PhUSE 2013

Paper HE03

Anatomy of an Epidemiology Project

Paul Murray, FTP Software Consultants Ltd, UK

Abstract

More pharmaceutical companies are becoming involved in epidemiological studies. These studies require efficient handling of large amounts of data. This paper will follow the steps involved in a typical epidemiology study. Looking at the challenges encountered and suggesting some techniques to overcome them. This will include looking at the initial structure of vendor data, and what changes can be made to the structure of the data to make the detailed exploration of the data easier. Following this the techniques used in extracting and processing data to create a cohort and controls and storing and reporting the results.

Epidemiology/Health Outcomes

Epidemiological studies describe the distribution of health outcomes, the study of the end results of health services that takes patients' experiences, preferences, and values into account—they are intended to provide scientific evidence relating to decisions made by all who participate in health care.

Studies can:

- provide information on “real world” use and practice;
- detect signals about the benefits and risks of the use of practices in the general population;
- help formulate hypotheses to be tested in subsequent experiments;
- provide part of the community-level data needed to design more informative pragmatic clinical trials;
- inform clinical practice.

This paper will give an overview of an epidemiology/health outcome study, showing the various phases of a study with some examples of SAS code used in such a study.

Skill Requirements

The skills required for creating an end to end epidemiology projects are:

- Handling large data sets efficiently in Base SAS and the SQL procedure
- Summarise and transform data using SAS procedures
- data manipulation
- macro programming
- report writing

The programmers job is to select the required data from across the database, decoding diagnosis, procedures and medications, combine the data in a multitude of different ways and carry out the required summarisation and analysis.

Data Sources

The main sources of data for studies:

- Insurance claims data
- Medical data
  - GP data
  - Hospital Data

There are a number of vendors for this data, some vendors include:

- IMS
- Pharmetrics (IMS since 2005)
- Thompson Reuters/Truven Health
- GPRD/CPRD
- Cegedim

Each database will have a different structure and strengths and weaknesses, the database selected for a study should be decided upon as the best fit for the study design.

Structure

Databases are usually structured separating various medical functions, say for a claims type database:

- enrollment
- inpatient
- outpatient
- medication

Within the data there will be Patient Ids, Patient Codes, ICD codes for diagnosis, NDC codes for drugs, maybe long free text fields, as well as the vendors own codes for diagnosis, drug and even free text. Verified code lists need to be created for medical conditions and medications.

Sometimes data sets are created per year, meaning an extract of data will have to span multiple large data sets.

Observations may have multiple items of interest per record

- diagnosis, diag1-diag15 in one area of database say in-patient, diag1- diag5 in outpatient
- similarly
  - enrolment
  - procedure

The databases can be very large – hundreds of millions of records, and straight off the vendor disc the data is usually not indexed.

What does a study request look Like

The programmer is generally given an SAP outlining:

- background to study/study design
- Data source
- Cohort definition –
  - Inclusion/Exclusion Criteria
    - diagnosis
      - co-morbidity
    - treatment
      - procedures
      - drugs
    - study period
      - index date definition
      - pre and post index periods
    - demographics
      - Age on index date
- control patients/matching
- co-variates
- statistical analysis
- outputs – tables listings and figures

- validation procedures

Index date

Usually the reporting for an study is pre and post index date, this specified in the SAP and will be the date of the first diagnosis of interest in a specified time window and the pre and post index periods will be before and after this date so these periods will be specific to the patient. Control patients are given an index date – the date of the patient they are matched to.

Using the Data

The initial solution is to let the programmers at the data "as is" – armed with the SAP, but this can cause problems:

- projects can overlap in scope or even be about the same set of diagnosis/treatments
- extraction can be slow due to the size and structure of the data- programmers extract once and keep data
- multiple private marts build up taking up storage
- duplication of effort
  - programming
  - reporting
  - validation

Approach

The vendor data structure is not sacrosanct, there is a need to make changes to the data that are:

- efficient
- understandable
- consistent
- easily validated
- additional structure to the original data

Marts

Creating a fully functioning data warehouse may take a lot of time to implement. This solution may replace a warehouse implementation or be a stopgap solution while a data warehouse is designed and implemented.

Related sets of data that are grouped together and separated out from the main body of data.

- Easy access to frequently needed data
- Creates collective view by a group of users
- Improves end-user response time
- Ease of creation
- Lower cost than implementing a full data warehouse
- Potential users are more clearly defined than in a full data warehouse
- Contains only business essential data and is less cluttered.

The patients meeting the primary selection criteria for the cohort can be found quickly from the Marts, very useful for feasibility counts, and more detail can be found from the main tables which have been sorted an indexed by patient identifier.

The marts may take up space but because they will aid in the extraction of the data and combined with the indexed principal data speed up execution – programmers may not feel the need to create their own marts, when an extraction of any data can be made in less than an hour rather than take the whole day.

As the input data is liable to change with periodic updates, use the SAS data dictionaries or SASHELP views containing metadata about SAS libraries, this:

- keeps the code generic
- gives a starting point when

- moving between vendor databases
- creating a macro solution

See Appendix 1.

Diagnosis/Procedure

Most studies are diagnosis driven and the databases can usually be searched via an ICD code, whether for a single ICD code like restless leg, or a multitude of diagnosis codes like COPD. Pharmaceutical companies investigate the potential and outcomes of their, and their competitors, treatments in the real world and a company would only examine a finite number of therapeutic areas.

Diagnosis Marts

Requiring a way to get the relevant diagnosis for the index period quickly for multiple studies. Diagnosis marts can be created for each therapy area/disease of interest, each observation containing, as this dataset is to be a signpost to extractions from the main data, the minimum amount of data is selected:

- patient id
- diagnosis/prescription/procedure
  - Example: Heart Disease
    - Diagnosis of Heart Disease
    - Implantation of Stent or other Related Treatment
    - Prescription for ACE Inhibitors or other Heart Related Medications
- source of "diagnosis" (optional)
- date of diagnosis/event (could be just the unique event per month and year).
- resulting data is sorted and indexed on date of event.

If the diagnosis data is over many variables, as described earlier, the diagnosis is turned into one column in the resulting table.See Appendix 2.

A clinical index such as the Charlson co-morbidity index may be being used in the study, this is quite a large set of diagnosis and may be worth creating a diagnosis mart for, this could be summarised per month as the CCI is asked for during a set period maybe before and after an index date.

Medication Data

This paper is concerned with diagnosis driven studies, a set of medication driven studies could be helped by medication marts, collecting together patients, dates and medication groups.

If a set of patients is derived from the diagnosis marts, the process of selecting medications can be greatly improved by indexes, in this case the patient identifier would be a useful index.

To see if an index is being used by a SAS process, the SAS option MSGLEVEL is set to I. This will do the following:

- if an index is used create a message in the log
- suggest what can be done to influence the use of an index
- sort data in index order
- use data set options to force index use

To force the use of an index when the SAS optimizer has decided to ignore the index use one of the following data set options:

- IDXNAME=*index-name*
- IDXWHERE=YES

The resulting resource usage could be tested to find the most efficient way to process extractions.

Enrolment/Eligibility

In order for a patient's data to be included in a study, patients, from a claims database, have to be fully enrolled for the whole of the pre- and post-index periods. A summary of enrollment is made for each patient across the entire database.Studies based on non claim data, such as GP or hospital data, may ask for further eligibility criteria – such as a doctor visit, diagnostic test, or prescription event every month for the duration of the study.

A summary of enrolment/eligibility can be turned into a character string containing a 1 or zero for each month where criteria is either met or not met. The required time frame can then be queried from the string and eligible patients found quicker.

Space can be saved on the enrolment summary by storing the enrolment "string" as bits, in this way an eight year summary can be stored in 12 bytes of 8 bits rather than 96 bytes for the 1 character byte per month solution.

Obs	Patid	dtstart	dtend	enrolchar
1	9101	01JAN2009	28FEB2011	000000000000000000000000000000000000000011111111111100000000000110000000000000000000000
2	9102	01JAN2006	28FEB2011	111111111111111111111111111111111111111111111111100000000000110000000000000000000000
3	9103	01JAN2006	31DEC2008	111111111111111111111111111111111111100000000000000000000000000000000000000000000000
4	9301	01JAN2009	30JUN2012	000000000000000000000000000000000000011111111111100000000000111111111111111111000000
5	9302	01JAN2006	30JUN2012	111111111111111111111111111111111111111111111111100000000000111111111111111111000000
6	9303	01JAN2006	31DEC2008	111111111111111111111111111111111111100000000000000000000000000000000000000000000000
7	9401	01JAN2009	30JUN2012	000000000000000000000000000000000000011111111111100000000000111111111111111111000000
8	9402	01JAN2009	30JUN2012	000000000000000000000000000000000000011111111111100000000000111111111111111111000000
9	9501	01JAN2009	30JUN2012	000000000000000000000000000000000000011111111111100000000000111111111111111111000000
10	9502	01JAN2006	30JUN2012	111111111111111111111111111111111111111111111111100000000000111111111111111111000000

See Appendix 3.

Study Cohort Identified

When the cohort is identified, if extraction of data is slow the programmer may extract all data for the cohort at this stage and again at the controls/matching stage, creating a stand alone database/mart for this study.

Sometimes a programmer creating a stand alone database for a particular study cannot be avoided, if the database is going to be updated and the study has progressed to far to re-run on new data.

Controls

Studies are designed to compare the cohort with a control group using one-to-one or one to many matching. to create a control group that is comparable to a case group on one or more risk factors.

Sampling methods:

- Random Sampling
  - exclude potential controls with diagnosis/symptoms under investigation
- Pair Matching
  - same or similar characteristics as cases
- Frequency/Category matching
  - controls selected in proportion

The diagnosis and drug data marts can be used to build a list of possible candidates, sometimes this can be a huge number of patients if the diagnosis under study is a rare occurrence.

The method used for matching will be specified in the SAP, from matching on demographic data, to including patient scores on clinical indexes to using a number of variables for propensity score matching.

Cost/Burden/Resource Use

The burden of the diagnosis is heavily reported, this can be actual cost, pill burden or resource use.

In the case of pill burden and resource the SAP may require reporting on most prevalent resources rather than all medications and all procedures.

Resource use will be a breakdown of visits such as:

- GP
- Neurologists
- Cardiologists
- ER
- non ER hospitalisation
- procedures
- diagnostic tests

A diagnosis by resource can be reported to establish how/where the diagnosis in question is usually found for first time in a patient.

Daily pill burden broken down by:

- antidepressants
- antihypertensive
- sedatives...

Depending on the data the cost can be reported by:

- Overall costs
- Inpatient Hospital
- Outpatient
  - physician office
  - nursing home
  - hospital
  - ancillary services
    - dialysis
    - chemotherapies
- ER (A&E)
- drug costs
  - study diagnosis related
  - not related to study diagnosis

See Appendix 4, for cost summary and storage.

Reporting

The output section of the SAP will contain table shells, listings and chart requests – similar to clinical trials reporting but of a much wider possible scope because of the data available in some databases. These requests may include any or all of the following:

- Inclusion breakdown
- Study patients vs control :
  - Demographics
  - pre-post index:
    - Costs
    - Daily pill burden
    - co-morbidity
    - drug class
    - diagnostic tests
    - procedures

Where applicable the relevant p-values will be included in the table results.

From working on various studies, similarities between reports can be found, repeated use of routines for different results can be identified helping develop reporting macros. These can be for creating statistics,

summarising, combining and formatting results into separate table lines and then the final report.

Summary Macros

- basic frequency counts
- frequency tables
- descriptive statistics
- check class levels
- p values

Formatting and output

- output of frequency results and p values
- output of statistical results and p values
- adding in spacing lines
- titles
- generate report from resulting formatted data

See Appendix 5 for macro examples.

Statistical Analysis

A statistical model analysis may be required in the SAP, along with tests to run to see if the analysis is valid:

Ancova analysis

- Test Multicollinearity
- Homogeneity of Variance Assumption
- Homogeneity of Regression Slopes Assumption

A follow up analysis may also be specified – dependent on the results of the initial analysis., such as an Anova analysis.

Validation

A programmer will thoroughly test their own programs during development and prior to completion of a project, independent validation by someone other than the programmer is necessary. This is to prove the reported output accurately represents the original data and that reports match the SAP specification.

- Logs with no errors or warnings
- Test Specs/scripts test data
- Double programming
- code review
- output review

Utility/Metadata Macros

During a project, repeated tasks and useful tools can be identified and used as the basis for macros.

Utility macros (See Appendix 6)

- current output data sets
- run times
- delete tables

Using a combination of %sysfunc and data step dataset functions creates some very useful metadata Macros (See Appendix 7):

- variable exists
- Variable labels
- variable type
- variable format

Conclusion

Completing an Epidemiology/Health outcome studies requires bringing together many diverse elements of data manipulation and analysis and because of the real world nature of the data, the extent of reporting that is possible. Any one of the sections in this paper have been or could be the source of a more in depth presentation.

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:

Paul Murray
FTP Software Consultants Ltd
p_a_murray@yahoo.co.uk

Appendix 1:

Using SASHELP.views to find input table names

```
proc sql noprint;
/* number of tables */
   select trim(left(put(count(memname),best3.))) into : numtabs
   from sashelp.vtable
   where libname="DBASE"
     and substr(memname,1,4) = "HLTH" and substr(memname,5,1) in ("I","O");
/* names of tables */
   select memname into : tab1-:tab&numtabs
   from sashelp.vtable
   where libname="DBASE"
     and substr(memname,1,4) = "HLTH" and substr(memname,5,1) in ("I","O");
quit;
...
proc sql;
%do ix=1 %to &numtabs;
     create table ext&ix as
     select * from &&tab&ix
     where ICD10CODE in (select code from codelist);
quit;
%end;
```

Appendix 2:

For the case of multiple diagnosis variables dx1-dx*n*

```
%do ix=1 %to &numtabs;
```

```
proc sql noprint;
/* find number of dx variables */
    select trim(left(put(count(*),best3.))) into : numdx
    from sashelp.vcolumn
    where libname="TR"
      and memname="&&tab&ix"
      and substr(name,1,2) ="DIAG";
quit;
data xd.&dsetout(label="01_DIAG_Patients &sysdate &systime" keep=patid enrolid year date) ;
   set &dsetin(keep=patid enrolid year &datevar dx: rename=(&datevar=date)) ;
   if diag1="&diagcode" then output;
   %do iy=2 %to &numdx;
   else if diag&iy="&diagcode" then output;
   %end;
   drop i ;
run ;
%end;
```

Appendix 3:

creating a 12 bit string for each year

```
proc sql;
   create table xd.&dsetout._m(label="05_DIAG &sysdate &systime") as
   select patid,dobyr,sex,
   input(%do ix=1 %to 12;
        trim(left(put(enrol&ix,1.)))%if &ix ne 12 %then %do;!!%end;%end;,
        $binary16.) as enrolment length=2 format=$binary12.
   from &dsetin;
quit;
```

concatenating yearly to span the study

```
...
enrolment=input(%do ix=1 %to &numyears;
                  put(enrolment&ix,$binary12.)
                  %if &ix ne &numyears %then %do; !! %end;
                %end;
         ,$binary%eval(&numyears*12).);
```

For using the bit string to find a two year fully enrolled stretch

```
if index(put(enrolment,$binary60.),repeat("1",23));
```

Appendix 4:

Summarise costs over any grouping required:

```
proc summary data=nz_cost nway;
  class &patid &grps;
  var cost;
  format &grps &grpfmt;
  where &daterange;
  output out=&prepost._cost_&nmsuff(drop=_type_ _freq_)  sum=&prepost._cost;
run;
```

transpose to get 1 row per patient:

```
proc transpose data=&prepost._cost_&nmsuff out= tran_&prepost._cost_&nmsuff
            prefix=&prepost._;
  by &patid ;
  id &grps;
  format &grps &grpfmt;
  var &prepost._cost;
run;
```

Example of output:

Obs	PATID	pre_cost	pre_I	pre_O	pre_D	pre_S	office	othout	pathrad	pre_er
1	27181101	11516.98	11399.05	117.92	.	.	117.92	.	.	.
2	28626706	5219.07	5184.03	32.15	2.88	.	.	32.151	.	.
3	29669001	8658.38	8632.10	16.07	10.20	.	16.07	.	.	.
4	48483301	3032.81	3032.81	.	.	.	.	.	.	.
5	48657002	33560.13	33411.63	148.50	.	.	148.50	.	.	124.88
6	49513201	8648.53	8323.23	.	325.29	.	.	.	220.45	.
7	53163101	11093.39	11024.44	.	68.94	.	.	.	.	.
8	53988801	11502.81	11484.41	12.08	6.32	.	12.08	.	.	76.12
9	55753702	6734.48	6623.11	63.22	48.14	.	.	63.221	.	.
10	113361201	28786.12	28783.58	.	2.54	.	.	.	.	.
11	156011202	2156.53	2080.15	76.37	.	.	76.37	.	.	98.75
12	156642802	27915.94	27826.00	89.94	.	.	.	89.941	.	.
13	160638802	13435.92	13401.71	3.24	30.96	.	.	3.245	.	.
14	186490601	4988.75	4988.75	.	.	.	.	.	12.34	.
15	190577501	7324.73	7094.66	.	230.06	.	.	.	.	.

Appendix 5:

macro to get basic frequency counts for inclusion report, optional append to named result file:

```
%macro getcounts(order=,datasource=,label=,output=);
proc sql;
  create table row_&order as
  select &order as order,"&label" as label length=120,count(distinct patid) as count
  from &datasource;
quit;
%if &output ne %then %do;
 proc append base=&output data=row_&order;
 run;
%end;
%mend getcounts;
```

Part of a macro to get P-Values:

```
proc freq data =&data;
%if %upcase(&exact) eq EXACT %then %do;
  tables &class*&var / chisq fisher;
  ods output FishersExact=&out(rename=(nvalue1=prob) where=(index(label1,"Pr <= P")>0));
%end;
%else %do;
  tables &class*&var / chisq;
  ods output chisq=&out( where=(statistic="Chi-Square"));
%end;
%if &format ne %then %do;
  format &var &format;
%end;
run;
```

Appendix 6:

At any stage of programming, find all permanent program output data sets, their names and source programs, and by reverse engineering – not needed datasets.

```
%macro dsetlist(lib=,nlabel=);
proc sql;
   create table sasuser.dsets_&sysdate  as
   select *
   from sashelp.vtable
   where libname="%upcase(&lib)" and memlabel is &nlabel
   order by  crdate descending
;
quit;
%mend;
%dsetlist(lib=xd,nlabel=null);
%dsetlist(lib=xd,nlabel=not null);
```

Part of Macro to find run times of jobs from SAS Log

```
   infile folder(&prog..log) firstobs=%eval(&records-6) obs=%eval(&records-2)                    truncover
recfm=v lrecl=256;
   retain prog "&prog";
   input line $256.;
   template="00:00:00";
   if index(line,"real time") then do;
     dotpos=index(line,".");
     rtpos=index(line,"real time");
     timedata=left(substr(line,rtpos+9,dotpos-(rtpos+9)));
     substr(template,8-length(timedata)+1,length(timedata))=timedata;
     time=input(template,hhmmss8.);
     output;stop;
   end;
   format time time8.;
```

Appendix 7:

Metadata macros:

```
%macro varexist(dataset,variable);
  %let dsid=%sysfunc(open(&dataset)) ;
  %sysfunc(varnum(&dsid,&variable))
  %let dsid=%sysfunc(close(&dsid)) ;
%mend varexist;
```

```
%if %varexist(&dset,&CHAR1)>0 %then %do;

%macro varlabel(dataset,variable) ;
  %let dsid=%sysfunc(open(&dataset)) ;
  %let varnum=%sysfunc(varnum(&dsid,&variable)) ;
  %sysfunc(varlabel(&dsid,&varnum))
  %let dsid=%sysfunc(close(&dsid)) ;
%mend varlabel ;

retain col1 "%varlabel(&srcedata,&linevar)";

%macro vartype(dataset,variable) ;
  %let dsid=%sysfunc(open(&dataset)) ;
  %let varnum=%sysfunc(varnum(&dsid,&variable)) ;
  %sysfunc(vartype(&dsid,&varnum))
  %let dsid=%sysfunc(close(&dsid)) ;
%mend vartype ;

%if %vartype(&out,&mainvar)=N %then %do;

%macro varfmtname(dataset,variable) ;
  %let dsid=%sysfunc(open(&dataset)) ;
  %let varnum=%sysfunc(varnum(&dsid,&variable)) ;
  %sysfunc(varfmt(&dsid,&varnum))
  %let dsid=%sysfunc(close(&dsid)) ;
%mend varfmtname ;

char=put(&linevar,%varfmtname(&grpdata,&linevar));
```