```
/*************************************************************************/
/* Program : stddiff.sas
/* Purpose : SAS macro to calculate the Standardized Difference
/* Usage : %stddiff(inds = Studydata, groupvar = dex,
/* numvars = age bmi/r glucose,
/* charvars = female surgtype,
/* stdfmt = 8.5,
/* outds = std_result);
/*************************************************************************/
/* NOTE: All binary variables must be coded as 0 and 1 in the dataset
/* PARAMETERS:
/* inds:      input dataset
/* groupvar:   a binary variable, must be coded as 0 and 1
/* numvars:    a list of continuous variables.
/*          "/r" denotes to use the rank-based mean and SD to calculate Stddiff
/* charvars:   a list of categorical variables. If a variable is a binary categorical variable,
/*          it must be coded as 0 and 1 since we use the level = 0 as the reference level.
/* stdfmt = 8.5 the format of Standardized Difference
/* outds output result dataset
/*************************************************************************/

options  symbolgen mlogic mprint;
%macro       stddiff( inds = ,
                                groupvar =,
                                numvars = ,
                                charvars = ,
                                wtvar = ,
```

```
                                stdfmt = 8.4,
                                outds = stddiff_result );

/* create a table to store stddiff */
proc sql;
  create table &outds.
    (VarName char(32),
                Stddiff char (10)
    );
quit;

/* delete records if the group variable is missing */

data base_data;
        set &inds.;
        where &GroupVar. ne .;
run;

/* remove leading or tailing blanks */
%let groupvar = %sysfunc(strip(&GroupVar.));

                                /****************************************/
                                /* part 1: compare continuous variables */
                                /****************************************/

%if %length(&numvars.) > 0 %then %do;
```

```
/* remove multiple blanks and get the total number of continuous variables */
        %let numvar = %sysfunc(compbl(&numvars.));
        %let numvar = %sysfunc(strip(&numvar.));
        %let n_convar = %sysfunc(countc(&numvar.,' '));
        %let n_convar = %eval(&n_convar. + 1);

/* summarize variables one-by-one */
        %do ii = 1 %to &n_convar.;
        %let convar = %sysfunc(scan(&numvar.,&ii.,' '));

  /* if requires rank-based mean and std for skewed variables */
                %if %index(&convar., /r) > 0 %then %do;
                %let convar = %sysfunc(scan(&convar.,1,'/'));
                %let convar = %sysfunc(strip(&convar.));

                data temp_1;
                        set base_data (keep = &groupvar. &convar. &wtvar.);
                run;

  /* rank a variable */
                proc rank data=temp_1 out=temp_2;
                var &convar.;
                ranks rank_&convar.;
                run;

  /* get ranked-mean and sd */
```

```
		proc means data = temp_2;
			class &groupvar.;
			var rank_&convar.;
			weight &wtvar.;
			output out = temp_3 mean = _mean_  std = _std_;
		run;

		data  temp_3;
			set temp_3;
			where _type_ = 1;
		run;

		proc sort data = temp_3;
			by &groupvar.;
		run;
	%end;

/* for normal-distributed variable */

	%else %do;
	%let convar = %sysfunc(strip(&convar.));
	data temp_1;
		set base_data (keep = &groupvar. &convar. &wtvar.);
	run;
	data temp_2;
		set temp_1;
	run;
```

```
        /* get mean and sd */

                proc means data = temp_2;
                        class &groupvar.;
                        var &convar.;
                        weight &wtvar.;
                        output out = temp_3 mean = _mean_  std = _std_;
                run;

                data  temp_3;
                        set temp_3;
                        where _type_ = 1;
                run;

                proc sort data = temp_3;
                        by &groupvar.;
                run;

        %end;

/* calculate stddiff */
        proc sql;
                create table temp_4 as
                        select (a._mean_ - b._mean_)/
                                sqrt((a._std_**2 + b._std_**2)/2) as d
                        from temp_3(where = (&groupvar = 1)) as a,
```

```
                                temp_3(where = (&groupvar = 0)) as b;
  quit;

  data temp_5;
                    set temp_4;
     stddiff = compress(put(d,&stdfmt.));
     keep stddiff;
   run;

 /* insert into std table */
  proc sql noprint;
        select stddiff into: std_value from temp_5;
        insert into &outds.  values("&convar.", "&std_value.");
  quit;

 /* delete temporary data sets */

  proc datasets lib = work nodetails nolist;
   delete  temp_1 - temp_5;
  quit;
  %end;
%end;

        /*********************************************/
        /* part 2: compare categorical variables     */
        /*********************************************/
```

```
%if %length(&charvars.) > 0 %then %do;
	%let n_charvar = %sysfunc(countw(&charvars.));

/* get column percents for each levels of the variable by the group */
	%do jj = 1 %to &n_charvar.;
		%let char_var = %scan(&charvars., &jj.);
		%let char_var = %sysfunc(strip(&char_var.));
		 data temp_1;
			set base_data (keep = &groupvar. &char_var. &wtvar.);
		 run;

		 proc sql;
			create table temp_2 as
			select distinct &char_var. as &char_var.
			from temp_1
			where &char_var. is not missing;
		 quit;

		 proc sql noprint;
			select count(*) into :_mylevel_ from temp_2;
		 quit;

		%let _mylevel_ = %sysfunc(strip(&_mylevel_.));

		 data temp_3;
			set temp_2;
				do &groupvar. = 0,1 ;
```

```
        output;
                end;
 run;

ods output CrossTabFreqs = temp_4;
proc freq data = temp_1;
        table &char_var. * &groupvar.;
        %if %length(&wtvar.) > 0 %then %do;
                weight &wtvar.;
                %end;
run;

proc sql;
        create table  temp_5 as
        select a.*, b.ColPercent
        from temp_3 as a
        left join temp_4 as b
        on      a.&groupvar. = b.&groupvar. and
        a.&char_var. = b.&char_var.;
quit;

data temp_6;
        set temp_5;
        if ColPercent = . then ColPercent = 0;
run;

proc sort data = temp_6 out = catfreq;
```

```
                by &groupvar. &char_var.;
        run;

        proc datasets lib = work nodetails nolist;
                delete  temp_1 - temp_6;
        quit;

/* if a categorical variable only has one level: 0 or 1 */
/* stddiff = 0 */
        %if &_mylevel_. = 1 %then %do;
                proc sql noprint;
                        insert into &outds.  values("&char_var.", "0");
                quit;
        %end;

/* if a categorical variable  has two level: 0 and 1 */
/* it is a binary variable, using two sample proportation formula */
        %else %if &_mylevel_. = 2 %then %do;

                data temp_7;
                        set catfreq;
                        where &char_var. = 1;
                        ColPercent = ColPercent/100;
                run;

                proc sql;
                        create table temp_8 as
```

```
                    select (a.ColPercent - b.ColPercent)/(sqrt((a.ColPercent*(1-
                                a.ColPercent) +
                            b.ColPercent*(1-b.ColPercent))/2)) as d
                    from temp_7(where = (&groupvar = 1)) as a,
                 temp_7(where = (&groupvar = 0)) as b;
                quit;

                data temp_9;
          set temp_8;
          stddiff = compress(put(d,&stdfmt.));
    keep stddiff;
  run;

                proc sql noprint;
                    select  stddiff into: std_value from temp_9;
                        insert into &outds.  values("&char_var.", "&std_value.");
                quit;

                proc datasets lib = work nodetails nolist;
                    delete  temp_7 temp_8 temp_9;
                quit;
   %end;
/* if a categorical variable  has more than two level such as a, b and c */
          %else %if &_mylevel_. > 2 %then %do;
                %let _k_ = %eval(&_mylevel_. - 1);
                %let _k_ = %sysfunc(strip(&_k_.));
                data temp_7;
```

```
                set catfreq;
                by &groupvar.;
                if last.&groupvar. then delete;
                ColPercent = ColPercent/100;
        run;

        proc sql noprint;
                select ColPercent into :tlist separated by ' '
                from temp_7 where &groupvar. = 1;

                select ColPercent into :clist separated by ' '
                from temp_7 where &groupvar. = 0;
        quit;

/* vector T, C and T-C */
        data t_1;
                array t{*}  t1- t&_k_.   (&tlist.);
                array c{*}  c1- c&_k_.   (&clist.);
                array tc{*} tc1 - tc&_k_. ;
                do i = 1 to dim(t);
                tc{i} = t{i} - c{i};
                end;
        drop i;
        run;

/* each column has one element of a S covariance matrix (k x k) */
```

```
%let _dm = ;
%let _dm = %eval(&_k_.*&_k_.);
data covdata;
	array t{*}  t1- t&_k_.  (&tlist.);
	array c{*}  c1- c&_k_.   (&clist.);
	array cv{&_k_.,&_k_.} x1 -x&_dm.;
	do i = 1 to &_k_.;
	do j = 1 to &_k_.;
		if i = j then do;
			cv{i,j} = 0.5*(t{i}*(1-t{i}) + c{i}*(1-c{i}));
			end;
		else do;
			cv{i,j} = -0.5 * (t[i] * t[j] + c[i] * c[j]);
			end;
		if cv{&_k_.,&_k_.] ne . then output;
	end;
	end;
run;

proc transpose data = covdata(keep = x1 -x&_dm.) out = covdata_1;
run;

data covdata_2;
	set covdata_1;
	retain id gp 1;
	if mod(_n_ - 1,&_k_.) = 0 then gp = gp + 1;
run;
```

```
proc sort data = covdata_2 ;
        by gp id;
run;

data covdata_3;
        set covdata_2;
        by gp id;
        retain lp;
        if first.gp then lp = 0;
        lp = lp+1;
run;

/* transpose to a S variance-covariance matrix format */

data covdata_4;
        set covdata_3;
        retain y1-y&_k_.;
        array cy{1:&_k_.} y1-y&_k_.;
        by gp id;
        if first.gp then do;
        do k = 1 to &_k_.;
                cy{k} = .;
          end;
        end;
        cy{lp} = col1;
        if last.gp then output;
```

```
                    keep y:;
                run;

/* get inverse of S matrix */
                data A_1;
                 set covdata_4;
                 array _I{*} I1-I&_k_.;
                 do j=1 to &_k_.;
                  if j=_n_ then _I[j]=1;
                  else _I[j]=0;
                 end;
                 drop j;
                run;

/* solve the inverse of the matrix */

 %macro inv;
        %do j=1 %to &_k_.;
                proc orthoreg data=A_1 outest=A_inv_&j.(keep=y1-y&_k_.)
                        noprint singular=1E-16;
                        model I&j=y1-y&_k_. /noint;
                run;
                quit;
        %end;

                data A_inverse;
                set %do j=1 %to &_k_.;
```

```
            A_inv_&j
        %end;;
            run;
%mend;
%inv;

            proc transpose data=A_inverse out=A_inverse_t;
            run;

 /* calculate the mahalanobis distance */
            data t_2;
                set A_inverse_t;
                array t{*}  t1- t&_k_.  (&tlist.);
                array c{*}  c1- c&_k_.  (&clist.);
                i = _n_;
                trt = t{i};
                ctl = c{i};
                tc = t{i} - c{i};
            run;

            data t_3;
                set t_2;
                array aa{&_k_.} col1 - col&_k_.;
                array bb{&_k_.} bb1- bb&_k_.;
                do i = 1 to &_k_.;
                bb{i} = aa{i}*tc;
                end;
```

```
run;

proc summary data = t_3 ;
	var bb1-bb&_k_.;
	output out = t_4 sum =;
run;

data t_5;
	merge t_1 t_4;
	array d1{*} tc1- tc&_k_. ;
	array d2{*} bb1-bb&_k_.;
	array d3{*} y1-y&_k_.;
	do i = 1 to &_k_.;
		d3{i} = d1{i}*d2{i};
	end;
	d = sqrt(sum(of y1-y&_k_.));
	stddiff = compress(put(d,&stdfmt.));
	keep stddiff;
run;

proc sql noprint;
	select  stddiff into: std_value from t_5;
	insert into &outds.  values("&char_var.", "&std_value.");
quit;

proc datasets lib = work nodetails nolist;
	delete  covdata covdata_1 covdata_2 covdata_3 covdata_4
```

```
                A_1 A_inverse A_inverse_t t_1 t_2 t_3 t_4 t_5
                A_inv_:;
                quit;
            %end;
        %end;
%end;

proc datasets lib = work nodetails nolist;
 delete Catfreq  Base_data temp_7;
quit;

proc print data = &outds.;
        title 'Calculated Standardized Difference';
run;

title;

%mend;
```