

```

/*****
/* Project: Prometheus          date: 04/15/08          */
/* pgm: DIAB_Playbook_Macros for Analysis & Modeling.sas          */
/*                               */
/* This program is part of the task to develop the Diabetes          */
/* episode case rate (ECR) using multiple linear regression          */
/* modeling.                               */
/*                               */
/* The current program contains specialized macros for the Diabetes          */
/* ECR analysis. The macros are used by the program: DIAB_Playbook_Modeling.sas. */
/*                               */
/* List of macros contained in this program:
  TDATA - run t-test and creates a dataset containing t-test results.
          The datasets can be concatenated and exported to Excel.
  NOTDATA - creates a dataset with cell counts for a dichotomous variable.
            Use when a t-test can't be done (i.e. one of the cells has
            no data. Creates a dataset to be merged with datasets created
            by TDATA.
  REGMOD - run linear regression model and optionally export to excel
            worksheet. Intended for models where a variable selection
            procedure is not used. Also creates dataset containing
            regression diagnostics. Calls GET_COUNT MACRO.
  GET_COUNT - calculates the cell counts for all dichotomous independent
              variables used in a regression model. It adds the counts to the
              ParameterEstimates ODS output dataset from proc reg. This macro
              is called by REGMOD macro and is not intended to run on its own.
*****/

```

```

***** Part I: TDATA *****;

```

```

* MACRO TDATA;

```

```

* PURPOSE: Macro to run t-test on a transformed variable and put the t-test p-values
and the untransformed means and standard deviations by the class variable in a dataset.

```

```

* NOTES:

```

```

(A) The class variable must be a dichotomous variable coded 0/1.

```

```

(B) In the output dataset, variable names end in 0 or 1 if they apply to class=0 or
class=1, respectively.

```

```

(C) This macro creates 1 dataset per t-test. The output dataset is called c&code.

```

```

These datasets can be concatenated and then output to Excel, but this must be done
outside the macro. Also, the macro cannot add labels for the class variable
to the output dataset, so this must be done outside of the macro.

```

```

* MACRO VARIABLES:

```

```

ds = dataset containing the variables to analyze (i.e. class variable and continuous
variable)

```

```

class = class variable (must be numeric 0/1 variable)

```

```

var = continuous dependent variable

```

```

nonlog = continuous dependent variable (assumed to be on the original scale, not logged)

```

```

code = Unique code to put in output dataset name

```

```

;

```

```

%MACRO TDATA(ds=,class=,var=,nonlog=,code=);

```

```

* (1) Run t-test;

```

```

ods listing exclude all;

```

```

proc ttest data=&ds;
  class &class;
  var &var;
  ods output statistics=stats TTests=tests Equality=vari;
run;

* (2) Run t-test just to get means on non-log scale (dont use t-test results otherwise
  becасue t-test will be based on logged variable);
proc ttest data=&ds;
  class &class;
  var &nonlog;
  ods output statistics=nonlog_stats;
run;
ods listing exclude none;

* (3) Put descriptive statistics for dependent variable in datasets;
data stats0 stats1;
  set stats (keep=Variable Class N Mean StdDev rename=(variable=v));

  * Make Variable have longer length;
  length variable $ 25;
  variable=v;
  drop v;

  if substr(class,12,1)='0' then output stats0;
  if substr(class,12,1)='1' then output stats1;
run;

* (4) Put non-logged descriptive statistics for dependent variable in datasets;
data nonlog_stats0 nonlog_stats1;
  set nonlog_stats (keep=Class N Mean StdDev);

  * Make Variable for merging. Make it the logged dependent variable;
  length variable $ 25;
  variable="&var";

  * Output data into separate files for each class;
  if substr(class,12,1)='0' then output nonlog_stats0;
  if substr(class,12,1)='1' then output nonlog_stats1;
run;

* (5) Put Pooled and Satterthwaite t-test info (t- value, p-value etc.) into separate
  datasets;
data tests0 tests1;
  set tests (keep=Variable Method Variances tValue df Probt rename=(variable=v));
  format df 8.;

  * Make Variable have longer length;
  length variable $ 25;
  variable=v;
  drop v;

  if Method='Pooled' then output tests0; * Assumes equal variance;
  if Method='Satterthwaite' then output tests1; * Assumes unequal variance;

```

run;

* (6) Combine descriptive stats, with t-tests and equality of variance tests;

data c&code;

```
merge nonlog_stats0 (keep=Variable Mean StdDev rename=(Mean=Mean0 StdDev=StdDev0))
      nonlog_stats1 (keep=Variable Mean StdDev rename=(Mean=Mean1 StdDev=StdDev1))
      stats0 (keep=Variable N Mean StdDev rename=(N=N0 Mean=Mean_Ln0 StdDev=StdDev_Ln0))
      stats1 (keep=Variable N Mean StdDev rename=(N=N1 Mean=Mean_Ln1 StdDev=StdDev_Ln1))
      tests0 (keep=Variable tValue DF Probt rename=(tValue=tValue_EQ DF=DF_EQ Probt=Probt_EQ))
      tests1 (keep=Variable tValue DF Probt rename=(tValue=tValue_neq DF=DF_neq Probt=Probt_neq))
      vari (keep=Variable NumDF DenDF FValue ProbF)
```

;

by variable;

* Make variable to identify class variable;

```
length class_var $ 25;
```

```
class_var=left(uppercase("&class"));
```

* Select the correct t-test depending on Equality of Variances test;

```
format Probt PVALUE6.4;
```

```
if ProbF>=0.05 then do;
```

```
  tValue=tValue_eq; DF=DF_eq; Probt=Probt_eq;
```

```
end;
```

```
else if .<ProbF<0.05 then do;
```

```
  tValue=tValue_neq; DF=DF_neq; Probt=Probt_neq;
```

```
end;
```

* Calculate % in each group;

```
N=N0+N1;
```

```
Pct0=100*N0/N;
```

```
Pct1=100*N1/N;
```

run;

```
%MEND TDATA;
```

```
/*
```

```
* SAMPLE STEPS FOR THE TDATA MACRO;
```

```
% TDATA(ds=mb,class=rf1,var=l_allow,nonlog=allow,code=a1);
```

```
% TDATA(ds=mb,class=rf2,var=l_allow,nonlog=allow,code=a2);
```

```
% TDATA(ds=mb,class=rf3,var=l_allow,nonlog=allow,code=a3);
```

```
% TDATA(ds=mb,class=rf4,var=l_allow,nonlog=allow,code=a4);
```

```
% TDATA(ds=mb,class=rf5,var=l_allow,nonlog=allow,code=a5);
```

```
* .....
```

```
* Steps to combine t-test results and label variables .....
```

```
* (Currently, these steps must be performed outside of TDATA macro);
```

```
* (A) Get labels for variables;
```

```
proc contents data=diab0 out=get_labels0 (keep=name label) noprint;
```

```
run;
```

```
data get_labels1;
```

```
  set get_labels0 (keep=name label rename=(label=l));
```

```
length class_var $ 25 label $ 80;
```

```
class_var=left(uppercase(name));
```

```
label=l;
drop name l;
```

```
proc sort; by class_var;
run;
```

```
* (B) Concatenate the datasets and put variables in order you want to appear in Excel;
```

```
data combo0;
```

```
set
CF1
```

```
CAARF0 CAARF1 CAARF2 CAARF3 CAARF4 CAARF5 CAARF6 CAARF7 CAARF8 CAARF9 CAARF10
CAARF11 CAARF12 CAARF13 CAARF14
```

```
CAAM1 CAAM2 CAAM3 CAAM4 CAAM5 CAAM6 CAAM7 CAAM8 CAAM9 CAAM10
CAAM11 CAAM12 CAAM13 CAAM14 CAAM15 CAAM16 CAAM17 CAAM18 CAAM19
```

```
CAAP1 CAAP2 CAAP3 CAAP4 CAAP5 CAAP6 CAAP7 CAAP8 CAAP9 CAAP10
CAAP11 CAAP12 CAAP13 CAAP14 CAAP15 CAAP16 CAAP17 CAAP18 CAAP19 CAAP20
CAAP21
```

```
CAAPH1 CAAPH2 CAAPH3 CAAPH4 CAAPH5 CAAPH6 CAAPH7 CAAPH8 CAAPH9 CAAPH10
CAAPH11
```

```
;
```

```
* Create variable to preserve original order;
```

```
order=_n_;
```

```
* Calculate Mean difference;
```

```
mean_diff=mean1-mean0;
```

```
proc sort; by class_var;
run;
```

```
data combo1;
```

```
retain variable class_var label N0 N1 Pct0 Pct1 Mean0 StdDev0 Mean1 StdDev1 mean_diff tValue DF Probt order;
```

```
merge combo0 (in=a)
```

```
get_labels1 (keep=class_var label)
```

```
;
```

```
by class_var;
```

```
if a;
```

```
keep variable class_var label N0 N1 Pct0 Pct1 Mean0 StdDev0 Mean1 StdDev1 mean_diff tValue DF Probt order;
```

```
proc sort; by order;
```

```
run;
```

```
* Output the combined dataset to Excel;
```

```
proc export data=work.combo1
```

```
OUTFILE="###\&ecr._Typical_T-Tests.xls"
```

```
replace;
```

```
sheet="TYPICAL";
```

```
run;
```

```
* END extra steps for tdata macro.....;
```

```
* .....;
```

```
*/
```

***** Part II: NOTDATA *****;

* MACRO NOTDATA;

* Macro to create a dataset. The TDATA macro above does not work if the class variable only has 1 category (i.e. the class variable =0 for all records). Until I have time to make the TDATA macro work, this macro will be used to create a dataset for each independent variable with only 1 category. The datasets will then be combined with the output datasets from the TDATA macro. The idea is to have 1 combined dataset with all the independent variables in it and the t-test results or an indication that there is no data to do a t-test. This combined file with then get exported to Excel.

* Macro variables

ds = dataset containing the variables to analyze (i.e. class variable and continuous variable)

var = continuous dependent variable (assumed to be on the log scale)

nonlog = continuous dependent variable (assumed to be on the original scale, not logged)

class = class variable (must be numeric 0/1 variable)

code = Unique code to put in dataset name to distinguish this t-test from others

* Note:

The parameters of this macro are going to be the same as the ones used for TDATA so that the code for NOTDATA can be changed easily to TDATA. When we analyze study set 2 some of the variables with only 1 category for study set 1 might have 2 categories for study set 2.

```
;
```

```
%MACRO NOTDATA(ds=,var=,nonlog=,class=,code=);
```

```
ods listing exclude all;
```

```
proc freq data=&ds;
```

```
tables &class;
```

```
ods output OneWayFreqs=temp;
```

```
run;
```

```
ods listing exclude none;
```

```
data c&code;
```

```
set temp;
```

```
* Make Variable to hold the name of the logged dependent variable;
```

```
length variable $ 25;
```

```
variable="&var";
```

```
* Make variable to identify class variable;
```

```
length class_var $ 25;
```

```
class_var=left(uppercase("&class"));
```

```
* Select 1 record (there should be only 1 record);
```

```
if _n_=1;
```

```
length N1 N0 8;
```

```
N1=0;
```

```
N0=CumFrequency;
```

```
Keep variable class_var N1 N0;
```

```
run;
```

```
%MEND NOTDATA;
```

***** Part III: REGMOD *****;

* MACRO REGMOD;

* PURPOSE: Macro to run linear regression model, compute the cell counts for dichotomous independent variables (per request of investigator) and optionally output the results to an Excel Workbook. This macro calls 1 macro Get_count (to get cell counts for independent variables).

* NOTES:

(1) Need to set a global macro variable called &file_loc. This tells the macro where to put the data exported to Excel. For example:

```
%let file_loc=c:\project\analysis\Lists
```

* MACRO VARIABLES:

ds = name of dataset to run model on

mod = code to label the model

y = dependent variable

x = independent variable

dolist = output model results to an Excel file y/n.

NOTE: if dolist=y, set dodx to y because the code to create the model results needs the regression diagnostics output dataset.

dodx = create output dataset to be used for regression diagnostics y/n.

NOTE: if dolist=y, set dodx to y because the code to create the model results needs the regression diagnostics output dataset.

idvar = name of id variable. Used in regression diagnostics output dataset.

extra = name of additional variables to include in the regression diagnostics output dataset (for example if the outcome is transformed and you want the untransformed variable to be included).

name= Name of output Excel file (do not use .xls extension)

sample = Label for regression model output (typically this gets set to the name of the sample being used)

cmmd0 = option for proc reg statement (optional)

cmmd1 = option for model statement (optional)

cmmd2 = statement to add to proc reg (optional)

;

```
%MACRO REGMOD(ds=, mod=, y=, x=, dolist=n, dodx=n,
  name=, sample=, idvar=, extra=,
  cmmd0=,cmmd1=,cmmd2=);
```

* (1) First, create an &x macro variable which is stripped of any brackets.

* This macro variable will be used in code to get stats on independent variables;

```
%let xvars=%sysfunc(compress(&x.,{ }));
```

* (2) Run regression model;

```
*ods listing exclude all;
```

```
title1 "BTE-Prometheus: Preliminary AMI Model";
```

```
title2 "...&sample....";
```

```
PROC REG data=&ds. OUTEST=ESTS_&MOD. &cmmd0;
```

```

&mod.:model &y. = &x. /vif &cmmd1;
&cmmd2;
%if &dolist=y %then %do;
ods output FitStatistics=fit_&mod. (drop=cvalue1 cvalue2)
ParameterEstimates=param_&mod.
/* NObs=nobs_&mod. ANOVA=anova_&mod. */
;
%end;
%if &dolist=y or &dodx=y %then %do;
output out=dx_&mod (keep=&idvar. &y. &xvars. &extra. pred resid studresd cookd lever dffits lcl ucl)
p=pred r=resid student=studresd cookd=cookd h=lever dffits=dffits lcl=lcl ucl=ucl;
%end;

run;
quit;
title;
*ods listing exclude none;

* .....;
* (3) Optionally format and output the regression results to Excel Workbook, 1 sheet for each;
%if &dolist=y %then %do;

* (3a) Create dataset that contains only observations used in the model (needed for get_count
macro, i.e. the macro which gets the cell counts for all of the independent variables);
data dx_nomiss_&mod;
set dx_&mod;
notuse=nmiss(of &xvars &y);
if notuse>0 then delete;

run;

* (3b) Calculate the cell counts for each independent variable in the model.
* Only include observations used in building the model;
* %put xvars is &xvars; * To check contents of xvars;
%get_count(ds=dx_nomiss_&mod,mod=&mod,xvar=&xvars);

* (3c) Put variables in the Parameter Estimates file in the order they should appear
in the exported Excell file below;
data paramN_&mod.;
retain Model Dependent Variable count DF Estimate StdErr tValue Probt VarianceInflation Label;
set paramN_&mod (keep=Model Dependent Variable count DF Estimate StdErr tValue Probt
VarianceInflation Label);

run;

* (3d) Output results;
* ANOVA;
proc export data=work.fit_&mod.
OUTFILE="&file_loc\&name..xls"
replace;
sheet=Fit;

run;
* Model;
proc export data=work.paramN_&mod.
OUTFILE="&file_loc\&name..xls"
replace;
sheet=Model;

```

```

run;
%end;
* .....;
%MEND REGMOD;

***** Part IV: GET_COUNT *****;

* MACRO GET_COUNT;
* PURPOSE: This macro calculates the cell counts (explained below) for all dichotomous
independent variables used in a regression model. It adds the counts to the
ParameterEstimates ODS output dataset from proc reg. The macro SELECTR calls this
macro so GET_COUNT is not intended to be run on its own.
* IMPORTANT NOTES:
(A) A parameter estimates dataset called param_&mod. must be created before this
macro can be run. This macro is called by the SELECTR (macro for running linear
regression models). So, you could just run SELECTR rather than running this macro.
(B) This macro does not calculate the cell count for each level of the variable. It only
determines the count in the second category. That is, for dichotomous variables
coded 0/1, this macro will give the count for variable=1. For dichotomous variables
coded 1/2, this macro will give the count for variable=2. If the variable has >2
categories or is continuous, it will give results for the second category, which
is not useful. So far, we are only using dichotomous variables. FIX if needed.
* MACRO VARIABLES:
ds = dataset that proc reg was run on
mod = code to label the model
xvar = independent variables from proc reg model statement.
;
%MACRO GET_COUNT(ds=,mod=,xvar=);
* (A) Get frequencies of all variables;
ods listing exclude all;
proc freq data=&ds;
tables &xvar;
ods output OneWayFreqs=freqs (drop=F_: percent CumFrequency CumPercent);
run;
ods listing exclude none;

* (B) Sort ods output dataset by the TABLE. TABLE is a variable is created by ods output
statement in proc freq. It is of the form: Table||" ||variable name. It labels the
rows with the correct variable name. Will use TABLE to label the columns once the
freqs dataset is transposed;
proc sort data=freqs; by table;
run;

* (C) Put frequencies in rows;
proc transpose data=freqs out=TransFreqs;
by table;
run;

* (D) Select the appropriate rows and create a merging variable (called VARIABLE);
data TransFreqs2;
set TransFreqs (rename=(col2=count));
* Select rows containing the frequency;

```

```
if _name_="Frequency";
```

```
* VARIABLE will contain the variable name. It will be used for merging.
```

```
* Make the length longer;
```

```
length variable $ 25;
```

```
variable=substr(table,7);
```

```
run;
```

```
* (E) Message proc reg output dataset containing model variables, param estimates, etc;
```

```
data getcount0;
```

```
set param_&mod. (rename=(variable=variable_));
```

```
* Make VARIABLE have a longer length;
```

```
length variable $ 25;
```

```
variable=left(variable_);
```

```
drop variable_;
```

```
* Record original order of variables in the model;
```

```
line=_n_;
```

```
proc sort; by variable;
```

```
run;
```

```
* (F) Put the n in parameter output dataset;
```

```
data ParamN_&mod.;
```

```
merge getcount0 (in=a)
```

```
TransFreqs2 (keep=variable count)
```

```
;
```

```
by variable;
```

```
if a;
```

```
proc sort; by line;
```

```
run;
```

```
%MEND GET_COUNT;
```

```
* end of DIAB_Playbook_Macros for Analysis & Modeling.sas;
```

```

/*****
/* Project: Prometheus                date: 04/15/08                */
/* pgm: DIAB_Playbook_Modeling.sas                */
/*                                */
/* This program is part of the task to develop the Diabetes        */
/* episode case rate (ECR) using multiple linear regression        */
/* modeling.                                */
/*                                */
/* The current program does the following:                        */
/* (a) Creates typical and PAC analysis datasets                */
/* (b) Fits final regression model (typical bucket)            */
/* (c) Runs t-tests (PAC bucket)                                */
/*                                */
/* In order to run this program:                                */
/* (1) The file containing macros must be available (via %include): */
/*     DIAB_Playbook_Macros for Analysis & Modeling.sas        */
/*                                */
/* (2) Path names for file locations must be filled in. Search for the */
/*     string ### to find these.                                */
*****/

```

***** Part I - Setup *****;

options orientation=landscape;

* Location of Data;

* Enter as: "c:\folder_a\folder_b\folder_c";

libname sdata "###";

* Location of specialized macros for Prometheus analyses.;

* Enter as: "c:\folder_a\folder_b\folder_c\DIAB_Playbook_Macros for Analysis & Modeling.sas";

%include "###\DIAB_Playbook_Macros for Analysis & Modeling.sas";

* Location to store output.

* Enter as: c:\folder_a\folder_b\folder_c;

* (i.e. indicate path but no quotes);

%let file_loc=###;

* Code to indicate ECR;

* To be used in output file names;

* Enter code: Diab for diabetes, AMI_Fac for AMI Inpatient Facility;

%let ecr=DIAB;

* Name of ECR;

* To be use in title statements;

* Enter text: Diabetes, AMI Inpatient Facility, AMI Professional, etc.;

```
%let ecr_title=Diabetes;
```

```
* Formats;
proc format;
value sampfmt
  1 = 'First 25% sample'
  2 = 'Second 25% sample'
  3 = '50% sample';
* Age 50-<65 yr is reference;
value agefmt
. = 'Missing'
1 = '1: <35'
2 = '2: 35-<50'
0 = '0: 50-<65, ref'
3 = '3: 65-<80'
4 = '4: 80+';
* Format for FEMALE variable;
value sexfmt
. = '.: Missing'
0 = '0: Male'
1 = '1: Female';
run;
```

```
***** Part II - Create Analysis Datasets *****;
```

```
* TYPICAL BUCKET;
* Create variables, labels, etc. on full dataset (model building + validation + test).
* Split datasets later;
data diab0;
  set sdata.diab_typicalv2 (keep=MEMBER_ID rf1-rf14 m1-m19 p1-p21 ref ph1-ph11
    age agegrp sex
    total_allowed_amt total_allowed_amt_nopharm total_allowed_amt_pharm
    sample
    rename=(MEMBER_ID=id
    total_allowed_amt=allow
    total_allowed_amt_nopharm=allowN total_allowed_amt_pharm=allowP
    sex=female
  ));
```

```
* Remove missing data for independent variables;
if age=. or female=. then delete;
```

```
* Remove episodes with total charges <$10;
* Rationale: copays are >=$10;
```

```
if allow<10 then delete;
```

```
* Recode ALLOWN to missing for episodes with medical charges <$10;
* Rationale: copays are >=$10;
* Do not delete the episode since total charges might be in a believable range;
if allowN<10 then allowN=.;
```

```
* Take natural log of total charges;
array y allow allown allowp;
array ly l_allow l_allown l_allowp;
do over y;
  if y>0 then ly=log(y);
end;
```

```
* Do Box-Cox transformation;
b25_allow=( (allow**0.25)-1 )/0.25;
```

```
* Create AGEGRP Dummies;
if agegrp=0 then do;   agegrp1=0; agegrp2=0; agegrp3=0; agegrp4=0; end;
else if agegrp=1 then do; agegrp1=1; agegrp2=0; agegrp3=0; agegrp4=0; end;
else if agegrp=2 then do; agegrp1=0; agegrp2=1; agegrp3=0; agegrp4=0; end;
else if agegrp=3 then do; agegrp1=0; agegrp2=0; agegrp3=1; agegrp4=0; end;
else if agegrp=4 then do; agegrp1=0; agegrp2=0; agegrp3=0; agegrp4=1; end;
```

```
format agegrp agefmt. sample sampfmt.;
```

```
label l_allow = "Ln (Total claims plus pharm charges)"
      l_allown = "Ln (Sum of episode claim charges without pharmacy)"
      l_allowp = "Ln (Sum of all Pharm claims)"

      b25_allow = "Box-cox txfm 0.25 (Total claims plus pharm charges)"
```

```
age   = "Age at beginning of ECR"
agegrp = "Age at beginning of ECR (grouped)"
agegrp1 = "Age: <35 vs. 50-<65"
agegrp2 = "Age: 35-<50 vs. 50-<65"
agegrp3 = "Age: 65-<80 vs. 50-<65"
agegrp4 = "Age: 80+ vs. 50-<65"
```

```
;
```

```
proc sort; by id;
```

```
run;
```

```
* Create the model building dataset (mb), the test set (test), validation (val);
data mb test val;
set diab0;
```

```
if sample=1 then output test;
else if sample=2 then output val;
else if sample=3 then output mb;
run;

* PAC BUCKET;
* Create variables, labels, etc.;
data pac0;
  set sdata.diab_pacv2 (keep=MEMBER_ID rf1-rf14 m1-m19 p1-p21 ref ph1-ph11 c1-c21
    age agegrp sex
    total_allowed_amt total_allowed_amt_nopharm total_allowed_amt_pharm
    rename=(MEMBER_ID=id
    total_allowed_amt=allow
    total_allowed_amt_nopharm=allowN total_allowed_amt_pharm=allowP
    sex=female
    ));

* Remove missing data for independent variables;
if age=. or female=. then delete;

* Remove episodes with total charges <$10;
* Rationale: copays are >=$10;
if allow<10 then delete;

* Remove episodes with total charges >$1 million;
* Rationale: copays are >=$10;
if allow>1000000 then delete;

* Recode ALLOWN to missing for episodes with medical charges <$10;
* Rationale: copays are >=$10;
* Do not delete the episode since total charges might be in a believable range;
if allowN<10 then allowN=.;

* Take natural log of total charges;
array y allow allown allowp;
array ly l_allow l_allown l_allowp;
do over y;
  if y>0 then ly=log(y);
end;

* Create AGEGRP Dummies;
if agegrp=0 then do;   agegrp1=0; agegrp2=0; agegrp3=0; agegrp4=0; end;
else if agegrp=1 then do; agegrp1=1; agegrp2=0; agegrp3=0; agegrp4=0; end;
else if agegrp=2 then do; agegrp1=0; agegrp2=1; agegrp3=0; agegrp4=0; end;
else if agegrp=3 then do; agegrp1=0; agegrp2=0; agegrp3=1; agegrp4=0; end;
else if agegrp=4 then do; agegrp1=0; agegrp2=0; agegrp3=0; agegrp4=1; end;
```

```
format agegrp agefmt.;
```

```
label l_allow = "Ln (Total claims plus pharm charges)"
      l_allown = "Ln (Sum of episode claim charges without pharmacy)"
      l_allowp = "Ln (Sum of all Pharm claims)"

      age = "Age at beginning of ECR"
      agegrp = "Age at beginning of ECR (grouped)"
      agegrp1 = "Age: <35 vs. 50-<65"
      agegrp2 = "Age: 35-<50 vs. 50-<65"
      agegrp3 = "Age: 65-<80 vs. 50-<65"
      agegrp4 = "Age: 80+ vs. 50-<65"
;

proc sort; by id;
run;
```

```
***** Part III - t tests - TYPICAL - TOTAL CHARGES *****;
```

```
* Identify variables with only 1 category (i.e. var=0 for all Y);
* Use NOTDATA macro to get descriptives for these variables;
proc means data=mb min max;
  var female ref rf1-rf14 m1-m19 p1-p21 ref ph1-ph11;
run;
```

```
%TDATA(ds=mb, class=female, var=l_allow, nonlog=allow, code=f1);
```

```
%TDATA(ds=mb, class=ref, var=l_allow, nonlog=allow, code=aarf0);
%TDATA(ds=mb, class=rf1, var=l_allow, nonlog=allow, code=aarf1);
%TDATA(ds=mb, class=rf2, var=l_allow, nonlog=allow, code=aarf2);
%TDATA(ds=mb, class=rf3, var=l_allow, nonlog=allow, code=aarf3);
%TDATA(ds=mb, class=rf4, var=l_allow, nonlog=allow, code=aarf4);
%TDATA(ds=mb, class=rf5, var=l_allow, nonlog=allow, code=aarf5);
%TDATA(ds=mb, class=rf6, var=l_allow, nonlog=allow, code=aarf6);
%TDATA(ds=mb, class=rf7, var=l_allow, nonlog=allow, code=aarf7);
%TDATA(ds=mb, class=rf8, var=l_allow, nonlog=allow, code=aarf8);
%TDATA(ds=mb, class=rf9, var=l_allow, nonlog=allow, code=aarf9);
%TDATA(ds=mb, class=rf10, var=l_allow, nonlog=allow, code=aarf10);
%TDATA(ds=mb, class=rf11, var=l_allow, nonlog=allow, code=aarf11);
%TDATA(ds=mb, class=rf12, var=l_allow, nonlog=allow, code=aarf12);
%TDATA(ds=mb, class=rf13, var=l_allow, nonlog=allow, code=aarf13);
```

```
%TDATA(ds=mb, class=rf14, var=l_allow, nonlog=allow, code=aarf14);
```

```
%TDATA(ds=mb, class=m1, var=l_allow, nonlog=allow, code=aam1);  
%TDATA(ds=mb, class=m2, var=l_allow, nonlog=allow, code=aam2);  
%TDATA(ds=mb, class=m3, var=l_allow, nonlog=allow, code=aam3);  
%TDATA(ds=mb, class=m4, var=l_allow, nonlog=allow, code=aam4);  
%TDATA(ds=mb, class=m5, var=l_allow, nonlog=allow, code=aam5);  
%TDATA(ds=mb, class=m6, var=l_allow, nonlog=allow, code=aam6);  
%TDATA(ds=mb, class=m7, var=l_allow, nonlog=allow, code=aam7);  
%TDATA(ds=mb, class=m8, var=l_allow, nonlog=allow, code=aam8);  
%TDATA(ds=mb, class=m9, var=l_allow, nonlog=allow, code=aam9);  
%TDATA(ds=mb, class=m10, var=l_allow, nonlog=allow, code=aam10);  
%TDATA(ds=mb, class=m11, var=l_allow, nonlog=allow, code=aam11);  
%TDATA(ds=mb, class=m12, var=l_allow, nonlog=allow, code=aam12);  
%TDATA(ds=mb, class=m13, var=l_allow, nonlog=allow, code=aam13);  
%TDATA(ds=mb, class=m14, var=l_allow, nonlog=allow, code=aam14);  
%TDATA(ds=mb, class=m15, var=l_allow, nonlog=allow, code=aam15);  
%TDATA(ds=mb, class=m16, var=l_allow, nonlog=allow, code=aam16);  
%TDATA(ds=mb, class=m17, var=l_allow, nonlog=allow, code=aam17);  
%TDATA(ds=mb, class=m18, var=l_allow, nonlog=allow, code=aam18);  
%TDATA(ds=mb, class=m19, var=l_allow, nonlog=allow, code=aam19);
```

```
%TDATA(ds=mb, class=p1, var=l_allow, nonlog=allow, code=aap1);  
%TDATA(ds=mb, class=p2, var=l_allow, nonlog=allow, code=aap2);  
%TDATA(ds=mb, class=p3, var=l_allow, nonlog=allow, code=aap3);  
%TDATA(ds=mb, class=p4, var=l_allow, nonlog=allow, code=aap4);  
%TDATA(ds=mb, class=p5, var=l_allow, nonlog=allow, code=aap5);  
%TDATA(ds=mb, class=p6, var=l_allow, nonlog=allow, code=aap6);  
%TDATA(ds=mb, class=p7, var=l_allow, nonlog=allow, code=aap7);  
%TDATA(ds=mb, class=p8, var=l_allow, nonlog=allow, code=aap8);  
%TDATA(ds=mb, class=p9, var=l_allow, nonlog=allow, code=aap9);  
%TDATA(ds=mb, class=p10, var=l_allow, nonlog=allow, code=aap10);  
%TDATA(ds=mb, class=p11, var=l_allow, nonlog=allow, code=aap11);  
%TDATA(ds=mb, class=p12, var=l_allow, nonlog=allow, code=aap12);  
%TDATA(ds=mb, class=p13, var=l_allow, nonlog=allow, code=aap13);  
%TDATA(ds=mb, class=p14, var=l_allow, nonlog=allow, code=aap14);  
%TDATA(ds=mb, class=p15, var=l_allow, nonlog=allow, code=aap15);  
%TDATA(ds=mb, class=p16, var=l_allow, nonlog=allow, code=aap16);  
%TDATA(ds=mb, class=p17, var=l_allow, nonlog=allow, code=aap17);  
%TDATA(ds=mb, class=p18, var=l_allow, nonlog=allow, code=aap18);  
%TDATA(ds=mb, class=p19, var=l_allow, nonlog=allow, code=aap19);  
%TDATA(ds=mb, class=p20, var=l_allow, nonlog=allow, code=aap20);  
%TDATA(ds=mb, class=p21, var=l_allow, nonlog=allow, code=aap21);
```

```
%TDATA(ds=mb, class=ph1, var=l_allow, nonlog=allow, code=aaph1);  
%TDATA(ds=mb, class=ph2, var=l_allow, nonlog=allow, code=aaph2);  
%TDATA(ds=mb, class=ph3, var=l_allow, nonlog=allow, code=aaph3);  
%TDATA(ds=mb, class=ph4, var=l_allow, nonlog=allow, code=aaph4);
```

```
%TDATA(ds=mb, class=ph5, var=l_allow, nonlog=allow, code=aaph5);
%TDATA(ds=mb, class=ph6, var=l_allow, nonlog=allow, code=aaph6);
%TDATA(ds=mb, class=ph7, var=l_allow, nonlog=allow, code=aaph7);
%TDATA(ds=mb, class=ph8, var=l_allow, nonlog=allow, code=aaph8);
%TDATA(ds=mb, class=ph9, var=l_allow, nonlog=allow, code=aaph9);
%TDATA(ds=mb, class=ph10, var=l_allow, nonlog=allow, code=aaph10);
%TDATA(ds=mb, class=ph11, var=l_allow, nonlog=allow, code=aaph11);
```

```
* .....;
* Steps to combine t-test results and label variables .....;
* (Currently, these steps must be performed outside of TDATA macro);
* (A) Get labels for variables;
```

```
proc contents data=diab0 out=get_labels0 (keep=name label) noprint;
run;
data get_labels1;
  set get_labels0 (keep=name label rename=(label=l));
```

```
length class_var $ 25 label $ 80;
class_var=left(uppercase(name));
label=l;
drop name l;
```

```
proc sort; by class_var;
run;
```

```
* (B) Concatenate the datasets and put variables in order you want to appear in Excel;
```

```
data combo0;
  set
  CF1
```

```
CAARF0 CAARF1 CAARF2 CAARF3 CAARF4 CAARF5 CAARF6 CAARF7 CAARF8 CAARF9 CAARF10
CAARF11 CAARF12 CAARF13 CAARF14
```

```
CAAM1 CAAM2 CAAM3 CAAM4 CAAM5 CAAM6 CAAM7 CAAM8 CAAM9 CAAM10
CAAM11 CAAM12 CAAM13 CAAM14 CAAM15 CAAM16 CAAM17 CAAM18 CAAM19
```

```
CAAP1 CAAP2 CAAP3 CAAP4 CAAP5 CAAP6 CAAP7 CAAP8 CAAP9 CAAP10
CAAP11 CAAP12 CAAP13 CAAP14 CAAP15 CAAP16 CAAP17 CAAP18 CAAP19 CAAP20
CAAP21
```

```
CAAPH1 CAAPH2 CAAPH3 CAAPH4 CAAPH5 CAAPH6 CAAPH7 CAAPH8 CAAPH9 CAAPH10
CAAPH11
```

```
;
```

```
* Create variable to preserve original order;
order=_n_;
```

```
* Calculate Mean difference;
mean_diff=mean1-mean0;
```

```
proc sort; by class_var;
run;
```

```
data combo1;
retain variable class_var label N0 N1 Pct0 Pct1 Mean0 StdDev0 Mean1 StdDev1 mean_diff tValue DF Probt order;
merge combo0 (in=a)
get_labels1 (keep=class_var label)
;
by class_var;
if a;
keep variable class_var label N0 N1 Pct0 Pct1 Mean0 StdDev0 Mean1 StdDev1 mean_diff tValue DF Probt order;
proc sort; by order;
run;
```

```
* Output the combined dataset to Excel;
proc export data=work.combo1
OUTFILE="###\Diab_TYPICAL_T-Tests.xls"
replace;
sheet="TYPICAL";
run;
* END extra steps for tdata macro.....;
* .....
```

```
***** Part IV - FINAL MODEL - TYPICAL *****;
```

```
* Typical Cases - total charges;
```

```
%REGMOD(mod=TYP_D_FINAL, ds=mb, y=b25_allow, sample=TYPICAL, idvar=id, cmmd1=bic,
name=&ecr._TYPICAL_Model_D_FINAL, dolist=y, dodx=y,
x= RF2 RF3
PH10 PH2 PH1 PH6 PH9 PH4
P16 PH5 P12 P17 P15 P10
RF5 PH3 P6
agegrp1-agegrp4 female
P5 M3 P14 P2 P13
);
```

```
***** Part V - t tests - PAC - TOTAL CHARGES *****;
```

```
* Need to identify variables with only 1 category (i.e. var=0 for all Y);
```

```
* TDATA macro cannot handle these variables. Need to use notdata to get
* descriptives;
proc means data=pac0 min max;
  var female ref rf1-rf14 m1-m19 p1-p21 ph1-ph11 c1-c21;
run;

* Demographic;
%TDATA(ds=pac0, class=female, var=l_allow, nonlog=allow, code=ppf1);

* Risk factors;
%TDATA(ds=pac0, class=ref, var=l_allow, nonlog=allow, code=pprf0);
%TDATA(ds=pac0, class=rf1, var=l_allow, nonlog=allow, code=pprf1);
%TDATA(ds=pac0, class=rf2, var=l_allow, nonlog=allow, code=pprf2);
%TDATA(ds=pac0, class=rf3, var=l_allow, nonlog=allow, code=pprf3);
%TDATA(ds=pac0, class=rf4, var=l_allow, nonlog=allow, code=pprf4);
%TDATA(ds=pac0, class=rf5, var=l_allow, nonlog=allow, code=pprf5);
%TDATA(ds=pac0, class=rf6, var=l_allow, nonlog=allow, code=pprf6);
%TDATA(ds=pac0, class=rf7, var=l_allow, nonlog=allow, code=pprf7);
%TDATA(ds=pac0, class=rf8, var=l_allow, nonlog=allow, code=pprf8);
%TDATA(ds=pac0, class=rf9, var=l_allow, nonlog=allow, code=pprf9);
%TDATA(ds=pac0, class=rf10, var=l_allow, nonlog=allow, code=pprf10);
%TDATA(ds=pac0, class=rf11, var=l_allow, nonlog=allow, code=pprf11);
%TDATA(ds=pac0, class=rf12, var=l_allow, nonlog=allow, code=pprf12);
%TDATA(ds=pac0, class=rf13, var=l_allow, nonlog=allow, code=pprf13);
%TDATA(ds=pac0, class=rf14, var=l_allow, nonlog=allow, code=pprf14);

* Medical;
%TDATA(ds=pac0, class=m1, var=l_allow, nonlog=allow, code=ppm1);
%TDATA(ds=pac0, class=m2, var=l_allow, nonlog=allow, code=ppm2);
%TDATA(ds=pac0, class=m3, var=l_allow, nonlog=allow, code=ppm3);
%TDATA(ds=pac0, class=m4, var=l_allow, nonlog=allow, code=ppm4);
%TDATA(ds=pac0, class=m5, var=l_allow, nonlog=allow, code=ppm5);
%TDATA(ds=pac0, class=m6, var=l_allow, nonlog=allow, code=ppm6);
%TDATA(ds=pac0, class=m7, var=l_allow, nonlog=allow, code=ppm7);
%TDATA(ds=pac0, class=m8, var=l_allow, nonlog=allow, code=ppm8);
%TDATA(ds=pac0, class=m9, var=l_allow, nonlog=allow, code=ppm9);
%TDATA(ds=pac0, class=m10, var=l_allow, nonlog=allow, code=ppm10);
%TDATA(ds=pac0, class=m11, var=l_allow, nonlog=allow, code=ppm11);
%TDATA(ds=pac0, class=m12, var=l_allow, nonlog=allow, code=ppm12);
%TDATA(ds=pac0, class=m13, var=l_allow, nonlog=allow, code=ppm13);
%TDATA(ds=pac0, class=m14, var=l_allow, nonlog=allow, code=ppm14);
%TDATA(ds=pac0, class=m15, var=l_allow, nonlog=allow, code=ppm15);
%TDATA(ds=pac0, class=m16, var=l_allow, nonlog=allow, code=ppm16);
%TDATA(ds=pac0, class=m17, var=l_allow, nonlog=allow, code=ppm17);
%TDATA(ds=pac0, class=m18, var=l_allow, nonlog=allow, code=ppm18);
%TDATA(ds=pac0, class=m19, var=l_allow, nonlog=allow, code=ppm19);
```

*** Procedures;**

```
%TDATA(ds=pac0, class=p1, var=l_allow, nonlog=allow, code=ppp1);
%TDATA(ds=pac0, class=p2, var=l_allow, nonlog=allow, code=ppp2);
%TDATA(ds=pac0, class=p3, var=l_allow, nonlog=allow, code=ppp3);
%TDATA(ds=pac0, class=p4, var=l_allow, nonlog=allow, code=ppp4);
%TDATA(ds=pac0, class=p5, var=l_allow, nonlog=allow, code=ppp5);
%TDATA(ds=pac0, class=p6, var=l_allow, nonlog=allow, code=ppp6);
%TDATA(ds=pac0, class=p7, var=l_allow, nonlog=allow, code=ppp7);
%TDATA(ds=pac0, class=p8, var=l_allow, nonlog=allow, code=ppp8);
%TDATA(ds=pac0, class=p9, var=l_allow, nonlog=allow, code=ppp9);
%TDATA(ds=pac0, class=p10, var=l_allow, nonlog=allow, code=ppp10);
%TDATA(ds=pac0, class=p11, var=l_allow, nonlog=allow, code=ppp11);
%TDATA(ds=pac0, class=p12, var=l_allow, nonlog=allow, code=ppp12);
%TDATA(ds=pac0, class=p13, var=l_allow, nonlog=allow, code=ppp13);
%TDATA(ds=pac0, class=p14, var=l_allow, nonlog=allow, code=ppp14);
%TDATA(ds=pac0, class=p15, var=l_allow, nonlog=allow, code=ppp15);
%TDATA(ds=pac0, class=p16, var=l_allow, nonlog=allow, code=ppp16);
%TDATA(ds=pac0, class=p17, var=l_allow, nonlog=allow, code=ppp17);
%TDATA(ds=pac0, class=p18, var=l_allow, nonlog=allow, code=ppp18);
%TDATA(ds=pac0, class=p19, var=l_allow, nonlog=allow, code=ppp19);
%TDATA(ds=pac0, class=p20, var=l_allow, nonlog=allow, code=ppp20);
%TDATA(ds=pac0, class=p21, var=l_allow, nonlog=allow, code=ppp21);
```

*** Pharmacy;**

```
%TDATA(ds=pac0, class=ph1, var=l_allow, nonlog=allow, code=ppph1);
%TDATA(ds=pac0, class=ph2, var=l_allow, nonlog=allow, code=ppph2);
%TDATA(ds=pac0, class=ph3, var=l_allow, nonlog=allow, code=ppph3);
%TDATA(ds=pac0, class=ph4, var=l_allow, nonlog=allow, code=ppph4);
%TDATA(ds=pac0, class=ph5, var=l_allow, nonlog=allow, code=ppph5);
%TDATA(ds=pac0, class=ph6, var=l_allow, nonlog=allow, code=ppph6);
%TDATA(ds=pac0, class=ph7, var=l_allow, nonlog=allow, code=ppph7);
%NOTDATA(ds=pac0, class=ph8, var=l_allow, nonlog=allow, code=ppph8);
%TDATA(ds=pac0, class=ph9, var=l_allow, nonlog=allow, code=ppph9);
%TDATA(ds=pac0, class=ph10, var=l_allow, nonlog=allow, code=ppph10);
%TDATA(ds=pac0, class=ph11, var=l_allow, nonlog=allow, code=ppph11);
```

*** Complications;**

```
%TDATA(ds=pac0, class=c1, var=l_allow, nonlog=allow, code=ppc1);
%TDATA(ds=pac0, class=c2, var=l_allow, nonlog=allow, code=ppc2);
%TDATA(ds=pac0, class=c3, var=l_allow, nonlog=allow, code=ppc3);
%TDATA(ds=pac0, class=c4, var=l_allow, nonlog=allow, code=ppc4);
%TDATA(ds=pac0, class=c5, var=l_allow, nonlog=allow, code=ppc5);
%TDATA(ds=pac0, class=c6, var=l_allow, nonlog=allow, code=ppc6);
%TDATA(ds=pac0, class=c7, var=l_allow, nonlog=allow, code=ppc7);
%TDATA(ds=pac0, class=c8, var=l_allow, nonlog=allow, code=ppc8);
%TDATA(ds=pac0, class=c9, var=l_allow, nonlog=allow, code=ppc9);
%TDATA(ds=pac0, class=c10, var=l_allow, nonlog=allow, code=ppc10);
%TDATA(ds=pac0, class=c11, var=l_allow, nonlog=allow, code=ppc11);
```

```
%TDATA(ds=pac0, class=c12, var=l_allow, nonlog=allow, code=ppc12);
%TDATA(ds=pac0, class=c13, var=l_allow, nonlog=allow, code=ppc13);
%TDATA(ds=pac0, class=c14, var=l_allow, nonlog=allow, code=ppc14);
%TDATA(ds=pac0, class=c15, var=l_allow, nonlog=allow, code=ppc15);
%TDATA(ds=pac0, class=c16, var=l_allow, nonlog=allow, code=ppc16);
%TDATA(ds=pac0, class=c17, var=l_allow, nonlog=allow, code=ppc17);
%TDATA(ds=pac0, class=c18, var=l_allow, nonlog=allow, code=ppc18);
%TDATA(ds=pac0, class=c19, var=l_allow, nonlog=allow, code=ppc19);
%TDATA(ds=pac0, class=c20, var=l_allow, nonlog=allow, code=ppc20);
%TDATA(ds=pac0, class=c21, var=l_allow, nonlog=allow, code=ppc21);
```

```
* .....;
* Steps to combine t-test results and label variables .....;
* (Currently, these steps must be performed outside of TDATA macro);
* (A) Get labels for variables;
```

```
proc contents data=pac0 out=get_labels0 (keep=name label) noprint;
run;
data get_labels1;
  set get_labels0 (keep=name label rename=(label=1));
```

```
  length class_var $ 25 label $ 80;
  class_var=left(uppercase(name));
  label=1;
  drop name l;
```

```
  proc sort; by class_var;
run;
```

```
* (B) Concatenate the datasets and put variables in order you want to appear in Excel;
```

```
data combo0;
  set
  CppF1
```

```
  CppRF0 CppRF1 CppRF2 CppRF3 CppRF4 CppRF5 CppRF6 CppRF7 CppRF8 CppRF9 CppRF10
  CppRF11 CppRF12 CppRF13 CppRF14
```

```
  CppM1 CppM2 CppM3 CppM4 CppM5 CppM6 CppM7 CppM8 CppM9 CppM10
  CppM11 CppM12 CppM13 CppM14 CppM15 CppM16 CppM17 CppM18 CppM19
```

```
  CppP1 CppP2 CppP3 CppP4 CppP5 CppP6 CppP7 CppP8 CppP9 CppP10
  CppP11 CppP12 CppP13 CppP14 CppP15 CppP16 CppP17 CppP18 CppP19 CppP20
  CppP21
```

```
  CppPH1 CppPH2 CppPH3 CppPH4 CppPH5 CppPH6 CppPH7 CppPH8 CppPH9 CppPH10
  CppPH11
```

```
  CppC1 CppC2 CppC3 CppC4 CppC5 CppC6 CppC7 CppC8 CppC9 CppC10
```

CppC11 CppC12 CppC13 CppC14 CppC15 CppC16 CppC17 CppC18 CppC19 CppC20 CppC21

;

* Create variable to preserve original order;

order=_n_;

* Calculate Mean difference;

mean_diff=mean1-mean0;

proc sort; by class_var;

run;

data combo1;

retain variable class_var label N0 N1 Pct0 Pct1 Mean0 StdDev0 Mean1 StdDev1 mean_diff tValue DF Probt order;

merge combo0 (in=a)

get_labels1 (keep=class_var label)

;

by class_var;

if a;

keep variable class_var label N0 N1 Pct0 Pct1 Mean0 StdDev0 Mean1 StdDev1 mean_diff tValue DF Probt order;

proc sort; by order;

run;

* Output the combined dataset to Excel;

proc export data=work.combo1

OUTFILE="&file_loc.\&ecr._PAC_T-Tests.xls"

replace;

sheet="PAC";

run;

* END extra steps for tdata macro.....;

*

* end of DIAB_Playbook_Modeling.sas;