The Stata Journal (*yyyy*) *vv*, Number *ii*, pp. 1–16

autolabel: Automating variable and value labeling in Stata

Jeffrey Clark
Stockholm University
Stockholm, Sweden
jeffrey.clark@su.se

Jie Wen
Stockholm School of Economics
Stockholm, Sweden
jie.wen@hhs.se

Abstract. Register-based datasets from national statistical agencies arrive with cryptic variable names and unlabeled coded values, requiring researchers to manually extract and apply metadata from external documentation. This tedious, error-prone process hinders reproducibility. The `autolabel` command automates variable and value labeling by matching dataset variables to centralized, multilingual CSV metadata repositories. The command introduces a *deferred execution pattern* using temporary do-files inside preserved sessions to safely separate dataset inspection from transformation. The domain-agnostic design operates at institutional scale, enabling organizations to programmatically compile metadata from their systems into standardized CSV repositories. Multilingual metadata enables instant language switching without modifying datasets, transforming manual labeling workflows into automated, reproducible processes operating at organizational and cross-national scale.

Keywords: autolabel, metadata automation, administrative data, reproducible research, multilingual metadata, deferred execution, Stata command

1 Introduction

Register-based datasets are essential to empirical research in economics, sociology, public health, and beyond. National statistical agencies routinely collect rich administrative data covering large populations over time, enabling powerful longitudinal and cross-sectional analyses unavailable with survey data. Yet these datasets often arrive in a form that is technically complete but practically inconvenient. Variables lack descriptive labels, coded values remain cryptic, and researchers must manually cross-reference external documentation to understand their data.

The issue is not the lack of metadata. National statistical agencies provide extensive documentation through publicly available registries, including codebooks, variable descriptions, and value label definitions. Rather, the challenge lies in the disconnect between that documentation and the raw datasets themselves. Variables arrive cryptically coded; value codes remain unlabeled. For register datasets with hundreds of variables, researchers must manually extract metadata from external files, match them to dataset variables, and write labeling commands. This manual process is time-consuming and prone to inconsistencies and errors.

This challenge extends beyond individual projects: metadata typically remains avail-

© *yyyy* StataCorp LLC

able only in the national language, limiting international collaboration. Automated, multilingual metadata application addresses this barrier directly.

In this paper, we introduce `autolabel`, a Stata command that automates the application of variable and value labels from centralized metadata repositories. Developed within the RegiStream project (Clark and Wen 2024–), `autolabel` eliminates manual labeling by applying standardized metadata through a single command. For researchers working with administrative microdata, this automation reduces interpretation errors, accelerates exploration, and enables international collaboration through multilingual metadata support.

The command makes three core contributions:

Technical pattern: To safely apply metadata when inspection requires destructive dataset operations, `autolabel` introduces a *deferred execution pattern*: using temporary do-files inside preserved sessions to separate inspection logic from application logic. This pattern generalizes beyond labeling to any workflow requiring post-inspection transformations.

Extensibility: The domain-agnostic design allows organizations to define custom domains following the documented metadata schema.

Multilingual support: External metadata enables instant language switching across entire projects without dataset modification, facilitating international collaboration and multilingual reporting.

The remainder of this article is organized as follows. Section 2 positions `autolabel` within existing metadata tools and discusses related work. Section 3 introduces command syntax and features. Section 4 presents the deferred execution implementation pattern. Section 5 demonstrates applied examples using register data. Section 6 describes the metadata file structure and domain specification. Section 7 concludes with discussion of contributions and future extensions.

2 Background and Related Work

2.1 The metadata application gap

National statistical agencies provide high-quality register data with extensive documentation through publicly available metadata registries. This metadata includes codebooks, variable descriptions, and value label mappings, often available in structured formats (CSV, XML). However, documentation remains distributed separately from datasets, which are delivered without variable or value labels.

While Stata provides robust labeling commands (`label variable`, `label define`, `label values`), applying metadata requires researchers to manually extract information from external files, match entries to dataset variables, and write labeling code. For large register datasets with hundreds of variables (many using complex classification schemes for industries, occupations, education levels, or geographic units), this manual

process becomes time-consuming and error-prone. Subtle labeling inconsistencies across research projects impede reproducibility, particularly in collaborative settings where multiple researchers work with the same data sources.

2.2 Existing approaches to metadata management

Several Stata commands address metadata manipulation and documentation. Klein (2019) introduced `elabel` for enhanced manipulation of existing labels within datasets. Weesie (2005) developed `mlanguage` for managing multilingual labels within datasets, requiring all translations to be embedded in the data file itself.

More recently, Iglésias et al. (2025) developed `metaxl`, which extracts metadata from datasets to Excel files, enabling harmonization across data extractions through Excel-based editing workflows. This approach is particularly valuable for managing longitudinal administrative data where variable definitions evolve over time, allowing researchers to identify inconsistencies, merge redundant value labels, and reapply harmonized metadata. `metaxl` addresses a different challenge than `autolabel`: building and harmonizing metadata for specific datasets rather than applying pre-existing metadata from centralized repositories.

Table 1 summarizes how `autolabel` complements existing Stata metadata tools.

Table 1: Comparison of Stata metadata management tools

Feature	**autolabel**	**metaxl**	**elabel/mlanguage**
Primary goal	Apply labels from external metadata repositories	Extract, audit, and harmonize metadata across datasets	Manipulate or refine existing labels within a dataset
Source of truth	External CSV repositories (outside .dta)	Excel metafiles linked to datasets	Dataset-embedded label definitions
Scale	Institutional/organizational (thousands of variables)	Project-level (specific datasets)	Dataset-level (existing labels)
Multilingual	Yes (lang() switching)	Single language	mlanguage: embedded translations
Safety pattern	`preserve` → write temp do → `restore` → execute	Standard in-memory operations (auto-backup file)	In-place modification
Typical use case	Applying standardized register metadata from statistical agencies at scale	Harmonizing longitudinal or multi-wave survey extracts	Label editing and translation refinement

Each tool occupies a distinct role in the metadata workflow. `metaxl` emphasizes extraction and harmonization across datasets using Excel metafiles; `autolabel` automates deterministic application of external CSV-based metadata; `elabel` and `mlanguage` handle label editing and translations within a dataset. The approaches are complementary rather than overlapping.

2.3 What makes autolabel different

`autolabel` is, to our knowledge, the first Stata command to fully automate label application directly from external metadata repositories. Unlike tools that manipulate existing labels within datasets (Klein (2019), Weesie (2005)) or extract metadata for harmonization (Iglésias et al. (2025)), `autolabel` applies centrally-maintained metadata to any dataset containing matching variables. Unlike `mlanguage` (which embeds translations within .dta files) or `metaxl` (which uses Excel as an intermediary), `autolabel` treats CSV repositories external to datasets as the source of truth. This separation ensures metadata remains version-controlled, auditable, and reusable across projects without modifying data files.

`metaxl` helps researchers build and harmonize metadata for specific datasets through Excel-based workflows, valuable when variable definitions evolve over time or across institutional extractions. `autolabel` applies pre-existing metadata from centralized repositories, serving a complementary role. Where `metaxl` focuses on metadata harmonization, `autolabel` focuses on metadata application.

The command assumes standardized variable naming and structured metadata, appropriate for register data from national statistical agencies. It is designed for metadata application at organizational scale, not data cleaning or variable harmonization; those tasks belong earlier in the workflow.

This approach aligns with FAIR principles for research data (Wilkinson et al. 2016): centralized repositories make metadata Findable, multilingual support makes it Accessible, standardized schemas ensure Interoperability, and domain-agnostic design enables Reusability.

3 The `autolabel` Command

`autolabel` automates the application of variable and value labels to register-based datasets, building directly on Stata's built-in labeling system. The command supports three modes: `variables`, `values`, and `lookup`.

3.1 Command structure

The general syntax for `autolabel` is as follows:

```
autolabel variables [varlist], domain(string) lang(string)
    [exclude(varlist) suffix(string)]

autolabel values [varlist], domain(string) lang(string)
    [exclude(varlist) suffix(string)]

autolabel lookup varlist, domain(string) lang(string)
```

The `domain()` and `lang()` options are required; the command returns an error if either is omitted. The command is designed to work with any domain; users can create custom domains following the documented structure (Section 6). Metadata repositories can be hosted centrally or locally. The RegiStream project (Clark and Wen 2024–) maintains a public repository including the Statistics Sweden (SCB) domain. The full SCB domain contains nearly 28,000 variable entries covering Swedish administrative registers and provides language options `swe` (Swedish) and `eng` (English). For demonstration purposes, this submission includes a representative 10-variable subset. If missing metadata files are requested, `autolabel` prompts users to download them from the RegiStream repository.

3.2 Modes of operation

The command supports three modes, each requiring both `domain()` and `lang()` options. The `variables` mode applies variable labels to specified variables in the current dataset. The `values` mode applies value labels to categorical variables. The `lookup` mode displays metadata without modifying the dataset, showing available variable labels, value label definitions, and descriptions.

If no variable list is specified (in `variables` or `values` mode), `autolabel` attempts to apply labels to all variables in the dataset that are found in the metadata.

3.3 Options

`domain(<string>)` Specifies the metadata domain (required). Domains can be user-created or accessed from centralized repositories; see Section 6 for domain structure.

`lang(<string>)` Specifies the label language (required). Available languages depend on the CSV metadata files provided for each domain.

`exclude(<varlist>)` Excludes specified variables from labeling.

`suffix(<string>)` Stores labeled variables as new variables with the given suffix, preserving the original dataset.

3.4 Scope and intended use cases

`autolabel` is designed for datasets with standardized variable naming conventions, typical of register data from national statistical agencies. The command performs case-insensitive string matching between dataset variables and metadata entries.

The `exclude()` option allows researchers to label all matching variables except a specified subset. The `suffix()` option enables non-destructive exploration, creating labeled copies of variables for inspection before committing to changes.

The command's case-insensitive matching ensures deterministic, auditable metadata

application: given the same metadata repository and dataset variable names, `autolabel` produces identical results across platforms and time.

3.5 Metadata files and directory structure

`autolabel` reads metadata from CSV files cached locally in `~/.registream/` (macOS/Linux) or `C:/Users/.../AppData/Local/registream/` (Windows). Missing files trigger a download prompt using Stata's `copy` command.

In secure or restricted computing environments where direct file downloads are not permitted or where default directory locations are not accessible, users can manually transfer metadata files and specify a custom path via `$registream_dir`. Metadata files follow standard folder conventions, stored as `~/.registream/autolabel_keys/scb_variables_eng.csv` and `~/.registream/autolabel_keys/scb_value_labels_eng.csv`. Metadata updates in restricted environments follow institutional procedures: authorized personnel transfer updated files to the system, with version tracking maintained through institutional version management systems.

`autolabel` automatically detects the operating system and resolves appropriate directories on macOS, Windows, and Linux.

Note on submission materials: This submission contains the `autolabel` module from RegiStream v2.0.0 using schema 1.0.

4 Implementation: A reproducibility pattern for safe labeling

Beyond its specific application, `autolabel` introduces a generalizable pattern for safe post-inspection transformations in Stata workflows. This *deferred execution pattern* uses temporary do-files inside a preserved session to separate inspection logic from application logic. The command must inspect the dataset to determine which variables to label and what transformations to apply, but these inspection steps themselves would modify the dataset. The solution: write labeling commands to a temporary do-file during inspection, restore the original dataset, then execute the commands. This ensures full reproducibility and auditability. The pattern emerged after alternative approaches (storing locals/globals during inspection, looping post-restore, direct encoding) proved brittle or opaque in production use.

The pattern applies beyond metadata labeling to any workflow requiring conditional command generation, including survey harmonization, quality control routines generating conditional recoding commands, or transformations applied after multi-dataset inspection.

4.1 The problem: safe labeling after inspection

Unlike tools such as `metaxl`, which reapply metadata to datasets already sharing an identical structure, `autolabel` must reconcile external metadata with datasets that may differ in variable type, naming, or coding. In practice, statistical agencies may deliver register data in varying formats: metadata might specify numeric types but agencies deliver strings, type specifications may be inconsistent across extractions, or coding schemes may evolve over time. Variables often require encoding to numeric form before value labels can be applied. Metadata repositories often include multiple versions of the same value-label groups reflecting updates across years or agencies.

To apply labels correctly, `autolabel` must inspect the dataset to determine which variables match available metadata, what transformations are required for each variable, and which metadata versions apply. However, inspection steps (merging metadata, subsetting variables, encoding transformations) modify the dataset structurally or in content. Applying labels during inspection risks overwriting user data, errors from altered variables, and reduced reproducibility from hard-coded intermediate steps.

This creates a fundamental tension: labeling must be *based on* metadata inspection but *applied after* returning to the original dataset.

4.2 The solution: preserve, inspect, write, restore, execute

To solve this, `autolabel` adopts a deferred execution strategy consisting of five steps:

1. `preserve` the dataset
2. Generate temporary metadata-based label commands (e.g., using merges and summaries)
3. Write those commands to a temporary `.do` file using `file write`
4. `restore` the original dataset (before any inspection or changes)
5. `do` the temporary file to apply all labeling commands

The dataset is never altered during metadata inspection; all changes are applied clearly and reproducibly at once. The pattern generalizes to any workflow requiring conditional command generation based on dataset inspection: survey harmonization, quality control routines, or transformations determined by multi-dataset analysis.

4.3 Code example: variable labeling logic

The following skeleton illustrates the approach for applying variable labels. Inside a preserved session, the command merges metadata, constructs label strings, and writes commands to a temporary do-file:

```
preserve
    keep ‘varlist’
    merge 1:1 variable using "‘var_metadata’", keep(match) nogen
    gen label_text = variable_desc + " (" + unit + ")" if unit != ""

    tempfile cmdfile
    file open fh using ‘cmdfile’, write replace
    forval i = 1/‘=_N’ {
        local vname = variable[‘i’]
        local vlabel = label_text[‘i’]
        file write fh ‘"label variable ‘vname’ "‘vlabel’""’ _n
    }
    file close fh
restore
do ‘cmdfile’
```

The result is a clean application of labels, based on metadata inspection, without touching the dataset until everything is known and ready. The same pattern applies to value labels: the temporary do-file builds value label definitions and encoding commands line-by-line during inspection, then applies them after restoration.

4.4 Why this matters

This pattern offers several advantages:

- **Safety**: No labels applied during inspection, eliminating corruption risk
- **Modularity**: Label generation logic fully separated from application
- **Reproducibility**: Transformation is deterministic and fully traceable through metadata files
- **Auditability**: Applied labels immediately inspectable via standard Stata commands (`describe`, `label list`); metadata sources remain transparent in CSV format
- **Automation**: Manual value labeling becomes impractical with large categorical sets (for example, hundreds of municipalities or detailed occupation codes); autolabel standardizes this process through metadata repositories, eliminating tedious manual label definition
- **Scalability**: The pattern's deterministic, file-based approach ensures reproducibility and auditability scale to thousands of variables; at institutional scale, manual processes become infeasible, making automation a methodological requirement rather than a convenience. The same method applies to both variable and value labels and generalizes to other post-inspection actions

This deferred execution pattern (a structured approach for safe post-inspection transformations) represents a methodological contribution to reproducible Stata programming, demonstrating how complex data transformations can be separated from

their application to ensure safety and reproducibility in sensitive workflows. Scalability is not merely a performance feature but a methodological necessity: reproducibility, auditability, and automation become feasible only when metadata application scales to organizational levels.

4.5 Error handling and robustness

Variables not found in the metadata are silently skipped, allowing researchers to apply available metadata without requiring completeness. The command gracefully handles missing files, format inconsistencies, and character encoding issues without compromising dataset integrity.

While the underlying tools (`preserve`, `restore`, `file write`) are familiar to experienced Stata users, their structured composition into a general-purpose reproducibility pattern has not, to our knowledge, been formally documented. The novelty lies in how they are composed to solve reproducibility and safety challenges common in metadata-rich workflows.

5 Applied examples

The following examples demonstrate `autolabel` functionality using real SCB variables and are based on the demonstration scripts provided with this submission. All examples are fully reproducible using the included data and scripts in the supplementary materials.

A key advantage of `autolabel` in exploratory work is the ability to label entire datasets containing hundreds of variables in a single command. Once variable labels are applied, Stata's built-in variable search (in the Variables window) automatically searches both variable names and variable labels. This makes register data immediately explorable: researchers unfamiliar with the registry variables can search for "pension" or "income" to locate relevant variables, rather than deciphering cryptic codes like `dispink04` or `aldtjptyp`. Even those familiar with the registry benefit from this searchability, as variable names are often abbreviated or coded in ways that obscure their meaning.

5.1 Variable labeling transformation

Before applying `autolabel`, the dataset contains only cryptic variable names:

```
. describe astsni2007 dispink04 kon

              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------
astsni2007      str5    %9s
dispink04       int     %9.0g
kon             byte    %9.0g
```

We can apply variable labels to all matching variables in the dataset using a single command (omitting the varlist applies labels to all variables found in the metadata):

```
. autolabel variables, domain(scb) lang(eng)
```

After running `autolabel variables`, all matching variables become immediately interpretable. The labels are now searchable in Stata's Variables window, enabling researchers to quickly locate variables by content (e.g., searching "income" or "gender") rather than memorizing cryptic variable names:

```
. describe astsni2007 dispink04 kon

              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------
astsni2007      str5    %9s         Industry sector for statistics, main
dispink04       int     %9.0g       Sub-component of disposable income, 2004 definition
kon             byte    %9.0g       Gender
```

5.2 Value labeling transformation

Beyond variable labels, `autolabel` also transforms coded values into readable categories. Before applying value labels, categorical variables display only numeric codes:

```
. tab astsni2007 in 1/100, sort

  Branch of |
   industry |
        for |
statistics, |
       main |      Freq.     Percent        Cum.
------------+-----------------------------------
      85201 |         17       17.00       17.00
      56100 |         13       13.00       30.00
      85100 |         11       11.00       41.00
      88101 |         11       11.00       52.00
      49410 |         10       10.00       62.00
      87301 |          9        9.00       71.00
      00000 |          8        8.00       79.00
      41200 |          7        7.00       86.00
      78200 |          5        5.00       91.00
      86102 |          5        5.00       96.00
      47112 |          4        4.00      100.00
------------+-----------------------------------
      Total |        100      100.00
```

We can apply value labels to all matching variables with another single command:

```
. autolabel values, domain(scb) lang(eng)
```

After applying value labels, all matching coded variables become immediately readable (note that Stata's `tab` command truncates long labels in display):

```
. tab astsni2007 in 1/100, sort

   Industry sector for statistics, main |      Freq.     Percent        Cum.
```

```
----------------------------------------+----------------------------------
    Primary schools and nursery schools |         17       17.00       17.00
             Food and beverage services |         13       13.00       30.00
                           Pre-schools |         11       11.00       41.00
Home care services, day-care centres an |         11       11.00       52.00
  Vehicles for the transport of persons |         10       10.00       62.00
Service houses, serviced apartments for |          9        9.00       71.00
                             Unknown |          8        8.00       79.00
Residential and other building contract |          7        7.00       86.00
                   Employment agencies |          5        5.00       91.00
End-of-life clinics for somatic health  |          5        5.00       96.00
  Grocery stores with a wide assortment |          4        4.00      100.00
----------------------------------------+----------------------------------
                                  Total |        100      100.00
```

This labeling process required no manual mapping. `autolabel` parses metadata and emits native `label define`/`label values` statements; work scales with metadata size and is independent of the number of observations, so overhead is negligible (i.e., interactive time). The primary value is eliminating the tedious process of manually cross-referencing documentation, typing label definitions, and debugging syntax for each variable and categorical code. For register datasets with hundreds of variables, this automation meaningfully reduces manual effort and potential for transcription errors. An additional advantage is instant language switching: by changing the `lang()` parameter, researchers can produce outputs in different languages without modifying data or analysis code, valuable for international collaboration or multilingual reporting.

The `exclude()` and `suffix()` options provide additional flexibility: `exclude()` skips specific variables when labeling all others, while `suffix()` creates labeled copies for non-destructive exploration before committing to permanent changes.

5.3 Looking up metadata without modifying the dataset

The `lookup` mode provides metadata inspection without modifying the dataset. This operates entirely offline by reading directly from local CSV metadata files, making it useful for exploring variable definitions and value label mappings before applying labels.

```
. autolabel lookup sun2000niva carb aldtjptyp, domain(scb) lang(eng)
Looking up variables in domain scb using language eng
------------------------------------------------------------------------------
| VARIABLE:     sun2000niva
| LABEL:        Education level (3 positions)
| DEFINITION:   The level of education according to the Swedish Educational
|               Nomenclature (SUN 2000). Full level code, three positions.
| VALUE LABELS: 000: Other and unspecified preschool education
|               001: Preschool
|               002: Preschool class
|               100: Other and unspecified upper secondary education
|               102: Primary school education, grades 1-6
|               106: Public school education
|               200: Other and unspecified pre-secondary education
|               204: Secondary school education
|               (and 46 more labels)
------------------------------------------------------------------------------
```

```
-------------------------------------------------------------------------------
| VARIABLE:     carb
| LABEL:        Labor income
| DEFINITION:   The sum of salary, business income, sickness benefit, parental
|               benefit and compensation in connection with military service.
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
| VARIABLE:     aldtjptyp
| LABEL:        Retirement/service pension, existence of
| DEFINITION:   Indicates if the person received income during the year due to
|               retirement or occupational pension.
| VALUE LABELS: 0: No
|               1: Yes
-------------------------------------------------------------------------------
```

6 Metadata file structure and domain specification

Each domain in RegiStream (e.g., `scb`, `health_survey`, etc.) must contain two CSV files: *domain_name*`_variables_`*lang*`.csv` contains metadata for each variable, including name, description, and value label group (if applicable), while *domain_name*`_value_labels_`*lang*`.csv` contains value labels for coded variables.

Here is a sample from `scb_variables_eng.csv`:

variable_name	variable_label	variable_definition	...	value_label_id
kommun	Municipality	Regional division into municipalities.	...	3
kon	Gender	The sex of the person.	...	1
lopnr	Serial number	The serial number.	...	

The `value_label_id` column links variables to their value label definitions. For example, `kommun` has `value_label_id=3`, indicating it uses value label group 3. Variables without value labels (like `lopnr`) have this field empty.

And the corresponding rows from `scb_value_labels_eng.csv`:

value_label_id	variable_name	value_labels_stata	value_labels_json
3	kommun	"0114" "Upplands Väsby" "0115" "Vallentuna" ...	{...}
1	kon	"K" "Female" "M" "Male" ...	{...}

The `value_labels_stata` column contains value labels in Stata's native format (alternating codes and labels), while `value_labels_json` stores the same mappings in JSON format for cross-platform use. The `value_label_id` serves as the linking key: when `autolabel` processes `kommun`, it looks up `value_label_id=3` in the value labels file to retrieve the appropriate municipality codes and labels.

The RegiStream project provides metadata infrastructure across multiple programming environments (Stata, Python, R), requiring a language-agnostic interchange format. CSV files are human-readable, version-controllable via Git, and impose no propri-

etary software dependencies. This design contrasts with approaches that embed labels inside .dta files (e.g., `mlanguage`), which tie metadata to individual datasets and prevent reuse across projects.

6.1 Programmatic generation for institutional metadata

These CSV files are designed to be programmatically compiled from institutional metadata systems, not manually created. Statistical agencies and research organizations typically maintain metadata in databases or documentation systems; `autolabel`'s CSV format enables automated export from these systems. For example, the SCB domain metadata was compiled programmatically from Statistics Sweden's documentation. This approach scales to thousands of variables and differs fundamentally from tools like `metaxl`, which facilitate manual metadata entry and harmonization via Excel interfaces. While `metaxl` excels at researcher-level metadata curation for specific projects, `autolabel`'s CSV infrastructure targets organizational-scale metadata distribution, where metadata is generated once by institutions and reused across projects. Researchers can inspect metadata provenance, track changes over time via version control, and contribute corrections upstream without touching analysis datasets.

The metadata files support offline use in secure environments. Researchers can copy, modify, and create their own metadata files using the same structure.

The domain structure is fully general, extending beyond register data to any metadata-rich domain: survey data, clinical trials, or administrative records from any institution.

7 Conclusion

This paper introduced `autolabel`, a Stata command that automates the application of variable and value labels from centralized metadata repositories. The command establishes a foundation for treating metadata as shared infrastructure rather than per-dataset documentation.

7.1 Contributions

The command makes three integrated contributions to Stata metadata workflows:

Technical contribution: The deferred execution pattern (Section 4) solves a fundamental challenge in metadata workflows: how to safely apply labels when inspection requires dataset modification. This structured composition of `preserve`, `restore`, and `file write` generalizes beyond metadata labeling to any workflow requiring inspection-based command generation, including survey harmonization and quality control routines.

Architectural contribution: The domain-agnostic metadata schema with CSV-based repositories (Section 6) operates at institutional scale. The SCB domain demon-

strates this with nearly 28,000 variable entries programmatically compiled from Statistics Sweden's documentation systems. Organizations can similarly export their metadata into standardized CSV repositories, treating metadata as shared infrastructure rather than per-dataset documentation. This design enables instant language switching across entire projects without dataset modification.

Practical contribution: Section 5 demonstrates that external metadata can be applied automatically at scale to register data. With a single command, researchers can label entire datasets containing hundreds of variables, making them immediately explorable through Stata's variable search. The command eliminates tedious manual labeling for large categorical sets like municipalities or occupation codes, transforming metadata application from a manual, error-prone process into automated, reproducible infrastructure. Scalability enables the methodological benefits of reproducibility, auditability, and automation to extend from individual projects to organizational and cross-national research infrastructure.

Together, these contributions establish the feasibility of centralized metadata repositories as infrastructure for reproducible research with administrative microdata.

7.2 Future extensions

The metadata schema will continue to evolve and improve in response to institutional and cross-agency needs. Future versions could accommodate additional metadata dimensions including temporal variation (definitions changing across years), harmonization provenance (documenting resolution decisions when conflicting metadata exists), and validation metadata (confidence scores, quality indicators).

A natural convergence point exists with metadata harmonization workflows. Tools like `metaxl` extract and harmonize metadata across institutional data extractions. Future schema versions could standardize ingestion of harmonized outputs, creating an integrated pipeline: institutions harmonize metadata locally, export to standardized format, and centralized repositories serve the research community. This would transform one-time harmonization efforts into shared infrastructure.

7.3 Global perspective

The domain-agnostic architecture enables expansion to statistical agencies worldwide. National statistical offices that maintain register data could programmatically compile their metadata systems into the standardized CSV format, making their documentation instantly accessible to researchers. International organizations like the World Bank, OECD, Eurostat, and UN agencies could similarly export their metadata repositories, enabling researchers to apply standardized labels to cross-national datasets with a single command. This transforms scattered institutional documentation into shared metadata infrastructure operating at global scale.

Availability

Upon publication, `autolabel` will be available through the Stata Journal website and SSC archive. This is a standalone version containing core labeling functionality, derived from RegiStream v2.0.0. A fuller-featured version with additional infrastructure (usage logging, automatic updates, configuration management) is available as part of the RegiStream package. Additional domains can be created following Section 6.

8 References

Clark, J., and J. Wen. 2024–. RegiStream: Streamline Your Register Data Workflow. Available at https://registream.org.

Iglésias, G., P. Guimarães, and M. Silva. 2025. metaxl: A package of tools to handle metadata. *The Stata Journal* 25(2): 285–308.

Klein, D. 2019. Extensions to the label commands. *The Stata Journal* 19(4): 176–184.

Weesie, J. 2005. Multilingual datasets. *The Stata Journal* 5(2): 162–187.

Wilkinson, M. D., M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. G. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. C. 't Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, and B. Mons. 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data* 3(1): 160018.

About the authors

Jeffrey Clark is a PhD student in Economics at Stockholm University. He led the development of autolabel and maintains the broader RegiStream project, which provides open tools for register data and metadata infrastructure. His research focuses on development, political, and labor economics.

Jie Wen is a PhD student in Business Administration at Stockholm School of Economics. He co-founded the RegiStream project and co-developed autolabel. With a background in actuarial science, his research interests lie in accounting, auditing, and the use of microdata in empirical analysis.