Software Manual Version 1.0.2

June 18, 2018

(Requires SAS 9.4 or higher)

Table of Contents *Page*

In publications, please cite as:

Jackson JW. Diagnostics for confounding of time-varying and other joint exposures. *Epidemiology* 2016; 27(6):859-69

© The President and Fellows of Harvard College

Authored by: Erin Schnellinger, Linda Valeri, and John W. Jackson

THE STANDARD MIT LICENSE APPLIES:
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Overview

This software implements three diagnostics for confounding that can be used in sequence (see Jackson 2016). These apply to any study of multivariate exposures e.g. time-varying exposures, direct effects, interaction, and censoring. The first two diagnostics pertain to the nature of confounding in the data, while the third is meant to examine residual confounding after applying inverse probability weighting. The third diagnostic can also be used to examine residual confounding within propensity-score strata (when these are used in place of covariates in the parametric g-formula). We recommend that the diagnostics be applied to covariates that investigators will use to control for confounding. We provide tools to help focus each diagnostic on the relevant covariate history (used to control for confounding).

- Diagnostic 1 is a generalization of a "Table 1" for multivariate exposures (i.e. multiple exposures that are distinct in timing or character). It examines whether the prior covariate means are the same across exposure groups, among persons who have followed a particular exposure trajectory up to that point in time. Like a "Table 1" it is meant to describe whether exposure groups have different distributions of *prior* covariates.

- Diagnostic 2 is meant to inform whether or not g-methods are necessary to control for confounding. G-methods are required if any covariate measurement is associated with a prior exposure after adjusting for covariates that precede the exposure. Here, the diagnostic describes whether the covariate mean differs across prior exposure groups, after adjustment for covariates (that precede exposure) through inverse probability weighting or propensity score stratification.

- Diagnostic 3 is meant to be applied after investigators have applied the earlier diagnostics and have chosen to use g-methods. The form of Diagnostic 3 is similar to that of Diagnostic 1 in that it is a generalized "Table 1." The difference is that it is applied to an inverse probability weighted population, where the weights are typically designed to remove confounding. It can also be applied to evaluate residual confounding within levels of time-varying propensity-score strata.

The SAS-based macros presented here allow users to construct balance tables and trellised plots. They can be used to diagnose multivariate exposures that are binary or categorical. The macros can also accommodate right-censoring (including the extension for Diagnostic 2 described in Jackson 2016). While the SAS-based macros could in theory be applied to diagnose continuous exposures, it may be better to use a regression model for this (see sample SAS code at the end of the example).

The user can request time-specific tables and plots for all times, or a subset of selected times. One can alternatively request summary metrics that average over non-referent exposure values, propensity-score strata and/or exposure history, time, and segments of distance, in that order (see Jackson 2016).

To use these macros, users will need a basic understanding of data structures and data manipulation in SAS. Note that the software will not run on SAS versions below 9.4.

Users should be aware that applying these macros to datasets with many observations, follow-up times, and covariates may require substantial computing memory. The approach we implement here is described in Jackson 2016 and could be adapted for other software. In addition, output from other software can be imported and used in the plotting macro provided here.

Format of Input Data

The software is designed to use data that analysts may readily have on hand in analyses of time-varying exposures. This includes time-varying exposures, time-varying covariates, time-varying weights (when applicable), and time-varying propensity-score strata (when applicable). Time-varying exposure history strata may also be required, and we provide functions to create these from the exposure variables.

These time-indexed variables should be organized into a "wide" data format where each row uniquely indexes a single subject's data, so that columns index measurement of each variable at each time. The indices should be indicated with an underscore suffix followed by the time, e.g. "variable_1", "variable_2", "variable_3." No other underscores should appear in the variable name. It is fine if variables were measured at different times (e.g. "varA_1", "varB_2", "varB_4", "varC_3", "varC_5"). The `%widen()` macro is provided to assist in this task.

The analysis proceeds as follows:

In a preliminary step, if person-time data needs to be reformatted into "wide" data format, use the `%widen()` macro. Similarly, if exposure history is needed, create time-indexed exposure history variables from the time-indexed exposure variables via the `%makehistory_one()` or `%makehistory_two()` macros.

1. Restructure this wide dataset into a "tidy" long dataset: `%lengthen()` macro

   By tidy, we mean that a row is uniquely identified by the pairing of exposure and covariate measurement times. This will typically result in an extremely long dataset, and may require substantial computing memory in SAS with many persons, covariates, and/or follow-up times.
   See Note (iii) for a solution to avoid memory issues with large and rich datasets.

2. From the long dataset, create a covariate balance table: `%balance()` macro

3. Plot the data in the covariate balance table: `%makeplot()` macro

Note (i) These steps should be followed in order as `%lengthen()` produces the dataset required by `%balance()` which produces the dataset used by `%makeplot()`.

Note (ii) These macros automatically save all data sets stored in the WORK library. However, given the substantial amount of computing memory required by the software, it is **highly** recommended that users clean their workspace (via *proc datasets*, for example) prior to running the macros.

Note (iii) When the data have a large number of observations, covariates, and/or measurement times, memory issues can be ameliorated by using `%diagnose()` to iteratively apply `%lengthen()` and `%balance()` by looping over covariates.

Note (iv) One can encode assumptions about which covariates are necessary to adjust for confounding. This is done by removing covariate history that does *not* support exchangeability assumptions, through applying the `%omit_history()` macro to the "tidy" dataset produced by `%lengthen()`, `%balance()`, or `%diagnose()`.

Note (v) Jackson 2016 proposes two methods for obtaining summary averages over person-time. The code here implements the standardization approach. However, regression models can be applied to the "tidy" dataset produced by `%lengthen()`. See the end of the software manual for sample code.

Note (vi) The code can accept a vector of time-indexed censoring indicators (1=censored, 0=otherwise). This can be used regardless of the source of censoring (i.e. an event or some artificial rule defined by the investigator).

Note (vii) Although the code will still run in the presence of missing data, the results may not be easily interpretable (especially when there is missing data in the exposures and their history).

Macro Calls

`%lengthen()` returns a dataset where each record is indexed by the observation identifier, exposure measurement time, exposure value, covariate name, covariate measurement time, and possibly exposure history and/or propensity score strata. Weights will appear as additional columns.

```
%lengthen(
       input              = dataset in wide format,
       output             = name for output dataset,
       diagnostic         = diagnostic of interest e.g. 1, 2, or 3,
       censoring          = use censoring indicators/weights e.g. yes or no,
       id                 = unique observation identifier e.g. id,
       times_exposure     = a vector of exposure measurement times e.g. 0 1 2,
       times_covariate    = a vector of covariate measurement times e.g. 0 1 2,
       exposure           = the root name for exposure measurements e.g. a,
       temporal_covariate = a vector of root names for covariates whose values change over time
                             e.g. l m n o p,
       static_covariate   = a vector of root names for covariates whose values do not change
                            (covariates listed here should not appear in the temporal_covariate
                             argument),
       history            = the root name for history measurements e.g. h,
       weight_exposure    = … for exposure weights e.g. wa,
       censor             = … for censoring indicators e.g. s,
       weight_censor      = … for censoring weights e.g. ws,
       strata             = … for propensity-score strata e.g. e
       )
```

This macro is designed to minimize user input by creating the covariate names automatically based on the *covariate* and *times* arguments. In the example above, it would create a lengthened dataset based on the following covariate measurements:

```
"l_0","m_0","n_0","o_0","p_0",
"l_1","m_1","n_1","o_1","p_1",
"l_2","m_2","n_2","o_2","p_2"
```

Now, it may be that variable “n” is really a static covariate, like sex. In this case, you would only have “n_0” in the dataset, not “n_1” and “n_2”. To specify a covariate like this, omit it from the *temporal_covariate* argument and instead include it in the *static_covariate* argument. Note that the software assumes that static covariates appear in the data with the lowest possible index specified in times. In this example, specifying “n” as a static covariate would create a lengthened dataset based on the following covariate measurements (notice that “n_0” is included but “n_1” and “n_2” are not):

```
"n_0",
"l_0","m_0","o_0","p_0",
"l_1","m_1","o_1","p_1",
"l_2","m_2","o_2","p_2"
```

Note that the software will automatically detect and ignore covariate measurements that are not present within the input dataset. For example, suppose variables “l*”* and “m*”* were only measured at times 0 and 2, and that “l_1” and “m_1” were not present in the dataset. The software would, after detecting their absence, automatically create a lengthened dataset based on the following covariate measurements, excluding “l_1” and “m_1”:

```
"l_0","m_0","n_0","o_0","p_0",
            "n_1","o_1","p_1",
"l_2","m_2","n_2","o_2","p_2"
```

`%balance()` takes the restructured dataset output by `%lengthen()` and returns a covariate balance table (possibly stratified by exposure history and/or propensity-score strata).

```
%balance (
       input                 = restructured dataset
       output                = name for output dataset,
       diagnostic            = diagnostic of interest e.g. 1, 2, or 3,
       approach              = adjustment method e.g. none or weight or stratify,
       censoring             = use censoring indicators/weights e.g. yes or no,
       scope                 = report the entire trellis e.g. all, the diagonal e.g. recent, or
                               a summary e.g. average,
       times_exposure        = vector of exposure measurement times e.g. 0 1 2,
       times_covariate       = vector of covariate measurement times e.g. 0 1 2,
       sort_order            = a vector of root names for all covariates listed in the order in which
                               they should appear in the table (and also plot) e.g. n m o l p. To
                               display covariates in alphabetical order (the default), leave blank or
                               type alphabetical
       exposure              = root name of exposure e.g. a,
       history               = …exposure history e.g. h,
       weight_exposure       = …IP exposure weights e.g. wa,
       weight_censor         = …IP censoring weights e.g. ws,
       strata                = …of propensity score strata e.g. e,
       recency               = an integer for the relative distance between exposures and covariate
                               measurements to focus on (e.g. 0 would represent the same timing). The
                               default is 0 for Diagnostics 1 and 3, and 1 for Diagnostic 2,
       average_over          = summary level for average metrics e.g. standardize over
                               values or history or time or distance,
       periods               = a list of contiguous segments of relative distance to pool over
                               e.g. 0 1:4 5:10 would yield summaries for three segments,
      list_distance          = a vector of distances to retain after averaging over time e.g. 0 2,
      ignore_missing_metric = yes or no depending on whether the user wishes to estimate averages over
                               person-time when there are missing values of the mean difference or
                               standardized mean difference. Missing values for the standardized mean
                               difference can occur when, for example, there is no covariate variation
                               within levels of exposure-history and measurement times. If this
                               argument is set to no and there are missing values, the average will
                               also be missing. If set to yes an average will be produced that ignores
                               missing values.
       metric                = the metric for which the user wishes to ignore missing values as
                                specified in the 'ignore_missing_metric' argument.
       sd_ref                = "yes" or "no" depending on whether the user wishes to use the standard
                               deviation of the reference group when calculating the SMD.
       loop                 = a housekeeping argument the user can ignore. It is automatically set
                               when the balance function is called by the %diagnose()macro described
                               later. The default is set to no.

       )
```

`%makeplot()` takes the covariate balance table produced by `%balance()` and returns a trellised plot described in Jackson 2016.

```
%makeplot (
       input            = output from %balance
       output           = name for output dataset,
       diagnostic       = the diagnostic of interest e.g. 1, 2, or 3,
       approach         = the adjustment method e.g. none or weight or stratify,
       metric           = scale e.g. D for mean difference, SMD for standardized mean
                          difference
       censoring        = use censoring indicators/weights e.g. yes or no,
       scope            = report the entire trellis e.g. all, the diagonal e.g. recent, or
                          a summary e.g. average,
       stratum          = the propensity-score stratum to plot
       average_over     = level of summary for average e.g. values or history or time
                          or distance
                        … additional arguments to control plot formatting parameters, see
                          below
       label_exposure    = common label used for exposure axis in plot (default = “A”)

       label_covariate   = common label used for covariate axis in plot (default = “C”)

       lbound            = lower bound for mean difference or standardized mean difference (default =
                           -1)

       ubound            = upper bound for mean difference or standardized mean difference (default =
                           1)

       ratio             = aspect ratio of plot (default = 2)

       columns_per_page  = number of columns to appear on each page of the output.
                           WARNING: By default, this parameter is set to 1. One can increase this
                           number so that more than one column appears on each page, but if too many
                           columns are forced to be on the same page, SAS may omit some data without
                           alerting the user!

       rows_per_page     = number of rows to appear on each page of the output.
                           WARNING: By default, this parameter is set to 1. One can increase this
                           number so that more than one row appears on each page, but if too many rows
                           are forced to be on the same page, SAS may omit some data without alerting
                           the user!

       text_axis_title   = font size of axis title (default = 8.5)

       text_axis_y       = font size of y-axis values (default = 7.5)

       text_axis_x       = font size of x-axis values (default = 7.5)

       text_strip_y      = font size of y-axis label (default = 10)

       text_strip_x      = font size of x-axis label (default = 10)


       point_size        = size of data points (default = 1.25)
       zeroline_size     = width of the line plotted at mean difference = 0 or standardized mean
                           difference = 0 (default = 0.1)

       refline_size      = width of the lines plotted at the specified fraction of the mean difference
                           or standardized mean difference (default = 0.1)

       refline_limit_a   = position of the lower reference line, specified as a fraction of the mean
                           difference or standardized mean difference (default = -0.25)

       refline_limit_b   = position of the upper reference line, specified as a fraction of the mean
                           difference or standardized mean difference (default = 0.25)

       panel_margin_size = space between each panel in the plot (default = 25)

       axis_title        = title of plot (default = NULL)

       )
```

Additional notes: SAS chooses the color and shape of each data point by default. That is, the first exposure group will always be represented by a blue filled-in circle, the second will always be represented by a red filled-in triangle, etc.

`%widen()` will transform raw data from "long" format (e.g., person-time format) to a "wide" format suitable for `%lengthen()`.

```
%widen(
       input           = dataset in long format e.g., a person-time format,
       output          = name for output dataset,
       id              = unique identifier at the unit (person) level e.g. id,
       time            = unique index for each observation within each unit e.g. time,
       exposure        = the exposure of interest at time t e.g. a,
       covariate       = a vector of covariates at time t e.g. l m n o p,
       history         = variable describing exposure history through time t e.g. h,
       weight_exposure = inverse probability weight for exposure, at or through time t e.g. wa,
       weight_censor   = cumulative inverse probability weight for censoring through time t e.g. ws,
       censor          = censoring indicators at time t e.g. s,
       strata          = propensity score strata at time t e.g. e

)
```

`%makehistory_one()` will create a set of exposure history variables for a time-varying exposure.

`%makehistory.two()` will create a set of joint exposure history variables for each of the two time-varying exposures, properly accounting for their temporal ordering (i.e. exposure "a" precedes exposure "b" at any time t). The new history variables will use the time-indices in the exposure vectors you supply.

See additional notes at end of software manual for more details.

```
%makehistory_one(
       input          = dataset in wide format,
       output         = name for output dataset,
       times          = a vector of measurement times e.g. 0 1 2,
       exposure       = the root name for exposure e.g. a,
       name_history   = desired root name for time-indexed history variables e.g. h,
       group          = an optional baseline variable upon which to segregate the exposure history.
                        This argument provides a way to adjust the metrics for a baseline covariate.
                        For example, in the context of a trial, the grouping variable could be
                        treatment assignment. In the context of a cohort study, this could be site e.g.
                        v.
       )

%makehistory_two(
       input          = dataset in wide format,
       output         = name for output dataset,
       times          = a vector of measurement times e.g. 0 1 2,
       exposure_a     = the root name for the first exposure e.g. a,
       exposure_b     = the root name for the second exposure e.g. z,
       name_history_a = desired root name for the first time-indexed history variables e.g. ha,
       name_history_b = … root name for the second time-indexed history variables e.g. hb,
       group          = an optional baseline variable upon which to segregate the exposure history.
                        This argument provides a way to adjust the metrics for a baseline covariate.
                        For example, in the context of a trial, the grouping variable could be
                        treatment assignment. In the context of a cohort study, this could be site e.g.
                        v.
       )
```

`%omit_history()` will take the dataset produced by `%lengthen()` and remove covariate measurements based on their fixed measurement time or relative distance from exposure measurements (at time t) i.e. ones that do **not** support exchangeability assumptions at time t. The *covariate_name* argument is used to name the covariate whose history you wish to modify. To process the same manipulation for a set of covariates, simply supply a vector of covariate names to *covariate_name*. The *omission* argument determines whether the covariate history is (i) set to missing for certain covariate measurement times (*omission*=*fixed* with *times*=a vector of integers) or (ii) set to missing only for covariate measurement times at or before a certain distance k

from exposure measurement times (*omission*=*relative* with *distance*=some integer) or (iii) set to missing only for covariate measurements that share the same timing as exposure measurements (*omission*=*same_time*). The removed values are set to missing. For example, using the *fixed* omission option for covariate "l" at time 2 will set all data on "l" at time 2 to missing, regardless of the exposure measurement time. In contrast, using the *relative* omission option for covariate "l" with distance 2 will only set to missing data on "l" that is measured two units or more before the exposure measurement time (i.e. $t-2$, $t-3$, $t-4$ and so on). Last, using the *same_time* omission option for covariate "l" will set to missing all data on "l" that is measured at the same time as the exposure. Missing data will be ignored when this dataset is supplied to the `%balance` macro. They will not contribute to the resulting covariate balance table, or to plots produced by `%makeplot()`; nor will they contribute to any summary metrics that are estimated by averaging over person-time.

```
omit_history(
     input           = restructured dataset from %lengthen,
     output          = name for output dataset,
     omission        = type of omission e.g. fixed or relative or same_time,
     covariate_name  = root name of the covariate e.g. m,
     distance        = the distance between exposure and covariate measurements e.g. 2,
     times           = a vector of measurement times for the covariate e.g. 1 2 3
     )
```

`%diagnose()` is a wrapper macro that calls the `%lengthen()` and `%balance()` macros in sequence, either in one step or iteratively across subsets of covariates. In both cases it outputs a dataset that is suitable for plotting via the `%makeplot()` macro. When the user opts to not iterate over covariates, there is no difference between calling the `%lengthen()` and `%balance()` macros one after the other. However, opting to iterate can be a useful way to process large and rich datasets without otherwise requesting as much memory.

```
diagnose(
        input                 = dataset in wide format,
        output               = output dataset for balance table,
        diagnostic            = diagnostic of interest e.g. 1, 2, or 3,
        censoring             = use censoring indicators/weights e.g. yes or no,
        approach              = adjustment method e.g. none or weight or stratify,
        scope                 = report the entire trellis e.g. all, the diagonal e.g. recent, or a
                                summary e.g. average,
        id                    = unique observation identifier e.g. id,
        times_exposure        = a vector of exposure measurement times e.g. 0 1 2,
        times_covariate       = a vector of covariate measurement times e.g. 0 1 2,
        exposure              = the root name for exposure measurements e.g. a,
        temporal_covariate    = a vector of root names for covariates whose values change over time
                                e.g. l m n o p,
        static_covariate      = a vector of root names for covariates whose values do not change
                                (covariates listed here should not appear in the temporal_covariate
                               argument),
        sort_order            = vector of root names for all covariates listed in the order in which
                                they should appear in the table (and also plot) e.g.
                                n m o l p. To display covariates in alphabetical order (the default),
                                leave blank or type alphabetical,
        history               = the root name for history measurements e.g. h,
        weight_exposure       = … for exposure weights e.g. wa,
        censor                = … for censoring indicators s,
        weight_censor         = … for censoring weights e.g. ws,
        strata                = … for propensity-score strata e.g. e,
        recency               = an integer for the relative distance between exposures and covariate
                                measurements to focus on (e.g. 0 would represent the same timing). The
                                default is 0 for Diagnostics 1 and 3, and 1 for Diagnostic 2,
        average_over          = summary level for average metrics e.g. standardize over
                                values or history or time or distance,
        periods               = a list of contiguous segments of relative distance to pool over
                                e.g. 0 1:4 5:10 would yield summaries for three segments,
        list_distance         = a vector of distances to retain after averaging over time e.g. 0 2,
        ignore_missing_metric = yes or no for whether the user wishes to estimate
                                averages over person-time when the balance metric has missing values.
                                For example, the standardized mean difference will be missing when
                                there is no covariate variation within levels of exposure-history and
                                measurement times. When this argument is set to no and there are
```

```
                           missing values, the average will also be missing. If set to yes an
                           average will be produced that ignores missing values.
metric                   = the metric for which the user wishes to ignore missing values as
                           specified in the ‘ignore_missing_metric’ argument.
loop                     = yes to iteratively apply %balance() and %lengthen() or no to process
                           all covariates and measurement times at once.
)
```

There is an important change in workflow when users wish to use the `%diagnose()` macro and also remove irrelevant covariate history from the balance table calculations and plot. Users who wish to do so will need to apply the `%omit.history()` function to the dataset output by `%diagnose()`. If the user wants to remove covariate history while averaging metrics over time or distance, the user will first have to call the `%diagnose()` macro with the *scope* argument set to *all*. The user would then apply the `%omit.history()` macro as many times as desired, and then use the `%apply.scope()` macro described next to average the metrics over person-time. This workflow change allows users to ensure that the summary metrics ignore covariate history the user deems irrelevant to confounding.

`%apply.scope()` is a helper macro that will take a dataset output by `%balance()` or `%diagnose()`, where the *scope* argument in those functions was set to *all*, and subset the table to covariate balance metrics at a certain distance (e.g. a certain recency) or produce estimates that average over person-time. This function is only useful when a user wishes to focus on proximal covariate balance metrics or produce summary estimates via `%diagnose()`, but also needs to remove covariate history that is irrelevant to confounding. In this situation, the user first applies the `%diagnose()` macro with the *scope* argument set to *all*, then applies the `%omit.history()` macro, followed by the `%apply.scope()` macro.

```
apply.scope(
            input                 = dataset output by %diagnose() or %balance() function,
            output                = output dataset for balance table,
            diagnostic            = diagnostic of interest e.g. 1, 2, or 3,
            approach              = adjustment method e.g. none or weight or stratify,
            scope                 = report the entire trellis e.g. all, the diagonal e.g. recent, or a
                                    summary e.g. average,
            recency               = an integer for the relative distance between exposures and
                                    covariate measurements to focus on (e.g. 0 would represent the
                                    same timing). The default is 0 for Diagnostics 1 and 3, and 1 for
                                    Diagnostic 2,
            average_over          = summary level for average metrics e.g. standardize over
                                   values or history or time or distance,
            periods               = a list of contiguous segments of relative distance to pool over e.g.
                                    0 1:4 5:10 would yield summaries for three segments,
            list_distance         = a vector of distances to retain after averaging over time e.g. 0 2,
            sort_order            = vector of root names for all covariates listed in the order in
                                    which they should appear in the table (and also plot) e.g. n m o l
                                    p. To display covariates in alphabetical order (the default),
                                    leave blank or type alphabetical,
            ignore_missing_metric = yes or no depending on whether the user wishes to
                                    estimate averages over person-time when there are missing values
                                    of the mean difference or standardized mean difference. Missing
                                    values for the standardized mean difference can occur when, for
                                    example, there is no covariate variation within levels of
                                    exposure-history and measurement times. If this argument is set to
                                    no and there are missing values, the average will also be missing.
                                    If set to yes an average will be produced that ignores missing
                                    values.
            metric                = the metric for which the user wishes to ignore missing values as
                                    specified in the ‘ignore_missing_metric’ argument.
                  )
```

Required Arguments for Macro Calls

For all macros users must specify the *diagnostic*, *approach*, *scope*, and *censoring* arguments. Depending on how these arguments are specified, other arguments may be required. Note that the user should not include any quotation marks when specifying the macro arguments; if an argument contains a vector or list of values, separate each value with a space (e.g. times_exposure = *0 1 2*).

Required arguments for `%lengthen()` by diagnostic, approach, and censoring arguments			
Diagnostic	***Approach***	***Censoring***	***Additional Required Arguments (in addition to Scope)***
1	none	no	id, exposure, temporal_covariate, times_exposure, times_covariate , history
		yes	(previous) + censor
2	weight	no	id, exposure, temporal_covariate, times_exposure, times_covariate, history, weight_exposure
		yes	(previous) + censor
	stratify	no	id, exposure, temporal_covariate, times_exposure, times_covariate, strata
		yes	(previous) + censor
3	weight	no	id, exposure, temporal_covariate, times_exposure, times_covariate, history, weight_exposure
		yes	(previous) + censor
	stratify	no	id, exposure, temporal_covariate, times_exposure, times_covariate, history, strata
		yes	(previous) + censor

Required arguments for `%balance` by diagnostic, approach, and censoring arguments			
Diagnostic	***Approach***	***Censoring***	***Additional Required Arguments (in addition to Scope)***
1	none	no	exposure, history, times_exposure, times_covariate
		yes	(previous)
2	none	no	times_exposure, times_covariate
		yes	(previous) + censor
2	weight	no	exposure, history, times_exposure, times_covariate, weight_exposure
		yes	(previous) + censor
	stratify	no	exposure, times_exposure, times_covariate, strata,
		yes	(previous) + censor
3	weight	no	exposure, history, times_exposure, times_covariate, weight_exposure
		yes	(previous) + censor
	stratify	no	exposure, history, times_exposure, times_covariate, strata
		yes	(previous) + censor

Required arguments for `%makeplot()` by diagnostic and approach arguments		
Diagnostic	***Approach***	***Additional Required Arguments (in addition to Scope and Metric)***
1	none	---
2	weight	---
	stratify	stratum
3	weight	---
	stratify	stratum

Note (i) The `%makeplot()` macro also requires the *metric* argument.

Note (ii) For the `%balance()` and `%makeplot()` macros, specifying *scope*=*average* will require you to specify an option for the *average_over* argument. If you chose *average_over*=*strata* then you do not need to choose a value for the *stratum* argument.

Note (iii) Specifying *scope*=*recent* will allows you to specify an option for the *recency* argument in `%balance()` i.e. compute metrics at a specific exposure-covariate distance of your choosing.

Note (iv) `%diagnose()`has the combined requirements of `%lengthen()` and `%balance()`.

Additional Notes

Format of initial dataset

- As stated earlier, the input dataset should have one record per observation (wide format) with the timing of variables indexed by an underscore followed by the time index (**underscores should NOT appear anywhere else in the variable name**). Any indexing scheme can be used (e.g. `"var_1","var_4","var_9")`, but it may be easiest to assign zero as the baseline index and increase it by one the unit for each subsequent measurement (e.g. `"var_0","var_1","var_2")`.

- The common referent value—to which all other exposure levels are compared—should be coded as the lowest value.

- Censored data should contain a vector of time-indexed censoring indicators (1=censored, 0 otherwise) for the `%lengthen()` macro.

Alignment of Censoring weights

- Generally speaking, the code asks for separate exposure and censoring weights. This is so because the `%lengthen()` macro will align censoring weights with exposure times, in the case of Diagnostics1 and 3, or with covariate times in the case of Diagnostic 2.

- The `%balance()` macro takes the product of exposure and censoring weights during the estimation process.

Time-indices for multivariate exposures and point exposures

- The macros provided here were developed for time-varying exposures and treat covariates as if they precede exposure when they share the same time index. What follows next is a workaround for multivariate exposures that generally applies when, for some times, exposures precede covariates.

  For multivariate joint exposures, some covariates L may intercede between the exposures, as in the example of exposures A and Z in the eAppendix of the Jackson 2016. Specifically, it may be the case that (i) at each time t, exposure $A(t)$ affects covariates $C(t)$ which affect exposure $Z(t)$, and (ii) covariates $C(t)$ affect subsequent exposures $A(t+k)$ and $Z(t+k)$ and also the outcome Y. The functions could be used as they are to assess confounding for the second exposure Z (since both covariates L and C precede Z); a workaround to assess confounding for the first exposure A would be to increase, by one unit (or some value appropriate for the data's indexing scheme), the indices for all covariates L that occur after A for any given time t (i.e. covariate index $\rightarrow$ covariate index+1). The functions can then be used to examine confounding for exposure A. Another approach for exposure A would be to remove the history on L measured at the same time as the exposure A using `%omit_history()` on the dataset produced by `%lengthen()`, but this only works for Diagnostics 1 and 3.

- The code can also be tricked to handle multivariate point exposures by simply adding a subscript "_0" to each exposure and covariate when using `%lengthen()`, and then specifying "0" for exposure and covariate times when using `%balance()`.

Multivariate time-varying exposures or point exposures

When the exposure is multivariate, the idea is to diagnose each exposure separately (see eAppendix of Jackson 2016). From the perspective of using the SAS-macros, the only difference is to use exposure history based on all exposures that comprise the multivariate exposure. It is important that such joint exposure history accurately reflect the ordering of each component exposure. The macro `%makehistory_two()` creates an appropriate joint exposure history for each of two exposures, assuming that exposures in its argument *list_exposure_a* (e.g. A) precede those in *list_exposure_b* (e.g. Z) at any given index as described in the eAppendix of Jackson 2016. In that example, exposure $A(t)$ always precedes exposure $Z(t)$ such that the joint history of $A(2)$ is $A(1), A(0), Z(0)$ while the joint history of $Z(2)$ is $A(1), A(0), Z(1), Z(0)$. If one exposure does not precede the other, investigators will still need to use an appropriate joint exposure history and can specify either order as desired. Note that the exposure history produced by the macro `%makehistory_two()` will be inappropriate if the relative ordering of $A(t)$ and $Z(t)$ varies over time.

Averaging over person-time

When using the `%balance()` macro, specifying *average_over*=*average* and *average_over*=*time* will return balance metrics for each "distance" value. The output can be subset to specific distances of interest (e.g. *k*=0 and *k*=2) by supplying a vector to *list_distance* (e.g. *0 2*) but this is optional. Specifying *average_over*=*distance*, you can opt to average within segments of distance using the *periods* argument (leaving this blank will average over all distance values). The *periods* argument requires a list of contiguous numeric vectors (e.g. *0 1:4 5:10*). For Diagnostic 3 this would report metrics at time t, averages over times $t-1$ to $t-4$, and averages over times $t-5$ to $t-10$. For Diagnostics 1 and 3 the entire range should lie between 0 and t. For Diagnostic 2 the entire range should lie between 1 and t.

Residual confounding for parametric g-formula w/ propensity score stratification

- Jackson 2016 emphasizes Diagnostic 3 to describe residual confounding in a weighted population (for marginal structural models). This can be accomplished by specifying *diagnostic*=3 and *approach*=*weight*. Any weight can be used.
- In the eAppendix of Jackson 2016, there is an alternative version of Diagnostic 3 that describes residual confounding within a propensity-score stratified population (for a special case of the parametric g-formula). This is done by specifying *diagnostic*=3 and *approach*=*stratify*. Note that one can average these metrics over propensity score strata, exposure history, time, and distance (i.e. by specifying *scope*=*average* and choosing *average_over*=*strata* or higher).

Notes on investigator supplied data

These macros can diagnose confounding for a single exposure or each distinct exposure (in a multivariate exposure) as long as the user provides appropriate history, inverse probability weights, propensity score strata, and censoring indicators. See Jackson 2016 for details. Note that those particular specifications may not apply to the user's causal question (e.g. the user has data where covariates are measured after exposure for every time point, instead of before exposure). The `%makehistory()` macros return nonsense when exposures are partially missing.

A warning on required arguments

The *diagnostic*, *approach*, *scope*, and *censoring* arguments for the `%lengthen()`, `%balance()`, and `%makeplot()` macros are required and must be identical in each macro call. Otherwise, the macros will return errors or incorrect results.

Example Code

Note: A setup file, confoundr_master.sas**, is provided that contains the sample code below for inputting raw data and calling each of the macros. This code requires SAS version 9.4 or higher.**

```
**************************************;
** Load raw data and include macros **;
**************************************;

** Clear contents in the output and log windows **;
   dm 'out;clear;log;clear;';

** Clear titles and footnotes **;
   title;
   footnote;

** Clear temporary data sets in the work library **;
   proc datasets nolist lib=work memtype=data kill;run;quit;

** Macro options **;
   options minoperator;

** Assign a fileref to each confoundr macro **;
   filename widedef "c:\widen.sas";
   filename hist1def "c:\makehistory_one.sas";
   filename hist2def "c:\makehistory_two.sas";
   filename longdef "c:\lengthen.sas";
   filename omitdef "c:\omit_history.sas";
   filename balncdef "c:\balance.sas";
   filename applyscp "c:\apply_scope.sas";
   filename plotdef "c:\makeplot.sas";
   filename diagnos "c:\diagnose.sas";

** Include the confoundr macros **;
  %include widedef;
  %include hist1def;
  %include hist2def;
  %include longdef;
  %include omitdef;
  %include balncdef;
  %include applyscp;
  %include plotdef;
  %include diagnos;

** Read in the sample data **;
proc import datafile="c:\example_sml.csv"
   out=mydata dbms=csv replace;
   getnames=yes;
run;


**********************************************************************;
** Example: Diagnostic 3 for a time-varying exposure without censoring **;
**********************************************************************;

** Preliminary step: Transform person-time data into wide format **;
%widen(input=myPersonTimeData,
       output= myWideData,
       id=id,
       time=time,
       exposure=a,
       covariate=l m n o p,
       weight_exposure=wax
       );


** Preliminary step: Make exposure history **;
%makehistory_one(id=id,
                 input=myWideData,
                 output= mydataPlusHistory,
                 exposure=a,
                 times=0 1 2,
                 name_history=h);
```

```
** Step 1: Restructure the data **;

%lengthen(input= mydataPlusHistory,
          output=mydataLong,
          diagnostic=3,
          censoring=no,
          id=id,
          times_exposure=0 1 2,
          times_covariate=0 1 2,
          exposure=a,
          temporal_covariate=l m o,
          static_covariate=n p,
          history=h,
          weight_exposure=wax
          );

** Example of how to remove relative covariate history **;

%omit_history(input=mydata_long,
              output=mydataLongOmit,
              omission=relative,
              covariate_name=n o,
              distance=2
              );


** Step 2: Create balance table **;

%balance(input=mydataLongOmit,
         output=mytable,
         diagnostic=3,
         approach=weight,
         censoring=no,
         scope=all,
         times_exposure=0 1 2,
         times_covariate=0 1 2,
         sort_order=l m o n p,
         exposure=a,
         history=h,
         weight_exposure=wax
         );


** Step 3: Plot balance metric **;

%makeplot(input=mytable,
          output=myplot,
          diagnostic=3,
          approach=weight,
          metric=SMD,
          scope=all
          );

** The following formatting arguments for %makeplot() are optional (defaults shown). **;

label_exposure=A,                       /* exposure label */
label_covariate=C,                      /* covariate label */
lbound=-1,                              /* lower bound for x-axis */
ubound=1,                               /* upper bound for x-axis */
ratio=2,                                /* plot aspect ratio */
columns_per_page=1,                     /* number of columns to appear on each page of the output */
rows_per_page=1,                        /* number of rows to appear on each page of the output */
text_axis_title=8.5,                    /* title font size */
text_axis_y=7.5,                        /* y-axis (covariate names) font size */
text_axis_x=7.5,                        /* x-axis font size */
text_strip_y=10,                        /* row panel label font size */
text_strip_x=10,                        /* column panel label font size */
point_size=1.25,                        /* dot size */
zeroline_size=.1,                       /* thickness of zero line on x-axis */
refline_size=.1,                        /* thickness of reference line on x-axis */
refline_limit_a=-.25,                   /* location for reference line 1 on x-axis */
refline_limit_b=0.25,                   /* location for reference line 2 on x-axis */
panel_margin_size=25,                   /* space between panels */
axis_title=Mean Difference              /* or Standardized Mean Difference (x-axis title) */
```

```
** Step 4: Save balance table and plot **;
```

To permanently save any of the output datasets created by the SAS macros, first create a new permanent library in which to store the dataset(s). Then save the dataset(s) in this new library:

```
** Create a new permanent library for the output dataset(s) **;

libname outs "c:\ ";


** Save the desired dataset in this new permanent library **;

data outs.mytable;
   set work.mytable;
run;
```

To save the final plot in PDF format, ods statements can be used. The initial "ods pdf file" statement must appear before the `%makeplot()` macro is called, while the "ods pdf close" command should be written after the `%makeplot()` macro call:

```
ods pdf file="c:\ myplot.pdf";

    %makeplot(input=mytable,
              output=myplot,
              diagnostic=3,
              approach=weight,
              metric=SMD,
              scope=all
              );

ods pdf close;




*****************************************************;
** Example of Regression Approach for Diagnostic 1 **;
*****************************************************;

** Create long dataset **;

%lengthen(input=mydataPlusHistory,
          output=longReg,
          diagnostic=1,
          censoring=no,
          id=id,
          times_exposure=0 1 2,
          times_covariate=0 1 2,
          exposure=a,
          temporal_covariate=l m n o p,
          history=h
          );


** Initialize time, distance, and history variables **;

data longReg;
   set longData;
   time = time_exposure;
   distance = time_exposure - time_covariate;
   history = h;
run;


** Filter data (keep observations with exposure time greater than or equal to covariate time) **;

data longReg2;
   set longReg;
   where time_exposure >= time_covariate;
run;
```

```
** Ensure time variable is numeric **;

data longReg3;
   set longReg2;
   time_num = input(time, 8.);
run;


** Sort data by covariate name **;

proc sort data=longReg3 out=longRegSort;
   by name_cov;
run;


** Make balance table using regression (PROC GLM) **;

ods output ParameterEstimates=glmEstimates; /*Store the parameter estimates*/
proc glm data=longRegSort;
   class a (REF=FIRST) history (REF=FIRST);
   model value_cov = a time_num distance history / SOLUTION;
   by name_cov;
run;
quit;



** Retain the parameter estimate for "a1" only, and rename the "estimate" variable **;

data glmEst2(keep=name_cov estimate rename=(estimate=D));
   set glmEstimates;
   where parameter eq "a         1";
run;
```

The parameter estimates for the non-referent exposure (e.g. "a 1") obtained for each covariate value (e.g. "l m n o p") in the regression model can then be compared to the corresponding "D" values given by the `%balance()` macro:

```
** Compare that to a direct calculation and standardization **;

%balance(input=longData,
         output=table_std,
         diagnostic=1,
         approach=none,
         censoring=no,
         scope=average,
         average_over=distance,
         times_exposure=0 1 2,
         times_covariate=0 1 2,
         sort_order=alphabetical,
         exposure=a,
         history=h,
         );


** Retain the "name_cov" and appropriate metric variable **;

    data table_std(keep=name_cov D);
       set work.balance_table;
    run;


** Compare the PROC GLM results to the results from the balance macro **;

ods pdf file="P:\myprojects\myoutputs\glm_compare_results.pdf";

   proc compare base=table_std compare=glmEst2 listall;
       title 'Comparing PROC GLM to Balance Macro';
    run;

ods pdf close;
```

Alternatively, PROC MIXED can be used to perform the regression analysis:

```
** Make balance table using regression (PROC MIXED) **;

ods output LComponents=mixedEstimates;  /* Store the parameter estimates */
proc mixed data=longRegSort;
   class a(REF=FIRST) history(REF=FIRST);
   model value_cov = a time_num distance history / SOLUTION LCOMPONENTS;
   by name_cov;
run;
quit;


** Retain the parameter estimate for "a" only, and rename the "estimate" variable accordingly **;

data mixedEst2(keep=name_cov estimate rename=(estimate=D));
   set mixedEstimates;
   where effect eq "a";
run;



** Compare that to a direct calculation and standardization **;

%balance(input=longData,
         output=table_std,
         diagnostic=1,
         approach=none,
         censoring=no,
         scope=average,
         average_over=distance,
         times_exposure=0 1 2,
         times_covariate=0 1 2,
         sort_order=alphabetical,
         exposure=a,
         history=h,
         );


** Retain the "name_cov" and appropriate metric variable **;

    data table_std(keep=name_cov D);
       set work.balance_table;
    run;


** Compare the PROC MIXED results to the results from the balance macro **;

ods pdf file="P:\myprojects\myoutputs\mixed_compare_results.pdf";

   proc compare base=table_std compare=mixedEst2 listall;
       title 'Comparing PROC MIXED to Balance Macro';
    run;

ods pdf close;
```