

EconEnv

One Notebook. Multiple Econometric Engines.

Python, R, Stata and EViews in a single Jupyter workflow

Installation and User Guide

for EconEnv 1.3.1

Developed by

Dr Merwan Roudane

<https://github.com/merwanroudane>

September 8, 2026

About this guide. It assumes nothing. Chapter 3 starts from a computer with no Python on it and finishes with all six programs talking to each other in one notebook. If you already have Python and Jupyter, skip to Section 3.3.

EconEnv does not contain, bundle or redistribute Stata or EViews. Those are commercial products which you must install and license yourself. R and Python are free and open source. EconEnv only connects to what is already on your machine.

Quick start. If you already have Python:

```
pip install econenv
```

then, in a notebook:

```
%load_ext econenv  
%econ doctor
```

Website: <https://merwanroudane.github.io/econenv/>
Repository: <https://github.com/merwanroudane/econenv> · Package:
<https://pypi.org/project/econenv/1.3.1/>

Contents

1	What EconEnv is, and why	4
1.1	The problem	4
1.2	What EconEnv does	4
1.3	What it is not	5
2	Requirements	6
2.1	Minimum	6
2.2	Optional engines	6
2.3	Operating systems	6
3	Installation from zero	7
3.1	Step 1 — Python	7
3.2	Step 2 — Jupyter	7
3.3	Step 3 — EconEnv	8
3.4	Step 4 — R (optional, free)	8
3.5	Step 5 — Stata (optional, commercial)	8
3.6	Step 6 — EViews (optional, commercial, Windows only)	9
3.7	Step 7 — Verify everything	9
4	First steps	10
4.1	Loading the extension	10
4.2	Sending data to an engine	10
4.3	Getting results back	10
4.4	Moving data between two engines directly	11
5	The magic commands	12
5.1	%econ — managing everything	12
5.2	%R — R	12
5.3	%stata — Stata	12
5.4	%eviews — EViews	13
6	EViews for people who use the GUI	14
6.1	Look commands up without leaving the notebook	14
6.2	The one rule that makes EViews commands guessable	14
6.3	Three ways to draw the same plot	14
6.4	The commands you will need first	15
7	MATLAB	16
7.1	Installing the bridge	16
7.2	Using it	16
7.3	Which release will start	17

8	GAUSS	18
8.1	Finding it	18
8.2	Three things that catch everyone out	18
8.3	A least squares cell, annotated	18
8.4	Finding the command you need	19
8.5	Data, and precision	19
8.6	What carries between cells	19
8.7	Figures	19
8.8	In the comparison	20
9	Exporting results for publication	21
9.1	One call	21
9.2	Two layouts	21
9.3	Formats	22
9.4	Significance stars	22
9.5	Figures, and why vector matters	22
9.6	What it will not do	23
10	Moving data between programs	24
10.1	One dataset, every engine	24
10.2	One at a time	24
10.3	What can move where	24
10.4	What survives the trip	25
11	Google Colab	26
11.1	What runs on a Colab cloud runtime	26
11.2	Stata on a Colab runtime	26
11.3	EViews on a Colab runtime — not possible	27
11.4	All six engines, still in Colab	27
12	A worked example	29
12.1	What it does	29
12.2	The model	29
12.3	A representative result	29
12.4	What deliberately does not agree	30
13	Reproducibility	31
13.1	Snapshots	31
13.2	Why this matters here more than usual	31
14	Troubleshooting	32
14.1	The extension will not load	32
14.2	An engine will not start	32
14.3	EViews reports the wrong version	32
14.4	rpy2 is installed but broken	32
14.5	A cell runs and shows nothing	33
15	Licensing and citation	34
15.1	EconEnv	34
15.2	The commercial programs	34
15.3	Developer	34
15.4	Citation	34

Chapter 1

What EconEnv is, and why

1.1 The problem

An applied econometrics paper rarely lives inside one program. The unit-root test is done in EViews because that is where the output is readable. The panel estimator is in Stata because `xtreg` is the reference implementation. The plots are in R because `ggplot2` is better. The data cleaning is in Python because `pandas` is better.

So the working day looks like this:

```
Python → to_csv() → Stata → export delimited → R → write.csv → EViews
```

Four programs open, four windows, four copies of the same data slowly drifting apart. A missing value that meant `.a` in Stata arrives as an empty cell in R. A quarterly index becomes a string. And when a referee asks why your robust standard error differs from theirs, there is no way to answer without redoing the whole chain by hand.

1.2 What EconEnv does

EconEnv turns the six programs into *execution engines behind one Python kernel*. You write:

```
%load_ext econenv
```

```
df = pd.read_csv("data.csv")          # Python, as usual
```

```
%%R -i df
fit <- lm(y ~ x1 + x2, data = df)
summary(fit)
```

```
%%stata
regress y x1 x2, robust
```

```
%%eviews -i df
equation eq1.ls y c x1 x2
eq1.output
```

One notebook. One kernel. One dataset. No CSV round-trip.

And then the part that is genuinely hard to do any other way:

```
econenv.compare_ols(df, "y ~ x1 + x2")
```

which estimates the same specification in every engine you have and puts the results side by side — reporting where they differ, and why, instead of quietly picking one.

1.3 What it is not

- It is **not** a new Jupyter kernel. It is a Python package and an IPython extension. Your notebook stays a normal Python notebook.
- It does **not** reimplement `%%stata`. Stata 17 and later ship official IPython magics, and EconEnv loads those rather than writing worse ones.
- It does **not** ship any commercial software, licence file or binary.
- It does **not** try to make the programs agree. Where their defaults differ, it says so.

Chapter 2

Requirements

2.1 Minimum

Component	Version	Needed for
Python	3.9 or newer	everything
pandas, numpy, IPython	installed automatically	everything
statsmodels	installed automatically	the Python engine
JupyterLab or Notebook	any recent	running notebooks

2.2 Optional engines

Install only what you use. None of these is required to install EconEnv.

Engine	Version	Notes
R	4.0+	Free. EconEnv finds it automatically; no PATH setup needed.
Stata	17 or newer	Commercial. PyStata ships with Stata 17+; earlier versions cannot be driven this way.
EViews	12, 13 or 14	Commercial. Windows only — automation is COM-based.

2.3 Operating systems

	Windows	Linux	macOS
Python	yes	yes	yes
R	yes	yes	yes
Stata	yes	yes	yes
EViews	yes	no	no

EViews automation uses Microsoft COM, which exists only on Windows. On Linux and macOS EconEnv installs and runs normally; the EViews engine simply reports itself unavailable and everything else works. Linux and macOS are supported by design but have not yet been exercised by the author — reports are welcome.

Chapter 3

Installation from zero

This chapter assumes a Windows machine with nothing installed. Adapt the paths if yours differ.

3.1 Step 1 — Python

Two reasonable routes.

Route A: python.org (simplest)

1. Go to <https://www.python.org/downloads/> and download Python 3.11 or 3.12 for Windows.
2. Run the installer. **Tick “Add python.exe to PATH”** on the first screen — this is the single most common thing people miss.
3. Open a new Command Prompt and check:

```
python --version
```

Route B: Anaconda / Miniconda

Convenient if you also want conda packages. Download from <https://www.anaconda.com/download>, then create a dedicated environment so you do not disturb the base one:

```
conda create -n econ python=3.12
conda activate econ
```

Whichever route you choose, remember *which* Python it is. If you install EconEnv into one interpreter and start Jupyter from another, the extension will not be found — and the error will not obviously say why. Section 14.1 covers this.

3.2 Step 2 — Jupyter

```
pip install jupyterlab
```

Start it with:

```
jupyter lab
```

3.3 Step 3 — EconEnv

```
pip install econenv
```

That is the whole core installation. It pulls in pandas, numpy, IPython and statsmodels, and nothing commercial.

Optional extras, installed only if you need them:

```
pip install "econenv[evIEWS]"    # comtypes, for EViews (Windows)
pip install "econenv[arrow]"    # faster data transfer to R
pip install "econenv[stata]"    # helper for locating PyStata
pip install "econenv[all]"      # all of the above
```

Check it worked:

```
econenv doctor
```

This prints a diagnosis of every layer — Python, Jupyter, R, Stata, EViews — and, for anything wrong, what to do about it.

3.4 Step 4 — R (optional, free)

1. Download R from <https://cran.r-project.org/bin/windows/base/>.
2. Run the installer and accept the defaults. It lands in C:\Program Files\R\R-4.x.x.
3. Nothing else is needed. EconEnv searches the registry and the usual install locations; you do not have to set R_HOME or touch PATH.

If you have several versions installed and want a specific one:

```
%econ config r.home "C:/Program Files/R/R-4.5.2"
```

You do not need rpy2. EconEnv's default R backend is a persistent Rterm child process, which needs no compiler and no R_HOME gymnastics. rpy2 publishes no Windows wheels, so `pip install rpy2` typically leaves you with a broken build. If you want it anyway, use conda-forge — see Section 14.4.

3.5 Step 5 — Stata (optional, commercial)

You need **Stata 17 or newer**, because that is when StataCorp began shipping PyStata.

1. Install and licence Stata normally.
2. Nothing else. EconEnv finds the installation, validates the executable and detects the edition (BE, SE or MP).

To pin a specific installation or edition:

```
%econ config stata.home "C:/Program Files/StataNow19"
%econ config stata.edition mp
```

3.6 Step 6 — EViews (optional, commercial, Windows only)

1. Install and licence EViews normally.
2. **Open EViews once, by hand, and close it.** This registers its COM automation server and validates the licence. Until you have done this, automation will not connect.
3. Install the COM bridge:

```
pip install "econenv[eviews]"
```

If you have more than one EViews version installed, the generic `EViews.Manager` identifier binds to whichever registered *last* — not the newest. On a machine with EViews 12, 13 and 14, it may well be 13 that answers. `%econ status` shows the version actually connected. To pin one:

```
%econ config eviews.progid EViews.Manager.14
```

Note the format: `EViews.Manager.14`, not `EViews14.Manager`.

3.7 Step 7 — Verify everything

In a notebook:

```
%load_ext econenv  
%econ status
```

You should see a table like this:

engine	state	version	edition	backend	location
Python	configured	3.11.0		cpython	C:\...\Python311
EViews	configured	13		comtypes	C:\Program Files\EViews 13
R	configured	4.5.2		subprocess	C:\Program Files\R\R-4.5.2
Stata	configured	19.5	mp	pystata	C:\Program Files\StataNow19

Then the full diagnosis:

```
%econ doctor
```

Every check reports PASS, WARNING or ERROR, and every warning and error carries a suggested fix.

Chapter 4

First steps

4.1 Loading the extension

Once per notebook:

```
%load_ext econenv
```

It prints which magics it registered and who owns each one:

```
EconEnv 1.3.1 loaded - one notebook, multiple econometric engines.
%econ                econenv
%Rec / %%Rec         econenv
%R / %%R             econenv (rpy2 not installed)
%evIEWS / %%evIEWS   econenv
%matlab / %%matlab    econenv
%stata / %%stata      pystata (official)
```

Note the last line. `%%stata` belongs to StataCorp, not to EconEnv. That means it takes *Stata's* options, not EconEnv's: `%%stata --result` is a syntax error. Move data with `%econ push` instead.

4.2 Sending data to an engine

Three equivalent ways:

```
%econ push r df          # line magic
```

```
%%R -i df                # inline, when running R code anyway
summary(df)
```

```
econenv.push("r", "df", df) # from Python
```

4.3 Getting results back

```
back = econenv.pull("stata")      # the current Stata dataset
one  = econenv.pull("evIEWS", "yf") # one EViews series
frame = econenv.pull("r", "mydata") # a named R data frame
```

R has no notion of a “current dataset” the way Stata and EViews do — every data frame is just a variable. `econenv.pull("r")` therefore returns the frame `EconEnv` last transferred; if there is none, it tells you so and lists the frames R actually holds. Name the frame explicitly when in doubt.

4.4 Moving data between two engines directly

```
econenv.move("stata", "eviews", "mydata")
```

No file is written, and no manual step is involved.

Chapter 5

The magic commands

5.1 %econ — managing everything

Command	What it does
%econ status	Table of every engine: state, version, backend, location
%econ engines	The same, in detail, with capabilities
%econ doctor	Full diagnosis with a fix for every problem
%econ versions	Version of EconEnv and of each engine
%econ capabilities	What each engine can do on this machine
%econ models	Which estimators are available in which engine
%econ start r	Start an engine explicitly
%econ stop / restart / reset	Session control
%econ config r.home "..."	Show or set a configuration value
%econ push <engine> <name>	Send a Python object to an engine
%econ pull <engine> [name]	Fetch an object back
%econ move <from> <to> <name>	Engine to engine
%econ snapshot	Reproducibility snapshot of every version
%econ eviews [topic]	Look up EViews commands (see below)
%econ ols y ~ x	Run one OLS across every engine

5.2 %%R — R

```
%%R -i df -o results
fit <- lm(y ~ x, data = df)
results <- coef(fit)
```

-i sends objects in, -o brings them back. -q suppresses output, --no-graphics suppresses plot capture.

If rpy2 is installed, EconEnv loads *rpy2's own* %R magics rather than shadowing them, and its own are always available as %Rec / %%Rec.

5.3 %%stata — Stata

This is StataCorp's official magic. Use Stata's own syntax and options:

```
%%stata
regress y x1 x2, robust
estat ic
```

5.4 %%eviews — EViews

```
%%eviews -i df
equation eq1.ls y c x1 x2
eq1.output
line y
```

-i pushes a DataFrame in as a workfile, -o pulls series out, -p selects a page, --no-graphs suppresses figures.

Chapter 6

EViews for people who use the GUI

This is the chapter most EViews users need, because in EViews you normally click, and in a notebook there is nothing to click.

6.1 Look commands up without leaving the notebook

```
%econ evIEWS          # the list of tasks
%econ evIEWS graph     # everything about plotting
%econ evIEWS find cointegration # search all of it
```

Each result gives the menu path you already know, the command, what it does, and an example:

```
g.coint(e)
  Johansen system cointegration test - trace and maximum-eigenvalue
  statistics for how many cointegrating relations exist.
  GUI: Group > View > Cointegration Test > Johansen
  e.g. g1.coint(e)
```

The catalogue holds 136 commands, 94 of them verified against EViews 13.

6.2 The one rule that makes EViews commands guessable

EViews objects work like this:

`object_name.what_you_want`

- Anything in an object's **View** menu is `object.thatview`.
- Anything in its **Proc** menu is `object.thatproc`, usually followed by a name for whatever it creates.

So `eq1.output` shows an equation's results, `x.line` plots a series, `v1.impulse` draws impulse responses.

6.3 Three ways to draw the same plot

All three work, and the GUI hides the distinction:

```
line x          % command form - quickest
x.line         % view form    - matches the View menu
graph gr1.line x % object form - keeps the graph so you can edit it
```

6.4 The commands you will need first

You would click	Command	What it does
File > New > Workfile	<code>wfcreate q 1990Q1 2020Q4</code>	Quarterly workfile
Quick > Generate Series	<code>series ly = log(y)</code>	Create or transform
Quick > Sample	<code>smp1 1995Q1 2015Q4</code>	Restrict the sample
Open as Group	<code>group g1 x y</code>	Bundle series
Series > View > Graph > Line	<code>x.line</code>	Plot
Descriptive Statistics	<code>x.stats</code>	Summary table
Correlogram	<code>x.correl</code>	ACF/PACF with Q-stats
Unit Root Test > ADF	<code>x.uroot(adf)</code>	Dickey–Fuller
Cointegration > Johansen	<code>g1.coint(e)</code>	Trace and max-eigen
Estimate Equation > LS	<code>equation eq1.ls y c x</code>	OLS; c is the constant
Estimate Equation > ARCH	<code>equation eq1.arch(1,1) y c x</code>	GARCH(1,1)
Estimate VAR	<code>var v1.ls 1 2 x y</code>	VAR, lags 1 to 2
Equation > View > Output	<code>eq1.output</code>	Results table
Actual, Fitted, Residual	<code>eq1.resids</code>	The first thing to look at
Serial Correlation LM	<code>eq1.auto(2)</code>	Breusch–Godfrey
Heteroskedasticity, White	<code>eq1.white</code>	White’s test
Recursive > CUSUM	<code>eq1.rls(q)</code>	Parameter stability
Recursive > CUSUM squares	<code>eq1.rls(v)</code>	Variance stability
Forecast	<code>eq1.forecast(g) yf</code>	Forecast and plot

The complete reference is in `docs/engines/evIEWS-commands.md`.

If a command runs but you see nothing, that is a bug in EconEnv, not in what you typed. Every display command should produce output, a plot, or a warning saying it could not be read. Silence is never correct — please report it.

Chapter 7

MATLAB

MATLAB joins the other four through the official MATLAB Engine API for Python, which MathWorks ships inside every installation and publishes on PyPI.

7.1 Installing the bridge

The engine package must match the MATLAB release *and* your Python. Both windows are narrow:

MATLAB	Engine package	Python
R2024a	<code>pip install "matlabengine==24.1.*"</code>	3.9–3.11
R2024b	<code>pip install "matlabengine==24.2.*"</code>	3.9–3.12
R2025a	<code>pip install "matlabengine==25.1.*"</code>	3.9–3.12
R2025b	<code>pip install "matlabengine==25.2.*"</code>	3.9–3.12
R2026a	<code>pip install "matlabengine==26.1.*"</code>	3.9–3.13

The Python range matters as much as the release. R2024a’s engine supports Python 3.9 to 3.11 and refuses anything newer, so a mismatch is an import error rather than a wrong answer. Python 3.13 needs R2026a or later; on an earlier MATLAB, make an environment your release can drive (`conda create -n econ python=3.11`). `econenv doctor` works this out for you — it names a pin for every release you have installed, and if none suits your Python it says which release would, and which Python your own MATLAB can drive.

7.2 Using it

```
%%matlab -i df -o beta
fit = fitlm(df, 'y ~ x1 + x2');
beta = fit.Coefficients.Estimate;
disp(fit)
```

`-i` pushes a DataFrame in as a MATLAB table; `-o` brings a variable back.

MATLAB is slow to start — roughly a minute cold. The first cell that uses it pays for the whole session and every later one is fast. That is also why `%econ status` never starts it.

If MATLAB is already open, share it and EconEnv attaches instead of launching a second copy: run `matlab.engine.shareEngine` in MATLAB, then `%econ config matlab.shared MATLAB_shared`.

7.3 Which release will start

If you have several MATLAB releases installed, the one that starts is the one your *engine package* matches — not the newest on disk. `%econ status` reports the release that will actually run.

Chapter 8

GAUSS

GAUSS is driven through its terminal executable, `tgauss`. Everything in this chapter was established by running GAUSS 26.1.1 rather than read from documentation, which matters because several plausible assumptions turned out to be wrong.

8.1 Finding it

Nothing to install beyond GAUSS itself. EconEnv looks at `gauss.home`, then the `GAUSSHOME` and `MTENGHOME` environment variables, then the usual locations.

GAUSS does *not* install into Program Files on Windows. A typical path is `C:\gauss26`, so a search that only looks where other vendors put things finds nothing. If yours is elsewhere:

```
%econ config gauss.home C:/gauss26
```

Several GAUSS versions can be installed and they do not all work. `%econ doctor gauss` names the one EconEnv will actually use and lists the others — reporting the newest directory instead would repeat the mistake the EViews adapter made with ProgIDs.

8.2 Three things that catch everyone out

1. **Every statement ends with a semicolon.** A missing one is a syntax error, not a warning.
2. **A bare expression prints nothing.** Where Python echoes the last value, GAUSS needs `print x;`.
3. **`~` joins columns, `|` stacks rows.** This is how a design matrix is built and has no counterpart in the other engines' syntax.

And one that looks like a bug: GAUSS **compiles the whole cell before running any of it**. A mistyped name on the last line means the first line never ran either, so a cell can produce no output at all rather than a partial result. EconEnv says so explicitly, because the natural reading — “my print statement did nothing” — sends people looking in the wrong place.

8.3 A least squares cell, annotated

```

%%gauss -i df -o b
y = df[:,3];
X = ones(rows(df),1) ~ df[:,1] ~ df[:,2];
b = y / X;
print b;

```

/ is the least-squares solve, not division: GAUSS's equivalent of `numpy.linalg.lstsq`, and more accurate than forming `inv(X'X)X'y` by hand. For standard errors use `ols`, which is what `compare_ols` calls.

8.4 Finding the command you need

```

%econ gauss           # the categories
%econ gauss regression # ols and friends
%econ gauss find missing # search everything

```

95 commands in 12 categories, 65 of them resolved against a live GAUSS. The operators are catalogued too, because they are the part a Python or R user cannot guess.

8.5 Data, and precision

A GAUSS matrix holds only numbers. A DataFrame's text columns cannot cross, and they are *named in a warning* rather than dropped in silence — a regression run on a frame that quietly lost a column is the worst outcome available. Column names are kept on the Python side, so the frame comes back with them. NaN crosses as a GAUSS missing value and returns as NaN.

Numbers cross at seventeen significant digits, which round-trips an IEEE double exactly. GAUSS's own `csvWriteM` writes about fifteen, and a round trip through it moves a value by roughly 3×10^{-15} — enough to put GAUSS out of step with the machine-precision agreement that is the point of `compare_ols`. EconEnv ships its own writer for that reason, and the measured round-trip difference is zero.

8.6 What carries between cells

Each cell is a fresh process, so EconEnv carries two things across: the *values*, using GAUSS's own `save` and `load`; and the *procedure definitions*, re-declared in each later cell. So a `proc` written in one cell is callable in the next.

Carrying a definition is safe where replaying a statement is not: a `proc ... endp;` block computes nothing and touches nothing, so re-declaring it cannot change an answer. `#includes` and `library` statements still do not carry.

That limit is stated rather than worked around: replaying earlier cells to fake a session would silently re-run their side effects.

8.7 Figures

Plots are captured and shown in the notebook. The default is SVG — vector, and what most journals ask for.

```

%econ config gauss.graphics svg # svg | png | pdf | off
%econ config gauss.width 1600  # pixels, for png only

```

A cell that draws nothing produces no figure: EconEnv only asks GAUSS to save a plot when the cell contains a plotting call, because `plotSave` would otherwise write out whatever was drawn last and the same figure would reappear under every later cell.

`plotSave` measures raster and vector differently: the two numbers are pixels for PNG and inches for SVG and PDF. The `12 | 9` that gives a sensible SVG gives a 12-by-9 *pixel* PNG. EconEnv passes the right units for the format you chose. A `struct` also cannot be carried between cells — GAUSS refuses to `save` one. Since a plotting cell declares `struct plotControl p;`, EconEnv recognises struct declarations and leaves them out of the carried workspace.

8.8 In the comparison

```
econenv.compare_ols(df, "y ~ x1 + x2")
```

GAUSS's own `ols` supplies the numbers. Maximum disagreement with Python across six engines is 1.8×10^{-15} ; R^2 , adjusted R^2 , log-likelihood, AIC, BIC and RMSE match statsmodels, Stata and MATLAB exactly. `ols` stops at coefficients, standard errors, sigma and R^2 , so the t statistics, p -values, interval and information criteria are computed in GAUSS from its own residuals using `cdftc` and `cdftci`.

Chapter 9

Exporting results for publication

Getting a table out of a notebook and into a paper is where reproducibility usually breaks: numbers are retyped, a coefficient is rounded twice, a regression is re-run and the manuscript is not. EconEnv exports from the result object itself, so the file and the estimation cannot drift apart.

9.1 One call

```
econenv.export(cmp, "paper/table1", formats=["tex", "docx", "xlsx"])
```

writes `table1.tex`, `table1.docx` and `table1.xlsx` from the same object. From a cell, importing nothing:

```
%econ export cmp paper/table1 --formats tex,docx --caption "Table 1"
```

What you pass can be a single model, a comparison across six engines, a DataFrame, a figure, or a list mixing them.

9.2 Two layouts

Journal — the default

What a paper prints: one column per model, the coefficient with significance stars, the standard error beneath it in parentheses, and a foot of sample size and fit statistics.

```
econenv.export(cmp, "table1.tex", caption="Consumption function",  
              label="tab:cons")
```

produces the following — and this output was compiled with `pdflatex` during development, which is how the escaping rules were settled:

	Python	EViews	MATLAB	R	Stata
<code>_cons</code>	1.4805*** (0.0350)	1.4805*** (0.0350)	1.4805*** (0.0350)	1.4805*** (0.0350)	1.4805*** (0.0350)
<code>x1</code>	1.9331*** (0.0395)	1.9331*** (0.0395)	1.9331*** (0.0395)	1.9331*** (0.0395)	1.9331*** (0.0395)
Observations	120	120	120	120	120
R^2	0.9597	0.9597	0.9597	0.9597	0.9597

Full — every statistic

One row per term per engine, with coefficient, standard error, test statistic, p-value and confidence interval as separate columns. For your own checking rather than for a paper.

```
econenv.export(cmp, "check.xlsx", style="full")
```

9.3 Formats

Format	Ext	Needs	Notes
LaTeX	.tex	—	booktabs rules, ready to input
Word	.docx	export extra	a real editable table, not a picture
Excel	.xlsx	export extra	one sheet per table, full precision
CSV	.csv	—	
HTML	.html	—	self-contained, styled like a journal table
Markdown	.md	export extra	
RTF	.rtf	—	for submission systems that still want it

```
pip install "econenv[export]"
```

A filename with a suffix selects that format; a stem gives you the journal bundle of `tex`, `docx` and `xlsx`.

9.4 Significance stars

The economics convention is the default — *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$ — and the matching note is written under the table automatically. For a journal that uses different thresholds:

```
econenv.export(cmp, "t.tex",
               stars=((0.001, "***"), (0.01, "**"), (0.05, "*")))
```

9.5 Figures, and why vector matters

A PNG is fixed pixels: print it at column width and it looks fine, print it larger and it does not. A PDF or EPS is instructions, so it stays sharp at any size — which is why most economics journals ask for vector figures.

Ask the engine for vector output *before* running the cell, because a bitmap cannot be converted into one afterwards:

```
%econ config matlab.graphics pdf
%econ config r.graphics pdf
%econ config eviews.graphics svg
```

EconEnv writes figures in whatever the engine actually produced, and the extension follows the *content* rather than the filename — naming a PNG `.pdf` makes a file nothing can open. If you export raster when vector was wanted, the result says so and names the setting to

change, rather than wrapping a bitmap in a PDF container and calling it vector.

9.6 What it will not do

- **Invent a statistic.** A number the engine did not report is left blank, not computed here under different assumptions. A table that quietly mixes two engines' conventions is worse than one with a gap.
- **Fake vector output.**
- **Round twice.** The journal layout formats once from the full-precision value; the Excel export keeps the unrounded number.

Chapter 10

Moving data between programs

This is the feature the whole project exists for: build a dataset once and use it everywhere, without a single CSV.

10.1 One dataset, every engine

```
econenv.broadcast("macro", df)
```

```
{'evIEWS': None, 'matlab': None, 'r': None, 'stata': None}
```

`None` means it arrived. An engine that is missing or fails is recorded rather than stopping the rest, because a machine without MATLAB should still get the data into R and Stata. Pass `strict=True` to raise instead, or name the engines you want.

10.2 One at a time

```
econenv.push("stata", "macro", df)      # Python to Stata
back = econenv.pull("r", "macro")       # R to Python
econenv.move("stata", "matlab", "macro") # Stata to MATLAB
```

Or inline, while running code anyway:

```
%%R -i macro -o results
%%evIEWS -i macro
%%matlab -i macro -o beta
```

10.3 What can move where

```
econenv.transfer_matrix()
```

reports what is possible on *this* machine rather than in principle. An engine that is installed but not configured cannot take your data, and learning that before you start is cheaper than a failure mid-session.

10.4 What survives the trip

Type	Behaviour
Numeric	exact
Dates	preserved; the index becomes a column in R and MATLAB, and a dated page in EViews
Booleans	preserved, except that MATLAB logicals cannot be missing
Categoricals	become factors in R, categoricals in MATLAB
Strings	preserved; missing strings become empty in MATLAB
Missing values	preserved, with Stata's extended missing values noted

Anything lossy warns, naming the column and what changed. Anything that would alter a value raises instead.

EViews names work differently. Pushing to EViews creates a *page of series*, not an object named after the frame — there is no `macro` object in the workfile, only `X`, `Y`, `Z`. EconEnv remembers the name you pushed under so `pull("eviews", "macro")` returns the page, but inside a `%%eviews` cell you refer to the series by their own names.

Chapter 11

Google Colab

Colab is Linux, and that settles what is possible before anything is installed.

11.1 What runs on a Colab cloud runtime

Engine	Colab	Why
Python	works	it is the kernel
R	works	R is already on the Colab image
Stata	possible	Stata for Linux exists and pystata supports it
EViews	no	no Linux build, and no permitted workaround

Install as usual:

```
%pip install econenv
```

11.2 Stata on a Colab runtime

This does work, if you hold a Stata **Linux** licence. Keep the Linux tarball and your `stata.lic` in Google Drive, mount Drive, and install at the top of the notebook:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
!mkdir -p /usr/local/stata19
!tar -xzf "/content/drive/MyDrive/stata/Stata19Linux64.tar.gz" -C /usr/local/stata19
!cd /usr/local/stata19 && yes | ./install
!cp "/content/drive/MyDrive/stata/stata.lic" /usr/local/stata19/
```

EconEnv finds it with no configuration: the Linux executable names (`stata-mp`, `stata-se`, `stata`) and `/usr/local` have been part of discovery from the start.

Three caveats, none of them technical. You need a **Linux** licence entitlement — a Windows-only licence does not cover this. The runtime is destroyed at the end of the session, so the install repeats every time. And whether your licence permits installation on a disposable cloud VM is a question for StataCorp, not for this project.

11.3 EViews on a Colab runtime — not possible

Three independent blockers, none of which EconEnv can route around:

1. **There is no Linux build.** EViews ships for Windows and macOS.
2. **Wine does not work.** EViews cannot read a valid machine ID under Wine, so licence activation fails.
3. **Remote access is forbidden.** The obvious workaround — run EViews on your Windows machine and reach it from Colab over a tunnel — is ruled out by EViews’ own documentation, which states that “*web server access to EViews via COM is not allowed*” and limits remote Distributed COM to a single instance.

The third is the one that settles it: that bridge is straightforward to build and contractually prohibited, so EconEnv does not ship it.

11.4 All six engines, still in Colab

There is a way to have the Colab interface *and* every engine, and it avoids all of the above.

Colab runs in a browser **on your own PC**. Google lets you point that interface at a Jupyter server running on that same PC — a **local runtime**. The notebook UI stays Colab; the kernel, and therefore Python, R, Stata and EViews, is your Windows machine.

Nothing is exposed to the internet. Your browser talks to `localhost`, Google’s servers never reach your machine, and EViews is driven by local COM exactly as it would be in a local notebook. There is no web server in front of EViews, so this is not the restriction quoted above.

It needs the **classic** Jupyter stack. The bridge package `jupyter_http_over_ws` was last released in March 2020 and is a notebook 5/6 server extension; on notebook 7 and `jupyter_server` 2 it does not load at all.

Stack	Enable step	/http_over_websocket
notebook 7.5.5	fails	404
notebook 6.5.7	fails	404
notebook 6.4.12	validates	HTTP 400

HTTP 400 is the correct answer: the endpoint exists and is waiting for the websocket upgrade Colab performs. Both other stacks run on `jupyter_server` 2, which will not load a legacy server extension.

So use a dedicated environment:

```
python -m venv colab-runtime
colab-runtime/Scripts/pip install notebook==6.4.12 jupyter_http_over_ws econenv
colab-runtime/Scripts/jupyter serverextension enable --py jupyter_http_over_ws
colab-runtime/Scripts/jupyter notebook --no-browser --port=8888
--NotebookApp.port_retries=0
--NotebookApp.allow_origin="https://colab.research.google.com"
```

Copy the `http://localhost:8888/?token=...` line it prints. In Colab, click the **Connect** arrow, choose **Connect to a local runtime**, and paste the URL.

Then everything works, because the kernel is your PC:

```
%load_ext econenv
%econ status
```

Two practical caveats: your PC must stay awake with the server running for as long as the notebook is open, and that environment is separate from your usual one, so install into it whatever the notebook needs.

Chapter 12

A worked example

The notebook `examples/10_real_data_four_engines.ipynb` works through a complete exercise on **real US quarterly macroeconomic data, 1959Q1 to 2009Q3** — 203 observations, shipped with statsmodels, so it needs no download.

12.1 What it does

Step	Engine	Why that one
Load and transform	Python	pandas is the best tool for this
Describe, plot, diagnose	R	<code>plot(fit)</code> and R's diagnostic vocabulary
HAC standard errors	Stata	<code>newey</code> is the reference implementation
Unit roots, cointegration	EViews	this is what EViews is for
Forecast and stability	EViews	CUSUM, forecast bands
Compare all six	EconEnv	one specification, six engines, one table

12.2 The model

A textbook consumption function,

$$\log C_t = \beta_0 + \beta_1 \log Y_t + \beta_2 r_t + \varepsilon_t,$$

with C real consumption, Y real GDP and r the real interest rate.

Both series trend strongly, so the interesting questions — are they non-stationary, are they cointegrated, are the standard errors trustworthy — are exactly the ones each program answers differently. That is what makes it a good test of the whole idea.

12.3 A representative result

EViews, on log real GDP:

```
Null Hypothesis: LRGDP has a unit root
Exogenous: Constant
Lag Length: 1 (Automatic - based on SIC, maxlag=14)

                                t-Statistic   Prob.*
Augmented Dickey-Fuller test statistic   -1.820451   0.3698
```

Test critical values:	1% level	-3.462901
	5% level	-2.875752

and the four-engine comparison of the same regression:

Coefficients	python	evIEWS	r	stata
term				
_cons	1.4805338	1.4805338	1.4805338	1.4805338
lrgdp	0.8863268	0.8863268	0.8863268	0.8863268
realint	-0.0936228	-0.0936228	-0.0936228	-0.0936228

Maximum disagreement across the four: of the order of 10^{-15} — floating point noise and nothing more.

12.4 What deliberately does not agree

The information criteria differ, and EconEnv reports why rather than choosing one:

- `statsmodels`: $-2\ell + 2k$
- R: counts σ^2 as a parameter, so $k + 1$
- EViews: divides by n
- Stata: from `estat ic`, with $k = e(\text{rank}) + 1$

None is wrong. They answer slightly different questions, and a comparison that hid that would be worse than useless.

Chapter 13

Reproducibility

13.1 Snapshots

```
econenv.snapshot()
```

records the EconEnv version, the Python version and packages, the R version, the Stata version and edition, the EViews version, the operating system, architecture and timestamp. Export it as JSON and keep it with your results.

13.2 Why this matters here more than usual

A result produced across four programs has four times as many ways to become irreproducible. A snapshot taken at the time of estimation is the difference between “we used Stata” and “we used StataNow 19.5 MP with PyStata, on Windows, in March”.

Chapter 14

Troubleshooting

14.1 The extension will not load

```
ModuleNotFoundError: No module named 'econenv'
```

Almost always: Jupyter is running on a different Python from the one you installed EconEnv into. Check from inside the notebook:

```
import sys; print(sys.executable)
```

then install into exactly that interpreter:

```
!{sys.executable} -m pip install econenv
```

14.2 An engine will not start

```
%econ doctor
```

Every failing check names a fix. The usual causes:

- **EViews**: never opened by hand, so its COM server is not registered. Open it once and close it.
- **EViews**: `comtypes` not installed — `pip install "econenv[eviews]"`.
- **Stata**: version 16 or earlier. PyStata requires Stata 17.
- **R**: installed somewhere unusual — set `r.home`.

14.3 EViews reports the wrong version

With several versions installed, the generic identifier binds to whichever registered last. `%econ status` always shows the one actually connected. Pin a version with `%econ config evIEWS.progid EViews.Manager.14`.

14.4 rpy2 is installed but broken

A typical symptom:

```
ImportError: cannot import name 'SexpVectorCCompatibleAbstract'  
from 'rpy2.rinterface_lib.sexp'
```

This means a version built for an older Python. PyPI has no Windows wheels for rpy2, so `pip install rpy2` cannot fix it. Use conda-forge:

```
pip uninstall -y rpy2
conda install -c conda-forge rpy2
```

conda-forge's rpy2 brings *its own R* with it, separate from any R already installed. Packages in your existing R will not be visible to it. Unless you specifically need rpy2, stay on EconEnv's subprocess backend, which works without any of this.

14.5 A cell runs and shows nothing

Report it. Every display command should produce output, a plot, or a warning. Silence is a bug: <https://github.com/merwanroudane/econenv/issues>

Chapter 15

Licensing and citation

15.1 EconEnv

MIT licence. Free for any use, including commercial.

15.2 The commercial programs

EconEnv’s licence says nothing about Stata or EViews. It contains no part of either, and connects only to installations you have licensed yourself. Your obligations under those licences are unchanged.

15.3 Developer

Dr Merwan Roudane — author and maintainer of EconEnv.

- GitHub: <https://github.com/merwanroudane>
- Repository: <https://github.com/merwanroudane/econenv>
- Package: <https://pypi.org/project/econenv/>
- Issues and feature requests: <https://github.com/merwanroudane/econenv/issues>

Contributions are welcome. If EconEnv reports something incorrectly, or a command runs and shows nothing, that is a bug worth reporting — silence is never the intended behaviour.

15.4 Citation

Roudane, M. (2026). EconEnv: One Notebook, Multiple Econometric Engines. Python package. <https://github.com/merwanroudane/econenv>

A CITATION.cff file in the repository is kept up to date, and GitHub will format a citation from it automatically.