Cluster_k: SAS Macro for Clustering Districts and Schools for Impact Studies in Education

Documentation

November 18, 2022

Azim Shivji
The University of Chicago
Harris School of Public Policy
ashivji@uchicago.edu

Daniel Litwok
Abt Associates
dan_litwok@abtassoc.com

Robert B. Olsen
George Washington Institute of Public Policy
The George Washington University
robolsen@gwu.egu

ACKNOWLEDGEMENTS

The research reported here is supported by the Institute of Education Sciences, US. Department of Education, through Grant R305D190020 to Westat.

SUGGESTED CITATIONS

Shivji, A., Litwok, D., & Olsen, R. (2022). Cluster_k: SAS Macro for Clustering Districts and Schools for Impact Studies in Education. https://osf.io/fehjc/.

LICENSE

Overview

This guide includes documentation for a SAS macro developed to support impact evaluations in the selection of a sample of sites that represent a study's target population. To this end, researchers may want to stratify the population based on potential impact moderators (e.g., see Tipton & Olsen, 2022).

The macro, "cluster_k," helps researchers define strata for sampling using k-means or k-medians clustering (Tipton, 2013). While designed to define sampling strata for selecting sites, the "cluster_k" macro can also be used for other clustering applications. Built around SAS's PROC FASTCLUS, this macro allows users to specify any combination of interval, binary, and nominal variables for clustering. Users can indicate a preferred centroid and distance measure or allow the macro to select these measures based on the characteristics of the clustering variables. Users can also select from a list of scaling options for the clustering variables. The macro supports user-defined weights indicating the relative importance of the clustering variables.

The macro is also designed to make the clustering procedure more convenient for users. For instance, if users are uncertain about their choice of *k*, users may specify multiple *k* values in a single macro call and use the diagnostic output produced by the macro to help select an optimal *k*. The macro also combines variable scaling and weighting together with clustering in a single macro call. And the macro accommodates datasets with nested structures—for example, users may cluster at either the district or school level using a single dataset. The macro also produces a convenient output data file with pre-scaled means and medians that is useful for diagnostic or reporting purposes.

This macro draws from the SAS code used in simulations to test the performance of stratified random and balanced sampling for impact studies (Litwok et al., 2022). The guide includes an overview that explains the goal of the macro, syntax for using the macro, programmatic details and notes, descriptions of macro parameters and user options, output, and a worked example.

The worked example uses the same data as Litwok et al. (2022), based on the Common Core of Data. The data are available for download at https://osf.io/fehjc/. This document's appendix includes a codebook for the data.

Cluster_k Macro

Introduction

This macro performs k-means or k-medians clustering to partition data into groups.

The macro allows users to cluster data based on user-specified clustering variables that can have different characteristics (interval, binary, nominal, or combinations of interval and nominal/binary). Users can indicate their preferred centroid measure (mean or median) and distance measure (Euclidean, Manhattan, or Gower). The literature recommends scaling cluster variables before conducting clustering; the macro allows users to select a scaling option (e.g., standard deviation, range, Euclidean length). The macro supports user-defined weights indicating the relative importance of the clustering variables.

If users identify multiple values for *k*, the macro will cluster in multiple ways and provide the user with diagnostics that can be used to determine the optimal value of *k*.

Syntax

```
%cluster_k(
   input_data =,
   unit_id =,
   cluster_vars_interval =,
   cluster_vars_binary =,
   cluster_vars_nominal =,
   k_clusters_list =,
   centroid_measure =,
   distance_measure =,
   scaling =,
   cluster_var_weights_data =,
   seed =,
   maxiter =,
   converge =,
   output_data =,
   output_var_cluster =,
   output_var_distance =,
   output_centroids =,
   output_summary =
);
```

Mandatory Parameters

The following parameters must be specified in each macro call. They are the minimum parameters required for the macro to run:

- input_data
- unit_id
- at least one of the three cluster_vars_[type] parameters:
    - cluster_vars_interval
    - cluster_vars_binary

    - cluster_vars_nominal
- k_clusters_list

Optional Parameters

The user may choose to specify the following parameters or let the macro choose default values for them:

- centroid_measure
- distance_measure
- scaling
- cluster_var_weights_data
- seed
- maxiter
- converge
- output_data
- output_var_cluster
- output_var_distance
- output_centroids
- output_summary

Macro parameters and user options

This section describes each of the macro parameters and user options in detail. For each parameter, the guide indicates whether it is mandatory or optional to specify, a description of the parameter, acceptable values, default values, and corresponding notes.

input_data

- *Mandatory/Optional to Specify:* **Mandatory**

- *Description:* The name of the dataset containing the data you wish to cluster.

- *Acceptable Values:* Any valid one- or two-level dataset name (note that two-level names including the libref are compatible).
    - Examples:
        - `input_data = mydata`
        - `input_data = mylib.mydata`

- *Default Value:* None

- *Notes:* If there is more than one observation per unique `unit_id` in `input_data`, then the macro will do the following, for the user's convenience:
    - Collapse the data before clustering, so that there is only one observation per unique `unit_id`. (Before doing this, the macro will also check whether the specified cluster variables are consistent among observations with the same `unit_id`.)
    - Attach the final results (the cluster assignments variable and the distance variable) to the original data structure (with multiple observations per unique `unit_id`).

  This may be useful for users who have a nested data structure. See the Examples section for an example with schools nested within districts.

unit_id

- *Mandatory/Optional to Specify:* **Mandatory**

  Description: The name of a variable in your `input_data` that uniquely identifies the units you wish to cluster (e.g., a district ID if you want to cluster districts, a school ID if you want to cluster schools, etc.)

- *Acceptable Values:* Any valid variable name in `input_data` that uniquely identifies the units you wish to cluster

- *Default Value:* None

- *Notes:* N/A

cluster_vars_interval

- *Mandatory/Optional to Specify:* **Optional** (but the three `cluster_vars_[type]` parameters cannot all be blank; you must specify at least one variable to cluster in at least one of the three `cluster_vars_[type]` parameters)

- *Description:* The names of any interval variables you want to use in the clustering. This is one of the three parameters where you specify cluster variables (also known as features in the clustering literature): `cluster_vars_interval`, `cluster_vars_binary`, and `cluster_vars_nominal`.
  Interval variables must be numeric variables.

- *Acceptable Values:* A space-delimited list of variable names. Abbreviated variable lists (such as name prefix lists or numbered range lists) are not supported. You may get either errors or incorrect results if you use abbreviated variable lists.
  For instance, the following space-delimited variable list is acceptable:
    - `cluster_vars_interval = var1 var2 var3`

Abbreviated variable lists such as the following are NOT acceptable:
  - `cluster_vars_interval = var:`
  - `cluster_vars_interval = var1-var3`

Variables identified as cluster variables cannot be missing for any observations.

- *Default Value:* None

- *Notes:* N/A

cluster_vars_binary

- *Mandatory/Optional to Specify:* **Optional** (but the three `cluster_vars_[type]` parameters cannot all be blank; you must specify at least one variable to cluster in at least one of the three `cluster_vars_[type]` parameters)

- *Description:* The names of any binary variables you want to use in the clustering. This is one of the three parameters where you specify cluster variables (also known as features in the clustering literature): `cluster_vars_interval`, `cluster_vars_binary`, and `cluster_vars_nominal`.
  Binary variables must be numeric variables, and the only acceptable non-missing values are 0 and 1.

- *Acceptable Values:* A space-delimited list of variable names. Abbreviated variable lists (such as name prefix lists or numbered range lists) are not supported. You may get either errors or incorrect results if you use abbreviated variable lists.
  For instance, the following space-delimited variable list is acceptable:
  - `cluster_vars_binary = var1 var2 var3`

  Abbreviated variable lists such as the following are NOT acceptable:
  - `cluster_vars_binary = var:`
  - `cluster_vars_binary = var1-var3`

  Variables identified as cluster variables cannot be missing for any observations.

- *Default Value:* None

- *Notes:* N/A

cluster_vars_nominal

- *Mandatory/Optional to Specify:* **Optional** (but the three `cluster_vars_[type]` parameters cannot all be blank; you must specify at least one variable to cluster in at least one of the three `cluster_vars_[type]` parameters)

- *Description:* The names of any nominal variables you want to use in the clustering. This is one of the three parameters where you specify cluster variables (also known as features in the clustering literature):

`cluster_vars_interval`, `cluster_vars_binary`, and `cluster_vars_nominal`.
Nominal variables may be either numeric or character variables.

- *Acceptable Values:* A space-delimited list of variable names. Abbreviated variable lists (such as name prefix lists or numbered range lists) are not supported. You may get either errors or incorrect results if you use abbreviated variable lists.
  For instance, the following space-delimited variable list is acceptable:
  - `cluster_vars_nominal = var1 var2 var3`

  Abbreviated variable lists such as the following are NOT acceptable:
  - `cluster_vars_nominal = var:`
  - `cluster_vars_nominal = var1-var3`

  Variables identified as cluster variables cannot be missing for any observations.

- *Default Value:* None

- *Notes:* N/A

k_clusters_list

- *Mandatory/Optional to Specify:* **Mandatory**

- *Description:* A number or list of numbers specifying the maximum number of clusters (i.e., the "k" value in k-means or k-medians clustering). (This value corresponds to the `MAXCLUSTERS` option in `PROC FASTCLUS`, the clustering procedure used by this macro.)

  If this parameter is a single value (e.g., `k_clusters_list=4`), then the clustering procedure will be run once, with this value as the maximum number of clusters.

  However, you can also supply a list of k values to the `k_clusters_list` parameter, in which case the macro will run the clustering procedure multiple times, once for each k value. You can use this feature to implement cluster analysis for different values of k, output diagnostics for each of those values, and choose your preferred value.

- *Acceptable Values:* There are several different ways to specify values for `k_clusters_list`.
  - If you only want to specify a single k value, such as 4:
    ```
    k_clusters_list = 4
    ```
  - If you want the macro to run the clustering procedure for multiple values of k, you can specify a space-delimited list of k values:
    ```
    k_clusters_list = 4 5 6
    ```
  - Alternatively, you can use a colon (:) to specify a range of k values. 4:6 means the numbers 4, 5, and 6. E.g.:
    ```
    k_clusters_list = 4:6
    ```

- And you can use both ranges and space-delimited lists together:
  ```
  k_clusters_list = 2 5 7:10 15 20:22
  ```
  The above example would tell the macro to run the clustering for each of the following k values: 2, 5, 7, 8, 9, 10, 15, 20, 21, and 22.

- *Default Value:* None

- *Notes:* The k value represents the maximum number of clusters permitted. The actual clusters that the procedure finds may be fewer than the maximum specified (if additional clusters would not reduce within-cluster variance).

centroid_measure

- *Mandatory/Optional to Specify:* **Optional** because there is a default value

- *Description:* Specifies whether the macro will perform k-means or k-medians clustering. More precisely, this parameter specifies how the cluster centroids (or cluster centers) are computed. See Notes for more details.

- *Acceptable Values:* `mean` or `median`
  The values are case-insensitive.

- *Default Value:* The default values for the `centroid_measure` and `distance_measure` macro parameters all vary based on what you specify in the macro. See the documentation for the `distance_measure` for a table of default values.

- *Notes:* The clustering procedure begins by randomly choosing a set of initial cluster centroids from the data. The procedure assigns each observation to the cluster of the nearest centroid. After each observation has been assigned to a cluster, the procedure recomputes the centroids. The assignment and centroid re-computation steps repeat until convergence or until the maximum number of iterations is reached. The `centroid_measure` parameter determines whether the procedure will compute the centroids using the mean or median of observations assigned to the cluster.

distance_measure

- *Mandatory/Optional to Specify:* **Optional** because there is a default value

- *Description:* Specifies how the distance of an observation from a cluster centroid is measured.

- *Acceptable Values:*

Value[1]	**Description**
`euclidean`	• Euclidean distance (also known as the L2 norm) • Supported for k-means clustering of interval data.

	• When distance_measure=euclidean, the clustering procedure will attempt "to minimize the root mean squared difference between the data and the corresponding cluster means" (as stated in the SAS documentation for PROC FASTCLUS).
manhattan	• Manhattan distance (also known as the L1 norm, or the absolute-value distance) • Supported for k-medians clustering of interval data. • When distance_measure=manhattan, the clustering procedure will attempt "to minimize the mean absolute difference between the data and the corresponding cluster medians" (as stated in the SAS documentation for PROC FASTCLUS).
gower	• Gower's dissimilarity coefficient • Supported for k-medians clustering of mixed data (including interval and binary/nominal data). • PROC FASTCLUS (the SAS clustering procedure this macro uses) does not natively support the Gower measure. The macro operationalizes the Gower distance by applying the Manhattan distance to transformed and rescaled versions of the cluster variables. The transformation and rescaling involve the following steps: 1) Nominal cluster variables are transformed into a series of binary dummy variables. A binary variable is constructed for each level of the nominal variable (e.g., a nominal variable for Census region would be transformed into four binary variables for the four regions--West, Midwest, Northeast, and South). 2) All cluster variables (including the binaries constructed from nominal variables) are rescaled, as described in the documentation for the macro's scaling parameter (see the description of the scaling when scaling=gower). Once the cluster variables are transformed and rescaled, the macro can simply calculate the Gower distance between an observation and a cluster centroid using the Manhattan distance in PROC FASTCLUS.

[1] Value is case-insensitive.

- *Default Value:* The default values for the `centroid_measure` and `distance_measure` macro parameters vary based on what you specify in the macro.

  If you only specify the type of data you want to cluster (in the `cluster_vars_[type]` parameters), then these are the default values used by the macro:

User-Specified Data Type (from `cluster_vars_[type]`)	**Default `centroid_measure`**	**Default `distance_measure`**
`interval`	`mean`	`euclidean`
`binary/nominal`	`median`	`gower`
`mixed`	`median`	`gower`

  Defaults will vary if you specify additional information. If you specify interval-only data and `centroid_measure = median`, then the default `distance_measure` will be `manhattan`. Likewise, if you specify interval-only data and `distance_measure = manhattan`, then the default `centroid_measure` will be `median`.

  The macro documents the final specifications (including any default values assigned) in two places:
  1) the log, right before `PROC FASTCLUS` runs
  2) the `output_summary` dataset (see the `output_summary` parameter)

- *Notes:* Only specific combinations of the `centroid_measure`, `distance_measure`, and data type (as specified in the `cluster_vars_[type]` parameters) are supported by the macro. The table below lists all compatible combinations of these parameters:

`Centroid_measure` **macro parameter**	`Distance_measure` **macro parameter**	**Supported data types**
`mean`	`euclidean`	`interval`
`median`	`manhattan`	`interval`
`median`	`gower`	`interval`
`median`	`gower`	`binary/nominal`
`median`	`gower`	`mixed (interval and binary/nominal)`

  Additionally, the following combination is technically supported (meaning the macro will not return an error if you specify this combination), but is not recommended:

`Centroid_measure` **macro parameter**	`Distance_measure` **macro parameter**	**Supported data types**
`mean`	`euclidean`	`mixed`

scaling

- *Mandatory/Optional to Specify:* **Optional** because there is a default value

- *Description:* Specifies whether and how the cluster variables should be rescaled before performing the clustering.

- *Acceptable Values:*

Value[1]	**Location**	**Scale**	**Formula (in PROC SQL pseudo-code)**
`std`	Mean	Standard deviation	$\frac{x - \text{mean}(x)}{\text{std}(x)}$
`range`	Minimum	Range	$\frac{x - \min(x)}{\text{range}(x)}$
`gower`	Minimum	Range x v	$\frac{x - \min(x)}{\text{range}(x) \times v}$
`euclen`	0	Euclidean length	$\frac{x}{\sqrt{\sum x^2}}$
`none`	N/A	N/A	N/A

[1] Value is case-insensitive.

For the Gower measure:
(1) v = the sum of the cluster variable weights (or simply the number of cluster variables, if there are no weights--see the documentation for the `cluster_var_weights_data` macro parameter)
(2) Gower here refers not to the distance measure, itself, but rather the scaling done in support of that measure. This scaling allows the macro to compute the Gower distance simply as the absolute value of the difference between an observation in the data and a cluster centroid.

- *Default Value:* If the `distance_measure` equals `gower`, then the macro uses `gower` as the default scaling. This is not merely a default; it also overrides any value the user specified (i.e., if the user specifies `distance_measure=gower`, then the user specification for scaling will be ignored, and the macro will use the default `gower` scaling). For all other cases (where the `distance_measure` is not `gower`), there is no default value for the scaling parameter.

- *Notes:* While the macro supports several common scaling options, you may wish to use a different or customized scaling. In that case, we recommend you rescale your data (with SAS's `PROC STDIZE` or other methods) before running the macro. Then, you can feed the rescaled data into the macro and specify `scaling=none`, so that no further scaling will be done by the macro.

Also note that the rescaled versions of the cluster variables, produced by this scaling parameter, will be discarded after clustering. The `output_data` returned by the macro will contain the original versions of these variables, not the rescaled versions.

cluster_var_weights_data

- *Mandatory/Optional to Specify:* **Optional** because there is a default value

- *Description:* The name of a dataset containing weights for the cluster variables.

  Note that these are *variable* weights, not *observation* weights. The dataset should have *only one* observation, and the variables in the dataset should have the values of the cluster variable weights. See the Examples section below for sample datasets.

- *Acceptable Values:* Any valid dataset name that contains weights for the cluster variables.

- *Default Value:* Each cluster variable is equally weighted, with a weight of 1. Additionally, if the user does specify a dataset for the `cluster_var_weights_data` parameter, then any cluster variables that are named in the `cluster_vars_[type]` macro parameters but do not appear in the `cluster_var_weights_data` dataset will be assigned a weight of 1.

- *Notes:* The scaling macro parameter and this `cluster_var_weights_data` parameter are tied closely together. The scaling is a form of weighting, but it equalizes the importance of all cluster variables, regardless of their original measurement units. However, a researcher may have a prior basis to distinguish the relative weights of different variables.

  The researcher may wish to adjust the relative importance of variables in the clustering, to for instance, inflate one variable's importance and deflate another's. Tipton (2013) notes that these weights can take advantage of information the researcher has on "the importance of particular covariates in explaining potential treatment effect heterogeneity."

  If you do want to apply different weights to different cluster variables, then you can supply a dataset to this macro parameter (the `cluster_var_weights_data` parameter). Note that the scaling parameter and the `cluster_var_weights_data` parameter are not mutually exclusive. You may use both if you wish. In fact, it may be useful to use both--to first adjust the cluster variables to a common scale (since the macro always performs the scaling before the weighting) and then distinguish their importance according to the weights you supply.

  It is important to understand that the macro will apply your given weights to the

variables, themselves (after performing any scaling requested in the scaling parameter), not to the distances. Therefore, the macro tailors your weights according to the `distance_measure` you are using.

For instance, if you want to cluster two variables and double the weight of one variable, x1, and halve the weight of the other variable, x2, then:

(1) If you are using the Manhattan or Gower distances, the macro uses the weights you specified, as is:

Variable	User-Specified Weight	Weight the Macro Uses
x1	2	2
x2	0.5	0.5

(2) However, if you are using the Euclidean distance measure, and you want a weighted Euclidean distance for variables x1 and x2, the equation for that distance is:

$$d(i,j) = \sqrt{w_1(x_{1i} - x_{1j})^2 + w_2(x_{2i} - x_{2j})^2}$$

where

$d(i,j)$ is the distance between observations i and j
x1 and x2 are two cluster variables
w1 is 2, the desired weight for variable x1
w2 is 0.5, the desired weight for variable x2

In this equation, the weights are applied to the squared distances. That equation is equivalent with directly weighting x1 and x2 by the square root of w1 and w2, respectively, instead of weighting the squared distances, as in:

$$d(i,j) = \sqrt{(\sqrt{w_1} \times x_{1i} - \sqrt{w_1} \times x_{1j})^2 + (\sqrt{w_2} \times x_{2i} - \sqrt{w_2} \times x_{2j})^2}$$

Since this macro applies the weights to the variables (x1 and x2) directly, and not to the distances, the macro uses this second, equivalent form of the equation and takes the square roots of the weights you specified for variables x1 and x2:

Variable	User-Specified Weight	Weight the Macro Uses
x1	2	$\sqrt{2}$
x2	0.5	$\sqrt{0.5}$

Additional notes:

a) You do not need to specify weights for all cluster variables in the `cluster_var_weights_data` dataset. Any cluster variables that you do not specify in this dataset will be assigned a weight of 1.

b) It doesn't matter if the order of the variables in the `cluster_var_weights_data` dataset does not match the order of the variables, either in your `input_data` or in the `cluster_vars_[type]` macro parameters. But the variable names must match. If you specified `cluster_vars_interval = x1`, but your `cluster_var_weights_data` dataset misnamed the variable as `x_1`, then the macro will not be able to match the weight with the appropriate variable. An error will trigger if the macro finds a variable in the `cluster_var_weights_data` dataset that does not appear in the `cluster_vars_[type]` macro parameters.

- *Examples:*
  - Example #1:

```
* create the dataset with the cluster variable weights;
data weights;
   x1 = 2;
   x2 = 0.5;
   x3 = 1;
run;

* run the clustering macro;
%cluster_k(
   input_data = mydata,
   unit_id = district_id,
   cluster_vars_interval = x1 x3,
   cluster_vars_binary = x2,
   cluster_vars_nominal = x4,
   k_clusters_list = 6,
   centroid_measure = MEAN,
   distance_measure = GOWER,
   cluster_var_weights_data = weights
);
* ^ note that the macro will assign a weight of 1 to cluster
variable x4 because it will not be able to find that variable in
the weights dataset;
* also note that the scaling parameter was not specified here
because, since distance_measure=GOWER, the GOWER scaling is
implied;
```

  - Example #2:

```
In describing the weighted Euclidean distance, Tipton (2013)
writes, "...if there is no information regarding which covariate
is a more important or better predictor of treatment effect
heterogeneity, then the obvious solution is to use inverse-
variance weights..."
```

As we showed above, weighting the squared distance for a variable by weight w1, in the weighted Euclidean distance equation, is equivalent with directly weighting that variable's values by the square root of w1. Therefore, to apply inverse-variance weights, we can simply standardize the variable, dividing it by its standard deviation (since the inverse of the standard deviation is the square root of the inverse of the variance).

The best way to implement that approach in this macro is to use the scaling parameter, as in:

```
%cluster_k(
   input_data = mydata,
   unit_id = district_id,
   cluster_vars_interval = x1 x2 x3,
   cluster_vars_binary = ,
   cluster_vars_nominal = ,
   k_clusters_list = 4,
   centroid_measure = MEAN,
   distance_measure = EUCLIDEAN,
   scaling = STD,
   cluster_var_weights_data = ,
);
```

Note that, in addition to dividing by the standard deviation, when you specify scaling=STD, the macro will also center the cluster variables around their means.

However, if you prefer, you can use the cluster_var_weights_data parameter to apply inverse-variance weights, instead of using the scaling parameter. The example below shows how (but note that this example below would not center the cluster variables around their means):

```
* create the dataset with the cluster variable weights;
proc sql;
   create table weights as
      select   1/var(x1) as x1,
               1/var(x2) as x2,
               1/var(x3) as x3
      from  mydata;
quit;

* run the clustering macro;
%cluster_k(
   input_data = mydata,
   unit_id = district_id,
   cluster_vars_interval = x1 x2 x3,
   cluster_vars_binary = ,
   cluster_vars_nominal = ,
```

```
    k_clusters_list = 4,
    centroid_measure = MEAN,
    distance_measure = EUCLIDEAN,
    scaling = NONE,
    cluster_var_weights_data = weights
);
```

seed

- *Mandatory/Optional to Specify:* **Optional** because there is a default value
- *Description:* A positive integer specifying the seed the clustering procedure will use.
- *Acceptable Values:* Positive integers
- *Default Value:* Automatically generated in SAS by running: `call streaminit(0)`

maxiter

- *Mandatory/Optional to Specify:* **Optional** because there is a default value
- *Description:* This parameter corresponds to the `MAXITER` option in `PROC FASTCLUS` and (as stated in the SAS documentation) "specifies the maximum number of iterations for recomputing cluster [centroids]."
- *Acceptable Values:* Positive integers
- *Default Value:* Depends on the value of the `distance_measure` parameter:

Distance Measure	Default maxiter
`euclidean`	10
`manhattan`	20
`gower`	20

- *Notes:* We chose to use the same default values as `PROC FASTCLUS` uses for the `MAXITER` option. In practice, you may find that the default is relatively low, and you may see this message after `PROC FASTCLUS` runs: "WARNING: Iteration limit reached without convergence." In that case, you may consider increasing the `maxiter` parameter value and/or increasing the `converge` parameter value.

converge

- *Mandatory/Optional to Specify:* **Optional** because there is a default value
- *Description:* The convergence criterion for the clustering algorithm. This parameter corresponds to the `CONVERGE` option in `PROC FASTCLUS`.

- *Acceptable Values:* Any non-negative numeric value

- *Default Value:* 0.0001

output_data

- *Mandatory/Optional to Specify:* **Optional** because there is a default value

- *Description:* The name of the output dataset that will be produced by the macro, and which will attach to your original data (supplied in the `input_data` parameter) the cluster assignments and the distances from the cluster centroids.

- *Acceptable Values:* Any valid one- or two-level dataset name (note that two-level names including the libref are compatible). Examples:
  - `output_data = outdata`
  - `output_data = mylib.outdata`

- *Default Value:* `_out_data`

- *Notes:* The `output_data` will maintain the structure of the `input_data`. If the `input_data` had multiple observations per unique `unit_id`, then the `output_data` will as well. See the Notes for the `input_data` parameter.

output_var_cluster

- *Mandatory/Optional to Specify:* **Optional** because there is a default value

- *Description:* The name of the variable in the output dataset containing the cluster assignments.

- *Default Value:* `_cluster`

output_var_distance

- *Mandatory/Optional to Specify:* **Optional** because there is a default value

- *Description:* The name of the variable in the output dataset containing the distance of each observation from its cluster centroid.

- *Default Value:* `_distance`

output_centroids

- *Mandatory/Optional to Specify:* **Optional** because there is a default value

- *Description:* The name of the output dataset that will be produced by the macro, and which will report the mean and median of each cluster variable for each cluster (and each value of k, if multiple values were specified in the `k_clusters_list` parameter).

Note that the means and medians are unweighted, and they are reported at the unique `unit_id` level. If your `input_data` contained multiple observations per unique `unit_id`, the means and medians will be computed from a deduplicated dataset, with only one observation per unique `unit_id`.

- *Acceptable Values:* Any valid one- or two-level dataset name (note that two-level names including the libref are compatible). Examples:
  - `output_centroids = outcentroids`
  - `output_centroids = mylib.outcentroids`

- *Default Value:* `_out_centroids`

output_summary

- *Mandatory/Optional to Specify:* **Optional** because there is a default value

- *Description:* The name of the output dataset that will be produced by the macro, and which will collect any summary statistics reported by `PROC FASTCLUS`, in the procedure's `OUTSTAT` option. Which statistics are reported will vary based on whether the `centroid_measure` equals `mean` or `median`. The pseudo-F statistic, for instance, is only reported when the `centroid_measure` equals `mean`. If you specified multiple values of k in the `k_clusters_list` parameter, the `output_summary` dataset will collect the summary statistics for each k value (i.e., for each clustering run) and stack them together.

  This output dataset will also include variables documenting the final specifications of the clustering, including any default values the macro assigned. These variables all begin with the prefix `_xyz_spec_` and report specifications such as the centroid measure, distance measure, scaling, and seed.

- *Acceptable Values:* Any valid one- or two-level dataset name (note that two-level names including the libref are compatible). Examples:
  - `output_summary = outsummary`
  - `output_summary = mylib.outsummary`

- *Default Value:* `_out_summary`

- *Notes:* Summary statistics on the cluster variables reported in this dataset are based on the adjusted versions of those variables. If applicable, these would be the versions of the variables that were rescaled (according to the scaling parameter), transformed (into binaries if the variable was nominal), and weighted (according to your specifications in the `cluster_var_weights_data`). If you want to report on the mean/median of the clustered variables in each cluster, in their original scale, we recommend using the mean/median reported in the `output_centroids` dataset, rather than the statistics reported in this `output_summary` dataset.

Output

The macro outputs a series of datasets as output. These datasets include:

(i) The original input dataset with additional variables that indicate clusters and distance measures calculated using either user selections or default options.

(ii) A dataset that reports the mean and median of each cluster variable for each cluster (and each value of k, if multiple values were specified in the `k_clusters_list` parameter).

(iii) A dataset that reports any summary statistics reported by `PROC FASTCLUS`, in the procedure's `OUTSTAT` option.

Example

The example that follows demonstrates how to use the macro for creating clusters of districts and schools. The example uses the sample data “base_school” available for download at https://osf.io/fehjc/. “base_school” is a school-level dataset with a nested structure (schools are nested within districts). The example demonstrates how to create clusters of districts and of schools.

```
* PREP
----------------------;
* run macro definition program;
%include "[your file path to the program]\Site Selection Macros.sas" /
lrecl=32767;

* load the sample data from the macro GitHub repository and import into SAS;
let indir = [your file path to the data];
proc import datafile=
   "&indir.\base_school.csv"
   out=base_school
   dbms=csv replace;
run;

* This is a school-level dataset containing both district-level
and school-level variables. In this example, we will first cluster the
districts, then cluster the schools.;


* CLUSTER DISTRICTS
----------------------;
* If you want to apply variable weights, create a weights dataset.;

data weights_district;
   nschools_elig = 5;
   epp_d_imp = 0.25;
```

```
   region_d = 1;
run;

* run clustering macro for districts;

%cluster_k(
   input_data = base_school,
   unit_id = leaid,

   cluster_vars_interval = nschools_elig epp_d_imp,
   cluster_vars_binary =,
   cluster_vars_nominal = region_d,

   k_clusters_list = 2:4,
   centroid_measure = MEDIAN,
   distance_measure = GOWER,
   scaling = GOWER,
   cluster_var_weights_data = weights_district,

   seed = 43290,

   output_data = clust_district,
   output_var_cluster = _cluster_d,
   output_var_distance = _distance_d,
   output_centroids = cent_district,
   output_summary = summary_district
);

** Specifying scaling=GOWER is unnecessary if you already specified
distance_measure=GOWER, but we explicitly included it here for comparison
with the second cluster_k call below.;

** Also, you could have left each of the following three parameters blank
and have gotten the same results: centroid_measure, distance_measure, and
scaling. If you had left those blank (or not included them in the macro call
at all), the macro would have determined default values for those parameters
based on the type of data you are clustering. And since the macro call above
specifies mixed data (interval and nominal variables), the macro would have
chosen, as its defaults, centroid_measure=MEDIAN, distance_measure=GOWER, and
scaling=GOWER. (See the documentation for the distance_measure parameter for
more information about how the macro chooses defaults for centroid_measure
and distance_measure.);

* finalize the district clustering by choosing one of the k values and
discarding the rest;
** In this example, we will choose the results from k=4. We will rename the
k4 versions of the cluster and distance variables and discard the versions
from the other values of k.;

data post_district;
```

```
   set clust_district(rename=(_cluster_d_k4=cluster_district
   _distance_d_k4=distance_district));
   drop _cluster_d: _distance_d:;
run;


* CLUSTER SCHOOLS
-----------------------;
* If you do not want to apply variable weights, there is no need to
create a weights dataset.;

* run clustering macro for schools;
%cluster_k(
   input_data = post_district,
   unit_id = ncessch,

   cluster_vars_interval = enr_tot_imp sme_pct_frp_tc_imp,
   cluster_vars_binary =,
   cluster_vars_nominal =,

   k_clusters_list = 2:4,
   centroid_measure = MEAN,
   distance_measure = EUCLIDEAN,
   scaling = STD,
   cluster_var_weights_data =,

   seed = 54908,

   output_data = clust_school,
   output_var_cluster = _cluster_s,
   output_var_distance = _distance_s,
   output_centroids = cent_school,
   output_summary = summary_school,

   maxiter = 50
);

* In this example, we increased the maximum number of iterations (maxiter)
to 50. In practice, you may want to tweak with the maxiter and/or
converge parameters if the clustering procedure reaches the iteration
limit before convergence.;

* The output_summary dataset has valuable information you can use to
assess the clustering and the different values of k.;
** The pseudo-F statistic, which we view below, is only available when
the centroid_measure=MEAN.;

proc print data=summary_school noobs;
   where _type_="PSEUDO_F";
   var k over_all;
```

```
run;

* finalize the school clustering by choosing one of the k values and
discarding the rest;
** In this example, we will choose the results from k=2. We will rename the
k2 versions of the cluster and distance variables and discard the versions
from the other values of k.;

data post_school;
   set clust_school(rename=(_cluster_s_k2=cluster_school
   _distance_s_k2=distance_school));
   drop _cluster_s: _distance_s:;
run;
*/
```

Programmatic Details and Notes

- SAS Version: These macros were programmed and tested in SAS 9.4.
- Requirements: The cluster_k macro requires the SAS/STAT product.
- Notes:
    - Both of the macros in this program create various intermediate datasets (temporary working datasets that are used in the processing of the macros and are deleted after they are no longer needed). To reduce the likelihood that these datasets conflict with (and overwrite) existing datasets in the user's work library, the macros use SAS's "DATAn naming convention."

      With this feature, SAS defines a set of potential dataset names (DATA1, DATA2, etc., all the way to DATA9999) and sequentially searches for unused dataset names among that group, in the work library. When it finds an unused name, it will assign that to the intermediate dataset the macros create.

      This *significantly reduces the likelihood of* (rather than categorically prevents) conflicts because there is still a possibility (however remote) that the user may have existing datasets in their work library that cover all of the potential names that SAS could assign (from DATA1 to DATA9999). If all 9,999 potential dataset names are in use, SAS will start from the beginning of the list and assign the name DATA1 to the new dataset (and overwrite the existing DATA1 dataset). In that unlikely scenario, the user should beware that their data may be overwritten by the macro.

    - As mentioned above, the macros delete the intermediate datasets when they are done with them. However, if the macros terminate early because of errors, there may be leftover intermediate datasets that were not yet

deleted (they should be easy to identify because they all follow the DATAn naming convention, as in DATA1, DATA2, etc.). The user should feel free to delete those intermediate datasets, if they want to clean up their work library, but there is no need to do so. Whether those datasets remain in the work library will not interfere with the processing of the macros.

References

Litwok, D., Nichols, A., Shivji, A., & Olsen, R. (2022). Selecting districts and schools for impact studies in education: A simulation study of different strategies. *Journal of Research on Educational Effectiveness.* DOI: https://doi.org/10.1080/19345747.2022.2128952.

SAS Institute Inc. (2022). SAS/STAT User's Guide. Cary, NC: SAS Institute, Inc. Available at: https://documentation.sas.com/doc/en/pgmsascdc/v_033/statug/titlepage.htm?fromDefault=

Tipton, E. (2013). Stratified sampling using cluster analysis: A sample selection strategy for improved generalizations from experiments. *Evaluation Review*, 37(2), 109-139.

Tipton, E., & Olsen, R. B. (2022). *Enhancing the Generalizability of Impact Studies in Education*. (NCEE 2022-003). National Center for Education Evaluation and Regional Assistance, Institute of Education Sciences, U.S. Department of Education. Washington, DC.

Appendix

Along with the macro, the Github repository makes available a sample data set that can be used to test the performance of the macros. Applying the macro to these data are the basis for many of the examples that appear within this documentation. This appendix includes a codebook for that sample dataset. See Litwok et al. (2022) for detailed discussion of (i) defining the population included in this dataset; (ii) imputing missing values for certain variables; (iii) our approach to "PPS" sampling; and (iv) strata for schools and districts.

Sample data codebook

Variable Name	Variable Description
leaid	NCES district identifier
strat_d	District stratum
distance_d_y	Distance from district stratum group mean (district-level)
lottery_d_start	Starting number for district lottery ("PPS" sampling)
lottery_d_end	Ending number for district lottery ("PPS" sampling)
ncessch	NCES school identifier
enr_tot_d	Total enrollment (district-level)
region_d	Region (district-level)
nschools_elig	Total number of eligible schools (district-level)
enr_tot_imp	Total enrollment (school-level)
sme_pct_frp_tc_imp	% eligible for free/reduced-price lunch (school-level)
epp_d_imp	Expenditures per pupil (district-level)
other_d	Administrator leadership (district-level)
es_nschools_elig	Number of eligible schools (district-level; standardized)
es_epp_d_imp	Expenditures per pupil (district-level; standardized)
es_enr_tot_imp	Total enrollment (school-level; standardized)
es_sme_pct_frp_tc_imp	% eligible for free/reduced-price lunch (school-level; standardized)
es_other_d	Administrator leadership (district-level; standardized)
nschools	Total number of schools (district-level)
strat_s	School stratum
distance_s_y	Distance from school stratum group mean (school-level)
distance_s_y_overall	Distance from overall mean (school-level)