

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

Description

`areg` fits linear regression absorbing categorical factors. `areg` is designed for datasets with categorical variables that have many groups, but the number of groups for each variable does not increase with the sample size. See the `xtreg`, `fe` command in [\[XT\] xtreg](#) for an estimator that handles the case in which one categorical variable (often a panel identifier) has a number of groups that increases with the sample size; additional categorical variables with fixed group sizes may also be included.

Quick start

Linear regression of `y` on `x`, absorbing indicator variables for the levels of `cvar`

```
areg y x, absorb(cvar)
```

Same as above, but add categorical variable `a`

```
areg y x i.a, absorb(cvar)
```

Same as above, but with cluster-robust standard errors

```
areg y x i.a, absorb(cvar) vce(cluster cvar2)
```

Linear regression of `y` on `x`, absorbing indicator variables for the levels of `cvar1` and `cvar2`

```
areg y x, absorb(cvar1 cvar2)
```

Menu

Statistics > Linear models and related > Other > Linear regression absorbing cat. variables

Syntax

```
areg depvar [indepvars] [if] [in] [weight] , absorb(varlist [ , method ]) [options]
```

<i>options</i>	Description
Model	
* <u>absorb</u> (<i>varlist</i> [, <i>method</i>])	specify categorical variables to be absorbed
† <u>dfabsorb</u>	adjust degrees of freedom for collinearity among absorbed variables
<u>noabsorbtest</u>	suppress the <i>F</i> test of absorbed indicators
SE/Robust	
<u>vce</u> (<i>vcetype</i>)	<i>vcetype</i> may be <u>ols</u> , <u>robust</u> , <u>cluster</u> <i>clustvarlist</i> , <u>bootstrap</u> , <u>jackknife</u> , or <u>hc2</u> [<i>clustvar</i>]
Reporting	
<u>level</u> (#)	set confidence level; default is <u>level</u> (95)
<u>clustertable</u>	display table of multiway cluster combinations
<u>display_options</u>	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Optimization	
† <u>tolerance</u> (#)	convergence tolerance for maximum absolute difference; default is <u>tolerance</u> (1e-8)
† <u>iterate</u> (#)	maximum number of iterations for alternating projection method (APM); default is <u>iterate</u> (50)
† <u>nolog</u>	suppress the APM iteration log
<u>coeflegend</u>	display legend instead of statistics

*absorb() is required.

†Ignored if only one absorbed variable is specified.

indepvars may contain factor variables; see [\[U\] 11.4.3 Factor variables](#).

depvar and *indepvars* may contain time-series operators; see [\[U\] 11.4.4 Time-series varlists](#).

bootstrap, by, collect, fp, jackknife, mi estimate, rolling, and statsby are allowed; see [\[U\] 11.1.10 Prefix commands](#).

vce(bootstrap) and vce(jackknife) are not allowed with the mi estimate prefix; see [\[MI\] mi estimate](#).

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

aweights are not allowed with the jackknife prefix; see [\[R\] jackknife](#).

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

coeflegend does not appear in the dialog box.

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

Options

Model

absorb(*varlist* [, *method*]) specifies the categorical variables to be absorbed. The results are adjusted as if indicator variables for each level of each variable in *varlist* were included in the regression. absorb() is required.

When more than one categorical variable is absorbed, an APM iterative algorithm is used to project the *depvar* and *indepvars* to absorb these variables. *method* specifies the APM and is one of *halperin* or *cimmino*.

halperin, the default, uses the product of the projection matrices.

cimmino uses the mean of the projection matrices.

The two methods typically perform similarly. See [Stammann \(2018\)](#) for details.

method is ignored if only one absorbed variable is specified.

dfabsorb adjusts the degrees of freedom to account for collinearity among the absorbed variables. The default degrees of freedom assumes all absorbed variables are independent. This option is ignored if only one absorbed variable is specified in *absorb()*.

noabsorbtest prevents computation of the F test that all coefficients on indicators for absorbed variables are jointly zero. Computation of this test requires an iterative search. If the F test is of no interest, you may specify *noabsorbtest* and save computational time.

SE/Robust

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

vce(ols), the default, uses the standard variance estimator for ordinary least-squares regression.

vce(cluster clustvarlist) specifies that standard errors allow for intragroup correlation within groups defined by one or more variables in *clustvarlist*, relaxing the usual requirement that the observations be independent. For example, *vce(cluster clustvar1)* produces cluster-robust standard errors that allow for observations that are independent across groups defined by *clustvar1* but not necessarily independent within groups. You could also type *vce(cluster clustvar1 clustvar2 ... clusterp)* to account for correlation within groups formed by p variables (multiway clustering).

areg also allows the following:

vce(hc2 [clustvar] [, dfadjust]) specifies an alternative bias correction for the robust variance calculation. In the unclustered case, *vce(robust)* uses $\hat{\sigma}_j^2 = \{n/(n-k)\}u_j^2$ as an estimate of the variance of the j th observation, where n is the number of observations, k is the number of regressors, u_j is the calculated residual, and $n/(n-k)$ is included to improve the overall estimate's small-sample properties.

vce(hc2) instead uses $u_j^2/(1 - h_{jj})$ as the observation's variance estimate, where h_{jj} is the diagonal element of the hat (projection) matrix. This estimate is unbiased if the model really is homoskedastic. *vce(hc2)* tends to produce slightly more conservative confidence intervals. *vce(hc2 clustvar)* produces estimates that allow for intragroup correlation within groups defined by *clustvar*. *dfadjust* computes the [Bell and McCaffrey \(2002\)](#) adjusted degrees of freedom based on *clustvar*. Note that *dfadjust* does not affect multiple-imputation results when the command is used with *mi* estimate. See [Methods and formulas](#) in [\[R\] regress](#) for a description of the computation when *clustvar* is specified.

vce(hc2) is not allowed when multiple variables are absorbed.

Reporting

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

`clustertable` displays a table reporting cluster combinations and the number of clusters per combination. This option is available only when `vce(cluster clustvarlist)` is specified with more than one variable in *clustvarlist* to compute multiway cluster-robust standard errors.

display_options: `nocl`, `nopvalues`, `dfci`, `dfpvalues`, `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fvwrap(#)`, `fvwrapon(style)`, `cformat(%fmt)`, `pformat(%fmt)`, `sformat(%fmt)`, and `nolstretch`; see [R] [Estimation options](#).

`dfci` specifies that parameter degrees of freedom and confidence intervals be reported in the coefficient table.

`dfpvalues` specifies that parameter degrees of freedom and *p*-values be reported in the coefficient table.

Optimization

`tolerance(#)` specifies the limit for the maximum absolute difference between iterations for the projected *depvar* and *indepvars*. The default is `tolerance(1e-8)`.

`iterate(#)` specifies the maximum number of iterations for the APM. The default is `iterate(50)`.

`noolog` specifies that no APM iterative log be displayed.

The following option is available with `areg` but is not shown in the dialog box:

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

Remarks and examples

Suppose that we have a linear regression model that includes among predictor variables indicators for the levels of a high-dimensional categorical variable *C*. For instance, the levels of *C* could represent counties, neighborhoods, or streets. We could write the model as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{d}_1\gamma_1 + \mathbf{d}_2\gamma_2 + \cdots + \mathbf{d}_m\gamma_m + \boldsymbol{\epsilon}$$

Alternatively, if there are *N* observations and *m* categories of variable *C*, we could write this as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{D}\boldsymbol{\gamma} + \boldsymbol{\epsilon}$$

where **D** is an $N \times m$ indicator matrix for the categories of *C* and

$$\mathbf{D} = (\mathbf{d}_1 \quad \mathbf{d}_2 \quad \cdots \quad \mathbf{d}_m) \quad \boldsymbol{\gamma} = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_m \end{pmatrix}$$

Each \mathbf{d}_a is a column vector, the indicator variable for category *a* of variable *C*.

One option would be to fit the model with `regress`,

```
. regress y x* i.C
```

but this may be computationally expensive when the dimension, m , is very large. Moreover, you may not be interested in the estimates of γ_i , but you want the remaining results to be adjusted for the inclusion of C in the model.

`areg` provides a faster way of obtaining estimates of β —but not the γ_i ’s—in these cases. The effects of the C variable are said to be absorbed.

► Example 1

So that we can compare the results produced by `areg` with Stata’s other regression commands, we will fit a model in which m is small. `areg`’s real use, however, is when m is large.

In our automobile data, we have a variable called `rep78` that is coded 1, 2, 3, 4, and 5, where 1 means poor and 5 means excellent. Let’s assume that we wish to fit a regression of `mpg` on `weight`, `gear_ratio`, and `rep78` (parameterized as a set of indicators).

```
. use https://www.stata-press.com/data/r19/auto2
(1978 automobile data)
```

```
. regress mpg weight gear_ratio b5.rep78
```

Source	SS	df	MS	Number of obs	=	69
Model	1575.97621	6	262.662702	F(6, 62)	=	21.31
Residual	764.226686	62	12.3262369	Prob > F	=	0.0000
				R-squared	=	0.6734
				Adj R-squared	=	0.6418
Total	2340.2029	68	34.4147485	Root MSE	=	3.5109

mpg	Coefficient	Std. err.	t	P> t	[95% conf. interval]	
weight	-.0051031	.0009206	-5.54	0.000	-.0069433	-.003263
gear_ratio	.901478	1.565552	0.58	0.567	-2.228015	4.030971
rep78						
Poor	-2.036937	2.740728	-0.74	0.460	-7.515574	3.4417
Fair	-2.419822	1.764338	-1.37	0.175	-5.946682	1.107039
Average	-2.557432	1.370912	-1.87	0.067	-5.297846	.1829814
Good	-2.788389	1.395259	-2.00	0.050	-5.577473	.0006939
_cons	36.23782	7.01057	5.17	0.000	22.22389	50.25175

To fit the `areg` equivalent, we type

```
. areg mpg weight gear_ratio, absorb(rep78)
```

```
Linear regression, absorbing indicators
Absorbed variable: rep78
```

```
Number of obs      =    69
No. of categories  =     5
F(2, 62)           =   41.64
Prob > F           =  0.0000
R-squared          =  0.6734
Adj R-squared      =  0.6418
Root MSE          =  3.5109
```

mpg	Coefficient	Std. err.	t	P> t	[95% conf. interval]	
weight	-.0051031	.0009206	-5.54	0.000	-.0069433	-.003263
gear_ratio	.901478	1.565552	0.58	0.567	-2.228015	4.030971
_cons	34.05889	7.056383	4.83	0.000	19.95338	48.1644

F test of absorbed indicators: F(4, 62) = 1.117

Prob > F = 0.356

Both `regress` and `areg` display the same R^2 values, root mean squared error, and—for `weight` and `gear_ratio`—the same parameter estimates, standard errors, t statistics, significance levels, and confidence intervals. `areg`, however, does not report the coefficients for `rep78`, and, in fact, they are not even calculated. This computational trick makes the problem manageable when m is large. `areg` reports a test that the coefficients associated with `rep78` are jointly zero. Here this test has a significance level of 35.6%. This F test for `rep78` is the same that we would obtain after `regress` if we were to specify `test 1.rep78 2.rep78 3.rep78 4.rep78`; see [R] [test](#).

The model F tests reported by `regress` and `areg` also differ. The `regress` command reports a test that all coefficients except that of the constant are equal to zero; thus, the indicators are included in this test. The `areg` output shows a test that all coefficients excluding the indicators and the constant are equal to zero. This is the same test that can be obtained after `regress` by typing `test weight gear_ratio`.

◀

□ Technical note

`areg` is designed for datasets with many groups, but not a number that grows with the sample size. Consider two different samples from the US population. In the first sample, we have 10,000 individuals and we want to include an indicator for each of the 50 states, whereas in the second sample we have 3 observations on each of 10,000 individuals and we want to include an indicator for each individual. `areg` was designed for datasets similar to the first sample in which we have a fixed number of groups, the 50 states. In the second sample, the number of groups, which is the number of individuals, grows as we include more individuals in the sample. For an estimator designed to handle the case in which the number of groups grows with the sample size, see the `xtreg`, `fe` command in [XT] [xtreg](#).

Although the point estimates produced by `areg` and `xtreg`, `fe` are the same, the estimated VCEs differ when `vce(cluster clustvarlist)` is specified because the commands make different assumptions about whether the number of groups increases with the sample size.

□

□ Technical note

The intercept reported by `areg` deserves some explanation because, given m mutually exclusive and exhaustive indicators, it is arbitrary. `areg` identifies the model by choosing the intercept that makes the prediction calculated at the means of the independent variables equal to the mean of the dependent variable: $\bar{y} = \bar{x}\hat{\beta}$.

```
. predict yhat
(option xb assumed; fitted values)
. summarize mpg yhat if rep78 != .
```

Variable	Obs	Mean	Std. dev.	Min	Max
mpg	69	21.28986	5.866408	12	41
yhat	69	21.28986	4.383224	11.58643	28.07367

We had to include `if rep78 != .` in our `summarize` command because we have missing values in our data. `areg` automatically dropped those missing values (as it should) in forming the estimates, but `predict` with the `xb` option will make predictions for cases with missing `rep78` because it does not know that `rep78` is really part of our model.

These predicted values do not include the absorbed effects (that is, the $\mathbf{d}_i\gamma_i$). For predicted values that include these effects, use the `xbd` option of `predict` (see [R] [areg postestimation](#)) or see [XT] [xtreg](#).

□

► Example 2

`areg, vce(robust)` is a Huberized version of `areg`; see [P] [_robust](#). Just as `areg` is equivalent to using `regress` with indicators, `areg, vce(robust)` is equivalent to using `regress, vce(robust)` with indicators. You can use `areg, vce(robust)` when you expect heteroskedastic or nonnormal errors. `areg, vce(robust)`, like ordinary regression, assumes that the observations are independent, unless the `vce(cluster clustvarlist)` or `vce(hc2 clustvar)` option is specified. If the `vce(cluster clustvarlist)` or `vce(hc2 clustvar)` option is specified, this independence assumption is relaxed and only the clusters identified by equal values of *clustvarlist* or *clustvar* are assumed to be independent.

Assume that we were to collect data by randomly sampling 10,000 doctors (from 100 hospitals) and then sampling 10 patients of each doctor, yielding a total dataset of 100,000 patients in a cluster sample. If in some regression we wished to include effects of the hospitals to which the doctors belonged, we would want to include an indicator variable for each hospital, adding 100 variables to our model. `areg` could fit this model by

```
. areg depvar patient_vars, absorb(hospital) vce(cluster doctor)
```



Stored results

`areg` stores the following in `e()`:

Scalars

<code>e(N)</code>	number of observations
<code>e(k_absorb)</code>	total number of absorbed categories
<code>e(mss)</code>	model sum of squares
<code>e(tss)</code>	total sum of squares
<code>e(df_m)</code>	model degrees of freedom
<code>e(rss)</code>	residual sum of squares
<code>e(df_r)</code>	residual degrees of freedom
<code>e(r2)</code>	R^2
<code>e(r2_a)</code>	adjusted R^2
<code>e(df_a)</code>	degrees of freedom for absorbed effect
<code>e(rmse)</code>	root mean squared error
<code>e(ll)</code>	log likelihood
<code>e(ll_0)</code>	log likelihood, constant-only model
<code>e(N_clust)</code>	number of clusters
<code>e(F)</code>	F statistic
<code>e(F_absorb)</code>	F statistic for absorbed effect (when <code>vce(robust)</code> is not specified)
<code>e(p)</code>	p -value for model F test
<code>e(p_absorb)</code>	p -value for F test of absorbed effect
<code>e(rank)</code>	rank of <code>e(V)</code>
<code>e(converged)</code>	1 if APM converged, 0 otherwise

Macros

<code>e(cmd)</code>	<code>areg</code>
<code>e(cmdline)</code>	command as typed
<code>e(depvar)</code>	name of dependent variable
<code>e(absvar)</code>	names of absorbed variables
<code>e(apm)</code>	alternating projection method
<code>e(wtype)</code>	weight type
<code>e(wexp)</code>	weight expression
<code>e(title)</code>	title in estimation output
<code>e(clustvar)</code>	names of cluster variables
<code>e(cluster#)</code>	cluster combination #
<code>e(vce)</code>	<i>vcetype</i> specified in <code>vce()</code>

e(vcetype)	title used to label Std. err.
e(datasignature)	the checksum
e(datasignaturevars)	variables used in calculation of checksum
e(properties)	b V
e(predict)	program used to implement predict
e(footnote)	program used to implement the footnote display
e(marginsnotok)	predictions disallowed by margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance
e(adj_df)	adjusted degrees of freedom when vce(hc2, dfadjust) is specified
e(kcluster)	cluster sizes, multiway clustering
e(kabsorb)	number of levels for each absorbed variable
e(dfabsorb)	adjusted degrees of freedom for each absorbed variable
e(ksingle)	number of singletons for each absorbed variable

Functions

e(sample)	marks estimation sample
-----------	-------------------------

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

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p -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

With one absorbed variable, `areg` begins by recalculating *depvar* and *indepvars* to have mean 0 within the groups specified by `absorb()`. The overall mean of each variable is then added back in. The adjusted *depvar* is then regressed on the adjusted *indepvars* with `regress`, yielding the coefficient estimates. This is a direct application of the Frisch–Waugh–Lovell theorem (see Hansen [2022, 82]). The degrees of freedom of the variance–covariance matrix of the coefficients is then adjusted to account for the absorbed variables—this calculation yields the same results (up to numerical roundoff error) as if the matrix had been calculated directly by the formulas given in [R] `regress`.

With multiple absorbed variables, an APM with a conjugate gradient acceleration technique is used to adjust the *depvar* and *indepvars* (Hernández-Ramos, Escalante, and Raydan 2011). Two projection methods are available: Halperin (the default) and Cimmino. You can specify which to use with the `halperin` or `cimmino` suboption within the `absorb()` option. We describe both methods below.

Suppose we have h categorical variables, C_1, \dots, C_h , that we would like to include as controls in our regression. For variable C_k , let m_k be its number of levels and $\mathbf{d}_{k(a)}$ be its $N \times 1$ indicator vector for category a of C_k . Let \mathbf{D}_k denote the $N \times m_k$ matrix of indicators for variable C_k :

$$\mathbf{D}_k = (\mathbf{d}_{k(1)} \quad \mathbf{d}_{k(2)} \quad \dots \quad \mathbf{d}_{k(m_k)})$$

The orthonormal projection matrix for C_k is therefore given by $\mathbf{P}_k = \mathbf{D}_k(\mathbf{D}_k' \mathbf{D}_k)^{-1} \mathbf{D}_k'$. Let \mathbf{y} be the $N \times 1$ vector with the values of the dependent variable in the sample. Similarly, let \mathbf{X} be the matrix with the values of the covariates. Thus, the product $\bar{\mathbf{y}}_k = \mathbf{P}_k \mathbf{y}$ is the projection of the dependent variable onto

the column space of \mathbf{D}_k . That is, $\bar{\mathbf{y}}_k$ is the $N \times 1$ vector containing the (repeated) means of y_i for each level of C_k , in the order that these levels appear in the sample. The same projection is applied to the columns of covariate matrix \mathbf{X} .

The Halperin algorithm first sets $\tilde{\mathbf{y}}_0^{(1)} = \mathbf{y}$ and loops over the h absorbed variables computing projection residuals $\tilde{\mathbf{y}}_k^{(1)} = (\mathbf{I} - \mathbf{P}_k)\tilde{\mathbf{y}}_{k-1}^{(1)}$, for $k = 1, \dots, h$. Then, it repeats the loop with $\tilde{\mathbf{y}}_0^{(j+1)} = \tilde{\mathbf{y}}_h^{(j)}$ until convergence at $j = j$. Convergence is declared when $|\tilde{\mathbf{y}}_h^{(j)} - \tilde{\mathbf{y}}_0^{(j)}|$ is less than the specified tolerance, where the matrix norm $|\cdot|$ is defined as the largest entry in absolute value. The same computations are applied to the columns of covariate matrix \mathbf{X} . On convergence, the overall mean $\bar{\mathbf{y}}$ is added to $\tilde{\mathbf{y}}_h^{(j)}$, and the vector of means $\bar{\mathbf{x}}$ is added to the columns of $\tilde{\mathbf{X}}_h^{(j)}$.

The Cimmino algorithm first sets $\tilde{\mathbf{y}}^{(0)} = \mathbf{y}$ and then iteratively computes

$$\tilde{\mathbf{y}}^{(j+1)} = \left(\mathbf{I} - \frac{1}{h} \sum_{k=1}^h \mathbf{P}_k \right) \tilde{\mathbf{y}}^{(j)}$$

until convergence at $j + 1 = j$. It then repeats the process for covariance matrix \mathbf{X} and adds the overall means, $\bar{\mathbf{y}}$ and $\bar{\mathbf{x}}$, to $\tilde{\mathbf{y}}^{(j)}$ and to the columns of $\tilde{\mathbf{X}}^{(j)}$, respectively.

Efficient computation of the projection $\mathbf{P}\mathbf{y}$ can be done in Mata without generating the $N \times N$ matrix \mathbf{P} directly (see function `panelsum()`, for instance).

If the `dfabsorb` option is specified, an approximation to the absorbed variables, degrees of freedom is computed to account for collinearity among the absorbed variables. By default, `areg` assumes none of the variables in `absorb()` are collinear. Checking for collinearity could be computationally taxing. Thus, when `dfabsorb` is specified, a bipartite graph (Correia 2016) is generated for each pair of absorbed variables. This bipartite graph technique used by `dfabsorb` detects collinearities among pairs of categorical variables specified in the `absorb()` option. However, this technique does not detect collinearities among more than two absorbed variables, which would occur if they are included as factor variables in a regression. Therefore, the absorbed variable degrees of freedom is an upper bound.

More specifically, when `dfabsorb` is specified, the graph for each variable pair is decomposed to a set of subgraphs using Kruskal's greedy search algorithm (Cormen et al. 2022). Consider two absorbed variables a_1 and a_2 with m_1 and m_2 levels, respectively, and assume we generate k subgraphs. The number of estimable absorbed effects is $m_1 + m_2 - k$ (Abowd, Creecy, and Kramarz 2002) and corresponds to the degrees of freedom used when `dfabsorb` is specified. Here k can be considered the number of collinear indicator variables corresponding to the levels of a_1 and a_2 . The default degrees of freedom, on the other hand, is $m_1 + m_2 - 2$, which is the degrees of freedom if a single spanning tree is generated from the bipartite graph.

`areg` with `vce(robust)` or `vce(cluster clustvarlist)` with only one variable specified in `clustvarlist` works similarly, calling `_robust` after `regress` to produce the Huber/White/sandwich estimator of the variance or its clustered version. See [P] [_robust](#), particularly [Introduction](#) and [Methods and formulas](#). For p -way clustering, `vce(cluster clustvar1 [clustvar2 [...]])`, `_robust` is called for each of the $2^p - 1$ cluster combinations; details can be found in [Multiway clustering](#) of [Methods and formulas](#) in [R] [regress](#). The model F test uses the robust or cluster-robust variance estimates. There is, however, no simple computational means of obtaining a robust or cluster-robust test of the absorbed indicators. Thus, this test is not displayed when the `vce(robust)`, `vce(cluster clustvarlist)`, or `vce(hc2 [clustvar])` option is specified.

`areg` with `vce(hc2 [clustvar], [dfadjust])` specifies an alternative bias correction for the robust variance calculation. See [Robust calculation for regress](#) of [Methods and formulas](#) in [R] [regress](#) for a description of this VCE and the adjusted degrees of freedom. Computing this alternative VCE after ab-

sorbing indicator variables requires modification of the described procedure because the scores, denoted \mathbf{u}_j , are not invariant to projection unless the absorbed variable is nested within the clusters defined by *clustvar*. Rather than constructing the VCE using the original data, which can be computationally intensive, *areg* computes corrections to the scores using summary information about the absorbed variables. Similar corrections are used to compute the adjusted degrees of freedom if `dfadjust` is specified. When weights are specified, the clustering variable is required to nest the absorbed variable.

The number of groups specified in `absorb()` are included in the degrees of freedom used in the finite-sample adjustment of the cluster-robust VCE estimator. This statement is valid only if the number of groups is small relative to the sample size. (Technically, the number of groups must remain fixed as the sample size grows.) For an estimator that allows the number of groups to grow with the sample size, see the `xtreg`, `fe` command in [XT] *xtreg*.

Acknowledgment

We thank Sergio Correia at the Federal Reserve Board, Paulo Guimarães at the Bank of Portugal, and Fernando Rios-Avila at the Levy Economics Institute at Bard College for their work on controlling for high-dimensional categorical variables.

References

- Abowd, J. M., R. H. Creecy, and F. Kramarz. 2002. Computing person and firm effects using linked longitudinal employer–employee data. Technical Paper TP-2002-06, Center for Economic Studies, US Census Bureau. <https://www2.census.gov/ces/tp/tp-2002-06.pdf>.
- Bell, R. M., and D. F. McCaffrey. 2002. Bias reduction in standard errors for linear regression with multi-stage samples. *Survey Methodology* 28: 169–181.
- Blackwell, J. L., III. 2005. *Estimation and testing of fixed-effect panel-data systems*. *Stata Journal* 5: 202–207.
- Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein. 2022. *Introduction to Algorithms*. 4th ed. Cambridge, MA: MIT Press.
- Correia, S. 2016. A feasible estimator for linear models with multi-way fixed effects. Unpublished manuscript, Duke University, <https://scoreia.com/research/hdfe.pdf>.
- Correia, S., P. Guimarães, and T. Zylkin. 2020. *Fast Poisson estimation with high-dimensional fixed effects*. *Stata Journal* 20: 95–115.
- Guimarães, P., and P. Portugal. 2010. *A simple feasible procedure to fit models with high-dimensional fixed effects*. *Stata Journal* 10: 628–649.
- Hansen, B. E. 2022. *Econometrics*. Princeton, NJ: Princeton University Press.
- Hernández-Ramos, L. M., R. Escalante, and M. Raydan. 2011. Unconstrained optimization techniques for the acceleration of alternating projection methods. *Numerical Functional Analysis and Optimization* 32: 1041–1066. <https://doi.org/10.1080/01630563.2011.591954>.
- McCaffrey, D. F., K. Mihaly, J. R. Lockwood, and T. R. Sass. 2012. *A review of Stata commands for fixed-effects estimation in normal linear models*. *Stata Journal* 12: 406–432.
- Rios-Avila, F. 2015. *Feasible fitting of linear models with N fixed effects*. *Stata Journal* 15: 881–898.
- Stammann, A. 2018. Fast and feasible estimation of generalized linear models with high-dimensional k -way fixed effects. arXiv:1707.01815 [stat.AP], <https://doi.org/10.48550/arXiv.1707.01815>.

Also see

- [R] [areg postestimation](#) — Postestimation tools for areg
- [R] [regress](#) — Linear regression
- [R] [wildbootstrap](#) — Wild cluster bootstrap inference
- [MI] [Estimation](#) — Estimation commands for use with mi estimate
- [XT] [xtreg](#) — Linear models for panel 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](#).