

PYTORCH



Microsoft

优酷

CNTK



Keras



Caffe2



Chainer



TensorFlow

theano



SymPy



scikit-image
image processing in python

mxnet



matplotlib
Version 3.2.1

seaborn

bokeh



PyMC3



NetworkX
Network Analysis in Python

NumPy

IP[y]:
IPython



NCAR/GEOSCAT

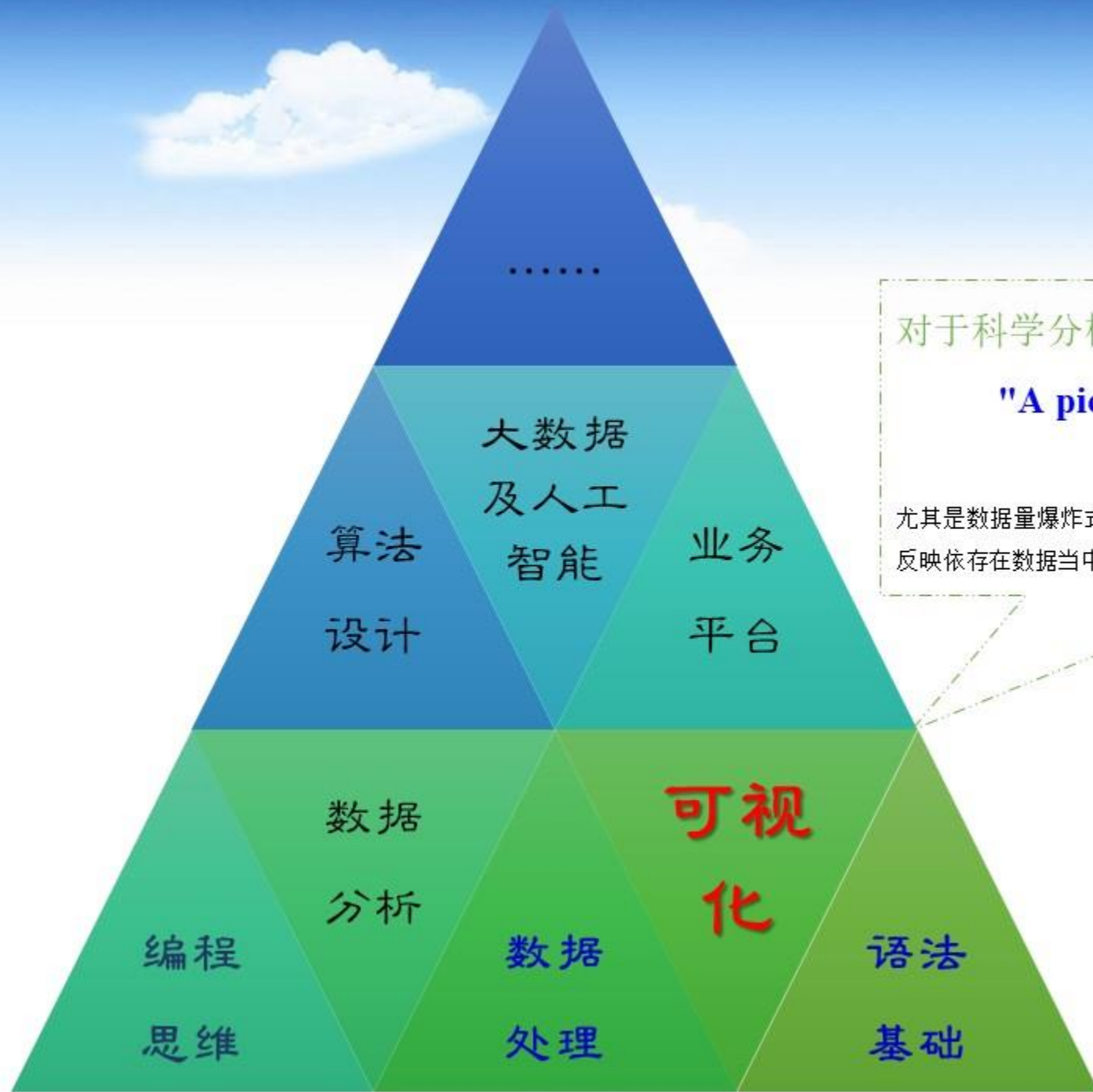


python™



Numba

QUANSIGHT



对于科学分析来说——

"A picture is worth a thousand words."

一张图胜过千言万语

尤其是数据量爆炸式增长的大数据时代，数据可视化能够更直观的呈现各类信息，反映依存在数据当中的客观逻辑或规律，进而高效提取、充分利用数据的有效信息。



Python语言在气象中应用

气象大数据：指在气象领域中，围绕智能预报和智慧服务，从数据采集、加工处理、预报预测、共享服务、存储归档等气象业务和科研工作各个环节所产生的各类数据总和。按照数据来源和产生方式可以分为：**气象观测数据、气象产品数据和互联网气象数据。**



互联网气象数据

包括第三方（研究机构、合作组织、企业等）、志愿者或个人搭建气象探测设备、智能终端搭载气象要素传感设备获取的探测数据，以及拍摄上传的天气状态、气象灾害灾情照片等社会化气象数据等。



气象产品数据


中国气象局通过发展完善观测数据质量控制、遥感数据处理、多源数据融合分析、资料同化与再分析等关键技术，建成了涵盖海洋、陆地及高空的三维业务和服务产品体系。



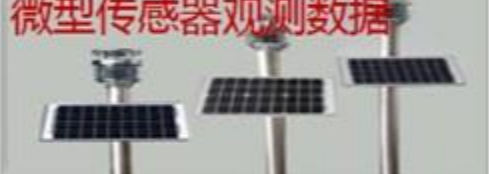
气象观测数据

按照气象观测规范，通过各种可能的观（探）测、遥测手段收集或加工处理得到的，来自地球大气圈及其它相邻圈层的，描述与大气及其变化有关的物理和化学状态的信息元素。

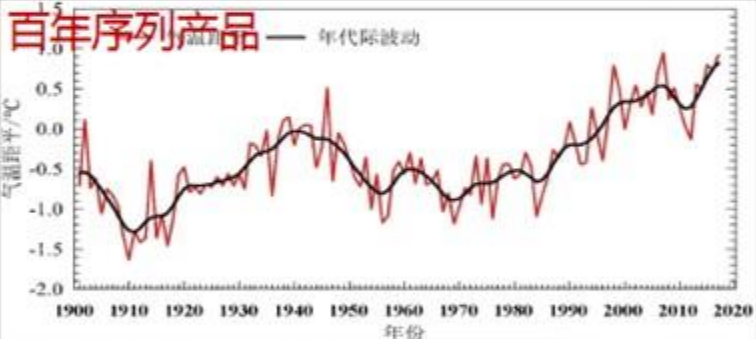
互联网下载数据



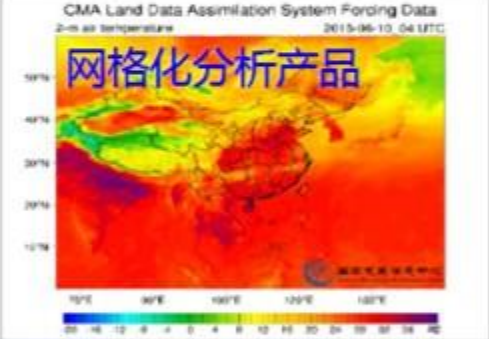
微型传感器观测数据






百年序列产品 — 年代际波动



网格化分析产品

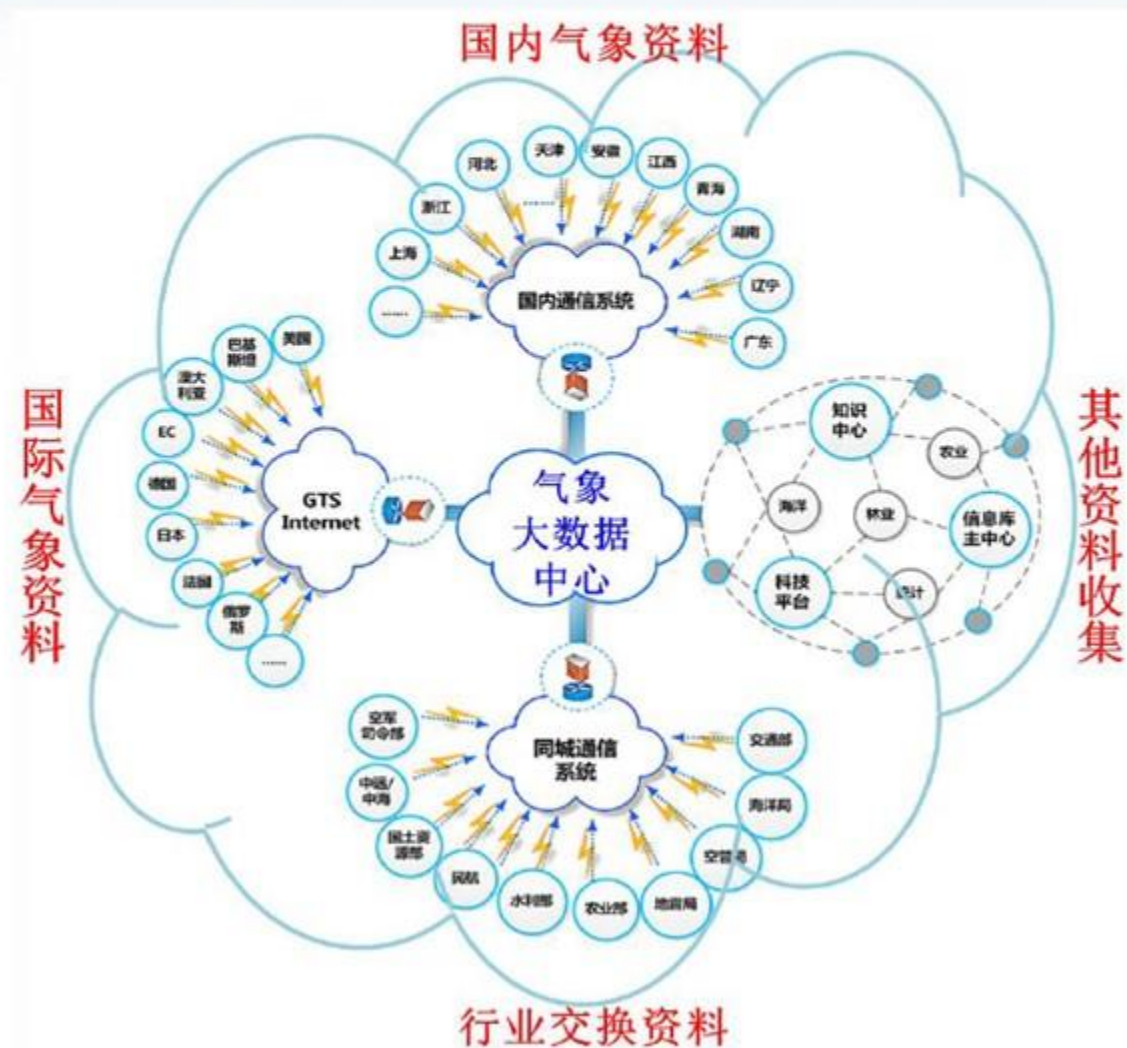


器测数据



气象大数据既有一般大数据的特点，也有气象学科数据特征，可以概括为**5个“V”**，即类型多（**Variety**）、体量大（**Volume**）、更新快（**Velocity**）、质量高（**Veracity**）、价值高（**Value**）等特征。

- ❑ **类型多**：从直接观测到遥感遥测；从大气物理变化到大气化学变化的观测，从大气圈到海洋等多圈层的观测。数据类型繁多，包括**结构化**气象观测数据，还包括图片、档案、音视频等**非结构化**数据以及图、文、数混编的灾情数据等**半结构化**数据。
- ❑ **体量大**：时间序列长，空间覆盖广；存量17PB，增量4PB/年。
- ❑ **更新快**：采样频率高，达分钟级、秒级。
- ❑ **质量高**：质量控制体系保障。
- ❑ **价值高**：行业联系紧密，行业融合价值高。



气象数据分类、格式

观测						模式		服务产品	再分析资料
地面	高空	雷达	卫星		其他	确定性预报	集合预报		
		Level2	FY-2	图像产品	台风	ECMWF	ECMWF-C3E	落区	NCEP_FNL
		Level3	FY-4A	数值产品	风廓线	GRAPES_GFS	GRAPES-EN	智能网格	ECMWF_ERA
			HMW-8	微波、闪电	GRAPES_MESO	指导预报	JAPAN
			极轨		环境、辐射	T639GDAS
客观	分析				海农灾考				CLDAS

"报文"BUFR > grib1/2 > netCDF > HDF|AWX > MICAPS(MDFS)

- ✓ 上述格式都是标准或通用格式，说明很多！ 也有其他格式
- ✓ 能用“一级数据”就别用“二手数据”。
- ✓ 格式很关键，知道格式才能读！！ 但根据不同格式读到数组里，更加关键！！！！

真空包装 vs 散称



大气是三维，甚至是四维的

但是学习认知、科学分析时，（绝大多数）却是从一维 / 二维入手

比如：降水随时间的日月年代变化

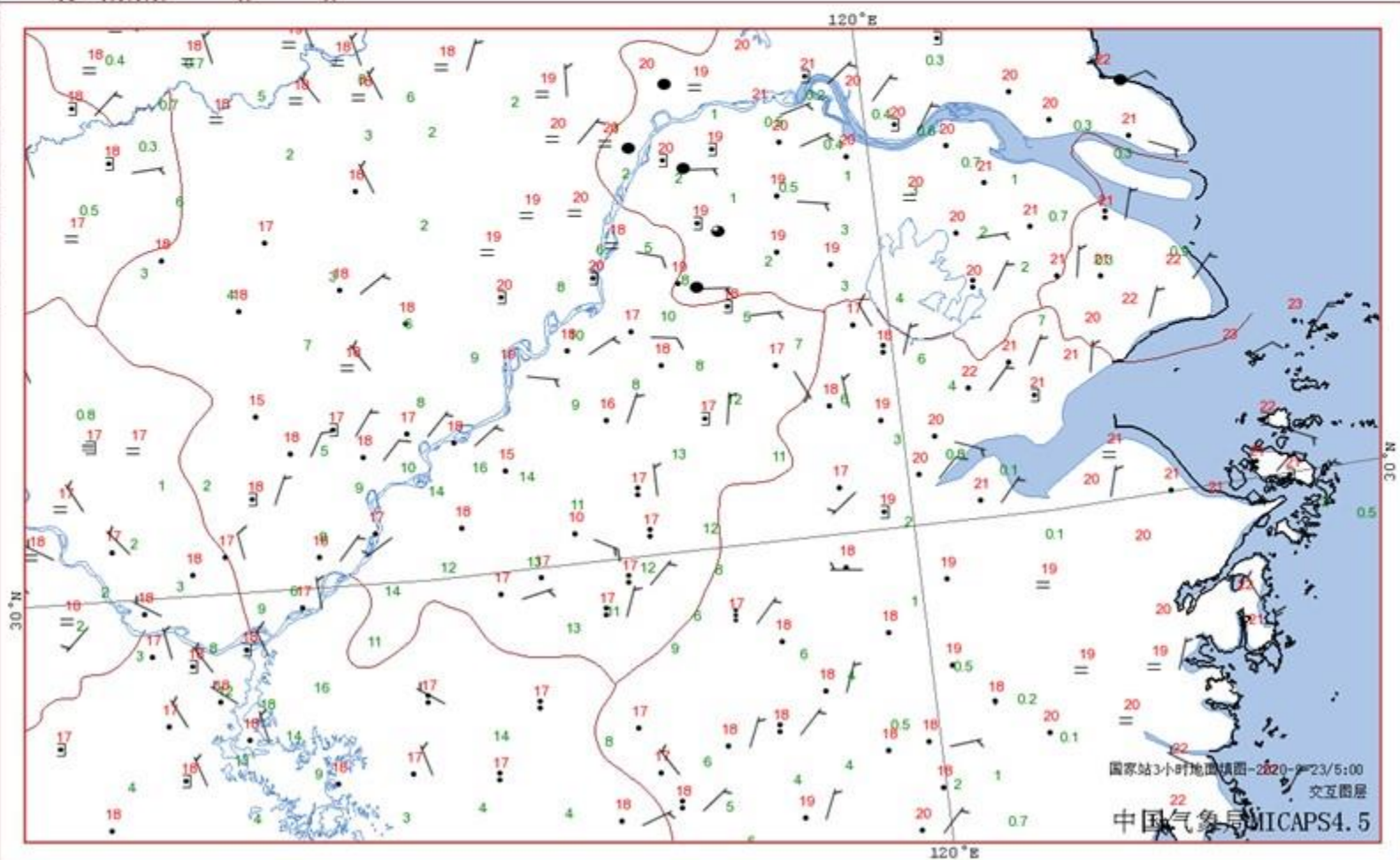
温度湿度随高度的变化……

再如：850hPa水汽通量/假相当位温的水平分布

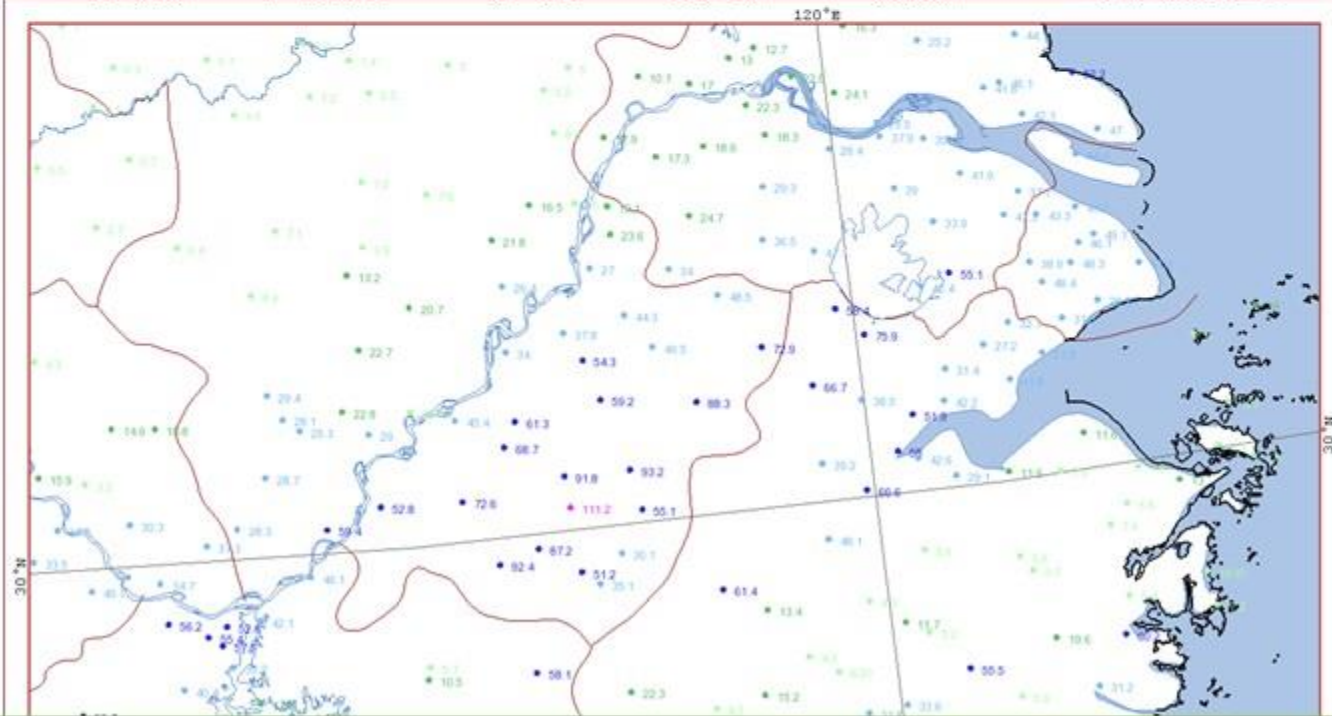
高空槽两侧垂直速度随高度的分布……

1	diamond 1	20年09月23日02时地面填图
2	20 09 23 02 5997	
3	64700	15.03 12.13 295 32 3 0 0 87 9999 0 0
4	0 30	0 9999 24.3 10.0 0 28.5 20 12 1 2 9999 9999
5	8594 -22.94	16.73 55 16 2 90 4 144 9999 0 0
6	0 32	2 300 22.5 25.0 0 30.3 20 10 1 2 9999 9999
7	8589 -23.51	14.90 35 16 2 50 4 140 9999 0 0
8	0 32	2 300 24.5 30.0 0 28.7 20 10 1 2 9999 9999
9	8583 -24.99	16.88 63 32 4 250 3 151 9999 0 0
10	0 38	4 300 24.1 15.0 0 29.5 20 10 1 2 9999 9999
11	41194 55.33	25.25 5 16 0 320 5 34 9999 0 0
12	0 9999	0 9999 25.0 10.0 0 33.1 9999 9999 1 2 9999 9999
13	41196 55.52	25.33 33 2 0 280 2 36 9999 0 0
14	0 9999	0 9999 25.3 10.0 0 32 2 36 9999 0 0
15	41217 54.65	24.43 27 16 0 50 2 36 9999 0 0
16	9999 9999	0 9999 27.1 10.0 0 31 2 36 9999 0 0
17	81225 -55.19	5.45 15 32 6 130 3 36 9999 0 0
18	0 32	4 600 21.2 20.0 0 35 2 36 9999 0 0
19	81200 -55.17	5.82 7 32 5 110 2 36 9999 0 0
20	0 32	4 600 21.8 20.0 0 34 2 36 9999 0 0
21	81202 -57.02	5.95 2 32 4 60 5 36 9999 0 0
22	0 32	1 600 24.0 20.0 0 31 2 36 9999 0 0

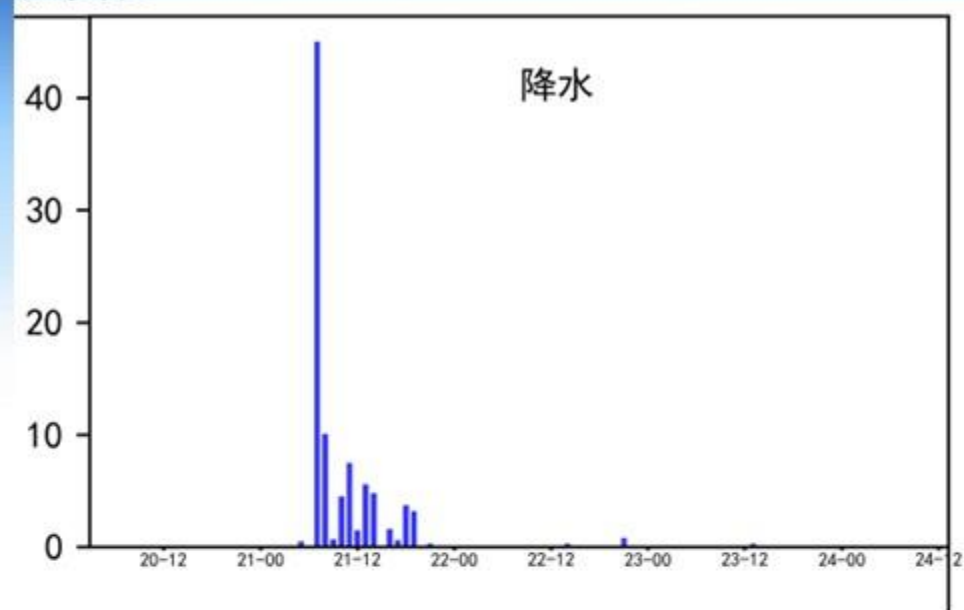
地面天气图



站号	经度	纬度	海拔高度	站点级别	观测值	站名
57345	109.62	31.4	337.8	9999	9.9	巫溪
55299	92.07	31.48	4507	9999	2.2	那曲
57348	109.48	31.03	299.8	9999	12.4	奉节
57349	109.87	31.08	606.8	9999	9	巫山
57355	110.35	31.05	337.5	9999	8.7	巴东
57358	110.97	30.83	295.5	9999	12.1	秭归
57359	110.73	31.35	336.8	9999	5.1	兴山
57361	111.27	31.88	327.7	9999	0.2	保康
57362	110.66	31.75	325.2	0000	6.2	神农架



站点24小时降水



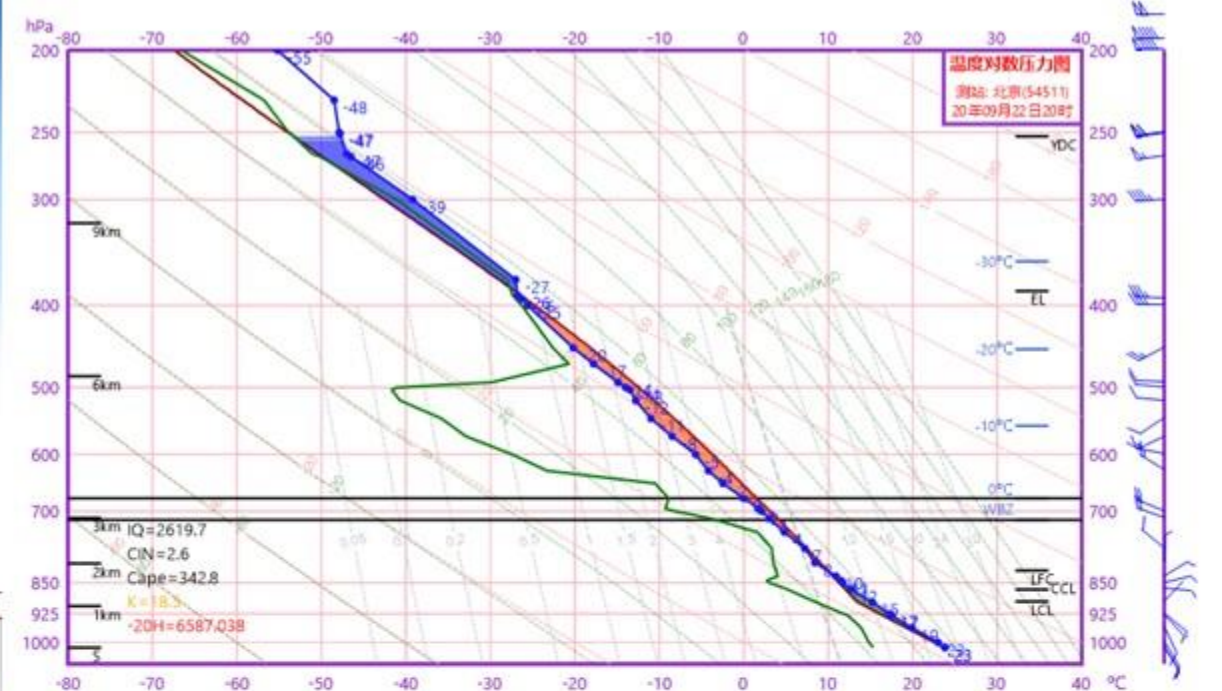
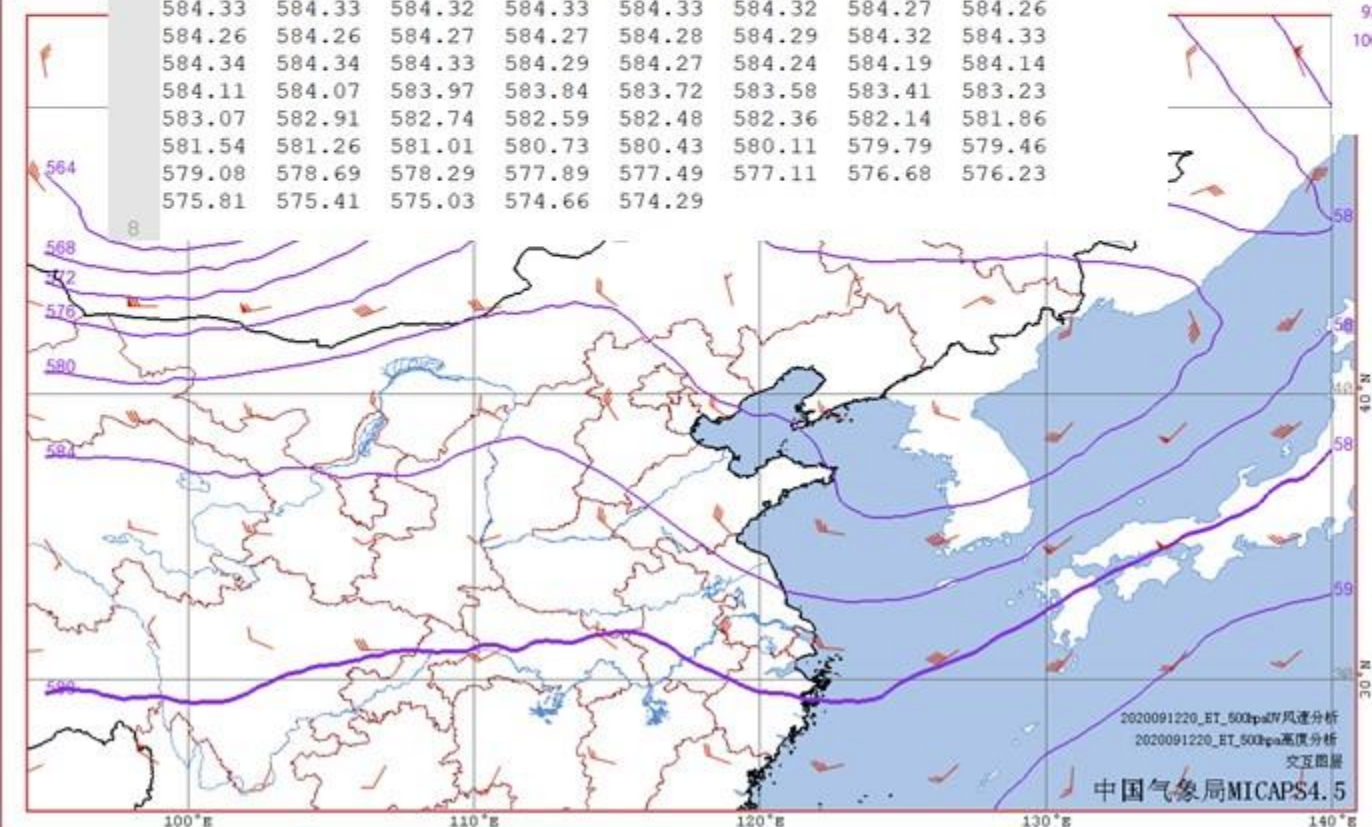
Time	Rain
2019/3/21 4:00	9999
2019/3/21 5:00	0.5
2019/3/21 6:00	0.1
2019/3/21 7:00	45
2019/3/21 8:00	10.1
2019/3/21 9:00	0.7
2019/3/21 10:00	4.5
2019/3/21 11:00	7.5
2019/3/21 12:00	1.5
2019/3/21 13:00	5.6
2019/3/21 14:00	4.8
2019/3/21 15:00	0.1
2019/3/21 16:00	1.6
2019/3/21 17:00	0.6
2019/3/21 18:00	3.7
2019/3/21 19:00	3.2
2019/3/21 20:00	9999

diamond 4 2020091220_ET_500hpa高度分析

2020 09 12 20 0 500

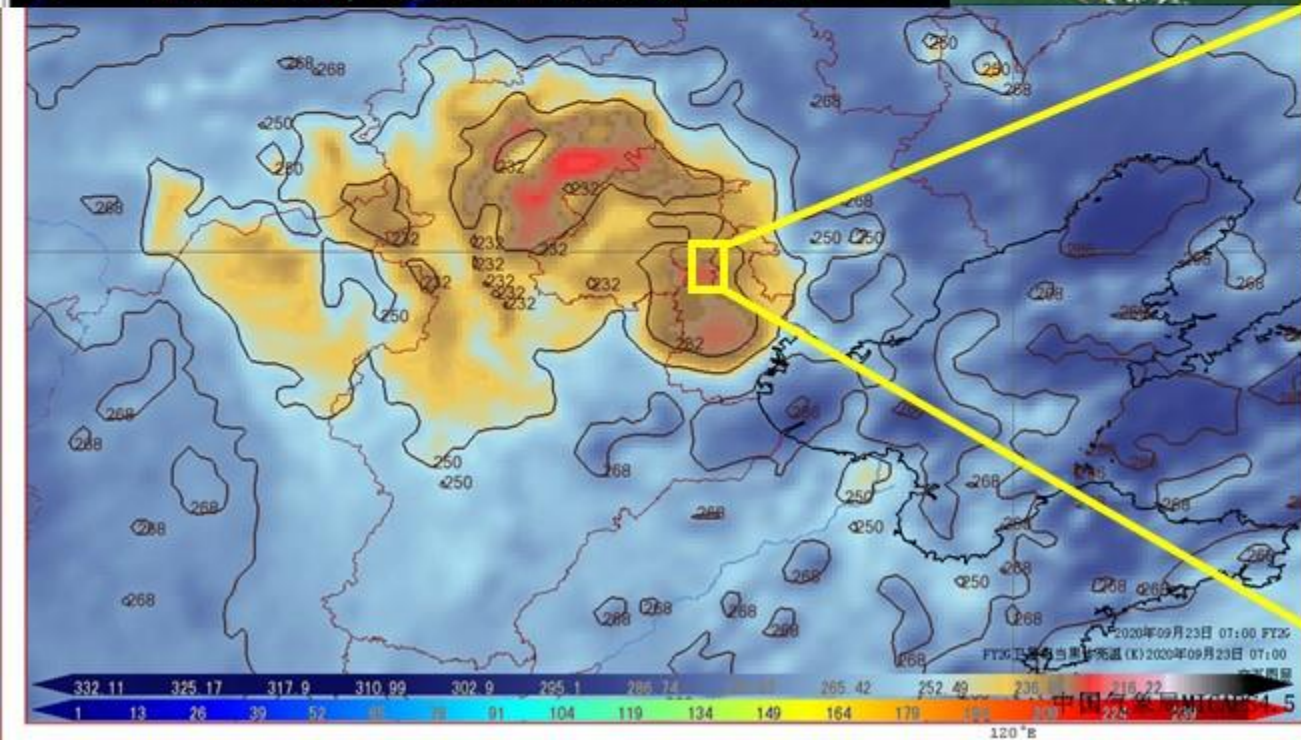
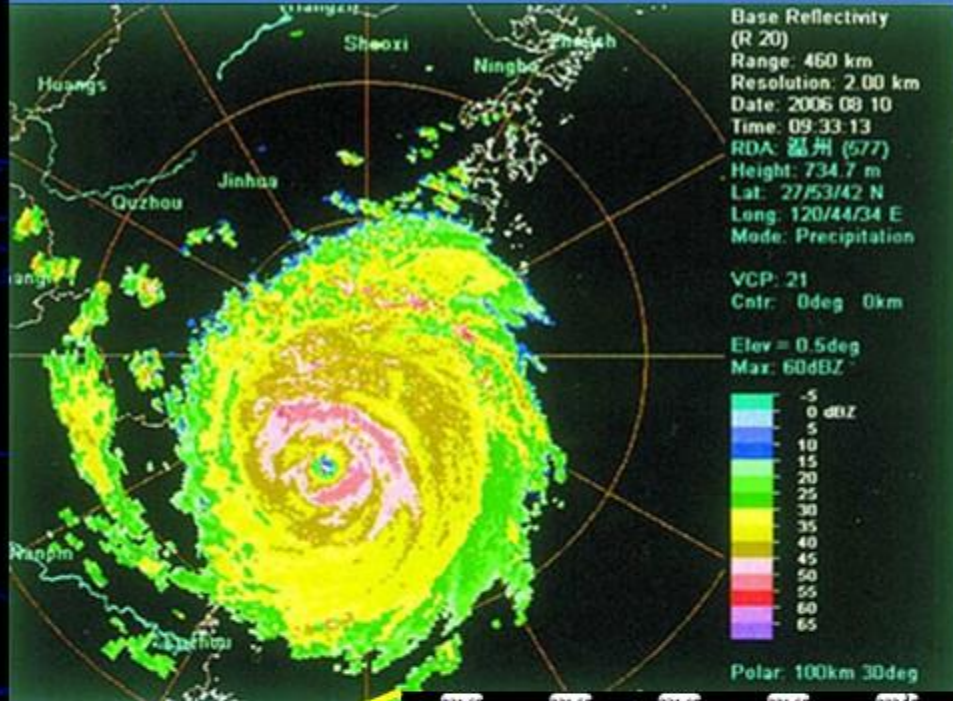
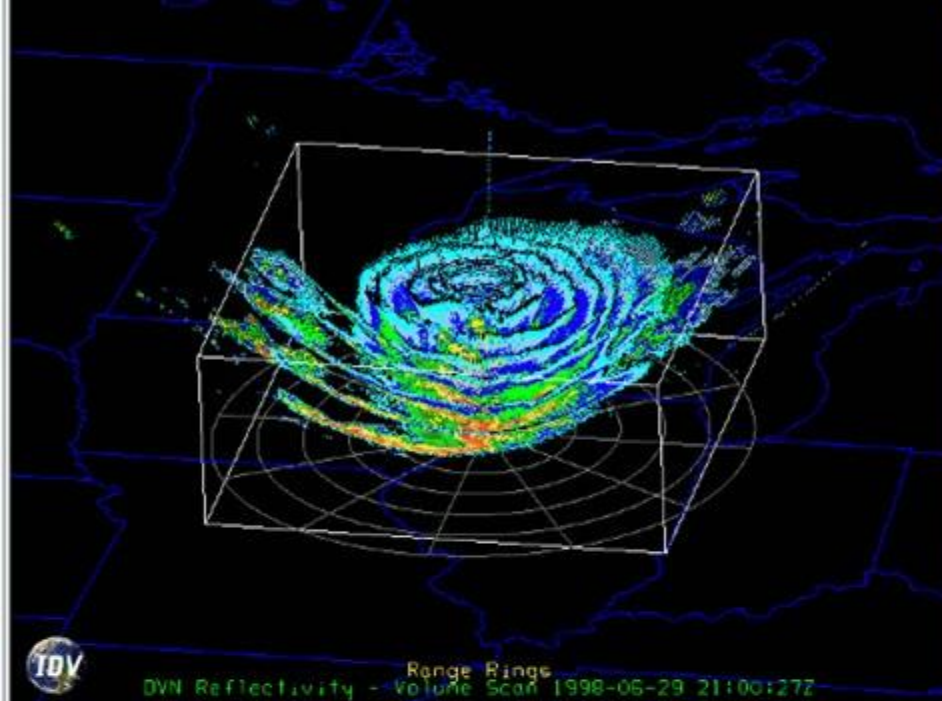
0.250000 -0.250000 95.000000 140.000000 55.000000 15.000000 181 161
 4.000000 480.000000 604.000000 1.000000 0.000000

564.24	564.04	563.81	563.59	563.38	563.18	562.96	562.73
562.54	562.36	562.19	562.06	561.96	561.86	561.73	561.61
561.51	561.47	561.44	561.44	561.46	561.46	561.43	561.43
561.44	561.44	561.42	561.44	561.57	561.77	562.01	562.27
562.57	562.88	563.22	563.58	563.96	564.31	564.66	565.01
565.34	565.68	566.01	566.31	566.61	566.88	567.17	567.47
567.77	568.07	568.37	568.68	569.01	569.31	569.59	569.86
570.14	570.46	570.78	571.11	571.41	571.69	571.96	572.22
572.47	572.68	572.88	573.11	573.41	573.76	574.11	574.44
574.78	575.13	575.48	575.78	576.07	576.36	576.64	576.92
577.19	577.47	577.76	578.06	578.36	578.64	578.92	579.19
579.47	579.73	579.97	580.22	580.47	580.72	580.96	581.18
581.39	581.61	581.84	582.07	582.27	582.46	582.66	582.84
583.01	583.17	583.33	583.46	583.57	583.67	583.78	583.89
583.99	584.06	584.12	584.19	584.26	584.29	584.31	584.32
584.33	584.33	584.32	584.33	584.33	584.32	584.27	584.26
584.26	584.26	584.27	584.27	584.28	584.29	584.32	584.33
584.34	584.34	584.33	584.29	584.27	584.24	584.19	584.14
584.11	584.07	583.97	583.84	583.72	583.58	583.41	583.23
583.07	582.91	582.74	582.59	582.48	582.36	582.14	581.86
581.54	581.26	581.01	580.73	580.43	580.11	579.79	579.46
579.08	578.69	578.29	577.89	577.49	577.11	576.68	576.23
575.81	575.41	575.03	574.66	574.29			

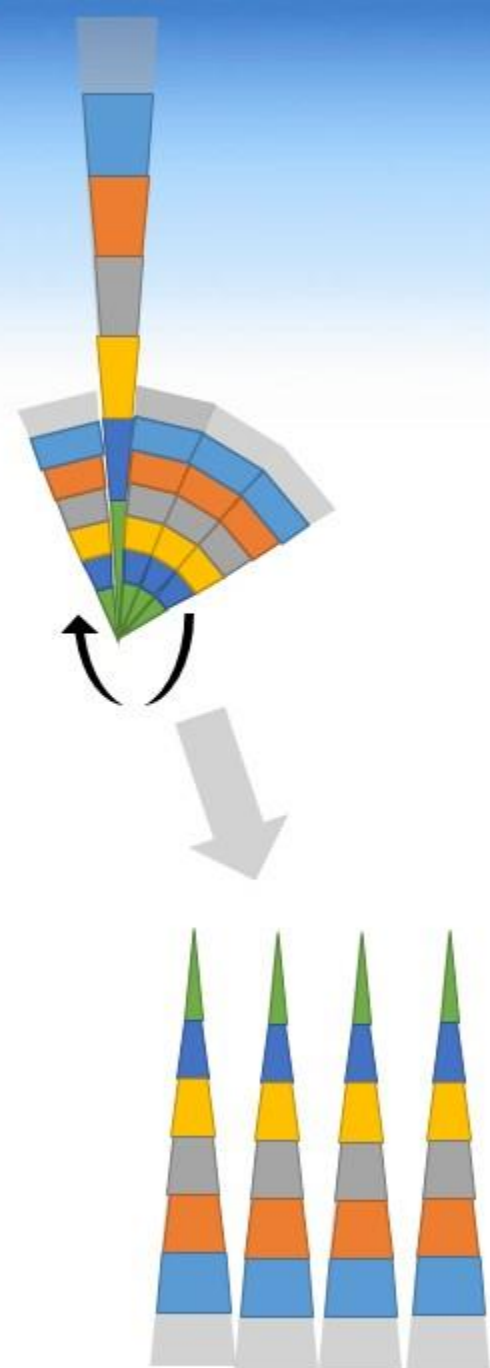
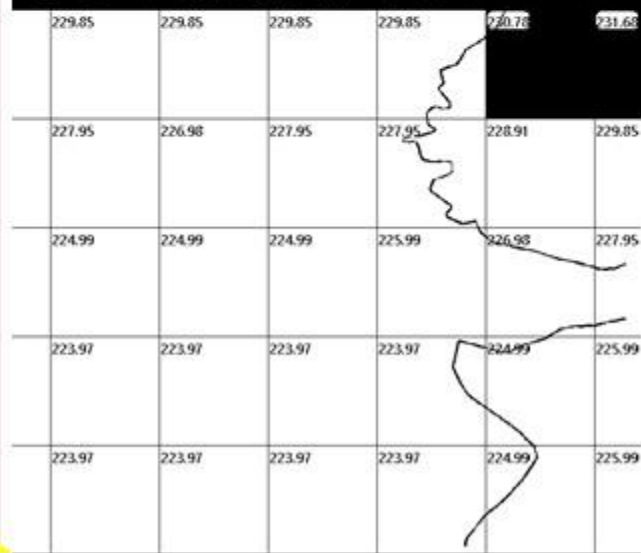


北京(54511)站各层物理量表

层号	层次	温度	露点	温度露点差	高度	风向	风速	比湿	饱和比湿	相对湿度	水汽压	凝结函数	位置	假相当位置	饱和假相当位置	假相当位置	静力总压
1	1014	23.8	15.3	8.5	-	148	1.6	10.74	18.3	59	17.4	4.57	22.62	53.19	74.6	17.75	23.8
2	1013.7	23.8	15.3	8.5	32	148	1.5	10.74	18.3	59	17.4	4.58	22.65	53.23	74.64	17.75	24.12
3	1000	23	14.8	8.2	151	147	3	10.55	17.67	60	16.85	4.56	23	53.06	73.26	17.75	24.51
4	961	19.9	13.9	6	500	153	4	10.35	15.19	68	15.89	4.68	23.25	52.8	66.47	17.64	24.9
5	929	17.5	12.3	5.2	795	137	1	9.64	13.51	72	14.31	4.55	23.69	51.31	62.21	17.17	25.45
6	925	17.3	11.8	5.5	824	128	1	9.37	13.4	70	13.85	4.45	23.85	50.72	62.08	17.05	25.54
7	896	15.2	8.5	6.7	1093	45	0	7.75	12.09	64	11.11	3.87	24.4	46.77	59.01	15.76	26.13
8	865	12.7	5.1	7.6	1391	94	1	6.34	10.64	60	8.79	3.34	24.81	43.29	55.37	14.59	26.61
9	850	11.7	3	8.7	1539	78	1	5.57	10.13	55	7.58	3.02	25.25	41.58	54.44	14	27.09
10	845	11.5	2.7	8.8	-	-	-	5.48	10.06	55	7.42	2.99	25.55	41.65	54.56	14	27.43
11	835	10.9	4	6.9	1701	61	1	6.08	9.78	62	8.14	3.34	25.93	43.77	54.21	14.82	27.91
12	805	8.4	3.5	4.9	2023	3	1	6.09	8.57	71	7.86	3.48	26.42	44.31	51.32	14.94	28.63
13	774	7.2	3.3	3.9	2344	310	4	6.25	8.21	76	7.75	3.71	26.51	47.01	52.61	15.88	30.64
14	741	4.8	1.6	3.2	-	-	-	5.78	7.26	80	6.86	3.62	29.68	46.92	51.16	15.76	31.71
15	714	3	-3.9	6.9	2987	290	7	3.85	6.63	58	4.41	2.58	30.93	42.65	50.72	14.36	32.87
16	700	2.1	-8.1	10.2	3136	291	7	2.73	6.34	43	3.07	1.91	31.66	40.12	50.68	13.54	33.46
17	695	1.8	-9.3	11.1	-	-	-	2.48	6.25	40	2.76	1.76	31.95	39.67	50.73	13.42	33.75
18	676	0	-8.9	8.9	3422	-	-	2.64	5.64	47	2.86	1.92	32.37	40.58	49.41	13.65	34.22
19	649	-2.5	-10.5	8	-	-	-	2.38	4.77	50	2.48	1.82	33.12	40.6	47.65	13.65	34.93
20	627	-4.2	-23.3	19.1	-	303	6	0.74	4.28	17	0.75	0.63	34.22	36.76	47.35	12.25	35.94
21	600	-5.7	-27.4	21.7	4360	281	5	0.51	3.93	13	0.49	0.47	36.37	36.23	48.59	12.83	37.9



北京时: 2020年09月23日 15时00分
通道: 1 [长波红外通道 10.3-11.5]



The background of the image is a clear blue sky with two fluffy white clouds. One cloud is in the upper left, and the other is slightly lower and to the right. The text is centered in the lower half of the image.

万数皆“表”

Python在气象中应用



——我是EXCEL⁺ & SPSS_{mini}

李开元

OLDLee



N-D labeled arrays and datasets



快速、灵活和富有表现力的数据结构
旨在使处理“关系”或“标签”数据既简单又直观
relational or labeled

Pandas数据处理Handle

1. Series与Dataframe

Data Structures

2. 创建、输入、输出

I/O

3. 索引、切片、筛选、重置

Indexing & Selecting

4. 合并、拼接、追加、删除_{-去重}

Merge, Concat, Append, Join, Drop_{-duplicate}

5. 排序、分组、汇总

Sort, Groupby

6. 计算、统计

Covariance\Correlation\rolling...

1. Series与Dataframe

标签label
键名key

索引index
列名columns

```
import pandas as pd
```

Series是一个一维的标记数组，可以保存任何数据类型（整数、字符串、浮点数、Python对象等）。
轴**标签**axis **labels**统称为**索引index**。

```
s = pd.Series([ 3, -5, 7, 4],index=[ 'a', 'b', 'c', 'd']) #pd.Series(data, index=index)
```

```
s1= pd.Series({'b': 'bb', 'a': 'aa', 'c': 'cc'})
```

```
s2= pd.Series(5.,index=[ 'a', 'b', 'c', 'd'])
```

```
s3= pd.Series(np.random.randn(5))
```

```
In [4]: s
```

```
Out[4]:
```

a	3
b	-5
c	7
d	4

```
dtype: int64
```

```
In [5]: s1
```

```
Out[5]:
```

b	bb
a	aa
c	cc

```
dtype: object
```

```
In [6]: s2
```

```
Out[6]:
```

a	5.0
b	5.0
c	5.0
d	5.0

```
dtype: float64
```

```
In [8]: s3
```

```
Out[8]:
```

0	1.392234
1	-1.020297
2	-0.393115
3	-1.281453
4	-1.509906

```
dtype: float64
```


1. Series与Dataframe

基础操作索引、切片、筛选

#Series is dict-like

```
s['a']
```

```
s.a
```

#Series is ndarray-like

```
s[0]
```

```
s[[0,2]]
```

```
s[1:2]
```

如果是数字标签, 就是label

如果是其他标签, 就是顺序

永远是顺序

```
In [43]: s3.index=[4,3,2,1,0]
In [44]: s3
Out[44]:
4    1.392234
3   -1.020297
2   -0.393115
1   -1.281453
0   -1.509906
dtype: float64
```

s3[4]
s3[[0,2]]
s3[0:2]

```
In [4]: s
```

```
Out[4]:
```

0	a	3
1	b	-5
2	c	7
3	d	4

dtype: int64

```
In [5]: s1
```

```
Out[5]:
```

0	b	bb
1	a	aa
2	c	cc

dtype: object

```
In [6]: s2
```

```
Out[6]:
```

0	a	5.0
1	b	5.0
2	c	5.0
3	d	5.0

dtype: float64

```
In [8]: s3
```

```
Out[8]:
```

0	0	1.392234
1	1	-1.020297
2	2	-0.393115
3	3	-1.281453
4	4	-1.509906

dtype: float64

1. Series与Dataframe

标签是管理复杂数据、关系数据的高效方式，个人还是习惯用“键key列名”或者传统意义上的“下标索引”顺序。
Dataframe亦然、netcdf、xarray亦然.....

标记数组???

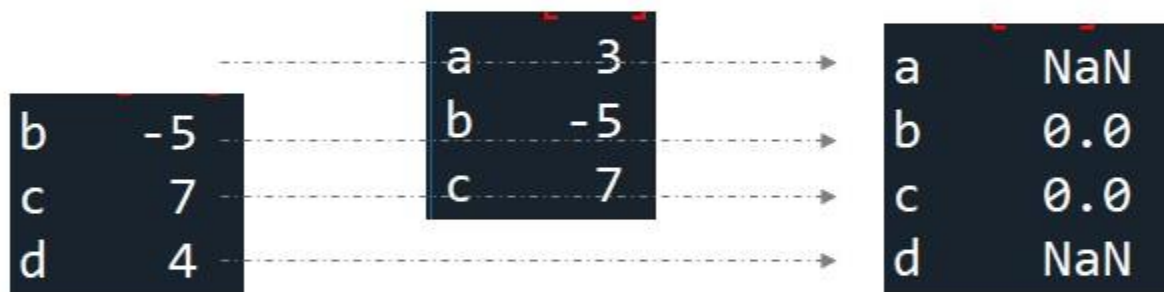
```
In [4]: s
Out[4]:
```

0	a	3
1	b	-5
2	c	7
3	d	4

dtype: int64

S + S

s[1:] - s[:-1] =



s[1:].reset_index(drop=True) - s[:-1].reset_index(drop=True)

s.values[1:] - s.values[:-1] = array([-8, 12, -3], dtype=int64)

1. Series与Dataframe

```
import pandas as pd
```

DataFrame是一种二维标记数据结构，其中包含可能不同类型的列。

你可以把它想象成一个电子表格或SQL表，或者一系列对象的dict。

```
data = {'Time':pd.Timestamp('20201002'), 'Stid':[ 54511, 54602, 53798, 53772],  
        'Name':['Beijing','Baoding','Xingtai','Taiyuan'],  
        'Lon':[ 110, 111, 112, 113], 'Lat':[ 35, 39, 40, 41], 'Rain':[9999, 0.01, 50. ,50.]}  
df = pd.DataFrame(data)
```

```
In [2]: df
```

```
Out[2]:
```

	Time	Stid	Name	Lon	Lat	Rain
0	2020-10-02	54511	Beijing	110	35	9999.00
1	2020-10-02	54602	Baoding	111	39	0.01
2	2020-10-02	53798	Xingtai	112	40	50.00
3	2020-10-02	53772	Taiyuan	113	41	50.00

- 一维数组、列表、字典、序列的字典
- 二维ndarray
- 结构化ndarray
- 序列
- 其他dataframe

1. Series与Dataframe

基础操作索引、切片、筛选

操作	语法	结果
选择列 Select column	df[col]	Series
依据标签选行 Select row by label	df.loc[label]	Series
依据行号选行 Select row by integer location	df.iloc[loc]	Series
行切片 Slice rows	df[5:10]	DataFrame

df['Rain']

df.Rain

s=df[df.columns[0]]

看起来像个属性attribute或方法method

df[['Rain','Time']]

df[0:2] #按照整数位置/行号筛选

df.loc[2] #按照标签选择行, index=[5,7,6,1] df.loc[5:6]

df.iloc[0] #按照整数位置/行号筛选

df.iloc[[0,1]]

df.iloc[[0],[1]]

df.iloc[[0,1],[0,1]]

df.iloc[0:2,[0,1]]

df.iloc[:, [5,1]]

df[::-1]

df[::-2]

df1=df[df['Rain']<999]

and in.....

1. Series与Dataframe

基础操作索引、切片、筛选

条件
筛选

```
df1= df[df['Rain']<999]
df = df[(df['lat']>35)&(df['lon']>115)&(df['lat']<50)&(df['lon']<130)] #筛选辽宁附近

whitelist = [ 54511, 53798, 53772] #高空站点

df2 = df[df.Stid.isin(whitelist)]

dd = df.query('Stid != [54602,53798]') #河北的
df['NoTlogP'] = df['Stid'].where(df['Stid'] == 54602)
```

添加
插入

```
df['flag'] = df['Rain']<999
df['T-td'] = df['T'] -df['td']
df.insert(1, 'level', df['Name'])
```

删除

```
df.drop(['flag',axis=1])
df.pop('flag')
del df['flag']
```

1. Series与Dataframe

```
df.index  
df.columns  
df.shape
```

```
df.dtypes  
df['Lon']=df['Lon'].astype('float32')  
pd.to_datetime(df['time'])  
ndarr = df.to_numpy()
```

```
df.info()
```

```
df.describe()
```

```
df.sum(0, skipna=False)
```

```
df.mean() #axis=0-按列平均 1-按行平均
```

```
df.median()
```

```
df.head()
```

```
df.tail()
```

```
df.T
```

绝大多数方法才能“撼动”原df

其他的都是筛选而已

df=df***

- 用列名
- 用0到len()-1的序号做label! 如果更改了行, reset_index(drop=True) !
- 用传统意义上的下标索引/顺序
- iloc

		0	1	2	3
MultiIndex*	Class1 男	-0.424972	0.567020	0.276232	-1.087401
	女	-0.673690	0.113648	-1.478427	0.524988
	Class2 男	0.404705	0.577046	-1.715002	-1.039268
	女	-0.370647	-1.157892	-1.344312	0.844885
	Class3 男	1.075770	-0.109050	1.643563	-1.469388
	女	0.357021	-0.674600	-1.776904	-0.968914
	Class4 男	-1.294524	0.413738	0.276662	-0.472035
	女	-0.013960	-0.362543	-0.006154	-0.923061

Pandas数据处理Handle

1. Series与Dataframe

Data Structures

2. 创建、输入、输出

`to_csv()`

I/O

3. 索引、切片、筛选、重置

Indexing & Selecting

4. 合并、拼接、追加、删除_{-去重}

Merge, Concat, Append, Join, Drop_{-duplicate}

5. 排序、分组、汇总

Sort, Groupby

6. 计算

Covariance\Correlation\rolling...

甲方需求1

站点信息.txt

从逐5分钟自动站观测文件中，提取研究所需十个站点的风向风速，按照如下格式保存成表格，便于其他程序调用读取分析。

201904161001.csv	海	站	总	3小	过	过	6小
201904161005.csv	站经纬	拔	云量	风风	时	去	去
201904161010.csv	号度度	高度	向速压	向速压	变	天	天
201904161015.csv		高度			压	气1	气2
201904161020.csv	100104 122 2188 42 2106	92	1	0000	228	5.1	0000
201904161025.csv							
201904161030.csv	低	低	低	能	现	中	高
201904161035.csv	云	云	云	见	在	云	云
201904161040.csv	状	量	高	度	天	状	状
201904161045.csv					气	1	2
201904161050.csv	0000	0	0	0000	21.5	0000	0



time	Stid		Stid		Stid	
	风向	风速	风向	风速	风向	风速
201904161	225	6.3	9	8.4	9999	9999
201904161	211	4.7	9	6.9	9999	9999
201904161	216	3.6	10	4.3	9999	9999
201904161	227	3.7	12	8.4	9999	9999
201904161	237	4.3	10	8.2	9999	9999
201904161	221	4.3	11	9	9999	9999

理需求、建思路；看数据、找工具；模块化

1. 打开一个文件，读入数据

2. 筛选数据

站点

风向风速

排序

时间←文件名

3. 一个文件输出

4. 对文件循环，追加时次

5. 最后输出

甲方需求1

站点信息.txt

从逐5分钟自动站观测文件中，提取研究所需十个站点的风向风速，按照如下格式保存成表格，便于其他程序调用读取分析。

理需求、建思路；看数据、找工具；模块化

1. 打开一个文件，读入数据

2. 筛选数据

站点

风向风速

排序

"转置"

时间←文件名

3. 一个文件输出

4. 对文件循环，追加时次

5. 最后输出

```
time = filename[-16:-4]
df = pd.read_csv(filename, usecols=[0, 3, 4])
# df=df[['站号', '风向', '风速']]
data = df[df['站号'].isin(stdls)]
data = data.sort_values(by='站号').reset_index(drop=True)
data = data.drop('站号', axis=1)
data = pd.DataFrame(data.values.reshape(1, 20), columns=['风向', '风速']*10)
# data = data.drop('站号', axis=1).stack().reset_index(drop=True)
data.insert(0, '时间', time)

out = pd.DataFrame()
for filename in glob.glob(pattern):
    out = out.append(data)

out.to_csv()
```

```
# 需求站点列表：从txt文件中读取需要站点站号，列表
with open('../data/站点信息.txt') as f:
    lines=f.read().split(sep='\n')
    stdls=[int(line) for line in lines]
    #s=list(map(int, lines))
```


甲方需求1

站点信息.txt

从逐5分钟自动站观测文件中，提取研究所需十个站点的风向风速，按照如下格式保存成表格，便于其他程序调用读取分析。

理需求、建思路；看数据、找工具；模块化

1. 读取csv文件
2. 访问[]、筛选isin
3. 排序sort_values、重置reset_index
4. 删除列drop
5. 取值.values ndarray变形
6. 由ndarray创建dataframe
7. 追加append
8. 输出to_csv 中文/编码
9. 读取文本文件，按行为列表
10. 遍历文件夹，获取文件名列表

```
# 需求站点列表：从txt文件中读取需要站点站号，列表
with open('../data/站点信息.txt') as f:
    lines=f.read().split(sep='\n')
    stidls=[int(line) for line in lines]
    #s=list(map(int, lines))
```

```
time = filename[-16:-4]
df = pd.read_csv(filename,usecols=[0,3,4])
# df=df[['站号','风向','风速']]
data = df[df['站号'].isin(stidls)]
data = data.sort_values(by='站号').reset_index(drop=True)
data = data.drop('站号',axis=1)
data = pd.DataFrame(data.values.reshape(1,20),columns=['风向','风速']*10)
# data = data.drop('站号',axis=1).stack().reset_index(drop=True)
data.insert(0,'时间',time)

out=pd.DataFrame()
for filename in glob.glob(pattern):
    out = out.append(data)

out.to_csv()
```

甲方需求2

读取MICAPS第1类数据，为后续绘制地面天气填图、对地面气象要素进行客观分析做准备。

理需求、建思路；看数据、找工具；模块化

1. 读取
2. 筛选、合并

```
import pandas as pd
```

```
a=r"..../data/Surf20092308.000" #a=sys.argv[1]
```

```
df=pd.read_csv(a,skiprows=3,header=None,encoding="GB2312",sep='\s+').drop([0])
```

```
df1=df.iloc[::2].reset_index(drop=True)
```

```
df2=df.iloc[1::2].reset_index(drop=True)
```

```
df1=df1.drop(columns=[12,13])
```

```
res=pd.concat([df1,df2],axis=1,ignore_index=True)
```

```
res.columns=["stid","lon",.....,"s1","s2","T24","P24"]
```

```
res[['stid',.....,'p3','r6]]=res[['stid',.....,'p3','r6']].astype('int')
```

```
res=res[res['stid']<80000]
```

```
res['weather']=res['weather'].replace(9999,0)
```

```
1 Diamond 1 20年09月23日02时地面填图
2 20 09 23 02 5997
3 64700 15.03 12.13 295 32 3 0 0 87 9999 0 0
4 0 30 0 9999 24.3 10.0 0 28.5 20 12 1 2 9999 9999
5 8594 -22.94 16.73 55 16 2 90 4 144 9999 0 0
6 0 32 2 300 22.5 25.0 0 30.3 20 10 1 2 9999 9999
7 8589 -23.51 14.90 35 16 2 50 4 140 9999 0 0
8 0 32 2 300 24.5 30.0 0 28.7 20 10 1 2 9999 9999
9 8583 -24.99 16.88 63 32 4 250 3 151 9999 0 0
10 0 38 4 300 24.1 15.0 0 29.5 20 10 1 2 9999 9999
11 41194 55.33 25.25 5 16 0 320 5 34 9999 0 0
12 0 9999 0 9999 25.0 10.0 0 33.1 9999 9999 1 2 9999 9999
13 41196 55.52 25.33 33 2 0 280 2 36 9999 0 0
14 0 9999 0 9999 25.3 10.0 0 32.6 9999 9999 1 2 9999 9999
15 41217 54.65 24.43 27 16 0 50 2 34 9999 0 0
16 9999 9999 0 9999 27.1 10.0 0 31.5 9999 9999 1 2 9999 9999
17 81225 -55.19 5.45 15 32 6 130 3 101 9999 0 0
18 0 32 4 600 21.2 20.0 0 35.2 20 14 1 2 9999 9999
19 81200 -55.17 5.82 7 32 5 110 2 102 9999 0 0
20 0 32 4 600 21.8 20.0 0 34.0 20 12 1 2 9999 9999
21 81202 -57.02 5.95 2 32 4 60 5 101 9999 0 0
22 0 32 1 600 24.0 20.0 0 31.4 20 14 1 2 9999 9999
```


读取MICAPS第4类数据，为后续绘制分析涡度做数据准备。

1. 读取 2. 筛选、合并 3. 变形

```
df=pd.read_csv(fileName, skiprows=4, header=None, encoding="GB2312", sep='\s+')
vor=df.values.reshape(29,60)
vor=np.delete(vor,list(range(53,60)),axis=1)
```

1	diamond 4 20年06月08日08时500百帕涡度场										
2	20	06	08	08	0	500					
3	2.500	-2.500	30.0	160.0	80.0	10.0					
4	53	29	20.0	-200.0	200.0	1	0				
5	-34.3	-41.3	-39.4	-19.5	-1.4	5.2	5.2	5.2	5.2	4.1	
6	4.1	5.2	.7	-13.6	-39.0	-61.1	-64.5	-47.1	-21.3	-3.2	
7	5.0	8.2	13.3	22.0	34.3	41.2	31.7	16.4	6.4	4.7	
8	5.7	5.7	5.7	5.7	5.7	5.7	5.7	6.1	7.5	10.0	
9	13.3	18.9	27.7	30.1	23.8	15.0	7.5	4.7	3.6	3.6	
10	3.6	3.6	3.6	53	54	55	56	57	58	59	
11	-29.8	-38.1	-39.5	-23.1	-7.5	1.4	4.1	4.1	4.1	3.3	
12	3.5	1.5	-5.2	-17.7	-35.3	-46.5	-42.6	-25.9	-8.0	2.5	
13	5.8	4.8	3.2	7.0	21.3	35.6	32.7	17.7	5.2	2.0	
14	3.3	4.4	4.4	4.5	4.5	4.7	5.5	6.2	8.3	13.4	
15	17.9	21.9	25.2	23.1	16.3	9.5	4.4	2.9	2.9	2.9	
16	2.9	2.9	2.9								
17	-14.3	-24.9	-37.2	-33.5	-22.3	-10.3	-1.1	2.7	3.6	3.2	
18	-.8	-8.8	-18.8	-27.8	-29.9	-22.5	-10.5	-1.2	3.5	6.0	
19	4.7	-5.4	-18.7	-21.1	1.3	39.3	54.1	30.6	.1	-10.1	
20	-5.2	-.4	1.7	3.2	3.1	4.5	7.7	11.8	19.0	26.0	
21	27.9	25.5	18.9	12.9	7.4	3.4	2.4	2.4	2.4	2.4	
22	2.4	2.4	2.4								

甲方需求4

每日降水，计算过程降水量，供后续绘制降水落区图使用。

RAIN2020091608.csv
RAIN2020091408.csv
RAIN2020091508.csv

理需求、建思路；看数据、找工具；模块化

1. 文件遍历 2. 读取→追加 3. 分组→求和

```
import glob
```

```
import pandas as pd
```

```
pattern='../data/'+ 'RAIN*.csv'
```

```
dfRain = pd.DataFrame(columns=['站号', '经度', '纬度', '海拔高度', '观测值'])
```

```
for filename in glob.glob(pattern):
```

```
df=pd.read_csv(filename) # print(df.shape)
```

#读取

```
df=df[['站号', '经度', '纬度', '海拔高度', '观测值']]
```

#筛选

```
dfRain=dfRain.append(df, ignore_index=True)
```

#追加

```
cumulative=dfRain.groupby('站号').sum().sort_values(by='站号')
```

#分组求和

站号	经度	纬度	海拔高度	站点级别	观测值	站名
57345	109.62	31.4	337.8	9999	9.9	巫溪
55299	92.07	31.48	4507	9999	2.2	那曲
57348	109.48	31.03	299.8	9999	12.4	奉节
57349	109.87	31.08	606.8	9999	9	巫山
57355	110.35	31.05	337.5	9999	8.7	巴东
57358	110.97	30.83	295.5	9999	12.1	秭归
57359	110.73	31.35	336.8	9999	5.1	兴山
57361	111.27	31.88	327.7	9999	0.2	保康
57362	110.66	31.75	935.2	9999	6.2	神农架
57363	111.83	31.8	151	9999	0.5	南漳
53272	112.63	42.75	1104.9	9999	0.001	苏尼特右旗
57368	111.64	31.05	141.5	9999	6.8	远安
57370	112.22	31.73	67.8	9999	0.6	宜城

甲方需求5*

时间序列，去重。

stID	staName	obsTime	WD2	WS2	WD10	WS10	Lon	Lat
B1620	云顶1	2020/2/4 7:00	282	3.2	290	3.6	115.4197	40.95972
B1627	云顶2	2020/2/4 7:00	349	2	326	2.1	115.4181	40.95972
B1628	云顶3	2020/2/4 7:00	274	3.3	273	3.4	115.4064	40.95472
B1629	云顶4	2020/2/4 7:00	329	3.9	332	3.6	115.4103	40.95583
B1630	云顶5	2020/2/4 7:00	239	2	260	1.5	115.4103	40.95778
B1637	云顶6	2020/2/4 7:00	267	4.6	273	4.5	115.4139	40.95778
B1638	冬两1	2020/2/4 7:00	306	2	291	2.1	115.4747	40.90972
B1639	冬两2	2020/2/4 7:00	283	1.6	280	1.7	115.4875	40.91028
B1640	冬两3	2020/2/4 7:00	53	1.9	51	1.8	115.4831	40.90778
B1646	冬两4	2020/2/4 7:00	305	2.2	309	2.2	115.4725	40.91111
B1647	冬两5	2020/2/4 7:00	115	0.8	86	0.9	115.4736	40.92
B1648	越野1	2020/2/4 7:00	282	1.7	285	1.7	115.4717	40.90083
B1649	越野2	2020/2/4 7:00	240	1.9	246	1.8	115.4739	40.89806
B1650	越野3	2020/2/4 7:00	293	1.2	279	0.8	115.4658	40.90167
B1620	云顶1	2020/2/4 8:00	295	3.6	297	3.6	115.4197	40.95972
B1627	云顶2	2020/2/4 8:00	309	3.3	318	2.5	115.4181	40.95972
B1628	云顶3	2020/2/4 8:00	275	3.9	276	4.3	115.4064	40.95472
B1629	云顶4	2020/2/4 8:00	345	3.3	343	3.6	115.4103	40.95583
B1630	云顶5	2020/2/4 8:00	253	2.1	244	1.9	115.4103	40.95778
B1637	云顶6	2020/2/4 8:00	275	5	281	4.4	115.4139	40.95778
B1638	冬两1	2020/2/4 8:00	294	2.4	293	2.2	115.4747	40.90972
B1639	冬两2	2020/2/4 8:00	283	3.1	272	2.9	115.4875	40.91028
B1640	冬两3	2020/2/4 8:00	40	2	55	1.8	115.4831	40.90778
B1646	冬两4	2020/2/4 8:00	295	2.5	291	2.7	115.4725	40.91111
B1647	冬两5	2020/2/4 8:00	77	1.2	80	1.3	115.4736	40.92
B1648	越野1	2020/2/4 8:00	284	2	284	1.9	115.4717	40.90083
B1649	越野2	2020/2/4 8:00	262	2.3	249	2.1	115.4739	40.89806
B1650	越野3	2020/2/4 8:00	280	2.5	284	1.3	115.4658	40.90167

```
pd.to_datetime(['1/1/2018', np.datetime64('2018-01-01'),  
datetime.datetime(2018, 1, 1)])
```

```
idx = pd.date_range('2018-01-01', periods=5, freq='H')
```

```
friday = pd.Timestamp('2018-01-05')  
saturday = friday + pd.Timedelta('1 day')
```

[::14]

vs

drop_duplicate()

Concept	Scalar Class	Array Class	pandas Data Type	Primary Creation Method
Date times	Timestamp	DatetimeIndex	datetime64[ns] or datetime64[ns, tz]	to_datetime or date_range
Time deltas	Timedelta	TimedeltaIndex	timedelta64[ns]	to_timedelta or timedelta_range
Time spans	Period	PeriodIndex	period[freq]	Period or period_range
Date offsets	DateOffset	None	None	DateOffset

YearEnd	'A'	calendar year end
MonthEnd	'M'	calendar month end
Day	'D'	one absolute day
Hour	'H'	one hour
Minute	'T' or 'min'	one minute
Second	'S'	one second

甲方需求6*

json格式文件解析。

#引用部分

```
import json
```

```
from pandas.io.json
```

```
import json_normalize
```

```
import numpy as np
```

```
import pandas as pd
```

```
a=sys.argv[1] #a='2019121614'
```

#读取json数据

```
with open('data/DM.json', 'r',encoding='utf-8') as fdm:
```

```
    ddm = json.load(fdm)
```

```
    dfdm = json_normalize(ddm['DS'])
```

```
dfdm = dfdm.astype('float64') #转换为数字类型
```

```
dfdm['Station_Id_d']=dfdm['Station_Id_d'].astype(np.int) #把站
```

```
df=dfdm.replace([999999,999999.0,999998,999998.0,999990,999990.0],[99999,99999,99999,99999,0.01,0.0
```

.....

```
{ "returnCode": "0",
  "returnMessage": "Query Succeed",
  "rowCount": "7796",
  "colCount": "38",
  "requestParams":
    "dataCode=SURF_GLB_MUL_HOR&times=20191216060000&elements=Station
    _Id_d,Lat,Lon,Alti,Station_level,CLO_Height_LoM,VIS,CLO_Cov,WIN_D
    ,WIN_S,TEM,DPT,RHU,PRS_Sea,PRS_Change_3h,PRE_1h,PRE_2h,PRE_3h,PR
    E_6h,PRE_9h,PRE_12h,PRE_15h,PRE_18h,PRE_24h,WEP_Now,WEP_Past_1,W
    EP_Past_2,CLO_COV_LM,CLO_Fome_Low,CLO_FOME_MID,CLO_Fome_High,PRS
    _Change_24h,TEM_CHANGE_24h,TEM_Max_12h,TEM_Max_24h,TEM_Min_12h,T
    EM_Min_24h,Snow_Depth",
  "requestTime": "2019-12-16 08:45:26",
  "responseTime": "2019-12-16 08:45:26",
  "takeTime": "0.706",
  "fieldNames": "区站号(数字) 纬度 经度 测站高度 测站级别
  云底高度 水平能见度(人工) 总云量 风向 风速 温度/气温 露点温度
  相对湿度 海平面气压 3小时变压 过去1小时降水量 过去2小时降水量
  过去3小时降水量 过去6小时降水量 过去9小时降水量
  过去12小时降水量 过去15小时降水量 过去18小时降水量
  过去24小时降水量 现在天气 过去天气1 过去天气2
  云量(低云或中云) 低云状 中云状 高云状 24小时变压
  过去24小时变温 过去12小时最高气温 过去24小时最高气温
  过去12小时最低气温 过去24小时最低气温 积雪深度",
  "fieldUnits": "- 度 度 米 - 米 米 百分率 度 米/秒 摄氏度(°C)
  摄氏度(°C) 百分率 百帕 百帕 毫米 毫米 毫米 毫米 毫米 毫米
  毫米 - - - 百分率 - - - 百帕 摄氏度(°C) 摄氏度(°C)
  摄氏度(°C) 摄氏度(°C) 摄氏度(°C) 厘米",
  "DS": [
    { "Station_Id_d": "999999", "Lat": "53.6", "Lon": "0.1", "Alti":
      "999999", "Station_level": "999999", "CLO_Height_LoM": "999999",
      "VIS": "999999", "CLO_Cov": "999998", "WIN_D": "999999", "WIN_S":
      "999999", "TEM": "4.4", "DPT": "3", "RHU": "999999", "PRS_Sea": "995.3",
      "PRS_Change_3h": "0.3", "PRE_1h": "999999", "PRE_2h": "999999",
      "PRE_3h": "999999", "PRE_6h": "999999", "PRE_9h": "999999", "PRE_12h":
      "999999", "PRE_15h": "999999", "PRE_18h": "999999", "PRE_24h":
      "999999", "WEP_Now": "999999", "WEP_Past_1": "999999", "WEP_Past_2":
      "999999", "CLO_COV_LM": "999999", "CLO_Fome_Low": "999999",
      "CLO_FOME_MID": "999999", "CLO_Fome_High": "999999",
      "PRS_Change_24h": "999999", "TEM_CHANGE_24h": "999999",
      "TEM_Max_12h": "999999", "TEM_Max_24h": "999999", "TEM_Min_12h":
      "999999", "TEM_Min_24h": "999999", "Snow_Depth": "999999"},
    { "Station_Id_d": "101", "Lat": "28.45", "Lon": "119.48", "Alti":
      "149.2", "Station_level": "999999", "CLO_Height_LoM": "999999", "VIS":
      "20000", "CLO_Cov": "999999", "WIN_D": "126", "WIN_S": "1.5", "TEM":
      "25.8", "DPT": "13", "RHU": "45", "PRS_Sea": "1015", "PRS_Change_3h":
      "-4.5", "PRE_1h": "0", "PRE_2h": "999998", "PRE_3h": "0", "PRE_6h": "0",
      "PRE_9h": "999998", "PRE_12h": "0", "PRE_15h": "999998", "PRE_18h":
      "999998", "PRE_24h": "0", "WEP_Now": "0", "WEP_Past_1": "0",
      "WEP_Past_2": "0", "CLO_COV_LM": "999999", "CLO_Fome_Low": "999999",
      "CLO_FOME_MID": "999999", "CLO_Fome_High": "999999",
```


Pandas数据处理Handle

1. Series与Dataframe

2. 创建、输入、输出

3. 索引、切片、筛选、重置

4. 合并、拼接、追加、删除_{-去重}

5. 排序、分组、汇总

6. 计算统计

- 访问列名、行号检索

- `df=df....reset_dindex()`

- 用啥搜啥，用啥学啥



netCDF4

- | | |
|---|----------|
| 1. Creating/Opening/Closing a netCDF file. | 创建/打开/关闭 |
| 2. Groups in a netCDF file. | 组 |
| 3. Dimensions in a netCDF file. | 维度 |
| 4. Variables in a netCDF file. | 变量 |
| 5. Attributes in a netCDF file. | 属性 |
| 6. Writing data to and retrieving data from a | 写入并检索 |
| 7. Dealing with time coordinates. | 处理时间坐标 |
| 8. Reading data from a multi-file netCDF dat | 多数剧集读取 |
| 9. Efficient compression of netCDF variables | 有效压缩 |
| 10. Beyond homogeneous arrays of a fixed ty | 复合数据类型 |
| 11. Variable-length (vlen) data types. | 数据类型 |
| 12. Enum data type. | 并行IO |
| 13. Parallel IO. | 处理字符串 |
| 14. Dealing with strings. | 内存数据集 |
| 15. In-memory (diskless) Datasets. | |



N-D labeled arrays and datasets

USER GUIDE

Terminology

Data Structures

Indexing and selecting data

Interpolating data

Computation

GroupBy: split-apply-combine

Reshaping and reorganizing data

Combining data

Time series data

Weather and climate data

Working with pandas

Reading and writing files

Parallel computing with Dask

Plotting

术语

数据结构

索引和选择

数据插值

数据计算

GroupBy:拆分应用组合

重塑和重组数据

组合数据

时间序列

数据天气气候资料

与Pandas协作

读写文件

Dask并行计算

绘图