Chapter 7
The FACTMAC Procedure

Contents

Overview: FACTMAC Procedure

The FACTMAC procedure implements the factorization machine model in SAS Viya. The flexible factorization machine model has applications in predictive modeling and recommendation (Rendle 2012). Factorization machines generalize matrix factorization, among other techniques. You can use the FACTMAC procedure to read and write data in distributed form, and to perform factorization in parallel by making full use of multicore computers or distributed computing environments.

The FACTMAC procedure estimates factors for each of the nominal input variables you specify, in addition to estimating a global bias and a bias for each level of those nominal input variables. You also specify an interval target variable. The procedure computes the biases and factors by using the stochastic gradient

descent (SGD) algorithm, which minimizes the root mean square error (RMSE) criterion on the input data table that you provide. In this method, each iteration attempts to reduce the RMSE. The SGD algorithm proceeds until the maximum number of iterations is reached.

PROC FACTMAC stores the results of the factorization an output data table, which is produced by the OUTMODEL statement. This data table contains the factors in addition to the global bias and the biases for all the levels of the input variables, in addition to the factors. The corresponding level names are listed for ease of reference. The biases and factors are used for scoring.

PROC FACTMAC Features

PROC FACTMAC enables you to use parallel execution for factorization in a distributed computing environment or on a single-machine. The following list summarizes the basic features of PROC FACTMAC:

- is highly distributed and multithreaded
- learns a factorization machine model based on a parallel implementation of the SGD optimization algorithm

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mycas cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mycas CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 11 in Chapter 3, "Shared Concepts."

Getting Started: FACTMAC Procedure

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 11 and "Loading a SAS Data Set onto a CAS Server" on page 12 in Chapter 3, "Shared Concepts."

This example shows how to use the FACTMAC procedure to learn a factorization machine model from observations in a SAS data table. This example uses the cars data set in the Sashelp library and illustrates the prediction of gas mileage of cars based on make and model. The analysis uses four variables: car make, car model, car type, and a variable named mpg_city, which measures the car's fuel usage (in miles per gallon) for city driving. The remaining variables in the data table are not used.

You can load the cars data set into your CAS session by naming your CAS engine libref in the first statement of the following DATA step:

```
data mycas.cars;
   set sashelp.cars;
run;
```

These statements assume that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

The following statements run PROC FACTMAC and output the results to ODS tables:

```
proc factmac data=mycas.cars outmodel=mycas.factors maxiter=50
             nfactors=5  learnstep=0.002;
   input make model type /level=nominal;
   target mpg_city  /level=interval;
   output out=mycas.score_out copyvars=(make model type mpg_city);
run;

proc print data=mycas.factors(obs=48);
run;
```

The NFACTORS= option requests that the model estimate five factors, the LEARNSTEP= option sets the optimization learning step to 0.002, the MAXITER= option requests that the optimization stop after 50 iterations, and the OUTMODEL= option requests that the model parameters be written to the mycas.factors data table. The INPUT statement specifies that the make, model, and type variables are to be used as nominal inputs. The TARGET statement specifies that mpg_city is the target variable to be predicted. The OUTPUT statement requests that the predictions be written to the data table mycas.score_out and that the make, model, type and mpg_city variables be copied from the mycas.cars data table to the mycas.score_out data table.

Figure 7.1 shows the global bias, the bias values for each level, and the list of the factors for the first 48 observations.

Figure 7.1 Bias Values and Factors

Obs	Variable	Level	Bias	Factor1	Factor2	Factor3	Factor4	Factor5
1	_GLOBAL_		20.0607	0.00000	0.00000	0.00000	0.00000	0.00000
2	Make	Acura	-0.6322	-1.65189	-2.34213	0.09553	-4.06551	0.72563
3	Make	Audi	-1.5871	-0.23131	0.07828	-0.35298	-0.45827	-0.20166
4	Make	BMW	-1.3607	0.00777	0.05455	-0.19466	-0.07272	0.22275
5	Make	Buick	-1.1719	2.96493	0.10815	3.14048	-4.74293	-0.28692
6	Make	Cadillac	-3.5607	-1.07033	-0.99187	1.49699	0.58245	0.95669
7	Make	Chevrolet	-0.3941	0.01216	0.00688	0.05321	0.00923	-0.03507
8	Make	Chrysler	-0.1941	0.56168	-0.90681	-0.77899	-1.38498	1.48220
9	Make	Dodge	-0.6761	0.53603	0.05044	0.03614	0.79867	0.46522
10	Make	Ford	-0.7999	-0.29532	-0.16454	0.05845	-0.27550	0.10528
11	Make	GMC	-4.6857	1.03171	-1.97310	-0.91069	-4.54418	2.00952
12	Make	Honda	7.7628	0.84902	-0.54073	-0.37149	-0.87898	-0.31334
13	Make	Hummer	-10.0607	-3.26652	2.06044	-3.31811	-1.32709	-6.03068
14	Make	Hyundai	2.9393	0.30075	0.72757	-0.78057	0.01371	-0.13113
15	Make	Infiniti	-2.8107	-1.20270	2.23762	6.27459	-3.97131	0.51608
16	Make	Isuzu	-4.0607	6.31966	-4.61052	2.32465	-5.98882	-2.33495
17	Make	Jaguar	-2.5607	-2.18175	-1.36395	-1.72034	2.88573	1.22202
18	Make	Jeep	-2.7274	1.34136	-4.71543	5.68740	0.25748	-5.30474
19	Make	Kia	1.8483	-5.30755	3.53864	-4.75619	5.24446	2.55098
20	Make	Land Rover	-6.0607	-5.98657	1.73501	3.16050	0.16469	-3.89031
21	Make	Lexus	-2.6062	-0.97068	1.83440	1.61661	0.77815	0.44383
22	Make	Lincoln	-3.2830	1.32684	0.16185	-0.68375	-0.66196	1.17094
23	Make	MINI	6.4393	-0.18316	1.88514	-0.79716	-2.62557	5.78783
24	Make	Mazda	1.3938	-0.91171	1.95718	-1.13241	0.70124	0.06483
25	Make	Mercedes-Benz	-2.7146	-0.16738	0.07799	-0.20517	-0.25945	-0.08443
26	Make	Mercury	-2.5052	0.93393	2.25607	0.17357	0.25786	1.27973
27	Make	Mitsubishi	0.8623	-0.24432	0.35635	0.16703	0.32479	-0.02810
28	Make	Nissan	-0.3549	-0.00132	-0.02561	0.00751	0.00230	-0.02652
29	Make	Oldsmobile	0.9393	0.21073	3.49856	5.92790	-6.28979	2.92394
30	Make	Pontiac	0.4847	-0.13061	-1.70502	-1.35086	1.14599	1.41535
31	Make	Porsche	-2.6322	-4.90259	4.56269	4.18555	0.66543	-4.09505
32	Make	Saab	0.3678	-0.18666	0.00714	0.16181	0.80439	0.61889
33	Make	Saturn	4.3143	1.90340	-3.38020	3.73003	0.31785	-0.34106
34	Make	Scion	11.4393	6.31009	-6.22994	6.26170	-1.74481	5.77012
35	Make	Subaru	0.2120	0.48494	-0.36550	-0.41698	-0.49696	1.03908
36	Make	Suzuki	2.0643	-1.72773	1.93579	-0.77099	-0.01766	-1.50695
37	Make	Toyota	4.3678	0.29650	-0.08574	-0.13681	0.20477	-0.54623
38	Make	Volkswagen	1.3393	-0.05236	0.70879	-1.49191	-1.33924	0.19614
39	Make	Volvo	-0.3107	0.91585	-0.20479	-0.64416	-0.23166	0.74650
40	Model	3.5 RL 4dr	-2.0607	4.29008	6.32127	6.32609	-6.32203	-4.13052
41	Model	3.5 RL w/Navigation 4dr	-2.0607	-2.53358	1.07033	6.32244	4.64185	0.88841
42	Model	300M 4dr	-2.0607	6.33258	6.32102	-1.46105	6.32544	6.32405
43	Model	300M Special Edition 4dr	-2.0607	-6.33347	4.20158	-6.32730	-0.68946	4.77917
44	Model	325Ci 2dr	-0.0607	0.75687	-6.32171	6.32257	-6.32459	6.32309
45	Model	325Ci convertible 2dr	-1.0607	6.33216	-4.54598	-0.49266	6.32496	6.32270
46	Model	325i 4dr	-0.0607	-0.11359	4.98979	2.21139	5.50177	6.32416
47	Model	325xi 4dr	-1.0607	-6.32539	1.49304	5.52169	-6.13325	6.32568
48	Model	325xi Sport	-1.0607	6.32095	-5.18676	-0.45827	-4.11622	-3.12254

Syntax: FACTMAC Procedure

The following statements are available in the FACTMAC procedure:

```
PROC FACTMAC < options > ;
    CODE FILE=filename ;
    DISPLAY < table-list > < / options > ;
    DISPLAYOUT table-spec-list < / options > ;
    ID variables ;
    INPUT variables < LEVEL=NOMINAL > ;
    OUTPUT OUT=CAS-libref.data-table < options > ;
    SAVESTATE RSTORE=CAS-libref.data-table ;
    TARGET variable < LEVEL=INTERVAL > ;
    AUTOTUNE < options > ;
```

The PROC FACTMAC statement, an INPUT statement, and the TARGET statement are required. You can specify multiple INPUT statements.

The following sections describe the PROC FACTMAC statement and then describe the other statements in alphabetical order.

PROC FACTMAC Statement

```
PROC FACTMAC < options > ;
```

The PROC FACTMAC statement invokes the procedure. Table 7.1 summarizes the *options* available in the PROC FACTMAC statement.

Table 7.1 PROC FACTMAC Statement Options

Option	Description
Input Data Table Options	
DATA=	Specifies the input data table
Factorization Options	
NFACTORS=	Specifies the number of factors to estimate for the model
MAXITER=	Specifies the maximum number of iterations
SEED=	Specifies the seed to be used for pseudorandom number generation
LEARNSTEP=	Specifies the learning step size for the SGD algorithm
NTHREADS=	Specifies the number of threads to use on each computation node
NONNEGATIVE	Requests nonnegative factorization

You can specify the following *options*:

DATA=*CAS-libref.data-table*
names the input data table for PROC FACTMAC to use. The default is the most recently created data table. *CAS-libref.data-table* is a two-level name, where

CAS-libref refers to a collection of information that is defined in the LIBNAME statement and includes the caslib, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *CAS-libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 112.

data-table specifies the name of the input data table.

LEARNSTEP= *number*
specifies the learning step size for the stochastic gradient descent (SGD) algorithm, where *number* is a positive real number. The learning step size controls the amount by which the factors are updated at each iteration.

By default, LEARNSTEP=0.001. This value can be tuned with the AUTOTUNE statement.

MAXITER=*number*
specifies the maximum number of iterations for the algorithm to perform, where *number* is an integer greater than or equal to 1. In each iteration of the SGD method, the factors are recomputed.

By default, MAXITER=1. This value can be tuned with the AUTOTUNE statement.

NFACTORS= *number*
specifies the number of factors to estimate for the model, where *number* is an integer greater than or equal to 1.

By default, NFACTORS=1. This value can be tuned with the AUTOTUNE statement.

NONNEGATIVE
performs nonnegative factorization, in which the estimated factors are greater than or equal to 0 and the estimated biases are 0.

By default, nonnegative factorization is not performed.

NOPRINT
suppresses ODS output.

NTHREADS=*number-of-threads*
specifies the number of threads to use for the computation, where *number-of-threads* is an integer from 1 to 64, inclusive. The default value is the maximum number of available threads per computer.

OUTMODEL=*CAS-libref.data-table*
specifies the output model data table to contain the computed factor parameters. *CAS-libref.data-table* is a two-level name, where *CAS-libref* refers to the caslib and session identifier, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 112.

SEED=*random-seed*

specifies an integer that is used to start the pseudorandom number generator. This option enables you to reproduce the same sample output, but only when NTHREADS=1. If you do not specify a seed or you specify a value less than or equal to 0, the seed is generated from reading the time of day from the computer's clock.

By default, SEED=0.

AUTOTUNE Statement

AUTOTUNE < *options* > **;**

The AUTOTUNE statement searches for the best combination of values of the NFACTORS=, LEARNSTEP=, and MAXITER= options in the PROC FACTMAC statement. You cannot specify both the OUTPUT and AUTOTUNE statements in the same run of PROC FACTMAC.

Table 7.2 summarizes the *options* that you can specify in the AUTOTUNE statement. For more information about all options except the TUNINGPARAMETERS= option, see the option's description in the section "AUTOTUNE Statement" on page 14 in Chapter 3, "Shared Concepts." The TUNINGPARAMETERS= option is described following Table 7.2.

Table 7.2 AUTOTUNE Statement Options

Option	Description
EVALHISTORY=	Specifies how to report the evaluation history of the tuner
FRACTION=	Specifies the fraction of observations to use for validation
KFOLD=	Specifies the number of folds for *k*-fold cross validation
MAXBAYES=	Specifies the maximum number of points in the kriging model
MAXEVALS=	Specifies the maximum number of evaluations
MAXITER=	Specifies the maximum number of iterations when SEARCHMETHOD=GA or SEARCHMETHOD=BAYESIAN
MAXTIME=	Specifies the maximum time for all iterations
MAXTRAINTIME=	Specifies the maximum time for a model train
NPARALLEL=	Specifies the number of parallel sessions
NSUBSESSIONWORKERS=	Specifies the number of workers in parallel sessions
OBJECTIVE=	Specifies the objective function
POPSIZE=	Specifies the population size when SEARCHMETHOD=GA or SEARCHMETHOD=BAYESIAN
SAMPLESIZE=	Specifies the sample size when SEARCHMETHOD=LHS or SEARCHMETHOD=RANDOM
SEARCHMETHOD=	Specifies the search method that the optimizer uses
TARGETEVENT=	Specifies the target event for ROC-based calculations
TRAINFRACTION=	Specifies the fraction of observations to use for training
TUNINGPARAMETERS=	Specifies the custom tuning parameters
USEPARAMETERS=	Specifies how to handle the TUNINGPARAMETERS= option

TUNINGPARAMETERS=(*suboption* **| . . . | <** *suboption* **>)**

TUNEPARMS=(*suboption* **| . . . | <** *suboption* **>)**

specifies which parameters to tune and which ranges to tune over. If USEPARAMETERS=STANDARD, this option is ignored.

You can specify one or more of the following *suboptions*:

NFACTORS (LB=*number* **UB=***number* **VALUES=***value-list* **INIT=***number* **EXCLUDE)**

specifies information about the number of factors to use for tuning the factorization machine model. For more information, see the NFACTORS= option in the PROC FACTMAC statement.

You can specify the following additional suboptions:

LB=*number*

specifies the minimum number of factors to consider during tuning. If you specify this suboption, you cannot specify the VALUES= suboption.

By default, LB=5.

UB=*number*

specifies the maximum number of factors to consider during tuning. If you specify this suboption, you cannot specify the VALUES= suboption.

By default, UB=30.

VALUES=*value-list*

specifies a list of values to consider for the number of factors during tuning, where *value-list* is a space-separated list of integer numbers greater than or equal to 1. If you specify this suboption, you cannot specify either the LB= or UB= suboption.

INIT=*number*

specifies the initial number of factors for the tuner to use.

By default, INIT=5.

EXCLUDE

excludes the number of factors from the tuning process. If you specify this suboption, any specified LB=, UB=, VALUES=, and INIT= suboptions are ignored.

LEARNSTEP (LB=*number* **UB=***number* **VALUES=***value-list* **INIT=***number* **EXCLUDE)**

specifies information about the learning step to use for tuning the factorization machine model. For more information, see the LEARNSTEP= option in the PROC FACTMAC statement.

You can specify the following additional suboptions:

LB=*number*

specifies the minimum learning step to consider during tuning. If you specify this suboption, you cannot specify the VALUES= suboption.

By default, LB=0.001.

UB=*number*
specifies the maximum learning step to consider during tuning. If you specify this suboption, you cannot specify the VALUES= suboption.

By default, UB=1.

VALUES=*value-list*
specifies a list of learning steps to consider during tuning, where *value-list* is a space-separated list of numbers greater than 0. If you specify this suboption, you cannot specify either the LB= or UB= suboption.

INIT=*number*
specifies the initial learning step for the tuner to use.

By default, INIT=0.001.

EXCLUDE
excludes the learning step from the tuning process. If you specify this suboption, any specified LB=, UB=, VALUES=, and INIT= suboptions are ignored.

MAXITER (LB=*number* **UB=***number* **VALUES=***value-list* **INIT=***number* **EXCLUDE)**
specifies information about the maximum number of iterations to use for tuning the factorization machine model. For more information, see the MAXITER= option in the PROC FACTMAC statement.

You can specify the following additional suboptions:

LB=*number*
specifies the minimum number of iterations to consider during tuning. If you specify this suboption, you cannot specify the VALUES= suboption.

By default, LB=10.

UB=*number*
specifies the maximum number of iterations to consider during tuning. If you specify this suboption, you cannot specify the VALUES= suboption.

By default, UB=200.

VALUES=*value-list*
specifies a list of numbers of trees to consider during tuning, where *value-list* is a space-separated list of positive integers. If you specify this suboption, you cannot specify either the LB= or UB= suboption.

INIT=*number*
specifies the initial number of iterations for the tuner to use.

By default, INIT=30.

EXCLUDE
excludes the number of iterations from the tuning process. If you specify this suboption, any specified LB=, UB=, VALUES=, and INIT= suboptions are ignored.

CODE Statement

CODE FILE=*filename* **;**

The CODE statement generates SAS DATA step code that mimics the computations that are performed. The generated SAS DATA step code can be used for scoring new observations. Only one CODE statement is processed. If you specify multiple CODE statements, only the first one is used.

You must specify the following option:

FILE=*filename*
specifies the name of the file to write the SAS score code to.

The CODE statement is optional. If you do not include a CODE statement, no score code is generated.

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 125. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create CAS output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of CAS output tables to create. Each entry in the list has either a *key=value* format or a *key* format:

key=value	specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the CAS output table name.
key	specifies *key* as the ODS table name and also as the CAS output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 125. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output CAS tables for all display tables. The name of the created output CAS table is the same as the corresponding display table name. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing CAS output table of the same name.

REPEATED
replicates all CAS output tables on all nodes.

ID Statement

ID *variables* **;**

The ID statement lists one or more variables that are to be copied from the input data table to the output data tables that are specified in the OUT= option in the OUTPUT statement and the RSTORE= option in the SAVESTATE statement.

INPUT Statement

INPUT *variables* < **LEVEL=INTERVAL | NOMINAL** > **;**

The INPUT statement specifies the names of the *variables* to be used in the factorization. It names one or more input variables that use common options. If you want to use different options for different variables, you can specify multiple INPUT statements.

You can include the following option in each INPUT statement:

LEVEL=INTERVAL | NOMINAL
specifies the level of measurement of the *variables*. You can specify the following values:

INTERVAL specifies that the level of measurement of the *variables* is interval.

NOMINAL specifies that the level of measurement of the *variables* is nominal.

By default, LEVEL=INTERVAL for numeric variables and LEVEL=NOMINAL for categorical variables.

You must specify at least two nominal input variables. You can also specify any number of interval input variables.

OUTPUT Statement

OUTPUT OUT=*CAS-libref.data-table* < *options* > **;**

The OUTPUT statement creates an output data table to contain the results of the procedure run. You cannot specify both the OUTPUT and AUTOTUNE statements in the same run of PROC FACTMAC.

You must specify the following *option*:

OUT=*CAS-libref.data-table*
names the output data table for PROC FACTMAC to use. You must specify this option before any other options. *CAS-libref.data-table* is a two-level name, where

CAS-libref refers to a collection of information that is defined in the LIBNAME statement and includes the caslib, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *CAS-libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 112.

data-table specifies the name of the output data table.

You can also specify the following *option*:

COPYVAR=*variable*

COPYVARS=(*variables***)**
lists one or more *variables* from the input data table to be transferred to the output data table.

SAVESTATE Statement

The SAVESTATE statement creates an analytic store for the model and saves it as a binary object in a data table. You can use the analytic store in the ASTORE procedure to score new data. For more information, see Chapter 4, "The ASTORE Procedure."

You must specify the following option:

RSTORE=*CAS-libref.data-table*
specifies a data table in which to save the analytic store for the model. *CAS-libref.data-table* is a two-level name, where *CAS-libref* refers to the caslib and session identifier, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 112.

TARGET Statement

TARGET *variable* < **LEVEL=INTERVAL** > **;**

The TARGET statement names the target variable whose values PROC FACTMAC predicts. The target must be interval and must be different from the variables in the INPUT statement. You can include the following option in the OUTPUT statement:

LEVEL=INTERVAL
specifies the level of measurement of the variables.

PROC FACTMAC currently accepts only interval target variables.

Details: FACTMAC Procedure

The factorization machines model is defined as

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{j=1}^{p} w_j x_j + \sum_{j=1}^{p} \sum_{j'=j+1}^{p} x_j x_{j'} \sum_{f=1}^{k} v_{jf} v_{j'f}$$

where $\mathbf{x} = (x_1, \ldots, x_p)$ is an observed p-dimensional input feature vector, $\hat{y}$ is the predicted target, w_0 is a global bias, w_j are per-feature biases, and v_{jf} denotes coordinate f of the vector $\mathbf{v}_j \in \mathbb{R}^k$. The overall

factor matrix $\mathbf{V} \in \mathbb{R}^{p \times k}$ is the concatenation of the row vectors $\mathbf{v}_j$ for $j = 1, \ldots, p$. The number of factors is k. PROC FACTMAC estimates the model parameters $w_0, w_1, \ldots, w_p$ and $\mathbf{V}$. The estimation is done by minimizing the root mean square error (RMSE), which is defined by

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

over the training set, subject to the max-norm regularization constraint

$$\|\mathbf{v}_j\|_\infty < B, j = 1, \ldots, p$$

The optimization uses a projected-gradient version of the stochastic gradient descent (SGD) algorithm. The constant B is automatically set, based on the range of the input and target variables.

The results of running PROC FACTMAC are reproducible only if you specify a value greater than 0 for the SEED= option and specify NTHREADS=1, because PROC FACTMAC uses a threaded SGD solver that purposefully uses shared memory without locks in each computation node. The variability between runs is nevertheless expected to be small. PROC FACTMAC can still use multiple machines for the analysis even when NTHREADS=1.

Displayed Output

The FACTMAC procedure displays various tables that are related to the factorization. The following sections describe the output tables in the order of their appearance when the related options are specified.

Model Information

The "Model Information" table displays basic information about the parameters that are used in the factorization analysis. This information includes the maximum number of iterations, learning step, number of factors, and seed value.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data table and used.

Iteration History

The "Iteration History" table displays the iteration history and approximate loss when the variables that are specified in the INPUT statement are interval.

The "displayed loss" is an approximation for computational efficiency reasons. The final exact loss is shown in the "Final Exact Loss" table.

Final Exact Loss

The "Final Exact Loss" table displays the actual, exact mean square error (MSE) and the root mean square error (RMSE) of the learned factorization machines model solution, which are computed on the training data.

Interval Variables

The "Interval Variables" table shows the mean and the standard deviation for the interval variables.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table created by the FACTMAC procedure has a name associated with it, and you must use this name to refer to the table when you use ODS statements. The names of each table and a short description of the contents are listed in Table 7.3.

Table 7.3 ODS Tables Produced by PROC FACTMAC

Table Name	Description	Statement	Option
DescStatsInt	Descriptive statistics for interval variables	PROC FACTMAC	Default
FinalLoss	Final exact loss	PROC FACTMAC	Default
ModelInfo	Model information	PROC FACTMAC	Default
NObs	Number of observations	PROC FACTMAC	Default
OptIterHistory	Iteration history	PROC FACTMAC	Default
OutputCASTables	See the section "OutputCasTables Table" on page 125	PROC FACTMAC	Default

Output Data Tables

The FACTMAC procedure creates a data table to which it writes the global biases and the factors. You specify the name of this data table in the OUTMODEL statement. Details about the data table are listed in Table 7.4.

Table 7.4 Output Data Table Produced by PROC FACTMAC

Data Table	Description
FACTORS	Lists the global bias, the name of each input variable, each level and the values of the estimated factors

Examples: FACTMAC Procedure

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 11 and "Loading a SAS Data Set onto a CAS Server" on page 12 in Chapter 3, "Shared Concepts."

Example 7.1: Running PROC FACTMAC with the MovieLens Data Set

This example draws on data that are derived from companies that provide movies for online viewing. A company wants to offer its customers recommendations of movies that they might like. These recommendations are based on ratings that are provided by users. The MovieLens data set was developed by the GroupLens project at the University of Minnesota and is available at `http://grouplens.org/datasets/movielens`. This example uses the MovieLens 100K version.

There are four columns in the MovieLens 100K data set: user ID, item ID (each item is a movie), timestamp, and rating. This example predicts the rating for a specified user ID and an item ID. The data set is very sparse because most combinations of users and movies are not rated.

You can download the compressed archive file from the website at `http://files.grouplens.org/datasets/movielens/ml-100k.zip` and use any third-party unzip tool to extract all the files in the archive to the destination directory of your choice. [1] The file that contains the ratings is *u.data*. Assuming the destination directory is */data*, the following DATA step loads the data table from the directory into your CAS session:

[1] Disclaimer: SAS may reference other websites or content or resources for use at Customer's sole discretion. SAS has no control over any websites or resources that are provided by companies or persons other than SAS. Customer acknowledges and agrees that SAS is not responsible for the availability or use of any such external sites or resources, and does not endorse any advertising, products, or other materials on or available from such websites or resources. Customer acknowledges and agrees that SAS is not liable for any loss or damage that may be incurred by Customer or its end users as a result of the availability or use of those external sites or resources, or as a result of any reliance placed by Customer or its end users on the completeness, accuracy, or existence of any advertising, products, or other materials on, or available from, such websites or resources.

```
proc casutil;
    load file="~/data/u.data"    /*or other user-defined location*/
    casout="movlens"
    importoptions=(filetype="CSV" delimiter="TAB" getnames="FALSE"
                   vars=("userid" "itemid" "rating" "timestamp"));
run;
```

The following statements show how to use PROC FACTMAC to predict movie ratings:

```
proc factmac data=mycas.movlens nfactors=10 learnstep=0.15
             maxiter=20  outmodel=mycas.factors;

   input userid itemid /level=nominal;
   target rating /level=interval;
   output out=mycas.out1 copyvars=(userid itemid rating);
run;
```

The following statements print the first 10 observations in the mycas.factors data table, which is specified in the OUTMODEL= option in the PROC FACTMAC statement. The output is shown in Output 7.1.1.

```
proc print data=mycas.factors(obs=10);
run;
```

Output 7.1.1 Bias Values and Factors

Obs	Variable	Level	Bias	Factor1	Factor2	Factor3	Factor4	Factor5	Factor6	Factor7	Factor8	Factor9	Factor10
1	_GLOBAL_		3.52986	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2	userid	1	0.08043	0.14034	0.36798	-0.52760	-0.31465	0.22486	-0.28397	-0.87780	0.24885	0.02000	-0.14646
3	userid	2	0.17982	0.54746	0.00647	-0.47194	-0.39421	-0.19759	0.51829	-0.10209	0.01729	0.26124	-0.12959
4	userid	3	-0.73356	-0.54006	0.01826	0.31498	0.11264	0.73452	0.31893	-0.06499	-0.63882	-0.45544	0.18285
5	userid	4	0.80347	0.26031	0.04870	-0.25062	-0.01312	-0.29526	0.53290	-0.58693	0.00283	0.36615	0.40131
6	userid	5	-0.65557	0.51211	0.07824	-0.08614	-0.01463	0.46066	-0.30982	-0.21790	0.37157	-1.06146	-0.29942
7	userid	6	0.10521	0.17515	0.18334	-0.51516	0.53364	-0.55709	-0.16770	-0.05254	0.39754	0.29666	0.23568
8	userid	7	0.43540	0.25194	0.07348	-0.05654	-0.02345	-0.24364	0.14093	-0.03300	-0.46717	0.51100	-0.19197
9	userid	8	0.26675	0.11208	0.57564	0.02860	-0.77657	-0.16036	-0.41215	-0.06069	0.76389	-0.00608	-0.11334
10	userid	9	0.74287	0.18856	-0.10608	0.22230	-1.01906	-0.18707	-0.08458	-0.02147	-0.82411	-0.21252	-0.09411

The following statements print the predicted movie ratings for the first 20 observations, as shown in Output 7.1.2.

```
proc print data=mycas.out1(obs=20);
run;
```

Output 7.1.2 Predicted Movie Ratings

Obs	userid	itemid	rating	P_rating
1	196	242	3	4.09834
2	186	302	3	3.78284
3	22	377	1	1.42463
4	244	51	2	3.20907
5	166	346	1	2.58391
6	298	474	4	4.60470
7	115	265	2	3.43183
8	253	465	5	4.57718
9	305	451	3	2.96174
10	6	86	3	4.44866
11	62	257	2	3.08217
12	286	1014	5	3.08536
13	200	222	5	4.46180
14	210	40	3	3.30407
15	224	29	3	3.15151
16	303	785	3	3.03463
17	122	387	5	4.47396
18	194	274	2	2.57941
19	291	1042	4	3.47518
20	234	1184	2	1.62934

References

Rendle, S. (2012). "Factorization Machines with libFM." *ACM Transactions on Intelligent Systems and Technology* 3:1–22.