```
      name:  <unnamed>
       log:  /Users/j4-smith/GitHub/AI-SDC/ACRO/stata/acro_demo_2026.smcl
  log type:  smcl
 opened on:  11 Feb 2026, 15:59:14

 1 . do acro_demo_2026.do

 2 . capture program drop engage

 3 . program engage
  1.   di ""
  2.   di as err "==press <return> to continue==" _request(dummy)
  3.   di ""
  4. end

 4 .
 5 .
 6 .
 7 . ************************************************************
 8 . * Simple introduction to using acro for Stata researchers. *
 9 . *                                                          *
10 . * Author: Jim Smith. 2026                                  *
11 . ************************************************************
12 .
13 . quietly {
  ACRO demonstration
 This is a simple do-file to get you started with using the acro package to add disclosure risk control  to your analysis.
 This is an interactive demonstration, so occasionally you will be prompted to hit <return> to either:
  - display the next piece of  information or
  - run the next code snippet.

 In displaying some of the examples below we have used the terms 'dollar' 'backtick_' and '_tick'
  You will need to replace them with the appropriate symbols in your actual code

 ==press <return> to continue==.

  A: The basic concepts

 1 A research session:
 by which we mean the activity of running a series of commands (interactively or via a script) that:
  -  ingest some data,
  -  manipulate it, and then
  -  produce (and store) some outputs.

 2 Types of commands:
 Whether interactive, or just running a final script, we can think of the commands that get run in a session as dividing into:
  - manipulation commands that load and transform data into the shape you want
  - feedback commands that report on your data - but are never intended to be exported.
    - in Stata these might use the variables window, or the display command
  - query commands that produce an output from your data (table/plot/regression model etc.)
      that you might want to export from the Trusted Research Environment (TRE)

  3 Risk Assessment vs decision making:
  SACRO stands for Semi-Automated Checking of Research Outputs.
  The prefix 'Semi' is important here - because in a principles-based system humans should make decisions about output requests.
  To help with that we provide the SACRO-Viewer, which collates all the relevant information for them.

  A key part of that information is the  Risk Assessment.
  - Since it involves calculating metrics and comparing them to thresholds (the TRE's risk appetite)
     it can be done automatically, at the time an output query runs on the data.
  - This is what the ACRO tool does when you use it as part of your workflow.


 ==press <return> to continue==.

  4 What ACRO does
  The ACRO package aims to support you in producing Safe Outputs within minimal changes to your work flow.
  To do that we provide:
  - drop-in replacements for the most commonly used output commands,
    - keeping the same syntax as the originals, and
    - supporting as many of the options as we can
        (features supported will increase over time in response to demand).
  - a set of session-management commands to help you manage the set of files you request for output.
 Important to note that currently acro outputs results (tables, details of regression models etc.) as .csv files.
    - In other words we separate the processes of creating outputs - which must be done inside the TRE.
      from the process of formatting them for publication - which can be done outside the TRE with your preferred toolchain.
    - ACRO handles creation. We are interested in hearing from researchers whether it is important to support them with formatting.

  5 What ACRO doesn't support (yet)
  - Weightings as an option when creating tables/regressions
    We would greatly appreciate input on which versions peope use most
    so we can plan their implementation
  - Abbreviations/synonyms e.g. reg for regress etc.
    this would make a nice (and easy) job for someone who wanted to get involved supporting this initiative!
  - Statements that combined manipulation and query commands, for example,
     xi: regress wage i.year age
    Workaround You can achieve this by creating the dummy variables before calling your query command:
     xi year
     regress wage  backtick_ds _I*_tick age
    NB: ACRO needs to be passed an explicit list of independent variables rather than Stata's i.year shortcut
    The easiest way to avoid listing the indicator variables manually is to use the macro ds_I* as shown
      - Stata will expand this automatically before it gets passed to acro
    See that Stata manual https://www.stata.com/manuals15/rxi.pdf for further discussion

 ==press <return> to continue==.
```

B Getting Started with the demonstration

Step 1: Setting up the environment with the tools we will use
The acro package should be installed in your system
 and you will just need to tell Stata where to find the *acro.ado* file that handles the interaction between Stata and the back-end Python package.
Depending on how *acro* has been set up for you, this may be installed system-wide, or you may need to have it in your working directory

```
adopath + "."
```

We will also set the version of Stata to use version 18 - feel free to edit this file and change this

```
version 18
```

Step 2 Starting an ACRO session
 To do this we create an acro object by running the command below.

```
acro init, config(default)
acro disable_suppression
```

 You can leave out the default parameters, but this example shows how you can:
 - provide the name of a *config* (risk appetite) file the TRE may have asked you to use
 - turn automatic suppression on or off right from the start of your session

 Note that when the cell runs it should report (in a different coloured font/background)
 - what version of acro is running: **this should be 0.4.12**
 - the TRE's risk appetite: that defines the rules your outputs will be checked against.
 - whether suppression is automatically applied to disclosive outputs.

```
INFO:acro:version: 0.4.12
INFO:acro:config: {'safe_threshold': 10, 'safe_dof_threshold': 10, 'safe_nk_n': 2, 'safe_nk_k': 0.9, 'safe_pratio_p': 0.1, 'check_missing_values': False, 'survival_sa
> fe_threshold': 10, 'zeros_are_disclosive': True}
INFO:acro:automatic suppression: False
acro analysis session created
suppression toggled off for subsequent commands
```

 Step 3: Loading some test data

 The following stages in this step just do standard ingestion and manipulation commands to load some data into Stata ready to be queried.
 We will use some open-source data about nursery admissions.

 There is no change to your workflow here
 - Do whatever you want in this step!
 - We just assume you end up with your data in Stata.

```
use "../data/nursery_data"
```

==press <return> to continue==.

C Producing tables that are 'Safe Outputs
 acro aims to reproduce the functionality of the Stata **table** command by mapping it across to the equivalent crosstab() command provided by the industry-standard Pytho
> n package *Pandas*.
Important to note the syntax remains the same as in Stata, just with the prefix **acro** - except for two differences:
1. We do not currently support the weights options
   - we need more input from researchers about how much of a priority these are,
    especially as the functionality can be recreated by creating extra variables in Stata.
2. We do not support abbreviations (creating a lookup table for this would be a nice first issue if someone wanted to contribute).

The code should automatically accomodate the syntax from Stata versions <=16 and 17+
  - in the examples below we will show the syntax from versions 17 and beyond,
 So the syntax is **acro table rowvars colvars [if] [in] [, *]**

and you can specify what the table cells contain by:
    - providing a statistic - for example: mean, count, std deviation, median etc.
    - specifying what variable to report on
 The acro version uses Stata functionality to interact with python, and all the pandas code.
 - but it adds extra code that checks for disclosure risks depending on the statistic you ask for.

==press <return> to continue==.

Example 1 A simple 2-D table of frequencies stratified by two variables

```
acro table recommend parents

Total
INFO:acro:get_summary(): fail; threshold: 5 cells may need suppressing;
INFO:acro:outcome_df:
------------------------------------------------------------------|
parents    |great_pret   |pretentious  |usual        |Total       |
recommend  |             |             |             |            |
------------------------------------------------------------------|
not_recom  |          ok |          ok |          ok |          ok|
priority   |          ok |          ok |          ok |          ok|
recommend  | threshold;  | threshold;  | threshold;  | threshold; |
spec_prior |          ok |          ok |          ok |          ok|
very_recom | threshold;  |          ok |          ok |          ok|
Total      |          ok |          ok |          ok |          ok|
------------------------------------------------------------------|

INFO:acro:records:add(): output_0
Total
-------------------------------------------------------|
parents     |great_pret  |pretentious  |usual  |Total|
recommend   |            |             |       |     |
-------------------------------------------------------|
not_recom   |1440        |1440         |1440   | 4320|
priority    | 858        |1484         |1924   | 4266|
recommend   |   0        |   0         |   2   |    2|
spec_prior  |2022        |1264         | 758   | 4044|
very_recom  |   0        | 132         | 196   |  328|
Total       |4320        |4320         |4320   |12960|
-----------------------------------------------------|


==press <return> to continue==.
```

How to understand this output
 The top part (in red font) is the risk analysis produced by acro.
 It is telling us that:
 - the overall summary is **fail** because 4 cells are failing the 'minimum threshold' check
 - then it is showing which cells failed so you can choose how to respond
 - finally it is telling us that is has saved the table and risk assessment to our acro session with id 'output_0'

 The part below is the normal output produced by python mimicking the the Stata **table** function.
 - As this is such a small table it is not hard to spot the four problematic cells with zero or low counts
 - but of course this might be harder for a bigger table.

How to respond to this input
There are basically three choices:
1. We might decide these low numbers reveal something where the public interest outweighs the disclosure risk.
Rather than being a strict rules-based system, acro lets you attach an 'exception request' to a named output, to send a message to the output checkers.
For example, you could type:
  acro add_exception "output_0" "I think you should let me have this because..."

2. We redesign our data so that table so that none of the cells in the resulting table represent fewer than *n* people (10 for the default risk appetite)
For example, we could recode '*very_recommend*' and '*priority*' into one label.
But maybe it is revealing that the '*recommend*' value is not used?

3. We can redact the disclosive cells - and **acro will do this for us.**
We simply enable the option to suppress disclosive cells and re-run the query.

The command below shows option 3.
When you run the cell below you should see that:
 - the status now changes to *review* (so the output-checker knows what has been applied)
 - the code automatically adds an exception request saying that suppression has been applied
 - and, most importantly, the cells are redacted.
acro enable_suppression
acro table recommend parents

```
==press <return> to continue==.

suppression toggled on for subsequent commands
Total
INFO:acro:get_summary(): review; threshold: 5 cells suppressed;
INFO:acro:outcome_df:
--------------------------------------------------------------------|
parents    |great_pret   |pretentious  |usual       |Total       |
recommend  |             |             |            |            |
--------------------------------------------------------------------|
not_recom  |          ok |          ok |          ok |          ok|
priority   |          ok |          ok |          ok |          ok|
recommend  | threshold;  | threshold;  | threshold;  | threshold; |
spec_prior |          ok |          ok |          ok |          ok|
very_recom | threshold;  |          ok |          ok |          ok|
Total      |          ok |          ok |          ok |          ok|
--------------------------------------------------------------------|

INFO:acro:records:add(): output_1
INFO:acro:records:exception request was added to output_1
Total
-----------------------------------------------------|
parents     |great_pret  |pretentious  |usual  |Total|
recommend   |            |             |       |     |
-----------------------------------------------------|
not_recom   |1440.0      |1440         |1440   | 4320|
priority    | 858.0      |1484         |1924   | 4266|
spec_prior  |2022.0      |1264         | 758   | 4044|
very_recom  |   NaN      | 132         | 196   |  328|
Total       |4320.0      |4320         |4318   |12958|
-----------------------------------------------------|
```

 An example of a more complex table
 Just to demonstrate the sort of tables that can be made, make something more complex.
acro enable_suppression
acro table (parents finance) recommend, statistic(mean children) statistic (mode children) margins('total')

 Going through the parameters in order:
 - passing a list of variable names to 'rowvars' (rather than a single variable/column name) tells it we want a hierarchy within the rows.
   - we can do the same to columns as well (or instead) if we want to
 - the two **statistic()** options specify reporting the mean (which introduces additional risks of *dominance*)and mode of the number of children
   We are aware of (and working on) some rare issues when reporting standard devistions in a suppressed table
 - setting **margins(total)** tells it to display row and column sub-totals

 It's worth noting that including the totals there are 6 columns in the risk assessment and 5 in the suppressed table.
 This is because after suppression has replaced numbers with 'NaN', pandas removes the fully suppressed column ('recommend') from the table.

 This highlights a wider issue about Stata's inconsistent treatment of missing values
For reasons best known to itself, Stata sometimes (but not always) interprets missing numerical values as **extremely high numbers**
    so in one example we saw recently it reported a lot of people aged 250 years and above ...
By contrast Pandas removes records with missing values if needed, whis is more sane
  You can replicate Stata's behaviour by explicitly mapping missing values (represented as .) to a suitably high number

```
==press <return> to continue==.
```

```
suppression toggled on for subsequent commandsTotal
INFO:acro:get_summary(): review; threshold: 4 cells suppressed; p-ratio: 18 cells suppressed; nk-rule: 18 cells suppressed;
INFO:acro:outcome_df:
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
> ----------------------------------|
                       mean                                                                                  |mode_aggfunc
>                                   |
recommend              not_recom priority recommend                      spec_prior very_recom          Total |not_recom    priority recommend                      sp
> ec_prior very_recom          Total|
parents     finance                                                                                           |
>                                   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
> ----------------------------------|
great_pret  convenient  ok        ok                  p-ratio; nk-rule;   ok         p-ratio; nk-rule;   ok   | ok           ok                  p-ratio; nk-rule;   o
> k         p-ratio; nk-rule;   ok  |
            inconv      ok        ok                  p-ratio; nk-rule;   ok         p-ratio; nk-rule;   ok   | ok           ok                  p-ratio; nk-rule;   o
> k         p-ratio; nk-rule;   ok  |
pretentious convenient  ok        ok                  p-ratio; nk-rule;   ok                         ok  ok   | ok           ok                  p-ratio; nk-rule;   o
> k                         ok  ok  |
            inconv      ok        ok                  p-ratio; nk-rule;   ok                         ok  ok   | ok           ok                  p-ratio; nk-rule;   o
> k                         ok  ok  |
usual       convenient  ok        ok       threshold; p-ratio; nk-rule;   ok                         ok  ok   | ok           ok       threshold; p-ratio; nk-rule;   o
> k                         ok  ok  |
            inconv      ok        ok                  p-ratio; nk-rule;   ok                         ok  ok   | ok           ok                  p-ratio; nk-rule;   o
> k                         ok  ok  |
Total                   ok        ok       threshold; p-ratio; nk-rule;   ok                         ok  ok   | ok           ok       threshold; p-ratio; nk-rule;   o
> k                         ok  ok  |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
> ----------------------------------|

INFO:acro:records:add(): output_2
INFO:acro:records:exception request was added to output_2
Total
-----------------------------------------------------------------------------------------------------------------------------|
                       mean                                                   |mode_aggfunc                                  |
recommend              not_recom priority  spec_prior very_recom Total     |not_recom    priority spec_prior very_recom Total|
parents     finance                                                           |                                              |
-----------------------------------------------------------------------------------------------------------------------------|
great_pret  convenient  3.170833  2.798828  3.351293        NaN   3.160185 | 3.0         1.0      3.0        NaN       3   |
            inconv      3.116667  2.317919  3.380256        NaN   3.122222 | 3.0         1.0      3.0        NaN       1   |
pretentious convenient  3.158333  3.085938  3.250000   2.625000   3.135648 | 3.0         1.0      3.0        1.0       1   |
            inconv      3.123611  3.051676  3.364706   1.363636   3.139815 | 1.0         1.0      3.0        1.0       1   |
usual       convenient  3.105556  3.110996  3.311047   2.623077   3.111677 | 3.0         1.0      3.0        1.0       2   |
            inconv      3.130556  3.076042  3.369565   1.363636   3.098148 | 3.0         2.0      3.0        1.0       3   |
Total                   3.134259  2.986873  3.345203   2.201220   3.127952 | 1.0         1.0      3.0        1.0       3   |
-----------------------------------------------------------------------------------------------------------------------------|
```

==press <return> to continue==.

D What other sorts of analysis does ACRO currently support?
We are continually adding support for more types of analysis as users prioritise them.

ACRO currently supports:
- **Tables** via **acro table** ... as described above.
  - supported statistics are: *mean, median, sum, std, count, mode*.
- **Regression** via: **acro regress..., acro logit** ... and **acro probit** ... using standard Stata syntax

We are aware that
- the python and R 'front-ends' to acro support more types of analysis,
- that some Stata users would like acro to support weighted analysis.

We welcome feedback, and suggestions for the next prioritise the development team.
Offers to contribute code are especially welcome!
There is more help on how to use all of these available from the *acro cheat sheet* or (if have have internet access) from www.sacro-tools.org

==press <return> to continue==.

E ACRO functionality to let users manage their outputs

As explained above, you need to create an *acro session* whenever your code is run.

After that, every time you run an acro *query* command both the output and the risk assessment are saved as part of the acro session.

But we recognise that:
- You may not want to request release of all your outputs - for example, the first table we produced above.
- It is good practice to provide a more informative name than just *output_n* for the .csv files that acro produces
- It helps the output checker if you provide some comments saying what the outputs are.
- You might want to add more things to the bundles of files you want to take out, such as:
  - outputs from analyses that acro doesn't currently support
  - your code itself (which many journals want)
  - maybe a version of your paper in pdf/word format etc.

Therefore acro provides the following commands for 'session management'

1 Listing the current contents of an ACRO session
This output is not beautiful (there's a GUI come soon) but should let you identify outputs you want to rename,comment on, or delete
acro print_outputs

==press <return> to continue==.

```
uid: output_0
status: fail
type: table
properties: {'method': 'crosstab'}
sdc: {'summary': {'suppressed': False, 'negative': 0, 'missing': 0, 'threshold': 5, 'p-ratio': 0, 'nk-rule': 0, 'all-values-are-same': 0}, 'cells': {'negative': [], '
```

```
> missing': [], 'threshold': [[2, 0], [2, 1], [2, 2], [2, 3], [4, 0]], 'p-ratio': [], 'nk-rule': [], 'all-values-are-same': []}}
command: safe_output = stata_config.stata_acro.crosstab(
summary: fail; threshold: 5 cells may need suppressing;
outcome: parents      great_pret  pretentious        usual        Total
recommend
not_recom            ok           ok           ok           ok
priority             ok           ok           ok           ok
recommend   threshold;   threshold;   threshold;   threshold;
spec_prior           ok           ok           ok           ok
very_recom  threshold;            ok           ok           ok
Total                ok           ok           ok           ok
output: [parents     great_pret  pretentious  usual  Total
recommend
not_recom         1440         1440   1440   4320
priority           858         1484   1924   4266
recommend            0            0      2      2
spec_prior        2022         1264    758   4044
very_recom           0          132    196    328
Total             4320         4320   4320  12960]
timestamp: 2026-02-11T15:59:32.691524
comments: []
exception:

uid: output_1
status: review
type: table
properties: {'method': 'crosstab'}
sdc: {'summary': {'suppressed': True, 'negative': 0, 'missing': 0, 'threshold': 5, 'p-ratio': 0, 'nk-rule': 0, 'all-values-are-same': 0}, 'cells': {'negative': [], 'm
> issing': [], 'threshold': [[2, 0], [2, 1], [2, 2], [2, 3], [4, 0]], 'p-ratio': [], 'nk-rule': [], 'all-values-are-same': []}}
command: safe_output = stata_config.stata_acro.crosstab(
summary: review; threshold: 5 cells suppressed;
outcome: parents      great_pret  pretentious        usual        Total
recommend
not_recom            ok           ok           ok           ok
priority             ok           ok           ok           ok
recommend   threshold;   threshold;   threshold;   threshold;
spec_prior           ok           ok           ok           ok
very_recom  threshold;            ok           ok           ok
Total                ok           ok           ok           ok
output: [parents     great_pret  pretentious  usual  Total
recommend
not_recom       1440.0         1440   1440   4320
priority         858.0         1484   1924   4266
spec_prior      2022.0         1264    758   4044
very_recom         NaN          132    196    328
Total           4320.0         4320   4318  12958]
timestamp: 2026-02-11T15:59:34.589461
comments: []
exception: Suppression automatically applied where needed

uid: output_2
status: review
type: table
properties: {'method': 'crosstab'}
sdc: {'summary': {'suppressed': True, 'negative': 0, 'missing': 0, 'threshold': 4, 'p-ratio': 18, 'nk-rule': 18, 'all-values-are-same': 0}, 'cells': {'negative': [],
> 'missing': [], 'threshold': [[4, 2], [4, 8], [6, 2], [6, 8]], 'p-ratio': [[0, 2], [0, 4], [0, 8], [0, 10], [1, 2], [1, 4], [1, 8], [1, 10], [2, 2], [2, 8], [3, 2],
> [3, 8], [4, 2], [4, 8], [5, 2], [5, 8], [6, 2], [6, 8]], 'nk-rule': [[0, 2], [0, 4], [0, 8], [0, 10], [1, 2], [1, 4], [1, 8], [1, 10], [2, 2], [2, 8], [3, 2], [3, 8
> ], [4, 2], [4, 8], [5, 2], [5, 8], [6, 2], [6, 8]], 'all-values-are-same': []}}
command: safe_output = stata_config.stata_acro.crosstab(
summary: review; threshold: 4 cells suppressed; p-ratio: 18 cells suppressed; nk-rule: 18 cells suppressed;
outcome:                             mean           ...        mode_aggfunc
recommend              not_recom priority  ...          very_recom Total
parents     finance                        ...
great_pret  convenient        ok       ok  ...  p-ratio; nk-rule;     ok
            inconv            ok       ok  ...  p-ratio; nk-rule;     ok
pretentious convenient        ok       ok  ...                 ok    ok
            inconv            ok       ok  ...                 ok    ok
usual       convenient        ok       ok  ...                 ok    ok
            inconv            ok       ok  ...                 ok    ok
Total                         ok       ok  ...                 ok    ok

[7 rows x 12 columns]
output: [                            mean            ... mode_aggfunc
recommend              not_recom  priority  ...   very_recom Total
parents     finance                         ...
great_pret  convenient  3.170833  2.798828  ...          NaN     3
            inconv      3.116667  2.317919  ...          NaN     1
pretentious convenient  3.158333  3.085938  ...          1.0     1
            inconv      3.123611  3.051676  ...          1.0     1
usual       convenient  3.105556  3.110996  ...          1.0     2
            inconv      3.130556  3.076042  ...          1.0     3
Total                   3.134259  2.986873  ...          1.0     3

[7 rows x 10 columns]]
timestamp: 2026-02-11T15:59:35.695347
comments: []
exception: Suppression automatically applied where needed



==press <return> to continue==.
```

2 Remove some ACRO outputs before finalising

At the start of this demo we made a disclosive output -it's the first one with status *fail*.

We don't want to waste the output checker's time so lets remove it.

```
acro remove_output "output_0"
INFO:acro:records:remove(): output_0 removed

==press <return> to continue==.

 3 Rename ACRO outputs before finalising
 This is an example of renaming the outputs to provide  more descriptive names.
rename_output output_1 "crosstab_recommendation_vs_parents"
rename_output output_2 "mean_children_by_parents_finance_recommendation"
INFO:acro:records:rename_output(): output_1 renamed to crosstab_recommendation_vs_parents
output output_1 renamed to crosstab_recommendation_vs_parents.
INFO:acro:records:rename_output(): output_2 renamed to mean_children_by_parents_finance_recommendation
output output_2 renamed to mean_children_by_parents_finance_recommendation.

==press <return> to continue==.

 4 Add a comment to output
 This is an example of adding a comment to outputs.
 It can be used to provide a description or to pass additional information to the TRE staff.
 They will see it alongside your file in the output checking viewer - rather than having it in an email somewhere.

acro add_comments "mean_children_by_parents_finance_recommendation" "too few cases of recommend to report"
INFO:acro:records:a comment was added to mean_children_by_parents_finance_recommendation
Comments added to output mean_children_by_parents_finance_recommendation.

==press <return> to continue==.

 5. Request an exception
 An example of providing a reason why an exception should be made
e.g.: acro add_exception "output_n" "This is evidence of systematic bias"

==press <return> to continue==.

 6 Adding a custom output.
 As mentioned above you might want to request release of all sorts of things
 - including your code,
 - or outputs from analyses *acro* doesn't support (yet)
'
 In ACRO we can add a file to our session with a comment describing what it is.
acro custom_output "acro_demo_2026.do" "this is the code that produced this session"
INFO:acro:records:add_custom(): output_3
file acro_demo_2026.do with comment This is the code that produced this session added to session.
==press <return> to continue==.

 F Finishing your session and producing a folder of files to release.
 This is an example of the function finalise() which the users must call at the end of each session.
 - It takes each output and saves it to a CSV file (or the original file type for custom outputs)
 - It also saves the SDC analysis for each output to a json file.
 - It adds checksums for everything - so we know they've not been edited.
 - It puts them all in a folder with the name you supply.

 ACRO will not overwrite previous sessions

 So every time you call finalise on a session you need to either:
   - delete the previous folder (by adding Stata commands to remove), or
   - provide a new folder name manually, or
 - create a const (e.g. macro) programmatically to hold the folder name

The code snippet below shows how to add the current time and date to a folder name.
Please remember the comments above about substituting symbols for dollar backtick_ and _tick
local suffix = subinstr("dollarS_DATE", " ", "_", .) + "_" + subinstr("dollarS_TIME",":","_",.)
local myfoldername= "my_acro_outputs_v1_" + "backtick_suffix_tick"
acro finalise "backtick_myfoldername_tick"
INFO:acro:records:outputs written to: my_acro_outputs_v1_11_Feb_2026_15_59_53
outputs and stata_outputs.json written

14 .
15 .
16 .
17 .
end of do-file

18 . log close
      name:  <unnamed>
       log:  /Users/j4-smith/GitHub/AI-SDC/ACRO/stata/acro_demo_2026.smcl
  log type:  smcl
 closed on:  11 Feb 2026, 16:00:03
```