The `sdi` Users' Manual

Description

`sdi` is a user-written program for Stata that computes significance of differences intervals (SDIs), which can visually indicate whether two points estimates are statistically or substantively distinct.[1]

Quick start

Estimated SDIs to compare predicted values

Estimate SDIs to compare the predicted probability at $x = 10$ and the predicted probability when x increases by 5 units (i.e., $x = 15$), after `logit y x`.

```
sdi, xofinterest(x=10) nunit(5)
```

Same as above but holding the other covariates (i.e, a and b) at their means, after `logit y x a b`.

```
sdi, xofinterest(x=10) nunit(5) atmeans
```

Same as above but separately for all possible combinations of $a = 50$, 100, and $b = 10$, 20, 30.

```
sdi, xofinterest(x=10) nunit(5) at(a=(50 100) b=(10(10)30))
```

Estimate SDIs to compare the predicted probability when x is set at mean and at one standard deviation above mean, respectively.

```
sdi, xofinterest((mean) x) nunit(sd)
```

Estimating SDIs to compare contrasts (differences) of predictions with `firstdiff`

Estimate SDIs to compare the effect of a 1-unit increase in x (i.e., $[\Pr(y|x+1) - \Pr(y|x)]$ a.k.a. first difference) across the values of the moderating variable z, after `logit y c.x##i.z`. Here z is a dummy variable, but it can also be a multi-level factor variable, or a continuous variable.

```
sdi, xofinterest((asobserved) x) nunit(1) across(z=(0 1)) firstdiff
```

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

The `sdi` Command Syntax

`sdi` [*if*] [*in*] [*weight*] , xofinterest(*xofispec*) [*options*]

options	Description
`across(`*acrspec*`)`	estimate `sdi` at specified values of the individual covariate
`asbalanced`	treat all factor variables as balanced
`atmeans`	estimate `sdi` at the means of covariates
`at(`*atspec*`)`	specify the values at which covariates be fixed
`difference`	report the difference in estimates
`expression(`*pnl_exp*`)`	estimate `sdi` for *pnl_exp*
`fd`	same as `firstdiff`
`firstdiff`	estimate `sdi` for contrasts rather than predicted values
`level(#)`	set confidence level; default is `level`(95)
`many`	report more than 100 results; maximum is 1,000
`mvalue(#)`	meaningful value *m*; default is 0
`nolegend`	suppress legend of fixed covariate values
`nonotes`	suppress the notes
`noweights`	ignore weights specified in estimation
`nunit(`# or "sd"`)`	specify the unit increase in *xofivar*
`precision(#)`	set level of precision for SDIs, {0,1,2,3,4,5,6}; default is 1
`predict(`*pred_opt*`)`	estimate `sdi` for `predict`, *pred_opt*
`range`	estimate SDIs for the min and max values comparison only
`xofinterest(`*xofispec*`)`	estimate `sdi` at the specified values of the individual covariate

Note: Underlining indicates minimal abbreviation.

Options:

`asbalanced` is shorthand for `at((asbalanced) _factor)` and specifies that factor covariates be evaluated as though there were an equal number of observations in each level. `asbalanced` differs from `at((asbalanced) _factor)` in that `asbalanced` will affect subsequent `at()` options.

`atmeans` specifies that covariates be fixed at their means and is shorthand for `at((mean) _all)`. `atmeans` differs from `at((mean) _all)` in that atmeans will affect subsequent `at()` options.

`at(`*atspec*`)` specifies values for covariates to be treated as fixed. Separate resuts are produced for all possible combinations of the specified values. Only one `at()` option is allowed. See **Syntax of** `at()` subsection for more information.

`at((median) a b=5 c=(40(10)80))` simultaneously fixes covariate *a* at its median value, *b* at 5, and fixes *c* first at 40, then at 50, . . . , and finally at 80 in increments of 10.

`across(`*acrspec*`)` specifies values for the *acrvar* covariate (i.e., the variable in `acr()`) to be treated as fixed. *acrvar* would typically indicate a covariate that is interacted with the variable in `xofinterest()`, such that one can estimate the effect of an *n*-unit increase in *x* across the range of *acrvar*. See **Syntax of** `across()` subsection for more information.

`xofinterest(`*xofispec*`)` specifies values for the *xofivar* covariate (i.e., the variable in `xofi()`) to be treated as fixed. See **Syntax of** `xofinterest()` subsection for more information.

`difference` reports the difference between the compared estimates with its CI. If the standard CI contains zero, it means that the two estimates are not statistically different and the SDIs must overlap. Conversely, if the estimates are distinct, the associated SDIs do not overlap and the CI of the difference does not contain zero.

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

`mvalue(#)` estimates SDIs that can indicate differences other than zero. When the difference between two estimates is different from zero, the estimates are statistically distinct. However, small effects can have little practical significance and researchers need to check whether the reported effects are also substantively meaningful (not just statistically significant). While context specific, theory may be used to derive the value of this meaningful difference, *m*. By typing `mvalue(1.3)` we practically test whether the difference between the compared estimates is larger than 1.3. *m* can also take negative values, if, for instance, we compare two negative estimates. By default the meaningful value *m* is set to zero.

`firsdiff` allows users to compare contrasts of predictions (using the Stata parlance) rather than predicted values. Here contrasts can be thought of as first differences, or the difference between two predicted values. An example may help clarify the difference between the two concepts. If we want to examine the effect of gender on the probability of being promoted, we would compare two predicted values: the probability of being promoted for male and females. If we want to examine whether effect of gender is different for minorities and the majority group, we would compare two contrasts. Specifically, we first compute the difference between male and female predictions for both the majority and minority groups, and then compare the resulting contrasts or differences.

`many` allows users to estimate up to 1,000 individual results. By default the `sdi` output cannot have more than 100 estimates. This restriction is to ensure the output is tractable. `sdi` automatically estimates SDIs for all pairwise comparisons with a given scenario, and combinations of two compound really fast.

`nolegend` specifies that the legend showing the fixed values of covariates be suppressed.

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

`nunit(`# or “sd”`)` indicates the specific *n*-unit increase in `xofi()`, whose effect is to be evaluated. Its argument can be either numeric or string. The numeric argument must be a single number (i.e., multiple values or numeric lists are not allowed). The string argument can be either *sd* or $-$*sd*, which evaluates the effect of one standard deviation increase (decrease) in *xofivar*. This option can be used only in conjunction with `xofi()`.

`precision(#)` specifies the decimal precision of reported SDIs, ranging from no decimals to up to six decimals (i.e., # $\in$ {0,1,2,3,4,5,6}). The difference between typing `precision(0)` or `precision(3)`, is that an 85-ish% SDI will be rounded to either 85% or 85.xxx%, respectively. The default is to report one digit precision SDIs.

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

`range` estimates only the SDI pair necessary to compare the quantities of interest at the minimum and maximum values within a given scenario. This option is designed to be used when graphing the results. Let us say we have a nonlinear interaction model, $y = x \times z$. To test whether the effect of x is conditional on z, we may assess whether the effect of x is different at the min and max values of z. To compare the two marginal effects we could type `sdi, xofi((asobserved)` *x*`) nunit(1) across(`*z*`=(0 100)) fd`. This estimates the effect of a 1-unit increase in x when z is 0 and 100. Alternatively, we could type `sdi, xofi((asobserved)` *x*`) nunit(1) across(`*z*`=(0(4)100)) fd range`. Without the `range` option, we would have 325 pairwise comparisons between the 26 estimates at each value of z. Arguably, this is not exactly what we want. With `range`, `sdi` estimates only the SDIs required to compare the effect of x at min and max values of z, but on the side it also computes the marginal effect of x at the 24 in-between z values. These estimates can then be used to plot the non-constant effect of x across the entire range of z.

Syntax of `at()`

atspec may contain one or more of the following specifications:

varname

(stat) varname

varname = #

varname = (*numlist*)

varname = generate(*exp*)

where

1. *varname*s must be covariates in the previously fit model (estimation command).
2. Variable names (whether in *varname* or *varlist* may be continuous variable, factor variables, or specific level variables.)
3. *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).
4. *stat* can be any of the following:

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

Any *stat* except `zero`, `base`, and `asbalanced` may be prefixed with an `o` to get the overall statistic–the sample over all `over()` groups. For example, `omean`, `omedian`, and `op25`. Overall statistics differ from their correspondingly named statistics only when either the `across(o.())` or `over()` option is specified. When no *stat* is specified, mean is assumed. If *stat* is not followed by a *varlist*, *stat* is ignored.

Syntax of `across()`

The syntax of `across()` is the same as the syntax of `at()`, with the exception that only one variable can be specified (*acrvar*).

`across()`, however, comes with three additional suboptions. First, if a factor variable is typed with the `i.` prefix, `across()` automatically retrieves all its levels. Practically, for a five category variable z, typing `across(i.`z`)` is equivalent to typing `across(`z `=(`*a, b, c, d, e*`))`. Second, if a positive integer variable is typed with the `o.` prefix, separate sets of SDIs are estimated for the groups defined by the levels of z. Using a `margins` analogy, the difference between `across(i.`z`)` and `across(o.`z`)` is similar to the difference between the output of `margins, at(`z `=(`*a, b, c, d, e*`))` and that of `margins, over(`z`)`.

Third, one can simultaneously specify two value scenarios for *acrvar*, via the `(asobserved)` and `generate(`*exp*`)` specifications. For example, to estimate SDI at the observed *acrvar* values and at the observed values plus a given unit increase, one could type `across((asobserved)` z z `=gen(`z`+#))`. In a typical `at()` environment, latter specifications (i.e., the ones more to the right) override previous ones and, as a result, one cannot specify two value scenarios for the same variable. For instance, in this particular example the `((asobserved)` z`)` specification would be ignored. This suboption is particularly useful to assess interaction effects, as one can directly compare the marginal effect of x at z and $z+1$ observed values of the modifying variable.

Syntax of `xofinterest()`

The syntax of `xofi()` is the same as the syntax of `at()`, with two notable exceptions. Specifically, only one variable is allowed, and this variable must be continuous. As a result, all `at()` specifications that apply solely to factor variables are not allowed with `xofi()`.