

RESSET 高频数据简介及使用说明

内容介绍：提供上海与深圳两个交易所上市交易工具的高频数据。相关工具包括股票、指数、债券、基金、权证、回购等。如交易的时间、成交价格、成交量、5 个卖价与卖量、5 个买价与买量等，及相应的市场买卖指标。

数据集命名规则：

数据集命名规则_分笔

类别：股票 Stk，债券 Bond，基金 Fund，指数 Indx，回购 Repo，权证 Wrnt，资产支持证券 Abs。

市场标识：上交所 sh，深交所 sz。

1999-2009：

分笔数据以某交易工具代码的命名规则：类别+年份后两位_代码。每年每个证券一个数据集。如 stk99_000001 为股票 000001 在 1999 年的高频分笔数据。

2010-2011：

分笔数据以某交易工具代码的命名规则：类别+年份_代码。每年每个证券一个数据集。如 fund2010_150001 为基金 150001 在 2010 年的高频分笔数据，stk2011_000001 为股票 000001 在 2011 年的高频分笔数据。

2012 年起：

以某交易工具代码的命名规则：类别+HF+年份_代码+市场标识。每年每个证券一个数据集。如 stkhf2012_000001sz 为股票深发展（000001）2012 年的高频分笔数据，indxhf2012_000001sh 为上证指数（000001）2012 年的高频分笔数据。

数据集命名规则_分时：

类别：股票 Stk，债券 Bond，基金 Fund，指数 Indx，回购 Repo，权证 Wrnt，资产支持证券 Abs。

市场标识：上交所 sh，深交所 sz。

八种分时区间：1 分钟、5 分钟、10 分钟、15 分钟、20 分钟、30 分钟、40 分钟、60 分钟。

1999-2011 年：

分时数据以市场类别的命名规则：

1 分钟分时：市场标识+y+年份+m+月份_分钟数。每月每个市场一个数据集。如：shy1999m07_1 为上交所 1999 年 7 月的 1 分钟数据，szy2011m12_1 为深交所 2011 年 12 月的 1 分钟数据。

其它分时区间：市场标识+y+年份_分钟数。每年每个市场一个数据集。如：shy1999_5 为上交所 1999 年的 5 分钟数据，szy2011_10 为深交所 2011 年的 10 分钟数据。

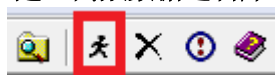
2012 年起：

分时数据文件以某交易工具代码的命名规则：类别+HF+年份_代码+市场标识_分钟数。每年每个证券一个数据集。如：stkhf2012_000001sz_5 为股票深发展（000001）2012 年的 5 分钟数据；indxhf2012_000001sh_5 为上证指数（000001）2012 年的 5 分钟数据。

变量信息请参考数据词典。

使用方法：

1、建立高频数据逻辑库：运行本机的 SAS 软件，在编辑器中输入以下命令后，按 F3 或点击图标



后运行：

```
%let ressethf=166.111.96.98 80;
options comamid=TCP remote=resethf;
signon ressethf username=reset password=reset;
/* 以下为按年份划分的分笔高频数据逻辑库 */
libname HF1999 'D:\RESSET\HF1999' server=resethf;
libname HF2000 'D:\RESSET\HF2000' server=resethf;
libname HF2001 'D:\RESSET\HF2001' server=resethf;
libname HF2002 'D:\RESSET\HF2002' server=resethf;
libname HF2003 'D:\RESSET\HF2003' server=resethf;
```

```
libname HF2004 'D:\RESSET\HF2004' server=ressethf;
libname HF2005 'D:\RESSET\HF2005' server=ressethf;
libname HF2006 'D:\RESSET\HF2006' server=ressethf;
libname HF2007 'D:\RESSET\HF2007' server=ressethf;
libname HF2008 'D:\RESSET\HF2008' server=ressethf;
libname HF2009 'D:\RESSET\HF2009' server=ressethf;
libname HF2010 'D:\RESSET\HF2010' server=ressethf;
libname HF2011 'D:\RESSET\HF2011' server=ressethf;
libname HF2012 'D:\RESSET\HF2012' server=ressethf;
libname HF2013 'D:\RESSET\HF2013' server=ressethf;
libname HF2014 'D:\RESSET\HF2014' server=ressethf;
libname HF2015 'D:\RESSET\HF2015' server=ressethf;
libname HF2016 'D:\RESSET\HF2016' server=ressethf;
libname HF2017 'D:\RESSET\HF2017' server=ressethf;
libname HF2018 'D:\RESSET\HF2018' server=ressethf;
libname HF2019 'D:\RESSET\HF2019' server=ressethf;
libname HF2020 'D:\RESSET\HF2020' server=ressethf;
libname HF2021 'D:\RESSET\HF2021' server=ressethf;
libname HF2022 'D:\RESSET\HF2022' server=ressethf;
```

/* 以下为按年份划分的分时高频数据逻辑库 */

```
libname HF1999M 'D:\RESSET\HF1999_M' server=ressethf;
libname HF2000M 'D:\RESSET\HF2000_M' server=ressethf;
libname HF2001M 'D:\RESSET\HF2001_M' server=ressethf;
libname HF2002M 'D:\RESSET\HF2002_M' server=ressethf;
libname HF2003M 'D:\RESSET\HF2003_M' server=ressethf;
libname HF2004M 'D:\RESSET\HF2004_M' server=ressethf;
libname HF2005M 'D:\RESSET\HF2005_M' server=ressethf;
libname HF2006M 'D:\RESSET\HF2006_M' server=ressethf;
libname HF2007M 'D:\RESSET\HF2007_M' server=ressethf;
libname HF2008M 'D:\RESSET\HF2008_M' server=ressethf;
libname HF2009M 'D:\RESSET\HF2009_M' server=ressethf;
libname HF2010M 'D:\RESSET\HF2010_M' server=ressethf;
libname HF2011M 'D:\RESSET\HF2011_M' server=ressethf;
libname HF2012M 'D:\RESSET\HF2012_M' server=ressethf;
libname HF2013M 'D:\RESSET\HF2013_M' server=ressethf;
libname HF2014M 'D:\RESSET\HF2014_M' server=ressethf;
libname HF2015M 'D:\RESSET\HF2015_M' server=ressethf;
libname HF2016M 'D:\RESSET\HF2016_M' server=ressethf;
libname HF2017M 'D:\RESSET\HF2017_M' server=ressethf;
libname HF2018M 'D:\RESSET\HF2018_M' server=ressethf;
libname HF2019M 'D:\RESSET\HF2019_M' server=ressethf;
libname HF2020M 'D:\RESSET\HF2020_M' server=ressethf;
libname HF2021M 'D:\RESSET\HF2021_M' server=ressethf;
libname HF2022M 'D:\RESSET\HF2022_M' server=ressethf;
```

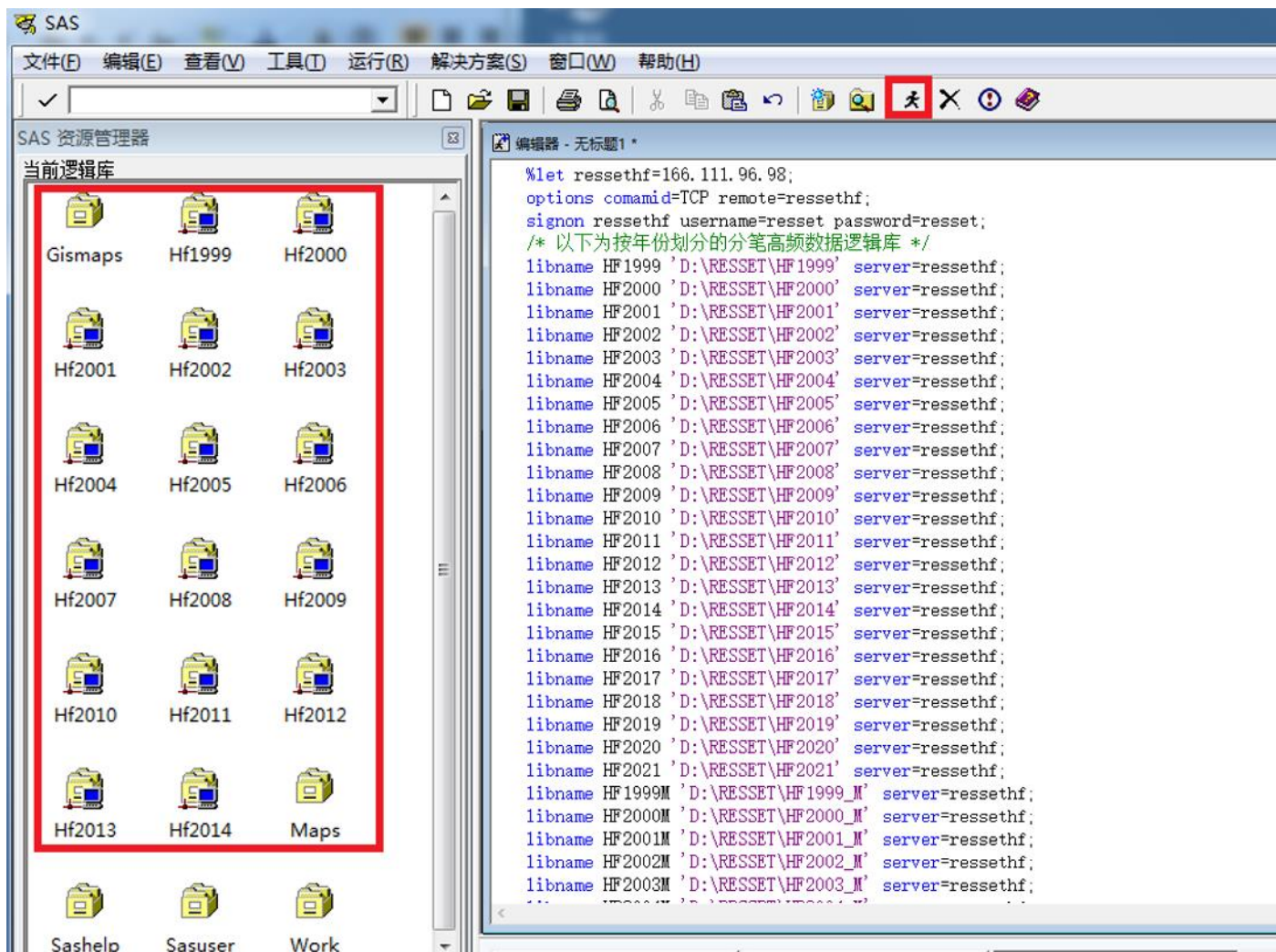


图1

2、提取及处理高频数据：通过第一步建立高频数据逻辑库以后即可通过 SAS 软件的导入导出功能或 SAS 命令来进行数据的提取及处理操作

根据数据词典中分笔逻辑库命名规则，逻辑库的名称规定了里面所有表格的时间范围，例如HF2016表示2016年的高频数据。请大家根据自己的需求进行调整。

根据数据词典中分笔数据集的命名方式，数据集的名称说明了存放内容的5种具体信息，

例如StkHF2016_600000SH，可以分为/Stk/HF/2016/600000/SH/

其中每一部分的含义如下。

Stk	:	表示证券类别是“股票”，其他的证券类别还包括：债券（Bond），基金（Fund），指数（Indx），回购（Repo），权证（Wrnt），资产支持证券（Abs）
HF	:	表示高频数据
2016	:	表示数据时间区间为2016年整年的数据
600000	:	表示股票代码是600000
SH	:	表示交易所为上交所（上海证券交易所）

示例一：以下示例查询代码为600000的上交所股票在2016年1月29日的分笔行情数据

```
data temp.a;
```

```
/*上面的语句，data a中最后的a，是输出数据集的名字，可以自行修改*/
```

```
set HF2016.StkHF2016_600000SH;
```

```
where qdate = '29Jan2016'd;
```

```
/*where语句是筛选语句，请在后面写明查询条件，将会返回符合查询条件的观测，如果没有符合条件的观测，将会创建空数据集。*/
```

```
/*查询条件在指定时间时可以直接使用字符串的方式指定，但是格式必须固定为这种格式 '29Jan2016'd
*/
run;
```

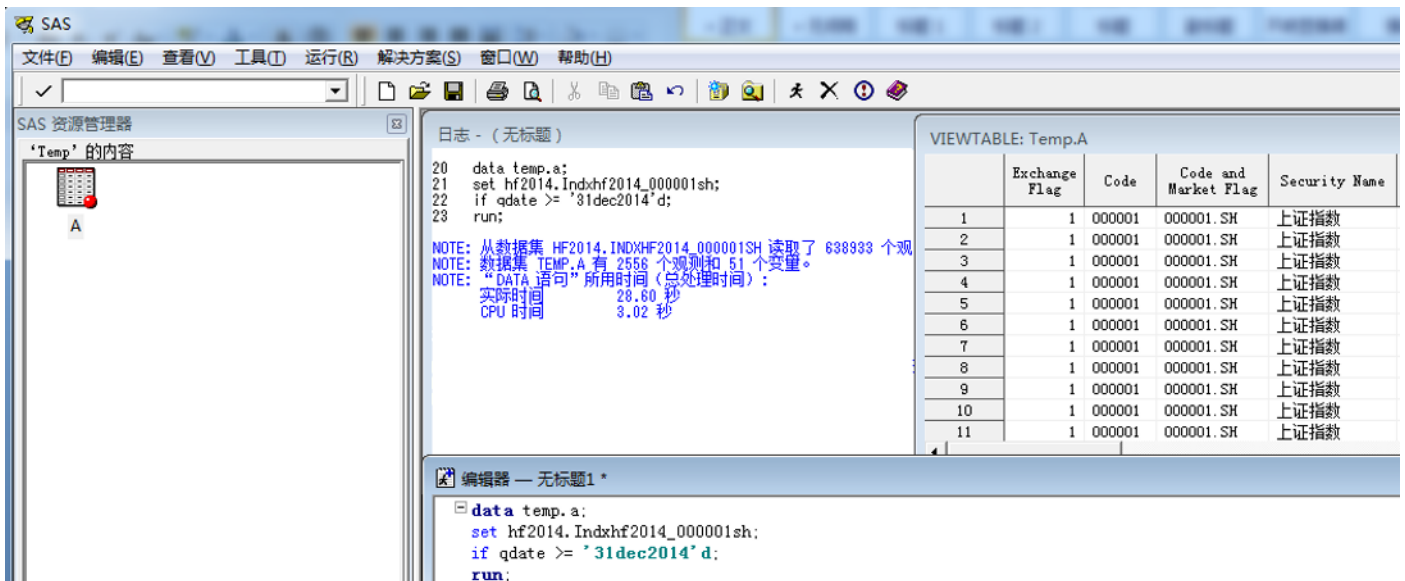


图2

示例二：以下示例查询代码为600000的上交所股票在2016年1月29日的分笔行情数据，使用mdy函数指定时间。mdy函数的三个参数分别代表 月、日、年。

```
data a;
set HF2016.StkHF2016_600000SH;
where qdate = mdy(1,29,2016);
run;
```

示例三：以下示例查询代码为010303的上交所债券在2018年8月3日至2018年8月23日的分笔行情数据

```
data a;
set HF2018.BondHF2018_010303SH;
/*逻辑库HF201808表示2018年8月的高频数据*/
/*数据集QUEUE_BondHF201808_010303SH中，QUEUE表示委托队列数据，Bond表示债券，201808表示
2018年8月份的数据，010303表示债券代码，SH表示上交所*/
where qdate >= '03Aug2018'd and qdate <= '23Aug2018'd;
/*若需要同时满足两个不同的查询条件，这里是时间既要在2018年8月3日及之后，又要在2018年8月23日及
之前。可以使用and连接两个条件*/
run;
```

示例四：以下示例演示如何处理多个不同的查询条件

```
data a;
set HF2018.BondHF2018_010303SH;
where (qdate >= '03Aug2018'd and qdate <= '23Aug2018'd) or (Trdirec = 'B');
/*时间既要在2018年8月3日及之后，又要在2018年8月23日之前；如果前面的条件不满足，但是有
OrderVol2 > 0的数据也可以*/
run;
```

为了避免高频数据的数据量过大，进而造成麻烦，我们按年进行分割，大家在取用时可以按需取用。如果需要取用的时间区间超过1年，可以进行数据的纵向合并。

示例五：以下示例查询代码为000688的深交所股票在2018年6月3日至2018年8月3日的分笔数据。2017年6月3日至2019年8月3日包含于2017年、2018年、2019年三个时间段，因此我们需要将这三个时间段内的数据集合并后进行查询。

```
data a;
set HF2017.StkHF2017_000688SZ
```



```

HF2018.StkHF2018_000688SZ
HF2019.StkHF2019_000688SZ;
where qdate >= '03Jun2017'd and qdate <= '03Aug2019'd;
run;

```

示例六：以下代码演示使用keep语句进行变量的选取

```

data a;
set HF2016.StkHF2016_600000SH;
where qdate = '29Jan2016'd;
/*假设我们仅需要code qdate qtime oppr hipr lopr, 其他的变量都不需要, 可以像如下代码那样写*/
keep code qdate qtime oppr hipr lopr;
run;

```

示例七：以下代码演示使用drop语句进行变量的选取

```

data a;
set HF2016.StkHF2016_600000SH;
where qdate = '29Jan2016'd;
/*假设我们不需要Depth1 Depth2这两个变量, 其他的变量都需要, 可以像如下代码那样写*/
drop Depth1 Depth2;
run;

```

示例八：以下代码演示数据集选项obs的用途。有时我们只想先取几条数据看看情况, 并不打算取全部, 这时可以使用数据集选项obs。

```

data a;
set HF2016.StkHF2016_600000SH(obs=10);
/*以上语句最后的括号中obs=10指定我们只需要10条观测, 大家可以根据自己的需要进行修改*/
where qdate = '29Jan2016'd;
run;

```

- 3、将数据集导出为其他格式：**如您对其他语言更为熟悉, 或希望使用其他软件进行数据处理, 您可以将SAS数据集导出为其他格式。SAS软件的数据导出过程步是proc export, 通过指定不同类型的dbms选项即可导出不同格式的文件, 支持的导出格式包括csv, Excel (xls, xlsx), MS Access数据库, SPSS数据文件, STATA数据文件等。

请注意, 如您使用rsubmit远程提交的方式在远程服务器上创建了数据集, 您需要先通过proc download过程步将数据集下载到本地并endrsubmit后再进行导出。如果直接导出, 数据集将会被导出到远程服务器的相应位置, 而不是本机。

示例一：以下代码将数据集a导出为csv文件, 导出文件位于D:/out.csv

```

proc export data=a /*data选项指定需要导出的数据集名称, 我们在前面使用的是a,
请您根据自己的实际情况进行填写*/
    outfile='D:/out.csv' /*导出文件位置, 注意文件的扩展名需要和文件类型对应, 例如
dbms=csv时, outfile文件扩展名应为csv, 如果不是, 那么SAS会在自动在后面加上.csv的后缀名*/
    dbms=csv /*指定输出格式为csv*/
    replace /*如果已有同名文件则自动覆盖, 如果不指定replace, 那么
SAS会保留原文件, 不会写入*/
;
run;

```

示例二：以下代码将数据集a导出为MS Access数据库文件, 导出文件位于D:/out.accdb, 表名为a

```

proc export data=a /*data选项指定需要导出的数据集名称, 我们在前面使用的是a,
请您根据自己的实际情况进行填写*/

```

```

        outtable='a'          /*导出后, Access数据库中该表格的名字*/
        dbms=access           /*指定输出格式为access*/
        replace               /*如果已有同名文件则自动覆盖, 如果不指定replace, 那么SAS
会保留原文件, 不会写入*/
    ;
DATABASE = 'D:/out.accdb';    /*导出文件的位置, 注意文件的扩展名需要和文件类型对应, 例如
dbms=access时, outfile文件扩展名应为accdb或mdb, 如果不是, 那么SAS会在自动在后面加上对应的
后缀名*/
run;

```

示例三：以下代码将数据集a导出为SPSS数据文件，导出文件位于D:/out.sav

```

proc export data=a           /*data选项指定需要导出的数据集名称, 我们在前面使用的是a,
请您根据自己的实际情况进行填写*/
    outfile='D:/out.sav' /*导出文件的位置, 注意文件的扩展名需要和文件类型对应, 例
如dbms=spss时, outfile文件扩展名应为sav, 如果不是, 那么SAS会在自动在后面加上.sav的后缀名*/
    dbms=SPSS               /*指定输出格式为SPSS*/
    replace                 /*如果已有同名文件则自动覆盖, 如果不指定replace, 那么SAS
会保留原文件, 不会写入*/
;
run;

```

示例四：以下代码将数据集a导出为Stata数据文件，导出文件位于D:/out.dta

```

proc export data=a           /*data选项指定需要导出的数据集名称, 我们在前面使用的是a,
请您根据自己的实际情况进行填写*/
    outfile='D:/out.dta' /*导出文件的位置, 注意文件的扩展名需要和文件类型对应, 例
如dbms=dta时, outfile文件扩展名应为dta, 如果不是, 那么SAS会在自动在后面加上.dta的后缀名*/
    dbms=dta                /*指定输出格式为dta*/
    replace                 /*如果已有同名文件则自动覆盖, 如果不指定replace, 那么
SAS会保留原文件, 不会写入*/
;
run;

```

示例五：以下代码将数据集a导出为Excel文件，导出文件位于D:/out.xlsx

请注意，在导出Excel文件时，或许会因为系统兼容性的问题而出现不能成功导出的情况，其原因可能包含本地电脑没有安装Office、Office软件的位数和SAS导出中间件的位数不一致、或SAS安装问题等多种不同情况。因此，在实际使用中，虽然可以进行尝试性操作，但请尽可能不要依赖将数据集直接导出为Excel文件这种提取方式。

```

proc export data=a           /*data选项指定需要导出的数据集名称, 我们在前面使用的是
a, 请您根据自己的实际情况进行填写*/
    outfile='D:/out.xlsx' /*导出文件的位置, 注意文件的扩展名需要和文件类型对应,
例如dbms=excel时, outfile文件扩展名应为xls或xlsx, 如果不是, 那么SAS会在自动在后面加上对应的
后缀名*/
    dbms=excel              /*指定输出格式为excel, 若excel设定不成功, 还可以尝试使
用xls或xlsx, 但是dbms=xls只能导出xls文件, dbms=xlsx只能导出xlsx文件*/
    replace                 /*如果已有同名文件则自动覆盖, 如果不指定replace, 那么
SAS会保留原文件, 不会写入*/
;
run;

```

4、向服务器提交数据处理命令并下载结果：通过第一步中的前三行命令远程登录服务器后，也可以不在本地建立

高频数据逻辑库，通过将数据处理的命令提交到服务器，再将服务器处理后的结果数据下载到本地磁盘。这种方式的好处在于不需要传输大规模的高频数据到本地，而是在服务器上处理后，直接下载结果数据集即可。通常情况下，结果数据集的大小会远远小于原始的高频数据文件。因此这样可以节约时间，提高效率。

rsubmit语句标记远程提交的开始，该语句后的所有语句，直到遇到**endrsubmit**之前，都会被提交到远程服务器上执行，本机只负责将语句上传到远程服务器，并不会执行这些语句。您可以在该代码段中编写数据处理逻辑，代码的实现上与本地没有任何不同，唯一的区别是产生的逻辑库和数据集都在远程服务器上，而不是本地磁盘，如果需要将某个数据集放到本机，需要使用**proc download**过程步进行下载。

如：通过运行以下代码即可从 2014 年所有的高频数据中取出 2014-12-31 的上证指数的所有数据并下载到本地磁盘中。

```
rsubmit; /* 开始远程提交，以后的指令将会在远程服务器上运行 */
/*以下语句在远程服务器上创建逻辑库HF201601，在您本地的电脑上不会看到该逻辑库*/
libname HF2014 'D:\RESSET\HF2014';
/*以下语句在远程服务器的work逻辑库中创建数据集a，在您本地的SAS软件中的work逻辑库不会看到a数据集*/
data a;
set hf2014.Indxhf2014_000001sh;
if qdate >= '31dec2014'd;
/*进行数据处理的方式与前一节完全一致，此处不再赘述*/
run;
/*以下语句将远程服务器的work逻辑库中的a数据集下载到本地的work逻辑库中，命名为b。该过程执行结束后，您将会在本机的work逻辑库中看到b数据集，但不会看到a数据集*/
proc download data=a out=b;
run;
endrsubmit; /* 结束远程提交，以后的指令将会在本机运行 */
```

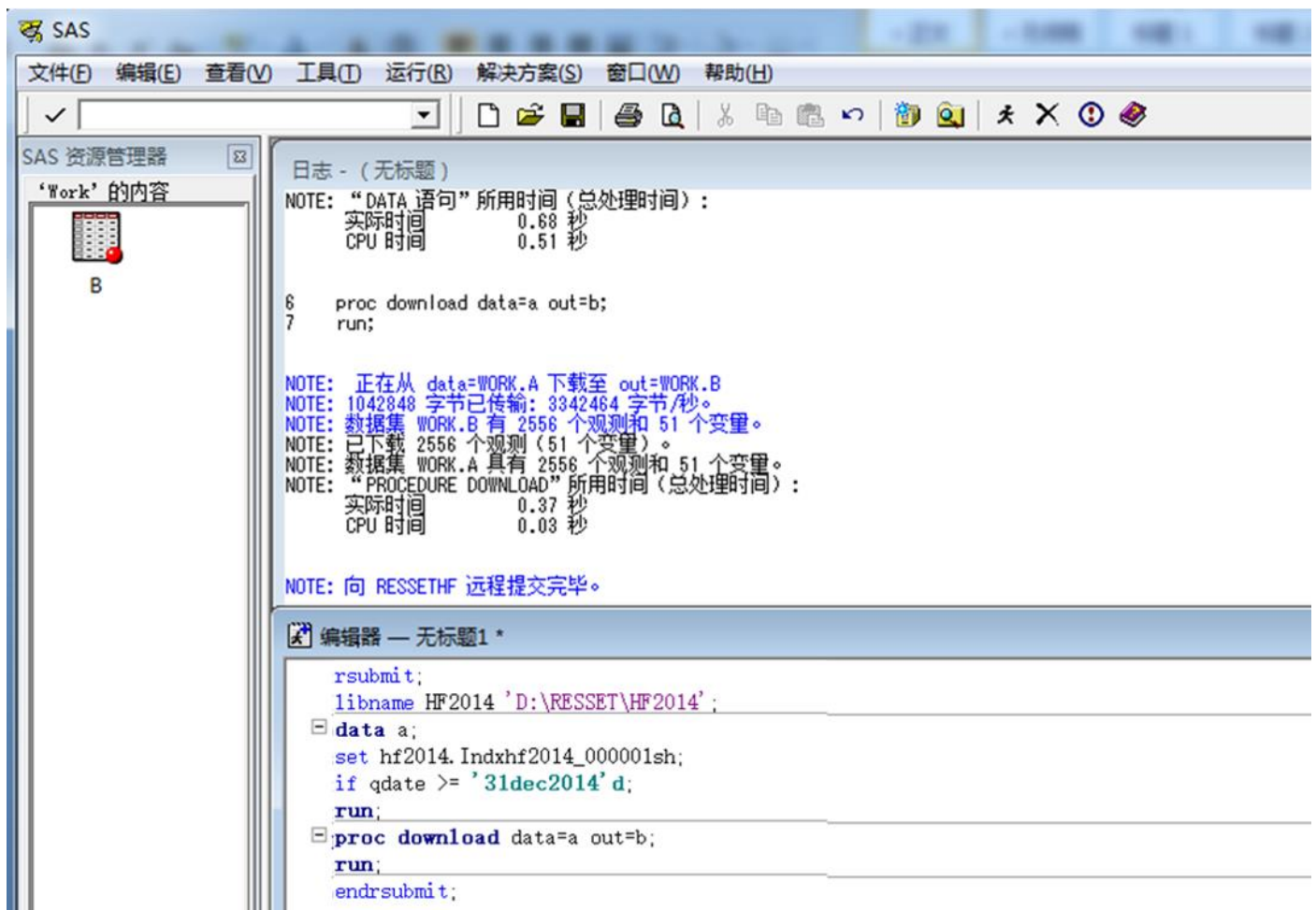
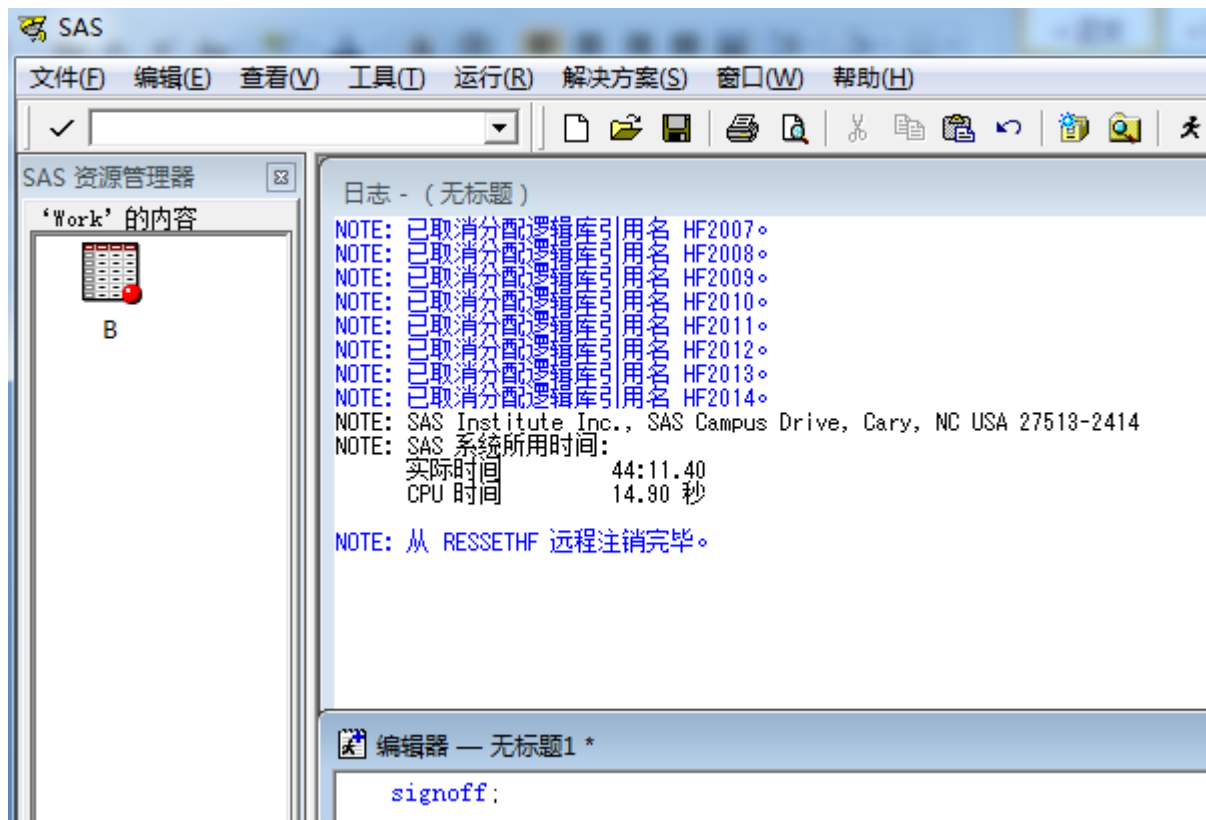


图3

- 5、**注销服务器登录：**如果不需要再进行服务器连接时，可以通过以下命令注销服务器的登录。

`signoff;`



- 6、**自动建立服务器连接：**如果客户端需要每次打开 SAS 软件后自动建立步骤 1 中服务器的远程登录或建立相关逻辑库，则可通过在客户端 SAS 的安装路径下（如：C:\Program Files\SAS\SAS 9.1）新建一个 SAS 程序，名为：autoexec.sas，并输入步骤 1 中的相关命令后保存即可。