

teqc数据质量检查批处理脚本使用手册

- 一、说明
- 二、功能
- 三、运行环境
- 四、参数说明
- 五、使用实例
 - 1、命令行输出结果
 - 2、文本文件输出
- 六、注意事项
 - 1、脚本存在的问题
 - 2、注意事项
- 七、脚本代码 (qualitycheck_2.py)

teqc数据质量检查批处理脚本使用手册

一、说明

`qualitycheck_2.py` 脚本是一个批量进行 RINEX 数据质量分析的脚本，通过在命令行调用 TEQC 程序对输入观测数据处理，质量分析，输出观测时长、信噪比、多路径效应、周跳等质量检查成果。

二、功能

我们知道TEQC分有**qc2lite（该模式下无数据完整率）**和**qc2full（有数据完整率）**两种检核方式。该脚本的功能实现是 `qc2full` 处理模式的批处理：即利用观测值文件和导航电文才能实现观测数据质量检查，并输出检查结果。

- 输入：观测值文件和导航电文的文件夹名称
- 输出：文件名称、时期、时间、时长、完整率、信噪比、多路径效应、周跳
- 输出模式：命令行输出、文件输出（只支持*.txt）

三、运行环境

由于本程序的质量分析操作依赖于 TEQC 程序，但是因不同版本的 TEQC 的输出信息格式有略微不同，本脚本保证只要teqc数据输出形式不变的情况下，在使用 2019 及以后版本的 TEQC 时测试都能通过。

对于 Windows 10、Windows 7 等操作系统，需保证运行脚本的文件夹内有 TEQC 程序，否则可能出现 `teqc` 不是内部或外部命令，也不是可运行的程序或批处理文件。”的错误。

四、参数说明

该脚本在执行时有四个参数：

```
1 | PS $ python qualitycheck_2.py -nav <folder_name> -obs <folder_name> [-out  
  | <format> -fn <filename>]
```

- `-nav`：导航电文；<folder_name>导航电文的文件夹名称。
- `-obs`：观测文件；<folder_name>观测数据所在的文件夹名称。
- `-out`：只有在要输出为文本文件时使用；为 `t` 或 `table` 时为表格形式。

- `-fn`：输出文件名称 (*.txt)
- 注意：只有在使用 `-out` 后才能使用 `-fn`

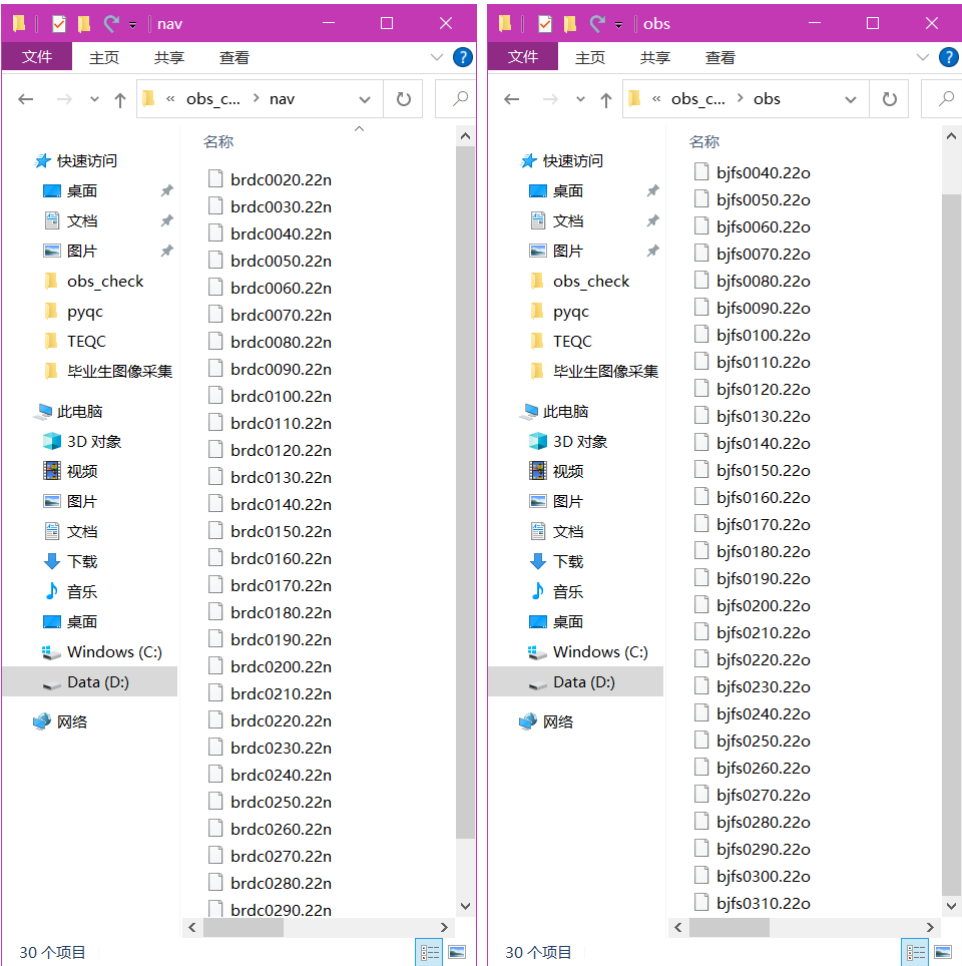
五、使用实例

数据准备：有观测数据文件夹 `obs`，导航电文文件夹 `nav`；注意：要保证两个文件的文件要一一对应。

名称	修改日期	类型	大小
nav	2022/10/16 16:08	文件夹	
obs	2022/10/16 17:07	文件夹	
qualitycheck_2.py	2022/10/16 15:49	PY 文件	7 KB
teqc.exe	2019/12/16 21:01	应用程序	1,634 KB

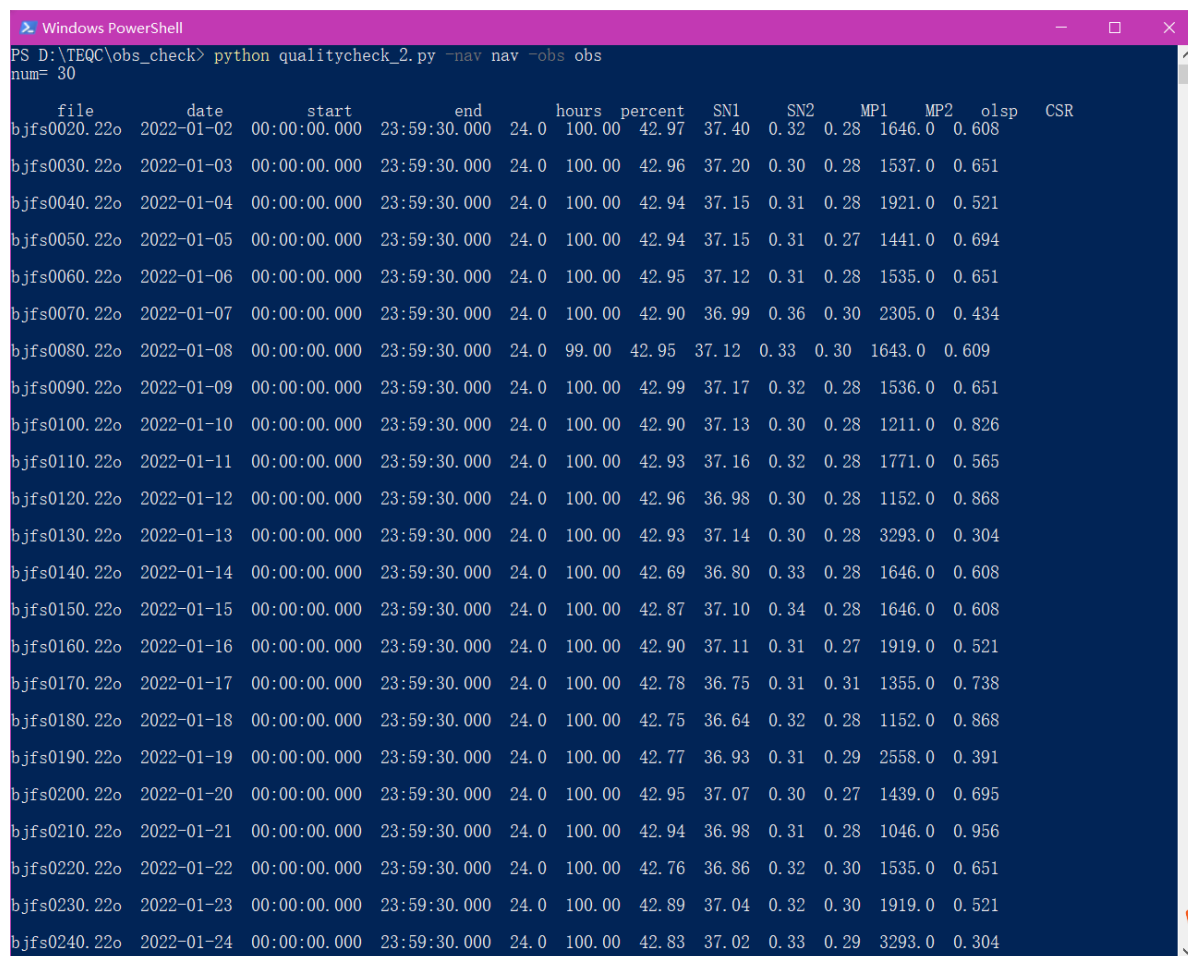
导航文件夹
观测文件夹
脚本
程序

obs	nav
bjfs0020.22o	brdc0020.22n
bjfs0030.22o	brdc0030.22n
bjfs0040.22o	brdc0040.22n
.....



1、命令行输出结果

```
1 | PS D:\TEQC\obs_check> python qualitycheck_2.py -nav nav -obs obs
```



```
Windows PowerShell
PS D:\TEQC\obs_check> python qualitycheck_2.py -nav nav -obs obs
num= 30

file      date      start      end      hours  percent  SN1  SN2  MP1  MP2  olsp  CSR
bjfs0020.22o  2022-01-02  00:00:00.000  23:59:30.000  24.0  100.00  42.97  37.40  0.32  0.28  1646.0  0.608
bjfs0030.22o  2022-01-03  00:00:00.000  23:59:30.000  24.0  100.00  42.96  37.20  0.30  0.28  1537.0  0.651
bjfs0040.22o  2022-01-04  00:00:00.000  23:59:30.000  24.0  100.00  42.94  37.15  0.31  0.28  1921.0  0.521
bjfs0050.22o  2022-01-05  00:00:00.000  23:59:30.000  24.0  100.00  42.94  37.15  0.31  0.27  1441.0  0.694
bjfs0060.22o  2022-01-06  00:00:00.000  23:59:30.000  24.0  100.00  42.95  37.12  0.31  0.28  1535.0  0.651
bjfs0070.22o  2022-01-07  00:00:00.000  23:59:30.000  24.0  100.00  42.90  36.99  0.36  0.30  2305.0  0.434
bjfs0080.22o  2022-01-08  00:00:00.000  23:59:30.000  24.0  99.00  42.95  37.12  0.33  0.30  1643.0  0.609
bjfs0090.22o  2022-01-09  00:00:00.000  23:59:30.000  24.0  100.00  42.99  37.17  0.32  0.28  1536.0  0.651
bjfs0100.22o  2022-01-10  00:00:00.000  23:59:30.000  24.0  100.00  42.90  37.13  0.30  0.28  1211.0  0.826
bjfs0110.22o  2022-01-11  00:00:00.000  23:59:30.000  24.0  100.00  42.93  37.16  0.32  0.28  1771.0  0.565
bjfs0120.22o  2022-01-12  00:00:00.000  23:59:30.000  24.0  100.00  42.96  36.98  0.30  0.28  1152.0  0.868
bjfs0130.22o  2022-01-13  00:00:00.000  23:59:30.000  24.0  100.00  42.93  37.14  0.30  0.28  3293.0  0.304
bjfs0140.22o  2022-01-14  00:00:00.000  23:59:30.000  24.0  100.00  42.69  36.80  0.33  0.28  1646.0  0.608
bjfs0150.22o  2022-01-15  00:00:00.000  23:59:30.000  24.0  100.00  42.87  37.10  0.34  0.28  1646.0  0.608
bjfs0160.22o  2022-01-16  00:00:00.000  23:59:30.000  24.0  100.00  42.90  37.11  0.31  0.27  1919.0  0.521
bjfs0170.22o  2022-01-17  00:00:00.000  23:59:30.000  24.0  100.00  42.78  36.75  0.31  0.31  1355.0  0.738
bjfs0180.22o  2022-01-18  00:00:00.000  23:59:30.000  24.0  100.00  42.75  36.64  0.32  0.28  1152.0  0.868
bjfs0190.22o  2022-01-19  00:00:00.000  23:59:30.000  24.0  100.00  42.77  36.93  0.31  0.29  2558.0  0.391
bjfs0200.22o  2022-01-20  00:00:00.000  23:59:30.000  24.0  100.00  42.95  37.07  0.30  0.27  1439.0  0.695
bjfs0210.22o  2022-01-21  00:00:00.000  23:59:30.000  24.0  100.00  42.94  36.98  0.31  0.28  1046.0  0.956
bjfs0220.22o  2022-01-22  00:00:00.000  23:59:30.000  24.0  100.00  42.76  36.86  0.32  0.30  1535.0  0.651
bjfs0230.22o  2022-01-23  00:00:00.000  23:59:30.000  24.0  100.00  42.89  37.04  0.32  0.30  1919.0  0.521
bjfs0240.22o  2022-01-24  00:00:00.000  23:59:30.000  24.0  100.00  42.83  37.02  0.33  0.29  3293.0  0.304
```

2、文本文件输出

```
1 | PS D:\TEQC\obs_check> python qualitycheck_2.py -nav nav -obs obs -out t -fn 20221016result.txt
```

注意：在当前的文件夹下会生成一个 20221016result.txt 的文件。

20221016result.txt - 记事本														
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)														
file	date	start	end	hours	percent	SN1	SN2	MP1	MP2	olsp	CSR			
bjfs0020.22o	2022-01-02	00:00:00.000	23:59:30.000	24.0	100.00	42.97	37.40	0.32	0.28	1646.0	0.608			
bjfs0030.22o	2022-01-03	00:00:00.000	23:59:30.000	24.0	100.00	42.96	37.20	0.30	0.28	1537.0	0.651			
bjfs0040.22o	2022-01-04	00:00:00.000	23:59:30.000	24.0	100.00	42.94	37.15	0.31	0.28	1921.0	0.521			
bjfs0050.22o	2022-01-05	00:00:00.000	23:59:30.000	24.0	100.00	42.94	37.15	0.31	0.27	1441.0	0.694			
bjfs0060.22o	2022-01-06	00:00:00.000	23:59:30.000	24.0	100.00	42.95	37.12	0.31	0.28	1535.0	0.651			
bjfs0070.22o	2022-01-07	00:00:00.000	23:59:30.000	24.0	100.00	42.90	36.99	0.36	0.30	2305.0	0.434			
bjfs0080.22o	2022-01-08	00:00:00.000	23:59:30.000	24.0	99.00	42.95	37.12	0.33	0.30	1643.0	0.609			
bjfs0090.22o	2022-01-09	00:00:00.000	23:59:30.000	24.0	100.00	42.99	37.17	0.32	0.28	1536.0	0.651			
bjfs0100.22o	2022-01-10	00:00:00.000	23:59:30.000	24.0	100.00	42.90	37.13	0.30	0.28	1211.0	0.826			
bjfs0110.22o	2022-01-11	00:00:00.000	23:59:30.000	24.0	100.00	42.93	37.16	0.32	0.28	1771.0	0.565			
bjfs0120.22o	2022-01-12	00:00:00.000	23:59:30.000	24.0	100.00	42.96	36.98	0.30	0.28	1152.0	0.868			
bjfs0130.22o	2022-01-13	00:00:00.000	23:59:30.000	24.0	100.00	42.93	37.14	0.30	0.28	3293.0	0.304			
bjfs0140.22o	2022-01-14	00:00:00.000	23:59:30.000	24.0	100.00	42.69	36.80	0.33	0.28	1646.0	0.608			
bjfs0150.22o	2022-01-15	00:00:00.000	23:59:30.000	24.0	100.00	42.87	37.10	0.34	0.28	1646.0	0.608			
bjfs0160.22o	2022-01-16	00:00:00.000	23:59:30.000	24.0	100.00	42.90	37.11	0.31	0.27	1919.0	0.521			
bjfs0170.22o	2022-01-17	00:00:00.000	23:59:30.000	24.0	100.00	42.78	36.75	0.31	0.31	1355.0	0.738			
bjfs0180.22o	2022-01-18	00:00:00.000	23:59:30.000	24.0	100.00	42.75	36.64	0.32	0.28	1152.0	0.868			
bjfs0190.22o	2022-01-19	00:00:00.000	23:59:30.000	24.0	100.00	42.77	36.93	0.31	0.29	2558.0	0.391			
bjfs0200.22o	2022-01-20	00:00:00.000	23:59:30.000	24.0	100.00	42.95	37.07	0.30	0.27	1439.0	0.695			
bjfs0210.22o	2022-01-21	00:00:00.000	23:59:30.000	24.0	100.00	42.94	36.98	0.31	0.28	1046.0	0.956			
bjfs0220.22o	2022-01-22	00:00:00.000	23:59:30.000	24.0	100.00	42.76	36.86	0.32	0.30	1535.0	0.651			
bjfs0230.22o	2022-01-23	00:00:00.000	23:59:30.000	24.0	100.00	42.89	37.04	0.32	0.30	1919.0	0.521			
bjfs0240.22o	2022-01-24	00:00:00.000	23:59:30.000	24.0	100.00	42.83	37.02	0.33	0.29	3293.0	0.304			
bjfs0250.22o	2022-01-25	00:00:00.000	23:59:30.000	24.0	100.00	42.94	37.07	0.34	0.29	2304.0	0.434			
bjfs0260.22o	2022-01-26	00:00:00.000	23:59:30.000	24.0	100.00	42.93	37.19	0.31	0.28	1918.0	0.521			
bjfs0270.22o	2022-01-27	00:00:00.000	23:59:30.000	24.0	99.00	42.86	36.83	0.30	0.28	1398.0	0.715			
bjfs0280.22o	2022-01-28	00:00:00.000	23:59:30.000	24.0	100.00	42.83	36.72	0.30	0.29	1589.0	0.629			
bjfs0290.22o	2022-01-29	00:00:00.000	23:59:30.000	24.0	100.00	42.95	36.88	0.31	0.29	1489.0	0.672			
bjfs0300.22o	2022-01-30	00:00:00.000	23:59:30.000	24.0	100.00	42.96	36.89	0.34	0.29	1588.0	0.630			
bjfs0310.22o	2022-01-31	00:00:00.000	23:59:30.000	24.0	100.00	42.94	36.92	0.33	0.34	1701.0	0.588			
第 1 行, 第 1 列					100%	Windows (CRLF)		UTF-8						

六、注意事项

1、脚本存在的问题

不管采用哪种输出模式，只要脚本顺利运行了，就会产生一系列对应的**质量汇总文件(*.S)**，这些文件默认在**观测值文件夹目录下**，若需要进行二次运行脚本需要将这些 (*.S) 文件移除，否则会报错，甚至得到错误的结果。

2、注意事项

- 使用了 `-nav` 就必须搭配 `-fn`。
- `-fn` 有默认的文件名称，如果 `-fn` 后未接文件名，会采用默认的 `result.txt` 的文件名
- 一定要保证观测值文件夹的文件与导航电文文件夹的文件一致。

七、脚本代码 (qualitycheck_2.py)

```
1 #Quality check for RINEX observation files using TEQC software.
2 #python3.8
3 #lijun
4 import argparse
5 import os,sys
6 import glob
7 from concurrent import futures
8 import datetime
9 import subprocess
10
11 check_information = \
12 (
```

```

13     {'name': 'start', 'flag': 'Time of start of window :', 'pos': slice(25,
14         {'name': 'end', 'flag': 'Time of end of window :', 'pos': slice(37,
15         {'name': 'length', 'flag': 'Time line window length :', 'pos':
16         {'name': 'MP1', 'flag': 'Moving average MP12      :', 'pos': slice(26,
17         {'name': 'MP2', 'flag': 'Moving average MP21      :', 'pos': slice(26,
18         {'name': 'SN1', 'flag': 'Mean S1                    :', 'pos': slice(26,
19         {'name': 'SN2', 'flag': 'Mean S2                    :', 'pos': slice(26,
20     )
21 #slice(start,end)从已有数组中返回选定的元素, 返回一个新数组, 包含从start到end (不包含
    该元素) 的数组元素
22 # 定义命令行中的参数
23 def get_args():
24     parser = argparse.ArgumentParser(description="quality check of using
    TEQC") #创建解释器-创建ArgumentParser()的对象parser
25     parser.add_argument('-nav', type=str, metavar='<nav_files>',
        #通过add_argument添加参数nav
26                             default='', help="Navigation files for complete
    mode")
27     parser.add_argument('-obs', type=str, metavar='<obs_files>',
        #通过add_argument添加参数obs
28                             default='', help="Observition files for complete
    mode" )
29     parser.add_argument('-out', metavar='<format>',
        #通过add_argument添加参数out
30                             choices=['table', 't'], help="Out format to txt or
    screen")
31     parser.add_argument('-fn', type=str, metavar='<filename>',
        #通过add_argument添加参数fn
32                             default='result.txt', help="Custom file name")
33     args=parser.parse_args()
        #命令行参数解析parser.parse_args()
34     return args
35
36 #根据返回的参数获取文件并遍历存储
37 def get_files():
38     global count                                # 定义局部的全局变量
39     args = get_args()
40
41     nav_fn, obs_fn = args.nav, args.obs          # 获取文件夹名称
42     out_format, out_fn = args.out, args.fn       # 获取输出形式和文件名
43
44     path = os.getcwd()
45     path_nav = os.path.join( path, nav_fn )      # 将当前路径与文件夹拼接
    (如'D:\\PycharmProjects\\nav')
46     path_obs = os.path.join( path, obs_fn )
47
48     filename_nav=os.listdir(path_nav)            # 遍历文件夹下的文件,存储为列
    表

```

```

49     filename_obs=os.listdir (path_obs)
50
51     obs_count = len(filename_obs)                # 获取文件数量
52     nav_count = len(filename_nav)
53     #异常处理
54     try:
55         if obs_count == nav_count:
56             count=obs_count
57     except:
58         print ( "|-----Tips:The number of obs does not equal the number of
nav-----|\n"
59                 "|-----The process is about to terminate----|" )
60         sys.exit ( 0 )
61
62     return nav_fn,obs_fn,filename_nav,filename_obs,count,out_format,out_fn
63
64 #
65 def quality_check(nav_file,obs_file):
66     args='teqc','+qc','-nav',nav_file,obs_file
67     status,output=subprocess.getstatusoutput(' '.join(args))
68     #print('status=',status)
69     #print('output=',output)
70     if status > 0:
71         out = None
72     else:
73         out = output.split('\n')
74     return out
75
76
77 def parallel_teqc():
78     nav_fn,obs_fn,nav_file0, obs_file0, num, out_fmt,out_fn0=get_files ()
79     #nav_fn,obs_fn文件夹名 (brdc, bjfs) ; nav_file, obs_file文件名
(1.11n,1.11o)
80     if out_fmt in ['t','table']:
81         f=open ( out_fn0, mode='a+', encoding='utf-8' )
82         header=print_header ()
83         f.write (header+'\n')
84         f.close ()
85     else:
86         print ( 'num=', num )
87         print ( print_header () )
88     # 线程池中创建最多执行1个线程, 同时通过ThreadPoolExecutor来生成一个executor对象
89     with futures.ThreadPoolExecutor(max_workers=1) as executor:
90
91         for i in range(num):
92             # 路径拼接(如: obs/bfdc1530.11n)
93             path_nav_file=nav_fn+'/'+nav_file0[i]
94             path_obs_file=obs_fn+'/'+obs_file0[i]
95             # 调用executor对象的submit方法, 提交1个任务
96             future =
executor.submit(quality_check,path_nav_file,path_obs_file)
97             # 调用Future对象的result方法, 返回被执行函数的结果
98             res=future.result ()
99             if res:
100                 record=parse_report(res)

```

```

101         results=str(obs_file0[i])+str(record)
102         res_0=results.replace("'",' ').replace('"', '"', '
').replace("'", '"', ' ').replace(", ", ', ')
103         res=res_0.replace(' ', ' ').replace('"', '\n')
104
105         if out_fmt in ['t','table']:
106             f=open ( out_fn0, mode='a+', encoding='utf-8' )
107             f.write ( res )
108             f.close ()
109         else:
110             print ( res )
111
112 # 从报表中获取需要的参数
113 def parse_report(report):
114     marks = {}
115     for item in check_information:
116         for line in report:
117             if item['flag'] in line:
118                 marks[item['name']] = line[item['pos']].strip()
119                 break
120
121     # 获取字典中对应键的值
122     sn1 = format(float(marks.get('SN1', 'nan')),'.2f')
123     sn2 = format(float(marks.get('SN2', 'nan')),'.2f')
124     mp1 = format(float(marks.get('MP1', 'nan')),'.2f')
125     mp2 = format(float(marks.get('MP2', 'nan')),'.2f')
126     date = datetime.datetime.strptime(marks['start'][0:11], '%Y %b %d')
127     start = marks['start'][11:].strip()
128     end = marks['end']
129
130     last_line = next(l for l in reversed(report) if l.startswith('SUM'))
131     last_line_pieces = last_line.split()
132     length = float(last_line_pieces[-8])
133
134     # Get the percentage of data, maybe unknown
135     percentage = last_line_pieces[-4]
136     if percentage == '-':
137         percentage = float('nan')
138     else:
139         percentage = format(float(percentage),'.2f')
140
141     # Get CSR from the last line of report, the olps may equal 0
142     olps =round(float(last_line_pieces[-1]),0)
143     if olps == 0:
144         csr = float ( 'nan' )
145     else:
146         csr = format(1000 / olps, '.3f')
147     result = (date.strftime('%Y-%m-%d'), start, end, length, percentage,
148 sn1,
149                 sn2, mp1, mp2, olps, csr)
149     return result
150
151 # 打印表头
152 def print_header():
153     header=('file', 'date', 'start', 'end', 'hours', 'percent',

```

```
154         'SN1', 'SN2', 'MP1', 'MP2', 'olsp' , 'CSR')
155     style=('\\n{0: ^14s} {1: ^12s} {2: ^14s} {3: ^14s} {4: >6s} {5: >7s}'
156         '{6: >6s} {7: >6s} {8: >6s} {9: >5s} {10: >5s} {11: >5s}')
157     format=style.format ( *header )
158     return format
159
160 if __name__ == '__main__':
161     parallel_teqc()
```