dstat: A new command for the analysis of distributions

Ben Jann

University of Bern

2021 Stata Conference
Virtual, August 5–6, 2021

Outline

Why this command?

- Stata provides great functionality for advanced statistics and econometric analyses.
- However, Stata sometimes appears a bit weak in terms of descriptive statistics.
- What do I mean by descriptive statistics?
  - Statistics describing points in a distribution; series of such statistics illustrate the shape of a distribution.
    - ★ Densities, cumulative distributions, histograms, probabilities, quantiles, lorenz ordinates, etc.
  - Summary statistics describing particular features of a distribution.
    - ★ Various types of location, dispersion, skewness, or kurtosis measures.
    - ★ Inequality, concentration, and poverty measures.
    - ★ Association measures
    - ★ ...?
  - I primarily mean univariate statistics (although some of the measures covered by `dstat` are bivariate and `dstat` has some other features to support "multivariate" analyses).

What do I mean by "weak"?

- With "weak" I do not mean that descriptive statistics cannot be computed in Stata. I just mean that things could be a bit improved in terms of functionality and convenience.
  - Different types of statistics are scattered across various commands.
  - Each command has its own logic (idiosyncratic syntax, idiosyncratic output, idiosyncratic returns).
  - Support for statistical inference greatly varies (some commands do not even allow weights, others fully support complex survey estimation).
  - Often difficult to create tables and graphs without too much effort (particularly if interested in confidence intervals).
  - Some topics such as, e.g., inequality measures are not covered at all in official Stata. In general, there is a large number of user add-ons for descriptive statistics which, again, all have their own idiosyncrasies, and also greatly vary in terms of functionality and quality.

Guiding principles for the new command

- Descriptive statistics are estimates and should be treated as such.
  - Provide standard errors/confidence intervals for everything.
  - Standardized output like any other estimation command.
  - Return results in `e()` like any other estimation command.
- Highly standardized and consistent, but as flexible as possible.
- Graphs are important in descriptive analysis; provide convenient graphing functionality.
- Framework should be easy to extend (integrate further statistics without too much effort).
- Personal interest: Provide support for covariate balancing a.k.a. compositional standardization ($\rightarrow$ treatment effect estimation, counterfactual decompositions).

Moment conditions and influence functions

- Think of statistic θ as a functional of a distribution F, that is $\theta = T(F)$.
- The influence function of θ is defined as the limit of the change in θ if a small amount of data mass is added at a specific point in the distribution:

$$\text{IF}(x, \theta, F) = \lim_{\epsilon \to 0} \frac{T((1-\epsilon)F + \epsilon\delta_x) - T(F)}{\epsilon}$$

  where δ_x is a distribution with all its mass at point x.
- Influence functions have been developed in robust statistics (e.g. Hampel 1974) to study the robustness properties of estimators.
- However, influence functions are super useful because the sampling variance of an estimator is equal to the sampling variance of the expected value of its influence function (e.g. Deville 1999)
- This means that the standard error of a mean estimate of the "empirical" influence function provides a consistent estimate of the standard error of $\hat{\theta}$.

Moment conditions and influence functions

- But how to compute influence functions in practice?
- Let h_i^θ be the moment condition of θ such that

$$\frac{1}{W}\sum_{i=1}^{n} w_i h_i^{\hat{\theta}} = 0$$

  where w_i are sampling weights and W is the sum of weights.
- For example, the moment condition is as follows:

$$\bar{y} = \frac{1}{W}\sum_{i=1}^{N} w_i Y_i \quad \Rightarrow \quad \frac{1}{W}\sum_{i=1}^{N} w_i (Y_i - \bar{y}) = 0 \quad \Rightarrow \quad h_i^{\bar{y}} = Y_i - \bar{y}$$

- The "empirical" influence function can then be obtained as

$$\text{IF}_i(\hat{\theta}) = \frac{1}{G} h_i^{\hat{\theta}} \qquad \text{with} \quad G = -\frac{1}{W}\sum_{i=1}^{n} w_i \left.\frac{\partial h_i^\theta}{\partial \theta}\right|_{\theta=\hat{\theta}}$$

- In case of the mean, G boils down to 1, such that the influence function simply is $\text{IF}_i(\bar{y}) = Y_i - \bar{y}$.

Moment conditions and influence functions

- Many statistics are constructed in a way such that they depend on a number of auxiliary estimates. For example, the trimmed mean depends on the quantiles at which the data is trimmed.
- Influence functions for such statistics can be derived using the chain rule.
- Let θ depend on additional parameters $\gamma_1, \ldots, \gamma_k$. The influence function of θ can then be obtained as

$$\text{IF}_i(\hat{\theta}) = \frac{1}{G}\left(h_i^{\hat{\theta}} - \sum_{j=1}^{k} G_j \text{IF}_i(\hat{\gamma}_j)\right) \quad \text{with} \quad G_j = -\frac{1}{W}\sum_{i=1}^{n} w_i \left.\frac{\partial h_i^{\hat{\theta}}}{\partial \gamma_j}\right|_{\gamma_j=\hat{\gamma}_j}$$

- If γ_j itself depends on further parameters, its influence function will have a similar form. In this way we can easily piece together influence functions also for very complex statistics.

Moment conditions and influence functions

- More generally (see Jann 2020), if θ is a vector of estimates $\theta_1, \ldots, \theta_k$ that may or may not depend on each other, the (k dimensional) influence function of θ can be obtained as

$$\mathrm{IF}_i(\hat{\theta}) = \mathbf{G}^{-1}\mathbf{h}_i^{\hat{\theta}}$$

  where

$$\mathbf{h}_i^{\theta} = \begin{bmatrix} h_i^{\theta_1} \\ \vdots \\ h_i^{\theta_k} \end{bmatrix} \quad \text{and} \quad \mathbf{G} = -\frac{1}{W}\sum_{i=1}^{n} w_i \left.\frac{\partial \mathbf{h}_i^{\theta}}{\partial \theta'}\right|_{\theta=\hat{\theta}}$$

- As David Drukker would say:

"Stack the moment equations!"

Moment conditions and influence functions

- Although things might look complicated at first sight, obtaining influence functions is actually quite easy in most cases. Also subpopulation estimation can be integrated without much trouble and there is a simple solution to take account of covariate balancing based on reweighting.
- Hence, influence functions are the method of choice for the new command.
- One great thing about influence functions is that support for complex survey estimation (`svy`) comes for free.
- Another great thing is that the influence function of a linear or nonlinear combination of several statistics can be obtained by linear combination of the individual influence functions (as implied by the chain rule).
- Furthermore, the RIFs (recentered influence functions) can be used in regression models to study approximate effects of covariates on a statistic (Firpo et al. 2009).

Estimation

- Distribution functions:

  `dstat` *subcmd varlist* [*if*] [*in*] [*weight*] [, *options*]

  where *subcmd* is one of `density` (or `pdf`), `histogram`, `proportion`, `frequency`, `cdf`, `ccdf`, `quantile`, `lorenz`, `share`, `tip`.

- Summary statistics:

  `dstat` [(*stats*)] *varlist* [(*stats*) *varlist* ...] [*if*] [*in*] [*weight*] [, *options*]

  where *stats* is a space-separated list of statistics. A large collection of statistics is available ...

Points in the distribution

quantile(*p***)**	*p*/100 quantile; *p* in [0,100]
p(*p***)**	alias for **quantile()**
hdquantile(*p***)**	*p*/100 Harrell/Davis (1982) quantile; *p* in [0,100]
density(*x***)**	kernel density at value *x*
hist(*x1***,***x2***)**	histogram density of data within (*x1*,*x2*]
cdf[*](*x***)**	cumulative distribution (CDF) at value *x*; suffix * is empty for default, **m** for mid-adjusted CDF, **f** for floor CDF
ccdf[*](*x***)**	complementary CDF at value *x*; suffix * is empty for default, **m** for mid-adjusted CCDF, **f** for floor CCDF
prop(*x1***[,***x2***])**	proportion of data equal to *x1* or within [*x1*,*x2*]
pct(*x1***[,***x2***])**	percent of data equal to *x1* or within [*x1*,*x2*]
freq(*x1***[,***x2***])**	frequency of data equal to *x1* or within [*x1*,*x2*]
total[(*x1***[,***x2***])]**	overall total, or total of data equal to *x1* or within [*x1*,*x2*]
min	observed minimum (standard error set to zero)
max	observed maximum (standard error set to zero)
range	**max**−**min** (standard error set to zero)
midrange	(**min**+**max**)/2 (standard error set to zero)

Location measures

mean	arithmetic mean
gmean	geometric mean (data must be positive)
hmean	harmonic mean (data must be positive)
trim(*p1***[,***p2***])**	trimmed mean with *p1*/100 lower trimming and *p2*/100 upper trimming; *p1* and *p2* in [0,50]; *p2*=*p1* if omitted; default is *p1*=*p2*=25
winsor(*p1***[,***p2***])**	winsorized mean with *p1*/100 lower winsorizing and *p2*/100 upper winsorizing; *p1* and *p2* in [0,50]; *p2*=*p1* if omitted; default is *p1*=*p2*=25
median	median; equal to **q50**
huber[(*p***)]**	Huber M estimate with gaussian efficiency *p* in [63.7,99.9]; default is *p*=95
biweight[(*p***)]**	biweight M estimate with gaussian efficiency *p* in [.01,99.9]; default is *p*=95
hl	Hodges–Lehmann location measure (Hodges and Lehmann 1963)

Scale measures

sd[(*df*)]	standard deviation; *df* applies small-sample adjustment; default is *df*=1
variance[(*df*)]	variance; default is *df*=1
mse[(*x*[,*df*])]	mean squared deviation from value *x* (mean squared error); default is *x*=0 and *df*=0
smse[(*x*[,*df*])]	square-root of mean squared deviation from value *x*; default is *x*=0 and *df*=0
iqr[n][(*p1*,*p2*)]	interquantile range; default is **iqr(25,75)** (interquartile range); specify **iqrn** for normalized IQR, equal to 1/(invnormal(*p2*) - invnormal(*p1*)) * **iqr**
mad[n][(*l*[,*t*])]	median (or mean if *l*!=0) absolute deviation from the median (or mean if *t*!=0); specify **madn** for normalized MAD, equal to 1/invnormal(0.75) * **mad** (or sqrt(pi/2) * **mad** if *l*!=0)
mae[n][(*l*[,*x*])]	median (or mean if *l*!=0) absolute deviation from value *x*; default is *x*=0; specify **maen** for normalized MAE, equal to 1/invnormal(0.75) * **mae** or (sqrt(pi/2) * **mae** if *l*!=0)
md[n]	mean absolute pairwise difference; equal to 2 * **mean** * **gini**; specify **mdn** for normalized MD, equal to sqrt(pi)/2 * **md**
mscale[(*bp*)]	M estimate of scale with breakdown point *bp* in [1,50]; default is *bp*=50
qn	Qn scale coefficient (Rousseeuw and Croux 1993)

Skewness measures

skewness	skewness
qskew[(*alpha*)]	quantile skewness (Hinkley 1975); *alpha* in [0,50]; default is *alpha*=25
mc	medcouple (Brys et al. 2004)

Kurtosis measures

kurtosis	kurtosis
qw[(*alpha*)]	quantile tail weight; *alpha* in [0,50]; default is *alpha*=25
lqw[(*alpha*)]	left quantile tail weight; *alpha* in [0,50]; default is *alpha*=25
rqw[(*alpha*)]	right quantile tail weight; *alpha* in [0,50]; default is *alpha*=25
lmc	left medcouple tail weight measure (Brys et al. 2006)
rmc	right medcouple tail weight measure (Brys et al. 2006)

Inequality measures

hoover	Hoover index (Robin Hood index)
gini[(*df*)]	Gini coefficient; *df* applies small-sample adjustment; default is *df*=0
agini[(*df*)]	absolute Gini coefficient
mld	mean log deviation; equal to **ge(0)**
theil	Theil index; equal to **ge(1)**
cv[(*df*)]	coefficient of variation; default is *df*=1; **cv(0)**=sqrt(2***ge(1)**)
ge[(*alpha*)]	generalized entropy (Shorrocks 1980) with parameter *alpha*
atkinson[(*epsilon*)]	Atkinson index with parameter *epsilon*>=0; default is *epsilon*=1
lvar[(*df*)]	logarithmic variance; default is *df*=1
vlog[(*df*)]	variance of logarithm; default is *df*=1
top[(*p*)]	outcome share of top *p* percent; default is *p*=10
bottom[(*p*)]	outcome share of bottom *p* percent; default is *p*=40
mid[(*p1*,*p2*)]	outcome share of mid *p1* to *p2* percent; default is *p1*=40 and *p2*=90
palma	palma ratio; equal to **top/bottom** or **sratio(40,90)**
qratio[(*p1*,*p2*)]	quantile ratio **q**(*p2*)/**q**(*p1*); default is *p1*=10 and *p2*=90
sratio[(*l1*,*u1*,*l2*,*u2*)]	percentile share ratio; default is *l1*=0, *u1*=10, *l2*=90, *u2*=100; can also specify **sratio**(*u1*,*l2*)
[*]**lorenz**(*p*)	Lorenz ordinate, *p* in [0,100]; prefix * is empty for default, **g** for generalized, **t** for total, **a** for absolute, **e** for equality gap
[*]**share**(*p1*,*p2*)	percentile share, *p1* and *p2* in [0,100]; prefix * is empty for default, **d** for density, **g** for generalized, **t** for total, **a** for average

Concentration measures

gci[(*zvar*[,*df*])]	Gini concentration index; *zvar* specifies the sort variable; default is as set by option **zvar()**; *df* applies small-sample adjustment; default is *df*=0; can also specify **gci**(*df*)
aci[(*zvar*[,*df*])]	absolute Gini concentration index; syntax as for **gci**
[*]**ccurve**(*p*[,*zvar*])	concentration curve ordinate, *p* in [0,100]; prefix * is empty for default, **g** for generalized, **t** for total, **a** for absolute, **e** for equality gap
[*]**cshare**(*p1*,*p2*[,*zvar*])	concentration share, *p1* and *p2* in [0,100]; prefix * is empty for default, **d** for density, **g** for generalized, **t** for total, **a** for average

Poverty measures

hcr[(*pline*)]	head count ratio (i.e. proportion poor); *pline* specifies the poverty line > 0; *pline* can be *varname* or #; default is as set by option **pline()**
[**a**]**pgap**[(*pline*)]	poverty gap (proportion by which mean outcome of poor is below *pline*); specify **apgap** for absolute poverty gap (*pline* - mean outcome of poor)
[**a**]**pgi**[(*pline*)]	poverty gap index; equal to **hcr*pgap**; specify **apgi** for absolute poverty gap index, equal to **hcr*apgap**
fgt[(*a*[,*pline*])]	Foster–Greer–Thorbecke index with *a*>=0 (Foster et al. 1984, 2010); default is *a*=0 (head count ratio); *a*=1 is equivalent to **pgi**
sen[(*pline*)]	Sen poverty index (Sen 1976; using the replication invariant version of the index, also see Shorrocks 1995)
sst[(*pline*)]	Sen–Shorrocks–Thon poverty index (see, e.g., Osberg and Xu 2008)
takayama[(*pline*)]	Takayama poverty index (Takayama 1979)
watts[(*pline*)]	Watts index (see, e.g., Saisana 2014)
chu[(*a*[,*pline*])]	Clark–Hemming–Ulph poverty index with *a* in [0,100] (Clark et al. 1981); default is *a*=50; *a*=0 is equivalent to 1-exp(-**watts**); *a*=100 is equivalent to **fgt(1)**
[**a**]**tip**(*p*[,*pline*])	TIP ordinate, *p* in [0,100]; specify **atip()** for absolute TIP ordinates

Association

corr[(*zvar*)]	correlation coefficient; *zvar* specifies the secondary variable; default is as set by option **zvar()**
cov[(*zvar*[,*df*])]	covariance; *df* applies small-sample adjustment; default is *df*=1; can also specify **cov**(*df*)
spearman[(*zvar*)]	Spearman's rank correlation

Some main options

- `over(`*overvar* [`,` *options*]`)` [`total`]
  - compute results by subpopulations, possibly including total, possibly accumulating or taking contrasts
- `balance(`[*method* `:`] *varlist* [`,` *options*]`)`
  - balance covariates using entropy balancing (see Jann 2021) or IPW
- `nocasewise`
  - exclude missing values for each variable individually (no casewise deletion of observations)
- `vce(`*vcetype* [`,` *options*]`)`
  - note: specify `vce(svy)` for complex survey estimation instead of applying the `svy` prefix!
- There are many more options; see `help dstat` for details.

After estimation

- Draw graph:

  `dstat graph` [, *graph_options*]

  or apply option `graph()` when estimating. The graphs will be created by an internal call to `coefplot` (Jann 2014).

- Obtain influence functions (optionally recentered):

  `predict` { *stub*`*` | *newvarlist* } [*if*] [*in*] [, *predict_options*]

  or apply option `generate()` when estimating.

```
. use sess16, clear
(Sample from Swiss Earnings Structure Survey 2016)
. describe
Contains data from sess16.dta
 Observations:        50,000                  Sample from Swiss Earnings Structure
                                                Survey 2016
    Variables:             5                  24 Jun 2021 21:38
```

Variable name	Storage type	Display format	Value label	Variable label
earnings	long	%10.0g		monthly earnings in CHF (full-time equivalent)
gender	byte	%8.0g	gender	gender
educ	byte	%27.0g	educ	highest educational degree
tenure	byte	%8.0g		tenure (in years)
wgt	double	%10.0g		sampling weight

```
Sorted by:
. summarize
```

Variable	Obs	Mean	Std. dev.	Min	Max
earnings	47,600	7848.055	4189.382	2314	103998
gender	49,771	.5547608	.4969972	0	1
educ	49,503	2.797063	1.304769	1	5
tenure	48,525	8.599588	8.934825	0	61
wgt	50,000	33.19645	61.75064	8.435029	2991.433

- If you specify no subcommand and no statistics, `dstat` behaves like official `mean`:

```
. dstat earnings [pw=wgt], over(gender)
mean                                  Number of obs    =     47,383
```

earnings	Coefficient	Std. err.	[95% conf.	interval]
gender				
female	6511.388	31.48064	6449.686	6573.091
male	8007.006	53.82319	7901.511	8112.5

```
. mean earnings [pw=wgt], over(gender)
Mean estimation                       Number of obs = 47,383
```

	Mean	Std. err.	[95% conf.	interval]
c.earnings@gender				
female	6511.388	31.48064	6449.686	6573.091
male	8007.006	53.82319	7901.511	8112.5

- Other than `mean` it allows you to include the total across subpopulations or to take contrasts:

```
. dstat earnings [pw=wgt], over(gender, contrast(1) ratio)
Ratio of mean                          Number of obs   =     47,383
                                       Contrast        =   1.gender
```

earnings	Coefficient	Std. err.	[95% conf.	interval]
gender				
female	.8132114	.0067335	.8000137	.8264091

```
. dstat earnings [pw=wgt], over(gender) total
mean                                   Number of obs   =     47,383
```

earnings	Coefficient	Std. err.	[95% conf.	interval]
gender				
female	6511.388	31.48064	6449.686	6573.091
male	8007.006	53.82319	7901.511	8112.5
total	7366.241	34.06905	7299.465	7433.017

- You can also select and reorder subpopulations (total will still be across all subpopulations):

```
. dstat earnings [pw=wgt], over(educ) total
mean                                Number of obs    =     47,129
```

earnings	Coefficient	Std. err.	[95% conf.	interval]
educ				
Lower secondary	5093.716	38.91248	5017.447	5169.985
Upper secondary: vocational	6380.539	34.30033	6313.31	6447.768
Upper secondary: general	7438.123	141.6259	7160.534	7715.711
Tertiary: vocational	9019.856	56.53847	8909.039	9130.672
Tertiary: academic	11471.11	170.9918	11135.96	11806.25
total	7369.873	34.33644	7302.573	7437.173

```
. dstat earnings [pw=wgt], over(educ, select(5 4)) total
mean                                Number of obs    =     47,129
```

earnings	Coefficient	Std. err.	[95% conf.	interval]
educ				
Tertiary: academic	11471.11	170.9918	11135.96	11806.25
Tertiary: vocational	9019.856	56.53847	8909.039	9130.672
total	7369.873	34.33644	7302.573	7437.173

- Furthermore, `dstat` supports IPW covariate balancing. Here is an "ATT" of being male (earnings gap if women's sample is reweighted):

```
. dstat earnings [pw=wgt], over(gender, contrast) balance(i.educ tenure, ref(1))
Difference in mean                              Number of obs     =     45,530
                                                Contrast          =   0.gender
                                                Balancing:
                                                        method    =        ipw
                                                     reference    =   1.gender
                                                      controls    = e(balance)
```

earnings	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
gender						
male	1261.559	55.857	22.59	0.000	1152.078	1371.039

```
. teffects ipw (earnings) (gender i.educ tenure) [pw=wgt], atet nolog
Treatment-effects estimation                    Number of obs     =     45,530
Estimator      : inverse-probability weights
Outcome model  : weighted mean
Treatment model: logit
```

	earnings	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
ATET							
	gender						
	(male vs female)	1261.559	55.85639	22.59	0.000	1152.082	1371.035
POmean							
	gender						
	female	6755.25	34.66481	194.87	0.000	6687.308	6823.192

- IPW does not perfectly balance the data . . .

```
. dstat (pr1 pr2 pr3 pr4 pr5) educ (mean) tenure [pw=wgt] if earnings<., ///
>     over(gender, contrast) balance(i.educ tenure, ref(1))
Difference in summary statistics                Number of obs     =     45,530
                                                Contrast          =   0.gender
                                                Balancing:
                                                          method  =        ipw
                                                       reference  =   1.gender
                                                        controls  = e(balance)

            0: gender = female
            1: gender = male
```

	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
1~educ						
pr1	-.0017249	.0005537	-3.12	0.002	-.0028101	-.0006396
pr2	.0006886	.0009046	0.76	0.447	-.0010844	.0024616
pr3	-.0002834	.0003077	-0.92	0.357	-.0008864	.0003197
pr4	.0003433	.000665	0.52	0.606	-.0009601	.0016466
pr5	.0009764	.0004809	2.03	0.042	.0000338	.0019189
1~tenure						
mean	.1103688	.0254176	4.34	0.000	.0605498	.1601877

- . . . so we may prefer entropy balancing (see Jann 2021):

```
. dstat earnings [pw=wgt], over(gender, contrast) balance(eb:i.educ tenure, ref(1))
Difference in mean                                  Number of obs     =     45,530
                                                    Contrast          =   0.gender
                                                    Balancing:
                                                               method =         eb
                                                            reference =   1.gender
                                                             controls = e(balance)
```

earnings	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
gender						
male	1247.068	55.99405	22.27	0.000	1137.318	1356.817

```
. kmatch eb gender i.educ tenure (earnings = i.educ tenure) [pw=wgt], att nomtable
(fitting balancing weights ... done)
Entropy balancing                                   Number of obs = 45,530
                                                    Balance tolerance =     .00001

Treatment    : gender = 1
Targets      : 1
Covariates   : i.educ tenure
RA equations: earnings = i.educ tenure _cons

Treatment-effects estimation
```

earnings	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
ATT	1247.068	55.99405	22.27	0.000	1137.318	1356.817

- Perfect balance!

```
. dstat (pr1 pr2 pr3 pr4 pr5) educ (mean) tenure [pw=wgt] if earnings<., ///
>     over(gender, contrast) balance(eb:i.educ tenure, ref(1))
Difference in summary statistics                Number of obs     =     45,530
                                                Contrast          =   0.gender
                                                Balancing:
                                                          method  =         eb
                                                       reference  =   1.gender
                                                        controls  = e(balance)

           0: gender = female
           1: gender = male
```

	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
1~educ						
pr1	-3.61e-16	2.12e-17	-17.05	0.000	-4.02e-16	-3.19e-16
pr2	2.22e-16	1.48e-17	14.95	0.000	1.93e-16	2.51e-16
pr3	2.08e-17	1.41e-18	14.78	0.000	1.81e-17	2.36e-17
pr4	1.39e-16	6.77e-18	20.50	0.000	1.26e-16	1.52e-16
pr5	6.94e-17	3.99e-18	17.39	0.000	6.16e-17	7.72e-17
1~tenure						
mean	-3.55e-15	2.22e-16	-15.99	0.000	-3.99e-15	-3.12e-15

- **All of the above you can do with any other statistic, also with multiple statistics and multiple variables at the same time!**

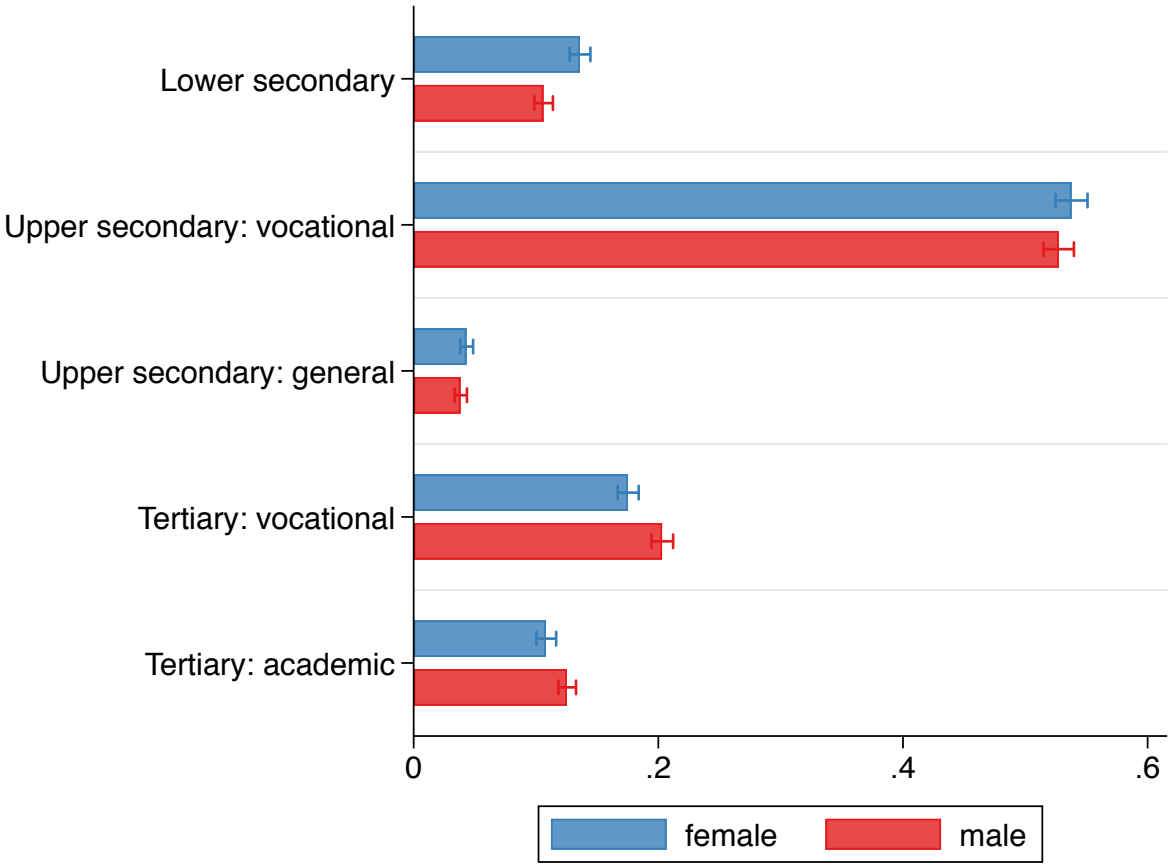
```
. generate lnearn = ln(earnings)
(2,400 missing values generated)
. dstat (mean gini mld vlog) earnings (mean var) lnearn [pw=wgt], ///
>     over(gender, contrast) balance(eb:i.educ tenure, ref(1))
Difference in summary statistics                Number of obs     =     45,530
                                                Contrast          =   0.gender
                                                Balancing:
                                                          method  =         eb
                                                       reference  =   1.gender
                                                        controls  = e(balance)

            0: gender = female
            1: gender = male
```

	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
1~earnings						
mean	1247.068	55.99405	22.27	0.000	1137.318	1356.817
gini	.0393687	.0038698	10.17	0.000	.0317838	.0469536
mld	.0298174	.0031885	9.35	0.000	.0235679	.036067
vlog	.0421788	.0043786	9.63	0.000	.0335967	.0507609
1~lnearn						
mean	.1392601	.0056122	24.81	0.000	.1282601	.1502602
var	.0421788	.0043786	9.63	0.000	.0335967	.0507609

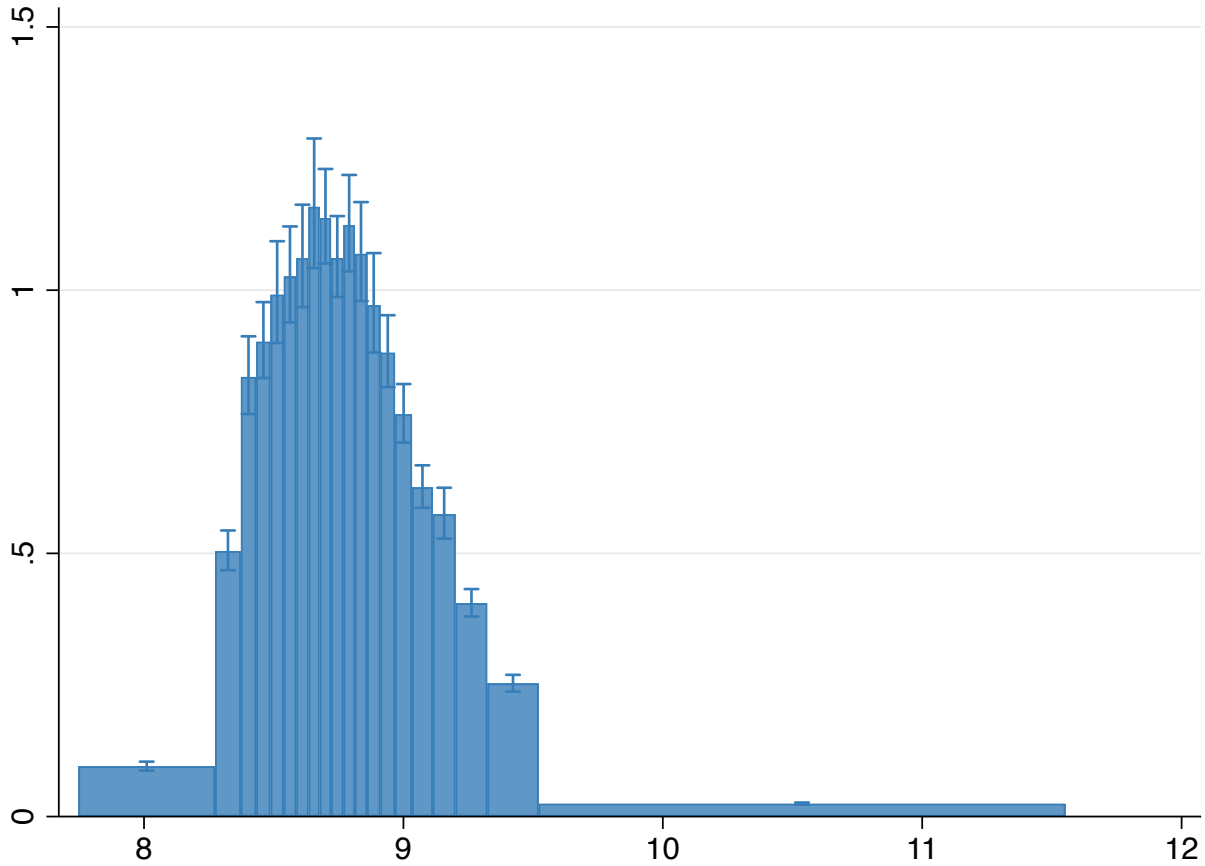
- Some other stuff. Here is the educational distribution by gender:

```
. dstat prop educ [pw=wgt], over(gender) graph(merge)
Proportion                          Number of obs    =     49,277
            0: gender = female
            1: gender = male
(coefficients table suppressed)
```



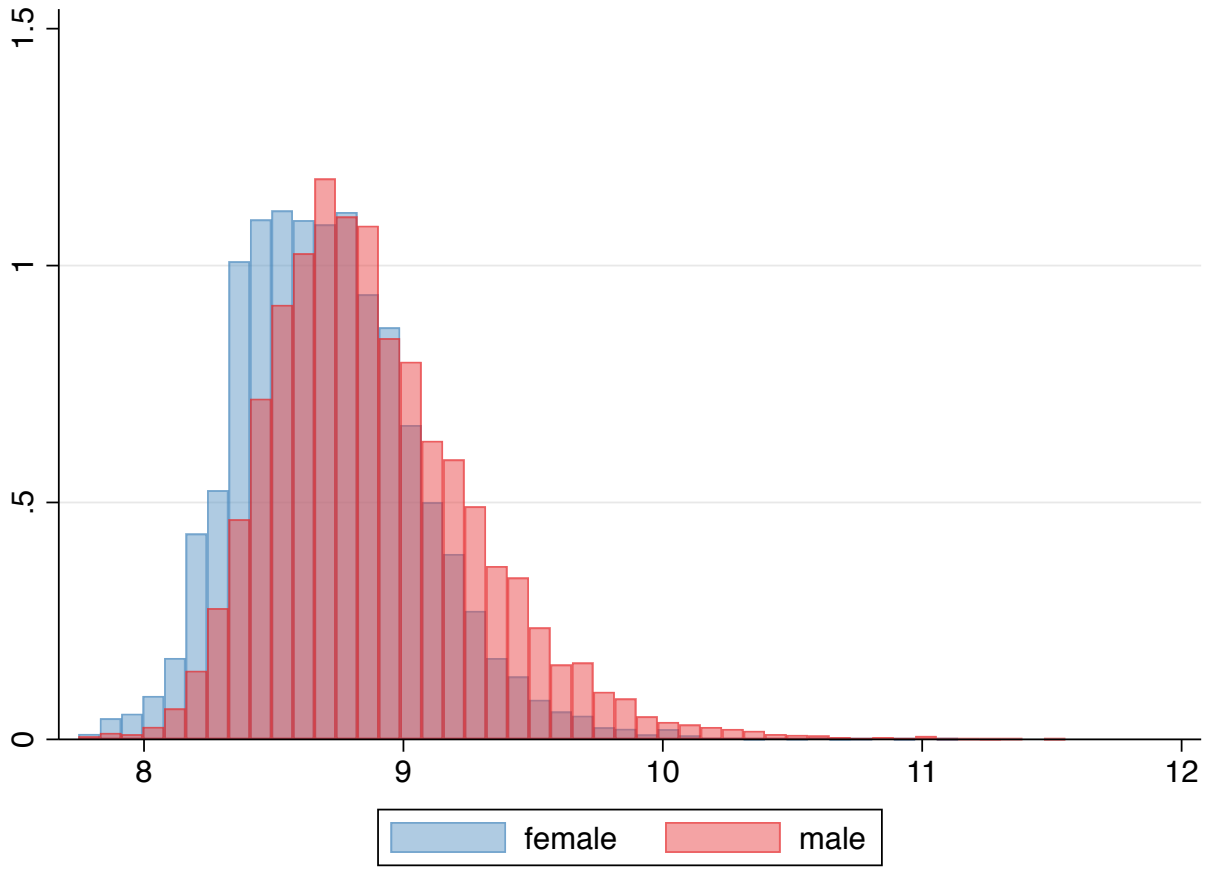
- Or an equal probability histogram of log earnings:

```
. dstat histogram lnearn [pw=wgt], ep n(20) graph
Histogram (density)                  Number of obs    =     47,600
(coefficients table suppressed)
```



- Or histograms of log earnings by gender printed on top of each other using the same bin definitions:

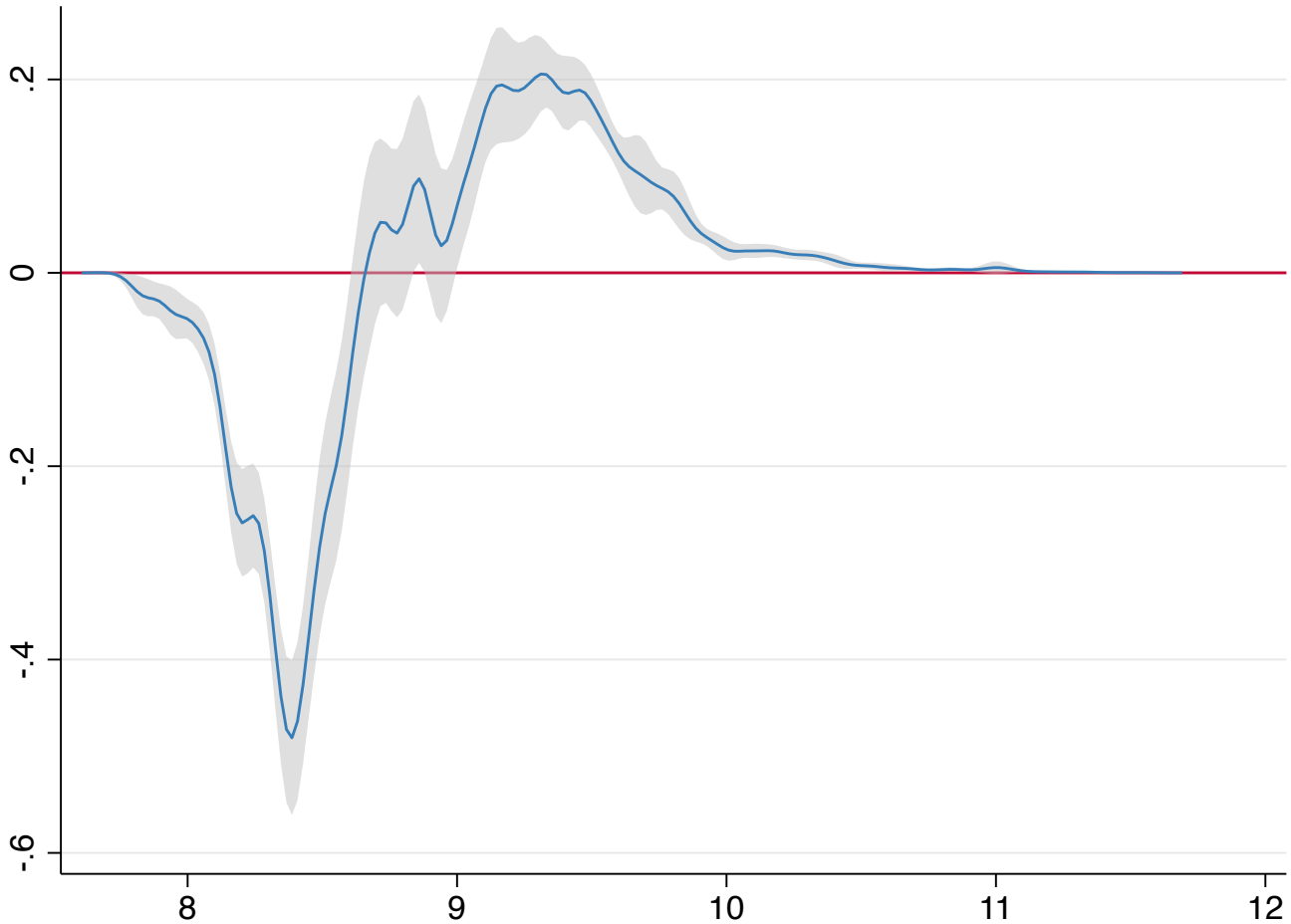
```
. dstat histogram lnearn [pw=wgt], over(gender) common nose graph(merge)
Histogram (density)                  Number of obs    =     47,383
            0: gender = female
            1: gender = male
(coefficients table suppressed)
```



- Or the difference in the density function of log earnings by gender:

```
. dstat density lnearn [pw=wgt], over(gender, contrast) n(200) ///
>     graph(yline(0))
```

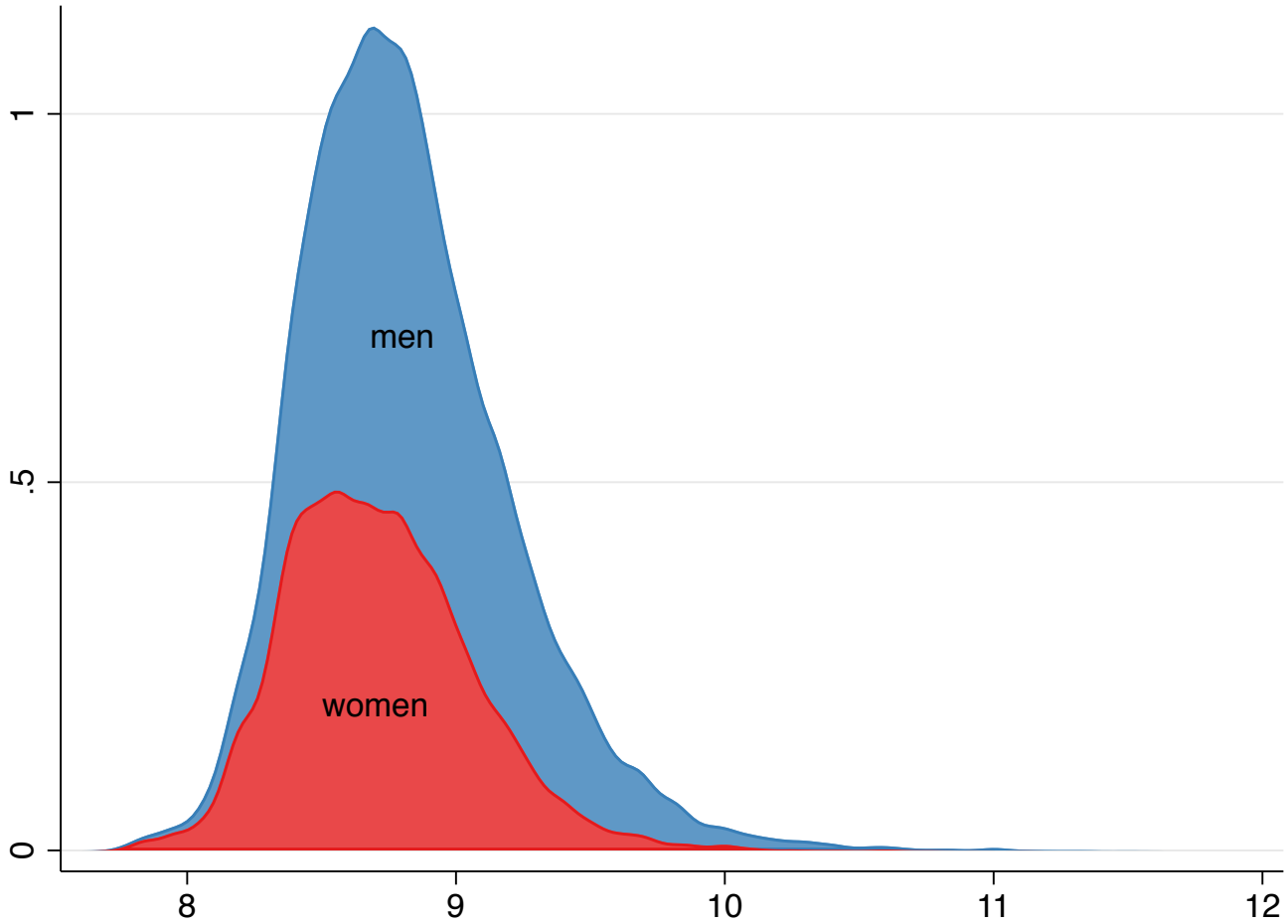
(*output omitted*)



- Or the composition of the overall density of log earnings by gender:

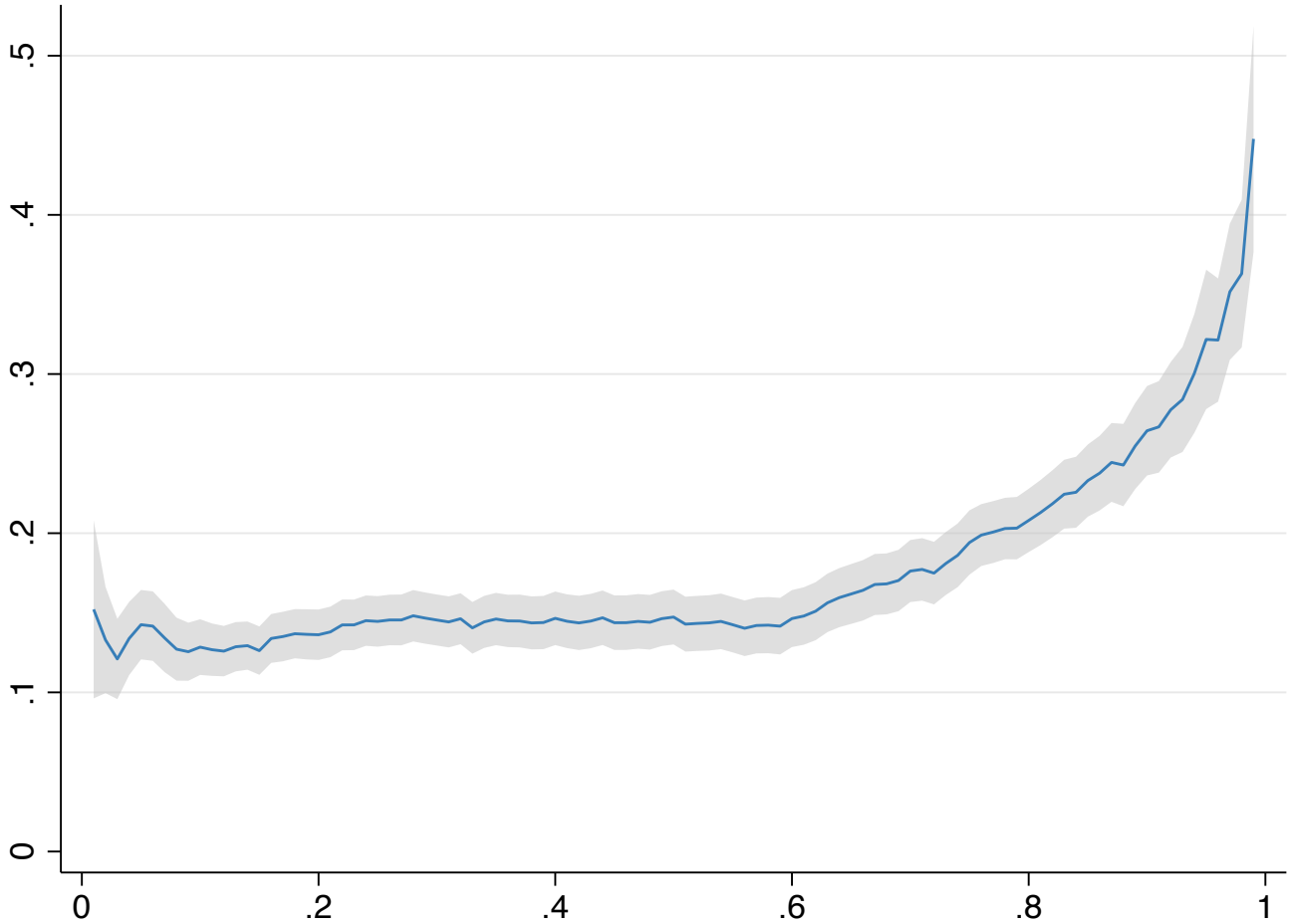
```
. dstat density lnearn [pw=wgt], over(gender) total nose n(200) unconditional ///
>     graph(recast(area) select(3 1) merge legend(off) ///
>         text(.2 8.7 "women" .7 8.8 "men"))
```

(*output omitted*)



- Or the difference in quantile functions of log earnings by gender:

```
. dstat quantile lnearn [pw=wgt], over(gender, contrast) graph(ylabel(0(.1).5))
```
(*output omitted*)



- `dstat` cannot compute balanced and unbalanced results at the same time. If you want include both sets of results in the same graph, you need to store the estimates and use `coefplot` manually.
- Here is how we can create a graph that shows the quantile wage gap function with and without covariate adjustment:

```
. dstat quantile lnearn [pw=wgt], over(gender, contrast) notable ///
>     balance(eb:i.educ tenure)
  (output omitted)
. estimates store balanced
. dstat quantile lnearn [pw=wgt] if e(sample), over(gender, contrast)
  (output omitted)
. estimates store raw
. coefplot raw balanced, se(se) at(at) keep(1:) ylabel(0(.1).5) ///
>     recast(line) ciopts(recast(rarea) pstyle(ci) color(%50) lcolor(%0)) ///
>     plotlabels("raw wage gap" "adjusted wage gap")
```


0
.1
.2
.3
.4
.5
0
.2
.4
.6
.8
1
raw wage gap
adjusted wage gap

Conclusions

- I could go on forever. There would be so much more to say . . .

  . . . for example, on the cool stuff you can do with the influence functions (or RIFs) generated by `dstat`.

- Have fun with the command!

- Drop me a note if you want me to add a specific statistic.

- Install from SSC

```
. ssc install dstat, replace
. ssc install moremata, replace
. ssc install coefplot, replace
```

  or from GitHub: http://github.com/benjann/dstat

References

- Deville, J.C. 1999. Variance estimation for complex statistics and estimators: linearization and residual techniques. Survey Methodology 25:193–203.
- Firpo, S., N.M. Fortin, T. Lemieux. 2009. Unconditional quantile regressions. Econometrica 77:953–973.
- Hampel, F.R. 1974. The influence curve and its role in robust estimation. Journal of the American Statistical Association 69:383–393.
- Jann, B. 2014. Plotting regression coefficients and other estimates. The Stata Journal 14:708-737.
- Jann, B. 2020. Influence functions continued. A framework for estimating standard errors in reweighting, matching, and regression adjustment. University of Bern Social Sciences Working Papers 35. Available from https://ideas.repec.org/p/bss/wpaper/35.html.
- Jann, B. 2021. Entropy balancing as an estimation command. University of Bern Social Sciences Working Papers 39. Available from https://ideas.repec.org/p/bss/wpaper/39.html.