

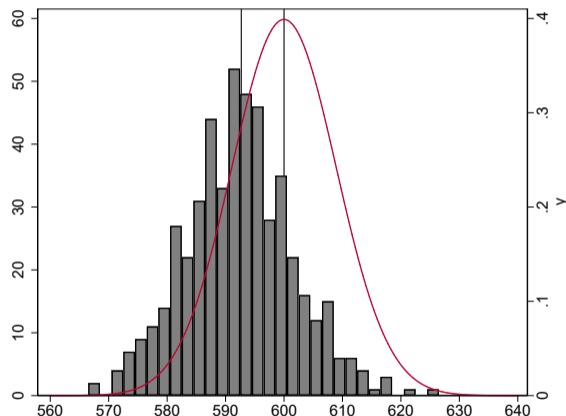
Exchangeably weighted bootstrap schemes

2022 UK Stata user group meeting – UCL, September 8–9, 2022

Philippe Van Kerm

University of Luxembourg and LISER

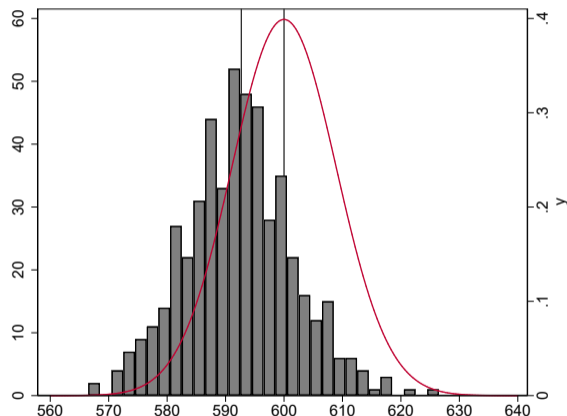
The bootstrap approach to statistical inference



Learn about unknown sampling distribution of a statistic from the distribution of estimates across bootstrap samples generated from the observed data

- How to generate bootstrap samples?
- How to make inference from them (confidence intervals)?

The bootstrap approach to statistical inference



Learn about unknown sampling distribution of a statistic from the distribution of estimates across bootstrap samples generated from the observed data

- How to generate bootstrap samples?
- How to make inference from them (confidence intervals)?

Various strategies for generating bootstrap samples

(Efron and Tibshirani, 1993, Davison and Hinkley, 1997)

- Non-parametric bootstrap
 - » Classic paired bootstraps – `bsample`
 - » Block bootstraps – `bsample`
 - » Balanced bootstraps – `bsweights` (Kolenikov, 2010)
 - » Survey bootstraps – `bsweights`, `rhsbsample` (Van Kerm, 2013);
 - » Exchangeable (weighted) bootstrap – `exbsample`
(Praestgaard and Wellner, 1993) (also see Chernozhukov et al., 2013)
- Residual bootstrap
 - » Wild bootstrap – `boottest` (Roodman et al., 2019)

(Fuzzy classification – incomplete and not mutually exclusive)

Exchangeable (weighted) bootstrap

- Paired bootstrap: obs appear an integer number of times in bootstrap samples
- ⇒ 'frequency weighting' of original sample
- Poisson bootstrap: draw from a $\text{Poisson}(1)$ distribution to set the bootstrap frequency weight
- Why stick to integer weights? Exponential bootstrap: make a draw from an $\text{exponential}(1)$ distribution
 - » each observation has a positive (non-integer) weight
 - » (rescale the weights to average to 1 (sum to n))
 - » Major advantage: no observation is ever 'excluded' from the sample (no issues of 'no observations' in resamples, or perfect collinearity; bootstrap for matching estimators (Otsu and Rai, 2017))

Exchangeable (weighted) bootstrap

- Paired bootstrap: obs appear an integer number of times in bootstrap samples
- ⇒ ‘frequency weighting’ of original sample
- Poisson bootstrap: draw from a $\text{Poisson}(1)$ distribution to set the bootstrap frequency weight
 - Why stick to integer weights? Exponential bootstrap: make a draw from an $\text{exponential}(1)$ distribution
 - » each observation has a positive (non-integer) weight
 - » (rescale the weights to average to 1 (sum to n))
 - » Major advantage: no observation is ever ‘excluded’ from the sample (no issues of ‘no observations’ in resamples, or perfect collinearity; bootstrap for matching estimators (Otsu and Rai, 2017))

Exchangeable (weighted) bootstrap

- Paired bootstrap: obs appear an integer number of times in bootstrap samples
- ⇒ 'frequency weighting' of original sample
- **Poisson** bootstrap: draw from a Poisson(1) distribution to set the bootstrap frequency weight
 - Why stick to integer weights? Exponential bootstrap: make a draw from an exponential(1) distribution
 - » each observation has a positive (non-integer) weight
 - » (rescale the weights to average to 1 (sum to n))
 - » Major advantage: no observation is ever 'excluded' from the sample (no issues of 'no observations' in resamples, or perfect collinearity; bootstrap for matching estimators (Otsu and Rai, 2017))

Exchangeable (weighted) bootstrap

- Paired bootstrap: obs appear an integer number of times in bootstrap samples
- ⇒ 'frequency weighting' of original sample
- **Poisson** bootstrap: draw from a Poisson(1) distribution to set the bootstrap frequency weight
 - Why stick to integer weights? **Exponential** bootstrap: make a draw from an exponential(1) distribution
 - » each observation has a positive (non-integer) weight
 - » (rescale the weights to average to 1 (sum to n))
 - » Major advantage: no observation is ever 'excluded' from the sample (no issues of 'no observations' in resamples, or perfect collinearity; bootstrap for matching estimators (Otsu and Rai, 2017))

Exchangeable (weighted) bootstrap

- Paired bootstrap: obs appear an integer number of times in bootstrap samples
- ⇒ ‘frequency weighting’ of original sample
- **Poisson** bootstrap: draw from a Poisson(1) distribution to set the bootstrap frequency weight
- Why stick to integer weights? **Exponential** bootstrap: make a draw from an exponential(1) distribution
 - » each observation has a positive (non-integer) weight
 - » (rescale the weights to average to 1 (sum to n))
 - » Major advantage: no observation is ever ‘excluded’ from the sample (no issues of ‘no observations’ in resamples, or perfect collinearity; bootstrap for matching estimators (Otsu and Rai, 2017))

A little helper: The `exbsample` command

The command `exbsample` (available on SSC shortly) generates bootstrap replication weights using Poisson or Exponential draws

Syntax

```
exbsample [#] [if] [in] [weight] [using filename]  
[ , stub(newvarnameprefix) distribution(poisson|exponential) norescale  
balance(#) strata(varlist) cluster(varlist) frame(name) ... ]
```

(A simple command really, but which takes care of nitty-gritty details.)

A little helper: The `exbsample` command

The command `exbsample` (available on SSC shortly) generates bootstrap replication weights using Poisson or Exponential draws

Syntax

```
exbsample [#] [if] [in] [weight] [using filename]  
[ , stub(newvarnameprefix) distribution(poisson|exponential) norescale  
balance(#) strata(varlist) cluster(varlist) frame(name) ... ]
```

(A simple command really, but which takes care of nitty-gritty details.)

A little helper: The `exbsample` command

The command `exbsample` (available on SSC shortly) generates bootstrap replication weights using Poisson or Exponential draws

Syntax

```
exbsample [#] [if] [in] [weight] [using filename]  
[ , stub(newvarnameprefix) distribution(poisson|exponential) norescale  
balance(#) strata(varlist) cluster(varlist) frame(name) ... ]
```

(A simple command really, but which takes care of nitty-gritty details.)

A little helper: The `exbsample` command

The command `exbsample` (available on SSC shortly) generates bootstrap replication weights using Poisson or Exponential draws

Syntax

```
exbsample [#] [if] [in] [weight] [using filename]  
[ , stub(newvarnameprefix) distribution(poisson|exponential) norescale  
balance(#) strata(varlist) cluster(varlist) frame(name) ... ]
```

(A simple command really, but which takes care of nitty-gritty details.)

Using replication weights

- The flexible (but hardest) way: repeat analysis with alternative weight variables
 - » e.g., passing weights as argument to do files (and looping):
`do mydofile.do rweightvar'i'`
 - » post results in files ('resultssets')
 - .. and combine resulting estimates 'manually' (allows flexibility in how CIs are constructed)
- Use the `svy bootstrap` prefix (instead of standard `bootstrap`: prefix)
- Use Jeff Pitblado's `bs4rw` prefix (a predecessor of `svy bootstrap`:)

A simple example

Generate the bootstrap weights

```
. sysuse auto , clear
(1978 Automobile Data)
. exbsample 499 , stub(rw)           // vars rw1 - rw499 created
.....
> .....
> .....
> .....
> .....
> .....
> .....
> .....
. summarize rw1 rw2 rw3 rw499
```

Variable	Obs	Mean	Std. Dev.	Min	Max
rw1	74	1	1.014414	.0495382	4.726035
rw2	74	1	1.152799	.0043677	8.042064
rw3	74	1	.9435121	.0204333	3.754344
rw499	74	1	1.12571	.0051524	5.829083

Option 1: J Pitblado's bs4rw prefix command

bs4rw (Bootstrapped command needs to accept iweight-s)

```
. qui net install bs4rw , from(http://www.stata.com/users/jpitblado/)
. bs4rw , rweight(rw1-rw499) nodots : mean price
Mean estimation      Number of obs   =      74
                   Replications   =     499
```

	Observed Mean	Bootstrap Std. Err.	Normal-based [95% Conf. Interval]	
price	6165.257	328.3822	5521.639	6808.874

```
. bs4rw mn=r(mean) , rweight(rw1-rw499) nodots : summarize price
BS4Rweights results      Number of obs   =      74
                        Replications   =     499

command: summarize price
mn: r(mean)
```

	Observed Coef.	Bootstrap Std. Err.	z	P> z	Normal-based [95% Conf. Interval]	
mn	6165.257	328.3822	18.77	0.000	5521.639	6808.874

Option 2: svy bootstrap prefix

svy bootstrap (Bootstrapped command needs to accept iweight-s too)

```
. svyset , bsrweight(rw*) vce(bootstrap)
  (output omitted)
. svy bootstrap , nodots : mean price
Survey: Mean estimation      Number of obs   =       74
                             Population size    =       74
                             Replications       =      499
```

	Observed Mean	Bootstrap Std. Err.	Normal-based [95% Conf. Interval]	
price	6165.257	328.053	5522.285	6808.229

```
. di e1(r(table),2,1)*sqrt(499/498)
328.38223
```

Option 2: svy bootstrap **prefix**

svy bootstrap (**force** non-estimation commands)

```
. svy bootstrap mn=r(mean), nodots force : summarize price
```

```
Bootstrap results                Number of obs   =       74  
                                Population size =       74  
                                Replications    =     499
```

```
command: summarize price  
mn:     r(mean)
```

	Observed Coef.	Bootstrap Std. Err.	z	P> z	Normal-based [95% Conf. Interval]	
mn	6165.257	328.053	18.79	0.000	5522.285	6808.229

Generate weighted replication weights

```
. exbsample 499 [iw=weight] , stub(rw) replace // vars rw1 - rw4999 created
```

```
.....  
> .....  
> .....  
> .....  
> .....  
> .....  
> .....
```

```
. bs4rw , rweight(rw1-rw499) nodots : mean price [iw=weight]
```

```
Mean estimation          Number of obs   =          74  
                        Replications     =          499
```

	Observed Mean	Bootstrap Std. Err.	Normal-based [95% Conf. Interval]	
price	6568.637	382.1837	5819.571	7317.703

Bootstrapped commands must accept *both* iw and pw with svy bootstrap

```
. svyset [pw=weight] , bsrweight(rw*) vce(bootstrap)  
  (output omitted)
```

```
. svy bootstrap , nodots : mean price
```

```
Survey: Mean estimation      Number of obs   =      74  
                             Population size    =   223,440  
                             Replications       =     499
```

	Observed Mean	Bootstrap Std. Err.	Normal-based [95% Conf. Interval]	
price	6568.637	381.8005	5820.322	7316.952

Bootstrapped commands must accept *both* iw and pw with svy bootstrap

```
. // convert pw into iw
. pr def mysu , properties(svyb)
1.     if (ustrregexm("`0'", "\[([\s*pwe?i?g?h?t?\s*=\.)*\s*\]")=1) {
2.         loc 0 = substr("`0'", "={ustrregexs(1)'}", "iw=", 1)
3.     }
4.     su '0'
5. end

. svy bootstrap mu=r(mean) , nodots : mysu price
```

Bootstrap results

Number of obs	=	74
Population size	=	223,440
Replications	=	499

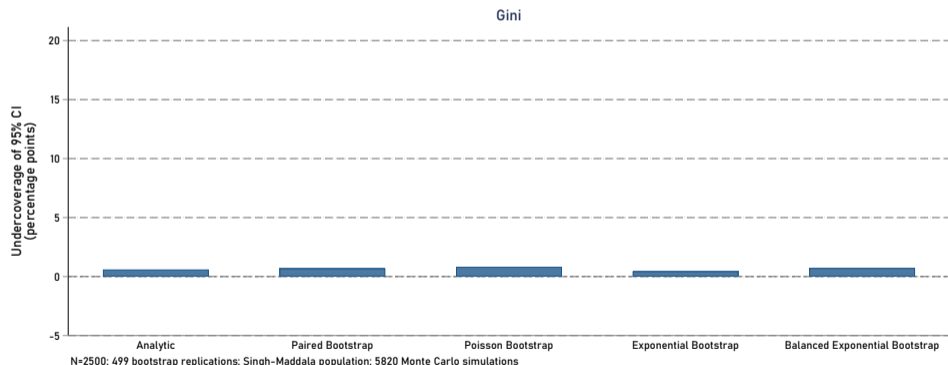
command: mysu price
mu: r(mean)

	Observed Coef.	Bootstrap Std. Err.	z	P> z	Normal-based [95% Conf. Interval]	
mu	6568.637	381.8005	17.20	0.000	5820.322	7316.952

Does it really 'work'?

Statistical properties of exchangeable bootstraps similar to paired bootstrap

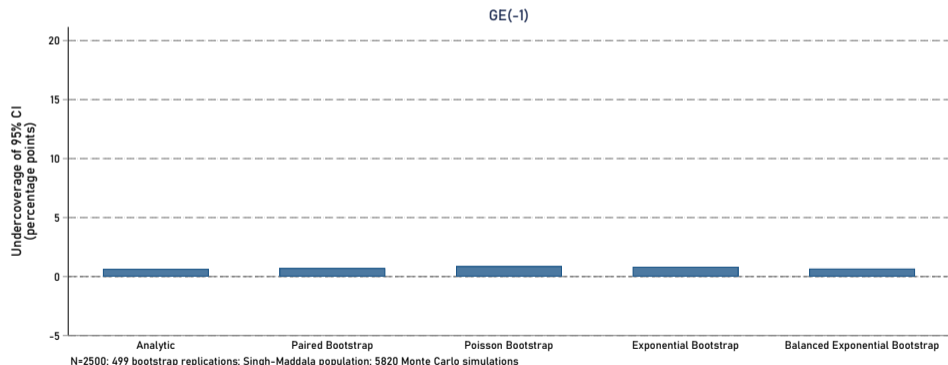
Ex.: coverage rate of 95% bootstrapped CI (normal approximation) for inequality measures



Does it really 'work'?

Statistical properties of exchangeable bootstraps similar to paired bootstrap

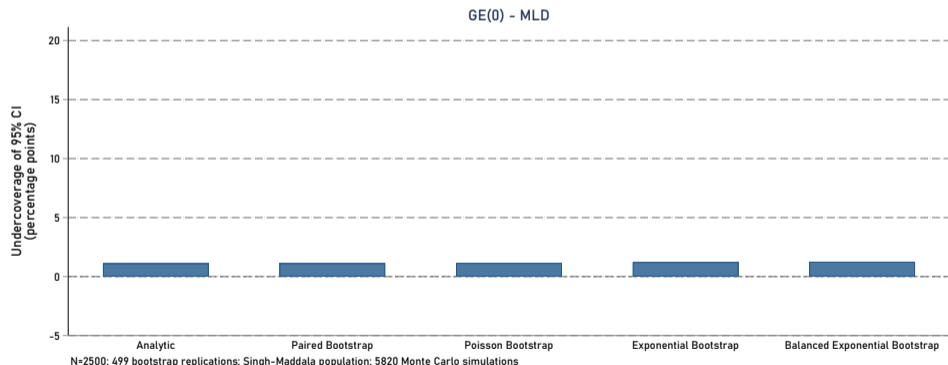
Ex.: coverage rate of 95% bootstrapped CI (normal approximation) for inequality measures



Does it really 'work'?

Statistical properties of exchangeable bootstraps similar to paired bootstrap

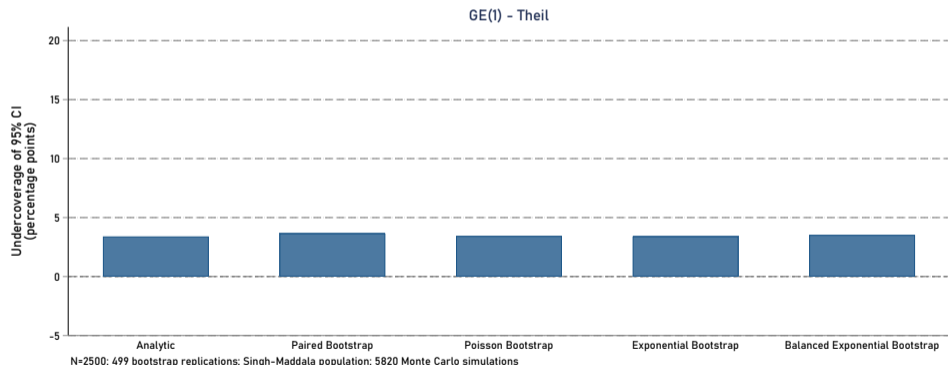
Ex.: coverage rate of 95% bootstrapped CI (normal approximation) for inequality measures



Does it really 'work'?

Statistical properties of exchangeable bootstraps similar to paired bootstrap

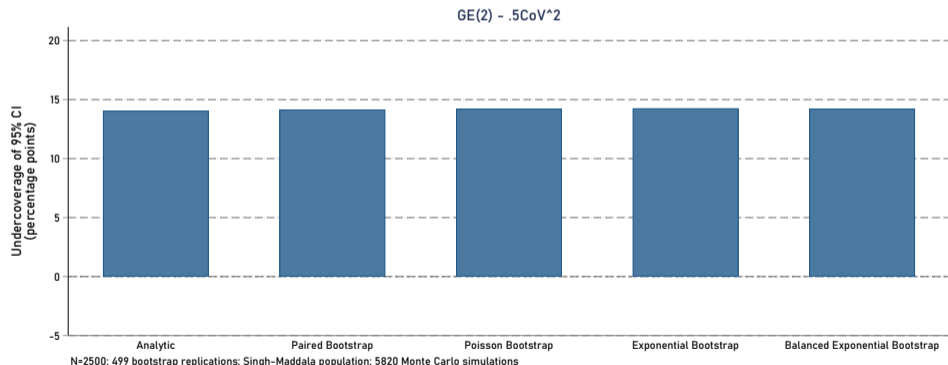
Ex.: coverage rate of 95% bootstrapped CI (normal approximation) for inequality measures



Does it really 'work'?

Statistical properties of exchangeable bootstraps similar to paired bootstrap

Ex.: coverage rate of 95% bootstrapped CI (normal approximation) for inequality measures



Conclusion

- Exchangeably weighted bootstrap schemes are straightforward and attractive (exponential bootstrap in particular)
 - ... and `exbsample` can help
 - Exploiting replication weights is admittedly limited if using built-in (prefix) commands only (some further programming for handling replications may be needed for more than small-scale applications)
- ⇒ Ideas for a revamp of Stata's built-in bootstrap capabilities in some future release maybe?
- » e.g., allowing bootstrap weights with standard `bootstrap` prefix, more bootstrap CI calculation (notably with `svy bootstrap`), calculation of 'studentized' bootstrap CIs

References

- Chernozhukov, V., Fernández-Val, I. and Melly, B. (2013), 'Inference on counterfactual distributions', *Econometrica* **81**(6), 2205–2268.
URL: <http://dx.doi.org/10.3982/ECTA10582>
- Davison, A. C. and Hinkley, D. V. (1997), *Bootstrap Methods and Their Application*, Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press, USA.
- Efron, B. and Tibshirani, R. J. (1993), *An Introduction to the Bootstrap*, Chapman and Hall, London, UK.
- Kolenikov, S. (2010), 'Resampling variance estimation for complex survey data', *Stata Journal* **10**, 165199.

- Otsu, T. and Rai, Y. (2017), 'Bootstrap inference of matching estimators for average treatment effects', *Journal of the American Statistical Association* **112**(520), 1720–1732.
- Praestgaard, J. and Wellner, J. A. (1993), 'Exchangeably weighted bootstraps of the general empirical process', *The Annals of Probability* **21**(4), 2053–2086.
- Roodman, D., Orregaard Nielsen, M., MacKinnon, J. G. and Webb, M. D. (2019), 'Fast and wild: Bootstrap inference in Stata using boottest', *The Stata Journal* **19**(1), 4–60.
- Van Kerm, P. (2013), 'Rhsbsample: Stata module for repeated half-sample bootstrap sampling', Statistical Software Components, Boston College Department of Economics.
URL: <http://ideas.repec.org/c/boc/bocode/s457697.html>