Programming, Data Management and Visualization

Module D: **Reporting results**

Alexander Ahammer

Department of Economics, Johannes Kepler University, Linz, Austria
Christian Doppler Laboratory Ageing, Health, and the Labor Market, Linz, Austria

β version, *more of less complete*
Last updated: Monday 20th January, 2020 (13:23)

Introduction

- In this module we will very simply learn how to
  - Exports results from different commands
  - Automate the exporting of these results
  - (and some selected issues regarding results)

- Doesn't sound like much, right? This is an extremely important topic though.

- Whatever work you do with Stata, you want your summary statistics, tests, and regressions to automatically generate a file you can include in your Word or LaTeX document.
  - NEVER copy coefficients, statistics, etc. manually — **NEVER!**
  - Why? Because this is error-prone and time-consuming.
  - Also, please don't screenshot regression outputs and copy them in your paper.
  - Visually appealing tables and graphs go a long way (more on that in module E).

Brief recap on stored results

- All Stata commands store their results (usually silently), and you can greatly improve the efficiency and simplicity of your codes if you know how to address and reuse them.
  - Remember: Avoid *magic numbers!*
- Stata distinguishes between r-class and e-class commands, with `ereturn list` and `return list` you can check what Stata saved after running a command.
- Importantly, results from can be saved as both **scalars** and **matrices.** Remember how to refer to each of the two.

Brief recap on stored results

```
. su sl_dur

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
      sl_dur |    322,375    6.020889    5.386769          1         44

. return list

scalars:
                  r(N) =  322375
              r(sum_w) =  322375
               r(mean) =  6.020888716556805
                r(Var) =  29.01728221067055
                 r(sd) =  5.386769181120586
                r(min) =  1
                r(max) =  44
                r(sum) =  1940984
```

Brief recap on stored results

```
. qui reg sl_dur p_female

. ereturn list

scalars:
                  e(N) =  322375
               e(df_m) =  1
               e(df_r) =  322373
                  e(F) =  93.52778612671118
                 e(r2) =  .0002900387422187
               e(rmse) =  5.385996292244489
                e(mss) =  2713.143438145518
                e(rss) =  9351704.191953391
               e(r2_a) =  .0002869376451626
                 e(ll) =  -1000245.078030031
               e(ll_0) =  -1000291.835430849
               e(rank) =  2

macros:
            e(cmdline) : "regress sl_dur p_female"
              e(title) : "Linear regression"
          e(marginsok) : "XB default"
                e(vce) : "ols"
             e(depvar) : "sl_dur"
                e(cmd) : "regress"
         e(properties) : "b V"
            e(predict) : "regres_p"
              e(model) : "ols"
          e(estat_cmd) : "regress_estat"

matrices:
                  e(b) :  1 x 2
                  e(V) :  2 x 2

functions:
             e(sample)
```

D.1

Creating publication-quality tables

Our goal for this lecture ...

- Suppose you write a thesis on the effect of tenure on sick leave duration using the data in `pdmv_sl.dta`. For worker i at start of sick leave t, your model reads

$$\texttt{sl\_dur}_{it} = \beta\texttt{e\_tenure}_{it} + \mathbf{x}_{it}\boldsymbol{\delta}' + u_{it}. \quad (1)$$

- You want to present your main results in a table showing first the $\widehat{\beta}$ from a model without any covariates $\mathbf{x}_{it}$, and then sequentially adding appropriate personal and firm level controls in additional columns.

- The creation of this table should work fully automatically.

- Make sure to install the user-written command `estout` before we start. An alternative is `outreg`, although I recommend the former.

Storing estimated results

- First, we should know how to save estimation results.
- This is pretty easily done with the `estimates store` command.
  - You can either use `est sto` *`modelname`* **after** an estimation command or
  - as a **prefix** once you have installed `estout` $\longrightarrow$ `est sto` *`modelname`* :
  - You can freely choose the *`modelname`*
- Up to 300 models can be saved, stored estimates can always be reviewed and loaded back into memory by typing `est replay` *`modelnamelist`* or `est use` if previously saved locally.
- `est table` *`modelnamelist`* generates a very simple and plain table from the models in *`modelnamelist`*

Storing estimated results

```
. use "data/pdmv_sl.dta", clear
(All sick leaves 2004-2012 for 10% sample of Austrian employees)

. replace e_tenure = e_tenure / 365.25
variable e_tenure was int now float
(322,153 real changes made)

. qui reg sl_dur e_tenure

. est store m1

. qui reg sl_dur e_tenure p_female e_wage

. est store m2

. est table m1 m2, b(%9.3f) se(%9.3f)

--------------------------------------
    Variable |     m1           m2
-------------+------------------------
    e_tenure |     0.074        0.095
             |     0.001        0.002
    p_female |                 -0.339
             |                  0.020
      e_wage |                 -0.000
             |                  0.000
       _cons |     5.628        6.271
             |     0.012        0.022
--------------------------------------
                          legend: b/se
```

Using `estout` to produce publication-quality tables

Let's try to reproduce this table now:

	OLS			Fixed effects	
	(1)	(2)	(3)	(4)	(5)
ln(tenure)	0.281***	0.011	0.118***	0.120***	0.248***
	(0.006)	(0.007)	(0.008)	(0.011)	(0.013)
Demographics	No	Yes	Yes	No	Yes
Employment info	No	No	Yes	No	Yes
Number of obs.	321,737	288,530	288,530	321,737	288,530
F-statistic	2224.0	999.9	937.3	122.3	183.1
p-value	0.000	0.000	0.000	0.000	0.000

Using estout to produce publication-quality tables

- Ben Jann's estout package is a wrapper for estimates table, and provides a full-featured solution to prepare publication-ready tables.
- Can export tables to different formats:
  - LaTeX
  - delimited txt, Word and rtf
  - HTML
- The companion command estadd allows the addition of user-specified statistics to the e() arrays accessible by estimates.
- A simplified version of estout is available as esttab.

Using estout to produce publication-quality tables

- estout can be tedious at first, but as soon as you have coded a few tables you can always refer back to older work, which speeds up the process.
- There is an excellent online documentation and collection of examples for estout, esttab, and the other commands in Ben Jann's suite:
  - http://repec.sowi.unibe.ch/stata/estout/
- This is the code I used for this table:

```
#delimit ;
esttab ols_* fe_* using slides/tables/eststo1.tex, keep(ln_tenure) cells(b(fmt(3) star) se(par fmt(3)))
stats(N F p, fmt(%9.0gc 1 3) label("Number of obs." `"\(F\)-statistic"' `"\(p\)-value"'))
star(* 0.10 ** 0.05 *** 0.01) gaps label collabels(none) varlabels(
    ln_tenure "ln(tenure)" ) indicate(
    "Demographics = p_female"
    "Employment info = *e_class*"
  )
mgroups("OLS" "Fixed effects", pattern(1 0 0 1 0) span prefix(\multicolumn{@span}{c}{) suffix(})
erepeat(\cmidrule(lr){@span}))
nomtitles booktabs replace ;
#delimit cr
```

- Let's go through it together.

Using `estout` to produce publication-quality tables

If you don't work with LaTeX you can easily tweak the command to export in other formats, for example as a text file:

	OLS			Fixed effects	
	(1)	(2)	(3)	(4)	(5)
ln(tenure)	0.281***	0.011	0.118***	0.120***	0.248***
	(0.006)	(0.007)	(0.008)	(0.011)	(0.013)
Demographics	No	Yes	Yes	No	Yes
Employment info	No	No	Yes	No	Yes
Number of obs.	321,737	288,530	288,530	321,737	288,530
F-statistic	2224.0	999.9	937.3	122.3	183.1
p-value	0.000	0.000	0.000	0.000	0.000

Adding computed statistics to `estout` tables

- `estadd` allows you to
  - add all `e()` statistics saved by the estimation command
  - add a custom scalar, even from other commands or matrices

- Example: You run an IV regression and you want to display the first stage F-statistic as well as the partial R^2 in your table. The F-statistic is saved in `ivreg2`'s `e()` results, but the partial R^2 is hidden in its `e(first)` matrix. In order to access both, you type this after the regression command:

```
ivreg2 y (x = z)
    estadd Ffirst = e(widstat)
    mat first = e(first)
    estadd partialR2 = first[2,1]
```

- Note also that `estadd ysumm` adds summary statistics of the outcome variable based on the regression's `e(sample)`. In the table you can then add `ymean`, `ysd`, `ymin`, and `ymax` as statistics.

- Let's see how it works in action.

```
. eststo clear
. eststo m1: qui xtreg sl_dur ln_tenure p_age, fe cluster(id_worker)
.         estadd ysumm
added scalars:
                e(ysd) =  5.3756565
               e(ymax) =  44
               e(ymin) =  1
              e(ymean) =  6.0136851
.         qui corr sl_dur ln_tenure
.         estadd scalar corrho = r(rho)
added scalar:
             e(corrho) =  .08401579
. eststo m2: qui xtreg sl_dur ln_tenure p_age if p_female == 1, fe cluster(id_worker)
.         estadd ysumm
added scalars:
                e(ysd) =  5.4094213
               e(ymax) =  44
               e(ymin) =  1
              e(ymean) =  5.9008974
.         qui corr sl_dur ln_tenure if e(sample)
.         estadd scalar corrho = r(rho)
added scalar:
             e(corrho) =  .08182695
. eststo m3: qui xtreg sl_dur ln_tenure p_age if p_female == 0, fe cluster(id_worker)
.         estadd ysumm
added scalars:
                e(ysd) =  5.3513607
               e(ymax) =  44
               e(ymin) =  1
              e(ymean) =  6.089984
.         qui corr sl_dur ln_tenure if e(sample)
.         estadd scalar corrho = r(rho)
added scalar:
             e(corrho) =  .08478323
```

Adding computed statistics to `estout` tables

```
. #delimit ;
delimiter now ;
. esttab m* using slides/tables/eststo2.tex, r cells(b(fmt(3) star) se(par fmt(3)))
>         stats(ymean ysd N N_g corrho, fmt(2 2 %9.0gc %9.0gc 3)
>         label("Mean of tenure" "SD of tenure" "Number of obs."
>                   "Number of workers" "Raw correlation"))
>         star(* 0.10 ** 0.05 *** 0.01) gaps label collabels(none) varlabels(
>                 ln_tenure       "ln(tenure)"
>                 p_age           "Age in years"
>                 _cons           "Intercept")
>         mtitles("Full sample" "Women" "Men") booktabs
>   ;
(output written to slides/tables/eststo2.tex)

. #delimit cr
delimiter now cr
```

Adding computed statistics to `estout` tables

	(1) Full sample	(2) Women	(3) Men
ln(tenure)	0.189***	0.238***	0.158***
	(0.011)	(0.019)	(0.014)
Age in years	-0.141***	-0.122***	-0.153***
	(0.005)	(0.009)	(0.007)
Intercept	9.965***	8.770***	10.724***
	(0.186)	(0.305)	(0.235)
Mean of tenure	6.01	5.90	6.09
SD of tenure	5.38	5.41	5.35
Number of obs.	321,737	129,825	191,912
Number of workers	52,680	22,812	29,868
Raw correlation	0.084	0.082	0.085

Presenting transformed coefficients

- Sometimes you may want to present certain transformations of coefficients, for example:
  - Exponentiated coefficients in non-linear models
  - Marginal effects
  - Linear transformations (e.g., coefficient $\times$ 2)
- Reporting exponentiated coefficients is easy, simply use the `eform` option in `estout`.
- For marginal effects, use `margins` with the `post` option after the estimation command and then save the results using `eststo`. For linear transformations save the results from the user-written `lincom` wrapper `lincomest`.

Presenting transformed coefficients

```
. g sl3 = sl_dur > 3

. qui probit sl3 p_female p_age

. margins, dydx(*) post

Average marginal effects                        Number of obs     =    322375
Model VCE    : OIM

Expression   : Pr(sl3), predict()
dy/dx w.r.t. : p_female p_age

------------------------------------------------------------------------------
             |            Delta-method
             |      dy/dx   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
    p_female |   -.027932   .0016791   -16.63   0.000     -.031223    -.024641
       p_age |   .0061386   .0000723    84.86   0.000     .0059968    .0062803
------------------------------------------------------------------------------

. eststo m_probit

. estout m_probit, cells("b(fmt(3) star) se(fmt(3))")

------------------------------------------
                 m_probit
                        b               se
------------------------------------------
p_female           -0.028***         0.002
p_age               0.006***         0.000
------------------------------------------
```

Presenting transformed coefficients

```
. eststo m1: qui reg sl_dur p_female p_age

. eststo m2: lincomest 2*p_female
Confidence interval for formula:
2*p_female
```

sl_dur	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
(1)	-.272862	.038023	-7.18	0.000	-.3473859	-.198338

```
. estout m1 m2, keep(p_female (1)) varlabels((1) "p_female * 2") ///
>                            cells("b(fmt(3) star) se(fmt(3))")
```

	m1		m2	
	b	se	b	se
p_female	-0.136***	0.019		
p_female * 2			-0.273***	0.038

D.2

Automation of result reporting

Using loops for automatic table and graph creation

- Loops can be helpful to create sets of tables and graphs.
- Write estimation or graph commands only **once,** and try to loop over the command for different variations of it. This is the least error-prone method of estimating different regressions or drawing different graphs.
- For example:

```
eststo base: reg y x
forval i = 0/1 {
    eststo m_female‘i’: reg y x if female = ‘i’
}
esttab _all
```

- *Big data tip* If you perform commands (esp. tests, regressions) only on **subsets** of the data, **drop the obs you don't use as early as possible in your code.** If you loop over subsets, use `preserve` and then `drop` instead of the command with an `if` condition.

Using loops for automatic table and graph creation

For graphs, consider the following example: We want to plot the relationship between age and wage for all years in the data, assigning different colors to men and women.

```
. g year = yofd(sl_start)

. forval y = 2005/2012 {
  2.   loc graphlist `graphlist´ y`y´
  3.         binscatter e_wage p_age if year == `y´, by(p_female) nq(70) line(none) ///
>         scheme(plotplain) mcolors(orange_red emidblue) ytitle("") ylab(, angle(vertical) ///
>         format(%9.0gc) nogrid) legend(order(1 "Men" 2 "Women") cols(2) pos(2)) xlab(, nogrid) ///
>         xtitle("") title("{it:`y´}", size(medsmall)) nodraw name(y`y´, replace)
  4. }

.
. grc1leg `graphlist´, name(graph, replace) scheme(plotplain) pos(1) imargin(tiny) ycommon ///
>         l1title("Wage in 1,000 EUR", size(small)) ///
>         b1title("Age in years", size(small))

.
. gr display graph, xsize(6) scale(1.1)

. gr export "slides/graphs/D_scattery.pdf", as(pdf) replace
(file slides/graphs/D_scattery.pdf written in PDF format)

. window manage close graph _all
```

Here I use the `binscatter` command, which groups the x-axis variable into equal-sized bins, computes the mean of the x-axis and y-axis variables within each bin, and then creates a scatterplot of these data points. This is helpful when working with *big data sets*, where scatter plots can often become too crowded due to the sheer amount of data points.

Using loops for automatic table and graph creation

Using loops for automatic table and graph creation

A few notes on the graph we just drew:

- In the graph command we use the `name()` option, which allows you to name the graph and store it in memory.
  - Most graph commands also allow for options such as `save()`, `saving()`, or `savegraph()`, which directly save the graph locally. However, since we do not need the individual graphs *per se*, but only the combined graph, it suffices to keep them in memory.
- I combine the graphs with `grc1leg`, which supports for a common legend on the bottom. The command has some limitations, that's why I `name()` it again and reopen it with `graph display` in order to adjust, e.g., the `xsize` and font scaling.
- Always save graphs as **vector graphics,** Stata supports pdf and eps.
- `nodraw` prevents the graph window from opening, this is useful if otherwise many windows would be opened.
- If you produce many graphs in a loop it may be helpful to close the graphs:

```
window manage close graph _all
```

Automating table creation

- Scientific text documents include a lot of statistical output, **manual inclusion** of this output is **tedious and error-prone.**

- Fully automating table creation means that you press a button in Stata and the resulting table will automatically appear in your paper or thesis.

- This can only be done with LaTeX through its `input` capabilities. With Word you will always have to copy updated tables.
  - Learn LaTeX $\longrightarrow$ steep learning curve and its typesetting is so much more beautiful than Word!

- I found that it is most efficient to only export coefficients and statistics from Stata; no numbers, model names, etc.

- Whenever you change something in the model, you simply have to compile your LaTeX file again — there is no need to change something unless you add additional columns (i.e., outcomes).

LaTeX

- It is not a word processor $\longrightarrow$ it is not WYSIWYG!
- Document preparation system for high-quality typesetting.
- Ideal for medium-to-large technical or scientific documents.
- Widely used in academia.
- But also for almost any form of publishing.
- Encourages authors to focus on content (and not on appearance).
- Facilitates automatic inclusion of statistical output.
- More here: `http://latex-project.org`.

Automating table creation

This is how a command exporting only a single coef + statistics could look like:

```
#delimit ;
estout $outcomes using "${paperdir}/tables/export/reg_rdd_coef.tex", r varlabels(
   MLdur "~~ Prenatal maternity leave")
keep(MLdur) style(tex) starlevels("\sym{*}" 0.10 "\sym{**}" 0.05 "\sym{***}" 0.01)
stats(N ymean ysd Fwid, fmt(%18.0gc 2 2 2) layout("\multicolumn{1}{c}{@}")
label(`"~~~~ No. of observations"'
      `"~~~~ Mean of outcome"'
      `"~~~~ Std.\ dev.\ of outcome"'
      `"~~~~ Kleinbergen-Paap \(rK\) Wald \(F\)-statistic"'))
collabels(none) cells(b(fmt(3) star) se(fmt(3) par)) prefoot("[1ex]")
mlabel(, none) postfoot("[2ex]")
;
#delimit cr
```

Automating table creation

This is how a table looks in LaTeX (bottom of the code is cut-off):

```
\begin{table}
    \centering
    \caption{Comparison of estimated treatment effects on health at birth outcomes obtained via different estimators}
    \small
    \resizebox{\linewidth}{!}{
    \begin{threeparttable}
        {
            \def\sym#1{\ifmmode^{#1}\else\(^{#1}\)\fi}
            \begin{tabular}{lSSSSS}
                \toprule
                &\multicolumn{1}{c}{(1)}&\multicolumn{1}{c}{(2)}&\multicolumn{1}{c}{(3)}&\multicolumn{1}{c}{(4)}&\multicolumn{1}{c}{(5)}\\
                &\multicolumn{1}{c}{Birth}&\multicolumn{1}{c}{Low birth}&\multicolumn{1}{c}{Symmetric}&\multicolumn{1}{c}{\multirow{2}{*}{Length}}&\multicolumn{1}{c}{Premature}\\
                &\multicolumn{1}{c}{weight}&\multicolumn{1}{c}{weight}&\multicolumn{1}{c}{growth restr.}&\multicolumn{1}{c}{}&\multicolumn{1}{c}{birth\tnote{$\dag$}}\\
                \midrule
                \multicolumn{6}{l}{\it Panel A. RDD}\\
                \input{tables/export/reg_rdd_coef.tex}

                \multicolumn{6}{l}{\it Panel B. RDD-DiD}\\
                \input{tables/export/reg_rdd-did_coef.tex}

                \multicolumn{6}{l}{\it Panel C. DiD}\\
                \input{tables/export/reg_did_coef.tex}

                \bottomrule
            \end{tabular}
        }
        \begin{tablenotes}\setlength\labelsep{0pt}\footnotesize
            \item \textit{Notes:} This table presents compares estimated local average treatment effects of extending compulsory ML duration by two weeks on health at birth outcomes
            obtained via different estimators. Each cell represents a separate regression. The sample in panel A consists of working mothers giving birth in April and June 1974, in
```

• • •

Essentially, you set up the table in LaTeX, including column headers, horizontal lines, table notes, etc. With the `input` command you include the stored estimation result. Indeed you can create such a table also in Stata.

This is part of the final result in the paper:

	(1) Birth weight	(2) Low birth weight	(3) Asymmetric growth restr.	(4) Length	(5) Premature birth[†]
Panel A. RDD					
Prenatal maternity leave	−0.005	0.000	−0.000	−0.001	0.005
	(0.003)	(0.003)	(0.003)	(0.001)	(0.004)
No. of observations	7,350	7,350	7,350	7,350	7,350
Mean of outcome	5.77	0.06	0.04	3.92	0.06
Std. dev. of outcome	0.19	0.23	0.19	0.06	0.24
Kleinbergen-Paap rK Wald F-statistic	756.45	756.45	756.45	756.45	756.45
Panel B. OLS (only pre-treatment period)					
Prenatal maternity leave	0.006***	−0.006***	−0.002	0.001**	−0.011***
	(0.002)	(0.002)	(0.002)	(0.001)	(0.003)
No. of observations	3,721	3,721	3,721	3,721	3,721
Mean of outcome	5.77	0.06	0.04	3.92	0.06
Std. dev. of outcome	0.19	0.23	0.19	0.06	0.24
Panel C. RDD-DiD					
Prenatal maternity leave	−0.003	0.001	−0.001	−0.000	
	(0.005)	(0.006)	(0.005)	(0.001)	
No. of observations	10,424	10,424	10,424	10,424	
Mean of outcome	5.78	0.05	0.03	3.92	
Std. dev. of outcome	0.19	0.22	0.18	0.06	
Kleinbergen-Paap rK Wald F-statistic	755.75	755.75	755.75	755.75	

• • •

D.3

Selected issues

How to plot regression results in a graph

- Sometimes you want to graphically represent estimated coefficients.
- If they come from one model, the user-written command `coefplot` or `margins` with `marginsplot` can be helpful. The former can also handle coefficients from different models, but I prefer a more general solution using `parmest`.
- `parmest` **produces datasets with one observation for each of a set of estimated parameters** from a model, and variables containing estimates, confidence intervals, p-values and other parameter attributes.
- Let's go back to the example from before, where sick leave duration is regressed on tenure. Suppose we want to graphically show the change in the tenure coefficient when covariates are sequentially added.

How to plot regression results in a graph

How parmest works

```
. qui reg sl_dur ln_tenure, r
. qui parmest, label format(estimate min95 max95 p) list(,) saving("do/results/tmp.dta", replace) level(95)
. use "do/results/tmp.dta", clear
. list
```

	parm	label	estimate	stderr	dof	t	p	min95	max95
1.	ln_tenure		.28081672	.00595465	321735	47.159226	0	.26914577	.29248766
2.	_cons	Constant	4.1462504	.03984067	321735	104.07079	0	4.0681638	4.2243369

⟹ parmest creates a new dataset where **variables are saved as rows** and **coefficient estimates as well as certain statistics are saved as columns.**

How to plot regression results in a graph

Plotting coefficients from different models

To draw the graph we first save all models in separate datasets using `parmest`. Along with the coefficient estimates and standard statistics we also save differently sized confidence intervals:

```
. qui reg sl_dur ln_tenure, r

. qui parmest, label format(estimate min95 max95 p n) list(,) ///
>         saving("do/results/m1.dta", replace) level(90 95 99)

. qui reg sl_dur ln_tenure $cov_p, r

. qui parmest, label format(estimate min95 max95 p n) list(,) ///
>         saving("do/results/m2.dta", replace) level(90 95 99)

. qui reg sl_dur ln_tenure $cov_p $cov_p2, r

. qui parmest, label format(estimate min95 max95 p n) list(,) ///
>         saving("do/results/m3.dta", replace) level(90 95 99)

. qui reg sl_dur ln_tenure $cov_p $cov_p2 $cov_e, r

. qui parmest, label format(estimate min95 max95 p n) list(,) ///
>         saving("do/results/m4.dta", replace) level(90 95 99)
```

The next step is to append all these datasets into one, keeping only the coefficient on tenure from each dataset. Note that I use a very general solution that consolidates all datasets in the folder we saved the regression results in:

```
. loc o: dir "do/results/" file "*.dta"

. tokenize `"`o´"´

. use if parm == "ln_tenure" using "do/results/`1´", clear

. g str reg = substr("`1´", 1, strpos("`1´",".dta")-1)

. loc j = 2

. while `"``j´´"´ != "" {
  2.  qui {
  3.   append using "do/results/``j´´", gen(_a`j´)
  4.   keep if parm == "ln_tenure"
  5.   replace reg = substr("``j´´", 1, strpos("``j´´", ".dta")-1) if _a`j´ == 1
  6.   drop _a`j´
  7.   loc ++j
  8.  }
  9. }

. list parm estimate stderr reg
```

	parm	estimate	stderr	reg
1.	ln_tenure	.28081672	.00595465	m1
2.	ln_tenure	.00463412	.00681765	m2
3.	ln_tenure	.01133682	.00722966	m3
4.	ln_tenure	.11802664	.00846961	m4

How to plot regression results in a graph

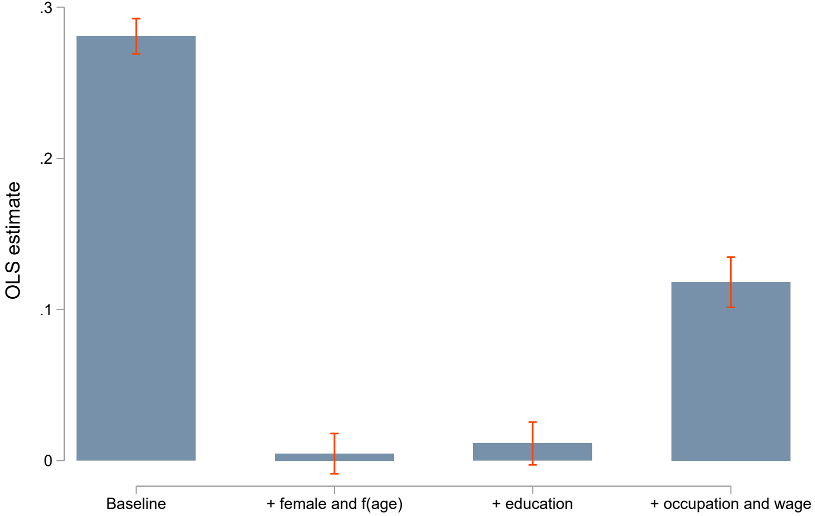
Plotting coefficients from different models

Now we draw the graph:

```
. encode reg, gen(model)
. #delimit ;
delimiter now ;
. tw      (bar estimate model, barw(0.6) fi(100) color("118 145 169"))
>         (rcap min95 max95 model, lcolor("255 69 0")),
>         ylab( , nogrid)
>         xlab(1 "Baseline" 2 "+ female and f(age)" 3 "+ education" 4 "+ occupation and wage", nogrid)
>         ytitle("OLS estimate")
>         xtitle("")
>         legend(off)
>   ;
. #delimit cr
delimiter now cr
. gr export "slides/graphs/D_parmest1.pdf", as(pdf) replace
(file slides/graphs/D_parmest1.pdf written in PDF format)
```

How to plot regression results in a graph

Plotting coefficients from different models



How to plot regression results in a graph

Plotting coefficients from one single model

```
. xtile d_tenure = e_tenure, n(10)

. qui reg sl_dur i.d_tenure $cov_p $cov_p2 $cov_e, r

. qui parmest, label format(estimate min95 max95 p n) list(,) ///
>         saving("do/results/np.dta", replace) level(90 95 99)

. use do/results/np.dta, clear

. keep if strpos(parm, "tenure")
(14 observations deleted)

. g d = substr(parm, 1, strpos(parm,".")-1)

. replace d = "1" if d == "1b"
(1 real change made)

. destring d, replace
d: all characters numeric; replaced as byte

. list parm estimate stderr d, sep(0)
```

	parm	estimate	stderr	d
1.	1b.d_tenure	0	0	1
2.	2.d_tenure	.27476247	.0430017	2
3.	3.d_tenure	.38190038	.04358004	3
4.	4.d_tenure	.45952766	.04489613	4
5.	5.d_tenure	.51199981	.04558257	5
6.	6.d_tenure	.48538377	.04674593	6
7.	7.d_tenure	.43303092	.04736986	7
8.	8.d_tenure	.49602748	.04882991	8
9.	9.d_tenure	.6006142	.05186733	9
10.	10.d_tenure	.6668995	.05872359	10

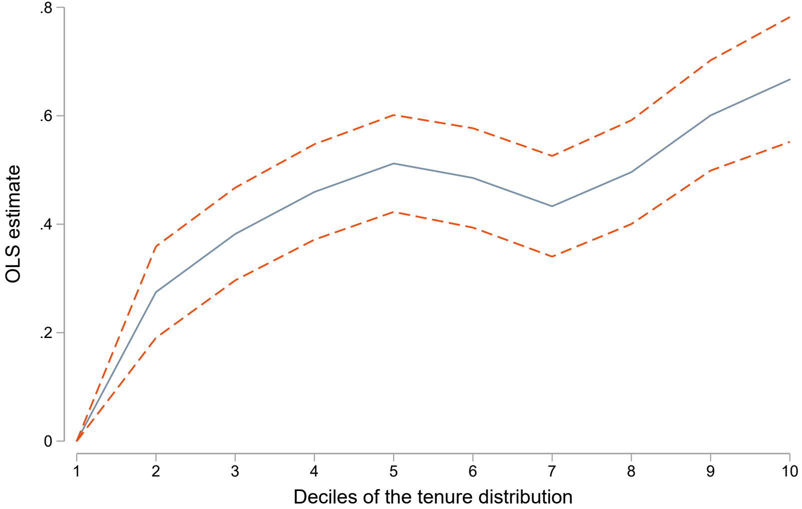
How to plot regression results in a graph

Plotting coefficients from one single model

```
. #delimit ;
delimiter now ;
. tw      (line estimate d, lcolor("118 145 169"))
>         (rline min95 max95 d, lpattern(dash) lcolor("255 69 0")),
>         xlab(1(1)10, nogrid)
>         ylab(, nogrid)
>         ytitle("OLS estimate")
>         xtitle("Deciles of the tenure distribution")
>         legend(off)
>   ;
. #delimit cr
delimiter now cr
. gr export "slides/graphs/D_parmest2.pdf", as(pdf) replace
(file slides/graphs/D_parmest2.pdf written in PDF format)
```

How to plot regression results in a graph

Plotting coefficients from <u>one single</u> model



How to plot regression results in a graph

Takeaway:

- `parmest` allows you to plot regression coefficients from multiple models in one graph.
- If you want to plot coefficients from a single model, `coefplot` or `marginsplot` probably saves some time, but the `parmest` solution allows more customization.
  - Also, `margins` is extremely slow $\longrightarrow$ problem with *big data*
- In module E we talk a lot about how to *present* data. When it comes to presenting regression coefficients, keep this in mind:
  - Maximize information–ink ratio (this applies to every graph and table!)
  - Always plot confidence intervals!
  - For estimates of a categorical variable: Choose the reference group wisely
  - Use bars, lines, or scatters depending on how the coefficients are related

How to make *custom* tables

- Sometimes you may want to create customized tables that are not supported by `estout`.
- The easiest way of creating custom tables is to store the information in a matrix and then copy the matrix to Excel. I will also show you a solution which uses `putexcel` to create a custom spread-sheet in Excel.
  - Another solution uses `file write` and Statas `export` capabilities, see chapter 9.4 in the textbook.
- From Excel you can easily export the table to Word or convert it into a LaTeX code with the *Excel2LaTeX* macro.[1]
- First, let's consider a very simple example on how to store information in matrices.

[1] `https://github.com/krlmlr/Excel2LaTeX`

How to make custom tables

```
. mat def output = J(2,2,.)

. mat colnames output = n mean

. mat rownames output = men women

. mat list output

symmetric output[2,2]
          n   mean
  men     .
women     .      .

.
. su sl_dur if p_female == 0

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
      sl_dur |     192279     6.09635    5.362018          1         44

. mat output[1,1] = r(N)

. mat output[1,2] = r(mean)

.
. su sl_dur if p_female == 1

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
      sl_dur |     130096    5.909359    5.421241          1         44

. mat output[2,1] = r(N)

. mat output[2,2] = r(mean)

.
. mat list output

output[2,2]
               n       mean
  men     192279  6.0963496
women     130096  5.9093592
```

How to make custom tables

Using putxcel

Goal now:

- Split the sample in 5 age groups according to quintiles of the age distribution.
- Run a regression of `sl_dur` on a blue collar dummy for each age sample.
- Make a table which presents these regression coefficients as rows.
- Add also the sample mean of both `sl_dur` and the blue collar dummy as columns.

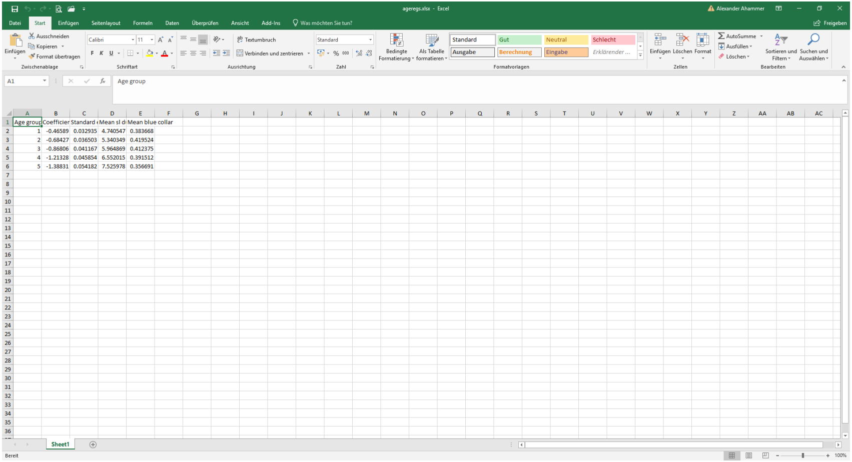
This is a very simple exercise, but can be extended in many ways. Note also that you could reimport the Excel table with `import excel`.

Also, here we only need one coefficient. If you need **all coefficients** from a model, paste the coefficient vector `e(b)` into your matrix or excel file. Standard errors can be obtained from the variance-covariance matrix `e(V)`.

```
. xtile agegroup = p_age, n(5)
. g bluecollar = e_class == 2 if !missing(e_class)
.
. mat def A = J(5,5,.)
. putexcel set "do/results/ageregs.xlsx", replace
.
. forval i = 1/5 {
  2.          mat A[`i´,1] = `i´
  3.          * regressions
.         qui reg sl_dur bluecollar if agegroup == `i´
  4.          mat A[`i´,2] = _b[bluecollar]
  5.          mat A[`i´,3] = _se[bluecollar]
  6.          * sample means
.         su sl_dur if agegroup == `i´, meanonly
  7.          mat A[`i´,4] = r(mean)
  8.          su bluecollar if agegroup == `i´, meanonly
  9.          mat A[`i´,5] = r(mean)
 10. }
.
. putexcel A1 = ("Age group")
file do/results/ageregs.xlsx saved
. putexcel B1 = ("Coefficient")
file do/results/ageregs.xlsx saved
. putexcel C1 = ("Standard error")
file do/results/ageregs.xlsx saved
. putexcel D1 = ("Mean sl duration")
file do/results/ageregs.xlsx saved
. putexcel E1 = ("Mean blue collar")
file do/results/ageregs.xlsx saved
. putexcel A2 = matrix(A)
file do/results/ageregs.xlsx saved
```

How to make custom tables

Using `putxcel`



Age grour	Coefficier	Standard (	Mean sl di	Mean blue collar
1	-0.46589	0.032935	4.740547	0.383668
2	-0.68427	0.036503	5.340349	0.419524
3	-0.86806	0.041167	5.964869	0.412375
4	-1.21328	0.045854	6.552015	0.391512
5	-1.38831	0.054182	7.525978	0.356691