

## Paper TT07

### Automating the process of choosing among highly correlated covariates for multivariable logistic regression

Michael C. Doherty, i3 Drug Safety, Waltham, MA  
Xiaochun Zhang, i3 Drug Safety, Waltham, MA

#### **Abstract**

In observational studies, there can be significant differences between the characteristics of a treatment and a control group. To reduce the potential for confounding, controls are matched to members of the treatment group using propensity scores estimated by multivariable logistic regression analyses. Propensity score modeling can involve the inclusion of hundreds of covariates including patient diagnoses, medical procedures, and medication exposures, highly correlated variables can complicate the multivariable logistic regression. This paper describes a statistical method to remove the variables automatically, with little input from the programmer. By utilizing the R statistic output from PROC CORR, followed by a MACRO to select which variables to keep and which ones to remove from the model, the programmer can save time in selecting the covariates to be used in the model statement in PROC REG.

#### **Introduction**

In observational studies, there can be substantial differences between the characteristics of a treatment and a control group. Propensity score matching is a multivariable technique that can achieve a high degree of balance between the comparison groups, producing groups that have very similar patterns of a large number of key variables, and thus, reducing the potential for confounding. Propensity score modeling can involve the inclusion of highly correlated variables which can complicate the multivariable logistic regression model. Instead of removing the highly correlated variables by hand, a statistical method has been developed to remove those variables automatically, with little input from the programmer.

#### **Describe Example**

In this example, the covariates under consideration are 0,1 flags indicating whether a subject had a particular diagnosis, procedure, or pharmacy dispensing during the baseline period. When two variables are highly correlated, it is often better to remove the covariate which occurs less frequently. The program described below selects which variable to retain in the regression model by choosing the factor with a larger absolute value of the R statistic and a higher prevalence in the study population.

#### **Describe Method**

The programmer's task is two fold. First, identify the variables that are highly correlated and second, remove the offending covariates using an iterative procedure.

To conceptualize the process, the table below shows the highly correlated covariates in descending order of their R statistic.

Covariate 1	Covariate 2	R Statistic
VarA	VarB	0.967
VarC	VarD	0.945
VarB	VarE	0.931
VarA	VarF	0.903
...	...	...
VarZ	VarB	0.715

In this example, let's assume we wish to keep VarA as a covariate in the model. Since VarA and VarB are so highly correlated, we would like to remove VarB from consideration. We would also like to remove VarF, since it is also highly correlated with VarA. Note how VarB is highly correlated to VarE and VarZ. Since VarB is being removed from consideration, we do not necessarily wish to remove either VarE or VarZ, unless they are also highly correlated with VarA.

After removing all variables that are highly correlated with VarA, we will then move onto VarC and find any variables that are highly correlated with it and remove them. The selection macro will loop through the list until it has worked its way through the entire list of variables and has removed the offending highly correlated variables.

The program creates a list of variables in order from the highest to lowest R value. However, we also need to choose which variable should be in the left hand column (Covariate 1) and which variable should be in the right hand column (Covariate 2). Since these covariates are indicators, the selection macro chooses those variables that occur more often in our sample to be in the left hand column, and thus, are more likely to be retained. For instance, in the example above, VarA is kept while VarB is removed because VarA occurs more often.

Now that we know what we want to do, we can begin setting up our program to do it. Our first step is to create some global macros. In this example, we set up a macro for the dataset containing the covariates (dt), the lower limit of the R statistic we are interested in (HighCorr), and the variables we wish to exclude from consideration (exvar), including any continuous variables (contv).

```
%let dt=outcomes;
%let HighCorr=0.7;
%let exvar=indv_id cohort i;
%let contv=scnddiabdxdt scnddysldxdt diabrxdt dyslrxdt diaboutdt
dysloutdt;
```

Since we are assessing hundreds of covariates, we use PROC CONTENTS to create our variables list. As a precaution, we select only those variable where=(type = 1), i.e., numeric. Create a dataset (varlabel) with the variables and their labels for later use using PROC SORT.

```
proc contents data=in.&dt(drop=&exvar &contv) noprint
```

```

    out=dtvname(keep=name label nobs type where=(type=1));
proc sort data=dtvname(rename=(name=vname))
    out=varlabel(keep=vname label);
    by vname;
run;

```

Next create macro variables for the total number of variables (totvar) and the total number of observations (totobs).

```

data _null_;
    set dtvname end=last;
    call symput('var' || left(put(_n_,4.)), trim(left(name)));
    if last then do;
        call symput('totvar', left(put(_n_,4.)));
        call symput('totobs', left(put(nobs, 12.)));
    end;
run;

```

Now create a macro that will create the list of variables (varlst) and a list of string variables (Mlst) v1 ... vN (where N is the total number of variables under consideration) that correspond to the variable names.

```

%macro varlst;
    %do i=1 %to &totvar;
        &&var&i
    %end;
%mend varlst;

%macro Mlst;
    %do i=1 %to &totvar;
        v&i="&&var&i";
    %end;
%mend;

```

Use PROC CORR to obtain the R statistic for each pairing. The output should keep only the R statistic (i.e., drop the N, MEAN and STD observations from the output dataset).

```

proc corr noprint data=finaldt
    out=temp(where=(type_ not in('MEAN', 'STD', 'N')));
    var %varlst;
run;

```

Now create a table of pairings and sort by the R statistic in descending order. Remove any pairs whose R value is below our lower limit (highcorr), as well as any pairing with a R value of one (e.g. the R statistic of VarA with itself equals one). Only three variables are retained. The variable ‘\_name\_’ is changed to CorrVar1. CorrVar2 is set to the variable name of the variable that is highly correlated with CorrVar1. CorrVal is an estimate of the R statistic.

```

data CorrTemp(keep=CorrVar1 CorrVar2 CorrVal);
    set temp(drop=_type_ rename=(name_=CorrVar1));
    %mlst;
    array ChkCorr(&totvar) %varlst;

```

```

array VN(&totvar) $ v1-v&totvar;
do i=1 to dim(chkcorr);
  if (chkcorr(i) >= abs(&highcorr)) and (chkcorr(i) ne 1) then
do;
  CorrVar2=vn(i);
  CorrVal=chkcorr(i);
  output;
end;
end;
run;
proc sort data=corrtemp;
  by descending corrval;
run;

```

At this point, we have duplicates in our dataset. The dataset CorrTemp looks like the following:

CorrVar1	CorrVar2	CorrVal
VarA	VarB	R1
VarB	VarA	R1
VarC	VarD	R2
VarD	VarC	R2
...	...	...

Note how we have the correlation between VarA and VarB in observation one and the correlation between VarB and VarA in observation two. The same holds for VarC and VarD in observations three and four. Remove those duplicates using the lag function. Note that pairs will have increasing odd values (1, 3, 5, etc...).

```

data ChkDup(keep=CorrVar1 CorrVar2 CorrVal pairs);
set corrtemp;
Name1=lag(corrvar1);
Name2=lag(corrvar2);
pairs=_n_;
if name1=corrvar2 and name2=corrvar1 then delete;
run;

```

Create a frequency count for each indicator flag for use in the selection criteria. Transpose the resulting dataset for ease of merging.

```

proc summary data=finaldt;
var %varlst;
output out=sumdt(drop=_freq_ _type_) sum= ;
run;
proc transpose data=sumdt out=transdt; run;

```

Split up the pairs and merge the counts onto the two resulting datasets (dt1 and dt2). We will use the 'pairs' variable to match our correlated pairings later on. Set the two datasets (dt1 and dt2) together to create our working dataset (one).

```

proc sql;
create table dt1 as

```

```

select a.corrvar1 as vname, a.corrval, b.coll as Count, a.pairs
from chkdup a, transdt b
where upcase(a.corrvar1)=upcase(b._name_);
quit;
proc sql;
create table dt2 as
select a.corrvar2 as vname, a.corrval, b.coll as Count, a.pairs
from chkdup a, transdt b
where upcase(a.corrvar2)=upcase(b._name_);
quit;
data one;
set dt1 dt2;
run;

```

The macro 'chkrcd' selects which variables to keep and which ones to remove from consideration in the modeling process. Any highly correlated pairs are ordered by their frequency of occurrence and the covariate which occurs more often is selected to be retained, and the other is set to be deleted.

```

%macro chkrcd;
%let i=1;
%let N=1;

proc datasets;
delete basedt deletedt;
run;

%do %while (&N > 0);
* Sort in descending order by R statistic, pairs and
frequency;
proc sort data=one;
by descending corrval pairs descending descending count;
run;
* Set PreNM to previous name and PrePair to previous pair
using lag function;
data one;
set one;
PreNM=lag(vname);
PrePair=lag(pairs);
run;
* During actual runs, you probably want to comment out print
statements;
proc print data=one;
title "Dataset at Loop &i";
run;

* CREATE TWO DATASETS: KEEP (kdt&i) AND DELETE(ddt&i);
data kdt&i(keep=rcd rename=(rcd=vname)) ddt&i(keep=rcd
rename=(rcd=vname));
set one;
length str krcd drcd $200 kbase dbase $30;
retain krcd drcd kbase dbase;
* 1ST RECORD IS ALWAYS KEPT;
* kvar will contain the list of variables to keep;

if _n_=1 then do;

```

```

        kbase=vname;
        call symput('kvar', left(trim(uppercase(vname))));
        krcd=symget('kvar');
        rcd=vname;
        output kdt&i;
    end;
else if _n_=2 then do;
    * 2ND RECORD IS ALWAYS DELETED;
    * dvar will contain the list of variables to delete;
    dbase=vname;
    call symput('dvar', left(trim(uppercase(vname))));
    drcd=symget('dvar');
    rcd=vname;
    output ddt&i;
end;
else do;
    * For _n_ ge 3, we need to make sure we are working on the
lines where pair = prepair. Then we search for the variable we
are keeping (krcd). If vname matches up with krcd, then set rcd
to prenm. Or if prenm matches with krcd then set rcd to vname;
    if (indexw(krcd,uppercase(vname)) > 0 or
indexw(krcd,uppercase(prenm)) > 0) and pairs=prepair then do;
    * Check to make sure the variable we are about to put into
the deleted variable list should not be kept;
        if indexw(krcd,uppercase(prenm)) > 0 and vname ne kbase
then rcd=vname;
        if indexw(krcd,uppercase(vname)) > 0 and prenm ne kbase
then rcd=prenm;
    * ADD NEW DELETING VARIABLE INTO MACRO VARIABLE LIST;
    str=symget('dvar')||'|' ||left(trim(uppercase(rcd)));
    call symput('dvar', left(trim(str)));
    * REMOVE THIS VARIABLE FROM KEEPING VARIABLE LIST;
    str=symget('kvar');
    str=tranwrd(str,uppercase(rcd),'');
    call symput('kvar', left(trim(str)));
    drcd=symget('dvar');
    * Output to deleted dataset (ddt&i) if rcd ne kbase;
    if rcd ne kbase then do;
        output ddt&i;
    end;
end;
    * Look for drcd (deleted variable from observation 2);
    else if (indexw(drcd,uppercase(prenm)) > 0 or
indexw(drcd,uppercase(vname)) > 0) and pairs=prepair then do;
        if indexw(drcd,uppercase(prenm)) > 0 and
indexw(krcd,uppercase(vname))=0 then rcd=vname;
        else if indexw(drcd,uppercase(vname)) > 0 and
indexw(krcd,uppercase(prenm))=0 then rcd=prenm;
        str=symget('dvar');
        * ADD NEW KEEPING VARIABLE TO LIST IF THIS VARIABLE IS
NOT IN THE DELETING LIST;
        if indexw(str,uppercase(rcd))=0 then do;
            str=symget('kvar')||'|' ||left(trim(uppercase(rcd)));
            call symput('kvar', left(trim(str)));
            krcd=symget('kvar');
            if rcd ne dbase then do;
                output kdt&i;
            end;
        end;
    end;
end;

```

```

        end;
    end;
end;
end;
run;
* Sort dataset one by vname and make sure there are no duplicates
or blanks in keeper dataset (kdt&i);
proc sort data=one;
    by vname;
proc sort nodupkey data=kdt&i;
    where vname ne '';
    by vname;
run;
* You may want to comment out print statements after debugging;
proc print data=kdt&i;
    title "Keep Records from Loop &i";
run;
* Make sure there are no duplicates or blanks in deleted dataset
(ddt&i);
proc sort nodupkey data=ddt&i;
    where vname ne '';
    by vname;
run;
* You may want to comment out print statements after debugging;
proc print data=ddt&i;
    title "Delete Records from Loop &i";
run;

    * UPDATE THE CHECKING FILE, REMOVE THE RECORDS IN KEEP/DELETE
FILES;
    * CREATE A SELECTED VARIABLE FILE;
data cdt one;
    merge kdt&i(in=in1) ddt&i(in=in2) one(in=in3);
    by vname;
    * Put into cdt if variable is a keeper, but not in deleted
dataset.;
    if in1=1 and in2=0 then output cdt;
    * If no determination has been made (i.e. not in keeper or
deleted dataset, output to dataset one;
    else if in1=0 and in2=0 and in3=1 then output one;
run;
    * Again, make sure there are no duplicates in dataset to
keep;
proc sort data=cdt(keep=vname) nodupkey;
    by vname;
run;
    * Append list of variables to keep onto dataset called
basedt;
proc append base=basedt data=cdt;
run;
    * Append list of variables to delete onto dataset called
deletedt;
proc append base=deletedt data = ddt;
run;

    * Make sure dataset one is not empty. If it is, then stop
loop;

```

```

data _null_;
  call symput('N', left(put(num,8.)));
  if not 0 then set one nobs=num;
  stop;
run;
* increment counter;
%let i=%eval(&i+1);
%end;

%mend;
%chkrcd;

```

The 'basedt' dataset contains our list of variables to keep. Make sure to remove any duplicates and reattach the labels to complete the process.

```

proc sort nodupkey data=basedt;
  by vname;
run;
data sel_var_vs;
  merge basedt(in=in1) varlabel;
  by vname;
  if in1;
proc print data=sel_var_vs;
  title "***** High (&highcorr) Correlation Variables Listing *****";
run;

```

The dataset 'deletedt' contains the list of variables we are removing. Remove any duplicates and reattach labels to complete the process.

```

proc sort nodupkey data=deletedt;
  by vname;
run;
data del_var_vs;
  merge deletedt(in=in1) varlabel;
  by vname;
  if in1;
proc print data=del_var_vs;
  title "***** High (&highcorr) Correlation Variables Deleted *****";
run;

```

## Conclusion

Propensity score modeling can use hundreds of covariates; however, not all of them are necessary. By eliminating the highly correlated variables from the logistic regression model, the process can run more efficiently. Selecting which variable to include between the highly correlated pairs is done by listing the correlated covariates by descending order of their R statistic and then choosing the variable which occurs more frequently. Often the programmer will want to examine the pairs that are retained and excluded, and thus, all covariates are listed in the output. The macro presented can be used to automate the process of removing the highly correlated covariates from the dataset before running the model and may increase efficiency.