SAS/STAT® 14.1 User's Guide
The IRT Procedure

This document is an individual chapter from *SAS/STAT® 14.1 User's Guide*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2015. *SAS/STAT® 14.1 User's Guide*. Cary, NC: SAS Institute Inc.

SAS/STAT® 14.1 User's Guide

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

July 2015

Chapter 65
The IRT Procedure

Contents

Overview: IRT Procedure

The item response theory (IRT) model was first proposed in the field of psychometrics for the purpose of ability assessment. It is most widely used in education to calibrate and evaluate items in tests, questionnaires, and other instruments and to score subjects on their abilities, attitudes, or other latent traits. Today, all major psychological and educational tests are built using IRT, because the methodology can significantly improve measurement accuracy and reliability while providing potential significant reductions in assessment time and effort, especially via computerized adaptive testing. In a computerized adaptive test, items are optimally selected for each subject. Different subjects might receive entirely different items during the test. IRT plays an essential role in selecting the most appropriate items for each subject and equating scores for subjects who receive different subsets of items. Notable examples of these tests include the Scholastic Aptitude Test (SAT), Graduate Record Examination (GRE), and Graduate Management Admission Test (GMAT). In recent years, IRT models have also become increasingly popular in health behavior, quality of life, and clinical research. The Patient Reported Outcomes Measurement Information System (PROMIS) project, funded by the US National Institutes of Health, is an excellent example. By using IRT, it aims to develop item banks that clinicians and researchers can use to collect important information about therapeutic effects that is not available from traditional clinical measures.

Early IRT models (such as the Rasch model and two-parameter model) concentrate mainly on dichotomous responses. These models were later extended to incorporate other formats, such as ordinal responses, rating scales, partial credit scoring, and multiple category scoring. Early applications of IRT focused primarily on the unidimensional model, which assumes that subject responses are affected only by a single latent trait. Multidimensional IRT models have been developed, but because of their greater complexity, the majority of IRT applications still rely on unidimensional models.

For an introduction to IRT models, see De Ayala (2009) and Embretson and Reise (2000).

Basic Features

The IRT procedure enables you to estimate various item response theory models. The following list summarizes some of the basic features of the IRT procedure:

- uses the Rasch model; one-, two-, three-, and four-parameter models; graded response model with logistic or probit link; and generalized partial credit model
- enables different items to have different response models
- performs multidimensional exploratory and confirmatory analysis
- performs multiple-group analysis, with fixed values and equality constraints within and between groups
- estimates factor scores by using maximum likelihood (ML), maximum a posteriori (MAP), and expected a posteriori (EAP) methods

Getting Started: IRT Procedure

This example shows how you can use all default settings in PROC IRT to fit an item response model. In this example, there are 50 subjects and each subject responds to 10 items. These 10 items have binary responses: 1 indicates correct and 0 indicates incorrect.

The following DATA step creates the SAS data set IrtBinary:

```
data IrtBinary;
   input item1-item10 @@;
   datalines;
1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 1 0 0 1 1 1 1 1 1
0 0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1

   ... more lines ...

1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1
;
```

The following statements fit an IRT model:

```
proc irt data=IrtBinary;
   var item1-item10;
run;
```

The PROC IRT statement invokes the procedure, and the DATA= option specifies the input data set IrtBinary. The VAR statement names the variables to be used in the model. As you can see from the syntax in this example, fitting a IRT model can be very simple when you use the default settings. These default settings are chosen to reflect setups that are common in practice. Some of the important default settings follow:

- The number of factors is 1.
- The two-parameter logistic model is assumed for binary variables, and the graded response model is assumed for ordinal variables.
- The link function is logistic link.
- The estimation method is based on marginal likelihood.
- The optimization method is the quasi-Newton algorithm.
- The quadrature method is adaptive Gauss-Hermite quadrature, in which the number of quadrature points per dimension is determined adaptively.

As a result, the preceding statements fit two-parameter logistic (2PL) models for all the variables that are listed in the VAR statement.

The first table that PROC IRT produces is the “Modeling Information” table, as shown in Figure 65.1. This table displays basic information about the analysis, such as the name of the input data set, the link function, the number of items and factors, the number of observations, and the estimation method. You can change the link function by using the LINK= option in the PROC IRT statement. You can change the response model for all the items by using the RESFUNC= option in the PROC IRT statement. You can specify different response functions or models for different set of variables by including a MODEL statement. If you want to do multidimensional exploratory analysis, you can simply change the number of factors by using the NFACTOR= option in the PROC IRT statement. For confirmatory analysis, you can use the FACTOR statement to specify the confirmatory factor pattern; the number of factors is implicitly defined by the number of distinctive factor names that you specify in the FACTOR statement.

Figure 65.1 Model Information

The IRT Procedure

Modeling Information	
Data Set	WORK.IRTBINARY
Link Function	Logit
Response Model	Two Parameter Model
Number of Items	10
Number of Factors	1
Number of Observations Read	100
Number of Observations Used	100
Estimation Method	Marginal Maximum Likelihood

The "Item Information" table, shown in Figure 65.2, is displayed by default and can be used to check the item-level information. In this case, each of the 10 variables has two levels, and the raw values for these two levels are 0 and 1, respectively.

Figure 65.2 Item Information

Item Information		
Item	**Levels**	**Values**
item1	2	0 1
item2	2	0 1
item3	2	0 1
item4	2	0 1
item5	2	0 1
item6	2	0 1
item7	2	0 1
item8	2	0 1
item9	2	0 1
item10	2	0 1

The eigenvalues of polychoric correlations are also computed by default and are shown in Figure 65.3. You can use the information from these eigenvalues to assess a reasonable range for the number of factors. For this example, you can observe that the first eigenvalue accounts for almost 50% of the variance, which suggests that there is only one dominant eigenvalue and that a unidimensional model is reasonable for this example. To produce the polychoric correlation table, you specify the POLYCHORIC option in the PROC IRT statement.

Figure 65.3 Eigenvalues of Polychoric Correlation

Eigenvalues of the Polychoric Correlation Matrix				
	Eigenvalue	**Difference**	**Proportion**	**Cumulative**
1	4.71105177	3.51149453	0.4711	0.4711
2	1.19955723	0.14183502	0.1200	0.5911
3	1.05772221	0.26577735	0.1058	0.6968
4	0.79194486	0.07204549	0.0792	0.7760
5	0.71989938	0.17782491	0.0720	0.8480
6	0.54207446	0.12713664	0.0542	0.9022
7	0.41493782	0.10631770	0.0415	0.9437
8	0.30862012	0.12256183	0.0309	0.9746
9	0.18605829	0.11792444	0.0186	0.9932
10	0.06813385		0.0068	1.0000

Next, the "Optimization Information" table, shown in Figure 65.4, lists the optimization technique, the numeric quadrature method, and the number of quadrature points per dimension. If you want to use the expectation-maximization (EM) technique, specify TECHNIQUE=EM in the PROC IRT statement. If you specify the NOAD option in the PROC IRT statement, PROC IRT uses the nonadaptive Gauss-Hermite quadrature to approximate the likelihood. You can change the number of quadrature points by specifying the QPOINTS= option in the PROC IRT statement.

Figure 65.4 Optimization Information

Optimization Information	
Optimization Technique	Quasi-Newton
Likelihood Approximation	Adaptive Gauss-Hermite Quadrature
Number of Quadrature Points	21
Number of Free Parameters	20

Figure 65.5 shows the "Iteration History" table. For each iteration, the table displays the current iteration number, number of function evaluations, objective function value, change of object function value, and maximum value of gradients. You can use this information to monitor the estimation status of the model. You can turn off the display of the "Iteration History" table by specifying the NOITPRINT option in the PROC IRT statement.

Following the "Iteration History" table is the convergence status table, shown in Figure 65.6. It shows whether the optimization algorithm converges successfully or not. You should make sure that the optimization converges successfully before you try to interpret the estimation results.

Figure 65.5 Iteration History

Iteration History					
Cycles	Iteration	Evaluations	Objective Function	Function Change	Max Abs Gradient
0	0	2	5.53317409		0.055021
0	1	4	5.43362007	-0.09955401	0.017514
0	2	6	5.41516422	-0.01845585	0.017074
0	3	8	5.40358645	-0.01157777	0.007847
0	4	10	5.40166068	-0.00192577	0.00726
0	5	12	5.40106299	-0.00059770	0.00349
0	6	15	5.40088871	-0.00017428	0.001478
0	7	18	5.40082522	-0.00006349	0.000859
0	8	21	5.40079959	-0.00002563	0.000804
0	9	24	5.40078941	-0.00001017	0.000441
0	10	27	5.40078290	-0.00000652	0.00014
0	11	30	5.40078218	-0.00000072	0.000156
0	12	32	5.40078188	-0.00000030	0.000202
0	13	34	5.40078147	-0.00000042	0.000032
0	14	37	5.40078144	-0.00000003	0.000024
0	15	40	5.40078142	-0.00000001	9.866E-6
1	0	2	5.40078179		9.953E-6

Figure 65.6 Convergence Status

Convergence criterion (ABSGCONV=.000010000) satisfied.

Next is the "Model Fit Statistics" table, shown in Figure 65.7, which includes the log likelihood, Akaike's information criterion (AIC), and the Bayesian information criterion (BIC). If all the response patterns are observed, Pearson's chi-square and likelihood ratio chi-square statistics are also included in this table. Because some of the response patterns in this example are not observed, the Pearson's chi-square statistic is not included in the table.

Figure 65.7 Fit Statistics

Model Fit Statistics	
Log Likelihood	-540.078179
AIC (Smaller is Better)	1120.1563581
BIC (Smaller is Better)	1172.2597618
LR Chi-Square	300.35009265
LR Chi-Square DF	1003

Finally, the "Item Parameter Estimates" table, shown in Figure 65.8, includes parameter estimates, standard errors, and *p*-values. Parameters are organized and displayed within each item. The items are listed in the order of their appearance in the modeling statements. For each item, there are two parameters: difficulty and slope. Difficulty parameters measure the difficulties of the items. As the value of the difficulty parameter increases, the item becomes more difficult. In Figure 65.8, you can observe that all the difficulty parameters are less than 0, which suggests that all the items in this example are relatively easy. The slope parameter values for this example range from 0.94 to 2.33, suggesting that all the items are adequate measures of the latent trait.

Figure 65.8 Parameter Estimates

Item Parameter Estimates

Item	Parameter	Estimate	Standard Error	Pr > |t|
item1	Difficulty	-0.86577	0.20049	<.0001
	Slope	2.21983	0.68394	0.0006
item2	Difficulty	-1.01629	0.21353	<.0001
	Slope	2.33878	0.76100	0.0011
item3	Difficulty	-0.91125	0.20824	<.0001
	Slope	2.18658	0.68338	0.0007
item4	Difficulty	-0.92384	0.22606	<.0001
	Slope	1.87303	0.57533	0.0006
item5	Difficulty	-1.09216	0.30469	0.0002
	Slope	1.33934	0.42769	0.0009
item6	Difficulty	-0.48811	0.24191	0.0218
	Slope	1.17672	0.37110	0.0008
item7	Difficulty	-0.61766	0.29991	0.0197
	Slope	0.94547	0.32651	0.0019
item8	Difficulty	-0.50800	0.28293	0.0363
	Slope	0.95665	0.32983	0.0019
item9	Difficulty	-0.41109	0.24309	0.0454
	Slope	1.11732	0.35666	0.0009
item10	Difficulty	-0.62318	0.27830	0.0126
	Slope	1.05213	0.34965	0.0013

Syntax: IRT Procedure

The following statements are available in the IRT procedure:

PROC IRT < *options* > ;
  BY *variables* ;
  COV *covariance parameters* ;
  EQUALITY *equality-constraints* ;
  FACTOR *factor-variables-relations* ;
  FREQ *variable* ;
  GROUP *variable* ;
  MODEL *model-specification* ;
  VAR *variables* ;
  VARIANCE *variance parameters* ;
  WEIGHT *variable* ;

PROC IRT Statement

PROC IRT < *options* > **;**

The PROC IRT statement invokes the IRT procedure. Table 65.1 summarizes the *options* available in the PROC IRT statement. The sections that follow the table describe the PROC IRT statement *options* and then describe the other statements in alphabetical order.

Table 65.1 PROC IRT Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data set
DESCENDING	Reverses the sort order of the levels of the response variable
INMODEL=	Inputs the model specifications
ITEMFIT	Computes the item fit statistics and displays them in a table
ITEMSTAT	Computes the classical item statistics and displays them in a table
LINK=	Specifies the link function
NFACTOR=	Specifies the number of factors
OUT=	Specifies the output data set for factor scores
OUTMODEL=	Outputs the model specifications
RESFUNC=	Specifies the response function
RORDER=	Specifies the sort order of the response variables
SCOREMETHOD=	Specifies the factor score estimation method
Computational Options	
ABSFCONV=	Specifies an absolute function difference convergence criterion
ABSGCONV=	Specifies an absolute gradient convergence criterion
ABSPCONV=	Specifies a maximum absolute parameter difference convergence criterion
FCONV=	Specifies a relative function convergence criterion
GCONV=	Specifies a relative gradient convergence criterion
MAXFUNC=	Specifies the maximum number of function calls in the optimization process
MAXITER=	Specifies the maximum number of iterations in the optimization process
MAXMITER=	Specifies the maximum number of iterations in the maximization step of the EM algorithm
NOAD	Specifies nonadaptive quadrature
QPOINTS=	Specifies the number of quadrature points per dimension
TECHNIQUE=	Specifies the optimization technique to obtain maximum likelihood estimates
Display Options	
NOITPRINT	Suppresses the display of the “Iteration History” table
NOPRINT	Suppresses all ODS output
PINITIAL	Displays initial parameter estimates
POLYCHORIC	Displays the polychoric correlation matrix

Table 65.1 *continued*

Option	Description
PLOTS=	Controls plots that are produced through ODS Graphics
Rotation Method and Properties	
RCONVERGE=	Specifies the convergence criterion for rotation cycles
RITER=	Specifies the maximum number of rotation cycles
ROTATE=	Specifies the rotation method

PROC IRT Statement Options

ABSFCONV=*r*

ABSFTOL=*r*

specifies an absolute function difference convergence criterion. Termination requires a small change of the function value in successive iterations,

$$|f(\boldsymbol{\psi}^{(k-1)}) - f(\boldsymbol{\psi}^{(k)})| \le r$$

where $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization and $f(\cdot)$ is the objective function. This criterion is not used by the expectation-maximization (EM) algorithm. By default, *r* = 0.

ABSGCONV=*r*

ABSGTOL=*r*

specifies an absolute gradient convergence criterion. Termination requires the maximum absolute gradient element to be small,

$$\max_j |g_j(\boldsymbol{\psi}^{(k)})| \le r$$

where $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization and $g_j(\cdot)$ is the gradient of the objective function with respect to the *j*th parameter. This criterion is not used by the EM algorithm. By default, *r* = 1E–5.

ABSPCONV=*r*

ABSPTOL=*r*

specifies a maximum absolute parameter difference convergence criterion. This criterion is used only by the EM algorithm. Termination requires the maximum absolute parameter change in successive iterations to be small,

$$\max_j |\boldsymbol{\psi}_j^{(k-1)} - \boldsymbol{\psi}_j^{(k)}| \le r$$

where $\boldsymbol{\psi}_j$ denotes the *j*th parameter that participates in the optimization. By default, *r* = 1E–4.

DATA=*SAS-data-set*

specifies the *SAS-data-set* to be read by PROC IRT. The default value is the most recently created data set.

DESCENDING

DESC

reverses the sorting order for the levels of the response variables. If you specify both the DESCENDING and RORDER= options, PROC IRT orders the levels according to the RORDER= option and then reverses that order.

FCONV=*r*

FTOL=*r*

specifies a relative function convergence criterion. Termination requires a small relative change of the function value in successive iterations,

$$\frac{|f(\boldsymbol{\psi}^{(k)}) - f(\boldsymbol{\psi}^{(k-1)})|}{|f(\boldsymbol{\psi}^{(k-1)})|} \leq r$$

where $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization and $f(\cdot)$ is the objective function. This criterion is not used by the EM algorithm. By default, $r = 10^{-\mathrm{FDIGITS}}$, where FDIGITS is, by default, $-\log_{10}\{\epsilon\}$ and ϵ is the machine precision.

GCONV=*r*

GTOL=*r*

specifies a relative gradient convergence criterion. For all techniques except CONGRA, termination requires the normalized predicted function reduction to be small,

$$\frac{\mathrm{g}(\boldsymbol{\psi}^{(k)})'[\mathbf{H}^{(k)}]^{-1}\mathrm{g}(\boldsymbol{\psi}^{(k)})}{|f(\boldsymbol{\psi}^{(k)})|} \leq r$$

where $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, $f(\cdot)$ is the objective function, and $\mathrm{g}(\cdot)$ is the gradient. For the CONGRA technique (for which a reliable Hessian estimate $\mathbf{H}$ is not available), the following criterion is used:

$$\frac{\parallel \mathrm{g}(\boldsymbol{\psi}^{(k)}) \parallel_2^2 \quad \parallel \mathrm{s}(\boldsymbol{\psi}^{(k)}) \parallel_2}{\parallel \mathrm{g}(\boldsymbol{\psi}^{(k)}) - \mathrm{g}(\boldsymbol{\psi}^{(k-1)}) \parallel_2 |f(\boldsymbol{\psi}^{(k)})|} \leq r$$

This criterion is not used by the EM algorithm. By default, *r* = 1E–8.

INMODEL< (SCORE) >=*SAS-data-set*

specifies an input data set that contains information about the analysis model. Instead of specifying and running the model in a new run, you can use the INMODEL= option to input the model specification saved as an OUTMODEL= data set in a previous PROC IRT run.

Sometimes, you might want to create an INMODEL= data set by modifying an existing OUTMODEL= data set. However, editing and modifying OUTMODEL= data sets requires a good understanding of the formats and contents of the OUTMODEL= data sets. This process could be difficult for novice users. For more information about the format of INMODEL= and OUTMODEL= data sets, see the section“Output Data Sets” on page 4836.

When you specify the INMODEL= option, the VAR, MODEL, GROUP, FACTOR, VARIANCE, COV, and EQUALITY statements are ignored. The DESCENDING, LINK, NFACTOR, RESFUNC, and RORDER options in the PROC IRT statement are also ignored. When there are duplicated specifications, the first specification is used.

Specify the SCORE suboption if you want to use the model specifications and parameter estimates from the INMODEL= data set to score a new subject without refitting the model.

You can use the INMODEL= option along with the SCORE suboption for many different purposes, including the following:

- If you specify the INMODEL= option, PROC IRT fits an IRT model to the DATA= data set based on the model specifications in the INMODEL= data set and uses the parameter estimates in the INMODEL= data set as initial values.
- If you specify the INMODEL= option and the OUT= option, PROC IRT fits an IRT model to the DATA= data set based on the model specifications in the INMODEL= data set and uses the parameter estimates in the INMODEL= data set as initial values. Then PROC IRT scores the DATA= data set by using the new parameter estimates obtained in the previous step.
- If you specify the INMODEL(SCORE)= option and the OUT= option, PROC IRT scores the DATA= data set by using the model specifications and parameter estimates in the INMODEL= data set without refitting the model.

ITEMFIT

displays the item fit statistics. These item fit statistics apply only to binary items that have one latent factor.

ITEMSTAT < (*itemstat-options*) >

displays the classical item statistics, which include the item means, item-total correlations, adjusted item-total correlations, and item means for *i* ordered groups of observations or individuals. You can specify the following *itemstat-options*:

NPARTITION=*i*

specifies the number of groups, where *i* must be an integer between 2 and 5, inclusive. By default NPARTITION=4.

The *i* ordered groups are formed by partitioning subjects based on the rank of their sum scores. By default, there are four groups, labeled G1, G2, G3, and G4, representing four ascending ranges of sum scores. The formula for calculating group values is

$$\mathrm{floor}(\mathrm{rank} \times i/(n+1))$$

where floor is the floor function, *rank* is the sum score's order rank, *i* is the value of the NPARTITION= option, and n is the number of observations that have nonmissing values of sum scores for TIES=LOW, TIES=MEAN, and TIES=HIGH. For TIES=DENSE, n is the number of observations that have unique nonmissing sum scores. If the number of observations is evenly divisible by the number of groups, each group has the same number of observations, provided that there are no tied sum scores at the boundaries of the groups. Sum scores with many tied values can create unbalanced groups because observations that have the same sum scores are assigned to the same group.

TIES=HIGH | LOW | MEAN | DENSE
specifies how to compute normal scores or ranks for tied data values.

HIGH assigns the largest of the corresponding ranks.

LOW assigns the smallest of the corresponding ranks.

MEAN assigns the mean of the corresponding rank.

DENSE computes scores and ranks by treating tied values as a single-order statistic. For the default method, ranks are consecutive integers that begin with the number 1 and end with the number of unique, nonmissing values of the variable that is being ranked. Tied values are assigned the same rank.

By default, TIES=MEAN.

Observations (subjects) that have missing values are excluded from the computations of the classical item statistics.

LINK=*name*
specifies the link function. You can specify the following *names*:

LOGIT requests the logistic link function.

PROBIT requests the probit link function.

By default, LINK=LOGIT.

MAXFUNC=*n*
MAXFU=*n*
specifies the maximum number of function calls in the optimization process. This option is not used by the EM algorithm. The default values are as follows, depending on which optimization technique is specified in the TECHNIQUE= option:

- NRRIDG: 125
- QUANEW: 500
- CONGRA: 1000

The optimization can terminate only after completing a full iteration. Therefore, the number of function calls that are actually performed can exceed the number that this option specifies.

MAXITER=*n*
MAXIT=*n*
specifies the maximum number of iterations in the optimization process. The default values are as follows, depending on which optimization technique is specified in the TECHNIQUE= option:

- NRRIDG: 50
- QUANEW: 200
- CONGRA: 400
- EM: 500

MAXMITER=*n*

MAXMIT=*n*

specifies the maximum number of iterations in the maximization step of the EM algorithm. By default, MAXMITER=1.

NFACTOR=*i*

NFACT=*i*

specifies the number of factors, *i*, in the model. You must specify the number of factors only for exploratory analysis, in which all the slope parameters of the items are freely estimated without being explicitly constrained by using the FACTOR statement. By default, NFACTOR=1. When you use the FACTOR statement to specify the confirmatory factor pattern, the number of factors is implicitly defined by the number of distinctive factor names that you specify in the statement.

NOAD

requests that the Gaussian quadrature be nonadaptive.

NOITPRINT

suppresses the display of the "Iteration History" table.

NOPRINT

suppresses all output displays.

OUT=*SAS-data-set*

creates an output data set that contains all the data in the DATA= data set plus estimated factor scores. For exploratory analysis, the factor scores are named _Factor1, _Factor2, and so on. For confirmatory analysis, user-specified factor names are used.

PROC IRT provides three estimation methods for factor scores. You can specify a method by using the SCOREMETHOD option. The default estimation method, maximum a posteriori (MAP), is used if the SCOREMETHOD option is not specified.

OUTMODEL=*SAS-data-set*

creates an output data set that contains the model specification, the parameter estimates, and their standard errors. You can use an OUTMODEL= data set as an input INMODEL= data set in a subsequent analysis by PROC IRT.

If you want to create a SAS data set in a permanent library, you must specify a two-level name. For more information about permanent libraries and SAS data sets, see *SAS Language Reference: Concepts*.

PINITIAL

displays the initial parameter estimates.

PLOTS < **(***global-plot-options***)** > < **=** *plot-request* < **(***options***)** > >

PLOTS < **(***global-plot-options***)** > < **=** **(***plot-request* < **(***options***)** > < **. . .** *plot-request* < **(***options***)** > > **)** >

controls the plots that are produced through ODS Graphics. When you specify only one *plot-request*, you can omit the parentheses around it. For example:

```
plots=all
plots=ICC(unpack)
plots(unpack)=(scree ICC)
```

ODS Graphics must be enabled before plots can be requested. For example:

```
ods graphics on;
proc irt plots=all;
run;
ods graphics off;
```

For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" on page 609 in Chapter 21, "Statistical Graphics Using ODS."

You can specify the following *global-plot-options*, which apply to all plots that the IRT procedure generates:

UNPACK | UNPACKPANEL

suppresses paneling. By default, multiple plots can appear in some output panels. Specify UNPACK to display each plot individually. You can also specify UNPACK as a suboption in the ICC, IIC, and SCREE options.

XVIEWMAX

specifies a maximum value for the X axis. You can also specify XVIEWMAX as a suboption in the ICC, IIC, and TIC options.

XVIEWMIN

specifies a minimum value for the X axis. You can also specify XVIEWMIN as a suboption in the ICC, IIC, and TIC options.

You can specify the following *plot-requests*:

ALL

displays all default plots.

ICC < (UNPACK | UNPACKPANEL), (XVIEWMAX=), (XVIEWMIN=) >

displays item characteristic curves (ICCs). By default, multiple ICC plots appear in some output panels. You can request an individual ICC plot for each item by specifying the UNPACK suboption. For binary items, the ICC plot includes only the curve for the higher category, which is often the correct response category or the endorsed category. For ordinal items that have more than two categories, the ICC plot includes curves for all the categories and also a legend with the values 0, 1, 2, and so on to indicate the curves for different categories.

IIC < (UNPACK | UNPACKPANEL), (XVIEWMAX=), (XVIEWMIN=) >

displays item information curves (IICs). By default, multiple IIC plots appear in some output panels. You can request an individual IIC plot for each item by specifying the UNPACK suboption.

NONE
suppresses all plots.

POLYCHORIC < *options* >

PLCORR< *options* >
displays a heat map of the polychoric correlation matrix. You can specify one or both of the following *options*:

FUZZ=*p*
displays polychoric correlations whose absolute values are less than *p* as 0 in the heat map. This option is useful when you want to focus on the patterns of sizable correlations that are larger than *p* in the heat map. By default, FUZZ=0.

OUTLINE=ON | OFF
specifies whether to display an outline of the regions in the polychoric correlation heat map. By default, OUTLINE=ON.

SCREE < **(UNPACK | UNPACKPANEL)** >
displays the scree and variance-explained plots in the same panel. You can display these plots individually by specifying the UNPACK suboption.

TIC < **(XVIEWMAX=), (XVIEWMIN=)** >
displays a test information curve (TIC) plot.

POLYCHORIC
displays the polychoric correlation matrix.

QPOINTS=*i*
specifies the number of quadrature points in each dimension of the integral. If there are d latent factors and n quadrature points, the IRT procedure evaluates n^d conditional log likelihoods for each observation to compute one value of the objective function. Increasing the number of quadrature nodes can substantially increase the computational burden. If you do not specify the number of quadrature points, it is determined adaptively by using the initial parameter estimates.

RCONVERGE=*p*

RCONV=*p*
specifies the convergence criterion for rotation cycles. Rotation stops when the scaled change of the simplicity function value is less than the RCONVERGE= value. The default convergence criterion is

$$|f_{new} - f_{old}|/K < \epsilon$$

where f_{new} and f_{old} are simplicity function values of the current cycle and the previous cycle, respectively; $K = \max(1, |f_{old}|)$ is a scaling factor; and ϵ is 1E–9 by default and is modified by the RCONVERGE= value.

RESFUNC=ONEP | TWOP | THREEP | FOURP | GRADED | RASCH

specifies the response functions for the variables that are included in the VAR statement. The response functions correspond to different response models. You can specify the following values:

ONEP	specifies the one-parameter model.
TWOP	specifies the two-parameter model.
THREEP	specifies the three-parameter model.
FOURP	specifies the four-parameter model.
GRADED	specifies the graded response model.
RASCH	specifies the Rasch model.

By default, RESFUNC=TWOP for binary items and RESFUNC=GRADED for ordinal items. The graded response model assumes that the response variables are ordinal-categorical up to 11 levels. All other models assume binary responses. For more information about these response models, see "Response Models" in the "Details: IRT Procedure" on page 4826 section.

RITER=*n*

specifies the maximum number of cycles for factor rotation. The default value is the maximum between 10 times the number of variables and 100.

RORDER=DATA | FORMATTED | FREQ | INTERNAL

specifies the sort order for the levels of the response variable. This order determines which threshold parameter in the model corresponds to each level in the data. If RORDER=FORMATTED for numeric variables for which you have supplied no explicit format, the levels are ordered by their internal values. This option applies to all the responses in the model. When the default, RORDER=FORMATTED, is in effect for numeric variables for which you have supplied no explicit format, the levels are ordered by their internal values. You can specify the following sort orders:

Value of RORDER=	Levels Sorted By
DATA	Order of appearance in the input data set
FORMATTED	External formatted value, except for numeric variables that have no explicit format, which are sorted by their unformatted (internal) value
FREQ	Descending frequency count; levels that contain the most observations come first in the order
INTERNAL	Unformatted value

For FORMATTED and INTERNAL, the sort order is machine-dependent. For more information about sort order, see the chapter on the SORT procedure in the *SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

ROTATE=*name*

R=*name*

specifies the rotation method.

You can specify the following orthogonal rotation methods:

BIQUARTIMAX | BIQMAX	specifies orthogonal biquartimax rotation.
EQUAMAX | E	specifies orthogonal equamax rotation.
NONE | N	specifies that no rotation be performed, leaving the original orthogonal solution.
PARSIMAX | PA	specifies orthogonal parsimax rotation.
QUARTIMAX | QMAX | Q	specifies orthogonal quartimax rotation.
VARIMAX | V	specifies orthogonal varimax rotation.

You can specify the following oblique rotation methods:

BIQUARTIMIN | BIQMIN	specifies biquartimin rotation.
COVARIMIN | CVMIN	specifies covarimin rotation.
OBBIQUARTIMAX | OBIQMAX	specifies oblique biquartimax rotation.
OBEQUAMAX | OE	specifies oblique equamax rotation.
OBPARSIMAX | OPA	specifies oblique parsimax rotation.
OBQUARTIMAX | OQMAX	specifies oblique quartimax rotation.
OBVARIMAX | OV	specifies oblique varimax rotation.
QUARTIMIN | QMIN	specifies quartimin rotation.

By default, ROTATE=VARIMAX.

SCOREMETHOD=ML | EAP | MAP
specifies the method of factor score estimation. You can specify the following methods:

ML	requests the maximum likelihood method.
EAP	requests the expected a posteriori method.
MAP	requests the maximum a posteriori method.

By default, SCOREMETHOD=MAP.

TECHNIQUE=CONGRA | EM | NONE | NRRIDG | QUANEW
TECH=CONGRA | EM | NONE | NRRIDG | QUANEW
OMETHOD=CONGRA | EM | NONE | NRRIDG | QUANEW
specifies the optimization technique to obtain maximum likelihood estimates. You can specify the following techniques:

CONGRA	performs a conjugate-gradient optimization.
EM	performs an EM optimization.
NONE	performs no optimization.
NRRIDG	performs a Newton-Raphson optimization with ridging.
QUANEW	performs a dual quasi-Newton optimization.

By default, TECHNIQUE=QUANEW.

For more information about these optimization methods (except EM), see the section "Choosing an Optimization Algorithm" on page 505 in Chapter 19, "Shared Concepts and Topics." For more information about the EM algorithm, see "Expectation-Maximization (EM) Algorithm" in the section "Details: IRT Procedure" on page 4826.

BY Statement

BY *variables* **;**

You can specify a BY statement with PROC IRT to obtain separate analyses of observations in groups that are defined by the BY variables. When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables. If you specify more than one BY statement, only the last one specified is used.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data by using the SORT procedure with a similar BY statement.
- Specify the NOTSORTED or DESCENDING option in the BY statement for the IRT procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.
- Create an index on the BY variables by using the DATASETS procedure (in Base SAS software).

Because sorting the data changes the order in which PROC IRT reads observations, the sort order for the levels of the response variables might be affected if you also specify RORDER=DATA in the PROC IRT statement.

For more information about BY-group processing, see the discussion in *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see the discussion in the *Base SAS Procedures Guide*.

COV Statement

COV *assignment* < **,** *assignment . . .* > **;**

where *assignment* represents

var-list < ∗ *var-list2* > < **=** *parameter-spec* >

The COV statement defines the factor covariances in confirmatory models. In each *assignment* of the COV statement, you specify variables in the *var-list* and *var-list2* lists, followed by the covariance parameter specification in the *parameter-spec* list. The last two specifications are optional.

You can specify the following five types of the parameters for the covariances:

- an unnamed free parameter

- an initial value
- a fixed value
- a free parameter with a name provided
- a free parameter with a name and initial value provided

Consider a multidimensional model that has the latent factors FACTOR1, FACTOR2, FACTOR3, and FACTOR4. The following COV statement shows the five types of specifications in five *assignments*:

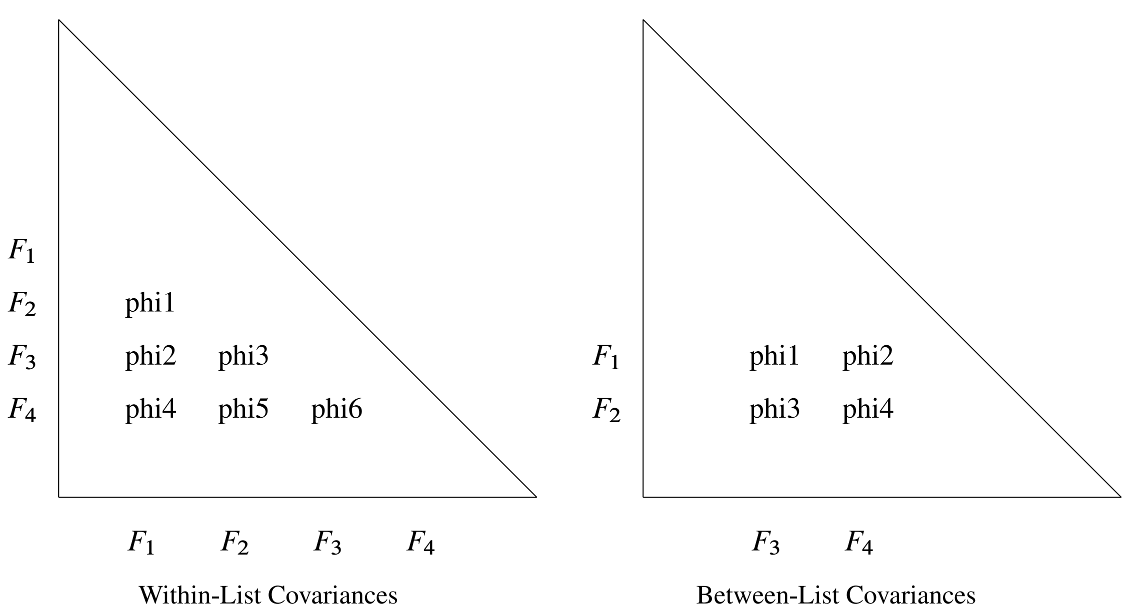
```
cov FACTOR2 FACTOR1 ,
    FACTOR3 FACTOR1 = (0.3),
    FACTOR3 FACTOR2 = 1.0,
    FACTOR4 FACTOR1 = phi1,
    FACTOR4 FACTOR2 = phi2(0.2);
```

In this statement, **cov(FACTOR2,FACTOR1)** is specified as an unnamed free parameter, **cov(FACTOR3,FACTOR1)** is an unnamed free parameter but with an initial value of 0.3, and **cov(FACTOR3,FACTOR2)** is a fixed value of 1.0. This value stays the same in the estimation. **cov(FACTOR4,FACTOR1)** is a free parameter named phi1, and **cov(FACTOR4,FACTOR2)** is a free parameter named phi2 that has an initial value of 0.2.

Note that the *var-list* and *var-list2* lists to the left of the equal sign in the COV statement should contain only names of latent factors that are specified in the FACTOR statement.

If you specify only the *var-list* list, then you are specifying the so-called within-list covariances. If you specify both the *var-list* and *var-list2* lists, then you are specifying the so-called between-list covariances. An asterisk is used to separate the two variable lists. You can use one of these two alternatives to specify the covariance parameters. Figure 65.9 illustrates the within-list and between-list covariance specifications.

Figure 65.9 Within-List and Between-List Covariances



Within-List Covariances

The left panel of Figure 65.9 shows that the same set of four factors is used in both the rows and the columns. This yields six nonredundant covariances (variances are not included) to specify. In general, for a *var-list* list that has k variables in the COV statement, you can specify $k(k-1)/2$ distinct covariance parameters. The variable order of the *var-list* list is important. For example, the left panel of Figure 65.9 corresponds to the following COV statement:

```
cov F1-F4 = phi1-phi6;
```

This statement is equivalent to the following statement:

```
cov F2 F1 = phi1,
    F3 F1 = phi2, F3 F2 = phi3,
    F4 F1 = phi4, F4 F2 = phi5, F4 F3 = phi6;
```

Another way to assign distinct parameter names that have the same prefix is to use the so-called prefix name. For example, the following COV statement is exactly the same as the preceding statement:

```
cov F1-F4 = 6*phi__; /* phi with two trailing underscores */
```

In the COV statement, phi__ is a prefix name that has the root phi. The notation **6*** means that this prefix name is applied six times, resulting in a generation of the six parameter names phi1, phi2, ..., phi6 for the six covariance parameters.

The root of the prefix name should have only a few characters so that the generated parameter name is not longer than 32 characters. To avoid unintentional equality constraints, the prefix names should not conflict with other parameter names.

You can also specify the within-list covariances as unnamed free parameters, as shown in the following statement:

```
cov F1-F4;
```

This statement is equivalent to the following statement:

```
cov F2 F1,
    F3 F1, F3 F2,
    F4 F1, F4 F2, F4 F3;
```

Between-List Covariances

The right panel of Figure 65.9 illustrates the application of the between-list covariance specification. The set of row variables is different from the set of column variables. You intend to specify the cross covariances of the two sets of variables. There are four of these covariances in the figure. In general, for k_1 and k_2 variable names in the two variable lists (separated by an asterisk) in a COV statement, there are $k_1 \times k_2$ distinct covariances to specify. Again, variable order is very important. For example, the right panel of Figure 65.9 corresponds to the following between-list covariance specification:

```
cov F1 F2 * F3 F4 = phi1-phi4;
```

This is equivalent to the following statement:

```
cov  F1 F3 = phi1, F1 F4 = phi2,
     F2 F3 = phi3, F2 F4 = phi4;
```

You can also use the prefix name specification for the same specification, as shown in the following statement:

```
cov  F1 F2 * F3 F4 = 4*phi__ ; /* phi with two trailing underscores */
```

Mixed Parameter Lists

You can specify different types of parameters for the list of covariances. For example, you use a list of parameters that have mixed types in the following statement:

```
cov F1-F4 = phi1(0.1)  0.2  phi3  phi4(0.4) (0.5) phi6;
```

This statement is equivalent to the following statement:

```
cov F2 F1 = phi1(0.1) ,
    F3 F1 = 0.2       , F3 F2 = phi3,
    F4 F1 = phi4(0.4) , F4 F2 = (0.5), F4 F3 = phi6;
```

Notice that an initial value that follows a parameter name is associated with the free parameter. Therefore, in the original mixed list specification, 0.1 is interpreted as the initial value of the parameter phi1, but not as the initial estimate of the covariance between F3 and F1. Similarly, 0.4 is the initial value of the parameter phi4, but not the initial estimate of the covariance between F4 and F2.

However, if you indeed want to specify that phi1 is a free parameter *without* an initial value and 0.1 is an initial estimate of the covariance between F3 and F1 (while keeping all other things the same), you can use a null initial value specification for the parameter phi1, as shown in the following statement:

```
cov F1-F4 = phi1() (0.1)  phi3  phi4(0.4) (0.5) phi6;
```

This way, 0.1 becomes the initial estimate of the covariance between F3 and F1. Because a parameter list that has mixed types might be confusing, you can break down the specifications into separate *assignments* to remove ambiguities. For example, you can use the following equivalent statement:

```
cov F2 F1 = phi1 ,
    F3 F1 = (0.1)     , F3 F2 = phi3,
    F4 F1 = phi4(0.4) , F4 F2 = (0.5), F4 F3 = phi6;
```

Shorter and Longer Parameter Lists

If you provide fewer parameters than the number of covariances in the variable lists, all the remaining parameters are treated as unnamed free parameters. For example, the following statement assigns a fixed value to `cov(F1,F3)` while treating all the other three covariances as unnamed free parameters:

```
cov  F1 F2 * F3 F4 = 1.0;
```

This statement is equivalent to the following statement:

```
cov  F1 F3 = 1.0, F1 F4, F2 F3, F2 F4;
```

If you intend to fill up all values by the last parameter specification in the list, you can use the continuation syntax `[...]`, `[..]`, or `[.]`, as in the following example:

```
cov  F1 F2 * F3 F4 = 1.0 phi [...];
```

This means that **cov(F1,F3)** is a fixed value of 1 and all the remaining three covariances are free parameters named phi. The last three covariances are thus constrained to be equal by having the same parameter name.

However, you must be careful not to provide too many parameters. For example, the following statement results in an error:

```
cov  F1 F2 * F3 F4 = 1.0 phi2(2.0) phi3 phi4 phi5 phi6;
```

The parameters after phi4 are excessive.

Default Covariance Parameters

In exploratory analysis, all factor covariances are fixed at zero for the unrotated or orthogonally rotated solutions. For confirmatory analysis, by default all factor covariances are fixed at zero. You can also use the COV statement to override these default covariance parameters in situations where you want to set parameter constraints or provide initial or fixed values.

EQUALITY Statement

EQUALITY | EQCON *equality-constraints* < **,** *equality-constraints* . . . > **;**

where *equality-constraints* is defined as

variable-list < / *constraint-options* >

The EQUALITY statement provides a versatile way to specify various types of equality constraints on the parameters in the model. You can specify within-group or between-group equality constraints on specific sets of parameters for particular sets of variables or factors. In the *variable-list*, you specify the set of variables that are subject to the equality constraints on their respective parameters. You can either specify the names of the variables or use one of the support keywords (see list later in this section) for *variable-list*. In the *constraint-options*, you specify the types of parameters, the specific groups (in multiple-group analysis), and the specific factors (in multidimensional models) on which the equality constraints are imposed.

For example, the following statements specify that all related parameters of x1 through x5 are constrained to be equal:

```
proc irt;
   model x1-x10/resfunc=graded;
   equality x1-x5;
run;
```

Because all items are fitted by the graded response model, all slopes for variables x1–x5 are constrained to be the same and the intercepts for variables x1–x5 are also constrained to be the same. For example, if each of these variables has five categories, there would be four set of constraints, respectively, for each of the four intercept parameters over the five variables.

You can limit the set of parameters for the equality constraints by specifying the PARM= option (one of the *constraint-options*). For example, the following statements constrain only the slope parameters of x1–x5, instead of all related parameters in the graded response model:

```
proc irt;
   model x1-x10/resfunc=graded;
   equality x1-x5/parm=[slopes];
run;
```

There are various ways to specify the target set of variables that are subject to the equality constraints. You can specify variables directly, or you can specify the following *variable-list*:

ALL
: specifies all variables and factors in the analysis. Equality constraints on parameters related to factors apply to multiple-group analysis only.

ALLITEM
: specifies all variables in the analysis.

ALLONEP

ALLONEPITEM
: specifies all variables that are fitted by the one-parameter model in the analysis.

ALLTWOP

ALLTWOPITEM
: specifies all variables that are fitted by the two-parameter model in the analysis.

ALLTHREEP

ALLTHREEPITEM
: specifies all variables that are fitted by the three-parameter model in the analysis.

ALLFOURP

ALLFOURPITEM
: specifies all variables that are fitted by the four-parameter model in the analysis.

ALLGPC

ALLGPCITEM
: specifies all variables that are fitted by the generalized partial credit model in the analysis.

ALLGR

ALLGRITEM
: specifies all variables that are fitted by the graded response model in the analysis.

ALLRASCH

ALLRASCHITEM
: specifies all variables that are fitted by the Rasch model in the analysis.

You can also specify the following keywords, with a list of *excluded-variables* for *variable-list*:

_ALL_BUT_ [*excluded-variables*]
: specifies all variables and factors except the *excluded-variables* in the analysis.

_ALLITEM_BUT_ [*excluded-variables*]
specifies all variables except the *excluded-variables* in the analysis.

_ALLONEP_BUT_ [*excluded-variables*]
_ALLONEPITEM_BUT_ [*excluded-variables*]
specifies all variables except the *excluded-variables* that are fitted by the one-parameter model in the analysis.

_ALLTWOP_BUT_ [*excluded-variables*]
_ALLTWOPITEM_BUT_ [*excluded-variables*]
specifies all variables except the *excluded-variables* that are fitted by the two-parameter model in the analysis.

_ALLTHREEP_BUT_ [*excluded-variables*]
_ALLTHREEPITEM_BUT_ [*excluded-variables*]
specifies all variables except the *excluded-variables* that are fitted by the three-parameter model in the analysis.

_ALLFOURP_BUT_ [*excluded-variables*]
_ALLFOURPITEM_BUT_ [*excluded-variables*]
specifies all variables except the *excluded-variables* that are fitted by the four-parameter model in the analysis.

_ALLGPC_BUT_ [*excluded-variables*]
_ALLGPCITEM_BUT_ [*excluded-variables*]
specifies all variables except the *excluded-variables* that are fitted by the generalized partial credit model in the analysis.

_ALLGR_BUT_ [*excluded-variables*]
_ALLGRITEM_BUT_ [*excluded-variables*]
specifies all variables except the *excluded-variables* that are fitted by the graded response model in the analysis.

_ALLRASCH_BUT_ [*excluded-variables*]
_ALLRASCHITEM_BUT_ [*excluded-variables*]
specifies all variables except the *excluded-variables* that are fitted by the Rasch model in the analysis.

For example, if you have mixed model types for the item responses, the equality constraints might be set on a particular set of response variables. The following example shows that the equality constraints are applied to those variables that are fitted by the three-parameter model (that is, x7–x10):

```
proc irt;
   model x1-x6/resfunc=graded,
         x7-x10/resfunc=threep;
   equality _allthreep_;
run;
```

Suppose that the preceding model does not fit well and you want to consider a less restricted model in which the equality constraints are imposed on all variables except x10 in the three-parameter model. The following statements achieve this purpose:

```
proc irt;
   model x1-x6/resfunc=graded,
         x7-x10/resfunc=threep;
   equality _allthreep_but_(x10);
run;
```

In the *constraint-options*, you can specify options for parameter types (PARM= option), the set of groups (BETWEEN_GP= and WITHIN_GP= options), and the set of factors. If you do not use these options, all related parameter types, all groups, and all factors are subject to the constraints for the specified set of variables. You can specify the following *constraint-options*:

BET < **= [** *group-list* **]** >

BETWEEN < **= [** *group-list* **]** >

BETWEEN_GP < **= [** *group-list* **]** >

specifies that the equality constraints be applied across or between groups in the multiple-group analysis. Setting between-group constraints is the default when you fit multiple-group models. Hence, it is not necessary to use this option when you want to set equality constraints between all groups. When only a subset of groups is subject to the intended constraints, you can specify the groups in the *group-list*. This option has no effect if you have only one group in the analysis.

PARM < **= [***parameter-types***]** >

specifies the particular types of parameters that are subject to equality constraints. By default, all related parameters are subject to the constraints. You can specify the following *parameter-types*:

CEIL

CEILING

indicates that the ceiling parameters are constrained.

GUESS

GUESSING

indicates that the guessing parameters are constrained.

INTERCEPT

indicates that the intercept parameters are constrained. For unidimensional models, the difficulty or threshold parameter rather than the intercept parameters are displayed in the parameter estimates table. Intercept parameter equals to difficulty or threshold parameter times the slope parameter.

SLOPE < **[** *factor-list* **]** >

DISCRIMINATION < **[** *factor-list* **]** >

indicates that the slope or discrimination parameters are constrained. The optional *factor-list* indicates the set of factors to which the constrained slope parameters pertain. The use of *factor-list* is relevant only when you conduct a confirmatory analysis by specifying the factor pattern in the FACTOR statement.

WIT < **=** [*group-list*] >
WITHIN < **=** [*group-list*] >
WITHIN_GP< **=** [*group-list*] >

specifies that the equality constraints be applied within groups in multiple-group analyses. Setting within-group constraints is the default when you fit a single-group model. Hence, this option is not necessary when you have only one group in the analysis. In multiple-group analyses, between-group constraints are set by default. When you specify this option, within-group constraints are set instead of between-group constraints. You can also specify the specific groups in the *group-list* that are subject to the within-group constraints. The default is to apply the equality constraints to all groups.

You can combine the *constraint-options* to set various types of constraints for your model. You can also specify more than one constraint in an EQUALITY statement. You can even use multiple EQUALITY statements for better organization of the constraints.

For example, suppose that a single-group analysis is conducted using three different types of models (two-parameter, graded responses, and three-parameter model) for the response variables. Consider the following statements:

```
proc irt;
   model x1-x10/resfunc=twop, x11-x20/resfunc=graded, x21-x30/resfunc=threep;
   equality _alltwop_but_(x9-x10),
         x11-x25 / parm=[slope],
         _allthreep_ / parm=[guess];
run;
```

The first set of equality constraints applies to the intercept and slope parameters of x1–x8, leaving the parameters of x9 and x10 freely estimated. The second set of equality constraints applies to the slope parameters of variables x11–x25, even though x21–x25 have a different model type than x11–x20. The third set of equality constraints applies to the guessing parameters of all variables that are fitted by the three-parameter model (that is, x21–x30).

In multiple-group analysis, constraints are set across groups by default. But within-group constraints can also be set by using the WITHIN_GP option. Suppose there are three groups in the analysis and the grouping variable GP has three distinct values, 1, 2, and 3. Consider the following example:

```
proc irt;
   group GP;
   model x1-x10/resfunc=twop,
         x11-x20/resfunc=graded,
         x21-x30/resfunc=threep;
   equality _alltwop_but_(x9-x10),
         x11-x25 / parm=[slope] between_gp=[1 2],
         _allthreep_ / parm=[guess] within_gp=[1 3];
run;
```

This example is quite similar to the preceding example, but with some modifications from using the BETWEEN_GP and WITHIN_GP options.

The first set of equality constraints is specified exactly the same way as in the preceding example. However, the effect is much different. In the current multiple-group example, the specification constrains the parameters across groups by default. This means that the intercept and slope parameters of x1–x8 are constrained over the three groups. So there would be 16 sets of equality constraints, respectively, for the 16 parameters in variables over the three groups. However, if you use the WITHIN_GP option, the parameters for x1–x8 are the same within each group. This results in three separate sets of equality constraints on 16 intercept parameters, respectively, for the three groups. Moreover, if you use the WITHIN_GP and BETWEEN_GP options together, all 48 parameters in the groups are constrained to be the same.

The second set of equality constraints applies to the slope or discrimination parameters of variables x11–x25 across groups 1 and 2 only, but not all groups. This means that there are 15 equality constraints, respectively, for the 15 slope or discrimination parameters in variables across groups 1 and 2. The discrimination parameters for these variables are not constrained within groups.

The third set of equality constraints applies to the guessing parameters of variables x21–x30 (that is, all the variables that are fitted by the three-parameter model) within groups 1 and 3, respectively. The guessing parameters for these variables are not constrained across groups.

FACTOR Statement

FACTOR *factor-variables-relation* < **,** *factor-variables-relation . . .* > **;**

where each *factor-variables-relation* is defined as

factor right-arrow var-list < **=** *parameter-spec* >

where *factor* is a name that represents an intended factor; *right-arrow* is **===>**, **--->**, **==>**, **-->**, **=>**, **->**, or **>**; *var-list* is a list of variables that have nonzero slopes associated with the factor; and *parameter-spec* represents the specifications of parameter name and values (fixed or initial).

You use the FACTOR statement to specify the pattern of relationships between variables and factors in confirmatory models. You do not need to use the FACTOR statement if you are fitting an exploratory model. To complete the specification of a confirmatory model, you might need to use the VARIANCE and COV statements to specify the variance and covariance parameters in the model, as shown in the following syntax:

FACTOR *factor-variable-relation* < **,** *factor-variables-relation . . .* > **;**
VARIANCE *variance-parameters* **;**
COV *covariance-parameters* **;**

The specifications in the FACTOR statement concern the pattern in the slope matrix. More details follow after a brief description of the VARIANCE and COV statements.

By default, the factor variances are fix parameters with a value of 1 in the confirmatory factor model. However, you can override these default parameters by specifying them explicitly in the VARIANCE statement. For example, in some confirmatory factor models, you might want to free some of these factor variances, or you might want to set equality constraints by using the same parameter name at different parameter locations in your model. Note that if you free some of the factor variances, you need to fix some slope parameters to identify the model.

By default, factor covariances are zeros in the confirmatory model. However, you can override these default covariance parameters by specifying them explicitly in the COV statement.

Because the default parameterization of the confirmatory model already covers most commonly used parameters, the specifications in the VARIANCE and COV statements are secondary to the specifications in the FACTOR statement, which specifies the pattern of the slope matrix. The following example statement illustrate the syntax of the confirmatory FACTOR statement. Suppose there are nine variables, V1–V9, in your sample and you want to fit a confirmatory IRT model with four factors, as follows:

```
factor
   g_factor   ===>   V1-V9 ,
   factor_a   ===>   V1-V3 ,
   factor_b   ===>   V4-V6 ,
   factor_c   ===>   V7-V9 ;
```

In this factor model, you assume a general factor, g_factor, and three group factors, factor_a, factor_b, and factor_c. The general factor, g_factor, is related to all variables in the sample, whereas each group factor is related to only three variables. This example fits the following pattern of the slope matrix:

	g_factor	factor_a	factor_b	factor_c
V1	x	x		
V2	x	x		
V3	x	x		
V4	x		x	
V5	x		x	
V6	x		x	
V7	x			x
V8	x			x
V9	x			x

Here an x represents an unnamed free parameter, and all other cells that are blank are fixed zeros.

You can specify the following five types of parameters (*parameter-spec*) at the end of each *factor-variables-relation*:

- an unnamed free parameter
- an initial value
- a fixed value
- a free parameter with a name provided
- a free parameter with a name and initial value provided

To illustrate these different types of parameter specifications, consider the following pattern of slopes:

	g_factor	factor_a	factor_b	factor_c
V1	g_load1	1.		
V2	g_load2	x		
V3	g_load3	x		
V4	g_load4		1.	
V5	g_load5		load_a	
V6	g_load6		load_b	
V7	g_load7			1.
V8	g_load8			load_c
V9	g_load9			load_c

Here an x represents an unnamed free parameter, a constant 1 represents a fixed value, and each name in a cell represents a name for a free parameter. You can specify this pattern by using the following FACTOR statement:

```
factor
   g_factor   ===>   V1-V9    = g_load1-g_load9 (9*0.6),
   factor_a   ===>   V1-V3    = 1. (.7  .8),
   factor_b   ===>   V4-V6    = 1. load_a (.9) load_b,
   factor_c   ===>   V7-V9    = 1. 2*load_c ;
```

In the first entry of the FACTOR statement, you specify that the slopes of V1–V9 on g_factor are the free parameters g_load1–g_load9, all of which are given an initial estimate of 0.6. The syntax **9*0.6** means that **0.6** is repeated nine times. Because they are enclosed in parentheses, all these values are treated as initial estimates but not as fixed values.

You can split the second entry of the FACTOR statement into the following specification:

```
   factor_a   ===>   V1    = 1.  ,
   factor_a   ===>   V2    = (.7),
   factor_a   ===>   V3    = (.8),
```

This means that the first slope is a fixed value of 1 and that the other slopes are unnamed free parameters that have initial estimates of 0.7 and 0.8, respectively.

You can split the third entry of the FACTOR statement into the following specification:

```
   factor_b   ===>   V4    = 1.  ,
   factor_b   ===>   V5    = load_a (.9),
   factor_b   ===>   V6    = load_b,
```

This means that the first slope is a fixed value of 1, the second slope is a free parameter named load_a with an initial estimate of 0.9, and the third slope is a free parameter named load_b without an initial estimate. PROC IRT generates the initial value of this free parameter.

The fourth entry of the FACTOR statement states that the first slope is a fixed 1 and the remaining two slopes are free parameters named load_c. No initial estimate is given. But because the two slopes have the same parameter name, they are constrained to be equal in the estimation.

Notice that an initial value that follows a parameter name is associated with the free parameter. For example, in the third entry of the FACTOR statement, the specification **(.9)** after load_a is interpreted as the initial value of the parameter load_a, but not as the initial estimate of the next slope of V6.

However, if you indeed want to specify that load_a is a free parameter *without* an initial value and **(0.9)** is an initial estimate for the slope of V6, you can use a null initial value specification for the parameter load_a, as shown in the following specification:

```
   factor_b   ===>   V4-V6    = 1. load_a() (.9),
```

This way, 0.9 becomes the initial estimate of the slope of V6. Because a parameter list that contains mixed parameter types might be confusing, you can split the specification into separate entries to remove ambiguities. For example, you can use the following equivalent specification:

```
   factor_b   ===>   V4    = 1.,
   factor_b   ===>   V5    = load_a,
   factor_b   ===>   V6    = (.9),
```

Shorter and Longer Parameter Lists

If you provide fewer parameters than the number of slopes that are specified in the corresponding *factor-variable-relation*, all the remaining parameters are treated as unnamed free parameters. For example, the following statement assigns a fixed value of 1.0 to the first slope, while treating the remaining two slopes as unnamed free parameters:

```
factor
   factor_a    ===>   V1-V3    = 1.;
```

This statement is equivalent to the following statement:

```
factor
   factor_a    ===>   V1      = 1.,
   factor_a    ===>   V2 V3      ;
```

If you intend to fill up all values with the last parameter specification in the list, you can use the continuation syntax `[...]`, `[..]`, or `[.]`, as shown in the following example:

```
factor
   g_factor    ===>   V1-V30    = 1.  (.5) [...];
```

This means that the slope of V1 on g_factor is a fixed value of 1.0 and that the remaining 29 slopes are unnamed free parameters, all of which are given an initial estimate of 0.5.

However, you must be careful not to provide too many parameters. For example, the following statement results in an error:

```
factor
   g_factor    ===>   V1-V3    = load1-load6;
```

The parameter list has six parameters for three slopes. Parameters after load3 are excessive.

Default Parameters

It is important to understand the default parameters in the FACTOR model. First, if you know which parameters are default free parameters, you can make your specification more efficient by omitting the specifications of those parameters that can be set by default.

FREQ Statement

FREQ *variable* **;**

If one variable in your data set represents the frequency of occurrence for the other values in the observation, specify the variable's name in a FREQ statement. PROC IRT then treats the data set as if each observation appears n_i times, where n_i is the value of the FREQ variable for observation i. Only the integer portion of the value is used. If the value of the FREQ variable is less than 1 or is missing, that observation is not included in the analysis. The total number of observations is considered to be the sum of the FREQ values.

GROUP Statement

GROUP *variable* **;**

The GROUP statement specifies the grouping variable that defines the groups of the observations. This statement is required if you intend to do a multiple-group analysis. The values of the grouping variable can be either integers or character strings. PROC IRT analyzes the input data set and determines the number of distinct groups in the data set by counting the number of distinct values in the grouping variable. Because there is no other explicit way to specify the number of groups or the grouping values, you must make sure that all (and only) the intended groups have been indexed properly by the grouping variable in the data set for a multiple-group analysis.

MODEL Statement

MODEL *model-specification* < **,** *model-specification* . . . > **;**

where *model-specification* is defined as

variable-list < / *model-option* >

The MODEL statement specifies the items and their response functions or models. You can specify different response models for different items. In the *variable-list*, you specify the set of variables that use the same model.

You can specify the following *model-option*:

RESFUNC < = [*response-model-types* **] >**
: specifies the response function or model. For available keywords, see the RESFUNC= option in the PROC IRT statement. For technical details about these response models, see "Response Models" in the section "Details: IRT Procedure" on page 4826.

You can specify mixed response models for different items as follows:

```
proc irt;
   model x1-x10/resfunc=twop, x11-x20/resfunc=graded, x21-x30/resfunc=threep;
run;
```

For variables that are also listed in the VAR statement, the model that is specified here overwrites the default model or the model that is specified by using the RESFUNC= option in the PROC IRT statement.

You can use the EQUALITY statement to set equality constraints on these parameters.

VAR Statement

VAR | VARIABLE *variables* **;**

The VAR statement lists the analysis variables or items in the model. If you do a multiple-group analysis, the same set of analysis variables is assumed for all groups. By default, all variables that you specify in the VAR statement are fitted by the graded response model, which assumes that the analysis variables are ordinal and have 2 to 19 levels.

You can overwrite the default response model for the analysis variables in the VAR statement by using the RESFUNC= option in the PROC IRT statement. If you want to analyze the data by using a mixed type of response model, you can use the MODEL statement.

VARIANCE Statement

VARIANCE *assignment* < **,** *assignment . . .* > **;**

where *assignment* represents

var-list < **=***parameter-spec* >

The VARIANCE statement specifies the factor variance parameters in connection with the FACTOR statement. Notice that the VARIANCE statement is different from the VAR statement, which specifies variables for analysis. You can list factors only in the *var-list* of the VARIANCE statement.

In each *assignment* of the VARIANCE statement, you include the *var-list* whose variances you want to specify. Optionally, you can provide a list of parameter specifications (*parameter-spec*) after an equal sign for each *var-list*.

You can specify the following five types of the parameters for the variances of the latent factor in the VARIANCE statement:

- an unnamed free parameter
- an initial value
- a fixed value
- a free parameter with a name provided
- a free parameter with a name and initial value provided

Consider a confirmatory model that has the latent factors F1, F2, F3, F4, and F5.

The following VARIANCE statement illustrates the five types of parameter specifications in five *assignments*:

```
variance
   F1 ,
   F2 = (.5),
   F3 = 1.0,
   F4 = fvar,
   F5 = fvar(0.7);
```

In this statement, the variance of F1 is specified as an unnamed free parameter. The variance of F2 is an unnamed free parameter that has an initial value of 0.5. The variance of F3 is a fixed value of 1.0. This value stays the same during the estimation. The variance of F4 is a free parameter named fvar1. The variance of F5 is a free parameter named fvar2 that has an initial value of 0.7.

Mixed Parameter Lists

You can specify different types of parameters for the list of variances. For example, the following statement uses a list of parameters that have mixed types:

```
variance
   F1-F6 = vp1 vp2(2.0) vp3  4. (.3) vp6(.4);
```

This is equivalent to the following statement:

```
variance
   F1 = vp1
   F2 = vp2(2.0),
   F3 = vp3,
   F4 = 4. ,
   F5 = (.3),
   F6 = vp6(.4);
```

As you can see, an initial value that follows a parameter name is associated with the free parameter. For example, in the original mixed list specification, the specification **(2.0)** after vp2 is interpreted as the initial value of the parameter vp2, but not as the initial estimate of the variance of F3.

However, if you indeed want to specify that vp2 is a free parameter *without* an initial value and 2.0 is an initial estimate of the variance of F3 (while keeping all other things the same), you can use a null initial value specification for the parameter vp2, as shown in the following statement:

```
variance
   F1-F6 = vp1 vp2() (2.0)  4. (.3) vp6(.4);
```

This way, 2.0 becomes the initial estimate of the variance of F3. Because a, parameter list that contains mixed parameter types might be confusing, you can break down the specifications into separate *assignments* to remove ambiguities. For example, you can use the following equivalent statement:

```
variance
   F1 = vp1
   F2 = vp2,
   F3 = (2.),
   F4 = 4. ,
   F5 = (.3),
   F6 = vp6(.4);
```

Shorter and Longer Parameter Lists

If you provide fewer parameters than the number of variances in the *var-list*, all the remaining parameters are treated as unnamed free parameters. For example, the following statement assigns a fixed value of 1.0 to the variance of F1 while treating the other three variances as unnamed free parameters:

```
variance
   F1-F4 = 1.0;
```

This statement is equivalent to the following statement:

```
variance
   F1 = 1.0, F2-F4;
```

If you intend to fill up all values with the last parameter specification in the list, you can use the continuation syntax `[...]`, `[..]`, or `[.]`, as shown in the following example:

```
variance
   F1-F100 = 1.0 psi [...];
```

This means that the variance of F1 is fixed at 1.0 and that the variances of F1–F100 are all free parameters named psi. All variances except that for F1 are thus constrained to be equal by having the same parameter name.

However, you must be careful not to provide too many parameters. For example, the following statement results in an error:

```
variance
   F1-F6 = 1.0 psi2-psi6 extra;
```

The parameters after psi6 are excessive.

Default Variance Parameters

In the IRT model, by default, the factor variances are fixed at ones. You can use the VARIANCE statement to override these default variance parameters in situations where you want to specify parameter constraints, provide initial or fixed values, or make parameter references.

WEIGHT Statement

WEIGHT *variable* **;**

The WEIGHT statement specifies the weight variable for the observations. The WEIGHT and FREQ statements have a similar effect, except that the WEIGHT statement does not alter the number of observations. An observation is used in the analysis only if the WEIGHT variable is greater than 0 and is not missing.

Details: IRT Procedure

Notation for the Item Response Theory Model

This section introduces the mathematical notation that is used throughout the chapter to describe the item response theory (IRT) model. For a description of the fitting algorithms and the mathematical-statistical details, see the section "Details: IRT Procedure" on page 4826.

A d-dimensional graded response IRT model that has K ordinal responses can be expressed by the equations

$$y_{ij} = \boldsymbol{\lambda}_j \boldsymbol{\eta}_i + \epsilon_{ij}$$

$$p_{ijk} = \Pr(u_{ij} = k) = \Pr(\alpha_{(j,k-1)} < y_{ij} < \alpha_{(j,k)}), \quad k = 1, \ldots, K$$

where u_{ij} is the observed ordinal response from subject i for item j, y_{ij} is a continuous latent response that underlies u_{ij}, $\boldsymbol{\alpha}_j = (\alpha_{(j,0)} = -\infty, \alpha_{(j,1)}, \ldots, \alpha_{(j,K-1)}, \alpha_{(j,K)} = \infty)$ is a vector of threshold parameters for item j, $\boldsymbol{\lambda}_j$ is a vector of slope (or discrimination) parameters for item j, $\boldsymbol{\eta}_i = (\eta_{i1}, \ldots, \eta_{id})$ is a vector of latent factors for subject i, $\boldsymbol{\eta}_i \sim N_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and $\boldsymbol{\epsilon}_i = (\epsilon_{i1}, \ldots, \epsilon_{iJ})$ is a vector of unique factors for subject i. All the unique factors in $\boldsymbol{\epsilon}_i$ are independent from one another, suggesting that $y_{ij}, j = 1, \ldots, J$, are independent conditional on the latent factor $\boldsymbol{\eta}_i$. This is the so-called local independence assumption. Finally, $\boldsymbol{\eta}_i$ and $\boldsymbol{\epsilon}_i$ are also independent.

Based on the preceding model specification,

$$p_{ijk} = \int_{\alpha_{(j,k-1)}}^{\alpha_{(j,k)}} p(y; \lambda_j \eta_i, 1)\, dy = \int_{\alpha_{(j,k-1)} - \lambda_j \eta_i}^{\alpha_{(j,k)} - \lambda_j \eta_i} p(y; 0, 1)\, dy$$

where p is determined by the link function. It is the density function of the standard normal distribution if the probit link is used, or the density function of the logistic distribution if the logistic link is used.

Let $\boldsymbol{\Lambda} = (\boldsymbol{\lambda}_1^T, \ldots, \boldsymbol{\lambda}_J^T)$ denote the slope matrix. To identify the model in exploratory analysis, the upper triangular elements of $\boldsymbol{\Lambda}$ are fixed as zero, the factor mean $\boldsymbol{\mu}$ is fixed as a zero vector, and the factor variance covariance matrix $\boldsymbol{\Sigma}$ is fixed as an identity matrix. For confirmatory analysis, it is assumed that the identification problem is solved by user-specified constraints.

The model that is specified in the preceding equation uses the latent response formulation. PROC IRT uses this parameterization for computational convenience. When there is only one latent factor, a mathematically equivalent parameterization for the model is

$$p_{ijk} = \int_{-a_j(\eta_i - b_{j,k-1})}^{-a_j(\eta_i - b_{j,k})} p(y; 0, 1)\, dy$$

where a_j is called the slope (discrimination) parameter and $b_{j,k}, k = 1, \ldots, K$, are called the threshold parameters. The threshold parameters under these two parameterizations can be translated as $b_{j,k} = \frac{\alpha_{j,k}}{\lambda_j}$, where $k = 1, \ldots, K$ and $\gamma_{j,k} = -\alpha_{j,k}$ is often called the intercept parameter.

The preceding model is called a graded-response model. When the responses are binary, this model reduces to the two-parameter model, which can be expressed as

$$y_{ij} = a_j(\eta_i - b_j) + \epsilon_{ij}$$

$$p_{ij} = \Pr(u_{ij} = 1) = \Pr(y_{ij} > 0)$$

where b_j is often called the item difficulty parameter.

The two-parameter model reduces to a one-parameter model when slope parameters for all the items are constrained to be equal. In the case where the logistic link is used, the one- and two-parameter models are often abbreviated as 1PL and 2PL. When all the slope parameters are set to 1 and the factor variance is set to a free parameter, the Rasch model is obtained.

You can obtain three- and four-parameter models by introducing the guessing and ceiling parameters. Let g_j and c_j denote the item-specific guessing and ceiling parameters, respectively. Then the four-parameter model can be expressed as

$$p_{ij} = \Pr(u_{ij} = 1) = g_j + (c_j - g_j) \Pr(y_{ij} > 0)$$

This model reduces to the three-parameter model when $c_j = 1$.

The generalized partial credit (GPC) model is another popular IRT model for ordinal items besides the graded response model. Introduced by Muraki (1992), it is an extension of the partial credit (PC) model proposed by Masters (1982). In the PC model, the slope (or discrimination) parameter is fixed as 1 for all the items. The GPC model releases this assumption by introducing the slope parameter for each item. The GPC model can be formulated as

$$p_{ijk} = \frac{\exp(\sum_{h=1}^{k} a_j(\eta_i - b_{j,h}))}{\sum_{k=1}^{K} \exp(\sum_{h=1}^{k} a_j(\eta_i - b_{j,h}))}$$

In this formulation, a_j is called the slope (discrimination) parameter and $b_{j,h}$ is called the step parameter.

Assumptions

The primary statistical assumptions that underlie the analyses that PROC IRT performs are as follows:

- The number of latent factors is known.
- Latent factors are normally distributed.
- Conditional on latent factors, observed responses (items) are independent. This is the so-called local independence assumption.

These assumptions are necessary if you want to make statistical inferences. In exploratory analysis, these assumptions do not apply.

PROC IRT Contrasted with Other SAS Procedures

IRT models are often referred to as latent trait models, especially in the field of sociology. The term latent trait is used to emphasize that observed discrete responses are manifestations of hypothesized traits, constructs, or attributes that cannot be directly observed. For that reason, IRT models belong to the more general modeling framework called latent variable models. Other models that belong to the latent variable model framework include factor analysis models, finite mixture models, and mixed-effects models. The relationships between these different latent variable models can be described as shown in Table 65.2.

Table 65.2 Latent Variable Models

		Latent Variable	
		Continuous	**Discrete**
Observed Variable	**Continuous**	Factor analysis	Finite mixture model
	Discrete	Item response theory	Latent class analysis

This table suggests that latent variable models can be classified into four groups, based on the measurement scale of observed and latent variables. These different latent variable models can be fitted by different SAS procedures: PROC FACTOR for factor analysis models, PROC FMM for finite mixture models, and PROC IRT for item response theory models. IRT models are more closely related to factor analysis models. They can be considered a version of factor analysis models of discrete rather than continuous responses.

Response Models

PROC IRT supports several response models for binary and ordinal responses, and it allows different items to have different response models. Details about these response models and their relationships follow:

- One-parameter model: This model assumes that items are binary. The distinctive feature of the one-parameter model, compared with the two-parameter model, is that the slopes (or the discrimination parameters) of the items are the same in the model. Statistically, the one-parameter model is equivalent to the Rasch model. They give the same model fit for the same data set.

- Two-parameter model: This model assumes that items are binary. The slopes (or the discrimination parameters) and the difficulty (or the intercept parameters) of the items are free parameters in the model. If all slopes of the two-parameter model are constrained to be same, it reduces to the one-parameter model.

- Three-parameter model: This model assumes that items are binary. The slopes (or the discrimination parameters), the difficulty (or the intercept parameters), and the guessing parameters of the items are free parameters in the model. If all the guessing parameters are fixed to 0, the three-parameter model reduces to the two-parameter model.

- Four-parameter model: This model assumes that items are binary. The slopes (or the discrimination parameters), the difficulty (or the intercept parameters), the guessing parameters, and the ceiling parameters of the items are free parameters in the model. If all the guessing parameters are fixed to 0 and all the ceiling parameters are fixed to 1, the four-parameter model reduces to the two-parameter model.

- Rasch model: This model assumes that items are binary. The distinctive feature of the Rasch model, compared with the two-parameter model, is that the slopes (or the discrimination parameters) of the items are all fixed to 1 (and with free factor variance parameters) in the model. Statistically, the Rasch model is equivalent to the one-parameter model. They give the same model fit for the same data set.

- Graded response model: This model assumes that items are ordinal-categorical with at most 19 levels. The slopes (or discrimination) and the thresholds parameters of the items are free parameters in the model.

- Generalized partial credit model: This model assumes that items are ordinal-categorical with at most 19 levels. The slopes (or discrimination) and the step parameters of the items are free parameters in the model.

You can specify the response function or model for all the variables that are listed in the VAR statement by using the RESFUNC= option in the PROC IRT statement. To specify different response functions or models for different set of variables, you can use the MODEL statement.

Marginal Likelihood

Based on the model that is specified in the section "Notation for the Item Response Theory Model" on page 4826, the marginal likelihood is

$$L(\boldsymbol{\theta}|U) = \prod_{i=1}^{N} \int \prod_{j=1}^{J} \prod_{k=1}^{K} (P_{ijk})^{v_{ijk}} \phi(\boldsymbol{\eta}; \boldsymbol{\mu}, \boldsymbol{\Sigma})\, d\boldsymbol{\eta} = \prod_{i=1}^{N} \int f(u_i|\boldsymbol{\eta})\phi(\boldsymbol{\eta}; \boldsymbol{\mu}, \boldsymbol{\Sigma})\, d\boldsymbol{\eta}$$

where $v_{ijk} = I(u_{ij} = k)$, $\phi(\boldsymbol{\eta})$ is the multivariate normal density function for the latent factor $\boldsymbol{\eta}$, and $\boldsymbol{\theta}$ is a set of all the model parameters. The corresponding log likelihood is

$$\log L(\boldsymbol{\theta}|U) = \sum_{i=1}^{N} \log \int \prod_{j=1}^{J} \prod_{k=1}^{K} (P_{ijk})^{v_{ijk}} \phi(\boldsymbol{\eta}; \boldsymbol{\mu}, \boldsymbol{\Sigma})\, d\boldsymbol{\eta}$$

Integrations in the preceding equation cannot be solved analytically and need to be approximated by using numerical integration,

$$\log \tilde{L}(\boldsymbol{\theta}|U) = \sum_{i=1}^{N} \log \left[\sum_{g=1}^{G^d} \left[\prod_{j=1}^{J} \prod_{k=1}^{K} (P_{ijk}(\mathbf{x}_g))^{v_{ijk}} \frac{\phi(\mathbf{x}_g; \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\phi(\mathbf{x}_g; 0, I)} \right] w_g \right]$$

where d is the number of factors, G is the number of quadrature points per dimension, and $\mathbf{x}_g$ and w_g are the quadrature points and weights, respectively.

Approximating the Marginal Likelihood

As discussed in the section "Marginal Likelihood" on page 4830, integrations that are involved in the marginal likelihood for IRT model cannot be solved analytically and need to be approximated by using numerical integration, mostly Gauss-Hermite quadrature.

Gauss-Hermite (G-H) Quadrature

In general, the Gauss-Hermite (G-H) quadrature can be presented as

$$\int_{-\infty}^{\infty} g(x)\, dx = \int_{-\infty}^{\infty} f(x)\phi(x)\, dx \approx \sum_{g=1}^{G} f(x_g) w_g$$

where G is the number of quadrature points and x_g and w_g are the integration points and weights, respectively, which are uniquely determined by the integration domain and the weighting kernel $\phi(x)$. Traditional G-H quadrature often uses e^{-x^2} as the weighting kernel. In the field of statistics, the density of standard normal distribution is more widely used instead, because for estimating various statistical models, the Gaussian density is often a factor of the integrand. In the case in which the Gaussian density is not a factor of the integrand, the integral is transformed into the form by dividing and multiplying the original integrand by the standard normal density.

Adaptive Gauss-Hermite Quadrature

The G order G-H quadrature is exact if $f(x)$ is a $2K-1$ degree polynomial in x. However, as many researchers (Lesaffre and Spiessens 2001; Rabe-Hesketh, Skrondal, and Pickles 2002) point out, integrands $f(u_i|\boldsymbol{\eta})\phi(\boldsymbol{\eta};\boldsymbol{\mu},\boldsymbol{\Sigma})$ often have sharp peaks and cannot be well approximated by low-degree polynomials in $\boldsymbol{\eta}$. Furthermore, the peak might be far from zero or be located between adjacent quadrature points so that substantial contributions to the integral are lost.

Note that the integrands in the marginal likelihood are a product of the prior density of $\boldsymbol{\eta}$, $\phi(\boldsymbol{\eta};\boldsymbol{\mu},\boldsymbol{\Sigma})$ and the joint probability of responses given $\boldsymbol{\eta}$, $f(\mathbf{u}_i|\boldsymbol{\eta})$. After normalization with respect to $\boldsymbol{\eta}$, the integrand, $f(u_i|\boldsymbol{\eta})\phi(\boldsymbol{\eta};\boldsymbol{\mu},\boldsymbol{\Sigma})$, is just the posterior density of $\boldsymbol{\eta}$, given the observed responses $\mathbf{u}_i$. This posterior density is approximately normal when the number of items is large. Let $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ be the mean and covariance matrix, respectively, of the posterior density. Then the ratio $\frac{f(\mathbf{u}_i|\boldsymbol{\eta})\phi(\boldsymbol{\eta};\boldsymbol{\mu},\boldsymbol{\Sigma})}{\phi(\boldsymbol{\eta};\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i)}$ can be well approximated by a low-degree polynomial if the number of items is relatively large. This suggests that the integral should be transformed as

$$\int f(\mathbf{u}_i|\boldsymbol{\eta})\phi(\boldsymbol{\eta})\,d\boldsymbol{\eta} = \int \frac{f(\mathbf{u}_i|\boldsymbol{\eta})\phi(\boldsymbol{\eta};\boldsymbol{\mu},\boldsymbol{\Sigma})}{\phi(\boldsymbol{\eta};\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i)}\phi(\boldsymbol{\eta};\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i)\,d\boldsymbol{\eta}$$

The integration points and weights that correspond to $\phi(\boldsymbol{\eta};\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i)$ are

$$z_g = \boldsymbol{\Sigma}_i^{1/2}x_g + \boldsymbol{\mu}_i$$
$$v_g = |\boldsymbol{\Sigma}_i|^{1/2}w_g$$

The preceding transformations move and scale the quadrature points to the center of the integrands such that the integrand can be better approximated using many fewer quadrature points.

Maximizing the Marginal Likelihood

You can obtain parameter estimates by maximizing the marginal likelihood by using either the expectation maximization (EM) algorithm or a Newton-type algorithm. Both algorithms are available in PROC IRT.

The most widely used estimation method for IRT models is the Gauss-Hermite quadrature–based EM algorithm, proposed by Bock and Aitkin (1981). However, this method has several important shortcomings, the most serious of which is the lack of reliable convergence criteria. Without reliable convergence criteria, estimates can be seriously biased because of spurious convergence. In comparison, gradient-based convergence criteria is readily available for Newton-type algorithms. As a result, PROC IRT uses the quasi-Newton algorithm instead of EM as the default optimization method.

Newton-Type Algorithms

Newton-type algorithms maximize the marginal likelihood directly, based on the first and second derivatives. Two of the most widely used estimation algorithms are the Newton-Raphson and Fisher scoring algorithms, which rely on the gradient and Hessian of the log likelihood. However, for latent variable models that contain categorical responses, the Hessian matrix is often expensive to compute. As a result, several quasi-Newton algorithms that require only gradients have been proposed. In the field of IRT, Bock and Lieberman (1970) propose replacing the Hessian with the following information matrix:

$$I(\boldsymbol{\theta}) = E\left[\frac{\partial\log\tilde{L}(\theta|U)}{\partial\theta}\left(\frac{\partial\log\tilde{L}(\theta|U)}{\partial\theta}\right)^T\right] = \sum_{h=1}^{2^J}\left[\frac{\partial\log\tilde{L}_i}{\partial\theta}\left(\frac{\partial\log\tilde{L}_i}{\partial\theta}\right)^T\right]$$

To calculate the preceding expectation, you need to sum over not just the observed but all 2^J possible response patterns; this becomes computationally very expensive when the number of items is large. Fortunately, other quasi-Newton algorithms that do not have these computational difficulties have been proposed. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) method is one of the most popular quasi-Newton algorithms that approximate the Hessian matrix with gradient.

For the objective function, $\log \tilde{L}(\theta)$, the first derivatives with respect to θ_j are

$$\frac{\partial \log \tilde{L}(\boldsymbol{\theta}|\mathbf{U})}{\partial \theta_j} = \sum_{i=1}^{N} \left[(\tilde{L}_i)^{-1} \frac{\partial \tilde{L}_i}{\partial \theta_j} \right] = \sum_{i=1}^{N} \left[(\tilde{L}_i)^{-1} \sum_{g=1}^{G^d} \left[\frac{\partial f_i(\mathbf{x}_g)}{\partial \theta_j} w_g^* \right] \right]$$

where

$$\tilde{L}_i = \sum_{g=1}^{G^d} \left[\prod_{j=1}^{J} \prod_{k=1}^{K} (P_{ijk}(\mathbf{x}_g))^{v_{ijk}} \frac{\phi(\mathbf{x}_g; \boldsymbol{\mu}, \boldsymbol{\Sigma})}{d(\mathbf{x}_g; 0, I)} \right] w_g = \sum_{g=1}^{G^d} f_i(x_g) w_g$$

$$f_i(x_g) = \prod_{j=1}^{J} \prod_{k=1}^{K} (P_{ijk}(\mathbf{x}_g))^{v_{ijk}} \frac{\phi(\mathbf{x}_g; \boldsymbol{\mu}, \boldsymbol{\Sigma})}{d(\mathbf{x}_g; 0, I)}$$

and

$$\frac{\partial f_i(\mathbf{x}_g)}{\partial \theta_{Ij}} = \frac{\partial [(P_{ijk}(\mathbf{x}_g))]}{\partial \theta_j} \frac{f_i(\mathbf{x}_g)}{(P_{ijk}(\mathbf{x}_g))}$$

$$\frac{\partial f_i(\mathbf{x}_g)}{\partial \theta_f} = \prod_{j=1}^{J} \prod_{k=1}^{K} (P_{ijk}(\mathbf{x}_g))^{v_{ijk}} \frac{\partial \phi(\mathbf{x}_g; \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \theta_f}$$

where, in the preceding two equations, θ_{Ij} indicate parameters that are associated with item j and θ_f represents parameters that are related to latent factors.

Expectation-Maximization (EM) Algorithm

The expectation-maximization (EM) algorithm starts from the complete data log likelihood that can be expressed as follows:

$$\begin{aligned} \log L(\boldsymbol{\theta}|\mathbf{U}, \boldsymbol{\eta}) &= \sum_{i=1}^{N} \left[\sum_{j=1}^{J} \sum_{k=1}^{K} v_{ijk} \log P_{ijk} + \log \phi(\boldsymbol{\eta}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \right] \\ &= \sum_{j=1}^{J} \sum_{i=1}^{N} \left[\sum_{k=1}^{K} v_{ijk} \log P_{ijk} \right] + \sum_{i=1}^{N} \log \phi(\boldsymbol{\eta}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \end{aligned}$$

The expectation (E) step calculates the expectation of the complete data log likelihood with respect to the conditional distribution of $\boldsymbol{\eta}_i$, $f(\boldsymbol{\eta}_i|\mathbf{u}_i, \boldsymbol{\theta}^{(t)})$ as follows:

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \sum_{j=1}^{J} \sum_{i=1}^{N} \left[\sum_{k=1}^{K} v_{ijk} E\left[\log P_{ijk}|\mathbf{u}_i, \boldsymbol{\theta}^{(t)}\right] \right] + \sum_{i=1}^{N} E\left[\log \phi(\boldsymbol{\eta}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma})|\mathbf{u}_i, \boldsymbol{\theta}^{(t)}\right]$$

The conditional distribution $f(\boldsymbol{\eta}|u_i, \theta^{(t)})$ is

$$f(\boldsymbol{\eta}|\mathbf{u}_i, \boldsymbol{\theta}^{(t)}) = \frac{f(\mathbf{u}_i|\boldsymbol{\eta}, \boldsymbol{\theta}^{(t)})\phi(\boldsymbol{\eta}; \boldsymbol{\mu}^{(t)}, \boldsymbol{\Sigma}^{(t)})}{\int f(\mathbf{u}_i|\boldsymbol{\eta}, \boldsymbol{\theta}^{(t)})\phi(\boldsymbol{\eta}; \boldsymbol{\mu}^{(t)}, \boldsymbol{\Sigma}^{(t)})\, d\boldsymbol{\eta}} = \frac{f(\mathbf{u}_i|\boldsymbol{\eta}, \boldsymbol{\theta}^{(t)})\phi(\boldsymbol{\eta}; \boldsymbol{\mu}^{(t)}, \boldsymbol{\Sigma}^{(t)})}{f(\mathbf{u}_i)}$$

These conditional expectations that are involved in the Q function can be expressed as follows:

$$E[\log P_{ijk}|\mathbf{u}_i, \boldsymbol{\theta}^{(t)}] = \int \log P_{ijk} f(\boldsymbol{\eta}|\mathbf{u}_i, \boldsymbol{\theta}^{(t)})\, d\boldsymbol{\eta}$$

$$E[\log \phi(\boldsymbol{\eta}; \boldsymbol{\mu}, \boldsymbol{\Sigma})|\mathbf{u}_i, \boldsymbol{\theta}^{(t)}] = \int \log \phi(\boldsymbol{\eta}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) f(\boldsymbol{\eta}|\mathbf{u}_i, \boldsymbol{\theta}^{(t)})\, d\boldsymbol{\eta}$$

Then

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \sum_{j=1}^{J} \int \left[\log P_{ijk} r_{jk}(\boldsymbol{\theta}^{(t)})\right] \phi(\boldsymbol{\eta}; \boldsymbol{\mu}^{(t)}, \boldsymbol{\Sigma}^{(t)})\, d\boldsymbol{\eta} + \int \log \phi(\boldsymbol{\eta}|\boldsymbol{\theta}) N(\boldsymbol{\theta}^{(t)})\phi(\boldsymbol{\eta}; \boldsymbol{\mu}^{(t)}, \boldsymbol{\Sigma}^{(t)})\, d\boldsymbol{\eta}$$

where

$$r_{jk}(\boldsymbol{\theta}^{(t)}) = \sum_{i=1}^{N}\sum_{k=1}^{K} v_{ijk} \frac{f(\mathbf{u}_i|\eta, \boldsymbol{\theta}^{(t)})}{f(\mathbf{u}_i)}$$

and

$$N(\boldsymbol{\theta}^{(t)}) = \sum_{i=1}^{N} \frac{f(\mathbf{u}_i|\boldsymbol{\eta}, \boldsymbol{\theta}^{(t)})}{f(\mathbf{u}_i)}$$

Integrations in the preceding equations can be approximated as follows by using G-H quadrature:

$$\tilde{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \sum_{g=1}^{G} \left[\log P_{ijk}(\mathbf{x}_g) r_{jk}(\mathbf{x}_g, \boldsymbol{\theta}^{(t)}) + \log \phi(\mathbf{x}_g|\boldsymbol{\theta}) N(\mathbf{x}_g, \boldsymbol{\theta}^{(t)})\right] \frac{\phi(\mathbf{x}_g; \boldsymbol{\mu}^{(t)}, \boldsymbol{\Sigma}^{(t)})}{d(\mathbf{x}_g; 0, I)} w_g$$

In the maximization (M) step of the EM algorithm, parameters are updated by maximizing $\tilde{Q}(\theta|\theta^{(t)})$. To summarize, the EM algorithm consists of the following two steps:

E step: Approximate $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ by using numerical integration.

M step: Update parameter estimates by maximizing $\tilde{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ with the one-step Newton-Raphson algorithm.

Factor Score Estimation

PROC IRT provides three methods of estimating factor scores: maximum likelihood (ML), maximum a posteriori (MAP), and expected a posteriori (EAP). You can specify them by using the SCOREMETHOD= option in the PROC IRT statement.

You can obtain the ML factor score by maximizing the likelihood for each observation with respect to the latent factor. You can also compute the MAP or EAP factor score by maximizing or by taking the expectation of the posterior distribution of latent factors for each observation. The likelihood and posterior distribution for each observation, $u_i = (u_{i1}, \ldots, u_{iJ})$, can be expressed, respectively, as

$$l(\boldsymbol{\eta}|u_i, \hat{\boldsymbol{\theta}}) = \prod_{j=1}^{J} \prod_{k=1}^{K} (P_{ijk})^{v_{ijk}}$$

and

$$p(\boldsymbol{\eta}|u_i, \hat{\boldsymbol{\theta}}) \propto \prod_{j=1}^{J} \prod_{k=1}^{K} (P_{ijk})^{v_{ijk}} \phi(\boldsymbol{\eta}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Factor scores are restricted to the range from –99 to 99. For unidimensional models, the ML factor score is not available for subjects whose response to all the items is either the lowest or the highest level. For example, suppose there are five binary items in the model. For subjects whose response is 1 or 0 to all five items, the ML factor score cannot be estimated. For subjects whose response to all items is the lowest level, the ML factor score is set to –99, and for subjects whose response to all items is the highest level, the ML factor score is set to 99.

Model and Item Fit

The IRT procedure includes five model fit statistics: log likelihood, Akaike's information criterion (AIC), Bayesian information criterion (BIC), likelihood ratio chi-square G^2, and Pearson's chi-square.

The following two equations compute the likelihood ratio chi-square G^2 and Pearson's chi-square,

$$G^2 = 2\left[\sum_{l=1}^{L} r_l \log \frac{r_l}{NP_l}\right]$$

$$\chi^2 = \sum_{l=1}^{L} \frac{(r_l - NP_l)^2}{NP_l}$$

where N is the number of subjects, L is number of possible response patterns, P_l is the estimated probability of observing response pattern l, and r_l is the number of subjects who have response pattern l. If the model is true, these two statistics asymptotically follow central chi-square distribution with degrees of freedom $L - m - 1$, where m is the number of free parameters in the model. When L (the number of possible response patterns) is much greater than N, the frequency table is sparse. This invalidates the use of chi-square distribution as the asymptotic distribution for these two statistics, and as a result the likelihood ratio chi-square and Pearson's chi-square statistics should not be used to evaluate overall model fit.

For item fit, PROC IRT computes the likelihood ratio G^2 and Pearson's chi-square. Pearson's chi-square statistic, proposed by Yen (1981), has the form

$$Q_{1j} = \sum_{k=1}^{10} N_k \frac{(O_{jk} - E_{jk})^2}{E_{jk}(1 - E_{jk})}$$

The likelihood ratio G^2, proposed by McKinley and Mills (1985), uses the following equation:

$$G^2 = 2 \sum_{k=1}^{10} N_k \left[O_{jk} \log \frac{O_{jk}}{E_{ik}} + (1 - O_{jk}) \log \frac{1 - O_{jk}}{1 - E_{ik}} \right]$$

These two statistics approximately follow a central chi-square distribution with $10 - m_j$ degrees of freedom, where m_j is the number of free parameters for item *j*.

To calculate these two statistics, first order all the subjects according to their estimated factor scores, and then partition them into 10 intervals such that the number of subjects in each interval is approximately equal. O_{jk} and E_{jk} are the observed proportion and expected proportion, respectively, of subjects in interval *k* who have a correct response on item *j*. The expected proportions E_{jk} are computed as the mean predicted probability of a correct response in interval *k*.

Item and Test information

Let $P_{jk}(\theta)$ be the probability of endorsing category *k* for item *j* for a subject whose ability score is θ. Then the item information function can be defined as

$$I_j(\theta) = \sum_{k=1}^{K} I_k(\theta) P_{jk}(\theta)$$

where

$$I_k(\theta) = -\frac{\partial^2}{\partial \theta^2} \log P_{jk}(\theta)$$

The test information function is the sum of the information functions of the items in the test. The information function of a test that has *J* items is

$$I(\theta) = \sum_{j=1}^{J} I_j(\theta)$$

Missing Values

PROC IRT handles missing values differently for different tasks. During model estimation and subject scoring, observations with missing values are still used, in the sense that the nonmissing values can still contribute to the estimation or scoring. When calculating the polychoric correlation for a pair of variables, PROC IRT does pairwise deletion, in which observations that have valid data about the corresponding pair of variables are used. When calculating the item statistics table, PROC IRT uses listwise deletion, in which observations with missing values for any variables in the analysis are omitted from the computations.

Output Data Sets

OUTMODEL= Data Set

The OUTMODEL= data set contains the model specification, the computed parameter estimates, and the standard error estimates. This data set is intended to be reused as an INMODEL= data set in a subsequent analysis by PROC IRT.

The OUTMODEL= data set contains the following variables:

- BY variables, if any
- _GPNUM_ variable for group numbers, if used
- _TYPE_, a character variable that takes various values that indicate the type of model specification
- _SUBTYP_, a character variable that takes various values that indicate the type of model parameters
- _NAME_, a character variable that indicates the model type, parameter name, or variable name
- _NUM_, a numeric variable that takes various values that indicate model specifications such as the number of items, number of factors, number of groups, and number of levels for each item
- _VAR1_, a character variable that is the name or number of the first variable in the specification
- _VAR2_, a character variable that is the name or number of the second variable in the specification
- _ESTIM_, a numeric variable that is the final estimate of the parameter
- _STDERR_, a numeric variable that is the standard error estimate of the parameter

Each observation (record) of the OUTMODEL= data set contains a piece of information about the model specification. Depending on the type of specification that the value of the _TYPE_ variable indicates, the meanings of the _SUBTYP_, _NAME_, _NUM_ _VAR1_, and _VAR2_ variables differ. Table 65.3 summarizes the meanings of these variables for each value of the _TYPE_ variable.

Table 65.3 Meaning of Variables in the OUTMODEL=Data Set

TYPE	Description	_SUBTYP_	_NAME_	_NUM_	_VAR1_	_VAR2_
MODEL	Model Info		NGROUP	Number of groups		
MODEL	Model Info		NITEM	Number of items		
MODEL	Model Info		NFACTOR	Number of factors		
MODEL	Model Info		LINK		Link function	
MODEL	Model Info		RORDER		RORDER option	
MODEL	Model Info		DESC		DESC option	
VAR	Variable		Name	Number of levels	Response function	
FACTOR	Factor		Name			
GPVAR	Group		Group var name			
PARM	Parameter	CEILING	Name	Number	Variable	
PARM	Parameter	GUESSING	Name	Number	Variable	
PARM	Parameter	INTERCEPT	Name	Number	Variable	Category
PARM	Parameter	SLOPE	Name	Number	Variable	Factor
PARM	Parameter	COV	Name	Number	1st variable	2nd variable
PARM	Parameter	MEAN	Name	Number	Variable	

For computational convenience, the intercept parameters rather than the threshold (or difficulty) parameters are saved in the OUTMODEL= data set. For multidimensional exploratory analysis, the OUTMODEL= data set includes the unrotated factor loading matrix. In the OUTMODEL= data set, fixed parameters do not have names but only parameter numbers. For free parameters, the OUTMODEL= data set includes the parameter name and the parameter number. When this data set is used for the INMODEL= option, PROC IRT sets an equality constraint for parameters that have the same name.

ODS Table Names

PROC IRT assigns a name to each table that it creates. You can use these names to refer to the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 65.4. For more information about ODS, see Chapter 20, "Using the Output Delivery System."

Table 65.4 ODS Tables Produced by PROC IRT

ODS Table Name	Description	Option
ConvergenceStatus	Convergence status	Default output
Eigenvalues	Polychoric correlation–based eigenvalues	Default output
FactorCov	Factor covariance estimates	Default output; available only for multidimensional models
FactorCovInit	Initial factor covariance estimates	PINITIAL option; available only for multidimensional models

Table 65.4 *continued*

ODS Table Name	Description	Option
FactorCovRot	Rotated factor covariance estimates	ROTATE= oblique rotation methods; available only for multidimensional exploratory models
FitStatistics	Model fit statistics	Default output
GroupInfo	Group information	Default output; available only for multiple group analysis
ItemFit	Item fit statistics	ITEMFIT option; available only for binary responses that have one latent factor
ItemInfo	Item information	Default output
ItemStat	Classical item statistics	ITEMSTAT option
IterHistory	Iteration history	Default output
ModelInfo	Model information	Default output
OptInfo	Optimization information	Default output
ParameterEstimates	Item parameter estimates. Slope parameters are not included in this table for multidimensional models.	Default output
ParameterEstimatesInit	Initial item parameter estimates. Slope parameters are not included in this table for multidimensional models.	PINITIAL option
PolyCorr	Polychoric correlation matrix	POLYCHORIC option
Slope	Slope parameter estimates	Default output; available only for multidimensional confirmatory models
SlopeInit	Initial slope parameter estimates	PINITIAL option; available only for multidimensional models
SlopeRot	Rotated slope parameter estimates	Default output; available only for multidimensional exploratory models

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS."

Before you create graphs, ODS Graphics must be enabled (for example, by specifying the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" on page 609 in Chapter 21, "Statistical Graphics Using ODS."

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" on page 608 in Chapter 21, "Statistical Graphics Using ODS."

You must also specify the PLOTS= option in the PROC IRT statement.

PROC IRT assigns a name to each graph that it creates using ODS. You can use these names to refer to the graphs when using ODS. The names are listed in Table 65.5.

Table 65.5 Graphs Produced by PROC IRT

ODS Graph Name	Plot Description	PLOTS= Option
ItemCharCurve	Item characteristic curves	PLOTS=ICC. The plot panels by default; specify PLOTS=UNPACK to produce an individual plot for each item.
ItemInfoCurve	Item information curves	PLOTS=IIC. The plot panels by default; specify PLOTS=UNPACK to produce an individual plot for each item.
PolyCorrHeatMap	Heat map for polychoric correlation matrix	PLOTS=POLYCHORIC|PLCORR
ScreePlot	Scree and variance-explained plots	PLOTS=SCREE
TestInfoCurve	Test information curve	PLOTS=TIC
VariancePlot	Plot of explained variance	PLOTS=SCREE(UNPACK)

Examples: IRT Procedure

Example 65.1: Unidimensional IRT Models

This example shows you the features that PROC IRT provides for unidimensional analysis. The data set comes from the 1978 Quality of American Life Survey. The survey was administered to a sample of all US residents aged 18 years and older in 1978. In this survey, subjects were asked to rate their satisfaction with many different aspects of their lives. This example selects eight items. These items are designed to measure people's satisfaction in the following areas on a seven-point scale: community, neighborhood, dwelling unit, life in the United States, amount of education received, own health, job, and how spare time is spent. For illustration purposes, the first five items are dichotomized and the last three items are collapsed into three levels.

The following DATA step creates the data set IrtUni.

```
data IrtUni;
   input item1-item8 @@;
   datalines;
1 0 0 0 1 1 2 1 1 1 1 1 1 3 3 3 0 1 0 0 1 1 1 1 1 0 0 1 0 1 2 3 0 0 0
0 0 1 1 1 1 0 0 1 0 1 3 3 0 0 0 0 0 1 1 3 0 0 1 0 0 1 2 2 0 1 0 0 1 1

   ... more lines ...

3 3 0 1 0 0 1 2 2 1
;
```

Because all the items are designed to measure subjects' satisfaction in different aspects of their lives, it is reasonable to start with a unidimensional IRT model. The following statements fit such a model by using several user-specified options:

```
ods graphics on;
proc irt data=IrtUni link=probit pinitial itemstat polychoric
   itemfit plots=(icc polychoric);
   var item1-item8;
   model item1-item4/resfunc=twop, item5-item8/resfunc=graded;
run;
```

The ODS GRAPHICS ON statement invokes the ODS Graphics environment and displays the plots, such as the item characteristic curve plot. For more information about ODS Graphics, see Chapter 21, "Statistical Graphics Using ODS."

The first option is the LINK= option, which specifies that the link function be the probit link. Next, you request initial parameter estimates by using the PINITIAL option. Item fit statistics are displayed using the ITEMFIT option. In the PROC IRT statement, you can use the PLOTS option to request different plots. In this example, you request item characteristic curves by using the PLOTS=ICC option.

In this example, you use the MODEL statement to specify different response models for different items. The specifications in the MODEL statement suggest that the first four items, item1 to item4, are fitted using the two-parameter model, whereas the last four items, item5 to item8, are fitted using the graded response model.

Output 65.1.1 displays two tables. From the "Modeling Information" table, you can observe that the link function has changed from the default LOGIT link to the specified PROBIT link. The "Item Information" table shows that item1 to item5 each have two levels and item6 to item8 each have three levels. The last column shows the raw values of these different levels.

Output 65.1.1 Basic Information

The IRT Procedure

Modeling Information	
Data Set	WORK.IRTUNI
Link Function	Probit
Number of Items	8
Number of Factors	1
Number of Observations Read	500
Number of Observations Used	500
Estimation Method	Marginal Maximum Likelihood

Item Information

Response Model	Item	Levels	Values
TwoP	**item1**	2	0 1
	item2	2	0 1
	item3	2	0 1
	item4	2	0 1
Graded	**item5**	2	0 1
	item6	3	1 2 3
	item7	3	1 2 3
	item8	3	1 2 3

Output 65.1.2 displays the classical item statistics table, which include the item means, item-total correlations, adjusted item-total correlations, and item means for *i* ordered groups of observations or individuals. You can produce this table by specifying the ITEMSTAT option in the PROC IRT statement.

Output 65.1.2 Classical Item Statistics

The IRT Procedure

Item Statistics

		Item-Total Correlations		Means			
Item	**Mean**	**Unadjusted**	**Adjusted**	**G1 (N=132)**	**G2 (N=139)**	**G3 (N=119)**	**G4 (N=110)**
item1	0.42400	0.57595	0.43291	0.11364	0.26619	0.53782	0.87273
item2	0.34400	0.53837	0.39480	0.06818	0.19424	0.45378	0.74545
item3	0.38800	0.51335	0.36132	0.09091	0.30216	0.47899	0.75455
item4	0.41000	0.44559	0.28197	0.16667	0.30935	0.47899	0.75455
item5	0.63000	0.43591	0.27436	0.32576	0.64029	0.71429	0.89091
item6	1.82800	0.50955	0.24040	1.34848	1.66187	1.98319	2.44545
item7	2.04200	0.65163	0.41822	1.30303	1.97842	2.35294	2.67273
item8	2.18600	0.66119	0.43254	1.40909	2.15827	2.50420	2.80909

Total N=500, Cronbach Alpha=0.6482

PROC IRT produces the "Eigenvalues of the Polychoric Correlation Matrix" table in Output 65.1.3 by default. You can use these eigenvalues to assess the dimension of latent factors. For this example, the fact that only the first eigenvalue is greater than 1 suggests that a one-factor model for the items is reasonable.

Output 65.1.3 Eigenvalues of Polychoric Correlations

Eigenvalues of the Polychoric Correlation Matrix

	Eigenvalue	Difference	Proportion	Cumulative
1	3.11870486	2.12497677	0.3898	0.3898
2	0.99372809	0.10025986	0.1242	0.5141
3	0.89346823	0.03116998	0.1117	0.6257
4	0.86229826	0.10670185	0.1078	0.7335
5	0.75559640	0.17795713	0.0944	0.8280
6	0.57763928	0.10080017	0.0722	0.9002
7	0.47683911	0.15511333	0.0596	0.9598
8	0.32172578		0.0402	1.0000

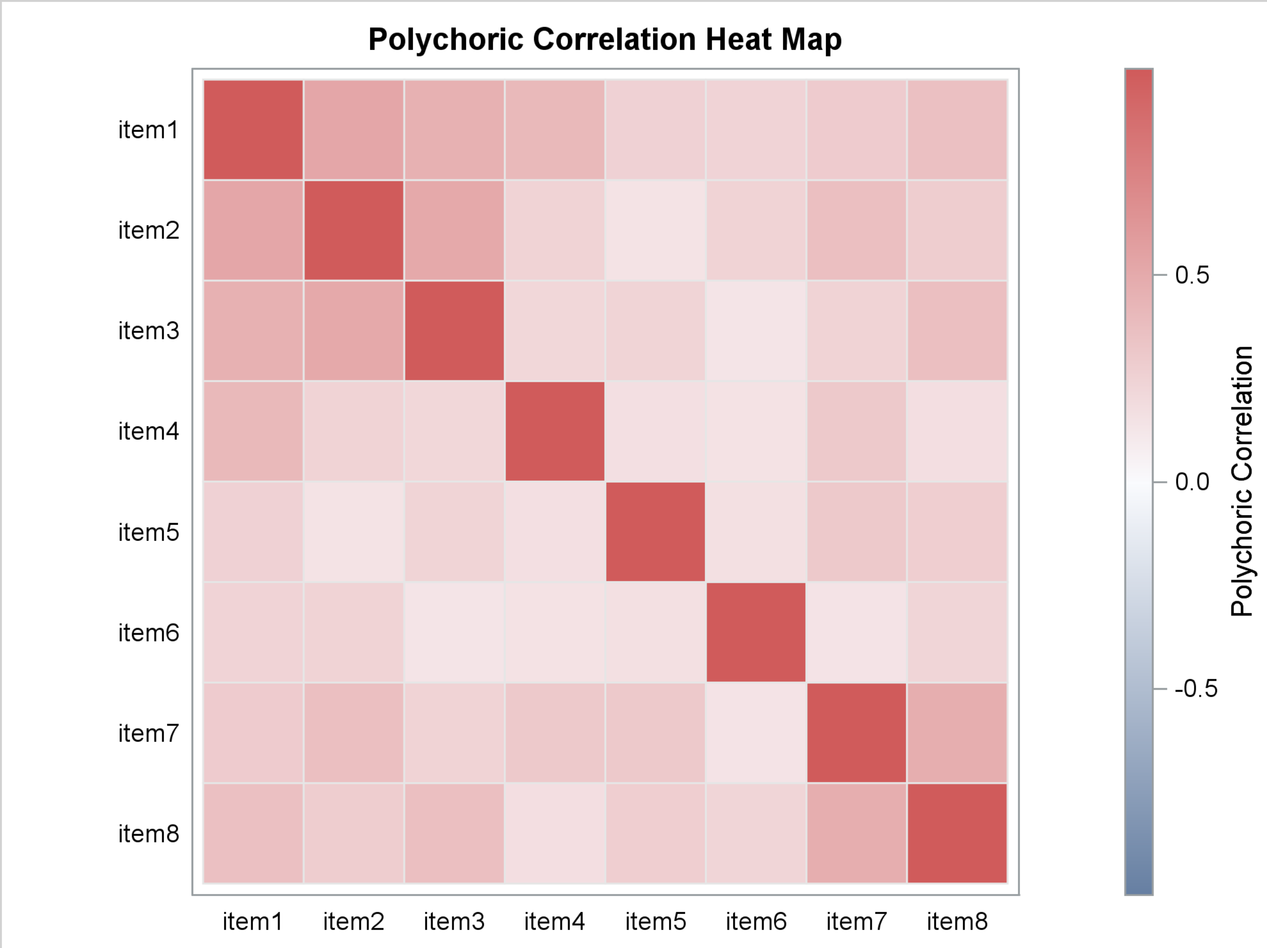
To get an overall idea of the correlations among all the items in the analysis, you can request the polychoric correlation matrix and the corresponding heat map. When you have a large number of items in the analysis, the heat map is especially useful to help you find patterns among these items. To produce the polychoric correlation matrix, specify the POLYCHORIC option in the PROC IRT statement. Specify PLOTS=POLYCHORIC to get the heat map for the polychoric correlation matrix. Output 65.1.4 includes the polychoric correlation table for this example, and Output 65.1.5 includes the heat map.

Output 65.1.4 Polychoric Correlation Matrix

Polychoric Correlation Matrix

	item1	item2	item3	item4	item5	item6	item7	item8
item1	1.0000	0.5333	0.4663	0.4181	0.2626	0.2512	0.3003	0.3723
item2	0.5333	1.0000	0.5183	0.2531	0.1543	0.2545	0.3757	0.2910
item3	0.4663	0.5183	1.0000	0.2308	0.2467	0.1455	0.2561	0.3771
item4	0.4181	0.2531	0.2308	1.0000	0.1755	0.1607	0.3181	0.1825
item5	0.2626	0.1543	0.2467	0.1755	1.0000	0.1725	0.3156	0.2846
item6	0.2512	0.2545	0.1455	0.1607	0.1725	1.0000	0.1513	0.2404
item7	0.3003	0.3757	0.2561	0.3181	0.3156	0.1513	1.0000	0.4856
item8	0.3723	0.2910	0.3771	0.1825	0.2846	0.2404	0.4856	1.0000

Output 65.1.5 Polychoric Correlation Heat Map



The PINITIAL option in the PROC IRT statement displays the "Initial Item Parameter Estimates" table, shown in Output 65.1.6.

Output 65.1.6 Initial Parameter Estimates

The IRT Procedure

Initial Item Parameter Estimates			
Response Model	**Item**	**Parameter**	**Estimate**
TwoP	**item1**	**Difficulty**	0.26428
		Slope	1.05346
	item2	**Difficulty**	0.58640
		Slope	0.93973
	item3	**Difficulty**	0.44607
		Slope	0.82826
	item4	**Difficulty**	0.50157
		Slope	0.50906
Graded	**item5**	**Threshold 1**	-0.86792
		Slope	0.41380
	item6	**Threshold 1**	-0.59512
		Threshold 2	2.00678
		Slope	0.36063
	item7	**Threshold 1**	-0.90743
		Threshold 2	0.69335
		Slope	0.64191
	item8	**Threshold 1**	-1.18209
		Threshold 2	0.26959
		Slope	0.67591

Output 65.1.7 includes tables that are related to the optimization. The "Optimization Information" table shows that the log likelihood is approximated by using seven adaptive Gauss-Hermite quadrature points and then maximized by using the quasi-Newton algorithm. The number of free parameters in this example is 19. The "Iteration History" table shows the number of function evaluations, the objective function (–log likelihood divided by number of subjects) values, the objective function change, and the maximum gradient for each iteration. This information is very useful in monitoring the optimization status. Output 65.1.7 shows the convergence status at the bottom. The optimization converges according to the GCONV=0.00000001 criterion.

Output 65.1.7 Optimization Information

The IRT Procedure

Optimization Information	
Optimization Technique	Quasi-Newton
Likelihood Approximation	Adaptive Gauss-Hermite Quadrature
Number of Quadrature Points	21
Number of Free Parameters	19

Output 65.1.7 *continued*

Iteration History					
Cycles	Iteration	Evaluations	Objective Function	Function Change	Max Abs Gradient
0	0	2	6.19423744		0.015499
0	1	5	6.19269765	-0.00153979	0.005785
0	2	8	6.19256563	-0.00013202	0.003812
0	3	10	6.19249848	-0.00006716	0.003284
0	4	12	6.19245354	-0.00004493	0.004647
0	5	15	6.19243615	-0.00001739	0.001284
0	6	18	6.19242917	-0.00000698	0.000491
0	7	21	6.19242859	-0.00000058	0.000192
0	8	24	6.19242845	-0.00000013	0.000104
0	9	27	6.19242842	-0.00000004	0.000051
0	10	30	6.19242841	-0.00000001	0.000011
1	0	2	6.19242841		0.000011
1	1	5	6.19242841	-0.00000000	6.365E-6

Convergence criterion (GCONV=.000000010) satisfied.

Output 65.1.8 displays the model fit and item fit statistics. Note that the item fit statistics apply only to the binary items. That is why these fit statistics are missing for item6 to item8.

Output 65.1.8 Fit Statistics

The IRT Procedure

Model Fit Statistics	
Log Likelihood	-3096.214205
AIC (Smaller is Better)	6230.4284092
BIC (Smaller is Better)	6310.505963
LR Chi-Square	825.73117916
LR Chi-Square DF	844

Item Fit Statistics						
Response Model	Item	DF	Pearson Chi-Square	Pr > P ChiSq	LR Chi-Square	Pr > LR ChiSq
TwoP	item1	8	34.16711	<.0001	49.39962	<.0001
	item2	8	30.34805	0.0002	37.53073	<.0001
	item3	8	27.54621	0.0006	36.34596	<.0001
	item4	8	22.76082	0.0037	26.13525	0.0010
Graded	item5	8	18.32269	0.0189	19.68379	0.0116
	item6	0	.	.	.	.
	item7	0	.	.	.	.
	item8	0	.	.	.	.

The last table for this example is the "Item Parameter Estimates " table in Output 65.1.9. This table contains parameter estimates, standard errors, and p-values. These p-values suggest that all the parameters are significantly different from zero.

Output 65.1.9 Parameter Estimates

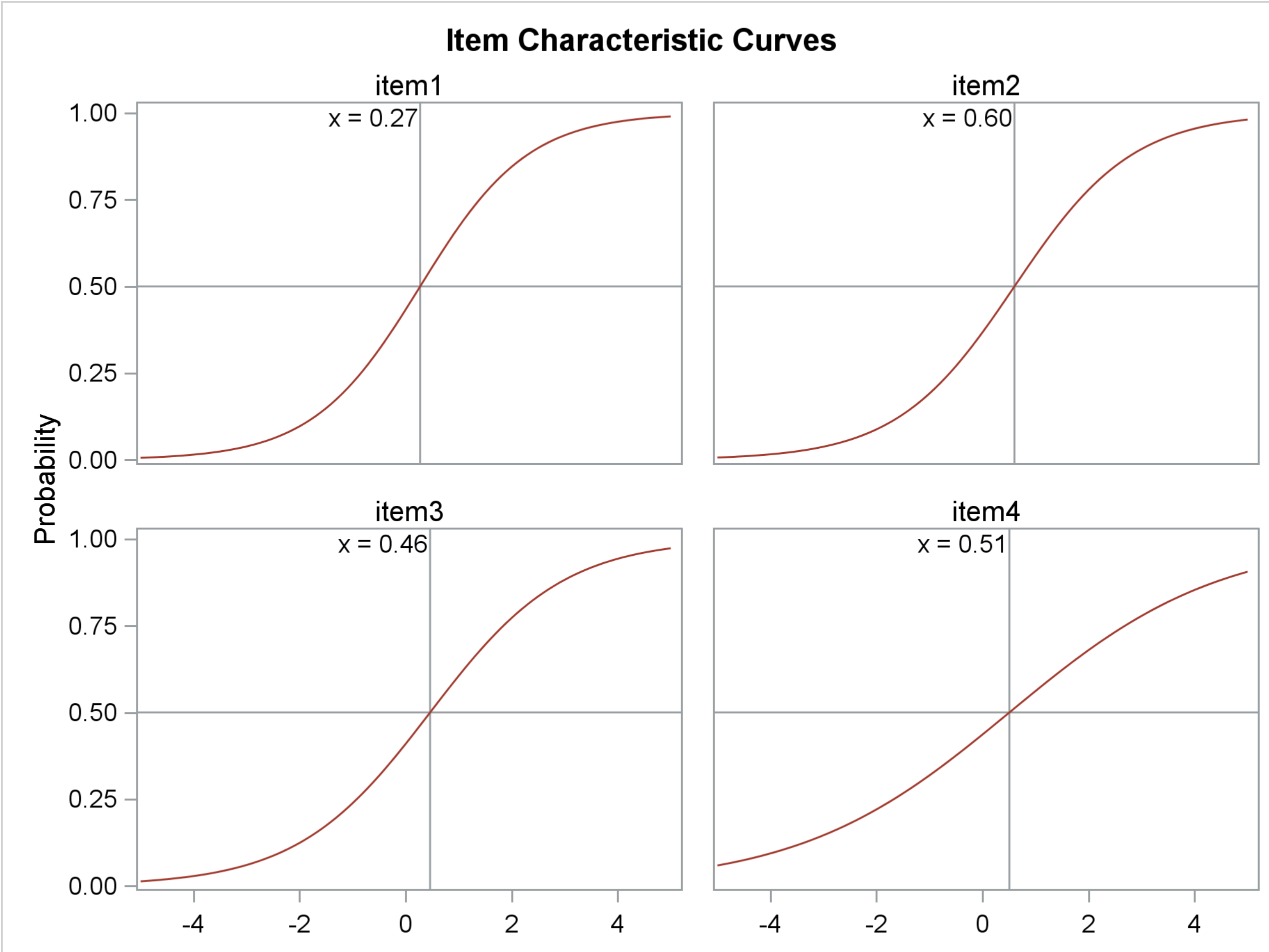
The IRT Procedure

Item Parameter Estimates

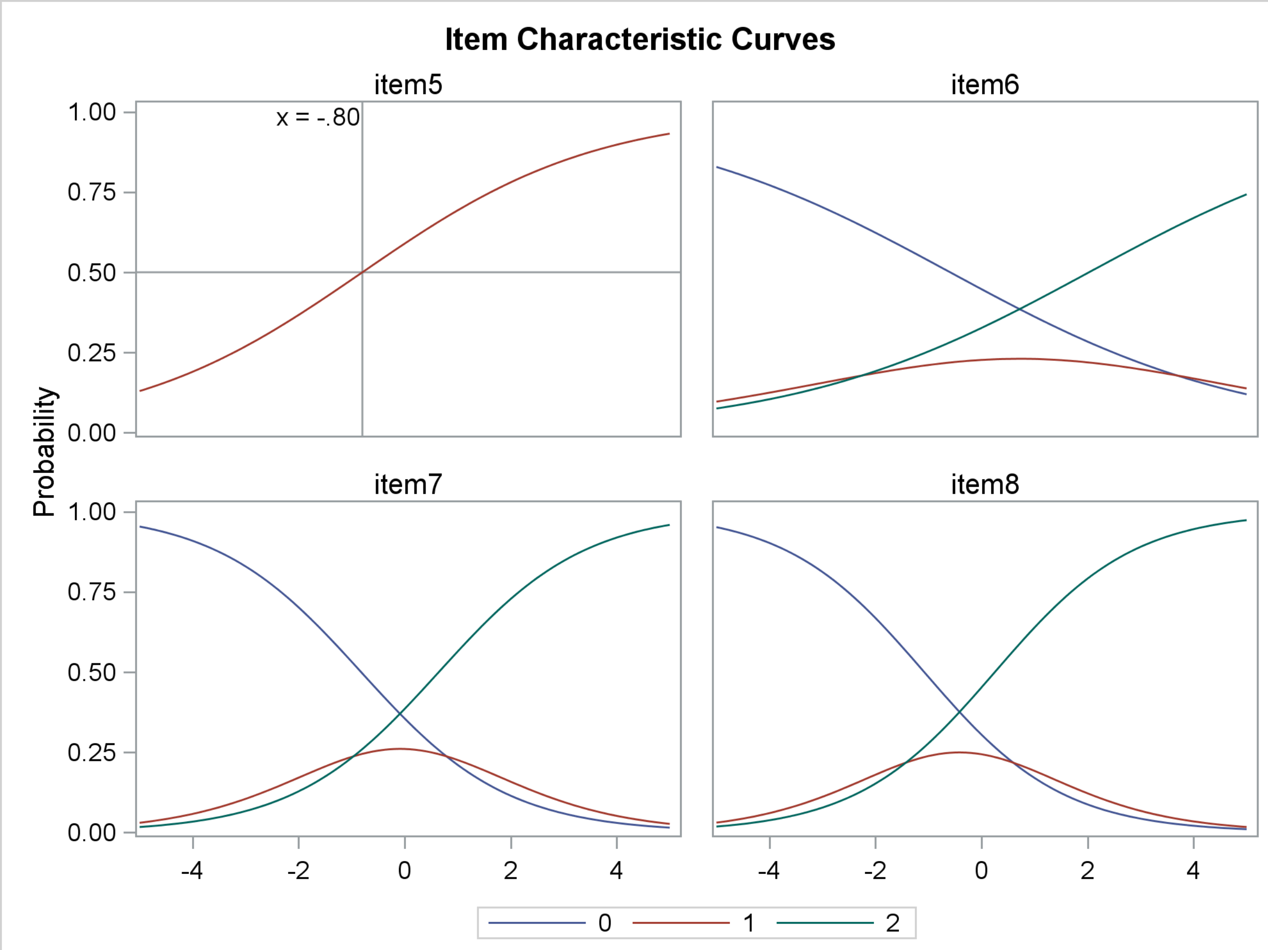
Response Model	Item	Parameter	Estimate	Standard Error	Pr > |t|
TwoP	item1	Difficulty	0.27339	0.08301	0.0005
		Slope	0.98378	0.14144	<.0001
	item2	Difficulty	0.60268	0.10047	<.0001
		Slope	0.90006	0.13111	<.0001
	item3	Difficulty	0.46111	0.10062	<.0001
		Slope	0.79520	0.11392	<.0001
	item4	Difficulty	0.50687	0.14411	0.0002
		Slope	0.50430	0.08567	<.0001
Graded	item5	Threshold	-0.79749	0.18707	<.0001
		Slope	0.45386	0.08238	<.0001
	item6	Threshold 1	-0.59135	0.19857	0.0015
		Threshold 2	2.02705	0.39594	<.0001
		Slope	0.35770	0.06777	<.0001
	item7	Threshold 1	-0.82132	0.12753	<.0001
		Threshold 2	0.64440	0.11431	<.0001
		Slope	0.72675	0.09313	<.0001
	item8	Threshold 1	-1.08126	0.14132	<.0001
		Threshold 2	0.25165	0.09535	0.0042
		Slope	0.76384	0.09754	<.0001

Item characteristic curves (ICC) are also produced in this example. By default, these ICC plots are displayed in panels. To display an individual ICC plot for each item, use the UNPACK suboption in the PLOTS= option in the PROC IRT statement.

Output 65.1.10 ICC Plots



Output 65.1.10 *continued*



Now, suppose your research hypothesis includes some equality constraints on the model parameters—for example, the slopes for the first four items are equal. Such equality constraints can be specified easily by using the EQUALITY statement. In the following example, the slope parameters of the first four items are equal:

```
proc irt data=IrtUni;
   var item1-item8;
   model item1-item4/resfunc=twop, item5-item8/resfunc=graded;
   equality item1-item4/parm=[slope];
run;
```

To estimate the factor score for each subject and add these scores to the original data set, you can use the OUT= option in the PROC IRT statement. PROC IRT provides three factor score estimation methods: maximum likelihood (ML), maximum a posteriori (MAP), and expected a posteriori (EAP). You can choose an estimation method by using the SCOREMETHOD= option in the PROC IRT statement. The default method is maximum a posteriori. In the following, factor scores along with the original data are saved to a SAS data set called IrtUniFscore:

```
proc irt data=IrtUni out=IrtUniFscore;
   var item1-item8;
   model item1-item4/resfunc=twop,
         item5-item8/resfunc=graded;
   equality item1-item4/parm=[slope];
run;
```

Sometimes you might find it useful to sort the items based on the estimated difficulty or slope parameters. You can do this by outputting the ODS tables for the estimates into data sets and then sorting the items by using PROC SORT. A simulated data set is used to show the steps.

The following DATA step creates the data set IrtSimu:

```
data IrtSimu;
   input item1-item25 @@;
   datalines;
1 1 1 0 1 1 0 0 1 1 0 0 0 0 1 0 1 1 1 0 0 0 1 0 1 1 1 1 1 1 0 1 1 0 0
0 0 0 0 0 1 1 0 0 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 1 0 0 0 1 0 1 1 0 1 1 0 0 1 1 0
1 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1 0 1 0 1 1

   ... more lines ...

1 1 0 1 1 1 1 1 1 1 0 1 1 0 0 0 1 1 0 1 1 1 1 1 1 0 0 1 0 1 0 1 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 1 1 1
;
```

First, you build the model and output the parameter estimates table into a SAS data set by using the ODS OUTPUT statement:

```
proc irt data=IrtSimu link=probit;
   var item1-item25;
   ods output ParameterEstimates=ParmEst;
run;
```

Output 65.1.11 shows the “Item Parameter Estimates” table. Notice that the difficulty and slope parameters are in the same column. The reason for this is to avoid having an extremely wide table when each item has a lot of parameters.

Output 65.1.11 Basic Information

The IRT Procedure

Item Parameter Estimates				
Item	**Parameter**	**Estimate**	**Standard Error**	**Pr > |t|**
item1	**Difficulty**	-1.32606	0.09788	<.0001
	Slope	1.44114	0.16076	<.0001
item2	**Difficulty**	-0.99731	0.07454	<.0001
	Slope	1.82041	0.18989	<.0001
item3	**Difficulty**	-1.25020	0.08981	<.0001
	Slope	1.58601	0.17477	<.0001
item4	**Difficulty**	-1.09617	0.07748	<.0001
	Slope	1.86641	0.20431	<.0001
item5	**Difficulty**	-1.07894	0.07806	<.0001
	Slope	1.78216	0.19062	<.0001
item6	**Difficulty**	-0.95086	0.09402	<.0001
	Slope	1.04073	0.10267	<.0001
item7	**Difficulty**	-0.65080	0.06949	<.0001
	Slope	1.45220	0.13450	<.0001
item8	**Difficulty**	-0.76378	0.07611	<.0001
	Slope	1.30280	0.12210	<.0001
item9	**Difficulty**	-0.72285	0.07058	<.0001
	Slope	1.50546	0.14220	<.0001
item10	**Difficulty**	-0.50731	0.06125	<.0001
	Slope	1.82144	0.17133	<.0001
item11	**Difficulty**	-0.01272	0.06470	0.4221
	Slope	1.26073	0.11260	<.0001
item12	**Difficulty**	0.04106	0.05584	0.2310
	Slope	2.01818	0.19994	<.0001
item13	**Difficulty**	0.16143	0.06878	0.0095
	Slope	1.12998	0.10180	<.0001
item14	**Difficulty**	0.01159	0.05670	0.4190
	Slope	1.88723	0.18049	<.0001
item15	**Difficulty**	0.07250	0.07360	0.1623
	Slope	0.96283	0.09036	<.0001
item16	**Difficulty**	-0.81425	0.07932	<.0001
	Slope	1.25217	0.11866	<.0001
item17	**Difficulty**	-0.92068	0.09314	<.0001
	Slope	1.02804	0.10108	<.0001
item18	**Difficulty**	-0.59398	0.06638	<.0001
	Slope	1.58229	0.14843	<.0001
item19	**Difficulty**	-0.97626	0.09768	<.0001
	Slope	0.97862	0.09745	<.0001
item20	**Difficulty**	-0.48838	0.05994	<.0001
	Slope	1.95459	0.18809	<.0001
item21	**Difficulty**	-0.60646	0.06851	<.0001
	Slope	1.45130	0.13402	<.0001
item22	**Difficulty**	-0.51245	0.06222	<.0001
	Slope	1.74241	0.16227	<.0001
item23	**Difficulty**	-0.90948	0.08476	<.0001

Output 65.1.11 *continued*

The IRT Procedure

Item Parameter Estimates

Item	Parameter	Estimate	Standard Error	Pr > |t|
	Slope	1.20134	0.11604	<.0001
item24	**Difficulty**	-0.56502	0.06327	<.0001
	Slope	1.74210	0.16361	<.0001
item25	**Difficulty**	-0.58894	0.06750	<.0001
	Slope	1.48756	0.13759	<.0001

Output 65.1.12 The Difficulty Parameter SAS Data Set

Obs	Item	Difficulty
1	item1	-1.32606
2	item2	-0.99731
3	item3	-1.25020
4	item4	-1.09617
5	item5	-1.07894
6	item6	-0.95086
7	item7	-0.65080
8	item8	-0.76378
9	item9	-0.72285
10	item10	-0.50731
11	item11	-0.01272
12	item12	0.04106
13	item13	0.16143
14	item14	0.01159
15	item15	0.07250
16	item16	-0.81425
17	item17	-0.92068
18	item18	-0.59398
19	item19	-0.97626
20	item20	-0.48838
21	item21	-0.60646
22	item22	-0.51245
23	item23	-0.90948
24	item24	-0.56502
25	item25	-0.58894

Then you save the estimates of slopes and difficulties in the data set ParmEst and create two separate data sets to store the difficulty and slope parameters:

```
data Diffs(keep=Item Difficulty);
   set ParmEst;
   Difficulty = Estimate;
   if (Parameter = "Difficulty") then output;
run;
proc print data=Diffs;
run;

data Slopes(keep=Item Slope);
   set ParmEst;
   Slope = Estimate;
   if (Parameter = "Slope") then output;
run;
proc print data=Slopes;
run;
```

The two SAS data sets are shown in Output 65.1.12 and Output 65.1.13.

Output 65.1.13 The Slope Parameter SAS Data Set

Obs	Item	Slope
1	item1	1.44114
2	item2	1.82041
3	item3	1.58601
4	item4	1.86641
5	item5	1.78216
6	item6	1.04073
7	item7	1.45220
8	item8	1.30280
9	item9	1.50546
10	item10	1.82144
11	item11	1.26073
12	item12	2.01818
13	item13	1.12998
14	item14	1.88723
15	item15	0.96283
16	item16	1.25217
17	item17	1.02804
18	item18	1.58229
19	item19	0.97862
20	item20	1.95459
21	item21	1.45130
22	item22	1.74241
23	item23	1.20134
24	item24	1.74210
25	item25	1.48756

Now you can use PROC SORT to sort the items by either difficulty or slope as follows:

```
proc sort data=Diffs;
   by Difficulty;
run;
proc print data=Diffs;
run;

proc sort data=Slopes;
   by Slope;
run;
proc print data=Slopes;
run;
```

Output 65.1.14 and Output 65.1.15 show the sorted data sets.

Output 65.1.14 Items Sorted by Difficulty

Obs	Item	Difficulty
1	item1	-1.32606
2	item3	-1.25020
3	item4	-1.09617
4	item5	-1.07894
5	item2	-0.99731
6	item19	-0.97626
7	item6	-0.95086
8	item17	-0.92068
9	item23	-0.90948
10	item16	-0.81425
11	item8	-0.76378
12	item9	-0.72285
13	item7	-0.65080
14	item21	-0.60646
15	item18	-0.59398
16	item25	-0.58894
17	item24	-0.56502
18	item22	-0.51245
19	item10	-0.50731
20	item20	-0.48838
21	item11	-0.01272
22	item14	0.01159
23	item12	0.04106
24	item15	0.07250
25	item13	0.16143

Output 65.1.15 Items Sorted by Slope

Obs	Item	Slope
1	item15	0.96283
2	item19	0.97862
3	item17	1.02804
4	item6	1.04073
5	item13	1.12998
6	item23	1.20134
7	item16	1.25217
8	item11	1.26073
9	item8	1.30280
10	item1	1.44114
11	item21	1.45130
12	item7	1.45220
13	item25	1.48756
14	item9	1.50546
15	item18	1.58229
16	item3	1.58601
17	item24	1.74210
18	item22	1.74241
19	item5	1.78216
20	item2	1.82041
21	item10	1.82144
22	item4	1.86641
23	item14	1.88723
24	item20	1.95459
25	item12	2.01818

Notice that the sorting does not work correctly if any of the items have more than one threshold (ordinal response) or slope (multidimensional model).

Now, suppose you want to group the items into subgroups based on their difficulty parameters and then sort the items in each subgroup by their slope parameters. First, you need to merge the two data sets, Diffs and Slopes, into one data set. Then, you add another variable, called DiffLevel, to indicate the subgroups. The following statements show these steps:

```
proc sort data=Slopes;
   by Item;
run;
proc sort data=Diffs;
   by Item;
run;
data ItemEst;
   merge Diffs Slopes;
   by Item;
   if Difficulty < -1.0 then DiffLevel = 1;
   else if Difficulty < 0 then DiffLevel = 2;
   else if Difficulty < 1 then DiffLevel = 3;
   else DiffLevel = 4;
run;
proc print data=ItemEst;
run;
```

Output 65.1.16 shows the merged data set.

Output 65.1.16 The Merged SAS Data Set

Obs	Item	Difficulty	Slope	DiffLevel
1	item1	-1.32606	1.44114	1
2	item10	-0.50731	1.82144	2
3	item11	-0.01272	1.26073	2
4	item12	0.04106	2.01818	3
5	item13	0.16143	1.12998	3
6	item14	0.01159	1.88723	3
7	item15	0.07250	0.96283	3
8	item16	-0.81425	1.25217	2
9	item17	-0.92068	1.02804	2
10	item18	-0.59398	1.58229	2
11	item19	-0.97626	0.97862	2
12	item2	-0.99731	1.82041	2
13	item20	-0.48838	1.95459	2
14	item21	-0.60646	1.45130	2
15	item22	-0.51245	1.74241	2
16	item23	-0.90948	1.20134	2
17	item24	-0.56502	1.74210	2
18	item25	-0.58894	1.48756	2
19	item3	-1.25020	1.58601	1
20	item4	-1.09617	1.86641	1
21	item5	-1.07894	1.78216	1
22	item6	-0.95086	1.04073	2
23	item7	-0.65080	1.45220	2
24	item8	-0.76378	1.30280	2
25	item9	-0.72285	1.50546	2

Then, you can sort the items by slope within each difficulty group as follows:

```
proc sort data=ItemEst;
   by difflevel slope;
run;
proc print data=ItemEst;
run;
```

Output 65.1.17 shows the data set after sorting.

Output 65.1.17 Item Sorted by Slope within Each Difficulty Group

Obs	Item	Difficulty	Slope	DiffLevel
1	item1	-1.32606	1.44114	1
2	item3	-1.25020	1.58601	1
3	item5	-1.07894	1.78216	1
4	item4	-1.09617	1.86641	1
5	item19	-0.97626	0.97862	2
6	item17	-0.92068	1.02804	2
7	item6	-0.95086	1.04073	2
8	item23	-0.90948	1.20134	2
9	item16	-0.81425	1.25217	2
10	item11	-0.01272	1.26073	2
11	item8	-0.76378	1.30280	2
12	item21	-0.60646	1.45130	2
13	item7	-0.65080	1.45220	2
14	item25	-0.58894	1.48756	2
15	item9	-0.72285	1.50546	2
16	item18	-0.59398	1.58229	2
17	item24	-0.56502	1.74210	2
18	item22	-0.51245	1.74241	2
19	item2	-0.99731	1.82041	2
20	item10	-0.50731	1.82144	2
21	item20	-0.48838	1.95459	2
22	item15	0.07250	0.96283	3
23	item13	0.16143	1.12998	3
24	item14	0.01159	1.88723	3
25	item12	0.04106	2.01818	3

Example 65.2: Multidimensional Exploratory and Confirmatory IRT Models

This example illustrates how to use the IRT procedure to fit multidimensional exploratory and confirmatory IRT models. The data set that is introduced in Example 65.1 is also used here. Two more items, item9 and item10, are added to the data set. These two items are designed to measure subjects' satisfaction with their friendships and their family life, respectively.

```
data IrtMulti;
   input item1-item10 @@;
   datalines;
1 0 0 0 1 1 2 1 2 1 1 1 1 1 1 3 3 3 3 3 0 1 0 0 1 1 1 1 1 1 0 0 1 0
1 2 3 2 2 0 0 0 0 0 1 1 1 1 1 1 0 0 1 0 1 3 3 1 2 0 0 0 0 0 1 1 3 3 2
0 0 1 0 0 1 2 2 3 2 0 1 0 0 1 1 1 2 2 2 0 0 0 0 0 2 2 3 3 2 0 1 0 1 0
2 3 3 3 3 0 0 1 0 1 1 2 3 2 3 1 1 1 1 1 2 2 3 2 2 0 0 0 0 1 1 2 2 3 1
1 0 1 1 1 2 3 3 2 3 0 1 0 0 1 1 2 3 3 3 1 0 1 1 1 2 3 3 3 3 0 1 0 1 1
3 2 3 3 2 1 1 1 0 0 1 3 3 2 1 1 1 0 0 1 2 3 3 3 3 0 1 1 1 1 1 2 1 2 3
1 0 0 1 1 3 1 1 1 1 1 0 0 0 1 1 1 3 3 1 0 0 0 0 1 1 3 3 3 3 0 0 0 0 0
1 1 1 3 2 1 0 0 0 0 1 3 3 3 3 1 1 0 1 1 3 1 1 3 3 1 0 1 1 1 1 3 1 1 1

   ... more lines ...

3 3 1 3 2 0 0 0 1 0 1 3 2 2 1 0 0 0 0 1 1 2 2 2 3 1 0 1 0 1 2 2 3 2 1
1 0 0 1 1 1 2 2 3 1 0 1 0 1 0 1 3 1 1 1 0 1 1 0 1 3 3 3 3 2 1 0 1 0 0
1 2 1 1 1 1 0 1 1 0 1 3 3 1 3 1 1 0 1 0 2 2 2 2 3 1 1 0 1 1 3 2 3 2 2
0 0 0 1 0 2 2 3 1 2 0 0 0 1 0 2 3 3 3 2 0 1 0 0 1 2 2 1 2 1
;
```

Now, suppose that previous research results suggest that two latent factors underlie these 10 items. However, knowledge about the factor structure is very limited. The first step you can take is to fit an exploratory IRT model by using two factors. This can be accomplished easily by submitting the following statements:

```
ods graphics on;
proc irt data=IrtMulti nfactor=2 plots=scree;
   var item1-item10;
run;
```

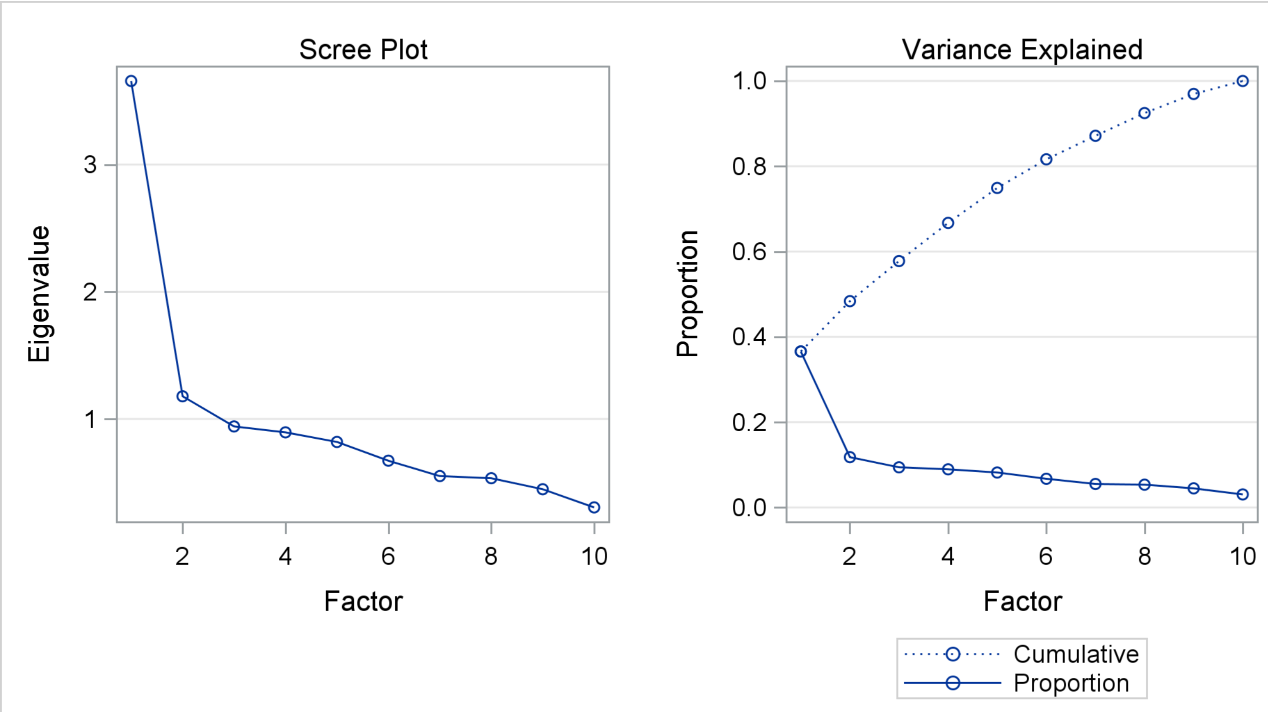
The first table that you want to check is the "Eigenvalue" table, shown in Output 65.2.1. There are only two eigenvalues greater than 1 in this example. This result, to some extent, suggests that two factors might be enough in this example. Output 65.2.2 include the scree and variance explained plots.

Output 65.2.1 Eigenvalues of the Polychoric Correlation matrix

The IRT Procedure

Eigenvalues of the Polychoric Correlation Matrix				
	Eigenvalue	Difference	Proportion	Cumulative
1	3.65750431	2.47883117	0.3658	0.3658
2	1.17867314	0.23738137	0.1179	0.4836
3	0.94129177	0.04672399	0.0941	0.5777
4	0.89456778	0.07508308	0.0895	0.6672
5	0.81948471	0.14774320	0.0819	0.7492
6	0.67174151	0.12081203	0.0672	0.8163
7	0.55092948	0.01698300	0.0551	0.8714
8	0.53394648	0.08647230	0.0534	0.9248
9	0.44747418	0.14308753	0.0447	0.9696
10	0.30438664		0.0304	1.0000

Output 65.2.2 Scree and Variance Explained Plots



If the optimization algorithm converges successfully, the original and rotated slope matrices are produced. The default rotation method is varimax. You can use the ROTATE= option in the PROC IRT statement to specify a different rotation method.

Output 65.2.3 Slope Matrix

Rotated Slope Matrix		
	_Factor1	_Factor2
item1	1.87508	0.63440
item2	1.71856	0.56393
item3	1.29605	0.54177
item4	0.74922	0.36337
item5	0.41024	0.63822
item6	0.44033	0.39976
item7	0.68677	1.18570
item8	0.74078	1.97550
item9	0.40199	1.26327
item10	0.40072	1.26552

This example uses the default varimax rotation. Output 65.2.3 shows the rotated slope matrices. The rotated slope matrix is displayed in the standard matrix format. From the rotated slope matrix, you can see that the first factor is mainly reflected by item1 to item4 and item6, and the second factor is mainly reflected by the rest of the items. The exploratory results suggest a hypothesis about the factor structure of the items. In practice, you might want to confirm this structure by a confirmatory analysis of the new data. However, for illustration purposes, the same data set is used here to demonstrate the confirmatory model fitting by using the following statements:

```
proc irt data=IrtMulti;
   var item1-item4 item6 item5 item7-item10;
   factor  Factor1->item1-item4 item6,
           Factor2->item5 item7-item10;
run;
```

Output 65.2.5 and Output 65.2.4 show the model fit statistics for the confirmatory model and the exploratory model, respectively. Output 65.2.6 shows the slope matrix from this confirmatory analysis. Because the number of response patterns in the data set, 500, is much lower than the total number of possible response patterns, $2^5 \times 3^5 = 7,776$, you cannot use Pearson's chi-square or the likelihood ratio statistic to test the overall fit of the confirmatory model.

If you have two nested models, you can still use the likelihood ratio statistic to test the difference between these two models. For illustration purposes, assume that the exploratory model and the confirmatory model are two independent models. Because the confirmatory model is nested within the exploratory model, you can use the likelihood ratio test to compare the two models. The likelihood ratio test statistic is 132.4. The degree of freedom is 9. The corresponding p-value is 0, which suggests that the difference between these two models is significant. There are many options that you can use to improve the model fit. For this example, you can try to free some of the fixed slope parameters or free the covariance between the factors.

Output 65.2.4 Model Fit Statistics for Exploratory Model

The IRT Procedure

Model Fit Statistics	
Log Likelihood	-3921.126181
AIC (Smaller is Better)	7910.2523628
BIC (Smaller is Better)	8053.5490382
LR Chi-Square	1897.9570322
LR Chi-Square DF	7741

Output 65.2.5 Model Fit Statistics for Confirmatory Model

The IRT Procedure

Model Fit Statistics	
Log Likelihood	-4126.853269
AIC (Smaller is Better)	8303.7065383
BIC (Smaller is Better)	8409.0717408
LR Chi-Square	2309.4112078
LR Chi-Square DF	7750

Output 65.2.6 Slope Matrix

Slope Matrix Estimate/StdErr/p-value		
	Factor1	Factor2
item1	2.03378 0.41059 <.00001	0.00000
item2	1.77557 0.32656 <.00001	0.00000
item3	1.35910 0.22796 <.00001	0.00000
item4	0.80427 0.15627 <.00001	0.00000
item6	0.55166 0.12128 <.00001	0.00000
item5	0.00000	0.73738 0.13901 <.00001
item7	0.00000	1.33303 0.17339 <.00001
item8	0.00000	2.14135 0.31847 <.00001
item9	0.00000	1.29833 0.16676 <.00001
item10	0.00000	1.30784 0.17014 <.00001

Example 65.3: Multiple-Group Analysis

This example shows how to use the IRT procedure to do multiple-group analysis. The following DATA step creates the data set IrtGroup:

```
data IrtGroup;
   input item1-item8 GroupVar @@;
   datalines;
1 0 0 0 1 1 2 1 2 1 1 1 1 1 3 3 3 1 0 1 0 0 1 1 1 1 1 1 0 0 1 0 1 2 3
2 0 0 0 0 0 1 1 1 1 1 0 0 1 0 1 3 3 1 0 0 0 0 0 1 1 3 2 0 0 1 0 0 1 2
2 1 0 1 0 0 1 1 1 2 1 0 0 0 0 0 2 2 3 1 0 1 0 1 0 2 3 3 2 0 0 1 0 1 1
2 3 1 1 1 1 1 1 2 2 3 2 0 0 0 0 1 1 2 2 1 1 0 1 1 1 2 3 3 2 0 1 0 0 1
1 2 3 2 1 0 1 1 1 2 3 3 2 0 1 0 1 1 3 2 3 2 1 1 1 0 0 1 3 3 1 1 1 0 0
1 2 3 3 2 0 1 1 1 1 1 2 1 2 1 0 0 1 1 3 1 1 2 1 0 0 0 1 1 1 3 2 0 0 0
0 1 1 3 3 1 0 0 0 0 0 1 1 1 2 1 0 0 0 0 1 3 3 2 1 1 0 1 1 3 1 1 1 1 0
1 1 1 1 3 1 2 1 1 1 1 1 2 3 2 2 0 0 1 0 0 2 2 2 1 0 0 1 0 1 1 2 3 2 1

   ... more lines ...
```

```
1 0 0 2 1 3 2 1 1 1 1 1 1 3 2 1 1 1 1 0 0 3 3 1 2 1 0 0 1 1 3 3 3 2 0
0 1 0 0 1 1 1 2 0 0 0 0 1 3 1 1 2 1 0 0 1 0 1 3 3 2 0 0 1 1 0 2 2 3 2
1 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 1 2 0 0 1 1 0 1 1 3 2 0 0 0 0 1 1 1 1
2 0 0 1 0 0 1 1 1 1
;
```

To set up a multiple-group IRT model, you need to specify the grouping variable by using the GROUP statement. Very often you also want to specify cross-group equality constraints for different parameters. You can accomplish this by using the EQUALITY statement.

The model that is specified in the following statements is an extension of the model in Example 65.1. The group variable, GroupVar, is specified in the GROUP statement. It has two values, 1 and 2, to indicate group membership. Equality constraints are specified in the EQUALITY statement.

```
proc irt data=IrtGroup;
   var item1-item8;
   group GroupVar;
   model item1-item4/resfunc=twop,
         item5-item8/resfunc=graded;
   equality item1-item4/parm=[intercept] between_gp=[1 2],
            _allgr_/parm=[slope] within_gp=[1];
run;
```

Two different sets of equality constraints have been specified. The first entry specifies equality constraints on the intercept parameters for item1 to item4 between group 1 and group 2:

```
item1-item4/parm=[intercept] between_gp=[1 2]
```

Notice that 1 and 2 are the actual values for the group variable GroupVar. The second entry specifies equality constraints on the slope parameters for all the graded response items within group 1:

```
_allgr_/parm=[slope] within_gp=[1]
```

Output 65.3.1 shows the “Modeling Information” table and the “Group Information” table for this example. For multiple-group analysis, the “Modeling Information” table contains two extra pieces of information: the group variable and the number of groups. The “Group Information” table contains information about the data for each group. There are 272 observations that have been read and used for group 1; this number for group 2 is 328.

Output 65.3.1 Modeling and Group Information

The IRT Procedure

Modeling Information	
Data Set	WORK.IRTGROUP
Group Variable	GroupVar
Link Function	Logit
Number of Items	8
Number of Factors	1
Number of Groups	2
Number of Observations Read	600
Number of Observations Used	600
Estimation Method	Marginal Maximum Likelihood

Output 65.3.1 *continued*

Group Information

GroupVar	Nobs Read	Nobs Used
1	272	272
2	328	328

Because there are two groups in this example, the IRT procedure produces two "Item Information" tables. For this example, these two tables contain the same information. That means that all the items have the same levels for the two groups. It is possible that the same item might have different numbers of levels, or maybe the same number of levels but different values. For example, an item has four levels, from 1 to 4, but one group might observe only levels 1 and 2, and the other group might observe only levels 3 and 4.

Output 65.3.2 Item Information

The IRT Procedure

Item Information
GroupVar = 1

Response Model	Item	Levels	Values
TwoP	item1	2	0 1
	item2	2	0 1
	item3	2	0 1
	item4	2	0 1
Graded	item5	2	0 1
	item6	3	1 2 3
	item7	3	1 2 3
	item8	3	1 2 3

Item Information
GroupVar = 2

Response Model	Item	Levels	Values
TwoP	item1	2	0 1
	item2	2	0 1
	item3	2	0 1
	item4	2	0 1
Graded	item5	2	0 1
	item6	3	1 2 3
	item7	3	1 2 3
	item8	3	1 2 3

Output 65.3.3 includes "Item Parameter Estimates" tables for both groups. You can see that the intercept parameters for item1 are the same for both groups. The same applies to item2 to item4. You can also see that the slope parameters have the same value for item5 to item8 in group 1. These results suggest that equality constraints that are specified in the EQUALITY statement have been fulfilled.

Output 65.3.3 Parameter Estimates

The IRT Procedure

Item Parameter Estimates
GroupVar = 1

Response Model	Item	Parameter	Estimate	Standard Error	Pr > |t|
TwoP	item1	Difficulty	0.24626	0.07816	0.0008
		Slope	1.89140	0.37441	<.0001
	item2	Difficulty	0.49593	0.10121	<.0001
		Slope	2.00194	0.37933	<.0001
	item3	Difficulty	0.41604	0.10697	<.0001
		Slope	1.40286	0.26914	<.0001
	item4	Difficulty	0.57879	0.16207	0.0002
		Slope	0.90518	0.20196	<.0001
Graded	item5	Threshold	-0.71305	0.16161	<.0001
		Slope	0.96028	0.10851	<.0001
	item6	Threshold 1	-0.15736	0.14693	0.1421
		Threshold 2	1.42604	0.21479	<.0001
		Slope	0.96028	0.10851	<.0001
	item7	Threshold 1	-0.86488	0.16877	<.0001
		Threshold 2	0.81548	0.17161	<.0001
		Slope	0.96028	0.10851	<.0001
	item8	Threshold 1	-1.24146	0.19173	<.0001
		Threshold 2	0.37687	0.15211	0.0066
		Slope	0.96028	0.10851	<.0001

Item Parameter Estimates
GroupVar = 2

Response Model	Item	Parameter	Estimate	Standard Error	Pr > |t|
TwoP	item1	Difficulty	0.32086	0.09339	0.0003
		Slope	1.45165	0.28558	<.0001
	item2	Difficulty	0.68892	0.12795	<.0001
		Slope	1.44111	0.28559	<.0001
	item3	Difficulty	0.51533	0.12286	<.0001
		Slope	1.13255	0.22767	<.0001
	item4	Difficulty	0.61813	0.16239	<.0001
		Slope	0.84758	0.18933	<.0001
Graded	item5	Threshold	-0.90713	0.27389	0.0005
		Slope	0.68620	0.17589	<.0001
	item6	Threshold 1	-0.77963	0.29946	0.0046
		Threshold 2	2.21438	0.57523	<.0001
		Slope	0.53993	0.14313	<.0001
	item7	Threshold 1	-0.84385	0.18252	<.0001
		Threshold 2	0.80750	0.16759	<.0001
		Slope	1.03837	0.18991	<.0001
	item8	Threshold 1	-1.18593	0.20728	<.0001
		Threshold 2	0.20024	0.12032	0.0480
		Slope	1.15474	0.21093	<.0001

Example 65.4: Item Selection Using Item and Test Information

The data set in this example comes from the 1978 Quality of American Life Survey. The survey was administered to a sample of US residents aged 18 years and older in 1978. Subjects were asked to rate their satisfaction with many different aspects of their lives. This example includes 14 items. Some of the items are as follows:

- satisfaction with community
- satisfaction with neighbors
- satisfaction with amount of education received
- satisfaction with health
- satisfaction with job
- satisfaction with income

Originally these items were designed with seven-point scales, where 1 indicates most unsatisfied and 7 indicates most satisfied. For illustration purposes, these items have been reorganized into a different number of categories, which ranges from 2 to 7. This example uses 1,000 random samples from the original data set. The following DATA step creates the data set IrtQls:

```
data IrtQls;
   input item1-item14 @@;
   datalines;
 1   1   2   1   1    2    2    2    .    2    2    2    2    2
 2   2   2   2   2    3    4    1    .    2    5    6    4    4

   ... more lines ...

 1   1   1   1   2    2    2    2    .    1    1    1    1    3
;
```

By default, the IRT procedure uses the graded response model (GRM) and the logistic link for all the ordinal items and uses the two-parameter logistic model for all the binary items. In PROC IRT, you can specify different types of response models for different items by using the MODEL statement.

Because all the items in this example are designed to measure subjects' satisfaction with their lives, it is reasonable to start with a unidimensional IRT model. The following statements fit such a model by using the default model options:

```
ods graphics on;
proc irt data=IrtQls plots=(IIC TIC);
   var item1-item14;
run;
```

This example requests item information curves (IICs) and a test information curve (TIC) by using the PLOTS=(IIC TIC) option.

Output 65.4.1 Eigenvalues of Polychoric Correlations

The IRT Procedure

Eigenvalues of the Polychoric Correlation Matrix				
	Eigenvalue	Difference	Proportion	Cumulative
1	5.57173396	4.19614614	0.3980	0.3980
2	1.37558781	0.29273244	0.0983	0.4962
3	1.08285537	0.12600033	0.0773	0.5736
4	0.95685504	0.09108909	0.0683	0.6419
5	0.86576595	0.09758221	0.0618	0.7038
6	0.76818374	0.12571683	0.0549	0.7586
7	0.64246691	0.06108305	0.0459	0.8045
8	0.58138386	0.04214553	0.0415	0.8461
9	0.53923833	0.10092835	0.0385	0.8846
10	0.43830998	0.07346977	0.0313	0.9159
11	0.36484021	0.04667935	0.0261	0.9419
12	0.31816085	0.03905135	0.0227	0.9647
13	0.27910950	0.06360101	0.0199	0.9846
14	0.21550849		0.0154	1.0000

Output 65.4.1 shows the eigenvalue table for this example. You can see that the first eigenvalue is much greater than the others, suggesting that a unidimensional model is reasonable for the data.

In the context of item response theory, the amount of information that each item or the entire test provides might not be evenly distributed across the entire continuum of latent constructs. The value of the slope parameter indicates the amount of information that the item provides. For this example, parameter estimates and item information curves are shown in Output 65.4.2 and Output 65.4.3, respectively. By examining the parameter estimates and the item information curves, you can see that items that have high slope values have tall, narrow information curves. For example, because the slope value of item9 is much larger than the slope value of item1, the information curve is taller and narrower for item9 than it is for item1.

Output 65.4.2 Parameter Estimates

The IRT Procedure

Item Parameter Estimates

Response Model	Item	Parameter	Estimate	Standard Error	Pr > |t|
Graded	**item1**	**Threshold 1**	-2.10586	0.34521	<.0001
		Threshold 2	3.26949	0.51165	<.0001
		Slope	0.45284	0.07033	<.0001
	item2	**Threshold 1**	-0.54235	0.07763	<.0001
		Threshold 2	0.47607	0.07308	<.0001
		Slope	1.20094	0.09670	<.0001
	item5	**Threshold 1**	-0.71115	0.09186	<.0001
		Threshold 2	0.62259	0.08600	<.0001
		Slope	1.03339	0.08727	<.0001
	item6	**Threshold 1**	-0.59481	0.11775	<.0001
		Threshold 2	1.20703	0.15230	<.0001
		Slope	0.70723	0.07602	<.0001
	item7	**Threshold 1**	-0.77669	0.06523	<.0001
		Threshold 2	0.25890	0.05286	<.0001
		Threshold 3	0.89742	0.06606	<.0001
		Slope	1.88555	0.12225	<.0001
	item8	**Threshold 1**	-0.73871	0.07456	<.0001
		Threshold 2	0.62587	0.06855	<.0001

The IRT Procedure

Item Parameter Estimates

Response Model	Item	Parameter	Estimate	Standard Error	Pr > |t|
		Threshold 3	1.38503	0.09804	<.0001
		Slope	1.40789	0.09825	<.0001
	item10	**Threshold 1**	-0.32385	0.05950	<.0001
		Threshold 2	0.69952	0.06626	<.0001
		Slope	1.66252	0.12039	<.0001
	item11	**Threshold 1**	-1.02188	0.07810	<.0001
		Threshold 2	-0.01845	0.05698	0.3731
		Threshold 3	0.66513	0.06466	<.0001
		Threshold 4	1.37500	0.09022	<.0001
		Slope	1.65682	0.10894	<.0001
	item12	**Threshold 1**	-1.87379	0.13805	<.0001
		Threshold 2	-0.79789	0.08674	<.0001
		Threshold 3	-0.08241	0.06987	0.1191
		Threshold 4	0.62089	0.07745	<.0001
		Threshold 5	1.25472	0.10243	<.0001
		Threshold 6	1.86433	0.13715	<.0001
		Slope	1.18637	0.08592	<.0001
	item13	**Threshold 1**	-0.80459	0.05879	<.0001

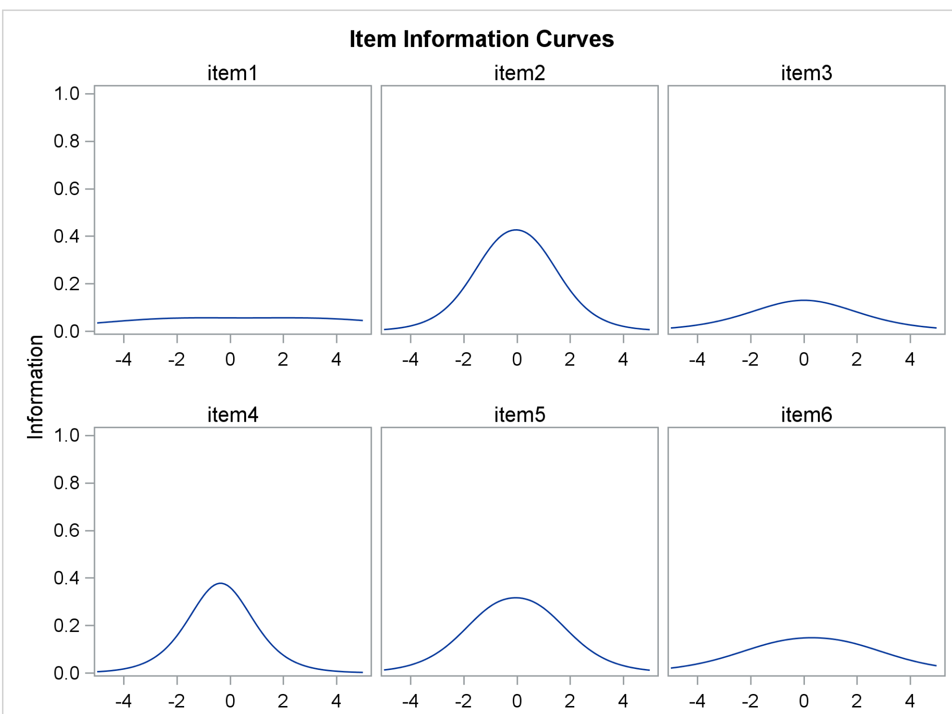
Output 65.4.2 *continued*

The IRT Procedure

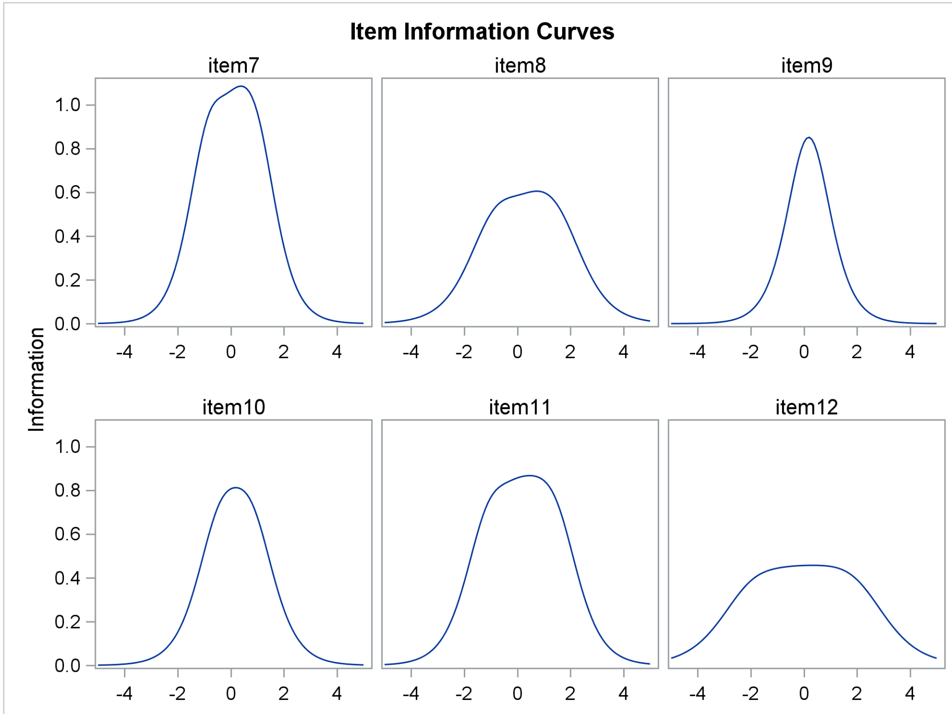
Item Parameter Estimates					
Response Model	**Item**	**Parameter**	**Estimate**	**Standard Error**	**Pr > |t|**
		Threshold 2	0.33318	0.04823	<.0001
		Threshold 3	1.10477	0.06442	<.0001
		Slope	2.48210	0.16037	<.0001
	item14	**Threshold 1**	-1.36389	0.09895	<.0001
		Threshold 2	0.37201	0.06328	<.0001
		Threshold 3	1.38161	0.09739	<.0001
		Slope	1.39349	0.09606	<.0001
TwoP	**item3**	**Difficulty**	0.00535	0.09877	0.4784
		Slope	0.72072	0.08488	<.0001
	item4	**Difficulty**	-0.35562	0.07124	<.0001
		Slope	1.22886	0.11089	<.0001
	item9	**Difficulty**	0.18415	0.06545	0.0024
		Slope	1.84562	0.20070	<.0001

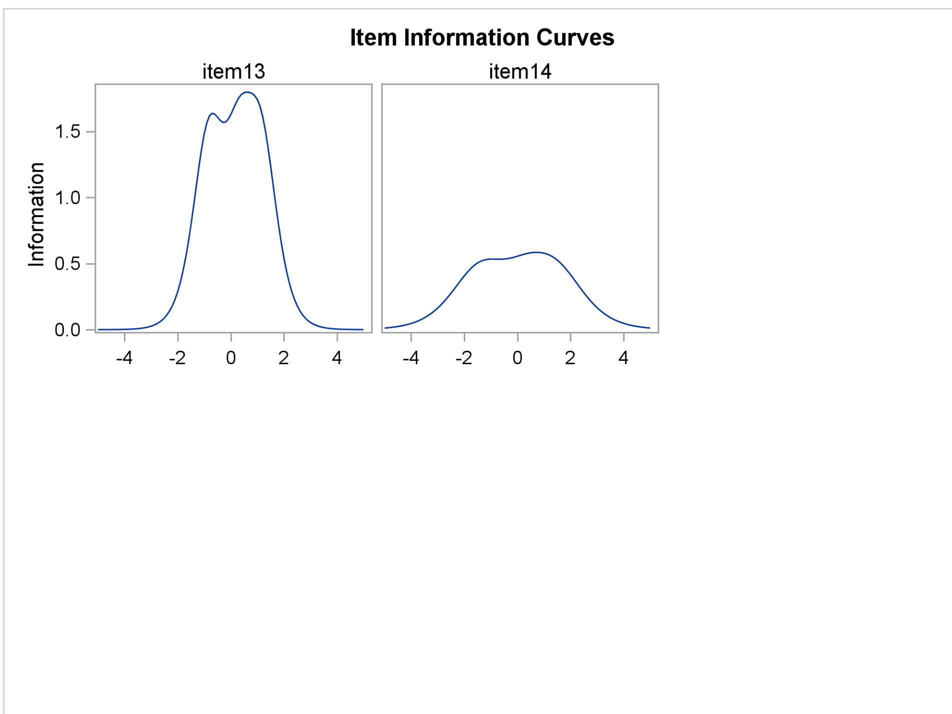
For individual items, most of the information concentrates around the area that is defined by the difficulty parameters. The binary response item provides most of the information around the difficulty parameter. For ordinal items, most of the information falls in the region between the lowest and the highest threshold parameters. By comparing the information curves for item7 and item9, you can also see that when response items have the same slope value, the ordinal item is more informative than the binary item.

Output 65.4.3 Item Information Curves

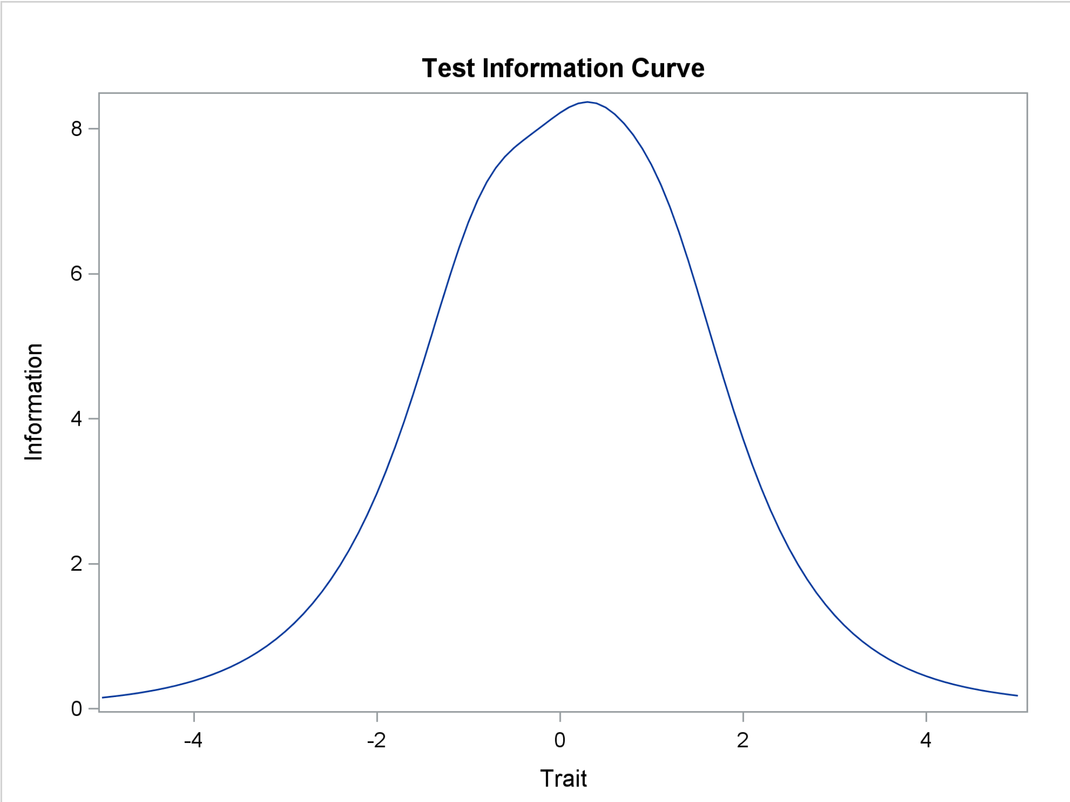


Output 65.4.3 *continued*





Output 65.4.4 Test Information Curves



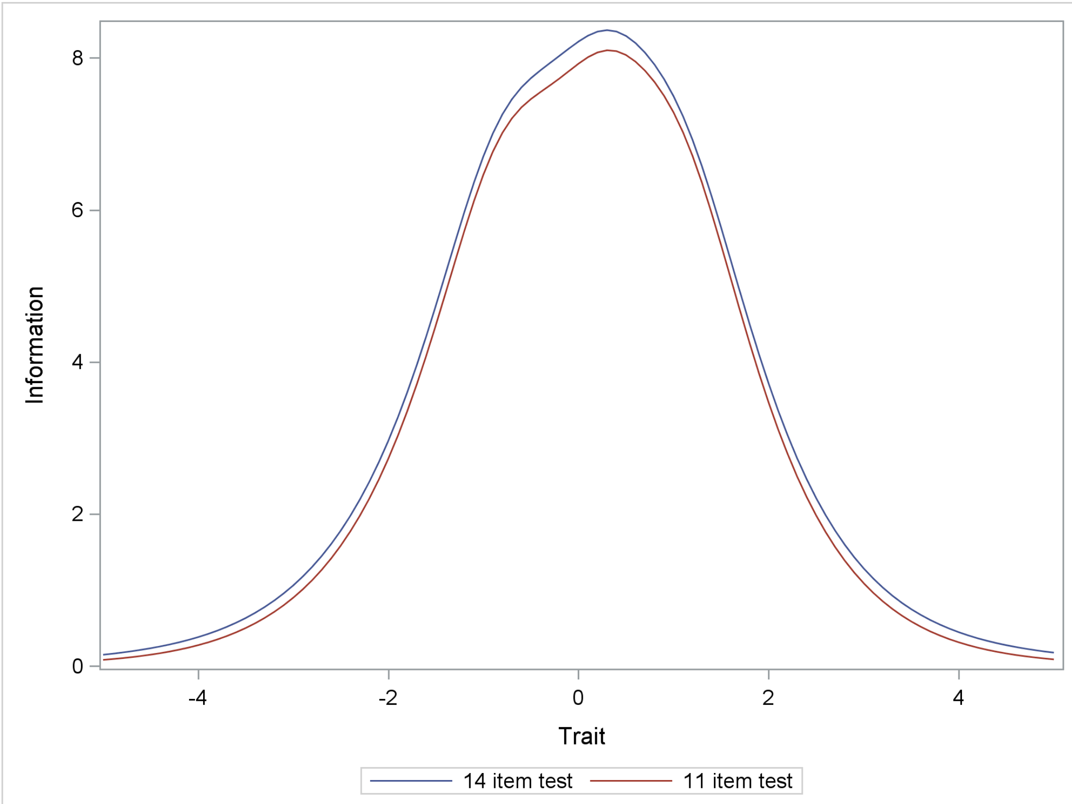
When all items in a test are considered together, the information for measuring the latent trait is called the test information. Test information is computed as a summation of the information that is provided by all the items in the test. Output 65.4.4 includes the test information curve for this example.

Item and test information are very useful for item selection. One important purpose of item selection is to maximize the test information across the continuum of latent construct of interest.

During the item selection process, ideally you want to select highly discriminating items whose threshold parameters cover the range of latent construct of interest. However, in practice you often encounter situations in which these highly discriminating items cannot provide enough information for a specific range of latent construct of interest, especially when these items are binary. In these situations, you might need to select some less discriminating items that can add information to the area that is not covered by these highly discriminating items.

For this example, the slope parameters range from 0.46 to 2.49, and the threshold parameters range from –2.1 to 3.2. Among these 14 items, three of them (item1, item3, and item6) have slope values less than 1. The slope value for item1 is less than 0.5, which is especially low. The item information curves suggest that these three items provide much less information than the other items. As a result, you might consider dropping these three items to economize future test administration. Output 65.4.5 shows the test information curves for the original test, which has 14 items, and the shorter test, which excludes item1, item3, and item6. The two information curves are almost identical, suggesting that the shorter test provides almost the same amount of information as the longer test. Because the shorter test is more economical, it is preferred for future testing.

Output 65.4.5 Test Information Curves



Example 65.5: Subject Scoring

This example also uses the American Quality of Life Survey data introduced in Example 65.1. The purpose of this example is to show you how to score subjects. You can score subjects in two different ways by using the IRT procedure. If you want to fit the model and score the subjects simultaneously, you simply use the OUT= option in the PROC IRT statement. The following statements fit an IRT model to the data set IrtUni and then score all the subjects in that data set based on the parameter estimates. Factor scores along with the original data in IrtUni are saved to a SAS data set called IrtScore. The first five observations in the OUT= data set are displayed in Output 65.5.1.

```
proc irt data=IrtUni out=IrtScore;
   var item1-item8;
run;

proc print data=IrtScore(obs=5);
run;
```

Output 65.5.1 Factor Scores for the First Five Observations

Obs	item1	item2	item3	item4	item5	item6	item7	item8	_Factor1
1	1	0	0	0	1	1	2	1	-0.38031
2	1	1	1	1	1	3	3	3	1.69165
3	0	1	0	0	1	1	1	1	-0.72590
4	1	0	0	1	0	1	2	3	0.12298
5	0	0	0	0	0	1	1	1	-1.59052

In applied research, it is not uncommon to want to save the parameter estimates (or item calibration result) and use them later to score new subjects without refitting the model. To accomplish this task, first you use the OUTMODEL= option in the PROC IRT statement to save the parameter estimates as follows:

```
proc irt data=CalData outmodel=IrtModel;
   var item1-item8;
run;
```

In the preceding statements, the data set CalData contains 500 random samples from the original data set IrtUni.

Then you use the following statements to score new subjects in the data set that is specified in the DATA= option, where the data set NewSub contains 50 random samples from the IrtUni data set. In the following statements, you use the INMODEL= option in the PROC IRT statement to input the model specification and parameter estimates from a previous analysis. To score the subjects without refitting the model, you also need to specify the SCORE suboption. Output 65.5.2 contains the first five observations in the OUT= data set.

```
proc irt data=NewSub inmodel(score)=IrtModel out=IrtScore2;
run;

proc print data=IrtScore2 (obs=5);
run;
```

Output 65.5.2 Factor Scores for the First Five Observations

Obs	Replicate	item1	item2	item3	item4	item5	item6	item7	item8	_Factor1
1	1	1	1	1	1	1	2	2	3	1.14800
2	1	1	0	1	1	1	1	3	1	0.45932
3	1	1	1	1	1	1	2	3	2	1.11391
4	1	1	1	1	1	1	1	1	2	0.56333
5	1	0	0	0	0	1	2	2	3	-0.20233

If you do not specify the SCORE suboption, PROC IRT uses the parameter values specified in the INMODEL= option as initial values, refits the model, and then uses the new parameter estimates to score the subjects.

References

Bock, R. D., and Aitkin, M. (1981). "Marginal Maximum Likelihood Estimation of Item Parameters: Application of an EM Algorithm." *Psychometrika* 46:443–459.

Bock, R. D., and Lieberman, M. (1970). "Fitting a Response Model for *n* Dichotomously Scored Items." *Psychometrika* 35:179–197.

De Ayala, R. J. (2009). *The Theory and Practice of Item Response Theory*. New York: Guilford Press.

Embretson, S. E., and Reise, S. P. (2000). *Item Response Theory for Psychologists*. Mahwah, NJ: Lawrence Erlbaum Associates.

Lesaffre, E., and Spiessens, B. (2001). "On the Effect of the Number of Quadrature Points in a Logistic Random Effects Model: An Example." *Journal of the Royal Statistical Society, Series C* 50:325–335.

Masters, G. N. (1982). "A Rasch Model for Partial Credit Scoring." *Psychometrika* 47:149–174.

McKinley, R. L., and Mills, C. N. (1985). "A Comparison of Several Goodness-of-Fit Statistics." *Applied Psychological Measurement* 9:49–57.

Muraki, E. (1992). "A Generalized Partial Credit Model: Application of an EM Algorithm." *Applied Psychological Measurement* 16:159–176.

Rabe-Hesketh, S., Skrondal, A., and Pickles, A. (2002). "Reliable Estimation of Generalized Linear Mixed Models Using Adaptive Quadrature." *Stata Journal* 2:1–21.

Yen, W. M. (1981). "Using Simulation Results to Choose a Latent Trait Model." *Applied Psychological Measurement* 5:245–262.

Subject Index

Syntax Index