SAS Linear Model Demo

Yupeng Wang, Ph.D, Data Scientist

Overview

SAS is a popular programming tool for biostatistics and clinical trials data analysis. Here I show an example of using SAS linear regression models to detect weight change-associated genes from a microarray dataset with gender and race factors corrected. Microarray data contain expression profiles of tens of thousands of genes in different samples or under multiple conditions, so the dataset of this demo is big. My SAS program shows a general procedure to deal with big datasets. The source code can be downloaded from https://github.com/wyp1125/SAS-Biostatistics-Toolset/blob/master/microarray_glm.sas.

Key techniques: SAS, macro, macro variable, importing data, proc format, proc contents, proc sql, proc transpose, merging datasets, do loop, p-value, multiple testing correction, FDR

Detailed procedure

1. Raw data

A small part of the raw data is available at https://github.com/wyp1125/SAS-Biostatistics-Toolset/blob/master/exp_data.xlsx. At the top are the metadata rows, starting with '!". As we want to build linear regression models of weight change on gene expression value, gender and race, we need to extract weight change, gender and race values from the metadata.

!Sample_characteristics_ch1	6 month weight change: 4%	6 month weight change: 7%	6 month weight change: 2%	6 month weight change: 23%
!Sample_characteristics_ch1	gender: Female	gender: Female	gender: Male	gender: Male
!Sample_characteristics_ch1	race: African American	race: Caucasian	race: Caucasian	race: African American

The row starting with "ID_REF" contains sample IDs, while subsequent rows are expression values, with the first column containing gene (probe_set) IDs.

ID_REF	GSM819096	GSM819097	GSM819098	GSM819099
7896736	6.59179	6.014724	6.0722923	6.291737
7896738	6.1537805	5.184265	5.538695	5.8583765
7896740	6.2080164	5.178695	5.5158367	5.793412
7896742	7.721287	7.0956464	7.9055643	8.23948

2. Importing data

```
proc import datafile='/folders/myfolders/gene/exp_data.xlsx' out=temp
    dbms=xlsx replace;
    sheet=Sheet2;
    getnames=NO;
run;
```

Because column names are not imported, A-AA are automatically assigned to them.

3. Processing raw data to generate weight change, gender, race and gene expression datasets

We first define some formats to facilitate conversion of gender and race values:

```
proc format;
    value $ frace
    'African American'=1
    'Caucasian'=2;
    value $ gd
    'M'=1
    'F'=2;
run;
```

We divide raw data into three datasets: sam_ch for metadata, sam_id for sample IDs, and exprs for gene expression:

```
data exprs sam_ch sam_id;
    set temp;
    if anydigit(substr(A,1,1),1)>0 then
    do;
        rename A=id;
        output exprs;
    end;
    else if find(A,"characteristics")>0 then
        output sam_ch;
    else if A='ID_REF' then
        output sam_id;
run;
```

Retrieve column names from exprs and save them in dataset vars, and then create new column names to facilitate data type conversion:

```
proc contents data=exprs out=vars(keep=name type);
run;

data vars;
    set vars;
    if type=2 and name ne 'id';
        newname=trim(left(name))||"_n";
run;
```

Assign old and new column names to macro variables:

```
proc contents data=exprs out=vars(keep=name type);
run;

data vars;
   set vars;
   if type=2 and name ne 'id';
      newname=trim(left(name))||"_n";
run;
```

Convert character data type to numeric data type for weight change and save the weight change data as the dataset pheno:

```
data pheno;
   set sam_ch;
   where B contains 'weight';
   id="weight";
   array ch(*) $ &c_list;
   array nu(*) &n_list;
   do it=1 to dim(ch);
      nu(it)=input(substr(ch(it),24),percent.);
   end;
   drop it &c_list;
   rename &renam_list;
   run;drop it &c_list;
   rename &renam_list;
run;
```

Generate the dataset gender:

```
data gender;
   set sam_ch;
   where B contains 'gender';
   id='gender';
   array ch(*) $ &c_list;
   do it=1 to dim(ch);
      ch(it)=trim(left(substr(ch(it),9,1)));
   end;
   format &c_list gd.;
   drop it;
run;
```

Generate the dataset race:

```
data race;
   set sam_ch;
   where B contains 'race';
   id="race";
   array ch(*) $ &c_list;
   do it=1 to dim(ch);
      ch(it)=trim(left(substr(ch(it),6)));
   end;
   format &c_list frace.;
   drop it;
run;
```

Convert character data type to numeric data type for gene expression values and save gene expression data as the dataset exprs1:

```
data exprs1;
   set exprs;
   array ch(*) $ &c_list;
   array nu(*) &n_list;
   do it= 1 to dim(ch);
      nu(it)=input(ch(it),8.);
   end;
   drop it &c_list;
   rename &renam_list;
run
```

4. Merging weight change, gender, race and gene expression datasets into one dataset for statistical analysis

Transpose individual datasets:

```
proc transpose data=pheno out=pheno1 (drop=_label_);
   var &c_list;
   id id;
run;

proc transpose data=gender out=gender1 (drop=_label_);
   var &c_list;
   id id;
run;

proc transpose data=race out=race1 (drop=_label_);
   var &c_list;
   id id;
run;

proc transpose data=exprs1 out=exprs2 (drop=_label_);
   var &c_list;
   id id;
run;
```

Sort individual datasets:

```
proc sort data=pheno1;
   by _name_;
run;

proc sort data=gender1;
   by _name_;
run;

proc sort data=race1;
   by _name_;
run;

proc sort data=exprs2;
   by _name_;
run;
```

Merge individual datasets into one dataset stat for statistical analysis:

```
data stat;
   merge pheno1 gender1 race1 exprs2;
   by _name_;
run;
```

In dataset stat, rows represent samples, while columns represent weight change, gender, race and genes.

5. Deriving some macro variables to facilitate statistical analysis

Output the variables of the stat dataset:

```
proc contents data=stat out=stat_var (keep=name) noprint;
run;
```

Compute the total number of genes and assign it to a macro variable nprob:

```
data _NULL_;
   dsid = open("stat_var");
   obss = attrn(dsid,"NLOBS")-4;
   call symput('nprob',compress(put(obss,5.)));
run;
```

Retrieve all gene names and assign them to macro variables:

```
proc sql noprint;
   select name into: pid1- :pid&nprob from stat_var
   where name not in ('_name_','gender','race','weight');
run;
```

6. Statistical analysis: detecting weight change-associated genes

Make a macro comput to

1) loop through each gene (columns of stat) to build a linear regression model of weight change on gene expression value, race and gender, and retrieve the p-value for the coefficient of gene expression value.

2) save the names and p-values of all genes as a dataset allp.

```
%macro comput;
   %do it=1 %to &nprob;
      proc glm data=stat outstat=ttt noprint;
         class gender race;
         model weight=race gender &&pid&it;
      run;
      data _NULL_;
         ln=4;
         set ttt point=ln;
         call symputx("pva"||compress(put(&it,5.)),prob);
      stop;
      run;
   %end;
   data allp;
      %do ii=1 %to &nprob;
         gene="&&pid&ii";
         pvalue="&&pva&ii";
        output;
      %end;
   run;
%mend;

%comput;
```

Process the allp dataset for multiple testing correction:

```
data allp1;
   set allp;
   raw_p=input(pvalue,best.);
   drop pvalue;
run;
```

Correct p-values by the FDR approach:

```
proc multtest inpvalues=allp1 FDR out=allp_adj;
run;
```

Print the genes associated with weight change according to FDR-adjusted p-value<0.1:

```
proc print data=allp_adj;
where fdr_p<0.1;
run;
```