

# 新型冠状病毒数据分析演示

新型冠状病毒（2019-nCov）的疫情牵动着全世界人民的心，而理性地对待离不开数据和分析。为了让人民大众及时了解情况，很多网站都公布疫情的实时信息。比方说[丁香园疫情实时动态](https://ncov.dxy.cn/ncovh5/view/pneumonia) (<https://ncov.dxy.cn/ncovh5/view/pneumonia>)，[腾讯疫情实时追踪](https://news.qq.com/zt2020/page/feiyang.htm) (<https://news.qq.com/zt2020/page/feiyang.htm>)，[约翰霍普金斯实时新冠地图](https://gisanddata.maps.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6) (<https://gisanddata.maps.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6>) 等等。这些网站快速地为公众提供了信息，增加了透明度。但是如果读者希望对疫情有进一步的了解，这些网站就不够用了。比方说，如果你想得到过去十天湖北省确诊人数，那就只能从趋势图上作个估计了。再比方说，如果你想对比一下湖南、广东、浙江三省在过去十天的新增确诊人数，那么单凭网页数据也无能为力了。

为了取得可以供研究使用的数据，[DXY-2019-nCoV-Data](https://github.com/BlankerL/DXY-2019-nCoV-Data) (<https://github.com/BlankerL/DXY-2019-nCoV-Data>) 项目利用网络爬虫不断从网上抓取数据，更新并存成 CSV 格式。然而，这个 CSV 文件包含的是不同时刻网页上的信息片段，有的时候只有这几个城市，有的时候只有那几个城市，数据并不规整。

为了进一步方便用户进行研究，本项目 [nCov2019\\_analysis](https://github.com/jianxu305/nCov2019_analysis) ([https://github.com/jianxu305/nCov2019\\_analysis](https://github.com/jianxu305/nCov2019_analysis)) 提供了一些基本工具，把实时数据规整为每日数据，方便用户按时间、省份、城市等方法检索。同时，本项目还提供了基本的时间序列和横向分析作图函数，方便用户取得基本信息。

以下是基本使用方法演示：

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
import utils    # some convenient functions

%load_ext autoreload
%autoreload 2
```

## 1. 获取原始 CSV 数据

```
In [2]: data = utils.load_chinese_data()

Last update: 2020-02-13 06:55:35.291000
Data date range: 2020-01-24 to 2020-02-13
Number of rows in raw data: 35330
```

```
In [3]: data.head(3) # 查看数据形式
```

Out[3]:

	province_name	city_name	province_confirmed	province_suspected	province_cured	province
0	山西省	晋中	126	0	33	
1	山西省	运城	126	0	33	
2	山西省	太原	126	0	33	

## 2. 数据预处理

### 2.1 把实时数据整合成每日数据，以更新日期 *update\_date* 为索引

*aggDaily()* 函数内部作了一部分数据除错处理，加入了每日新增案例，以及英文（拼音）地名以方便检索

```
In [4]: daily_frm = utils.aggDaily(data)
```

```
In [5]: daily_frm.tail(3)
```

Out[5]:

	update_date	province_name	province_name_en	city_name	city_name_en	cum_confirmed
233	2020-02-13	黑龙江省	Helongjiang	鹤岗	Hegang	5
232	2020-02-13	黑龙江省	Helongjiang	黑河	Heihe	6
227	2020-02-13	黑龙江省	Helongjiang	齐齐哈尔	Qiqihaer	33

## 3. 数据查看

### 3.1 提取部分信息

用 **province\_name** 检索省级数据

```
In [6]: daily_frm[daily_frm['province_name'] == '广东省'].head()
```

Out[6]:

	update_date	province_name	province_name_en	city_name	city_name_en	cum_confirmed
34351	2020-01-24	广东省	Guangdong	中山	Zhongshan	1
34345	2020-01-24	广东省	Guangdong	佛山	Foshan	1
34346	2020-01-24	广东省	Guangdong	广州	Guangzhou	1
34347	2020-01-24	广东省	Guangdong	惠州	Huizhou	1
34343	2020-01-24	广东省	Guangdong	深圳	Shenzhen	1

用 **city\_name** 检索市级数据

```
In [7]: daily_frm[daily_frm['city_name'] == '武汉'].head()
```

Out[7]:

	update_date	province_name	province_name_en	city_name	city_name_en	cum_confirmed
34542	2020-01-24	湖北省	Hubei	武汉	Wuhan	49
33367	2020-01-25	湖北省	Hubei	武汉	Wuhan	57
32718	2020-01-26	湖北省	Hubei	武汉	Wuhan	61
31539	2020-01-27	湖北省	Hubei	武汉	Wuhan	69
30735	2020-01-28	湖北省	Hubei	武汉	Wuhan	159

也可以用 **province\_name\_en**, **city\_name\_en** 进行英文（拼音）检索

（请注意，因为可能存在不同城市的汉字对应相同的拼音，所以检索城市时请尽量使用汉字。如果使用拼音检索，请务必保证其对应汉字城市的唯一性）

```
In [8]: daily_frm[daily_frm['city_name_en'] == 'Guangzhou'].head()
```

Out[8]:

	update_date	province_name	province_name_en	city_name	city_name_en	cum_confirmed
34346	2020-01-24	广东省	Guangdong	广州	Guangzhou	1
34008	2020-01-25	广东省	Guangdong	广州	Guangzhou	1
32599	2020-01-26	广东省	Guangdong	广州	Guangzhou	1
31425	2020-01-27	广东省	Guangdong	广州	Guangzhou	4
30642	2020-01-28	广东省	Guangdong	广州	Guangzhou	5

用 `updateDate` 检索单日数据

```
In [9]: daily_frm[daily_frm['update_date'] == pd.to_datetime('2020-01-27')].head()
```

Out[9]:

	update_date	province_name	province_name_en	city_name	city_name_en	cum_confirmed
31576	2020-01-27	上海市	Shanghai	嘉定区	Jiadingqu	1
31566	2020-01-27	上海市	Shanghai	外地来沪人员	Non_Residence	1
31577	2020-01-27	上海市	Shanghai	奉贤区	Fengxian	1
31575	2020-01-27	上海市	Shanghai	宝山区	Baoshanqu	1
31570	2020-01-27	上海市	Shanghai	徐汇区	Xuhuiqu	1

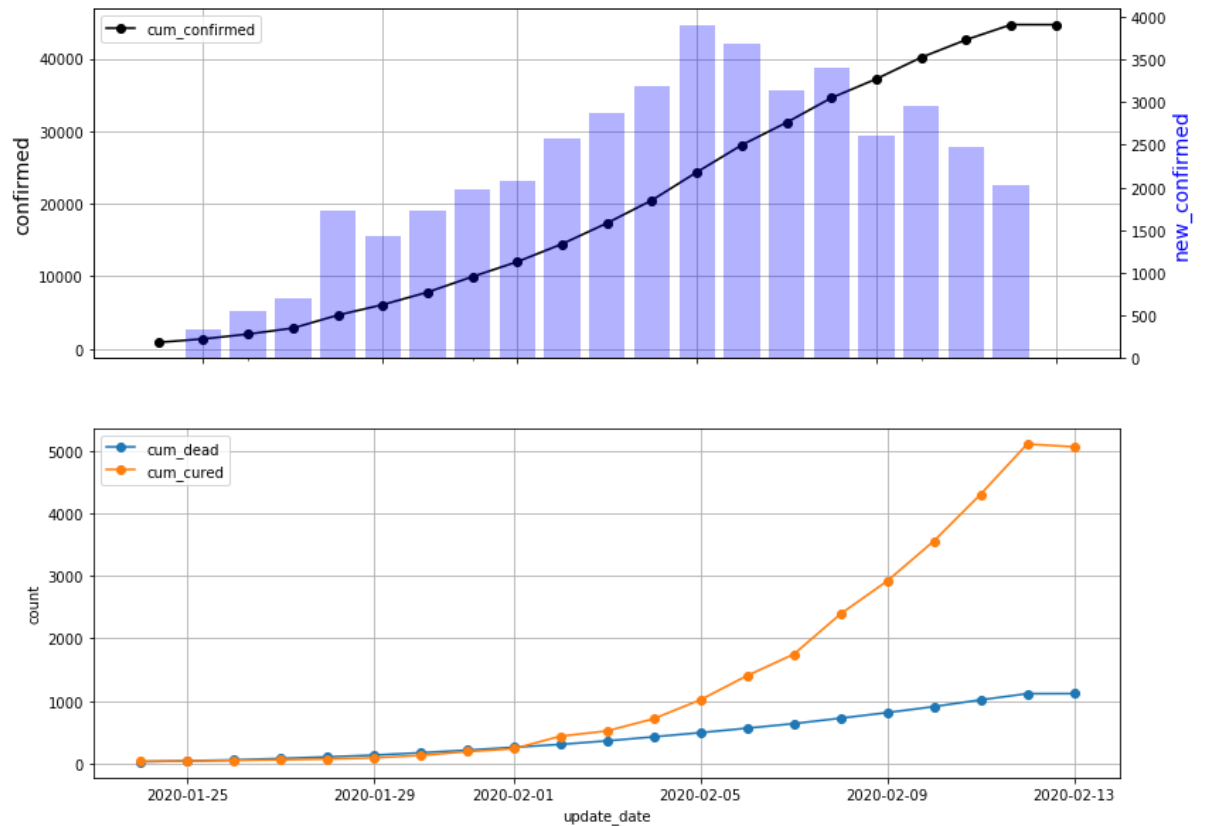
## 3.2 时序比较图 `utils.tsplot_conf_dead_cured()`

全国累计确诊、每日新增确诊、死亡、治愈时间序列图

```
In [10]: import matplotlib.font_manager as mfm
_FONT_PROP_ = mfm.FontProperties(fname='C:/Windows/Fonts/STFANGSO.TTF') # 如果想要在图中显示中文，必须指定字体文件
```

```
In [11]: fig = utils.tsplot_conf_dead_cured(daily_frm)
fig.suptitle('全国确诊、死亡、治愈人数', fontproperties=_FONT_PROP_, fontsize=18)
plt.show()
```

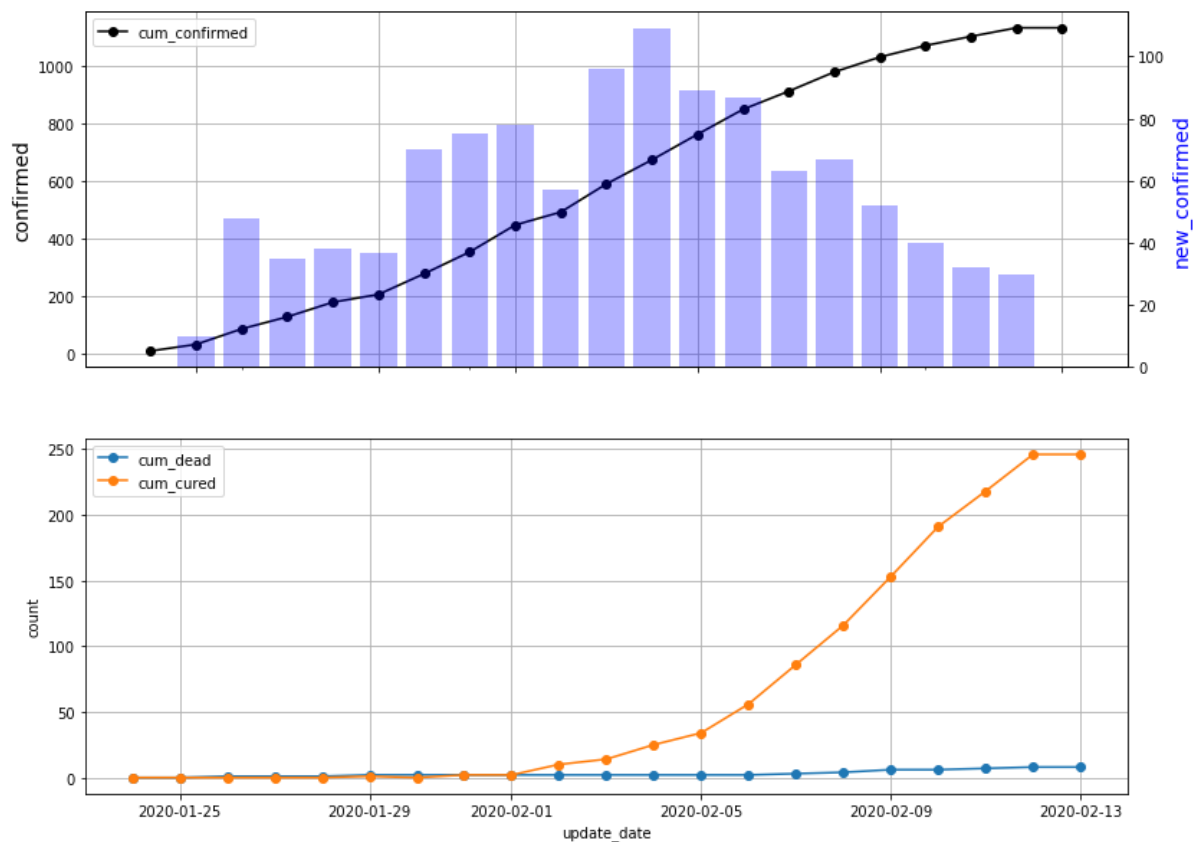
全国确诊、死亡、治愈人数



单个省份的时间序列也很容易，只要把想要的省份数据检索出来作为输入就可以了

```
In [12]: province = '河南省' # 输入你所要的省份
fig = utils.tsplot_conf_dead_cured(daily_frm[daily_frm['province_name'] == province])
fig.suptitle(province + ' 确诊、死亡、治愈人数', fontproperties=_FONT_PROP_, fontsize=18)
plt.show()
```

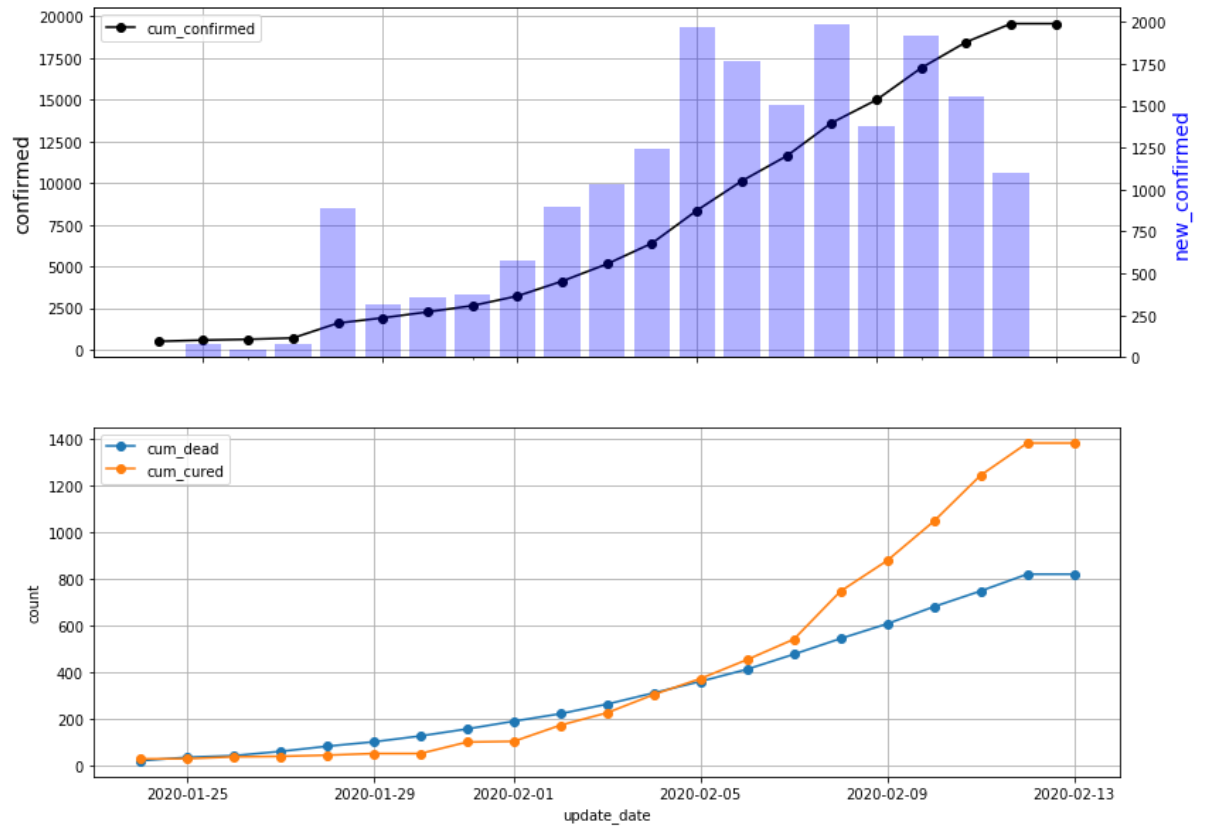
河南省 确诊、死亡、治愈人数



单个城市用法也是一样的, 还可以使用 **logy=True** 画指数图, 看人数是否指数增长

```
In [13]: city = '武汉'
fig = utils.tsplot_conf_dead_cured(daily_frm[daily_frm['city_name'] == city],
logy=False)
fig.suptitle(city + ' 确诊、死亡、治愈人数', fontproperties=_FONT_PROP_, fontsize=18)
plt.show()
```

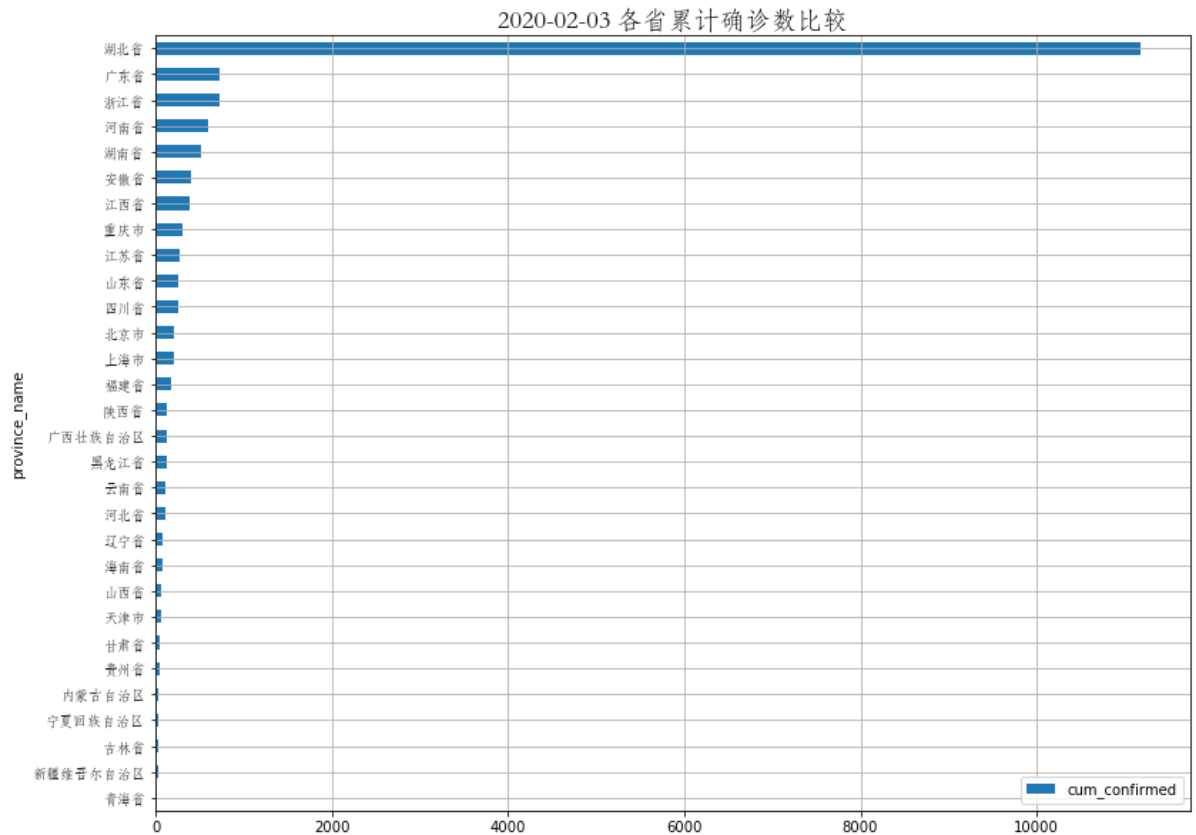
武汉 确诊、死亡、治愈人数



### 3.3 横向比较图 `utils.cross_sectional_bar()`

各省份在2月三号确诊数比较

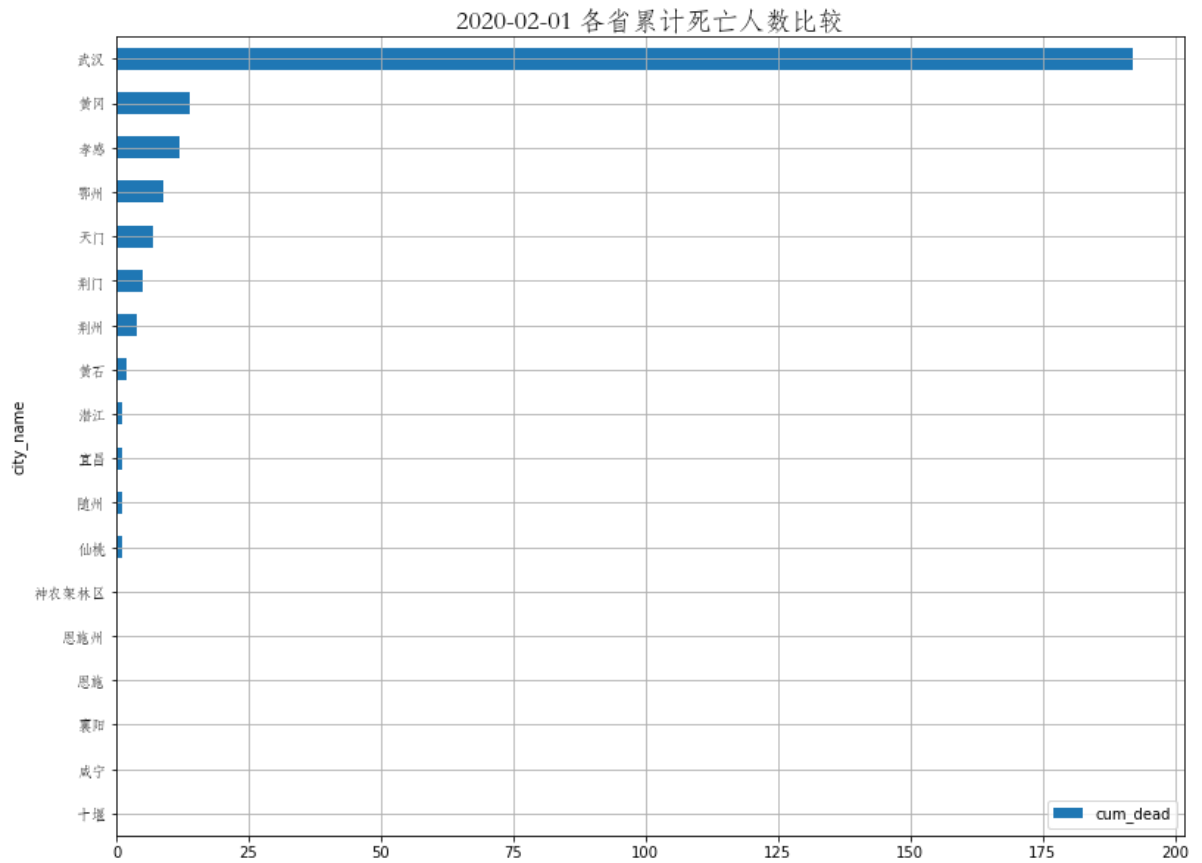
```
In [14]: ax = utils.cross_sectional_bar(daily_frm, '2020-02-03', col='cum_confirmed', groupby='province_name')
ax.set_title('2020-02-03 各省累计确诊数比较', fontproperties=_FONT_PROP_, fontsize=18)
plt.show()
```



湖北省各地2月1号死亡数比较



```
In [15]: ax = utils.cross_sectional_bar(daily_frm[daily_frm['province_name'] == '湖北省'], '2020-02-01', col='cum_dead', groupby='city_name')
ax.set_title('2020-02-01 各省累计死亡人数比较', fontproperties=_FONT_PROP_, font size=18)
plt.show()
```



全国2月5日新增确诊最多的十个城市（用 **largestN** 参数限制横条数目）

```
In [16]: ax = utils.cross_sectional_bar(daily_frm, '2020-02-05', col='new_confirmed', groupby='city_name', largestN=10)
ax.set_title('2020-02-05 全国各市新增确诊人数前十', fontproperties=_FONT_PROP_, fontsize=18)
plt.show()
```

