The `ginteff` User's Manual

`ginteff` – compute two- and three-way interaction effects[1]

Description

`ginteff` computes the average and individual-level interaction effects for two- and three-way interactions. The effect of the interacted variables can be computed via either the partial derivative or the first difference.[2]

Quick start

Compute the two-way interaction effect between x1 and x2 after

```
logit y c.x1##c.x2 x3 x4
```

by taking the cross partial derivative with respect to x1 and x2, while holding x3 and x4 at observed values:

```
ginteff, dydxs(x1 x2)
```

As above, but for all combinations of x3 = 10, 20, 30, 40 and x4 = 50, 100:

```
ginteff, dydxs(x1 x2) at(x3=(10(10)40) x4=(50 100))
```

Compute the interaction effect between x1 and x2 via the first difference approach. Specifically, calculate the effect of increasing both x1 and x2 by 1-unit:

```
ginteff, firstdiff((asobserved) x1 x2) nunit((1) x1 x2)
```

Compute the three-way interaction effect between x1, x2, and x3 after

```
probit y c.x1##c.x2##c.x3
```

by taking the third-order cross partial derivative with respect to x1, x2, and x3:

```
ginteff, dydxs(x1 x2 x3)
```

Compute the same three-way interaction effect via the first difference approach. Specifically, calculate the effect of increasing x1 by 2 units from its mean, x2 by 8 units from its 25th percentile, and x3 by 1 unit from x3 = 50:

```
ginteff, fd((mean) x1 (p25) x2 x3=50) nunit((2) x1 (8) x2 (1) x3)
```

Compute the two-way interaction effect between x1 and x2 after

```
poisson y c.x1##i.x2
```

for the predicted count, by taking the partial derivative with respect to x1 (a continuous variable), and calculating the discrete change from the base level for x2 (a factor variable):

```
ginteff, dydxs(x1 x2)
```

---

[1] The `ginteff` program and its associated manual come "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the suitability and fitness for a particular purpose. Improvements and/or changes in the product and the program described in this manual may be made at any time and without notice.

[2] The description of the syntax and that of various options borrow heavily, or reproduce excerpts ad litteram, from Stata's official `margins` manual. Instead of referencing that manual repeatedly, this helps make the `ginteff` manual self-contained.

Compute the same two-way interaction effect, but use the first difference to calculate the change in x1. Specifically, calculate the effect of a 0.5 units decrease in x1 raised to the power of 2, and the discrete change from the base level for x2:

```
ginteff, fd(x1=gen(x1^2)) nunit((-0.5) x1) dydxs(x2)
```

Compute the three-way interaction effect between x1, x2, and x3 after

```
tobit y i.x1##i.x2##i.x3, ll(0)
```

for the censored expected value of `y, ystar(0,.)`, *for all combinations between the discrete changes from the base level for x1, x2, and x3:*

```
ginteff, dydxs(x1 x2 x3) predict(ystar(0,.))
```

Syntax

`ginteff` [*if*] [*in*] [*weight*] , *effect_computation* [*options*]

effect_computation	Description
`dydxs(`*dxspec*`)`	specify the interacted variables for which to compute the effect via partial derivative
`fd(`*fdspec*`)`	shorthand for `firstdiff()`
`firstdiff(`*fdspec*`)`	specify the interacted variables for which to compute the effect via first difference

One of `dydxs()` or `firstdiff()` is required. A minimum of two and a maximum of three variables must be specified in `dydxs()` and/or `firstdiff()`.

options	Description
Main	
`atdxs(`*atdxspec*`)`	fix the interacted variables in `dydxs()` to specified values
`nunit((#)` *varlist*`)`	specify the unit increase for each variable in `firstdiff()`
`obseff(`*stub*`)`	create new variable(s) with the interaction effect for each observation
Auxiliary	
`at(`*atspec*`)`	compute the interaction effect at specified values of covariates
`intequation(`*eqno*`)`	identify the interaction equation; default is `intequation(#1)`
`level(#)`	set confidence level; default is `level(95)`
`many`	report more than 100 results; maximum is 1,000
`nolegend`	suppress output legend
`noweights`	ignore weights specified in estimation
`post`	post interaction effects and their VCE as estimation results
`predict(`*pred_opt*`)`	compute the interaction effect for `predict`, *pred_opt*
`vce(`*vcetype*`)`	specify how the VCE and standard errors are calculated; the default is `vce(delta)`

Note: Syntax elements within square brackets [ ] are optional. Underlining indicates minimal abbreviation.

Options:

Main

`atdxs(`*atdxspec*`)` fixes the continuous variables in `dydxs()` at specific values. See the **Syntax of** `atdxs()` section for more information.

`dydxs(`*dxspec*`)` specifies the interacted variables for which the effect is to be computed via the partial derivative. For factor variables, `dydxs()` calculates the discrete change from the base level. See the **Syntax of** `dydxs()` section for more information.

`firstdiff(`*fdspec*`)` specifies the interacted variables for which the effect is to be computed via the first difference, and also sets their starting values. Variables in `firstdiff()` must be continuous. See the **Syntax of** `firstdiff()` section for more information.

`nunit((#)` *varlist*`)` indicates the unit increase for each variable in `firstdiff()`. See the **Syntax of** `nunit()` section for more information.

`obseff(`*stub*`)` creates a new variable containing the interaction effect for each observation in the sample data. The (possibly many) variables are named consecutively, starting with *stub*1. *stub* may not exceed 16 characters in length.

Auxiliary

`at(`*atspec*`)` specifies values for covariates to be treated as fixed. See the **Syntax of** `at()` section for more information. Only one `at()` option can be specified.

`intequation(`*eqno*`)` is relevant only when you have previously fit a multi-equation model, and identifies the interaction equation. For instance, `intequation(#1)` would mean the interacted variables are in the first equation, `intequation(#2)` would mean the second, and so on. You could also refer to the equations by their names. `intequation(`*health*`)` would refer to the equation named *health* and `intequation(`*diabetes*`)` to the equation named *diabetes*. If you do not specify `intequation()`, results are the same as if you specified `intequation(#1)`.

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level`(95) or as set by `set level`.

`many` raises the maximum number of output estimates from 100 to 1,000.

`nolegend` specifies that the legend detailing the `ginteff` output and that showing the fixed values of covariates be suppressed.

`noweights` specifies that any weights specified on the previous estimation command be ignored by `ginteff`. By default, `ginteff` uses the weights specified on the estimator. If weights are specified on the `ginteff` command, they override previously specified weights, making it unnecessary to specify `noweights`.

`post` causes `ginteff` to behave like a Stata estimation (e-class) command. `ginteff` posts the vector of interaction effects along with the estimated variance–covariance matrix to `e()`, so you can treat these estimates just as you would results from any other estimation command. For example, you could use `nlcom` to test whether two interaction effects are statistically different.

`predict(`*pred_opt*`)` specifies the option(s) to be used with the `predict` command to produce the variable that will be used as the response. After estimation by `logistic`, one could specify `predict(xb)` to obtain linear predictions rather than the `predict` command's default, the probabilities. Only one `predict()` option can be specified.

`vce(delta)` and `vce(unconditional)` specify how the VCE and standard errors are calculated.

`vce(delta)` is the default. The delta method is applied to the formula for the response and the VCE of the estimation command. This method assumes that values of the covariates used to calculate the response are given or, if all covariates are not fixed using `at()`, that the data are given.

`vce(unconditional)` specifies that the covariates that are not fixed be treated in a way that accounts for their having been sampled. The VCE is estimated using the linearization method. This method allows for heteroskedasticity or other violations of distributional assumptions and allows for correlation among the observations in the same manner as `vce(robust)` and `vce(cluster ...)`, which may have been specified with the estimation command. This method also accounts for complex survey designs if the data are `svyset`.

Syntax of `at()`

In `at(`*atspec*`)`, *atspec* may contain one or more of the following specifications:

varlist

(stat) varlist

varname = #

varname = (*numlist*)

varname = generate(*exp*)

where

1. Variable names (whether in *varname* or *varlist*) may be continuous variables, factor variables, or specific level variables, such as `age`, `group`, or `3.group`.
2. *varlist* may also be one of the three standard lists:
   a. _all (all covariates),
   b. _factor (all factor-variable covariates), or
   c. _continuous (all continuous covariates).
3. (*stat*) can be any of the following:

stat	Description	Variables allowed
asobserved	at observed values in the sample (default)	all

(continued on next page)

stat	Description	Variables allowed
mean	means (default for *varlist*)	all
median	medians	continuous
p1	1st percentile	continuous
p2	2nd percentile	continuous
...	3rd–49th percentiles	continuous
p50	50th percentile (same as median)	continuous
...	51st–97th percentiles	continuous
p98	98th percentile	continuous
p99	99th percentile	continuous
min	minimums	continuous
max	maximums	continuous
zero	fixed to zero	continuous
base	base level	factors
asbalanced	all levels equally probable and sum to 1	factors

Note: Underlining indicates minimal abbreviation.

When no (*stat*) is specified, (`mean`) is assumed. If (*stat*) is not followed by a *varlist*, (*stat*) is ignored.

The various *stat*s are computed using the estimation sample. Specifically, the value of *x* in option `at((mean) x)`, equals the mean obtain by typing `sum x if e(sample)`. To also include the *x* values of dropped observations (if any) when calculating the mean, type instead

```
. sum x, meanonly
. local m `r(mean)´
. ginteff, at(x=`m´)
```

`at()` cannot be used to set the interacted variables listed in `dydxs()` or `firstdiff()`. If the interacted variables are listed in `at()`, the program will stop and issue an error. The standard variable lists (i.e., `_all`, `_factor`, and `_continuous`) can still be used with `at()`, but they will affect the variables in the respective categories *except* the interacted variables.

Syntax of `atdxs()`

`atdxs(`*atdxspec*`)` can be used only in combination with `dydxs()`, and the variables listed in the two options must match. In other words, only the `dydxs()` variables can be set via `atdxs()`. Save the exceptions below, the specifications of *atdxspec* are identical to those of *atspec* (see the **Syntax of `at()`** section).

The exceptions of *atdxspec*:

1. Factor variables cannot be set via `atdxs()` since the derivative is the discrete *change* from the base level. As a result, factor variables cannot be fixed at specific values or levels. `_factor` variable list is not allowed, but one may still employ the remaining standard lists (i.e., `_all` and `_continuous`). Since factor variables cannot be set via `atdxs()`, specifying either `_all` or `_continuous` produces the same result.

2. If specifying *varname* = #, # must be a single value (i.e., numeric lists are not allowed).

Syntax of `dydxs()`

`dydxs(`*dxspec*`)` specifies the covariates for which the effect is to be computed by partial derivative. Up to three variables can be specified ($xs \in \{x_1, x_2, x_3\}$), to respectively indicate a partial, a second- or a third-order cross partial derivative, i.e., $\frac{\partial y}{\partial x_1}$, $\frac{\partial^2 y}{\partial x_1 \partial x_2}$, or $\frac{\partial^3 y}{\partial x_1 \partial x_2 \partial x_3}$. For factor variables, `dydxs()` calculates the discrete change from the base level.

In `dydxs`(*dxspec*), *dxspec* may contain one or more of the following specifications:

varlist

`j.`*factorvar*

`bk.`*factorvar*

`bk.j.`*factorvar*

where

1. Variable names (whether in *varname* or *varlist*) may be continuous or factor variables that are interacted in the model.

2. In the syntax for factor variables only

   a. `j` and `k` are actual factor level values.

   b. `b` stands for base level.

As the base level, `k` must be a single value. `j` indicates the specific factor levels for which the discrete change is to be calculated, and can be either one value or a list of factor levels separated by a dot (e.g., `1.2.3.`*factorvar*). To illustrate the use of specific factor levels, let us say we have a three-level factor variable, {1, 2, 3}. Assuming 1 is the base level, `dydxs(`*factorvar*`)` calculates two discrete changes, (2 vs 1) and (3 vs 1). Typing `dydxs(3.`*factorvar*`)` calculates a single discrete change, (3 vs 1), as only one level is specified. This is particularly useful when a factor variable has are many levels but the researcher is interested in one particular contrast.

Typing `dydxs(b2.`*factorvar*`)` changes on the fly the base level from 1 to 2, without having to reestimate the model. The two discrete changes are now (1 vs 2) and (3 vs 2). Since `ginteff` automatically calculates the discrete change for all factor levels, typing `dydxs(b2.`*factorvar*`)` produces the same result as `dydxs(b2.1.3.`*factorvar*`)`. By contrast, `dydxs(b2.3.`*factorvar*`)` calculates a single discrete change, (3 vs 2). When resetting the base level that value must be specified first. Thus, `dydxs(3.b2.`*factorvar*`)` is not a valid specification.

Only one argument per interacted variable is allowed. Thus, to examine a subset of factor contrasts, the respective levels must be listed together, e.g., `dydxs(2.3.`*factorvar*`)` and not `dydxs(2.`*factorvar* `3.`*factorvar*`)`.

Syntax of `firstdiff()`

In `firstdiff`(*fdspec*), *fdspec* may contain one or more of the following specifications:

varlist

(fdstat) varlist

varname = #

varname = generate(*exp*)

where

1. Variable names (whether in *varname* or *varlist*) must be continuous variables that are interacted in the model.
2. (#) must be a single value (i.e., numeric lists are not allowed).
3. *fdstat* can be any of the following:

fdstat	Description
asobserved	at observed values in the sample (default)
mean	means
median	medians
p1	1st percentile
p2	2nd percentile
...	3rd–49th percentiles
p50	50th percentile (same as median)
...	51st–97th percentiles
p98	98th percentile
p99	99th percentile
min	minimums
max	maximums
zero	fixed to zero

Note: Underlining indicates minimal abbreviation.

When no (*fdstat*) is specified, (`asobserved`) is assumed. If (*fdstat*) is not followed by a *varlist*, (*fdstat*) is ignored.

The various *fdstat*s are computed using the estimation sample. Specifically, the value of *x* in option `firstdiff((mean) x)`, equals the mean obtain by typing `sum x if e(sample)`. To also include the *x* values of dropped observations (if any) when calculating the mean, type instead

```
. sum x, meanonly
. local m `r(mean)´
. ginteff, firstdiff(x=`m´)
```

Syntax of `nunit()`

`nunit()` can be used only in combination with `firstdiff()`, and the variables listed in the two options must match. `nunit()` takes just one specification

(#) varname

where

1. (#) indicates the unit increase for the respective variable.
2. (#) must be a single value (i.e., numeric lists are not allowed).

3. All or a subset of the interacted variables can be listed after the same (#). Alternatively, separate values for the unit increase can be specified for each individual variable. For example, both `nunit((5) x1 x2 x3)` and `nunit((3) x1 (10) x2 (2) x3)` are valid arguments. The former specification evaluates a 5-unit increase in x_1, x_2, and x_3, whereas the latter a 3-unit increase in x_1, a 10-unit increase in x_2, and a 2-unit increase in x_3.

4. Only one (#) can be applied to a given covariate. If more than one is specified, the rightmost specification is respected. For example, `nunit((1) x1x2 (2) x1x3)` evaluates a 1-unit increase in x_2, and a 2-unit increase in x_1 and x_3.

If `nunit()` is missing, `ginteff` automatically computes the effect of a 1-unit increase for all variables in `firstdiff()`. Thus, `ginteff, fd(x1 x2) nunit((1) x1 x2)` produces the same result as `ginteff, fd(x1 x2)`.

Remarks and examples

The upcoming examples use the data from the Second National Health and Nutrition Examination Survey, available from the StataCorp website (`nhanes2f.dta`). The dependent variable, *health*, is a five-point indicator of respondents' wellbeing (i.e., poor, fair, average, good, and excellent). For Example 1–7, we use a simplified, two-level indicator of health, *health_2l*, which is coded 1 if health is above average (i.e., good or excellent), and 0 otherwise. There are five independent variables; three are continuous (*age*, *height*, and *weight*), and two factors (*female* and *race*). *female* is coded 0 for males, and 1 for females. *race* is a three category variable, where 1 = white, 2 = black, and 3 = other.

Example 1: Compute the average and observation-level interaction effects for factor variables

The first example illustrates how to compute the interaction effect after a logistic regression, with *female* and *race* being the interacted variables. First we upload the data and create the dummy health indicator, and then estimate the model.

```
. webuse nhanes2f, clear

. keep health diabetes race female age height weight

. clonevar health_2l = health
(2 missing values generated)

. recode health_2l (1/3=0) (4/5=1) // two-level health
(health_2l: 10335 changes made)

.
. logit health_2l b0.female##b1.race age height weight, nolog

Logistic regression                             Number of obs     =     10,335
                                                LR chi2(8)        =    1400.39
                                                Prob > chi2       =     0.0000
Log likelihood = -6457.9191                     Pseudo R2         =     0.0978

------------------------------------------------------------------------------
   health_2l |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
    1.female |    .216087   .0628679     3.44   0.001     .0928681    .3393059
             |
        race |
```

```
       Black   -.813307   .1043427    -7.79   0.000    -1.017815   -.6087991
       Other  -.4054997   .2153029    -1.88   0.060    -.8274856    .0164862

 female#race
    1#Black  -.2540739   .1462076    -1.74   0.082    -.5406354    .0324877
    1#Other   .1834579   .3059663     0.60   0.549    -.4162251    .7831409

         age  -.0369468   .0013163   -28.07   0.000    -.0395266   -.0343669
      height   .0335985   .0034593     9.71   0.000     .0268183    .0403786
      weight  -.0096669   .0016101    -6.00   0.000    -.0128227   -.0065112
       _cons   -3.26368   .5924921    -5.51   0.000    -4.424943   -2.102417
```

Next we compute the interaction effect using `ginteff`. In this particular case, we specify three options. In option `dydxs()` we list the two interacted variables, *female* and *race*. Since both are factor variables, their effect is calculated as the discrete change from the base level, or, in Stata's parlance, a contrast. By using option `obseff()`, we instruct `ginteff` to also compute the individual interaction effects for all cases in the data. The observation-level effects are stored in two variables, called *obseff_fr1* and *obseff_fr2*, one for each contrast of *race*. The name of these variables is taken from the stub argument in `obseff(`*obseff_fr*`)`. Lastly, via the `level()` option we change the default 0.05 significance level to 0.1, by requesting 90% CIs around the estimated effects.

```
. ginteff, dydxs(female race) obseff(obs_fr) level(90)

Interaction Effects
    Statistic        :         Average interaction effect
    Standard error   :         Delta-method
    Δ(i.x1)          :         dy/dx w.r.t. x1; x1 : b0.i(1).female
    Δ(i.x2)          :         dy/dx w.r.t. x2; x2 : b1.i(2.3).race
    Number of obs    =         10,335

    Expression       :         Pr(health_21), predict()

                        Statistic     Std. Err.     [90% Conf.     Interval]

Δ(1.x1)#Δ(2.x2)        -.05460184     .02826438     -.10109261     -.00811106
Δ(1.x1)#Δ(3.x2)         .03876595     .06576437      -.0694068      .14693871

Note: dy/dx for factor levels is the discrete change from the base level.

. sum obs_fr*

    Variable        Obs        Mean    Std. Dev.       Min        Max

      obs_fr1     10,335   -.0546018     .006274  -.0618385  -.0247582
      obs_fr2     10,335     .038766    .0088642   .0102487    .048169
```

To keep things concise, we frame the discussion around the effect of gender on health (i.e., the change in the probability of being in good health between a female and a male), attributable to specific changes in racial status. The interaction effect is computed separately for the two contrasts of *race*, using whites as the base category. The first estimate indicates that the effect of gender on health attributable to the difference in racial status between blacks and whites, is negative and statistically significant. Specifically, the probability of being in good health decreases by 0.055 (-0.101, -0.008) percentage points. This means that, on average, black women fare worse than white women. By contrast, women from racial groups other than black may fare better than white women, as the second estimate is positive 0.039 (-0.069, 0.147). But this effect is not statistically significant at the 0.1 significance level, since its CI contains zero.

What about the observation-level effects? Since `ginteff` computes the average interaction effect, the mean of the individual effects should match the default results. This is indeed the case, as the mean of variables *obseff1* and *obseff2* are identical to the first and second `ginteff` point estimates, respectively.

Example 2: Compute the interaction effect for specific factor levels, and alternative `at()` scenarios

In Example 1 the interaction effect is calculated assuming we move from males to females, and from whites to either black or other racial minorities. The respective reference categories for the interacted variables (i.e., male for *gender* and white for *race*), match the base levels from the estimation model. What if the analyst wants to calculate the effect associated with a change in racial status from, let us say, other minorities (race = 3) to black (race = 2), while also switching the reference category for *gender* to female? One approach is to rerun the analysis with the updated base levels (i.e., `logit health_2l b1.female##b3.race age height weight`), and then reissue the `ginteff` command. However, there is an easier way to achieve this. `ginteff` allows the analyst to change on the fly the base level of factors, or request a subset of contrasts. The example below illustrates these features, as well as the use of option `at()`. Researchers can liberally employ the `at()` option to specify any number of relevant scenarios. Here we compute the interaction effect for all possible combinations of the minimum and maximum values of *age*, the 10th and 90th percentiles of *weight*, while *height* is set at its median value.

```
. quietly sum age

. local a1 `r(min)´

. local a2 `r(max)´

. _pctile weight, percentiles(10 90)

. local w1 `r(r1)´

. local w2 `r(r2)´

. ginteff, dydxs(b3.2.race b1.female) at(age=(`a1´ `a2´) (median) height weight=(`w1´ `w2´))

Interaction Effects
    Statistic        :          Average interaction effect
    Standard error   :          Delta-method
    Δ(i.x1)          :          dy/dx w.r.t. x1; x1 : b1.i(0).female
    Δ(i.x2)          :          dy/dx w.r.t. x2; x2 : b3.i(2).race
    Number of obs    =          10,335

    Expression       :          Pr(health_2l), predict()

    1._at            :          age        =          20
                     :          height     =          167.297 (median)
                     :          weight     =          53.52
    2._at            :          age        =          20
                     :          height     =          167.297 (median)
                     :          weight     =          91.63
    3._at            :          age        =          74
                     :          height     =          167.297 (median)
                     :          weight     =          53.52
    4._at            :          age        =          74
```

```
            :          height     =          167.297 (median)
            :          weight     =          91.63
```

	Statistic	Std. Err.	[95% Conf.	Interval]
1._at#Δ(0.x1)#Δ(2.x2)	.09147061	.07089599	-.04748299	.2304242
2._at#Δ(0.x1)#Δ(2.x2)	.10248955	.07815005	-.05068174	.25566083
3._at#Δ(0.x1)#Δ(2.x2)	.0796744	.05976606	-.03746492	.19681373
4._at#Δ(0.x1)#Δ(2.x2)	.06453411	.04867005	-.03085744	.15992565

Note: dy/dx for factor levels is the discrete change from the base level.

Example 3: Compute the interaction effect for continuous variables via the partial derivative

This example illustrates how to compute a three-way interaction effect between continuous variables via the partial derivative. First, we estimate a new model where *age*, *weight*, and *height* are interacted. Second, we issue the `ginteff` command with the respective variables listed in `dydxs()`. By taking the third-order cross partial derivative, we compute the interaction effect attributable to a very small increase in all three variables. We also set factor variables to their respective base levels.

```
. logit health_21 c.age##c.height##c.weight i.female i.race, nolog

Logistic regression                             Number of obs     =     10,335
                                                LR chi2(10)       =    1417.51
                                                Prob > chi2       =     0.0000
Log likelihood = -6449.3627                     Pseudo R2         =     0.0990
```

health_21	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
age	.001678	.1032571	0.02	0.987	-.2007021	.2040582
height	.0253216	.0309574	0.82	0.413	-.0353539	.085997
c.age#c.height	-.0002789	.0006242	-0.45	0.655	-.0015023	.0009444
weight	-.1002978	.070644	-1.42	0.156	-.2387575	.0381618
c.age#c.weight	.0008552	.0014173	0.60	0.546	-.0019227	.0036332
c.height#c.weight	.0005045	.0004184	1.21	0.228	-.0003155	.0013245
c.age#c.height#c.weight	-4.40e-06	8.45e-06	-0.52	0.603	-.000021	.0000122
1.female	.1912482	.0608111	3.14	0.002	.0720605	.3104358
race						
Black	-.9377468	.0736429	-12.73	0.000	-1.082084	-.7934094
Other	-.3235315	.1540024	-2.10	0.036	-.6253707	-.0216924
_cons	-1.482623	5.156587	-0.29	0.774	-11.58935	8.624102

```
. ginteff, dydxs(age height weight) at((base) _factor)

Interaction Effects
    Statistic        :          Average interaction effect
    Standard error   :          Delta-method
    Δ(x1)            :          dy/dx w.r.t. x1; x1 : age (asobserved)
```

```
    Δ(x2)            :         dy/dx w.r.t. x2; x2 : height (asobserved)
    Δ(x3)            :         dy/dx w.r.t. x3; x3 : weight (asobserved)
    Number of obs    =         10,335

    Expression       :         Pr(health_21), predict()

    at               :         female     =          0
                     :         race       =          1
```

	Statistic	Std. Err.	[95% Conf.	Interval]
Δ(x1)#Δ(x2)#Δ(x3)	-1.704e-06	8.473e-07	-3.364e-06	-4.291e-08

Example 4: Compute the interaction effect for continuous variables via the first difference

Alternatively, we can compute the three-way interaction effect from Example 3 using the first difference approach. In this exercise we compute the interaction effect attributable to a 1-unit increase in all three variables from their observed values. Factor variables are still set at their base level. Specifying `nunit()` is optional when all variables listed in `firstdiff()` are to be increased by 1-unit. Spelling out the specific unit increase by typing `nunit((1) age height weight)`, would lead to the same result.

```
. ginteff, firstdiff(age height weight) at((base) _factor)

Interaction Effects
    Statistic        :         Average interaction effect
    Standard error   :         Delta-method
    Δ(x1)            :         (y|x1+n1)-(y|x1); x1 : age (asobserved), n1 = 1
    Δ(x2)            :         (y|x2+n2)-(y|x2); x2 : height (asobserved), n2 = 1
    Δ(x3)            :         (y|x3+n3)-(y|x3); x3 : weight (asobserved), n3 = 1
    Number of obs    =         10,335

    Expression       :         Pr(health_21), predict()

    at               :         female     =          0
                     :         race       =          1
```

	Statistic	Std. Err.	[95% Conf.	Interval]
Δ(x1)#Δ(x2)#Δ(x3)	-1.684e-06	1.428e-06	-4.483e-06	1.115e-06

Example 5: Compute the interaction effect for continuous variables via the first difference, cont.

The `firstdiff()` option, or `fd()` for short, can be used to compute far more complex scenarios than a uniform 1-unit increase in all interacted variables. For this exercise, we compute the interaction effect as *age* increases from mean to one standard deviation above the mean, *height* from min to max, and *weight* decreases from its 50th percentile (median) to 10 units below median. Since the factor variables are no longer specified, they are now set to their observed values (the default). Lastly, the `nolegend` option (`nol` for short) suppresses the legend detailing the `ginteff` output and that showing the fixed values of covariates.

```
. quietly sum age

. local a `r(sd)´

. quietly sum height

. local h = r(max)-r(min)
```

```
. ginteff, fd((mean) age (min) height (p50) weight) nunit((`a´) age (`h´) height (-10) weight) nol
Interaction Effects
```

	Statistic	Std. Err.	[95% Conf.	Interval]
Δ(x1)#Δ(x2)#Δ(x3)	.01436578	.01974282	-.02432943	.05306099

Example 6: Compute the interaction effect for multi-equation models

The above exercises are based on a simple logistic regression. But `ginteff` can accommodate more complex models, including multi-equation ones. Let us consider a bivariate probit with two seemingly unrelated equations. Specifically, we simultaneously estimate the probability of being in good health and having diabetes, which should be negatively correlated. *female* and *race* are interacted in the health equation, and *age* and *female* in the diabetes equation.

```
. biprobit (health_21 = i.female##i.race age) (diabetes = c.age##i.female), nolog

Seemingly unrelated bivariate probit            Number of obs     =     10,335
                                                Wald chi2(9)      =    1395.30
Log likelihood = -8268.4552                     Prob > chi2       =     0.0000
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
health_21						
1.female	-.0721203	.0272677	-2.64	0.008	-.1255639	-.0186766
race						
Black	-.4591418	.0622029	-7.38	0.000	-.5810572	-.3372264
Other	-.3279656	.1293722	-2.54	0.011	-.5815305	-.0744006
female#race						
1#Black	-.1889397	.0863215	-2.19	0.029	-.3581268	-.0197527
1#Other	.1192973	.184136	0.65	0.517	-.2416026	.4801972
age	-.0249512	.0007582	-32.91	0.000	-.0264371	-.0234652
_cons	1.243855	.0415001	29.97	0.000	1.162516	1.325193
diabetes						
age	.0321037	.0027224	11.79	0.000	.0267679	.0374396
1.female	.5661857	.2036856	2.78	0.005	.1669693	.9654021
female#c.age						
1	-.0082077	.0033993	-2.41	0.016	-.0148703	-.0015452
_cons	-3.461487	.1652544	-20.95	0.000	-3.78538	-3.137594
/athrho	-.3574045	.0326712	-10.94	0.000	-.4214389	-.2933702
rho	-.3429258	.0288291			-.3981419	-.2852338

```
Wald test of rho=0: chi2(1) = 119.671                     Prob > chi2 = 0.0000
```

After estimating the model, we first compute the interaction effect between *female* and *race*. Option `intequation()` is used to identify in which of the two equations the specific interaction is. In this case, it is the first equation. This is a necessary step because `ginteff` checks that the specified variables are actually interacted and there are no missing terms. The same variable (e.g., *female* in our example), can have different interacting terms in different equations. Equations can be specified by using either their position number, or the name of the respective dependent variable. If `intequation()` is not specified, equation #1 is assumed. Lastly, option `predict(p11)` instructs `ginteff` to compute the effect on the bivariate predicted probability that both *health_2l* and *diabetes* equal 1, Pr(*health_2l* = 1, *diabetes* = 1). (Biprobit's specialized `predict()` suboptions are detailed in the model's postestimation manual; see `https://www.stata.com/manuals/rbiprobitpostestimation.pdf`).

```
. ginteff, dydxs(female race) predict(p11) inteq(#1)

Interaction Effects
    Statistic        :          Average interaction effect
    Standard error   :          Delta-method
    Δ(i.x1)          :          dy/dx w.r.t. x1; x1 : b0.i(1).female
    Δ(i.x2)          :          dy/dx w.r.t. x2; x2 : b1.i(2.3).race
    Number of obs    =          10,335

    Expression       :          Pr(health_2l=1,diabetes=1), predict(p11)
```

	Statistic	Std. Err.	[95% Conf.	Interval]
Δ(1.x1)#Δ(2.x2)	-.00178306	.00076936	-.00329098	-.00027513
Δ(1.x1)#Δ(3.x2)	.00065051	.00167562	-.00263363	.00393466

```
Note: dy/dx for factor levels is the discrete change from the base level.
```

Example 7: Compute the interaction effect for multi-equation models, cont.

In this example, we compute the interaction effect between *age* and *female*, which are interacted in the second equation. Specifying `predict()` with suboption pmarg2 indicates that we want to estimate the effect of the simultaneous change in *age* and *female* on the univariate (marginal) predicted probability of success in the second equation, Pr(*diabetes* = 1).

```
. ginteff, dydxs(age female) predict(pmarg2) inteq(diabetes)

Interaction Effects
    Statistic        :          Average interaction effect
    Standard error   :          Delta-method
    Δ(i.x1)          :          dy/dx w.r.t. x1; x1 : b0.i(1).female
    Δ(x2)            :          dy/dx w.r.t. x2; x2 : age (asobserved)
    Number of obs    =          10,335

    Expression       :          Pr(diabetes=1), predict(pmarg2)
```

	Statistic	Std. Err.	[95% Conf.	Interval]
Δ(1.x1)#Δ(x2)	-.00035901	.00033591	-.00101739	.00029937

```
Note: dy/dx for factor levels is the discrete change from the base level.
```

Example 8: Compute the interaction effect for models with a polychotomous dependent variable

The last two examples concern a scenario where we have a polychotomous dependent variable. Specifically, for these exercises we use the original five-level *health* variable, and the estimation model is ordered logit.

Notably, `ginteff` computes the interaction effect between *age* and *female* separately for each level of *health*. In this exercise we compute the partial effect of *age* when this variable is set at mean. The default is to set the continuous variables listed in `dydxs()` at their observed values. Using option `atdxs()`, the analyst can set these variables to different values.

```
. ologit health c.age##i.female, nolog

Ordered logistic regression                     Number of obs     =     10,335
                                                LR chi2(3)        =    1465.82
                                                Prob > chi2       =     0.0000
Log likelihood = -15031.489                     Pseudo R2         =     0.0465

------------------------------------------------------------------------------
      health |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |  -.0441621   .0015608   -28.29   0.000    -.0472212   -.0411029
    1.female |  -.4835603   .1039157    -4.65   0.000    -.6872313   -.2798894
             |
female#c.age |
           1 |    .007615   .0020643     3.69   0.000     .0035692    .0116609
-------------+----------------------------------------------------------------
       /cut1 |  -4.931698    .090312                     -5.108706    -4.75469
       /cut2 |  -3.467161   .0829925                     -3.629824   -3.304499
       /cut3 |   -2.06566   .0784233                     -2.219367   -1.911953
       /cut4 |   -.828686   .0763793                     -.9783867   -.6789852
------------------------------------------------------------------------------

. ginteff, dydxs(age female) atdxs((mean) age)

Interaction Effects
    Statistic       :          Average interaction effect
    Standard error  :          Delta-method
    Δ(i.x1)         :          dy/dx w.r.t. x1; x1 : b0.i(1).female
    Δ(x2)           :          dy/dx w.r.t. x2; x2 : age at (mean)
    Number of obs   =          10,335

    1._pr           :          Pr(health==1), predict(pr outcome(1))
    2._pr           :          Pr(health==2), predict(pr outcome(2))
    3._pr           :          Pr(health==3), predict(pr outcome(3))
    4._pr           :          Pr(health==4), predict(pr outcome(4))
    5._pr           :          Pr(health==5), predict(pr outcome(5))

--------------------------------------------------------------------------------------
                      |    Statistic       Std. Err.      [95% Conf.         Interval]
----------------------+---------------------------------------------------------------
1._pr#Δ(1.x1)#Δ(x2)   |    -.00018327      .00013524     -.00044833          .0000818
2._pr#Δ(1.x1)#Δ(x2)   |    -.00062313       .0002427     -.00109881        -.00014745
3._pr#Δ(1.x1)#Δ(x2)   |    -.00114956      .00022866     -.00159772         -.0007014
4._pr#Δ(1.x1)#Δ(x2)   |     .00022737      .00021533     -.00019467          .0006494
5._pr#Δ(1.x1)#Δ(x2)   |     .00172859      .00036701      .00100926         .00244793
--------------------------------------------------------------------------------------
Note: dy/dx for factor levels is the discrete change from the base level.
```

Example 9: Compute the interaction effect for models with a polychotomous dependent variable, cont.

When there are many prediction levels, the output may become intractable (especially if we also specify multiple `at()` scenarios). To focus on a given prediction, we can explicitly request a given outcome. Out of the five *health* levels, in this exercise we compute the interaction effect for the second outcome (*health* = 2). As expected, the result matches the second effect from the full output in Exercise 8.

```
. ginteff, dydxs(age female) atdxs((mean) age) predict(outcome(#2))

Interaction Effects
    Statistic       :         Average interaction effect
    Standard error  :         Delta-method
    Δ(i.x1)         :         dy/dx w.r.t. x1; x1 : b0.i(1).female
    Δ(x2)           :         dy/dx w.r.t. x2; x2 : age at (mean)
    Number of obs   =         10,335

    Expression      :         Pr(health==2), predict(outcome(#2))
```

	Statistic	Std. Err.	[95% Conf.	Interval]
Δ(1.x1)#Δ(x2)	-.00062313	.0002427	-.00109881	-.00014745

```
Note: dy/dx for factor levels is the discrete change from the base level.
```

Stored results

`ginteff` stores the following in `r()`:

Scalars

`r(df_r)`	variance degrees of freedom, survey data only
`r(level)`	confidence level of confidence intervals
`r(N)`	number of observations
`r(N_psu)`	number of sampled PSUs, survey data only
`r(N_strata)`	number of strata, survey data only

Macros

`r(atstat)`	the `at()` specification
`r(cmd)`	`ginteff`
`r(cmdline)`	command as typed
`r(est_cmd)`	`e(cmd)` from original estimation results
`r(est_cmdline)`	`e(cmdline)` from original estimation results
`r(fdstat)`	the `firstdiff()` specification
`r(model_vce)`	*vcetype* from estimation command
`r(obseff)`	the list of new variable(s) created because of the `obseff()` option
`r(vce)`	*vcetype* specified in `vce()`

Matrices

`r(at)`	matrix of values from the `at()` option
`r(b)`	the interaction effect estimates
`r(fd)`	matrix of values from the `firstdiff()` option
`r(ginteff)`	matrix containing the average interaction effects with their standard errors, test statistics, *p*-values, upper and lower confidence limits, and critical values
`r(nunit)`	matrix of values from the `nunit()` option

r(V)	variance–covariance matrix of the interaction effect estimates

ginteff with the post option also stores the following in e():

Scalars

e(df_r)	variance degrees of freedom, survey data only
e(N)	number of observations
e(N_psu)	number of sampled PSUs, survey data only
e(N_strata)	number of strata, survey data only

Macros

e(cmd)	ginteff
e(properties)	b V

Matrices

e(b)	estimates
e(V)	variance–covariance matrix of the estimates

Functions

e(sample)	marks estimation sample