Title

stata.com

dydx — Calculate numeric derivatives and integrals

Syntax

Derivatives of numeric functions

dydx *yvar xvar* [*if*] [*in*] , generate(*newvar*) [*dydx_options*]

Integrals of numeric functions

integ *yvar xvar* [*if*] [*in*] [, *integ_options*]

dydx_options	Description
Main	
* generate(*newvar*)	create variable named *newvar*
replace	overwrite the existing variable

*generate(*newvar*) is required.

integ_options	Description
Main	
generate(*newvar*)	create variable named *newvar*
trapezoid	use trapezoidal rule to compute integrals; default is cubic splines
initial(#)	initial value of integral; default is initial(0)
replace	overwrite the existing variable

by is allowed with dydx and integ; see [D] **by**.

Menu

dydx

Data > Create or change data > Other variable-creation commands > Calculate numerical derivatives

integ

Data > Create or change data > Other variable-creation commands > Calculate numeric integrals

Description

dydx and integ calculate derivatives and integrals of numeric "functions".

Options

Main

generate(*newvar*) specifies the name of the new variable to be created. It must be specified with dydx.

trapezoid requests that the trapezoidal rule [the sum of $(x_i - x_{i-1})(y_i + y_{i-1})/2$] be used to compute integrals. The default is cubic splines, which give superior results for most smooth functions; for irregular functions, trapezoid may give better results.

initial(*#*) specifies the initial condition for calculating definite integrals; see *Methods and formulas* below. The default is initial(0).

replace specifies that if an existing variable is specified for generate(), it should be overwritten.

Remarks and examples

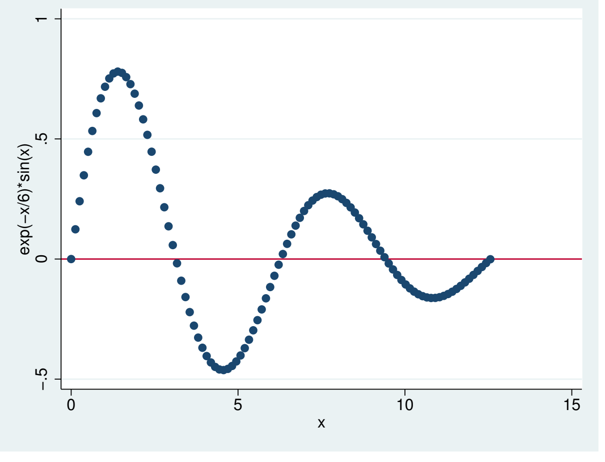
stata.com

dydx and integ lets you extend Stata's graphics capabilities beyond data analysis and into mathematics. (See Gould [1993] for another command that draws functions.)

▷ Example 1

We graph $y = e^{-x/6}\sin(x)$ over the interval $[0, 12.56]$:

```
. range x 0 12.56 100
obs was 0, now 100
. generate y = exp(-x/6)*sin(x)
. label variable y "exp(-x/6)*sin(x)"
. twoway connected y x, connect(i) yline(0)
```

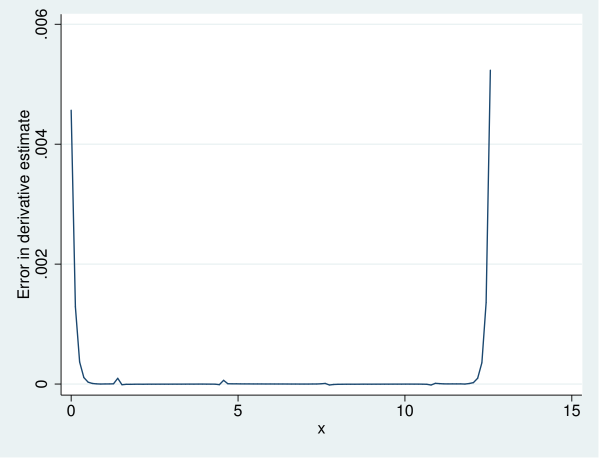


We estimate the derivative by using dydx and compute the relative difference between this estimate and the true derivative.

```
. dydx y x, gen(dy)
. generate dytrue = exp(-x/6)*(cos(x) - sin(x)/6)
. generate error = abs(dy - dytrue)/dytrue
```

The error is greatest at the endpoints, as we would expect. The error is approximately 0.5% at each endpoint, but the error quickly falls to less than 0.01%.

```
. label variable error "Error in derivative estimate"
. twoway line error x, ylabel(0(.002).006)
```

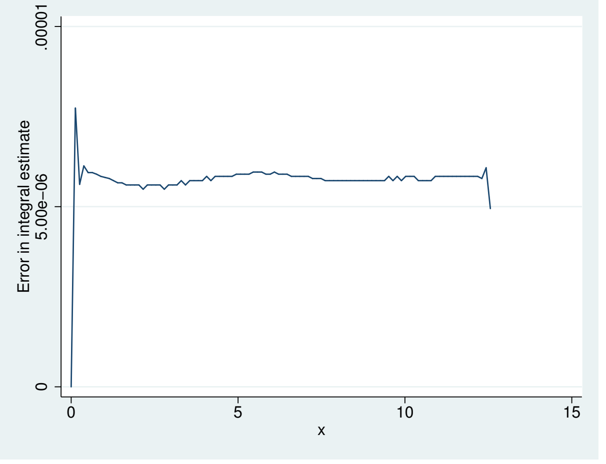


We now estimate the integral by using integ:

```
. integ y x, gen(iy)
number of points = 100
integral         = .85316396
. generate iytrue = (36/37)*(1 - exp(-x/6)*(cos(x) + sin(x)/6))
. display iytrue[_N]
.85315901
. display abs(r(integral) - iytrue[_N])/iytrue[_N]
5.799e-06
. generate diff = iy - iytrue
```

The relative difference between the estimate [stored in r(integral)] and the true value of the integral is about 6×10^{-6}. A graph of the absolute difference (diff) is shown below. Here error is cumulative. Again most of the error is due to a relatively poorer fit near the endpoints.

```
. label variable diff "Error in integral estimate"
. twoway line diff x, ylabel(0(5.00e-06).00001)
```



◁

Stored results

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

Macros

`r(y)`	name of *yvar*

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

Scalars

`r(N_points)`	number of unique x points
`r(integral)`	estimate of the integral

Methods and formulas

Consider a set of data points, $(x_1, y_1), \ldots, (x_n, y_n)$, generated by a function $y = f(x)$. `dydx` and `integ` first fit these points with a cubic spline, which is then analytically differentiated (integrated) to give an approximation for the derivative (integral) of f.

The cubic spline (see, for example, Press et al. [2007]) consists of $n-1$ cubic polynomials $P_i(x)$, with the ith one defined on the interval $[x_i, x_{i+1}]$,

$$P_i(x) = y_i a_i(x) + y_{i+1} b_i(x) + y_i'' c_i(x) + y_{i+1}'' d_i(x)$$

where

$$a_i(x) = \frac{x_{i+1} - x}{x_{i+1} - x_i} \qquad b_i(x) = \frac{x - x_i}{x_{i+1} - x_i}$$

$$c_i(x) = \frac{1}{6}(x_{i+1} - x_i)^2 a_i(x)[\{a_i(x)\}^2 - 1] \qquad d_i(x) = \frac{1}{6}(x_{i+1} - x_i)^2 b_i(x)[\{b_i(x)\}^2 - 1]$$

and y_i'' and y_{i+1}'' are constants whose values will be determined as described below. The notation for these constants is justified because $P_i''(x_i) = y_i''$ and $P_i''(x_{i+1}) = y_{i+1}''$.

Because $a_i(x_i) = 1$, $a_i(x_{i+1}) = 0$, $b_i(x_i) = 0$, and $b_i(x_{i+1}) = 1$. Therefore, $P_i(x_i) = y_i$, and $P_i(x_{i+1}) = y_{i+1}$. Thus the P_i jointly define a function that is continuous at the interval boundaries. The first derivative should be continuous at the interval boundaries; that is,

$$P_i'(x_{i+1}) = P_{i+1}'(x_{i+1})$$

The above $n-2$ equations (one equation for each point except the two endpoints) and the values of the first derivative at the endpoints, $P_1'(x_1)$ and $P_{n-1}'(x_n)$, determine the n constants y_i''.

The value of the first derivative at an endpoint is set to the value of the derivative obtained by fitting a quadratic to the endpoint and the two adjacent points; namely, we use

$$P_1'(x_1) = \frac{y_1 - y_2}{x_1 - x_2} + \frac{y_1 - y_3}{x_1 - x_3} - \frac{y_2 - y_3}{x_2 - x_3}$$

and a similar formula for the upper endpoint.

`dydx` approximates $f'(x_i)$ by using $P_i'(x_i)$.

`integ` approximates $F(x_i) = F(x_1) + \int_{x_1}^{x_i} f(x)\,dx$ by using

$$I_0 + \sum_{k=1}^{i-1} \int_{x_k}^{x_{k+1}} P_k(x)\,dx$$

where I_0 (an estimate of $F(x_1)$) is the value specified by the `initial(#)` option. If the `trapezoid` option is specified, `integ` approximates the integral by using the trapezoidal rule:

$$I_0 + \sum_{k=1}^{i-1} \frac{1}{2}(x_{k+1} - x_k)(y_{k+1} + y_k)$$

If there are ties among the x_i, the mean of y_i is computed at each set of ties and the cubic spline is fit to these values.

Acknowledgment

The present versions of `dydx` and `integ` were inspired by the `dydx2` command written by Patrick Royston of the MRC Clinical Trials Unit, London, and coauthor of the Stata Press book *Flexible Parametric Survival Analysis Using Stata: Beyond the Cox Model*.

References

Gould, W. W. 1993. ssi5.1: Graphing functions. *Stata Technical Bulletin* 16: 23–26. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 188–193. College Station, TX: Stata Press.

——. 1997. crc46: Better numerical derivatives and integrals. *Stata Technical Bulletin* 35: 3–5. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 8–12. College Station, TX: Stata Press.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 2007. *Numerical Recipes: The Art of Scientific Computing*. 3rd ed. New York: Cambridge University Press.

Also see

[D] **obs** — Increase the number of observations in a dataset

[D] **range** — Generate numerical range