USING THE PHARMACOEPI TOOLBOX IN SAS

Macro Version 2.4.16
May 14, 2014

Jeremy A. Rassen, Sc.D.
Michael Doherty, M.S.
Wei Huang, M.S.
Sebastian Schneeweiss, M.D., Sc.D.

ABOUT THE TOOLBOX

The Pharamcoepi Toolbox contains a number of elements useful for modern pharmacoepidemiology. Included in this version of the toolbox are:

- High-Dimensional Variable Selection
- High-Dimensional Propensity Scores
- High-Dimensional Disease Risk Scores (*experimental*)
- 1:1, *n*:1, and *r*:1 Greedy Matching
- 1:1, *n*:1, and *r*:1 Nearest-Neighbor Matching
- 1:1:1 Nearest-Neighbor Matching
- Epidemiology Table Creation

All elements in the toolbox are implemented as Java programs, which can be linked into SAS, R, or custom code. For convenience, we have created SAS macros that encapsulate the Java programs. This guide details how to use the elements of the Toolbox from within SAS.

Included in the distribution is example code for using the Toolbox in R. The Java classes are self-documented with Javadocs.

HOW TO CITE

When using the Toolbox, please cite it as follows:

Rassen JA, Doherty M, Huang W, Schneeweiss S. Pharmacoepidemiology Toolbox. Boston, MA. http://www.hdpharmacoepi.org.

SYSTEM REQUIREMENTS

The Toolbox requires SAS Version 9.2 or later. It will not run on earlier versions of SAS.

The macro makes use of SAS's Java subsystem. Giving sufficient memory to the Java subsystem will substantially increase performance; adding a line such as

```
-jreoptions (-Xmx1024m)
```

to the SASV9_LOCAL.CFG file or similar should suffice. For more information on how to allocate Java memory in SAS, please see http://support.sas.com/documentation/cdl/en/grstatgraph/63878/HTML/default/viewer.htm#n099d3bkbc1lg5n1eniitj9b8y1f.htm.

Note to SAS Batch Mode users. The Toolbox macros use PROC EXPORT for transferring data; PROC EXPORT will hang batch SAS unless the `-noterminal` option is specified:

```
sas -noterminal program.sas
```

HIGH DIMENSIONAL VARIABLE SELECTION
HIGH DIMENSIONAL PROPENSITY SCORES
HIGH DIMENSIONAL DISEASE RISK SCORES

MACROS DEFINED
`%InitHDMacros`
`%DoHDVariableSelection`
`%EstimateHDPS`
`%EstimateHDDRS` *[experimental]*
`%RunOutcomeModels`

DESCRIPTION
This macro implements a multi-step algorithm to implement high-dimensional proxy adjustment in claims data. The algorithm is described in detail in *High-dimensional proxy adjustment in claims data studies of treatment effects* by Schneeweiss et al. The additions to the second version of the algorithm are described in *Observed performance of high-dimensional propensity score analyses of treatment effects in small samples* by Rassen et al. (in press).

Briefly, the macro's steps include 1) identify data dimensions, e.g. diagnoses, procedures, and medications, 2) empirically identify candidate covariates, 3) assess recurrence, i.e. multiplicity of same code, 4) prioritize covariates, 5) select covariates for adjustment, 6) estimate exposure propensity score, and 7) estimate outcome model.

FILES REQUIRED
Main SAS file: `hdmacros.sas`
Core program file: `pharmacoepi.jar`

MACROS INCLUDED
The hd-PS macro is divided into several key steps: variable creation and selection, score model building, and outcome models. Each step has a distinct associated macro, plus a single initialization macro that sets parameters for all subsequent steps. In more detail, the macros are:

`%InitHDMacros(…)`
Initializes the macro and sets parameters used in all subsequent steps. Takes the parameters described below.

`%DoHDVariableSelection`
Performs variable creation and selection. Takes no parameters.

`%EstimateHDPS(name_of_dataset)`
Estimates propensity scores and places the scores as variables called `ps2, ps3, ps4, ps5` in the specified dataset. There is no `ps1` since the first model run is a crude (unadjusted) model. The variables contained in `ps2` to `ps5` are described under "`score_type_X`", below.

`%RunOutcomeModels(name_of_dataset, score_var` `[ps or drs]``)`
Runs the specified matched or decile-based outcome models. With the specified dataset, searches for

any score variables with the specified score variable base name, and runs outcome models for each score variable. For hd-PS the `score_var` should be `ps`, while for hd-DRS (experimental) the `score_var` should be `drs`.

MACRO PARAMETERS THAT CONTROL PRE-DEFINED VARIABLES

`var_patient_id` ***required***
Name of unique patient identifier variable.

`var_exposure` ***optional; default=***`exposure`
Name of binary exposure variable. A value of 1 for this variable will be considered exposed; 0 will be considered unexposed/referent.

`var_outcome` ***optional; default=***`outcome`
Name of binary outcome variable. A value of 1 for this variable will be considered outcome positive; 0 will be considered outcome negative.

`vars_demographic`
optional; default=none
A list of variables to be considered as demographic information, such as age, sex, and race.

`vars_predefined`
optional
A list of variables to be considered as pre-defined, such as co-morbidities or past drug usage. **If this option is not specified, all variables in the dataset other than the patient ID, exposure, outcome, and demographic variables will be considered pre-defined.**

`vars_force_categorical`
optional; default=none
An optional list of numeric variables that should be treated as categorical. Year of service is a common example. (See note below.)

`vars_ignore` ***optional; default=none***
A list of variables to be ignored in the propensity score estimation and simply passed through to output data sets. Examples include follow-up time and other variables important to an analysis but which do not figure into confounder adjustment. **This option will only take effect when** `vars_predefined` **is not specified.**

MACRO PARAMETERS THAT CONTROL INPUT AND OUTPUT DATA

`input_cohort` ***required***
Name of the input SAS dataset containing the patients, exposure, outcome, and any predefined covariates. Datasets will not be modified. See note below on format of the data.

`input_dimX` ***one or more dimensions required***
This parameter is specified one or more times to indicate the names of the datasets containing the dimension data. Datasets will not be modified. See note below for dataset format.

Each time this parameter is specified, two data items are needed: the name of the dimension dataset and the name of the field containing the code of interest. These two items should be supplied on one line with only spaces in between. (See example.)

The parameters should be numbered sequentially and with no gaps in the numbering (`input_dim1, input_dim2`, etc.).

`output_cohort`
optional; default=`output_scored_cohort`
Name of a SAS dataset that will be created or replaced. This dataset will contain the input cohort (including their exposure, outcome, and fixed covariates), as well as the series of empirical covariates calculated from each dimension of data. It is intended for diagnostic use or further exploration of the generated variables and may be quite large.

`result_diagnostic`
optional; default=`result_diagnostic`
Base name of a series of SAS datasets that will be created or replaced. Two datasets will be created. (1) A dataset with all codes considered; the default name is `result_diagnostic_dim_codes`. (2) A dataset with details on all variables constructed from those codes and subsequently considered for selection; the default name is `result_diagnostic_all_vars`.

`result_estimates`
optional; default=`result_estimates`
Name of a SAS dataset that will be created or replaced. This dataset will contain the point estimates and confidence intervals for each of the outcome models that were requested (if any).

`path_temp_dir` ***required***
The path to a directory where the hd-PS macro can store temporary files. There should be enough space in the directory to hold a second copy of the input cohort and each of the dimensions.

`path_jar_file` ***required***
The path to the `pharmacoepi.jar` file.
NOTE: This path cannot contain spaces. This is a SAS limitation.

MACRO PARAMETERS THAT CONTROL VARIABLE CREATION & SELECTION

`top_n` ***optional; default=***`200`
The *n* most prevalent empirical covariates to consider from each dimension of data.

`k` ***optional; default=***`500`
The number of empirical covariates to include in the resulting propensity score.

`frequency_min`
optional; default=`0`
The minimum number of occurrences an empirical variable must have before being considered for selection. **For compatibility with older versions of the software, set the value to 100.**

`ranking_method`
optional; default=`BIAS`
A text variable that indicates one of three modes for selecting variables. Specifying `BIAS` will yield a variable list in which the top k variables are selected as ranked by the Bross bias formula.

Specifying `EXP_ASSOC` will yield a variable list in which the top k variables are selected as ranked by the confounder/exposure "risk ratio" (ratio of prevalence of the confounder in the exposed versus the unexposed). This is most suitable for cases where there are fewer than 150 exposed outcomes.

Specifying `OUTCOME_ASSOC` will yield a variable list in which the top k variables are selected as ranked by the confounder/outcome risk ratio. This is most suitable for disease risk scores.

`outcome_zero_cell_corr`
optional; default=`0`
An indicator for whether to screen variables with a zero correction added to each cell in the confounder/outcome 2x2 table. Recommended when the number of exposed outcomes is fewer than 150.

`infer_service_intensity_vars`
optional; default=`0`
An indicator for whether to infer the intensity of health service utilization by computing quartile of (a) number of codes per patient, and (b) number of *unique* codes per patient within each dimension. These quartile indicators are then screened like all other hd-PS variables and allowed to enter the final propensity score. This option may be used in place of investigator-defined variables such as "number of unique medications used" or "number of office visits".

MACRO PARAMETERS THAT CONTROL SCORE TYPES AND OUTCOME MODELS

`outcome_model_deciles`
optional; default=`1`
An indicator for whether to run outcome models using deciles.

`outcome_model_matched`
optional; default=`0`
An indicator for whether to run outcome models using greedy matching.

`score_type_`*`X`*
optional (X=1 to 5); default is that all =1

An indicator for whether to create propensity (or disease risk) score types 1 through 5, and, if analysis is selected, whether to execute an outcome model adjusted by (or matched on) each of those propensity (or disease risk) scores. The score types are as follows:
(1) No confounders (unadjusted)
(2) Demographics variables only
(3) Demographics and predefined variables
(4) Demographics, predefined, and empirical (macro-selected) variables
(5) Demographics and empirical (macro-selected) variables

MACRO PARAMETERS THAT CONTROL NETEZZA DATABASE PROCESSING MODE

`selection_mode`

optional; default=`LOCAL`

A text variable that indicates whether variable selection will be done locally using traditional hd-PS processing, or whether it should be done in-database with Netezza information appliances. The default, `LOCAL`, will run within SAS using the Java subsystem. Specifying `DB` will cause variable selection to run within a Netezza appliance. The mode requires several additional parameters to be specified.

`input_cohort_table`

required for* `DB` *mode

A table on the database server that mirrors the cohort specified in `input_cohort`. This table is used to speed processing.

`db_driver_classpath`

required for* `DB` *mode

The full filesystem path to the database driver class. The driver is often packaged in `nzjdbc3.jar`.

`db_url`

required for* `DB` *mode

The JDBC URL to the Netezza database. This should take the form of `jdbc:netezza://server.domain/database`.

`db_username`

required for* `DB` *mode

The username for the Netezza database.

`db_password`

required for* `DB` *mode

The password for the Netezza database.

OTHER MACRO PARAMETERS

`upload_results`

optional; default = 0

An indicator for whether to upload results to the hdpharmacoepi.org web site.

analysis_num

required if upload_results = 1
If results upload is selected, the analysis number assigned by the hdpharmacoepi.org web site; specifying this number will associate the results uploaded with the proper analysis.

outcome_type

optional; default = DICHOTOMOUS
Experimental.

NOTES

Input cohort data format. The input_cohort dataset must contain only the patient ID, exposure, outcome, and any covariates that should be adjusted for in the outcome model. If the vars_predefined option is not specified, covariates in the input data that are not to be included in the outcome model should be dropped before running the macro or specified in vars_ignore.

Input dimension data format. Each dimension of data (drugs, inpatient procedures, etc.) should be supplied in a unique dataset. This dataset should have at least two fields: the patient identifier and the dimension code. Patients receiving multiple codes, and/or a single code more than one time, should be listed over multiple rows. (For example, in an outpatient procedure dimension, the dimension code would likely be a CPT code.) Additional fields will be ignored.

Covariates. All character covariates in the input cohort will be treated as categorical. Any numeric covariates that should be treated as categorical (such as year of service) should be either converted to character before running the macro or included in the vars_force_categorical list.

EXAMPLE CODE

In the following example code, the required input is printed in bold. Other options are printed as an example but are not required for running the macro.

```
%include "/path/to/macro/directory/hdmacros.sas";

Title1 'High-dimensional propensity score adjustment';
Title2 '(study description)';

%InitHDMacros (
      var_patient_id          = id,
      var_exposure            = exposed,
      var_outcome             = outcome,
      vars_demographic        = age sex race,
      vars_force_categorical  = year,
      vars_ignore             = followup_time,

      top_n                   = 200,
      k                       = 400,

      path_temp_dir           = %QUOTE(/path/to/temp/dir),
      /* remember, no spaces in path_jar_file! */
      path_jar_file           = %QUOTE(/path/to/jar/file/pharmacoepi.jar),
```

```
      outcome_model_deciles   = 0,
      outcome_model_matched   = 1,
      score_type_4            = 0,

      input_cohort            = master_file,
      input_dim1              = drug_claims           generic_name,
      input_dim2              = outpatient_diagnoses  icd9_dx,
      input_dim3              = inpatient_diagnoses   icd9_dx,
      input_dim4              = inpatient_procedures  icd9_proc,
      input_dim5              = outpatient_procedures cpt,
      output_cohort           = scored_cohort,
      result_estimates        = estimates,
      result_diagnostic       = variable_info
);

%DoHDVariableSelection;
%EstimateHDPS(dataset_with_ps);
%RunOutcomeModels(dataset_with_ps, ps);
```

1:1, 1:N, AND 1:R GREEDY MATCHING
1:1, 1:N, AND 1:R NEAREST NEIGHBOR MATCHING
1:1:1 NEAREST NEIGHBOR MATCHING

DESCRIPTION
These macros implement greedy and nearest-neighbor matching in SAS>

FILES REQUIRED
Main SAS file: `matching.sas`
Core program file: `pharmacoepi.jar`
→ ***Due to SAS limitations, the `pharmacoepi.jar` file must be stored on a path without spaces!***

GENERAL INFORMATION
The greedy and nearest-neighbor matching methods each implement 1:1, *n*:1 (variable-ratio), and *r*:1 (fixed-ratio) matching. **In each case, it is assumed that 1 or more referent patients will be matched to a single treated patient; if the opposite is desired, simply reverse the group designations.**

The resulting matched cohort will contain a variable called `set_num`; this variable indicates the matched set. Lower values of `set_num` will tend to be better matches.

MATCHING MACROS

`%match_GreedyMatch(…)`
Performs a 1:1, n:1, or r:1 digit-based greedy match using a modified version of the Parsons greedy matching technique.

`in_dataset` ***required***
Name of a SAS library in which the source data (cohort to be matched) resides.

`out_dataset` ***required***
Name of a SAS library that will contain the matched cohort. If this dataset exists, it will be overwritten.

`working_directory`
required
The path to a directory where the macro can store temporary files. There should be enough space in the directory to hold a second copy of the input cohort.

`var_patient_id` ***required***
Name of unique patient identifier variable.

`var_exposure` ***optional; default=***`exp`
Name of binary exposure variable.

`var_ps` ***optional; default=***`ps`
Name of variable containing the propensity score (or other continuous value) to match on.

`exp_groups` ***optional; default=***`0 1`
A list of values that define the *two* exposure groups. **The first group listed will be assumed to be the referent group**.

`start_digit` ***optional; default=***`5`
The first digit at which the matching algorithm will attempt to match. For example, with the default of 5, it will first match propensity scores that are equal to the first 5 digits.

`end_digit` ***optional; default=***`1`
The digit at which the matching will stop. For example, with the default of 1, the matching algorithm will last match patients whose propensity scores are equal to the first digit.

`ratio` ***optional; default=***`1`
The desired match ratio, e.g,. the value of *n* or *r* in *n*:1 or *r*:1 matching.

`fixed_ratio` ***optional; default=***`0`
An indicator (values 1 or 0) for whether fixed ratio (*r*:1) or variable ratio (*n*:1) matching is desired.

`%match_NearestNeighborMatch(…)`
Performs a 1:1, n:1, or r:1 nearest neighbor match.

`in_dataset` ***required***
Name of a SAS library in which the source data (cohort to be matched) resides.

`out_dataset` ***required***
Name of a SAS library that will contain the matched cohort. If this dataset exists, it will be overwritten.

`working_directory`
required
The path to a directory where the macro can store temporary files. There should be enough space in the directory to hold a second copy of the input cohort.

`var_patient_id` ***required***
Name of unique patient identifier variable.

`var_exposure` ***optional; default=***`exp`
Name of binary exposure variable.

`var_ps` ***optional; default=***`ps`
Name of variable(s) containing the propensity score(s) (or other continuous value) to match on.

`exp_groups` ***optional; default=***`0 1`
A list of values that define the *two or three* exposure groups. **In two-way matching, the first group listed will be assumed to be the treatment group**.

`caliper` ***optional; default=***`0.05`
The maximum distance between two patients allowable for a match. In two-way matching, recommendations for this value can be found in papers by Peter Austin. In three-way matching, this value sum of the Euclidean distance among the three patients (the perimeter of the triangle defined by the three patients). No specific value is yet recommended, but 3x the value used in two-way matching may be a good starting point. Caliper must be specified as a constant (e.g., `0.05`) or a macro variable (e.g. `&caliper`).

`ratio` ***optional; default=***`1`
The desired match ratio, e.g,. the value of *n* or *r* in *n*:1 or *r*:1 matching.

`fixed_ratio` ***optional; default=***`0`
An indicator (value 1 or 0) for whether fixed ratio (*r*:1) or variable ratio (*n*:1) matching should be performed.

`balanced` ***optional; default=***`0`
An indicator (value 1 or 0) for whether nearest-neighbor matching should be balanced (requiring that referent matches alternate between being to the right and left of the treated patient).

EXAMPLE CODE

```
%INCLUDE "/path/to/toolbox/sas/java_utils.sas";
%INCLUDE "/path/to/toolbox/sas/matching.sas";

/* remember, no spaces in the path! */
%toolbox_Start(%QUOTE(/path/to/toolbox/java/pharmacoepi.jar));


/* Estimate 2-group propensity score */
PROC LOGISTIC DATA=input_cohort DESCENDING;
    MODEL exposure = covariate1 covariate2 ... ;
    OUTPUT OUT=match_input_cohort PRED=prob;
RUN;

/* Run 1:1 nearest neighbor match */
%match_NearestNeighborMatch ( match_input_cohort,
                              match_output_cohort,
                              %QUOTE(c:\temp\),
                              var_patient_id = id,
                              var_exposure = exp,
                              var_ps = prob,
```

```
                                exp_groups = 0 1,
                                caliper=0.05,
                                ratio=3,
                                fixed_ratio = 1
                                );

/* Estimate 3-group propensity score with polytomous */
/* logistic regression */
PROC LOGISTIC DATA=input_cohort_3way ORDER=data DESCENDING;
    CLASS expcat;
    MODEL expcat(ref="1") = covariate1 covariate2 ... / LINK=glogit;
    OUTPUT OUT=input_3way_cohort_long(KEEP=id expcat _level_ prob) PRED=prob;
RUN;

/* The output from above will contain one row per patient per */
/* level.  Transform to one row per patient with one variable per level */
PROC TRANSPOSE DATA=input_3way_cohort_long OUT=input_3way_cohort
                 PREFIX=prob;
    BY id expcat;
RUN;

/* Run 1:1:1 nearest neighbor match */
%match_NearestNeighborMatch ( input_3way_cohort,
                              output_3way_cohort,
                              %QUOTE(c:\temp\),
                              var_patient_id = id,
                              var_exposure = expcat,
                              var_ps = prob1 prob2,
                              exp_groups = 1 2 3,
                              caliper=0.05,
                              ratio=3,
                              );


/*
Perform analysis: crude, adjusted, conditioned on match set (set_num) ...
*/

...
```

EPIDEMIOLOGY TABLE CREATION

DESCRIPTION
This macro implements epidemiology tables in SAS.

Note. There are two ways to define tables: through a series of macro calls, or via Excel spreadsheet(s). They are functionally equivalent, though the latter may be more convenient. While the methods should be able to be mixed-and-matched, it is probably best to use one or the other.

SYSTEM REQUIREMENTS
The macro requires SAS Version 9.2 or later. It will not run on earlier versions of SAS.

FILES REQUIRED
Main SAS file: `table_creator.sas`
Required include file: `java_utils.sas`
Core program file: `pharmacoepi.jar`
→ ***Due to SAS limitations, the `pharmacoepi.jar` file must be stored on a path without spaces!***

EXCEL SPREADSHEET FORMAT
The Excel spreadsheet must be formatted in a specific way in order to allow the Table Creator to extract table definitions. For convenience, a sample spreadsheet is provided. Excel files can be in either the older XLS or the newer XLSX format.

The sections of the spreadsheet are each color-coded; the choice of colors is very specific and must not be altered. The color refers to the **Fill Color** of the cell. Opening the Fill Color dialog box and hovering over the color swatches will reveal color names.

Section	Color	Notes
Table ID	N/A	The Table ID will be the worksheet name. Each worksheet in the Excel file will define a new table.
Table Name	Yellow	The name and description of the table, to be printed as the table's header.
Row Header	Orange	The text to display above the rows. Example: Characteristics Measured.
Columns	Blue	The table's columns. If there are multiple rows of column definitions, then columns and sub-columns will be created. The column definition should consist of the column text, and, if you want to refer to a column (such as when filling cells), also a column ID. The ID should be set off from the text by two slashes (//). Example: Treatment // TX. "Treatment" will display, while TX will be assigned as the column ID. You can

		also specify a footnote reference, set off by two carats (^^). Example: Treatment // TX ^^ A. This will create a reference to footnote A, as described below.
Rows	Red	The table's rows. If there are multiple columns of row definitions, then rows and sub-rows will be created. The row definition should consist of the row text, and, if you want to refer to a row (such as when filling cells), also a row ID. The ID should be set off from the text by two slashes (//). Example: Age // AGE. "Age" will display, while AGE will be assigned as the row ID. You can also specify a footnote reference, set off by two carats (^^). Example: Age // AGE ^^ A. This will create a reference to footnote A, as described below.
Footnotes	Green	The footnote definitions. Each definition starts with a footnote reference, as used above, followed by two carats (^^), followed by the footnote text. Example: A ^^ As measured in the 180 days prior to the index date.

TABLE CREATION MACROS

`%table_Init(table_library)`
Initializes the table creation functions. Call this macro before beginning any other table operations.

`table_library` ***required***
Name of a SAS library in which the table data files can be stored.

`%table_AddTablesFromExcel(workbook_path);`
Adds all the tables defined in the Excel spreadsheet. See the section on "Excel Spreadsheet Format" for more information.

`workbook_path` ***required***
A file path to the Excel workbook with the table(s) defined.

`%table_AddTable(table_id, description);`
Adds a table.

`table_id` ***required***
Name of a the new table; internal name only. Should be one word with no special characters.

`description` ***required***
Description of the table. Will be included in the output.

`%table_Copy(source, dest);`

After creating a table, copies the structure (rows, columns) of a source table to a destination table.

`source` ***required***
Name of a the table to copy from.

`dest` ***required***
Name of a the table to copy to.

`%table_AddRow(table_id, row_id, description, parent_id=.);`
`%table_AddCol(table_id, col_id, description, parent_id=.);`
Adds a row (column) to the table.

`table_id` ***required***
ID of an existing table.

`row_id (col_id)` ***required***
Unique ID of the new row in this table.

`description` ***required***
Description (text) of the row (column). Will be included in the output.

`parent_id` ***optional; default=.***
Used to create sub-rows (columns). ID of an existing row (column) in this table, or "." to indicate no parent.

`%table_AddHeaderRow(table_id, row_id, description, parent_id=.);`
`%table_AddHeaderCol(table_id, col_id, description, parent_id=.);`
Adds a header (bolded) row to the table. See above for syntax.

`%table_AddEmptyRow(table_id, parent_id=.);`
Adds an empty (blank) row to the table. See above for syntax.

`%table_AddToRowDescription(table_id, row_id, description);`
`%table_AddToColDescription(table_id, col_id, description);`
Adds text to the row description. Used to add information to a row's output text after the row has been created.

`table_id` ***required***
ID of an existing table.

`row_id` ***required***
ID of an existing row in this table.

`description` ***required***
Description (text) to be added to the row's existing description. Will be included in the output.

%table_SetRowsTitle(table_id, description);
Adds text above the first table row.

`table_id` ***required***
ID of an existing table.

`description` ***required***
Description (text) to be printed above the first row of the table.

%table_FillCell(table_id, row_id, col_id, contents);
Sets the specified cell to output the specified contents.

`table_id` ***required***
ID of an existing table.

`row_id` ***required***
ID of an existing row. Note that in tables with sub-rows, this must indicate a row in the lowest level of nesting.

`col_id` ***required***
ID of an existing column. Note that in tables with sub-columns, this must indicate a column in the lowest level of nesting.

`contents` ***required***
Contents to output. Contents will be output as-is and can contain HTML.

%table_FillCellNum(table_id, row_id, col_id, num, num_decimals = 0);
Fill a cell with formatted numeric information. Syntax as above, plus:

`num` ***required***
The number to insert into the cell.

`num_decimals` ***optional; default = 0***
The number of decimal places to print.

%table_FillCellCommaNum(table_id, row_id, col_id, num, num_decimals = 0);
Fill a cell with formatted numeric information, adding commas. Syntax as above.

%table_FillCellPct(table_id, row_id, col_id, pct, num_decimals = 1, multiply=0);
Fill a cell with formatted percentage information. Syntax as above, plus:

`pct` ***required***
The percentage to insert into the cell.

`num_decimals` ***optional; default = 1***
The number of decimal places to print.

`multiply` ***optional; default = 0***
An indicator (1 or 0) specifying whether to multiply pct by 100 before printing.

`%table_FillCellMeanSD(table_id, row_id, col_id, mean, sd, num_decimals = 1, percentage = 0);`

Fill a cell with a formatted mean and standard deviation. Syntax as above, plus:

`mean` ***required***
The mean to insert into the cell.

`sd` ***required***
The standard deviation to insert into the cell.

`num_decimals` ***optional; default = 1***
The number of decimal places to print.

`percentage` ***optional; default = 0***
An indicator (1 or 0) specifying whether the mean and SD should be formatted as a percentage.

`%table_FillCellMedianIQR(table_id, row_id, col_id, median, q1, q3, num_decimals = 1, percentage = 0);`

Fill a cell with a formatted median and interquartile range. Syntax as above, plus:

`median` ***required***
The median to insert into the cell.

`q1` ***required***
The lower quartile to insert into the cell.

`q3` ***required***
The upper quartile to insert into the cell.

`num_decimals` ***optional; default = 1***
The number of decimal places to print.

`%table_FillCellEstCI(table_id, row_id, col_id, est, ci_low, ci_high, num_decimals = 2);`

Fill a cell with a formatted estimate and a confidence interval. Syntax as above, plus:

`est` ***required***
The point estimate to insert into the cell.

`ci_low` ***required***
The lower confidence interval of the point estimate to insert into the cell.

`ci_high` ***required***
The upper confidence interval of the point estimate to insert into the cell.

`num_decimals` ***optional; default = 2***
The number of decimal places to print.

`%table_FillCellOR(table_id, row_id, col_id, est, ci_low, ci_high, num_decimals = 2);`
Fill a cell with a formatted odds (hazard, rate) ratio and a confidence interval. Syntax as above.

`%table_FillCellRD(table_id, row_id, col_id, est, ci_low, ci_high, num_decimals = 2);`
Fills a cell with a formatted risk (rate) difference and a confidence interval. Syntax as above.

`%table_FillCellFromPHREG(table_id, row_id, col_id, phreg_out_ds, variable, num_decimals = 2);`
Fills a cell with a formatted hazard ratio using output from PROC PHREG. Syntax as above, plus:

`phreg_out_ds` ***required***
The dataset produced by PHREG using ODS OUTPUT XXX=phreg_out_ds.

`variable` ***required***
The name of the variable of interest in the dataset (usually the exposure).

`%table_FillCellFromLOGISTIC(table_id, row_id, col_id, logistic_out_ds, variable, num_decimals = 2);`
Fills a cell with formatted odds ratio using output from PROC LOGISTIC. Syntax as above, plus:

`logistic_out_ds` ***required***
The dataset produced by LOGISTIC using ODS OUTPUT OddsRatios = logistic_out_ds.

`%table_FillCellFromGENMOD(table_id, row_id, col_id, genmod_out_ds, variable, exponentiate = 0, num_decimals = 2);`
Fill a cell with a formatted estimate and CI using output from PROC GENMOD. Syntax as above, plus:

`genmod_out_ds` ***required***
The dataset produced by GENMOD using ODS OUTPUT XXX = `genmod_out_ds.`

`exponentiate` ***optional; default = 0***
An indicator (1 or 0) specifying whether to exponentiate the figures in the GENMOD output before filling the cell.

%table_FillCellMedianFromUNIVAR(table_id, row_id, col_id, univariate_out_ds, variable, num_decimals = 2, percentage = 0);

Fills a cell with formatted a median and interquartile range using ODS output from UNIVARIATE. Syntax as above, plus:

`univariate_out_ds` ***required***
The dataset produced by UNIVARIATEusing ODS OUTPUT Quartiles = `univariate_out_ds`.

%table_FillCellMeanFromUNIVAR(table_id, row_id, col_id, univariate_out_ds, variable, num_decimals = 2, percentage = 0);

Fills a cell with a mean and SD using ODS output from UNIVARIATE. Syntax as above, plus:

`univariate_out_ds` ***required***
The dataset produced by UNIVARIATEusing ODS OUTPUT Moments = `univariate_out_ds`.

%table_FillCellMedianFromMEANS(table_id, row_id, col_id, means_out_ds, variable, num_decimals = 2, percentage = 0);

Fills a cell with formatted a median and interquartile range using ODS output from MEANS. MEANS should be run with the options including MEDIAN, Q1, and Q3. Syntax as above, plus:

`means_out_ds` ***required***
The dataset produced by MEANS using ODS OUTPUT Summary = `means_out_ds`.

%table_FillCellMeanFromMEANS(table_id, row_id, col_id, means_out_ds, variable, num_decimals = 2, percentage = 0);

Fill a cell with a mean and SD using ODS output from MEANS. MEANS should be run no options, or options including MEAN and STDDEV. Syntax as above, plus:

`means_out_ds` ***required***
The dataset produced by MEANS using ODS OUTPUT Summary = `means_out_ds`.

%table_FillCellNumFromFREQ(table_id, row_id, col_id, freq_out_ds, where=);

Fill a cell with a frequency (number) using ODS output from FREQ. Syntax as above, plus:

`freq_out_ds` ***required***
The dataset produced by FREQ using ODS OUTPUT CrossTabFreqs = `freq_out_ds`.

`where` ***optional; default = (1=1)*** *[all rows]*
The WHERE clause used to identify the single value desired from the PROC FREQ. If FREQ is run with `TABLE exposure*covariate`, then one WHERE clause would be `WHERE=(exposure=1 AND covariate=1)`.

%table_FillCellRowPctFromFREQ(table_id, row_id, col_id, freq_out_ds, where=, num_decimals = 2);
%table_FillCellColPctFromFREQ(table_id, row_id, col_id, freq_out_ds, where=, num_decimals = 2);
Fill a cell with a percentage of the row (column) using ODS output from FREQ. Syntax as above.

%table_GetCell(table_id, row_id, col_id);
Get the value of a cell that has been filled with one of the %table_FillCell macros. Syntax as above.

%table_Output(table_id, filepath);
%table_OutputAll(filepath);
Outputs all defined tables to files in filepath. Files will be named using the tables' IDs.

`filepath` ***required***
The path at which to output the tables.

EXAMPLE CODE
In the following example code, the required input is printed in bold. Other options are printed as an example but are not required for running the macro.

```
/* include necessary files */
%include "/path/to/macro/directory/table_creator.sas";
%include "/path/to/macro/directory/java_utils.sas";

/* initialize the Java code */
%javautils_InitClasspathUpdate;
/* remember, no spaces in the path! */
%javautils_AddToClasspath(/path/to/java/files/pharmacoepi.jar);

/* create a library where the table data files will be stored */
LIBNAME tables "/path/to/table/output/library";


/* initialize the table code */
%table_Init(tables);


/* ------------------------------------ */
/* OPTION 1: Define tables from Excel   */
/* ------------------------------------ */
%table_AddTablesFromExcel(%NRBQUOTE(c:\files\workbook.xls));



/* ------------------------------------ */
/* OPTION 2: Define tables manually     */
/* ------------------------------------ */

/* add a new Table 1 */
%table_AddTable(TABLE1, %NRBQUOTE(Table 1));
```

```
/* create columns for exposed patients */
%table_AddHeaderCol(TABLE1, EXPOSED, %NRBQUOTE(Exposed));
/* two sub-columns: mean and median */
%table_AddCol(TABLE1, EXPOSED_MEAN, %NRBQUOTE(Mean), parent_id=EXPOSED);
%table_AddCol(TABLE1, EXPOSED_MEDIAN, %NRBQUOTE(Median), parent_id=EXPOSED);

/* create columns for unexposed patients */
%table_AddHeaderCol(TABLE1, UNEXPOSED, %NRBQUOTE(Unexposed));
%table_AddCol(TABLE1, UNEXPOSED_MEAN, %NRBQUOTE(Mean), parent_id=UNEXPOSED);
%table_AddCol(TABLE1, UNEXPOSED_MEDIAN, %NRBQUOTE(Median),
                parent_id=UNEXPOSED);

/* create a row: age */
%table_AddRow(TABLE1, AGE, %NRBQUOTE(Age));


/* ------------------------------------ */
/* Fill the table cells with information */
/* ------------------------------------ */

/* run UNIVARIATE and generate the data for the cells */
ODS OUTPUT Quantiles = quantiles;
ODS OUTPUT Moments = moments;
PROC UNIVARIATE;
    WHERE exposed = 1;
    VAR age;
RUN;

/* fill the exposed mean cell with data from UNIVARAIATE */
%table_FillCellMeanFromUNIVAR(TABLE1, AGE, EXPOSED_MEAN, moments, age);
/* fill the exposed median cell with data from UNIVARAIATE */
%table_FillCellMedianFromUNIVAR(TABLE1, AGE, EXPOSED_MEDIAN, quantiles, age);

/* repeat for unexposed patients */
ODS OUTPUT Quantiles = quantiles;
ODS OUTPUT Moments = moments;
PROC UNIVARIATE;
    WHERE exposed = 0;
    VAR age;
RUN;

%table_FillCellMeanFromUNIVAR(TABLE1, AGE, UNEXPOSED_MEAN, moments, age);
%table_FillCellMedianFromUNIVAR(TABLE1, AGE, UNEXPOSED_MEDIAN, quantiles,
age);


/* ------------------------------------ */
/* Output tables                        */
/* ------------------------------------ */

/* output the table HTML file in the specified directory */
%table_Output(%NRBQUOTE(/path/to/table/output));
```

FREQUENTLY ASKED QUESTIONS (FAQ)

Question: The hd-PS model does not converge; instead, SAS warns of quasi-complete separation of data points. What do I do about this?

Quasi-complete separation is an issue when the goal is to interpret the point estimates of the PS model, rather than just use the predicted values. Since a PS uses only the predicted values, quasi-complete separation is not ideal but does not pose a problem. Those variables for which there is separation will not contribute to the predicted value, but also should not in any way harm the prediction.

Question: I am having memory problems in SAS.

SAS requires configuration of the Java subsystem to allocate the system its needed memory. There are SAS support documents that describe this, including: http://support.sas.com/documentation/cdl/en/grstatgraph/63878/HTML/default/viewer.htm#n099d3bkbc1lg5n1eniitj9b8y1f.htm.

Question: I am getting a classpath-related error.

Check that you specified the path_jar_file variable correctly. **NOTE: For use in SAS, classpaths cannot contain spaces! This is a SAS limitation.**

Question: Where is the core of hd-PS? Can I see the code?

The core components of hd-PS – the variable creation, ranking and selection – are all within the Java code. The Java system allows for much faster and more efficient computation, and allows for parallel computation of each of the dimensions. (Version 1 of hd-PS was implemented entirely in SAS and was rather slow.) The Java source code is contained within the pharmacoepi.jar file and is released under and open-source license.