

[Description](#)[mopts](#)[Methods and formulas](#)[Quick start](#)[Options](#)[References](#)[Menu](#)[Remarks and examples](#)[Also see](#)[Syntax](#)[Stored results](#)

## Description

`irt hybrid` fits IRT models to combinations of binary, ordinal, and nominal items.

## Quick start

1PL model for binary items `b1` to `b5` and 2PL model for binary items `b6` to `b10`

```
irt hybrid (1pl b1-b5) (2pl b6-b10)
```

Plot ICCs for each item

```
irtgraph icc
```

GRM for ordinal items `o1`, `o2`, and `o3`, 3PL model for binary items `b1` and `b2`, and NRM for nominal items `n1` to `n5`

```
irt hybrid (grm o1 o2 o3) (3pl b1 b2) (nrm n1-n5)
```

Plot CCCs for `o1`

```
irtgraph icc o1
```

## Menu

Statistics > IRT (item response theory)

## Syntax

```
irt hybrid (model varlist1 [ , mopts ]) (model varlist2 [ , mopts ]) [ ... ]
      [ if ] [ in ] [ weight ] [ , options ]
```

<i>model</i>	Description
<code>1pl</code>	One-parameter logistic model
<code>2pl</code>	Two-parameter logistic model
<code>3pl</code>	Three-parameter logistic model
<code>grm</code>	Graded response model
<code>pcm</code>	Partial credit model
<code>gpcm</code>	Generalized partial credit model
<code>rsm</code>	Rating scale model
<code>nrm</code>	Nominal response model

<i>mopts</i>	Description
<code>cns(<i>spec</i>)</code>	apply specified parameter constraints
<code>sepguessing</code>	estimate a separate pseudoguessing parameter for each item; allowed only with a 3pl model
<code>gsepguessing</code>	estimate separate pseudoguessing parameters for each group; allowed only with a group 3pl model

<i>options</i>	Description
<code>group(<i>varname</i>)</code>	fit model for different groups

Model	
<code>listwise</code>	drop observations with any missing items
SE/Robust	
<code>vce(<i>vcetype</i>)</code>	<i>vcetype</i> may be <code>oim</code> , <code>robust</code> , <code>cluster <i>clustvar</i></code> , <code>bootstrap</code> , or <code>jackknife</code>

Reporting	
<code>level(#)</code>	set confidence level; default is <code>level(95)</code>
<code>notable</code>	suppress coefficient table
<code>noheader</code>	suppress output header
<code>display_options</code>	control columns and column formats

Integration	
<code>intmethod(<i>intmethod</i>)</code>	integration method
<code>intpoints(#)</code>	set the number of integration points; default is <code>intpoints(7)</code>

Maximization	
<code>maximize_options</code>	control the maximization process; seldom used
<code>startvalues(<i>svmethod</i>)</code>	method for obtaining starting values
<code>noestimate</code>	do not fit the model; show starting values instead
<code>estmetric</code>	show parameter estimates in the estimation metric
<code>dnnumerical</code>	use numerical derivative techniques
<code>coeflegend</code>	display legend instead of statistics

<i>intmethod</i>	Description
<i>mv</i> aghermite	mean–variance adaptive Gauss–Hermite quadrature; the default
<i>mc</i> aghermite	mode–curvature adaptive Gauss–Hermite quadrature
<i>g</i> hermite	nonadaptive Gauss–Hermite quadrature

`bootstrap`, `by`, `collect`, `jackknife`, `statsby`, and `svy` are allowed; see [U] 11.1.10 Prefix commands.

Weights are not allowed with the `bootstrap` prefix; see [R] [bootstrap](#).

`vce()` and weights are not allowed with the `svy` prefix; see [SVY] [svy](#).

`fweights`, `iwweights`, and `pweights` are allowed; see [U] 11.1.6 [weight](#).

`startvalues()`, `noestimate`, `estmetric`, `dnumerical`, and `coeflegend` do not appear in the dialog box.

See [U] 20 [Estimation and postestimation commands](#) for more capabilities of estimation commands.

mopts

`cns(spec)` constrains item parameters to a fixed value or constrains two or more parameters to be equal; see [IRT] [irt constraints](#) for details.

`sepguessing` specifies that a separate pseudoguessing parameter be estimated for each item. This option is allowed only with a 3p1 model; see [IRT] [irt 3p1](#) for details.

`gsepguessing` specifies that separate pseudoguessing parameters be estimated for each group. This option is allowed only with a group 3p1 model.

Options

`group(varname)` specifies that the model be fit separately for the different values of *varname*; see [IRT] [irt, group\(\)](#) for details.

Model

`listwise` handles missing values through listwise deletion, which means that the entire observation is omitted from the estimation sample if any of the items are missing for that observation. By default, all nonmissing items in an observation are included in the likelihood calculation; only missing items are excluded.

SE/Robust

`vce(vcetype)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory (`oim`), that are robust to some kinds of misspecification (`robust`), that allow for intragroup correlation (`cluster clustvar`), and that use bootstrap or jackknife methods (`bootstrap`, `jackknife`); see [R] [vce\\_option](#).

Reporting

`level(#)`; see [R] [Estimation options](#).

`notable` suppresses the estimation table, either at estimation or upon replay.

`noheader` suppresses the output header, either at estimation or upon replay.

`display_options:` `noci`, `nopvalues`, `cformat(%fmt)`, `pformat(%fmt)`, `sformat(%fmt)`, and `nolstretch`; see [R] [Estimation options](#).

## Integration

`intmethod(intmethod)` specifies the integration method to be used for computing the log likelihood. `mvaghermite` performs mean and variance adaptive Gauss–Hermite quadrature; `mcaghermite` performs mode and curvature adaptive Gauss–Hermite quadrature; and `ghermite` performs nonadaptive Gauss–Hermite quadrature.

The default integration method is `mvaghermite`.

`intpoints(#)` sets the number of integration points for quadrature. The default is `intpoints(7)`, which means that seven quadrature points are used to compute the log likelihood.

The more integration points, the more accurate the approximation to the log likelihood. However, computation time increases with the number of integration points.

## Maximization

`maximize_options`: `difficult`, `technique(algorithm_spec)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, and `from(init_specs)`; see [R] [Maximize](#). Those that require special mention for `irt` are listed below.

`from()` accepts a properly labeled vector of initial values or a list of coefficient names with values. A list of values is not allowed.

The following options are available with `irt` but are not shown in the dialog box:

`startvalues()` specifies how starting values are to be computed. Starting values specified in `from()` override the computed starting values.

`startvalues(zero)` specifies that all starting values be set to 0. This option is typically useful only when specified with the `from()` option.

`startvalues(constantly)` builds on `startvalues(zero)` by fitting a constant-only model for each response to obtain estimates of intercept and cutpoint parameters.

`startvalues(fixedonly)` builds on `startvalues(constantly)` by fitting a full fixed-effects model for each response variable to obtain estimates of coefficients along with intercept and cutpoint parameters. You can also add suboption `iterate(#)` to limit the number of iterations `irt` allows for fitting the fixed-effects model.

`startvalues(ivloadings)` builds on `startvalues(fixedonly)` by using instrumental-variable methods with the generalized residuals from the fixed-effects models to compute starting values for latent-variable loadings. This is the default behavior.

`noestimate` specifies that the model is not to be fit. Instead, starting values are to be shown (as modified by the above options if modifications were made), and they are to be shown using the `coeflegend` style of output. An important use of this option is before you have modified starting values at all; you can type the following:

```
. irt ..., ... noestimate
. matrix b = e(b)
. ... (modify elements of b) ...
. irt ..., ... from(b)
```

`estmetric` displays parameter estimates in the slope-intercept metric that is used for estimation.

dnnumerical specifies that during optimization, the gradient vector and Hessian matrix be computed using numerical techniques instead of analytical formulas. By default, `irt` uses analytical formulas for computing the gradient and Hessian for all integration methods.

`coeflegend`; see [R] [Estimation options](#).

## Remarks and examples

The following discussion is about how to use `irt` to fit hybrid IRT models. In a hybrid model, one can fit different IRT models to subsets of items and perform a single calibration for the whole instrument. If you are new to the IRT features in Stata, we encourage you to read [\[IRT\] irt](#) first. If you are interested in the details of a specific IRT model, we refer you to the following.

### Binary response models

<code>irt 1pl</code>	One-parameter logistic model
<code>irt 2pl</code>	Two-parameter logistic model
<code>irt 3pl</code>	Three-parameter logistic model

### Categorical response models

<code>irt grm</code>	Graded response model
<code>irt nrm</code>	Nominal response model
<code>irt pcm</code>	Partial credit model
<code>irt rsm</code>	Rating scale model

## ► Example 1: Combining an NRM and a PCM within a single instrument

In [example 1](#) of [\[IRT\] irt nrm](#), we applied a NRM to the physical science test data from [de Ayala \(2022\)](#). The last item is in fact an open-ended question scored on a scale of 1 to 4; thus, a PCM may be more appropriate for this item.

We fit an NRM to items q1–q3 and a PCM to item q4 as follows:

```
. use https://www.stata-press.com/data/r19/science
(Physical science data from de Ayala (2009))

. irt hybrid (nrm q1-q3) (pcm q4)

Fitting fixed-effects model:
Iteration 0:  Log likelihood = -9256.1514
Iteration 1:  Log likelihood = -9256.1514

Fitting full model:
Iteration 0:  Log likelihood = -9383.3438   (not concave)
Iteration 1:  Log likelihood = -9251.6343   (not concave)
Iteration 2:  Log likelihood = -9200.7879
Iteration 3:  Log likelihood = -9183.2288
Iteration 4:  Log likelihood = -9169.5042
Iteration 5:  Log likelihood = -9168.4031
Iteration 6:  Log likelihood = -9168.2855
Iteration 7:  Log likelihood = -9168.2854
```

Hybrid IRT model

Number of obs = 1,799

Log likelihood = -9168.2854

		Coefficient	Std. err.	z	P> z	[95% conf. interval]	
nrm							
q1							
	Discrim						
	2 vs 1	.4646853	.1344318	3.46	0.001	.2012039	.7281667
	3 vs 1	-.7170683	.1835435	-3.91	0.000	-1.076807	-.3573297
	4 vs 1	-.5973805	.1601828	-3.73	0.000	-.9113331	-.2834279
	Diff						
	2 vs 1	-.7692399	.3102269	-2.48	0.013	-1.377274	-.1612063
	3 vs 1	-1.137519	.2353162	-4.83	0.000	-1.59873	-.6763076
4 vs 1	-.5488615	.1512123	-3.63	0.000	-.845232	-.2524909	
q2							
	Discrim						
	2 vs 1	-.1940623	.2058631	-0.94	0.346	-.5975465	.2094219
	3 vs 1	.5554902	.2001945	2.77	0.006	.1631162	.9478641
	4 vs 1	-.0926769	.1957145	-0.47	0.636	-.4762702	.2909165
	Diff						
	2 vs 1	5.986206	6.587602	0.91	0.364	-6.925257	18.89767
	3 vs 1	-3.529391	1.256803	-2.81	0.005	-5.99268	-1.066102
4 vs 1	17.80318	37.97769	0.47	0.639	-56.63173	92.23809	
q3							
	Discrim						
	2 vs 1	-.2375187	.2286107	-1.04	0.299	-.6855875	.2105501
	3 vs 1	.701352	.2196849	3.19	0.001	.2707775	1.131927
	4 vs 1	1.274855	.2460035	5.18	0.000	.7926971	1.757013
	Diff						
	2 vs 1	1.128287	1.746527	0.65	0.518	-2.294843	4.551418
	3 vs 1	-1.824874	.4596278	-3.97	0.000	-2.725728	-.9240203
4 vs 1	-1.131441	.1882566	-6.01	0.000	-1.500417	-.7624645	
pcm							
q4							
	Discrim	.3300808	.0526185	6.27	0.000	.2269504	.4332111
	Diff						
	2 vs 1	-2.056822	.322035	-6.39	0.000	-2.687999	-1.425644
	3 vs 2	-.2976236	.1923535	-1.55	0.122	-.6746294	.0793823
4 vs 3	.8472731	.2048051	4.14	0.000	.4458626	1.248684	

Note how the NRM and PCM are separated, so it is easy to tell which parameters correspond to which model. Because the PCM is nested in the NRM, we could perform a likelihood-ratio test to see whether our model is preferable to a pure NRM model; see [example 1](#) in [\[IRT\] irt](#) and [\[R\] lrtest](#) for more information.



## ► Example 2: The 3PL model revisited

In [example 1](#) of [\[IRT\] irt 3pl](#), we used the mathematics and science data from [De Boeck and Wilson \(2004\)](#) to fit a 3PL model where the pseudoguessing parameter was constrained to be the same across items q1–q9. We mentioned that model identification problems can occur when one tries to estimate a separate guessing parameter for each item. In this example, we show how to deal with identification problems by constraining some pseudoguessing parameters to zero and fitting a 3PL model with separate guessing parameters to the remaining items.

We first fit a full 3PL model to items q1–q9, where each item has its own pseudoguessing parameter. Because the corresponding fixed-effects model is not identified, we limit the number of iterations `irt` spends fitting the fixed-effects model to 5.

```
. use https://www.stata-press.com/data/r19/masc1
(Data from De Boeck & Wilson (2004))

. irt 3pl q1-q9, sepguessing startvalues(iterate(5))
Fitting fixed-effects model:
Iteration 0: Log likelihood = -5322.8824
Iteration 1: Log likelihood = -4291.3914 (not concave)
Iteration 2: Log likelihood = -4270.0005 (not concave)
Iteration 3: Log likelihood = -4269.7927 (not concave)
Iteration 4: Log likelihood = -4269.7825 (not concave)
Iteration 5: Log likelihood = -4269.7825 (not concave)
Fitting full model:
Iteration 0: Log likelihood = -4227.4731 (not concave)
Iteration 1: Log likelihood = -4188.8074 (not concave)
Iteration 2: Log likelihood = -4134.829 (not concave)
Iteration 3: Log likelihood = -4121.9664 (not concave)
Iteration 4: Log likelihood = -4120.161 (not concave)
Iteration 5: Log likelihood = -4119.33
Iteration 6: Log likelihood = -4118.0626
Iteration 7: Log likelihood = -4117.0488
Iteration 8: Log likelihood = -4115.6541
Iteration 9: Log likelihood = -4115.4168
Iteration 10: Log likelihood = -4114.5522
Iteration 11: Log likelihood = -4114.3738
Iteration 12: Log likelihood = -4114.1039
Iteration 13: Log likelihood = -4113.9668
Iteration 14: Log likelihood = -4113.9036
Iteration 15: Log likelihood = -4113.7972 (not concave)
Iteration 16: Log likelihood = -4113.7712
Iteration 17: Log likelihood = -4113.7505
Iteration 18: Log likelihood = -4113.7226
Iteration 19: Log likelihood = -4113.7178
Iteration 20: Log likelihood = -4113.7089
Iteration 21: Log likelihood = -4113.7033
Iteration 22: Log likelihood = -4113.6975
Iteration 23: Log likelihood = -4113.6964
Iteration 24: Log likelihood = -4113.6948
Iteration 25: Log likelihood = -4113.6939
Iteration 26: Log likelihood = -4113.6936
Iteration 27: Log likelihood = -4113.6934
Iteration 28: Log likelihood = -4113.6934
Iteration 29: Log likelihood = -4113.6933
```

Three-parameter logistic model  
Log likelihood = -4113.6933

Number of obs = 800

		Coefficient	Std. err.	z	P> z	[95% conf. interval]	
q1	Discrim	2.412093	.8686457	2.78	0.005	.709579	4.114608
	Diff	-.0919186	.2094281	-0.44	0.661	-.5023902	.318553
	Guess	.2054622	.1067005			-.003667	.4145914
q2	Discrim	.6595413	.1142254	5.77	0.000	.4356637	.883419
	Diff	-.14912	.1209748	-1.23	0.218	-.3862263	.0879862
	Guess	6.49e-06	.004595			-.0089996	.0090126
q3	Discrim	1.002138	.1672076	5.99	0.000	.6744172	1.329859
	Diff	-1.61108	.218446	-7.38	0.000	-2.039226	-1.182934
	Guess	9.89e-08	.0004223			-.0008275	.0008277
q4	Discrim	1.32466	.5435057	2.44	0.015	.2594084	2.389911
	Diff	.811461	.2349897	3.45	0.001	.3508897	1.272032
	Guess	.1921895	.0961314			.0037755	.3806035
q5	Discrim	.8519931	.1450072	5.88	0.000	.5677843	1.136202
	Diff	1.653157	.244456	6.76	0.000	1.174032	2.132282
	Guess	1.27e-08	.0001261			-.0002471	.0002471
q6	Discrim	2.160352	.8886889	2.43	0.015	.4185542	3.90215
	Diff	.8876168	.1201165	7.39	0.000	.6521929	1.123041
	Guess	.1725436	.0486371			.0772168	.2678705
q7	Discrim	.9442741	2.196661	0.43	0.667	-3.361102	5.24965
	Diff	2.599643	1.80189	1.44	0.149	-.9319954	6.131282
	Guess	.1862207	.2136893			-.2326026	.605044
q8	Discrim	1.477403	.2581921	5.72	0.000	.9713561	1.983451
	Diff	-1.664011	.1868776	-8.90	0.000	-2.030284	-1.297737
	Guess	1.33e-09	.0000219			-.0000429	.0000429
q9	Discrim	.6233966	.1200201	5.19	0.000	.3881615	.8586317
	Diff	-1.536892	.2867222	-5.36	0.000	-2.098858	-.9749272
	Guess	2.22e-08	.0001789			-.0003506	.0003506



We see that the pseudoguessing parameters for items q2, q3, q5, q8, and q9 are very close to zero. This suggests that we could fit a 2PL model to these five items and a full 3PL model with separate guessing parameters to the remaining four items.

```
. irt hybrid (2pl q2 q3 q5 q8 q9) (3pl q1 q4 q6 q7, sepg), startval(iter(5))
```

Fitting fixed-effects model:

```
Iteration 0: Log likelihood = -4846.1954
Iteration 1: Log likelihood = -4274.9988 (not concave)
Iteration 2: Log likelihood = -4269.8038 (not concave)
Iteration 3: Log likelihood = -4269.7889 (not concave)
Iteration 4: Log likelihood = -4269.7825 (not concave)
Iteration 5: Log likelihood = -4269.7825 (not concave)
```

Fitting full model:

```
Iteration 0: Log likelihood = -4237.32 (not concave)
Iteration 1: Log likelihood = -4156.6562
Iteration 2: Log likelihood = -4122.4275
Iteration 3: Log likelihood = -4115.0165
Iteration 4: Log likelihood = -4113.7357
Iteration 5: Log likelihood = -4113.7317
Iteration 6: Log likelihood = -4113.7155 (not concave)
Iteration 7: Log likelihood = -4113.7153
Iteration 8: Log likelihood = -4113.7124
Iteration 9: Log likelihood = -4113.7041
Iteration 10: Log likelihood = -4113.6966
Iteration 11: Log likelihood = -4113.6965
Iteration 12: Log likelihood = -4113.6938
Iteration 13: Log likelihood = -4113.694
Iteration 14: Log likelihood = -4113.6933
Iteration 15: Log likelihood = -4113.6933
```

Hybrid IRT model

Number of obs = 800

Log likelihood = -4113.6933

		Coefficient	Std. err.	z	P> z	[95% conf. interval]	
2pl							
q2	Discrim	.6595289	.1141777	5.78	0.000	.4357448	.8833131
	Diff	-.1491663	.1200093	-1.24	0.214	-.3843802	.0860477
q3	Discrim	1.002143	.1672015	5.99	0.000	.6744346	1.329852
	Diff	-1.611069	.2184352	-7.38	0.000	-2.039194	-1.182944
q5	Discrim	.8519284	.1449869	5.88	0.000	.5677594	1.136097
	Diff	1.65315	.2444541	6.76	0.000	1.174029	2.132271
q8	Discrim	1.477406	.2581479	5.72	0.000	.9714453	1.983366
	Diff	-1.664009	.1868519	-8.91	0.000	-2.030232	-1.297786
q9	Discrim	.6233934	.1200188	5.19	0.000	.3881608	.858626
	Diff	-1.536899	.286721	-5.36	0.000	-2.098862	-.9749362

3p1

q1	Discrim	2.4122	.868651	2.78	0.005	.7096758	4.114725
	Diff	-.0918963	.2094073	-0.44	0.661	-.502327	.3185344
	Guess	.2054735	.1066903			-.0036357	.4145828
q4	Discrim	1.324646	.5435556	2.44	0.015	.2592968	2.389995
	Diff	.8114595	.2349801	3.45	0.001	.350907	1.272012
	Guess	.1921879	.0961354			.0037661	.3806098
q6	Discrim	2.160387	.8880769	2.43	0.015	.4197886	3.900986
	Diff	.8876139	.120097	7.39	0.000	.652228	1.123
	Guess	.1725445	.0485985			.0772932	.2677957
q7	Discrim	.9441699	2.16823	0.44	0.663	-3.305483	5.193823
	Diff	2.599752	1.781284	1.46	0.144	-.8915	6.091004
	Guess	.1862199	.2109529			-.2272402	.5996801

Looking at the output, we see that the guessing parameters for q1, q4, q6, and q7 are similar. So we could further simplify this model by constraining the pseudoguessing parameter to be the same for the 3PL items. We could use a series of likelihood-ratio tests to choose among the competing models; see [example 1](#) in [\[IRT\] irt](#) and [\[R\] lrtest](#) for more information.



## Stored results

irt stores the following in `e()`:

### Scalars

<code>e(N)</code>	number of observations
<code>e(k)</code>	number of parameters
<code>e(k_eq)</code>	number of equations in <code>e(b)</code>
<code>e(k_dv)</code>	number of dependent variables
<code>e(k_rc)</code>	number of covariances
<code>e(k_rs)</code>	number of variances
<code>e(irt_k_eq)</code>	number of IRT equations
<code>e(k_items#)</code>	number of items in <code>#th</code> IRT equation
<code>e(sepguess#)</code>	1 if <code>#th</code> IRT model contains a separate pseudoguessing parameter
<code>e(k_cat#)</code>	number of categories for the <code>#th</code> item, ordinal
<code>e(k_out#)</code>	number of outcomes for the <code>#th</code> item, nominal
<code>e(ll)</code>	log likelihood
<code>e(N_clust)</code>	number of clusters
<code>e(N_groups)</code>	number of groups
<code>e(n_quad)</code>	number of integration points
<code>e(rank)</code>	rank of <code>e(V)</code>
<code>e(ic)</code>	number of iterations
<code>e(rc)</code>	return code
<code>e(converged)</code>	1 if target model converged, 0 otherwise

### Macros

<code>e(cmd)</code>	<code>gsem</code>
<code>e(cmd2)</code>	<code>irt</code>
<code>e(cmdline)</code>	command as typed
<code>e(model#)</code>	name of IRT model for the <code>#th</code> equation

<code>e(items#)</code>	names of items in #th IRT equation
<code>e(depvar)</code>	names of all item variables
<code>e(wtype)</code>	weight type
<code>e(wexp)</code>	weight expression
<code>e(title)</code>	title in estimation output
<code>e(clustvar)</code>	name of cluster variable
<code>e(groupvar)</code>	name of group variable
<code>e(family#)</code>	family for the #th <i>item</i>
<code>e(link#)</code>	link for the #th <i>item</i>
<code>e(intmethod)</code>	integration method
<code>e(vce)</code>	<i>vcetype</i> specified in <code>vce()</code>
<code>e(vcetype)</code>	title used to label Std. err.
<code>e(opt)</code>	type of optimization
<code>e(which)</code>	max or min; whether optimizer is to perform maximization or minimization
<code>e(method)</code>	estimation method: <code>ml</code>
<code>e(ml_method)</code>	type of <code>ml</code> method
<code>e(user)</code>	name of likelihood-evaluator program
<code>e(technique)</code>	maximization technique
<code>e(datasignature)</code>	the checksum
<code>e(datasignaturevars)</code>	variables used in calculation of checksum
<code>e(properties)</code>	<code>b</code> <code>V</code>
<code>e(estat_cmd)</code>	program used to implement <code>estat</code>
<code>e(predict)</code>	program used to implement <code>predict</code>
<code>e(covariates)</code>	list of covariates
<code>e(footnote)</code>	program used to implement the footnote display

#### Matrices

<code>e(_N)</code>	sample size for each item
<code>e(b)</code>	parameter vector
<code>e(b_pclass)</code>	parameter class
<code>e(cat#)</code>	categories for the #th item, ordinal
<code>e(out#)</code>	outcomes for the #th item, nominal
<code>e(Cns)</code>	constraints matrix
<code>e(ilog)</code>	iteration log (up to 20 iterations)
<code>e(gradient)</code>	gradient vector
<code>e(V)</code>	covariance matrix of the estimators
<code>e(V_modelbased)</code>	model-based variance
<code>e(groupvalue)</code>	vector of group values in <code>e(groupvar)</code>
<code>e(nobs)</code>	vector with number of observations per group

#### Functions

<code>e(sample)</code>	marks estimation sample
------------------------	-------------------------

In addition to the above, the following is stored in `r()`:

#### Matrices

<code>r(table)</code>	matrix containing the coefficients with their standard errors, test statistics, <i>p</i> -values, and confidence intervals
-----------------------	--

Note that results stored in `r()` are updated when the command is replayed and will be replaced when any `r-class` command is run after the estimation command.

## Methods and formulas

Methods and formulas are presented under the following headings:

*The likelihood*

*Groups*

*Gauss–Hermite quadrature*

*Adaptive quadrature*

## The likelihood

Let  $y_{ij}$  be the observed outcome for item  $i$  from person  $j$ . Define  $p_{ij} = \Pr(Y_{ij} = y_{ij} | \mathbf{B}_i, \theta_j)$ , where  $Y_{ij}$  represents the (yet to be observed) outcome,  $\mathbf{B}_i$  contains parameters for item  $i$ , and  $\theta_j$  is the ability (the latent trait) of person  $j$ . The functional form of  $p_{ij}$  and the parameters that go into  $\mathbf{B}_i$  depend on the choice of IRT model for item  $i$ .

Conditional on  $\theta_j$ , the item responses are assumed to be independent, so the conditional density for person  $j$  is given by

$$f(\mathbf{y}_j | \mathbf{B}, \theta_j) = \prod_{i=1}^I p_{ij}$$

where  $\mathbf{y}_j = (y_{1j}, \dots, y_{Ij})$ ,  $\mathbf{B} = (\mathbf{B}_1, \dots, \mathbf{B}_I)$ , and  $I$  is the number of items.

Missing items are skipped over in the above product by default. When the `listwise` option is specified, persons with any missing items are dropped from the estimation sample.

The likelihood for person  $j$  is computed by integrating out the latent variable from the joint density

$$L_j(\mathbf{B}) = \int_{-\infty}^{\infty} f(\mathbf{y}_j | \mathbf{B}, \theta_j) \phi(\theta_j) d\theta_j$$

where  $\phi(\cdot)$  is the density function for the standard normal distribution. The log likelihood for the estimation sample is simply the sum of the log likelihoods from the  $N$  persons in the estimation sample.

$$\log L(\mathbf{B}) = \sum_{j=1}^N \log L_j(\mathbf{B})$$

The integral in the formula for  $L_j(\mathbf{B})$  is generally not tractable, so we must use numerical methods.

## Groups

When the `group()` option is specified, each group has its own model parameters. The collection of model parameters is

$$\mathbf{B} = \begin{pmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \vdots \\ \mathbf{B}_G \end{pmatrix}$$

where  $G$  is the number of groups.

The overall log likelihood is

$$\log L(\mathbf{B}) = \sum_{g=1}^G \log L(\mathbf{B}_g)$$

## Gauss–Hermite quadrature

The integral of a function multiplied by the kernel of the standard normal distribution can be approximated using Gauss–Hermite quadrature (GHQ). For  $Q$ -point GHQ, let the abscissa and weight pairs be denoted by  $(x_q^*, w_q^*)$ ,  $q = 1, \dots, Q$ . The GHQ approximation is then

$$\int_{-\infty}^{\infty} f(x) \exp(-x^2) dx \approx \sum_{q=1}^Q w_q^* f(x_q^*)$$

Using the standard normal distribution yields the approximation

$$\int_{-\infty}^{\infty} f(x) \phi(x) dx \approx \sum_{q=1}^Q w_q f(x_q)$$

where  $x_q = \sqrt{2}x_q^*$  and  $w_q = w_q^*/\sqrt{\pi}$ . The GHQ approximation to the likelihood for person  $j$  is

$$L_j^{\text{GHQ}}(\mathbf{B}) = \sum_{q=1}^Q w_q f(\mathbf{y}_j | \mathbf{B}, x_q)$$

## Adaptive quadrature

This section sets the stage for mean–variance adaptive Gauss–Hermite quadrature (MVAGHQ) and mode-curvature adaptive Gauss–Hermite quadrature (MCAGHQ).

If we fix the item variables and the model parameters, we see that the posterior density for  $\theta_j$  is proportional to

$$\phi(\theta_j) f(\mathbf{y}_j | \mathbf{B}, \theta_j)$$

It is reasonable to assume that this posterior density can be approximated by a normal density with mean  $\mu_j$  and variance  $\tau_j$ . Instead of using the prior density of  $\theta_j$  as the weighting distribution in the integral, we can use our approximation for the posterior density,

$$L_j(\mathbf{B}) = \int_{-\infty}^{\infty} \frac{f(\mathbf{y}_j | \mathbf{B}, \theta_j) \phi(\theta_j)}{\phi(\theta_j, \mu_j, \tau_j)} \phi(\theta_j, \mu_j, \tau_j) d\theta_j$$

The likelihood is then approximated with

$$L_j^*(\mathbf{B}) = \sum_{q=1}^Q \omega_q f(\mathbf{y}_j | \mathbf{B}, \xi_q)$$

where  $\xi_q$  and the  $\omega_q$  are functions of  $x_q$  and  $w_q$  and the adaptive parameters  $\mu_j$  and  $\tau_j$ .

For MVAGHQ,  $\mu_j$  is the posterior mean, and  $\tau_j$  is the posterior variance of  $\theta_j$ . They are computed iteratively by updating the posterior moments by using the MVAGHQ approximation, starting with a zero mean and unit variance.

For MCAGHQ,  $\mu_j$  is the posterior mode for  $\theta_j$ , and  $\tau_j$  is the curvature at the mode. They are computed by optimizing the joint density with respect to  $\theta_j$ .

## References

- de Ayala, R. J. 2022. *The Theory and Practice of Item Response Theory*. 2nd ed. New York: Guilford Press.
- De Boeck, P., and M. Wilson, eds. 2004. *Explanatory Item Response Models: A Generalized Linear and Nonlinear Approach*. New York: Springer. <https://doi.org/10.1007/978-1-4757-3990-9>.

## Also see

- [IRT] [irt hybrid postestimation](#) — Postestimation tools for irt hybrid
- [IRT] [irt](#) — Introduction to IRT models
- [IRT] [irt 1pl](#) — One-parameter logistic model
- [IRT] [irt 2pl](#) — Two-parameter logistic model
- [IRT] [irt 3pl](#) — Three-parameter logistic model
- [IRT] [irt constraints](#) — Specifying constraints
- [IRT] [irt grm](#) — Graded response model
- [IRT] [irt nrm](#) — Nominal response model
- [IRT] [irt pcm](#) — Partial credit model
- [IRT] [irt rsm](#) — Rating scale model
- [SEM] [gsem](#) — Generalized structural equation model estimation command
- [SVY] [svy estimation](#) — Estimation commands for survey data
- [U] [20 Estimation and postestimation commands](#)

Stata, Stata Press, Mata, NetCourse, and NetCourseNow are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. StataNow is a trademark of StataCorp LLC. Other brand and product names are registered trademarks or trademarks of their respective companies. Copyright © 1985–2025 StataCorp LLC, College Station, TX, USA. All rights reserved.



For suggested citations, see the FAQ on [citing Stata documentation](#).