Using Hash Tables for Case-Control Matching

Efficient Way for Control Selection from Large Administrative Data

George Zhu, Fu-Lin Wang
Alberta Health, Government of Alberta

Edmonton SAS User Group Meeting
May 8, 2013

Case Control Study Design

- Case-Control study is an observational study design, often used in epidemiology, population based public health studies
- Case-Control study can eliminate the effects of confounders (age, gender, social economic status, etc) by matching controls to a case based based on the same confounders, and focus on the main factors
- The process of this design is relatively simple: For each study case, randomly select one or more control from the non-case population with matching characteristics
- A case may have more than one matching controls, but a control can only be matched to one case

What are the difficulties?

- Need to keep track of the matching situation of both cases and control:
  - Once a control is matched to a case, the control can not be used any more
  - Once a case has enough controls, the case will not be matched to other controls any more
- There may have multiple records for a unique control with varying characteristic values (for example, different ages at different years)
- You may need to find out all possible matches: If 1000 cases all can be matched with 1 million controls, the number of all possible matches will be 1 billion - problems with storage, sorting, etc.

What SAS Tools to Use?

There are a few conventional SAS tools can be used to solve the above problems

- Proc SurveySelect: Can only handle categorical characteristics, but not for fussy matching, such as difference of age within 5 years
- Data Step: Difficult to keep track of which control has been matched or which case has enough controls, due to its sequential processing of records
- Proc SQL: Would generate huge intermediate data set, and need more sorting process
- Hash Table object: This maybe the ONLY efficient solution in SAS!!

What is Hash Table Object

What is Hash Table Object?

- It is a Data Step Component Object. Meaning:
  - It only exists in a data step - once the data step finishes running, it will no longer exist
  - It is an object - as in Object Oriented Programming (OOP): with a constructor (declaration), attributes, and methods (the dot notation)
  - It is not a SAS data set, and does not exist in external storage - it is in memory, so there is a size limit for the table, but the searching and updating are fast
- It is a special tree structure with 2 components: Key and Data
  - The Key component is mandatory, and usually it must be unique
  - The data component is for the storage of the actual data values. You may have none or multiple data fields (variables) for the data component

Define a Hash Table

The following statements are required to define a hash table, usually at the very beginning at a data step (`_n_=1`):

SAS Code: Define a Hash Table

```
declare hash cases (dataset:'AllCases');
cases.DefineKey('case_id');
cases.DefineData('Gender','Age');
cases.DefineDone();
```

- Use `Declare` to define a hash table.
  `cases` is the table name (object name), which will be used in the dot notation with methods.
  The attribute `dataset:'AllCases'` is used to fill in the table with records from the SAS data set `AllCases`.
- Use `.Definekey()` and `.DefineData()` methods to define the key and data components.
- Finish the declaration by using the `.DefineDone()` method

Hash Table Methods

Here are some of the methods with Hash Table

- Add method: Adds a record to the hash table
- Replace method: Replaces the data component with given key value
- Find method: Locate the record in the hash table base on given key value
- Remove method: Removes the records in the hash table with given key value
- Output method: Creates a SAS data set, and dump the records in the hash table to the data set
- Delete method: Delete the hash table. Same effect as exiting the SAS data step

All methods have a return value: A 0 return value indicates success, any non-zero values indicate failure or error

Hash Iterator Object

Hash Iterator object is defined on a hash table object.

- It can be used to retrieve records in hash table in either forward or reverse order
- The Find method with hash table is based on the Key value, while case-control matching is mostly based on the Data value
- 3 Hash Iterator methods can be used to scan for matching case and controls:
  - First method: Go to the first record of underlying hash table
  - Next method: Go to the next record in the hash table
  - Prev method: Go to the previous record in the hash table
- Define a Hash Iterator: use `declare hiter`

SAS Code: Define a Hash Iterator object

```
declare hiter h_cases('cases');
```

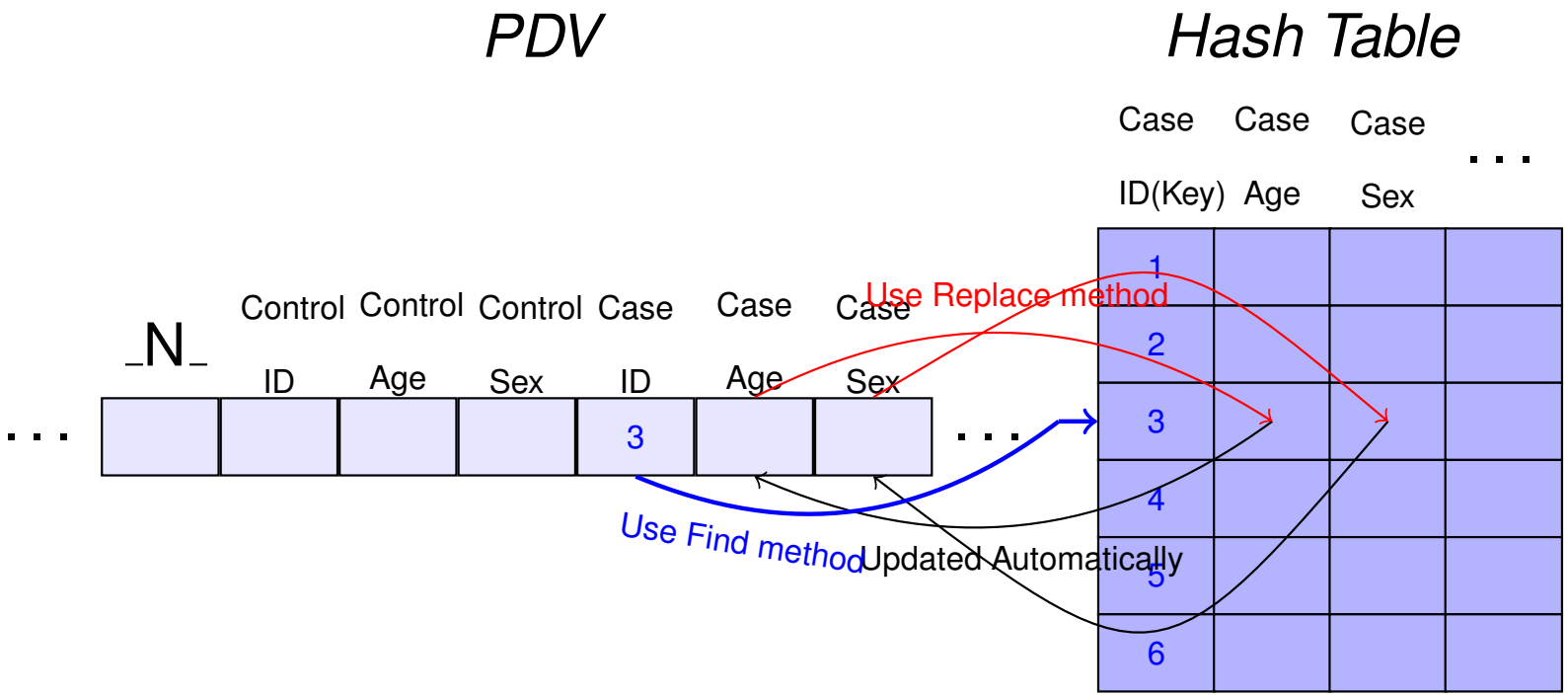
where `cases` is the underlying hash table defined before

Communication between PDV and Hash Table

Both Program Data Vector (PDV) and Hash Table are in the memory

- PDV stores data values from the input data set, processes the data values, and outputs to resulting data set, one observation at a time.
- The communication is basically through common variable name, but in different directions
  - When the current record in hash table changes, the variables in PDV with the same names in Hash table will be updated automatically
  - But when the values in PDV variables change, they will not be updated automatically in hash table
  - To update the values for the current record in hash table, use the `Replace()` method
  - To locate a record in the hash table, assign the key variable in PDV with a value, then use `Find()` method
- To avoid getting missing values, you will have to explicitly define variables used in Hash table to be used in PDV

Communication between PDV and Hash Table



Data generation

SAS Code: Generate cases and controls data sets

```
%let nCase=1000;
%let nControl=1000000;
%let ratio=5;

* Generate the Cases data set;
data cases(drop=i);
  retain id age gender;
  length gender $1;
  do i=1 to &nCase.;
    id=i;
    age=floor(ranuni(0)*100);
    gender=ifc(ranuni(0)>0.5,"F","M");
    output;
  end;
run;

* Generate the Controls data set;
data controls(drop=i);
  retain id age gender;
  length gender $1;
  do i=1 to &nControl.;
    id=i;
    age=floor(ranuni(0)*100);
    gender=ifc(ranuni(0)>0.5,"F","M");
    output;
  end;
run;
```

SAS Codes with Hash Table Objects

SAS Code: Hash Tables for Case-Control Matching

```
data Controls_H;
  set Controls;
  control_rand=ranuni(0);
  rename age=control_age gender=control_gender id=control_id;
run;
proc sort data=controls_H;  *scramble the control list for randomness;
  by control_rand;
run;
data Cases_H;
  set Cases;
  case_rand=ranuni(0); *for scramble the order of cases;
  rename age=case_age gender=case_gender id=case_id;
  count=0;              *for recording number of controls matched;
run;

* the main data step;
data _null_;
  if _n_=1 then do;
    set cases_h(obs=1);  *make the variables in Cases data set available in the PDV;

    *put the cases in a hash table;
    declare hash cases(dataset:'cases_H',hashexp:8,ordered:'y');
    cases.definekey("case_rand","case_id");
    cases.definedata("case_rand","case_id","count","case_age","case_gender");
    cases.definedone();

    declare hiter hi_cases('cases');  * declare a hash table iterator object;
```

SAS Codes with Hash Table Objects

SAS Code: Hash Tables for Case-Control Matching

```
  declare hash matches(); *declare a hash table for matched cases and controls;
  matches.definekey("case_id","control_id");
  matches.definedata("case_id","control_id");
  matches.definedone();

  *declare a hash table for recording matched controls;
  control_id_hash=case_id;
  declare hash m_control();
  m_control.definekey("control_id_hash");
  m_control.definedone();
  m_control.clear();
end;

set controls_h end=eof;
control_id_hash=control_id;          *get current control_id for searching;
if (m_control.find() ne 0) then do; *not matched to a case yet;
  rc=hi_cases.first();               *search cases table using hash iterator object;
  do while(rc=0);
    if (count<&ratio. and
        case_gender=control_gender and abs(case_age-control_age)<=5) then do;
      count+1;
      cases.replace();
      matches.add();
      m_control.add();
      leave;
    end;
    rc=hi_cases.next();
  end;
end;
```

SAS Codes with Hash Table Objects

SAS Code: Hash Tables for Case-Control Matching

```
  *check if all the cases have matches (ie, count=&ratio.);
  done=1;
  rc=hi_cases.first();
  do while(rc=0);
    if count<&ratio. then do;
      done=0;
      leave;
    end;

    rc=hi_cases.next();
  end;

  *if all the cases are matched or run out of controls, output the resulting data sets;
  if (done or eof) then do;
    matches.output(dataset:"matches");
    cases.output(dataset:"matched_cases");
    m_control.output(dataset:"matched_controls");
    stop;
  end;
run;
```

Alternative Method: Use Proc SQL and data step

SAS Code: Using Proc SQL and Data Step

```
data controls_S;
  set controls;
  rand_num=ranuni(0); *for scramble the order of controls;
run;
data cases_S;
  set cases;
  rand_num=ranuni(0); *for scramble the order of cases;
run;

proc sql;
  create table all_matches_S as
    select a.id as case_id, a.age as case_age, a.gender as case_gender,
           b.id as control_id, b.age as control_age, b.gender as control_gender,
           a.rand_num as case_rand,b.rand_num as control_rand
    from cases_S a, controls_S b
    where a.gender=b.gender and abs(a.age-b.age)<=5; *matching condition;
quit;

data all_matches_S;
  set all_matches_S;
  match_rand=ranuni(0);  *for scrambling the order cases;
run;

proc sort data=all_matches_S;
  by control_rand match_rand;
run;
```

Alternative Method: Use Proc SQL and data step

SAS Code: Using Proc SQL and Data Step

```
data all_matches_S;
  set all_matches_S;
  by control_rand;
  if first.control_rand then output;
run;

proc sort data=all_matches_S;
  by case_rand;
run;

data all_matches_S(drop=count);
  set all_matches_S;
  retain count;
  by case_rand;
  if first.case_rand then count=0;
  count=count+1;
  if count<=&ratio. then output;
run;
```

Comparison

Cases	Controls	Method	Real Time (seconds)	User Time (seconds)	System Time (seconds)	Max Used Memory	Max OS Memory
1000	1000000	Proc SQL	709.2	101.4	50.6	66983	74616
		Hash Table	6.3	1.7	0.4	64828	72560
2000	1000000	Proc SQL	1566.7	207.3	122.8	66985	76920
		Hash Table	7.2	3.8	0.3	64830	74864
3000	1000000	Proc SQL	3111.2	316.1	207.7	66983	78448
		Hash Table	14.5	8.3	0.5	64828	75888
1000	2000000	Proc SQL	1740.3	208.0	119.4	66983	77944
		Hash Table	14.5	2.4	0.6	67005	77944
2000	2000000	Proc SQL	4211.0	424.6	281.5	66983	104816
		Hash Table	12.8	4.4	0.5	67006	78200
3000	2000000	Proc SQL	8286.7	637.3	512.0	66985	79984
		Hash Table	25.6	9.0	0.7	67007	77944

Questions

Comments & Questions?