## Introduction

In different SDMX agencies, the same semantic entity (e.g., frequency, currency) is described by multiple versions and variants of code lists. The project’s goal is to automatically:

* detect these discrepancies,
* compile “consolidated” code dictionaries in Linked Data (Turtle/RDF) format,
* eliminate duplicates, single (“local”) lists, and noisy values,
* quantitatively verify the quality of the resulting model.

## Project Stages

| **Stage** | **What the script does** | **Why this matters** |
| --- | --- | --- |
| **Loading and normalization** | parse\_codelist\_v3() reads the original SDMX XML-files and save\_codelist\_data() converts them into two tables — “codelist” and “codes” with metadata. | A unified tabular layer allows applying identical filtering and analysis rules regardless of agency and version. |
| **Technical analysis** | tech\_analisys() measures the size of the “heaviest” lists, finds frequent and “non-specific” codes. | Enables ranking lists by their contribution to the overall code space and identifying candidates for merging or noise removal. |
| **Filtering “noisy” CL** | In cl\_analysis() the following are excluded: • pre-defined single/group lists SINGLE\_CODELISTS and GROUP\_CODELISTS;  • all CLs where all codes are unique and do not intersect with others. | Consolidates only those lists that actually share common codes and can be merged into universal schemes; eliminates local directories. |
| **Semantic merging** | For each future scheme, get\_concept\_scheme\_str() is called, which creates a SKOS ConceptScheme and makes skos:exactMatch to base SDMX concepts. | Ensures consistency with the official SDMX model and provides an anchor point for external data. |
| **RDF generation** | generation\_ttl() produces Turtle files for each Scheme ID and a general code.ttl, aggregating codes from different agencies (parse\_save\_clanalyze\_func). | Results in a machine-readable, extensible, and reusable representation. |
| **Quality control** | check\_rdf\_quality() evaluates prefixes, labels, relations, variety of skos:notation, and outputs quality\_score10 / value\_score10 (analyze\_sdmx\_cl). | Provides an objective numeric criterion to automatically block weak models and repeat generation. |

## Principles Underlying the Project Implementation

1. **Automation with feedback:** Each step (parsing → filtering → generation → QA) is performed by a script and immediately verified quantitatively; the cycle can be run on any set of XMLs without manual intervention.
2. **Reuse of existing standards:** SKOS/SDMX URIs for exactMatch ensure that new semantic models align with existing ontologies, and code values retain their meaning during exchange.
3. **Minimization of duplicates:** Lists where 100% of codes are unique are discarded, and versions of the same scheme are merged; the result is a semantic model of consolidated code lists instead of dozens of variants.
4. **Measuring usefulness, not just “validity”:** In addition to syntactic correctness (missing\_prefixes, skos:inScheme), the value\_score10 metric (based on the number of unique triples) assesses how semantically “rich” the graph is (get\_funcsgen\_cl\_ttl).

## Advantages

* **Order-of-magnitude reduction in volume:** after filtering out “unique” lists, only ~20–30% of the original CLs remain, but they cover >95% of occurring codes.
* **High automatic QA:** files scoring above 9.5/10 on both metrics usually require no manual post-processing (quality\_checkgen\_cl\_ttl).
* **Easy updates:** a new XML version is added to the directory, and the pipeline automatically decides whether to update an existing scheme or create a new one.
* **Interoperability:** the presence of skos:exactMatch, rdfs:seeAlso, and NEW\_PREF\_CODE makes the dictionaries suitable for both SDMX messages and any Linked Data systems.

## How to Use the Results

1. Run cl\_analysis() → get filtered CSV files.
2. Pass them to sdmx\_codelist\_gen() → obtain a package of .ttl + a consolidated code.ttl.
3. Review QA reports; if scores are below the threshold, adjust settings (e.g., expand SINGLE\_CODELISTS) and repeat.

This cycle provides a sustainable, reproducible methodology: any set of code lists from different agencies can be transformed into high-quality, unified semantic models.