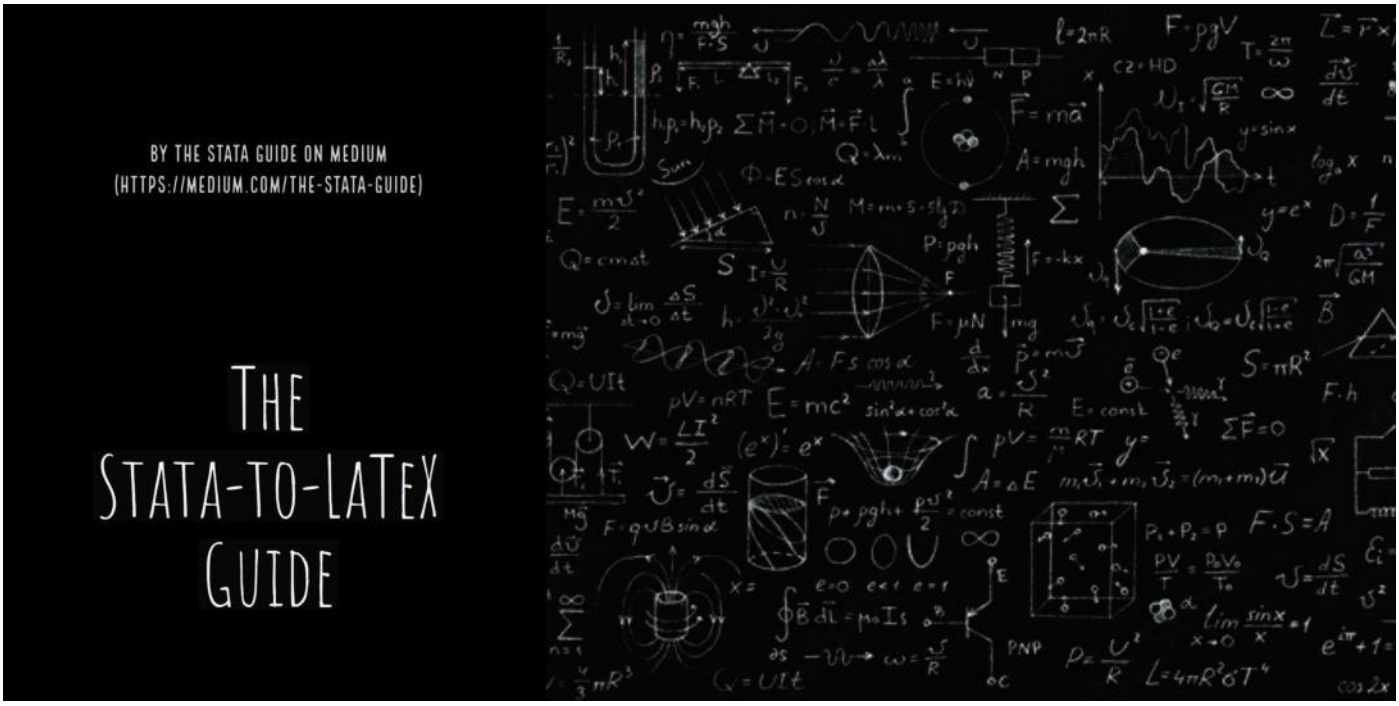


The Stata-to-LaTeX guide

Asjad Naqvi Follow
Feb 25 · 30 min read ★

Last updated: 08 Dec 2021 (Overleaf doc updated to make it compatible with TexLive 2021. See change log in document for details.)

In this guide learn how to export Stata tables and regressions to LaTeX to generate customized tables.

Table 2.7: Summary statistics - Grouping variables

	Mean	SD	Min	Max	N
COVID-19 indicators					
Total cases (units)	612,132	2,441,679	1	28,190,159	180
Total deaths (units)	14,233	48,604	1	500,310	172
Total tests (units)	10,525,008	37,326,709	29,225	327,903,249	89
Total vaccinations (units)	2,609,408	8,552,571	0	64,177,474	62
Socio-economics indicators					
Median age (years)	30.17	9.18	15.10	48.20	172
Age 65+ (years)	8.59	6.24	1.14	27.05	170
Life expectancy (years)	73.02	7.77	53.28	86.75	184
Extreme poverty (%)	14.10	20.60	0.10	77.60	119
HDI (Index)	72.13	15.32	39.40	95.70	175

Note: Yes, there are countries with just one case and one death reported to date!

Baseline	0.003**	0.088	0.005*	0.033
	(0.002)	(0.078)	(0.003)	(0.035)
Region FE	0.003***	0.088	0.005*	0.033*
	(0.001)	(0.095)	(0.003)	(0.017)
Controls	0.003	0.159*	0.004	0.028
	(0.002)	(0.095)	(0.003)	(0.043)
Controls + Region FE	0.003**	0.159***	0.004**	0.028
	(0.001)	(0.053)	(0.002)	(0.034)
Observations	37452	40876	24227	1583

Note: *** p<0.01, ** p<0.05, * p<0.10.

Unlike previous guides, this one is planned to be regularly updated. The aim of this guide is to not explain custom user-written commands like `esttab` or `tabout` in detail but provide a replicable set of templates that can be easily and quickly adapted. Furthermore, this guide is not intended to be an introduction to regression analysis in Stata or a tutorial on LaTeX. Instead, it is aimed at users who are already familiar with both and would prefer not to spend hours searching bits and pieces of code online.

The LaTeX part is provided in a shared Overleaf document that can be viewed online. Users can either duplicate the template, or download the

source code and locally compile the files. The document also contains a change log which tracks updates to this guide.

In order to follow the guide, the Stata part is discussed here for each table and the LaTeX code is provided in the Overleaf document. Tables generate in Stata are referenced with the same table number in the Overleaf. So the Table exported as table1.tex will be Table *section*.1 in Overleaf. In order to preserve table numbering in LaTeX , I might sometimes call the same table within the same table command as follows:

```
\begin{table}

%% table code 1

 \begin{tabular}{lrrrrc}
       \input{table1.tex}
  \end{tabular}

  \vspace{2cm}

%% table code 2

  \estwide{table1.tex}

\end{table}
```

This is unusual and not optimal but it avoids creating a new table numbers.

Background: This guide was initially written in 2015 and was on my personal website (no longer there). The essence is the same but I have changed the dataset from the classic Stata auto.dta to more recent COVID-19 data. This data is an unbalanced panel and not clean which makes it a bit more realistic to play around with.

Two additional points. First, the tables that you generate might show different numbers if you have a more latest dataset than the one used in this guide. Additionally, I might also update screenshots to reflect changes in the tables. So numbers across screenshots might not match all the time. This should not be an issue as long as the code is working fine. Second, this guide also shows several regression results. Most of the regressions are for illustrative purposes only and should be interpreted with caution.

Preamble

Stata

Like other guides, a basic knowledge of Stata is assumed. This guide deals with advanced usage of locals, loops, and code structures that require some experience and familiarity with Stata programming.

- Install the `estout` package:

```
ssc install estout, replace

which estout  // check versions
which esttab
```

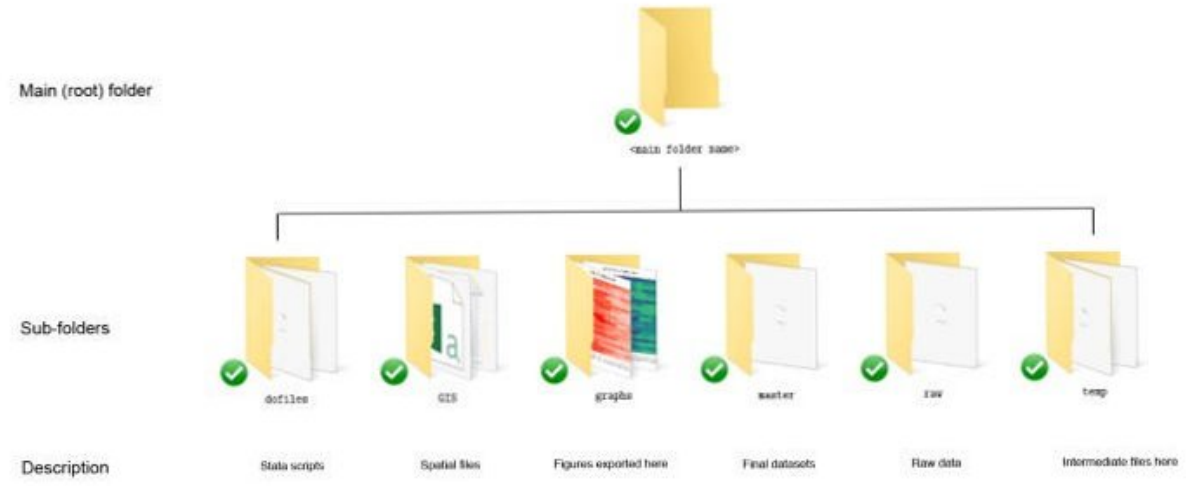
The versions should show the following displays:

```
*! version 3.23  31may2019  Ben Jann

*! version 2.0.9  06feb2016  Ben Jann
*! wrapper for estout
```

For workflow management, I use the following folder structure to

organize the files, and paths refer to subfolders relative to the root folder:



Paths are specified in this guide since all TeX files are exported in Stata and imported in Overleaf. Please modify these accordingly.

LaTeX

For LaTeX, either you should have an online account on services like Overleaf or a local deployment of a LaTeX compiler (MikTeX, TeXLive etc). Personally, I have moved all my stuff to Overleaf since I am working across multiple computers so I don't have to deal with updating packages across the board. Plus, at my workplace, I also don't have the permissions to install open-source software or packages for security reasons so that headache is also avoided. Additionally Overleaf, like Dropbox, keeps everything in the cloud, so it can be access anywhere in any browser and it also allows for easy collaborations. Sometimes, I also get the monthly subscription if track changes are needed for specific projects or faster compilation times are required. The free version is more than sufficient for simple documents.

The LaTeX preamble including the required packages are given on this read-only Overleaf document:

Stata-to-Latex guide on Overleaf

www.overleaf.com/read/fgrpmwxbtmvr

The document is complete and compiles as a standard article class file. It can also be easily modified to comply with other class files like `elsarticle` etc. For beamer presentations, I will do a separate guide.

For Stata to LaTeX examples, it is good to bookmark Ben Jann's own page who also wrote this brilliant package:

estout

The estout package provides tools for making regression tables in Stata. The package currently contains the following...

repec.sowi.unibe.ch

The help guide for the `estout` is also fairly extensive and worth exploring carefully:

estout

estout assembles a regression table from one or more models previously fitted and stored. The full syntax of estout is...

repec.org

Statalist has a lot of useful posts on `estout` so one can always search there as well.

Here I would also like to acknowledge the work done by Jörg Weber who wrote a bunch of LaTeX wrappers for `estout` back in 2012 and I have used his code since:

Automated Table generation in Stata and integration into LaTeX (1)

I use estout to generate tables of summary statistics and regression results that can be easily imported into LaTeX...

www.jwe.cc

He introduced two custom LaTeX commands: `estauto` and `estwide` that minimize the effort required to do extra work with the Stata tex output. This includes adding the tabular preamble and adjusting column widths.

I also utilize the `adjustbox` package a lot in LaTeX to resize tables. This has worked for me so far but it might not be the best way to fit tables since each resize basically gives a different font size (unless all the tables that are resized are of equal width). Manual customization of tables is possible but I won't touch it in this guide.

Let's get started

Anyone writing a paper eventually has to deal with generating tables in Stata and formatting them in LaTeX. While standard statistics and regressions are easy to handle, as soon as customized outputs are required, it quickly becomes tricky.

Making tables by hand is really not advisable, since outputs change as data files are updated and/or cleaned. In this post, we will look at various examples of generating summary statistic and regression tables. The main aim is to create a LaTeX-ready output that can be directly compiled using the Ben Jann's `estout` package.

The guide is split into two parts. The first part covers tables and the second part covers regressions. Here outputs are shown using a minimum working example in Stata using the COVID-19 dataset from

Our World in Data (OWID). The tables and regressions are for illustration purposes only and are not to be interpreted as actual analysis. We will deal with the analysis part in future guides.

Let's get the data. In spirit of the previous guides, we will stick to the COVID-19 repository from OWID:

```
************************
***  COVID 19 data   ***
************************

// once the file is downloaded, this code can be marked out to
avoid
// updating it every time the dofile is executed.

insheet using "https://covid.ourworldindata.org/data/owid-covid-
data.csv", clear
save ./raw/full_data_raw.dta, replace

gen date2 = date(date, "YMD")
format date2 %tdDD-Mon-yy
drop date
ren date2 date

ren location country
replace country = "Slovak Republic" if country == "Slovakia"

drop if date < 21915

save "./master/OWID_data.dta", replace


*********************************
***  Country classifications   ***
*********************************

*copy "https://github.com/asjadnaqvi/COVID19-Stata-Tutorials
/blob/master/master/country_codes.dta?raw=true" "./master
/country_codes.dta", replace

*********************************
**** put the files together
*********************************

use "./master/OWID_data.dta", clear
merge m:1 country using "./master/country_codes.dta"
```

```
drop if _m!=3
drop _m
```

The dataset is also merged with World Bank 2020 country classifications to allow us to generate tables by different country groups. The above code downloads a cleaned Stata file from my GitHub. This file was used in previous guides as well.

Next step, for now, replace all negative values to missing, and replace 0 vaccinations to missing as well:

```
foreach x of varlist new_cases new_deaths new_tests
new_vaccinations {
 replace `x' = . if `x'<0
}

recode new_vaccinations (0=.)
```

Then, based on the World Bank classifications, define seven regions of the World:

```
gen region = .
 replace region = 1 if  group6==1  // East Asia and Pacific (EAP)
 replace region = 2 if  group8==1  // Europe and Central Asia
(ECA)
 replace region = 3 if group20==1  // Latin America and Caribbean
(LAC)
 replace region = 4 if group26==1  // Middle East & North Africa
(MENA)
 replace region = 5 if group37==1  // Sub-saharan Africa (SSA)
 replace region = 6 if group35==1  // South Asia (SAsia)
 replace region = 7 if group29==1  // North America (NAmerica)

lab de region 1 "EAP" 2 "ECA" 3 "LAC" 4 "MENA" 5 "SSA" 6 "S. Asia"
7 "N. America"
lab val region region
```

Clean up the the variables and declare the data as a panel dataset:

```
drop group*

lab var new_cases         "Daily cases"
lab var new_deaths        "Daily deaths"
lab var new_tests         "Daily tests"
lab var new_vaccinations "Daily vaccinations"

encode country, gen(id)
order id date
xtset id date
```

Tables

In this section we will cover tables. Tables are broadly referred to as summary statistics that one can generate from `summarize`, `tabstat`, `ttest`, `corr` etc. or commands that are not regressions. These are handled differently in the `estout` package.

Table 1: Basics

Let's start with a classic summary statistic table:

```
tabstat new_cases new_deaths new_tests new_vaccinations, c(stat)
stat(mean sd min max n)

ereturn list
```

which gives us:

```
. tabstat new_cases new_deaths new_tests new_vaccinations, c(stat) stat(mean sd min max n)

    variable |      mean        sd       min       max         N
-------------+--------------------------------------------------
   new_cases |  1735.487  9437.458         0    299786     62278
  new_deaths |  44.12956  187.8334         0      4401     54822
   new_tests |  37979.42  147649.1         1   2945871     30782
new_vaccin~s |  75446.62  213167.5         1   2242472      1664
-------------------------------------------------------------------

. ereturn list

scalars:
                  e(N) =  62647

macros:
               e(cmd) : "estpost"
            e(subcmd) : "tabstat"
             e(stats) : "sum mean sd min max count"
              e(vars) : "new_cases new_deaths new_tests new_vaccinations"

matrices:
               e(sum) :  1 x 4
              e(mean) :  1 x 4
                e(sd) :  1 x 4
               e(min) :  1 x 4
               e(max) :  1 x 4
             e(count) :  1 x 4
```

Note that numbers will vary based on when the data is pulled. All output is saved in the e-class local. All variables that are named in the `e()` matrices are essentially what can be picked in the `estout` package. This important to know since this is what we want to play around with. For example, for the very latest and more advanced Stata commands, sometimes one needs to covert r-class to e-class locals to push it to `estout`. Future updates of this guide will cover this aspect.

We can store this information in the `estout` local as follows:

```
est clear  // clear the est locals


estpost tabstat new_cases new_deaths new_tests new_vaccinations,
c(stat) stat(sum mean sd min max n)


esttab, ///
 cells("sum(fmt(%13.0fc)) mean(fmt(%13.2fc)) sd(fmt(%13.2fc)) min
```

```
  max count") nonumber ///
    nomtitle nonote noobs label collabels("Sum" "Mean" "SD" "Min"
  "Max" "N")
```

which is displayed on screen as:

```
. estpost tabstat new_cases new_deaths new_tests new_vaccinations, c(stat) stat(sum mean sd min max n)

Summary statistics: sum mean sd min max count
     for variables: new_cases new_deaths new_tests new_vaccinations

             |    e(sum)    e(mean)      e(sd)     e(min)     e(max)   e(count)
-------------+--------------------------------------------------------------------
   new_cases |  1.08e+08   1735.487   9437.458          0     299786      62278
  new_deaths |   2419271   44.12956   187.8334          0       4401      54822
   new_tests |  1.17e+09   37979.42   147649.1          1    2945871      30782
new_vaccin~s |  1.26e+08   75446.62   213167.5          1    2242472       1664

.
.
. esttab, ///
>         cells("sum(fmt(%13.0fc)) mean(fmt(%15.2fc)) sd(fmt(%15.2fc)) min max count") nostar unstack nonumber ///
>                 nomtitle nonote noobs label collabels("Sum" "Mean" "SD" "Min" "Max" "N")

                          Sum            Mean            SD             Min             Max               N
Daily cases       108,082,631        1,735.49        9,437.46               0         299,786          62,278
New deaths          2,419,271           44.13          187.83               0           4,401          54,822
New tests       1,169,082,516       37,979.42      147,649.14               1       2,945,871          30,782
New vaccinations  125,543,181       75,446.62      213,167.53               1       2,242,472           1,664
```

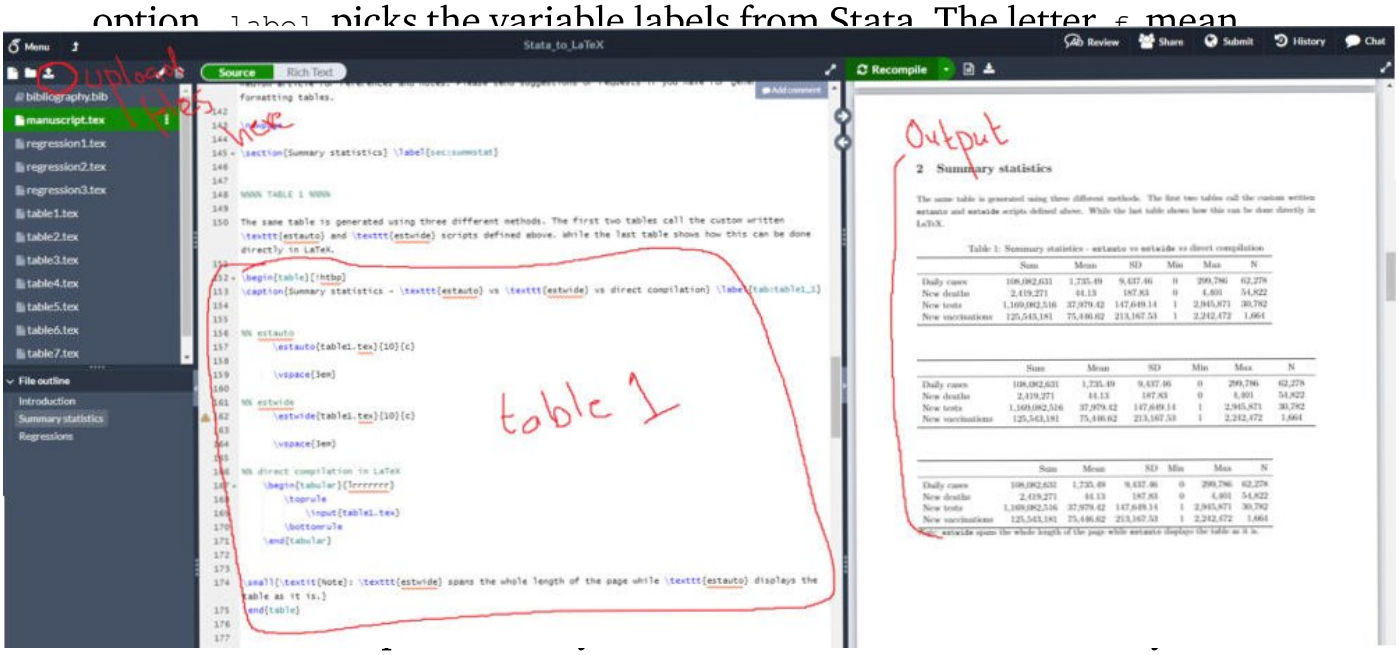
The first code shows how `estpost` stores the information, which is then processed by `esttab`, a wrapper for `estpost` , and displays the table as it should look like in LaTeX. Here quite a lot of options are specified which can be looked up in the `help esttab` page. The important ones are `cells` which picks up the names from the e-class variables and their formatting which is defined in the brackets. There are other ways to format the tables as well and these will be discussed below. What is very important here is that the very first formatting that is defined, `%13.0fc` in this case, is passed on to the remaining variables. Therefore if the formatting of other variables needs to be changed, they have to be manually adjusted. If you are unfamiliar with formatting and a good look at `help format` is highly recommended. Here we are saying that the number should be 13 characters long and show the thousands separator with commas. The number 13 is not important as long as it is more than the length of the highest digit. The separator can also be swapped with a decimal as is the

norm in some European countries. But since mean and standard deviation are averages, they also have decimal points. So we custom format them with `%13.2fc` to show two decimal places. The remaining variables utilize the formatting defined in the first variable.

Once we get the table as we need it, we can export it as a LaTex table with additional options:

```
esttab using "./graphs/guide80/table1.tex", replace ////
 cells("sum(fmt(%13.0fc)) mean(fmt(%13.2fc)) sd(fmt(%13.2fc)) min
max count") nonumber ///
  nomtitle nonote noobs label booktabs f ///
  collabels("Sum" "Mean" "SD" "Min" "Max" "N")
```

The key options are highlighted. The file is exported as table1.tex in the `booktabs` format. Booktabs is a LaTeX package that gives a neat-looking tables. We manually replace the column headers with the `collabels` option. `label` picks the variable labels from Stata. The letter `f` mean



clicking the small icon shown below. The LaTeX syntax is shown in the right window while the compiled output is shown in the left pane. The LaTeX code is annotated in the Overleaf file, so the LaTeX part is not discussed here.

Tables 2 and 3: `mean/aux` versus cells

We can also export summary statistics by groups. For example, we can calculate the mean and the standard deviation (SD) by country groups:

```
est clear
estpost tabstat new_cases new_deaths new_tests new_vaccinations,
by(region) c(stat) stat(mean sd) nototal
```

Also check what is stored in the e-class:

```
ereturn list
```

This table can be exported through two ways. We can use the simpler `mean` and `aux` option of `esttab`:

```
*** display on screen:

esttab, main(mean) aux(sd) nostar nonumber unstack ///
   nonote noobs label ///
   collabels(none) ///
   eqlabels("EAP" "ECA" "LAC" "MENA" "SSA" "S. Asia" "N. America")
///
   nomtitles
```

which we can export to LaTeX as follows:

```
esttab using "./graphs/guide80/table2.tex", replace ///
  main(mean %15.2f) aux(sd %15.2f) nostar nonumber unstack ///
   compress nonote noobs gap label booktabs f   ///
   collabels(none) ///
   eqlabels("EAP" "ECA" "LAC" "MENA" "SSA" "S. Asia" "N. America")
///
   nomtitles
```

or the `cells` option, which is exported as follows:

```
*** export to LaTeX:

esttab using "./graphs/guide80/table3.tex", replace ////
 cells("mean(fmt(%15.2fc))" "sd(par fmt(%15.2fc))") nostar unstack
nonumber ///
  compress nonote noobs gap label booktabs f  ///
   collabels(none) ///
   eqlabels("EAP" "ECA" "LAC" "MENA" "SSA" "S. Asia" "N. America")
///
   nomtitles
```

While most of the syntax elements are common across the two tables, I would like to highlight the differences first. These are the second lines of the two codes above. Here we can see that with the `mean/aux` format we pick elements from the e-class table namely "mean" and "sd". This also the same in the `cells` option. The main different is how the formatting is defined. In the first case, the format is just given in front of the variable, while in the cells option it is needs to be defined using the `fmt` option.

Now for the common elements. Note that here we are defining the column headers using `eqlabels` and not `collabels` . This is because we are taking a stacked output (all the `by` variables are shown in one long column on top of each other), and unstacking it using the `unstack` option (all the `by` variables are different columns). Hence each column is treated as an "equation". This might be a bit confusing but if you are stuck not knowing which one to you, try either one of them.

Table 4: Formatting cells

As briefly discussed above, we can export summary statistic tables using the more generic `cells` option. This also allows for more flexibility in customizing the output. Let's try some options. We again start with a simple table:

```
est clear
estpost tabstat new_cases new_deaths new_tests new_vaccinations,
by(region) c(stat) stat(mean sd) nototal
```

and we try a different format option:

```
*** display locally
```

```
esttab, cells(mean(fmt(2)) sd(par)) nostar  nonumber unstack ///
  nomtitle nonote noobs label  ///
   collabels(none) gap ///
   eqlabels("EAP" "ECA" "LAC" "MENA" "SSA" "S. Asia" "N. America")
///
   nomtitles
```

which gives us the same output as for Table 3:

This can be exported to LaTeX as follows:

```
esttab using "./graphs/guide80/table4.tex", replace ////
 cells(mean(fmt(2)) sd(par)) nostar unstack nonumber ///
  compress nonote noobs gap label booktabs f ///
   collabels(none) ///
   eqlabels("EAP" "ECA" "LAC" "MENA" "SSA" "S. Asia" "N. America")
///
   nomtitles
```

Here we use another formatting option `fmt(2)` which is short for two decimal places. This formatting is also passed on the SD. For SD, we also specify `par` in brackets which stands for parenthesis. This gives us the curly brackets in the display.

Table 5: Custom brackets

The curly brackets can also be modified. For example, the code below replace curly brackets with square brackets:

```
esttab, cells(mean sd(par([ ]))) nostar  nonumber unstack ///
  nomtitle nonote noobs label  ///
   collabels(none) ///
   eqlabels("EAP" "ECA" "LAC" "MENA" "SSA" "S. Asia" "N. America")
///
   nomtitles
```

Effectively, any options specified in `par( )` will replace the opening and closing brackets. Here one can for example also use `par(< >)`.

Tables 6 and 7: Custom decimal places

Table 5 assigns 2 decimal points to the mean and 3 decimal points to the standard deviation (SD):

```
esttab using "./graphs/guide80/table5.tex", replace ////
 cells(mean(fmt(2)) sd(fmt(3) par)) nostar unstack nonumber ///
  compress nonote noobs gap label booktabs f alignment(D{.}
{.}{-1})  ///
   collabels(none) ///
   eqlabels("EAP" "ECA" "LAC" "MENA" "SSA" "S. Asia" "N. America")
///
   nomtitles
```

Table 6 fully customizes the decimal points. For example here the mean has increasing number of decimal places while the SD has decreasing number of decimal places as one does down the rows:

```
esttab using "./graphs/guide80/table6.tex", replace ////
 cells(mean(fmt(1 2 3 4)) sd(fmt(3 2 1 0) par)) nostar unstack
nonumber ///
  compress nonote noobs gap label booktabs f alignment(D{.}
```

```
{.}{-1})  ///
   collabels(none) ///
   eqlabels("EAP" "ECA" "LAC" "MENA" "SSA" "S. Asia" "N. America")
///
   nomtitles
```

This allows one to customize tables where the units across variables are very different. For example, some variables might be given in millions while some might be in fractions.

The number of decimals to display in tables is all context dependent but in general it is good to scale variables so they are roughly similar, otherwise you don't want to see rows and rows of significance levels that look like `0.000***`. This would annoy any referee or a reader.

Table 8: A fully formatted table with grouped variables

Here we generate a table where variables in the rows are grouped. We can also incorporate LaTeX elements in the labels:

```
replace human_development_index = human_development_index * 100

lab var total_cases    "\hspace{0.25cm} Total cases (units)"
lab var total_deaths   "\hspace{0.25cm} Total deaths (units)"
lab var total_tests    "\hspace{0.25cm} Total tests (units)"
lab var total_vaccinations  "\hspace{0.25cm} Total vaccinations
(units)"

lab var median_age        "\hspace{0.25cm} Median age (years)"
lab var aged_65_older     "\hspace{0.25cm} Age 65+ (years)"
lab var life_expectancy   "\hspace{0.25cm} Life expectancy
(years)"
lab var extreme_poverty   "\hspace{0.25cm} Extreme poverty (\%)"
lab var human_development_index "\hspace{0.25cm} HDI (Index)"
lab var gdp_per_capita    "\hspace{0.25cm} GDP per capita (USD)"
lab var population_density  "\hspace{0.25cm} Population density
(per sq km)"
```

Here we add a space of 0.25 cm before the description of the variable label.

Next we generate the summary statistic table:

```
estpost tabstat ///
 total_cases total_deaths total_tests total_vaccinations ///
 median_age aged_65_older  life_expectancy extreme_poverty
human_development_index gdp_per_capita population_density, ///
  c(stat) stat(mean sd min max n)
```

And export it as follows:

```
esttab using "./graphs/guide80/table8.tex", replace ////
refcat(total_cases "\emph{COVID-19 indicators}" median_age
"\vspace{0.1em} \\ \emph{Socio-economics indicators}", nolabel)
///
 cells("mean(fmt(%15.0fc %15.0fc %15.0fc %15.0fc  2)) sd min max
count(fmt(0))") nostar unstack nonumber ///
  compress nomtitle nonote noobs gap label booktabs f ///
  collabels("Mean" "SD" "Min" "Max" "N")
```

We can use the `refcat` option that picks variables as reference points, and adds a name to each group of variables. From the code above, we get this LaTeX table:

Note here how the use of `\hspace` in `var label` gives the need formatting above where we define the groups by using the `refcat` option.

Table 9: T-tests

For t-tests, we define two groups:

```
gen dummy = .

replace dummy = 0 if region==4  // MENA
replace dummy = 1 if region==5  // SSA
```

where we want to compare the Middle East and North Africa (MENA) region with Sub-Saharan Africa (SSA). Let's a basic output:

```
ttest life_expectancy, by(dummy) unequal
```

where we compare the life expectancy and assume that the groups have different variances which needs to he adjusted for. We get this output:

Here we see a bunch of info including observations, mean and standard errors across the two groups plus the difference, and the probabilities which show the results for the null hypothesis that the difference = 0 `Ho: diff = 0`. The bottom right output shows that the difference is clearly greater than zero and highly significant (also unsurprising given that we see the means are far apart already).

For these two groups we can perform a t-test across a bunch of variables which we can store in a local:

```
global controls aged_65_older gdp_per_capita life_expectancy
extreme_poverty population_density
```

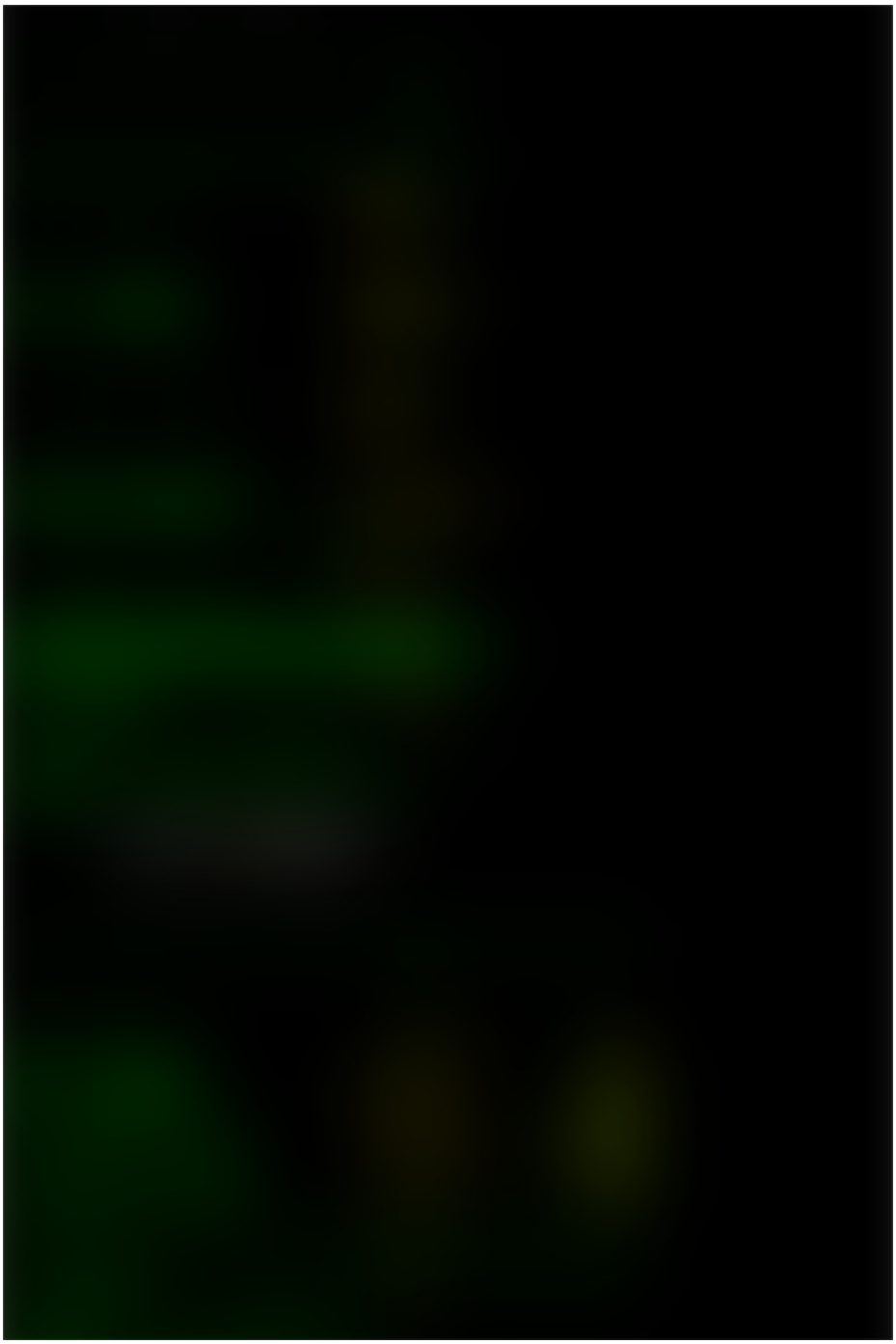
Note that t-test itself does not allow several variables to be defined by `estout` allows it. So for the above groups we use the following command:

```
est clear
 estpost ttest $controls  ///
    , by(dummy) unequal
```

We can display the output in Stata window using either of the two commands:

```
esttab, label
esttab, wide label
```

where the first option stacks the mean and S.E. of the difference while the second option shows the S.E. in the same row as the mean.



We can also export this table to LaTeX as follows:

```
esttab using "./graphs/guide80/table9.tex", replace ///
  cells("mu_1(fmt(3)) mu_2  b(star) se(par) count(fmt(0))") ///
  collabels("MENA" "SSA" "Diff. (MENA - SSA)" "S.E." "Obs." ) ///
  star(* 0.10 ** 0.05 *** 0.01) ///
  label booktabs nonum gaps f noobs compress
```

which gives this simple table:

Note the use of commands in cells to recover the coefficients, star and parenthesis. As mentioned earlier, anything variable named in the `ereturn list` can be used here and the table can be customized. The table is formatted to show values up to 3 decimal places except for the observations.

Such a table can be used to show baseline covariates for treatment and control groups where the difference should be significantly zero.

Table 10: Summary statistics by different control groups

Sometimes we want to generate summary statistic tables by different control groups. Note that there are not “by” groups which are mutually exclusive, but other controls that can have overlaps in categories. For example within groups like the World, Europe, European Union, Euro Area, High income countries there are high overlaps but there are other countries in there as well.

In order to summary these categories we need to move away from `tabstat`. Instead here we need to use a combination of `eststo` or estimates store, and the `summarize` or `summ` command. Other commands can also be used but for now we stick to this basic, yet highly relevant

example.

We operationalize these command as follows:

```
est clear

eststo grp1: estpost summ new_cases new_deaths new_tests

eststo grp2: estpost summ new_cases new_deaths new_tests if
region==2  // Europe and Central Asia

eststo grp3: estpost summ new_cases new_deaths new_tests if
continent=="Europe"

eststo grp4: estpost summ new_cases new_deaths new_tests if
human_development_index>80
```

where the summary statistics are stored in the `grp*` estimates. Here `eststo` basically creates a copy of the estimates. One can collect as many as one want and call them later as well.

Next step, we can can generate a mock table on screen by calling the stored estimates:

```
esttab grp*, main(mean %6.2f) aux(sd) mtitle("All" "ECA" "Europe"
"High HDI")
```

And finally, we can export to LaTeX easily as follows:

```
esttab grp* using "./graphs/guide80/table10.tex", replace ///
  main(mean %15.2fc) aux(sd %15.2fc) nostar nonumber unstack ///
   compress nonote noobs gap label booktabs f   ///
   collabels(none) mtitle("All" "ECA" "Europe" "High HDI")
```

which gives us:

Regressions

For regression outputs countless examples exist online for the `estout` package. Here I go over some regression table templates including some slightly more advanced ones. These deal with non-standard aspects like lagged terms, interactions, and generating tables from several combinations of regressions.

First let's clean up the data:

```
replace new_deaths_per_million = . if new_deaths_per_million < 0
replace new_cases_per_million  = . if new_cases_per_million < 0
replace new_tests_per_thousand = . if new_tests_per_thousand < 0
replace new_tests_per_thousand = . if new_tests_per_thousand > 100

ren new_deaths_per_million         deaths_norm
ren new_cases_per_million          cases_norm
ren new_tests_per_thousand         tests_norm
ren people_vaccinated_per_hundred  vaccine_norm

replace gdp_per_capita = gdp_per_capita / 100000
replace population_density = . if population_density> 4000 //
Monaco is an outlier
```

And label everything neatly:

```
. esttab

------------------------------------------------------------
                      (1)             (2)
                deaths_norm     deaths_norm
------------------------------------------------------------
cases_norm        0.0112***       0.0114***
                  (15.64)         (14.03)

tests_norm        0.0796**         0.105***
                  (2.69)          (3.63)

stringency~x      0.0244          0.0242
                  (1.51)          (1.53)

aged_65_ol~r                       0.124**
                                   (2.69)

gdp_per_ca~a                      -2.531***
                                  (-4.60)

life_expec~y                      -0.00460
                                  (-0.12)

extreme_po~y                      -0.00117
                                  (-0.27)

population~y                     -0.000849*
                                  (-2.09)

_cons             -1.046          -1.208
                  (-1.28)         (-0.58)
------------------------------------------------------------
N                   29181           22609
------------------------------------------------------------
t statistics in parentheses
* p<0.05, ** p<0.01, *** p<0.001
```

```
. esttab, label

------------------------------------------------------------
                      (1)             (2)
                 Death..)        Death..)
------------------------------------------------------------
Cases (norm.)     0.0112***       0.0114***
                  (15.64)         (14.03)

Tests (norm.)     0.0796**         0.105***
                  (2.69)          (3.63)

Stringency Index  0.0244          0.0242
                  (1.51)          (1.53)

Age 65+ (years)                    0.124**
                                   (2.69)

GDP per capita                    -2.531***
                                  (-4.60)

Life expectancy (y~)              -0.00460
                                  (-0.12)

Extreme poverty (\%)              -0.00117
                                  (-0.27)

Pop. density (pop/~)             -0.000849*
                                  (-2.09)

Constant          -1.046          -1.208
                  (-1.28)         (-0.58)
------------------------------------------------------------
Observations        29181           22609
------------------------------------------------------------
t statistics in parentheses
* p<0.05, ** p<0.01, *** p<0.001
```

Let's fix the decimal places and get rid of the column headers since this is redundant information that usually goes in the table caption:

```
esttab, b(3) se(3) nomtitle label
```

which gives us:

The next step is extremely important (and was a major cause of stress until this was figured out): `esttab` by default shows significance levels at the 5%, 1% and 0.1% level, while we are trained to assume that ***, **, and * represent 10%, 5%, and 1%. This can be corrected as follows:

```
esttab, b(3) se(3) nomtitle label star(* 0.10 ** 0.05 *** 0.01)
```

which gives us this table. Compare this to the one above and notice the differences. Minor in this version but in some cases can make quite a bit of a difference:

Now the next step is to export this table in LaTeX format. There are two ways of doing this. Either we export it ready for LaTeX directly with everything formatted in Stata, or we just export the table body and plug-in the table premable and formatting in LaTeX.

For the first option, this is done as follows:

```
esttab using "./graphs/guide80/regression1_1.tex", replace  ///
 b(3) se(3) nomtitle label star(* 0.10 ** 0.05 *** 0.01) ///
 booktabs alignment(D{.}{.}{-1}) ///
 title(My very first basic regression table \label{reg1})   ///
 addnotes("Dependent variable: Deaths (norm.). A lot of
endogeneity in this specification." "Data source: Our World in
```

```
Data COVID-19 database.")
```

The first two lines are standard code we used earlier. Third line onwards is LaTeX formatting. Here we use the `booktabs` package which neatly formats a table. The `alignment(D{.}{.}{-1})` option is used by the `dcolumn` package to align the decimals. The last two lines add the title and footnotes. Note that in the `addnote` option, one can start a new line by using a new set of quotation marks `addnote("Line 1" "Line 2")`.

All this requires in LaTeX is the following code:

```
\input{regression1_1.tex}
```

The second option where we just export the core table, for example to use with the custom `estauto` or `estwide` wrappers is done as follows:

```
esttab using "./graphs/guide80/regression1_2.tex", replace f  ///
 b(3) se(3) nomtitle label star(* 0.10 ** 0.05 *** 0.01) ///
 booktabs alignment(D{.}{.}{-1})
```

Note here that everything is exactly the same except that the title and notes options is gone, and the `f` has been added. This command, which stands for *fragment* essentially gets rid of the table preamble.

Please see Overleaf for the implementation of the two tables. Here I would also advice to look at the table output file which can be viewed in a local LaTeX editor, if you have one installed, otherwise any text reading software like Notepad or Notepad++ (highly recommended) in Windows can be used as well. Here is what the TeX output of the two files looks like:

regression1_1.tex

```
\begin{table}[htbp]\centering
\def\sym#1{\ifmmode^{#1}\else\(^{#1}\)\fi}
\caption{My very first basic regression table \label{reg1}}
\begin{tabular}{l*{2}{D{.}{.}{-1}}}
\toprule
                    &\multicolumn{1}{c}{(1)}         &\multicolumn{1}{c}{(2)}\\
\midrule
Cases (norm.)       &       0.011\sym{***}&       0.011\sym{***}\\
                    &     (0.001)         &     (0.001)         \\
\addlinespace
Tests (norm.)       &       0.080\sym{***}&       0.105\sym{***}\\
                    &     (0.030)         &     (0.029)         \\
\addlinespace
Stringency Index    &       0.024         &       0.024         \\
                    &     (0.016)         &     (0.016)         \\
\addlinespace
Age 65+ (years)     &                     &       0.124\sym{***}\\
                    &                     &     (0.046)         \\
\addlinespace
GDP per capita      &                     &      -2.531\sym{***}\\
                    &                     &     (0.550)         \\
\addlinespace
Life expectancy (years)&                   &      -0.005         \\
                    &                     &     (0.040)         \\
\addlinespace
Extreme poverty (\%)&                     &      -0.001         \\
                    &                     &     (0.004)         \\
\addlinespace
Pop. density (pop/km$^2$)&                  &      -0.001\sym{**} \\
                    &                     &     (0.000)         \\
```

regression1_2.tex

```
                    &\multicolumn{1}{c}{(1)}         &\multicolumn{1}{c}{(2)}\\
\midrule
Cases (norm.)       &       0.011\sym{***}&       0.011\sym{***}\\
                    &     (0.001)         &     (0.001)         \\
\addlinespace
Tests (norm.)       &       0.080\sym{***}&       0.105\sym{***}\\
                    &     (0.030)         &     (0.029)         \\
\addlinespace
Stringency Index    &       0.024         &       0.024         \\
                    &     (0.016)         &     (0.016)         \\
\addlinespace
Age 65+ (years)     &                     &       0.124\sym{***}\\
                    &                     &     (0.046)         \\
\addlinespace
GDP per capita      &                     &      -2.531\sym{***}\\
                    &                     &     (0.550)         \\
\addlinespace
Life expectancy (years)&                   &      -0.005         \\
                    &                     &     (0.040)         \\
\addlinespace
Extreme poverty (\%)&                     &      -0.001         \\
                    &                     &     (0.004)         \\
\addlinespace
Pop. density (pop/km$^2$)&                  &      -0.001\sym{**} \\
                    &                     &     (0.000)         \\
\addlinespace
Constant            &      -1.046         &      -1.208         \\
                    &     (0.815)         &     (2.066)         \\
\midrule
Observations        &       29181         &       22609         \\
```

The screenshots do now show the complete output

Notice the preamble before `tabular` missing in the second file. The table generated from the first option looks like this in LaTeX:

Table 3.1: My very first basic regression table

	(1)	(2)
Cases (norm.)	0.011***	0.011***
	(0.001)	(0.001)
Tests (norm.)	0.080***	0.105***
	(0.030)	(0.029)
Stringency Index	0.024	0.024
	(0.016)	(0.016)
Age 65+ (years)		0.124***
		(0.046)
GDP per capita		−2.531***
		(0.550)
Life expectancy (years)		−0.005
		(0.040)
Extreme poverty (%)		−0.001
		(0.004)
Pop. density (pop/km^2)		−0.001**
		(0.000)
Constant	−1.046	−1.208
	(0.815)	(2.066)
Observations	29181	22609

Standard errors in parentheses
Dependent variable: Deaths (norm.). A lot of endogeneity in this specification.
Data source: Our World in Data COVID-19 database.
* $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

Regression 2: Lags, advanced variable labels, and column

descriptors

Here we utilize the following (randomly defined) specification:

```
deaths_t = b0 + b1 cases_t-10 + b2 tests_t-10 + b3 stringency_t-10 +
errors
```

This regression basically we are saying that deaths now are predicted by cases, tests, and policy stringency from 10 days ago. Since the data is a panel, we can define a bunch of different random effect, fixed effect, time effect regressions (or whatever other specification on wants to use):

```
est clear

eststo: xtreg deaths_norm L(10).cases_norm L(10).tests_norm
L(10).stringency_index  , re robust
 estadd local  FE  "No"
 estadd local  TE  "No"

eststo: xtreg deaths_norm L(10).cases_norm L(10).tests_norm
L(10).stringency_index i.date, re robust
 estadd local  FE  "No"
 estadd local  TE "Yes"

eststo: xtreg deaths_norm L(10).cases_norm L(10).tests_norm
L(10).stringency_index  , fe robust
 estadd local  FE  "Yes"
 estadd local  TE  "No"

eststo: xtreg deaths_norm L(10).cases_norm L(10).tests_norm
L(10).stringency_index i.date, fe robust
 estadd local  FE  "Yes"
 estadd local  TE  "Yes"

eststo: xtreg deaths_norm L(10).cases_norm L(10).tests_norm
L(10).stringency_index  , vce(cluster region)
 estadd local RFE  "Yes"
 estadd local  TE  "No"

eststo: xtreg deaths_norm L(10).cases_norm L(10).tests_norm
L(10).stringency_index i.date, vce(cluster region)
 estadd local RFE  "Yes"
 estadd local  TE  "Yes"
```

For each regression we can also specify time fixed effects to capture daily variations. This information is stored in the local `TE` . Similarly fixed effects are stored in the local `FE` which basically says that countries are independent. The last two regressions cluster at the region variable level, which makes more sense that country FE since internal regional mobility might be contributing to deaths as well, for example in Europe.

All of the above information is exported as follows:

```
esttab using "./graphs/guide80/regression1.tex", replace f  ///
 b(3) se(3) ///
 keep(L10.cases_norm L10.tests_norm L10.stringency_index) ///
 coeflabel(L10.cases_norm "Cases (norm.) (t-10)" L10.tests_norm
"Tests (norm.) (t-10)" L10.stringency_index "Stringency Index
(t-10)" ) ///
 star(* 0.10 ** 0.05 *** 0.01) ///
 label booktabs nomtitle collabels(none) compress alignment(D{.}
{.}{-1}) ///
 scalars( "N Obs." "rho \$\rho\$" "TE Time FE" "FE Country FE"
"RFE Region FE" ) sfmt(3)
```

where the key parts of the code are highlighted. Since it is a regression, coefficients can picked directly using the `b` and `se` options with a three decimal place format. We only keep the variables we want to show since some regressions also have time dummies which is a very very long output. Note here that the reason I used lagged variables is to show how to deal with them in `esttab` . Since lags do not exist as variables in Stata, they don't have labels (they can be generated though, and it is also advisable for other reasons like maximizing the observations available). So the `coeflabel` option is used here to fix them. Otherwise a simple regression like `reg y x1 x2 x3` would just require the `label` option as shown in the first example. The `stars` option can be used to modify the symbols as well, for example the three stars (***) can be replaced with a single symbol to optimize the space. The scalars formatted for display at

the end of the regression table using the `sfmt(3)` :

All the scalars that exist in a standard regression output can be called here directly, for example `rho` . Whatever is specified individually after each regression using the `estadd local` option can also be added here.

To see all the possible options here is what one can do: Run one regression:

```
est clear

eststo: xtreg deaths_norm L(10).cases_norm L(10).tests_norm
L(10).stringency_index  , re robust
 estadd local  FE  "No"
 estadd local  TE  "No"
```

and type `return list` . This will basically show that there is an r-class table stored as a matrix (a relatively new feature in Stata). If you type `mat li r(table)` you get:

This table is useful for picking variable names in case you are using lags or interaction terms and it is not clear how the names are stored.

If you also type `ereturn list` , then one gets a table of all the stored locals that can be called directly in the `scalars` option in `esttab` :

Since I am using `xtreg` , the `ereturn list` shows the output for panel regressions. Other types of regressions will have their own set of variables stored in the e-class locals.

Regression 3: Regressions over groups using loops

In this table two different regression specifications are used for different dependent variables.

Here, we can just use a global option to define all the controls without having to define very long regressions:

```
global controls aged_65_older gdp_per_capita life_expectancy
extreme_poverty population_density
```

Storing sets of variables in global or local macros is also highly advisable for neatness and avoiding mistakes.

We can loop over several variables:

```
est clear

foreach x of varlist deaths_norm cases_norm tests_norm
vaccine_norm {
 eststo: xtreg `x' stringency  $controls       , re robust
  estadd local RFE  "No"

 eststo: xtreg `x' stringency  $controls       , vce(cluster
region)
  estadd local RFE  "Yes"
 }
```

For each dependent variable, a RE model and a regional FE model is estimated. It can be formatted and exported as follows:

```
esttab using "./graphs/guide80/regression2.tex", replace f  ///
 b(3) se(3) star(* 0.10 ** 0.05 *** 0.01) ///
 label booktabs nonotes nomtitle collabels(none)  ///
 scalars("RFE Region FE") sfmt(3) ///
 mgroups("Deaths" "Cases" "Tests" "Vaccines", pattern(1 0 1 0 1 0
1 0) prefix(\multicolumn{@span}{c}{) suffix(}) span
erepeat(\cmidrule(lr){@span}))
```

The `mgroups` option clusters the regressions in groups. Since each dependent variable has two regressions, it follows a 1 0 pattern. If it were three regressions, it would have been 1 0 0 and so on. This LaTeX file is compiled as follows:

The column numbers and the table footer can also be customized to add more information.

Regression 4: Collecting the coefficient of interest from multiple regression combinations

Here we will generate a very specific table where one coefficient of interest is collected from a bunch of regressions. This can be useful for just displaying the coefficients of interest of one or more interventions for a bunch of tests and specifications. One comes across such tables in papers frequently which show the key coefficients without showing all possible regression outputs.

For example, a simple diff-in-diff is defined as follows:

```
y_t = b0 + b1*Post + b2*X + b3*Post*X + theta + errors
```

where T is the common time trend, X is the intervention, T*X is the

interaction term, and theta are covariates. So the coefficent we are interested in is b3 which has to significant, impactful and in the right direction (the dream outcome of many who are running causal inference designs).

Let's just define an intervention as a binary variable `policy` which is 1 if the stringency is greater than 70 and 0 otherwise (reminder that all of this is arbitrary):

```
gen    policy = stringency_index > 70  // strict lockdown
```

We can also use the Stata interaction term `##` to generate all possible combinations. In order to make the example a bit more complicated, the time variable `date` is interacted with the 10-period lag of the `policy` variable which in Stata language is `c.date##i.L(10).policy`. This is just to illustrate how to recover coefficient when variable names are not so straight forward.

We regress four dependent variables deaths, cases, tests, and vaccines which defines the columns, and four specifications using RE, FE, and both with controls, which form the four rows. So we want to get a 4x4 table of b3 coefficients.

Here we need to split the table generation in three parts. The top panel which has the header information but nothing below the coefficients, the middle panels which just have the coefficients and nothing else including the lines, and the bottom panel which has the last set of coefficients and the table footer.

Top panel

```
**** top panel

est clear

foreach x of varlist deaths_norm cases_norm tests_norm
vaccine_norm {
eststo: xtreg `x' c.date##i.L(10).policy, robust
}

esttab using "./graphs/guide80/regression3.tex", replace f  ///
 b(3) se(3) star(* 0.10 ** 0.05 *** 0.01)  ///
 keep(1L10.policy#c.date) coeflabel(1L10.policy#c.date "Baseline")
///
 label booktabs noobs nonotes  collabels(none) alignment(D{.}
{.}{-1}) ///
 mtitles("Deaths" "Cases" "Tests" "Vaccines")
```

Here we keep the interaction term only. Note that one can see the name from `mat li r(table)` option and we manually label it using the `coeflabel` option. The rest of the code is fairly straightforward.

Middle panels

The middle panels have all the options of labels and lines turned off:

```
**** center panel 1

est clear

foreach x of varlist deaths_norm cases_norm tests_norm
vaccine_norm {
eststo: xtreg `x' c.date##i.L(10).policy, vce(cluster region)
}

esttab using "./graphs/guide80/regression3.tex", append f  ///
 b(3) se(3) star(* 0.10 ** 0.05 *** 0.01)  ///
 keep(1L10.policy#c.date) coeflabel(1L10.policy#c.date "Region
FE") ///
 label booktabs nodep nonum nomtitles nolines noobs nonotes
collabels(none) alignment(D{.}{.}{-1})
```

```
**** center panel 2

est clear

foreach x of varlist deaths_norm cases_norm tests_norm
vaccine_norm {
eststo: xtreg `x' c.date##i.L(10).policy $controls,  robust
}

esttab using "./graphs/guide80/regression3.tex", append f  ///
 b(3) se(3) star(* 0.10 ** 0.05 *** 0.01)  ///
 keep(1L10.policy#c.date) coeflabel(1L10.policy#c.date "Controls")
///
 label booktabs nodep nonum nomtitles nolines noobs nonotes
collabels(none) alignment(D{.}{.}{-1})
```

Each panel is generated separately and is appended to the original file using the `apend` option.

Bottom panel

The bottom panel is the most complicated one since here we replace `coeflabel` with `varlabels` which is a programming option that can overwrite a lot of the internal `estout` code (so use it carefully).

```
est clear

foreach x of varlist deaths_norm cases_norm tests_norm
vaccine_norm {
eststo: xtreg `x' c.date##i.L(10).policy $controls, vce(cluster
region)
}

esttab using "./graphs/guide80/regression3.tex", append f  ///
 b(3) se(3) star(* 0.10 ** 0.05 *** 0.01)  ///
 keep(1L10.policy#c.date)  ///
 label booktabs collabels(none) nomtitles nolines nonum
alignment(D{.}{.}{-1})  ///
 varlabels(1L10.policy#c.date "Controls + Region FE",
elist(1L10.policy#c.date \bottomrule))
```

Here we turn off the lines using the `noline` option and manually specify a line in the `varlabels` using the `elist` and them `\bottomrule` option. Note that `bottomrule` is a `booktabs` option in LaTeX. For tables without `booktabs`, `bottomrule` can be replaced with `hline`.

And from the code above, we get the following 4x4 table:

This table can also be extended to have grouped columns, and scalars and locals in the footer.

Regression 5: Stack standard errors

In this table we take essentially stack standard errors on top of each other from different regressions where the mean value of the parameter is the same. For example random effects or clustered models, or models with spatial adjustments to S.E.s This is also parsimonious in terms of space and is neater in terms of presentation.

For this we just run a regression but in the `esttab` output modify the code to skip the bottom part of the table (as shown in the previous example), and more importantly, move the star from in front of the

mean value to the standard error:

```
*** top part

est clear
foreach x of varlist deaths_norm cases_norm tests_norm
vaccine_norm {
 eststo: xtreg `x' c.date##i.L(10).policy $controls, re
 }

esttab using "./graphs/guide80/regression5.tex", replace f  ///
 cells(b(fmt(3)) se(fmt(3) star par)) keep(1L10.policy#c.date)
coeflabel(1L10.policy#c.date "Policy $\times$ Post") ///
 label booktabs noobs nonotes collabels(none)  ///
 star(* 0.10 ** 0.05 *** 0.01)  ///
 mtitles("Deaths" "Cases" "Tests" "Vaccines")
```

We can now generate the second regression and note here that we just keep the S.E. add the start to it, put the value in square brackets, and append to the original file:

```
*** bottom part

est clear
foreach x of varlist deaths_norm cases_norm tests_norm
vaccine_norm {
 eststo: xtreg `x' c.date##i.L(10).policy $controls, vce(cluster
region)
 }

 esttab using "./graphs/guide80/regression5.tex", append f  ///
 cells(se(fmt(3) star par([ ]))) noobs keep(1L10.policy#c.date)
///
 varlabels(1L10.policy#c.date " ", elist(1L10.policy#c.date
\bottomrule))    ///
 star(* 0.10 ** 0.05 *** 0.01)  ///
 label booktabs nomtitle noline nonumber collabels(none)
scalars("N Obs.") sfmt(0)
```

And we get this table:

Note that here it is immediately easy to see which model gives what results. This type of table output was partially inspired by Melissa Dell's Mining Mita paper.

Regression 6: Rotating tables

Here is a common table request where you want to throw everything in the regression across multiple specifications and show the result in a table which can only fit in landscape of wide format.

So let's generate some controls and also generate monthly dummies:

```
global controls aged_65_older gdp_per_capita life_expectancy
extreme_poverty population_density

gen month = month(date)
```

Monthly dummies are just for the next step. Daily dummies were too computationally intensive.

Let's loop over a bunch of regressions for two specifications each: one without controls and one with controls:

```
est clear


foreach x of varlist deaths_norm cases_norm tests_norm
vaccine_norm {
 eststo: xtreg `x' L(0 4 8 12 16).stringency  i.month ,
vce(cluster region)
  estadd local CNTL  "No"

 eststo: xtreg `x' L(0 4 8 12 16).stringency $controls i.month ,
vce(cluster region)
  estadd local CNTL  "Yes"
 }
```

We can view the output in the Stata window as follows:

```
esttab, keep(stringency_index L* $controls _cons) label
```

Note here the use of variable names and wildcards in the `keep` option where `L*` keeps all the lags and `$controls` picks the names of the variables used in the global. We are basically throwing out the monthly dummies from the output.

Like the previous tables, we can export this table as follows:

```
esttab using "./graphs/guide80/regression6.tex", replace f  ///
 b(3) se(3) star(* 0.10 ** 0.05 *** 0.01) ///
 keep(stringency_index L* $controls _cons) ///
 label booktabs nonotes noobs nomtitle collabels(none)  ///
 scalars("N Obs." "CNTL Controls") sfmt(0) ///
 mgroups("Deaths" "Cases" "Tests" "Vaccines", pattern(1 0 1 0 1 0
1 0) prefix(\multicolumn{@span}{c}{) suffix(}) span
erepeat(\cmidrule(lr){@span}))
```

and in Overleaf (see document) we call the `rotating` package which allows us to use the `\sidewaystable` option and we get this table:

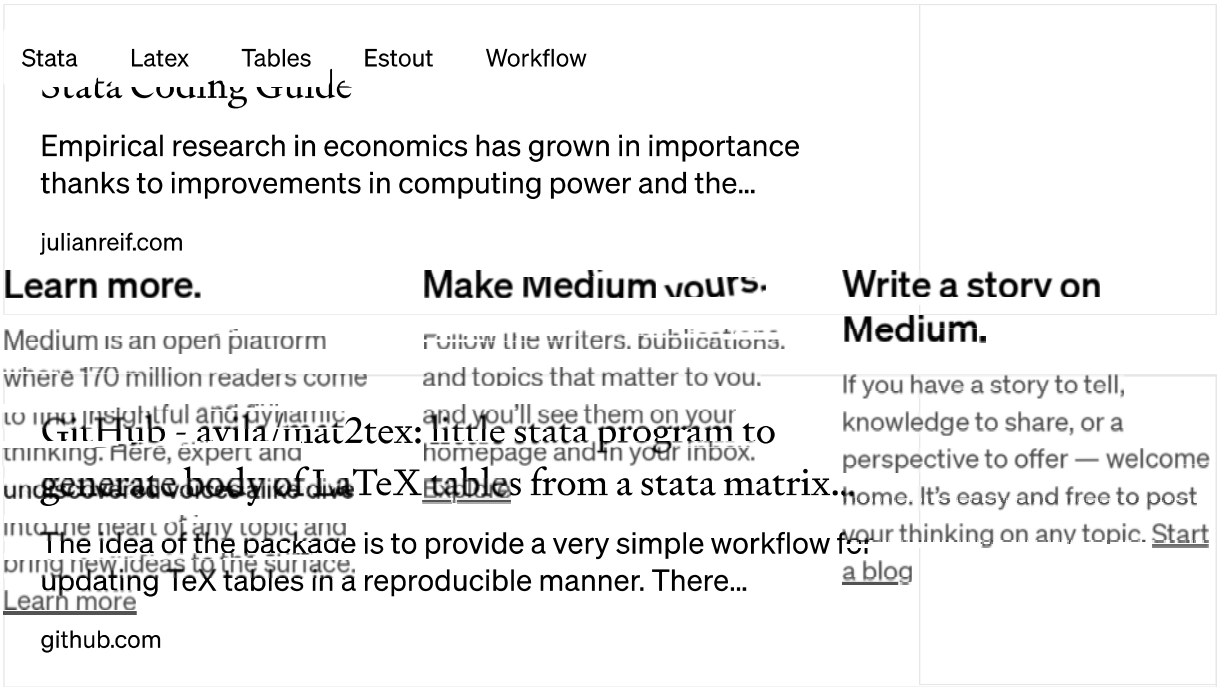
There are other ways of rotating a table as well. Here I have also not properly labeled the lags but these can be fixed directly in the code as well.

Miscellaneous

Please free to comment, report errors, or request additional tables. I will add more tables over time so this document will get populated with new templates. If you have a neat table code and would like it added here,

then feel free to email me the code and I can add it and attribute it your name.

There are also a lot of other great resources online that cover multiple



GitHub - avila/mat2tex: little stata program to generate body of LaTeX tables from a stata matrix...

The idea of the package is to provide a very simple workflow for updating TeX tables in a reproducible manner. There...

github.com

Please message if you know of more resources on this topic!

About the author

I am an economist by profession and I have been using Stata since 2003. I am currently based in Vienna, Austria where I work at the Vienna University of Economics and Business (WU) and at the International Institute for Applied Systems Analysis (IIASA). You can find my research