TAVG > 30°C）天数以及寒冷（TAVG < 10°C）

统计数据的代码如下：

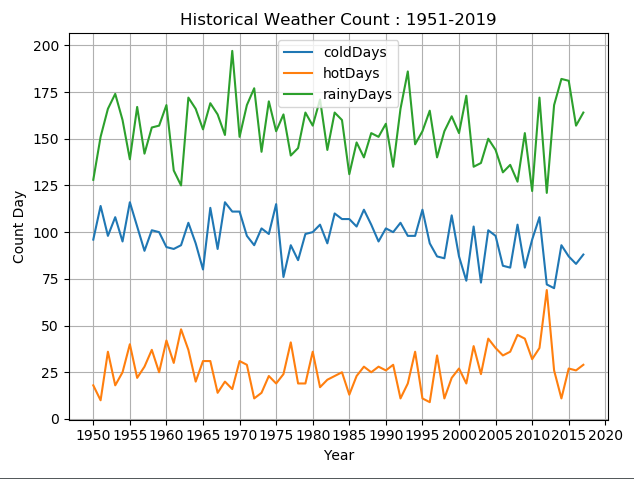


通过DataFrame的groupby()方法来将逐日数据按照年份进行分组，之后再使用count()方法对分组进行计数统计，产生的数据集即为我们需要的年统计数据。

有了数据之后，绘制线性图的方法也很简单：



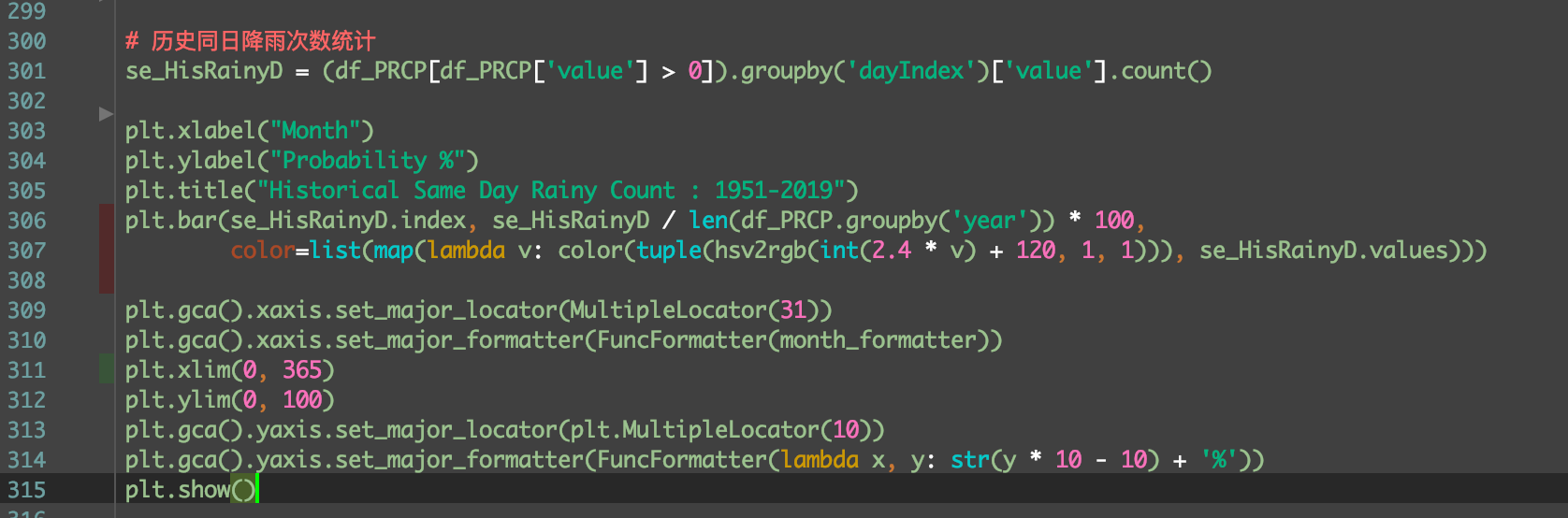
在同一张画布上调用三次plot()方法，分别绘制年降雨天数、年高温天数、年低温天数这三个数据集的逐年线性变化曲线：



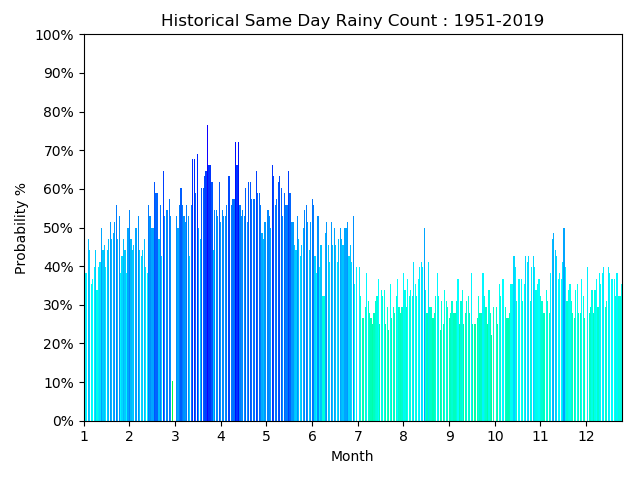
Step4:柱状图绘制

之后我们还需要再统计两类直观数据，即历史上同一天的事件发生频率（如某一天的历史降雨频率）以及所有年份中事件连续发生的最值和时间段（如每一年连续降雨最长天数以及发生的日期）。

首先是事件发生频率图，以降雨为例：



运行结果如下：



事件发生频率还可用于统计历史年份某一天严寒或者酷暑的概率，数据集的筛选方式不同但原理相同。

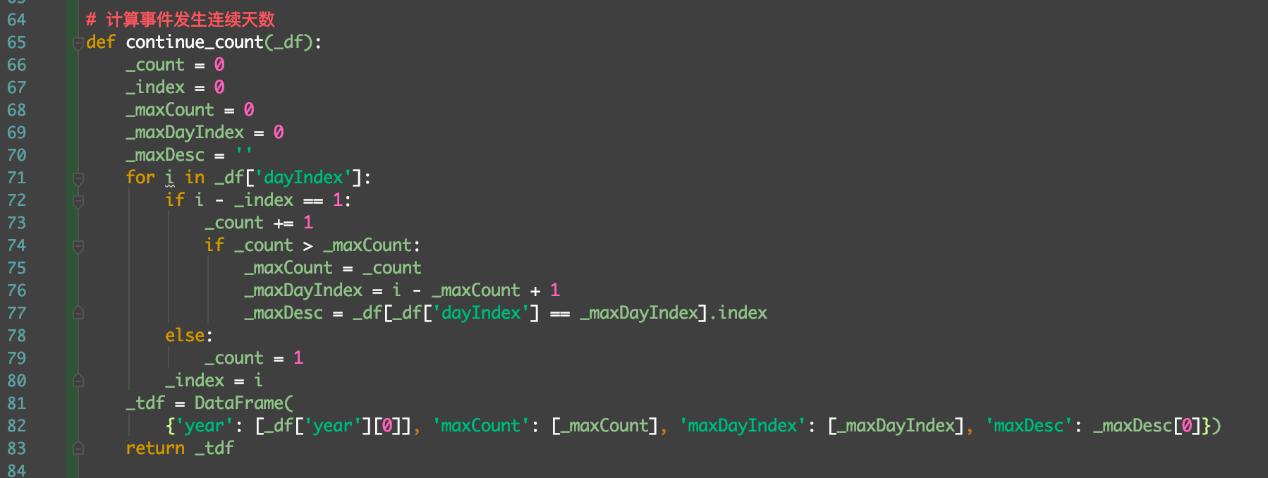
接下来就是我们最后的一类数据分析了，即事件连续发生天数逐年最值以及发生日期。这里仍旧分析降雨数据。我们需要计算出每一年连续下雨最长的天数以及从什么时候开始到什么时候结束。

代码如下：



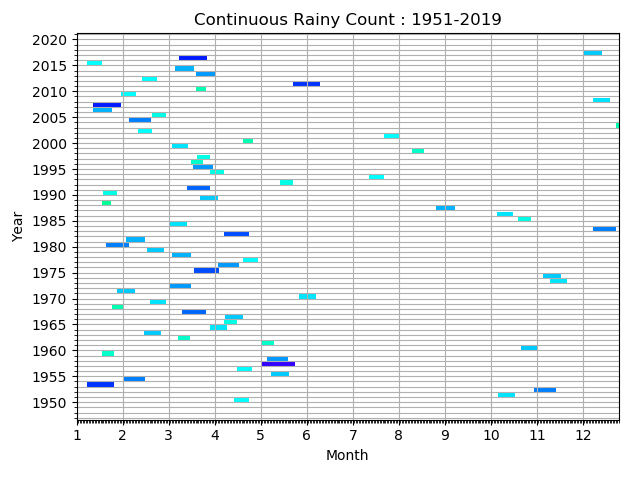
注意到这里用的是plt.barh()方法，即横向柱状图绘制，同时添加了left=？标签，用于对柱状图块进行位移（位移用来表示从哪一天开始，不设置的话默认都是从x=0开始）。

另外自定义了一个计算函数，稍微复杂一些所以不适合用lambda表达式来直接写：



基本思路就是循环结果集中的所有数据，遇到连续索引的日期就进行累加，当累加超过之前保存过的最大值时就覆盖掉之前的记录，所有结果循环完毕之后，最后留下的记录就是最大值记录。

最后的效果如图：



每一个柱状图块的长度表示了每一年的降雨最大天数，图块的位置表示了连续降雨的起始和结束时间。

编码写到这里，最核心的几个步骤就结束了，剩下的就是一些同类型的拓展统计需求，使用上述的数据集和方法基本上都能够实现，这里就不再拓展编码说明了。

最后，我们还需要再做一件事情——图表分析。

【分析篇】

首先要感谢能够看到这里或直接跳到这里的同学，可能大家比较感兴趣的也就是这一部分了。



基于上一篇的程序运算，我们一共生成了8张图：

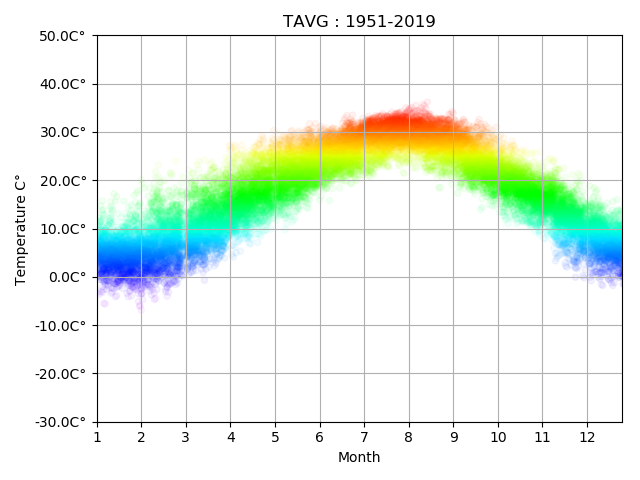


图1-长沙历史平均气温分布图

这是一张散点图，每一个点表示70年中某一天的平均气温，直接呈现了气温与年单位时间的对应关系，可以看到历史同比的平均气温分布跨度都比较大，不同年份同比最大甚至出现了20C°的差异。

但综合来看还是比较直观的，长沙从11月中旬开始进入低温模式（ TAVG<10C°），到次年的2月初开始逐步回温，直到3月中旬彻底结束长达4个月的寒冷天气。

也不排除有少数非常暖的年份，在2月初就早早迎来了春天。但醒醒吧，那只是少数。



总之，长沙的冬天真的就是一场噩梦。

而温度不是最主要的原因，湿度才是噩梦的主角，这在后面会重点说明。

而长沙的炎热天气对比之下就不是很长了，综合来看，从6月底气温逐步上升到30C°左右，经过7-8这两个月的盛夏之后，在9月初气温开始逐步回落。

之前有小伙伴问我什么时候来长沙玩比较合适，那么就气温来说，上图中出现绿色比较多的时间是比较宜居的。即3月中旬到5月中旬，9月中旬到11月中旬，合计是4个月的时间。

那剩下8个月在干嘛呢？剩下8个月都在一边哭唧唧的和天气做斗争，一边在等待这4个月。



接下来就气温情况还有一张图作为补充：

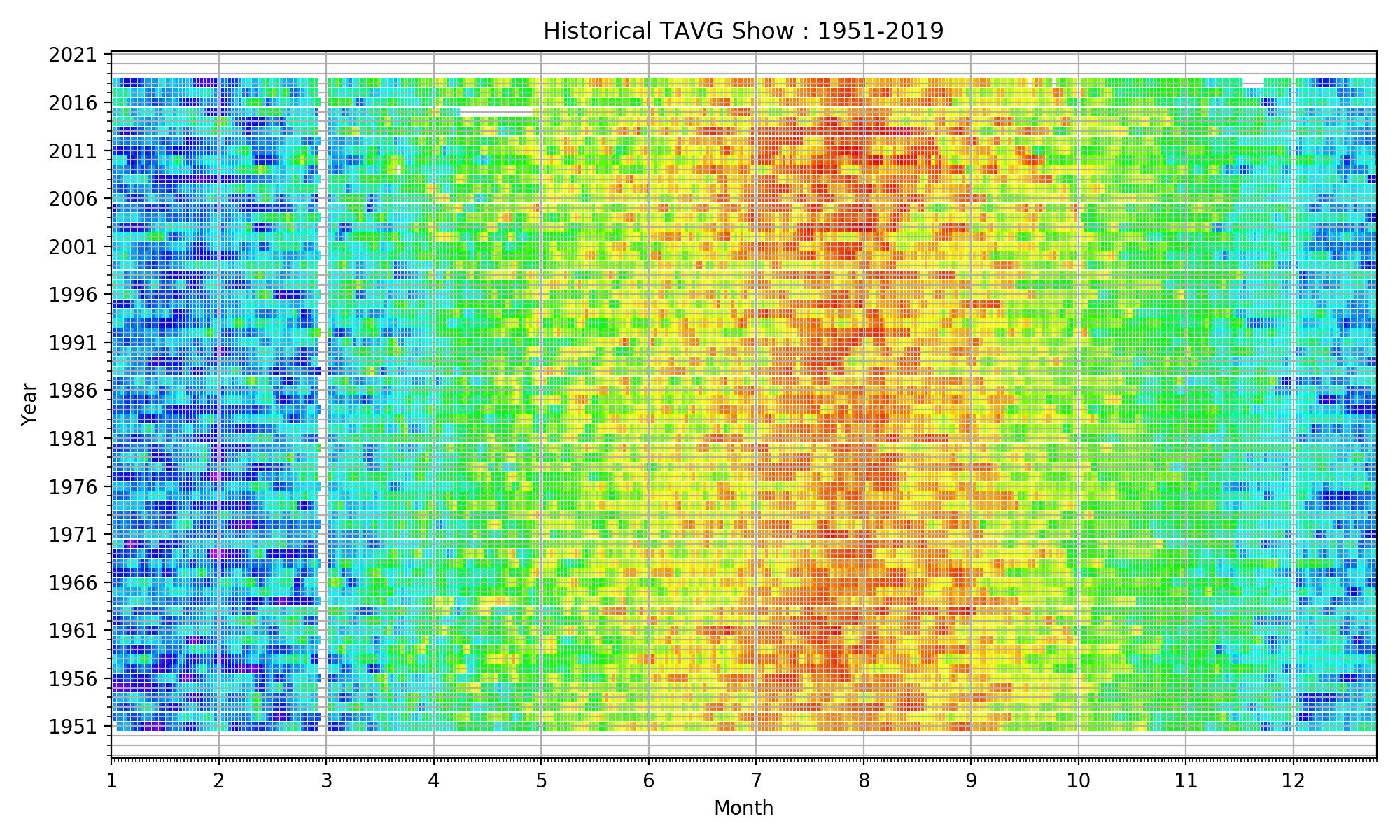


图2 -长沙历史气温分布图

从图中可以看到夏季连续高温的天气并不长，只是在2013年7月出现了一次连续一个半月的高温天气。

而连续的低温天气相对就比较漫长了，甚至在2012年出现了一次长达两个半月的超长低温天气，这还没有算上2011年入冬之后的连续低温。

所以抱怨长沙的夏天有多热的小伙伴，还是多思考一下接下来的冬天有多难熬吧…



说完温度接下来就要说最可怕的降雨情况了。冬天其实并不可怕，可怕的是冬天没有太阳还下着雨。



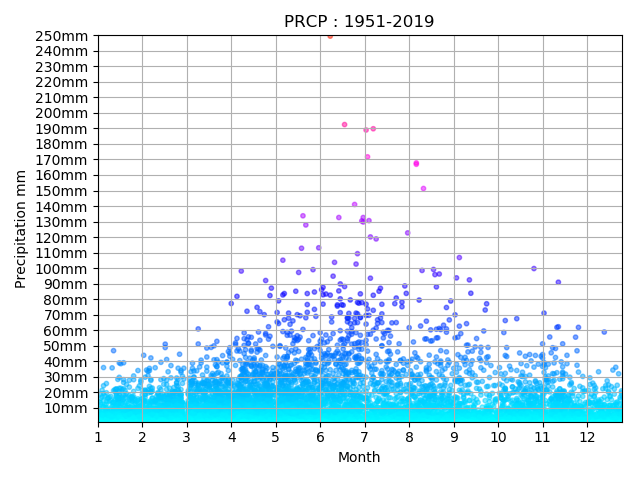


图3 – 长沙历史降雨量分布图

首先不要被这张图吓到，一片蓝色不代表一年365天都在下雨，而是表示这70年里，几月几号这一天有下过雨。

可以看到365天都有过下雨记录，肯定是不存在不下雨的某一个幸运日的。雨量随着夏季的到来呈上升趋势，在6月到达顶峰，接下来炎热的7月和8月降雨量相对开始减少。而其他时间也是有概率降雨的，这一点亚热带季风气候的城市都是如此，并不会有太大的区别。

重点是后面的两张图：

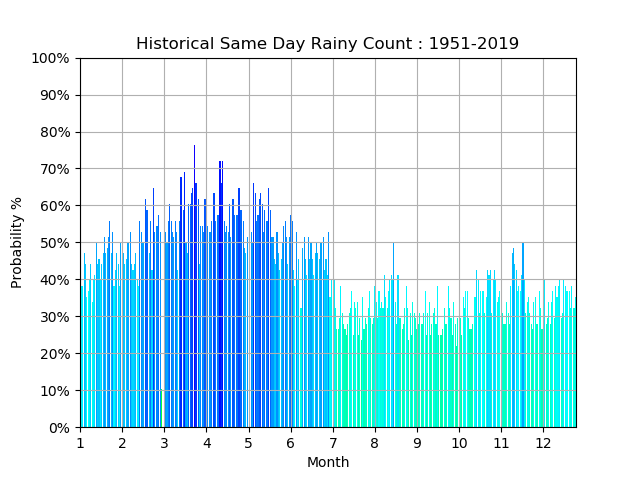


图4- 长沙历年降雨频率统计图

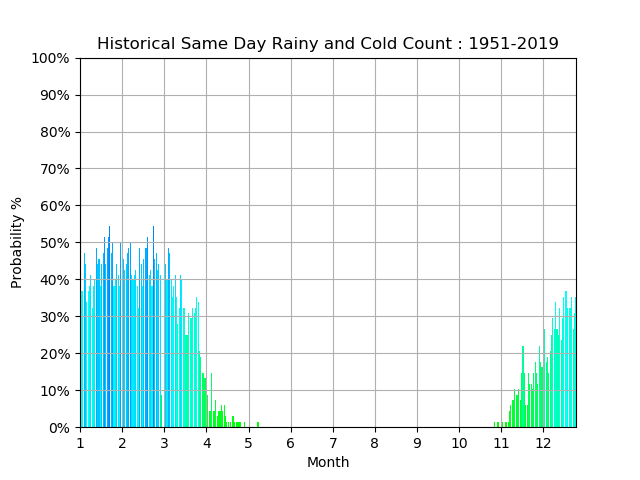


图5- 长沙历年湿冷天气频率统计图

这两张图很直观的展示了，历史同比日单位的降雨频率……



确实有一点绕，通俗说就是统计了这70年以来，某月某日出现过降雨的次数，以此来计算出这一天会下雨的概率。

从图4可以看到长沙常年降雨频率都在30%以上，从1月份开始上升到了40%，2月中下旬开始彻底爆发，超过了50%、60%，一直到7月盛夏才有所下降。

其实下雨就下吧，这也不是重点，重点是图5——出现降水同时气温又低于10C°的频率统计。可以看到，1、2两月这样的天气出现的频率均值达到了45%以上，所以这也就是最近这两个月这么难熬的原因了。

空气湿度长期在80%以上，空调加热一屋子的空气要比北方难太多了，即便是学北方用了暖气片也只能加热周围一圈的局部温度，靠在边上身体一边是热的一边却是凉的。。。



出门到外面，空气中无限接近0度又不结冰的水分子加速了体温的热传导，再厚实的棉袄也挨不住水分子的渗透，风再一吹加速一下热对流，那种绝望的感受只有长江以南一带的小伙伴才会懂。



而历史上连续下雨的天数可以参考下图：

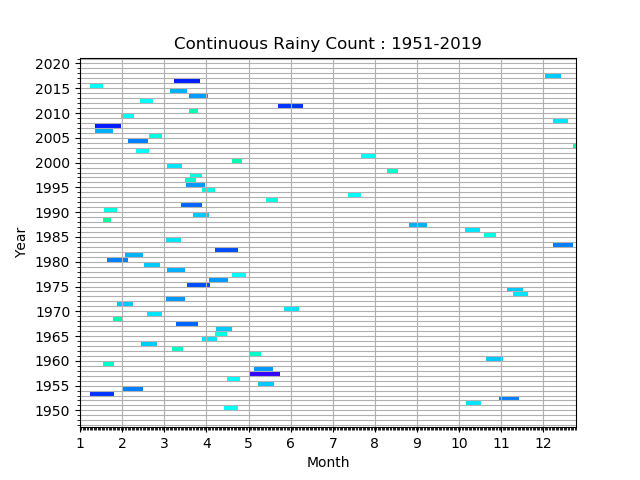


图6 – 长沙历史连续降雨分布图

可以看到并不是每年都有超长的时间一直连续下雨，大部分都会在10天以内结束，但是有可能出一天太阳接着下十天雨这样的情况，为了调查到底，于是有了图7

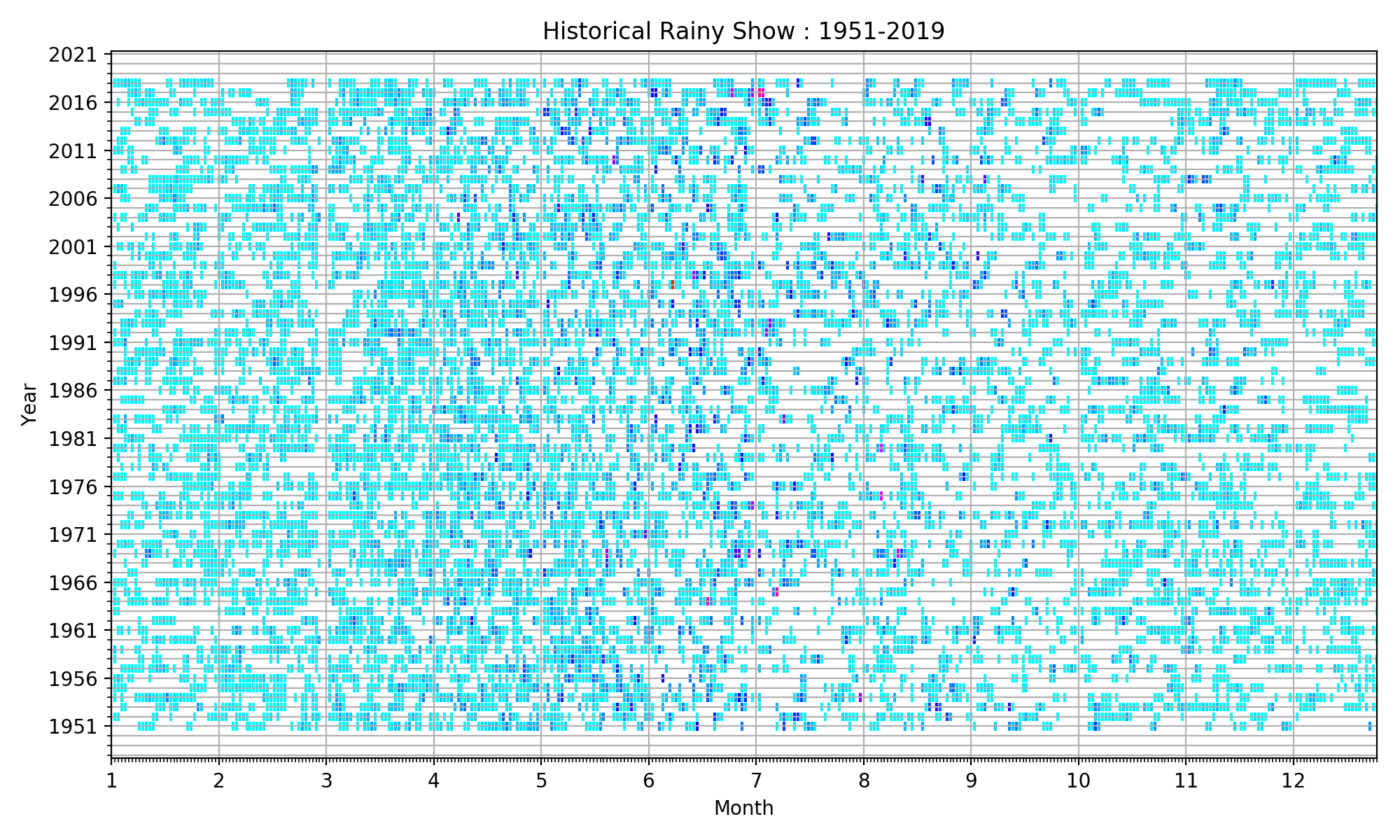


图7 – 长沙历史降水展示图

结果，当然是很绝望。



图中每一个蓝色的点都表示历史上当天有降雨记录，空白区域则表示当天没有降雨记录。综合来看每年接近一半的时间都在降雨，冬末至夏初的降雨尤为频繁。

最后我统计了一下长沙历史的降雨及高温 (大于30 C°)和低温（小于10 C°）的各年天数，如下图：

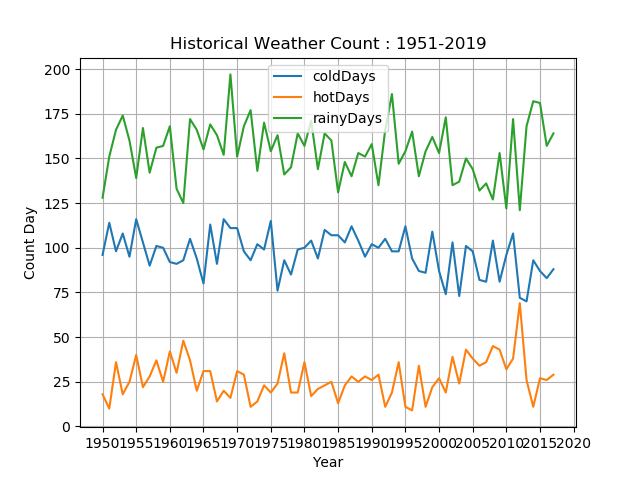


图8 – 长沙历史高低温及降雨天数统计图

可以看到，橙色、蓝色、绿色线条分别表示高温、低温和降雨天数。这些数据已经非常充分的说明了长沙是一个湿冷多过炎热的城市。

而我们需要再坚持坚持，熬过了这个二月还有一个三月，虽然四月也有不少雨水，但已经不会再冷了。70年的数据统计，几乎涵盖了许多人的一生所经历的种种气象，虽然反复无常，但还是起起落落总有一定的规律可循，而生活不就是这样吗。

最后，再奉上统计出来的长沙历史的气象极值：

历史最高气温：1953年8月13日，40.6 C°

历史最低气温：1972年2月9日，-11.3 C°

历史单日最大降水量：1997年6月7日，249.5mm

历史连续降雨天数记录：1958年5月1日至5月22日，连续降雨22天

本文关于数据的获取及分析的内容到此就全部结束，若有纰漏欢迎指正。如果有小伙伴想分析其他地区的天气情况，欢迎留言或私信我。

本文涉及的所有代码已开源并上传至GitHub，欢迎感兴趣的小伙伴一起继续开发优化。

END

BeiTown

2019-02-28