Using Generative AI to Calculate Price Indexes

Producer Prices Division, Statistics Canda

Authors: Xin Ha, Lizaveta Vasileuskaya and Shaoxiong Wang

Abstract

Producer Prices Division at Statistics Canada has modernized price index production through reproducible analytical pipelines built with open-source tools. However, building and maintaining these pipelines still involves significant repetitive coding. This paper presents a multi-agent workflow that uses generative AI to transform approved methodological requirements into validated R-based analytical pipelines. The multi-agent workflow consists of specification intake, code generation, and code validation stages, with human oversight embedded throughout. Results suggest that can reduce manual programming effort and improve scalability and efficiency while maintaining reproducibility, transparency, and methodological control.

1 Introduction

Price indexes are a core component of official economic statistics, and the systems used to produce them must support transparency, consistency, and reproducibility. In the Producer Prices Division (PPD) at Statistics Canada, these requirements have been central to a broader modernization effort aimed at improving how producer price indexes are built and maintained.

Before 2021, production systems were largely IT-driven and centrally managed. Although these systems were functional, they were also relatively rigid. Adapting them to new requirements was often slow and extending them to support new methods or workflows required considerable effort. As part of PPD modernization, the traditional workflow was gradually replaced with open-source, reproducible analytical pipelines built with tools such as R, git, and domain-specific packages including piar, gpindex, and sps (Ha & Martin 2025). The piar package has been especially important in PPD modernization because it supports price index aggregation in a way that is both flexible and transparent (Martin 2024). This shift enabled more transparent, efficient, and collaborative workflows and aligned with a broader move toward reproducible analytical pipelines in official statistics.

That modernization solved many important problems, but it also made another issue more visible. Even in a more flexible open-source environment, analysts across programs still perform similar implementation tasks repeatedly. Pipelines often need to be adapted, extended, or rebuilt by hand. The repeated translation of approved statistical methods into production-ready code can be time-consuming and resource-intensive.

This paper describes ongoing work to address that gap using generative AI. Our aim is not to use AI to decide statistical methods, nor to replace expert judgment. Instead, the goal is to use generative AI to accelerate the translation from approved statistical design to reproducible

production code. The current focus is on automatically generating R-based pipelines for price index calculation without having to build each one from scratch.

2 Background

A reproducible analytical pipeline is a structured sequence of coded steps that transform raw input data into repeatable outputs, which means more than automation alone in official statistics. It supports validation, documentation, version control, and consistent re-execution over time. Our production process for official statistics generally encompasses the ingestion of data, data cleaning, processing and transformation activities, the calculation of elementary indexes, the aggregation of these indexes using appropriate weighting schemes, and the compilation and dissemination of final indexes (Figure 1).

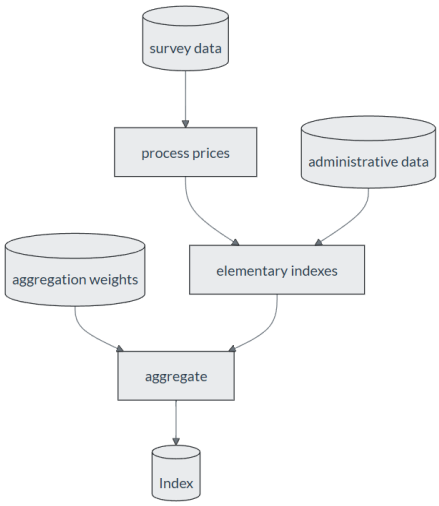

Figure 1: Example of a PPI data pipeline.

This structure is already familiar with PPD's current production environment. Modernized pipelines are designed to reflect the logic of index construction in a transparent way, rather than hiding it inside monolithic systems. In practice, building and maintaining these pipelines still requires substantial coding effort. Analysts must define input structures, adapt code to index-specific requirements, implement aggregation logic, standardize outputs, and ensure that scripts remain aligned with approved methods. Despite differences in subject matter, these tasks often exhibit substantial similarities across programs and over time.

At the same time, generative AI has evolved rapidly beyond simple chatbot use cases. Foundation models, including large language models, can now generate code, summarize requirements, transform structured specifications, and interact with external tools. This creates a

practical opportunity for statistical production. If used carefully, generative AI can help automate repetitive implementation work, reduce technical barriers to coding, and allow analysts to focus more on methodology, data quality, and analysis.

However, models alone are insufficient for this type of work, as they generate plausible outputs based on input text and patterns learned from pre-trained data. For that reason, we use an agent-based approach. In this setting, agents combine a model with instructions, tools, and task-specific responsibilities. Rather than asking one model to perform everything at once, the work is broken into specialized stages. This makes the process easier to control, easier to review, and better suited to production requirements.

The result is a multi-agent workflow that follows the logic of standard price index production: it takes user requirements, interprets them as a structured specification, generates code to construct the index pipeline, and then validates the result before release.

3 Multi-Agent Workflow

The proposed workflow is implemented as a multi-agent orchestration pipeline. It consists of three specialized large language model agents, intake agent, code generator, and code validator, interconnected through workflow control logic that manages user interaction, iterative refinement, and termination conditions (Figure 2). Each agent is powered by a model selected according to its task requirements and is configured with task-specific system instructions, enabling specialization while maintaining a modular architecture.

The workflow is initiated when a user submits a natural language request describing the context (i.e. input files and their structures), the method (i.e. the index name and a high-level description of its computation), and the objective (i.e. the desired output). This request is first processed by the Intake Agent, whose responsibility is to transform the prompt into a structured specification following a predefined schema. In addition to extracting the required information, the agent identifies ambiguities, missing details, or inconsistencies in the request and records them explicitly as part of the specification.

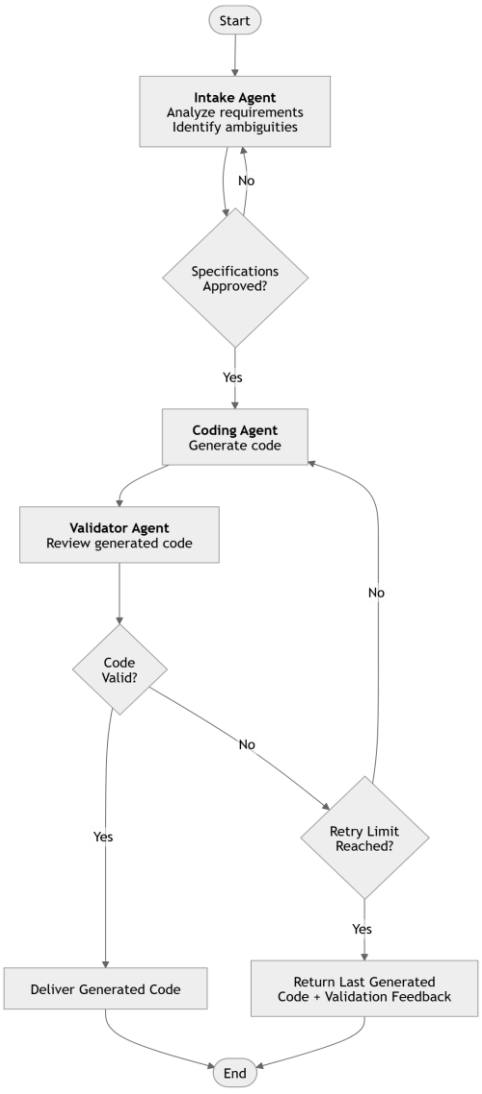

Figure 2: Multi-Agent Workflow.

To ensure that the generated code accurately reflects the user's intent, the workflow incorporates a human-in-the-loop verification stage. The structured specification, together with the identified ambiguities, is presented to the user for review. The user may either approve the specification or provide additional clarifications. This interaction is implemented using a conditional branch

within the workflow. If clarifications are provided, they are appended to the existing specification and returned to the Intake Agent for re-evaluation. This iterative process continues until the user confirms that the specification is complete and accurately captures the requested functionality.

Once the specification has been approved, it is provided to the code generator agent, which uses it as the basis for producing the corresponding R code. Rather than delivering the generated code directly to the user, the workflow introduces a second quality assurance step through a separate code validator agent. The validator independently reviews the code to assess its technical correctness, methodological consistency, and alignment with the approved specification. To perform this validation, the agent uses multiple sources of information, including the task specification, its pre-trained knowledge, documentation retrieved through retrieval-augmented generation (RAG), system instructions, and web searches when additional information is required.

The code generation stage is also iterative. If the validator identifies any issues, it produces structured feedback describing the detected problems. This feedback, together with the previously generated code, is automatically supplied to the code generator. The generator receives feedback and can determine whether the suggested changes are justified. If so, it revises the code to address the identified deficiencies. Then, the revised code is subsequently re-evaluated by the code validator. This generation-validation cycle continues until the validator confirms that the code is correct, or a predefined maximum number of iterations is reached. The workflow maintains an internal loop counter to prevent indefinite execution. If the iteration limit is exceeded, the workflow terminates and returns both the latest generated code and the corresponding validation feedback to the user.

The effectiveness of this architecture stems from separating code generation and code validation into distinct agents. The code generation agent is optimized to produce a complete and plausible implementation, while the code validation agent independently evaluates that implementation to identify logical flaws, unsupported assumptions, inconsistencies, and coding errors. By reasoning independently and optimizing for different objectives, the two agents complement one another, providing an additional layer of quality assurance that improves the likelihood of detecting issues before human review.

When the code validator confirms that no further issues remain, the final R code is returned to the user and the workflow terminates. The code consists of data ingestion, specified transformations, index calculation, and returns the final index values in a specified format. As a final verification step, users can execute the generated code within their own R environment to confirm that it performs as intended under their specific data and execution conditions.

4 Results

The multi-agent workflow was explored using several statistical production programs representing common price index calculation tasks. These included workflows involving data processing, elementary index calculation, aggregation using specified weighting structures, index chaining and rebasing, and the preparation of dissemination outputs. Across the workflows examined, the generated R code was executed successfully and produced results consistent with the expected index calculations. The generated code reflected the methodological requirements specified by users and, in some cases, offered improvements over existing production programs through more efficient or streamlined programming approaches. The iterative generation and validation process also helped identify and resolve ambiguities in the original requirements, improving alignment between the final implementation and the intended methodology.

In addition, the project contributed to the development of a structured prompting approach for code generation. This approach identifies key information required in prompts, including input data structures, expected outputs, and methodological specifications, and provides a reusable reference for similar coding tasks.

Overall, the results indicate that the GenAI workflow can support the translation of approved methodological requirements into reproducible implementations while reducing some of the manual effort associated with code development.

5 Limitations and Mitigation

Using generative AI in official statistics raises several non-trivial challenges. The first is hallucination: models can produce incorrect or fabricated information with a high degree of confidence. In code generation, this can appear as nonexistent functions, incorrect arguments, or implementation logic that looks plausible but does not reflect the requested method. The second challenge is data dependency. Model performance depends heavily on training data, which may not contain the internal or niche knowledge needed for specialized price index production. This is especially relevant in PPD, where workflows rely on internal conventions, domain-specific packages, and detailed methodological practices that are unlikely to be well represented in public training corpora. The third challenge is bias. Models can reproduce or amplify patterns embedded in training data, including inappropriate defaults or generic coding habits that do not fit the requirements of statistical production. The fourth challenge is inconsistency. Since these AI models are probabilistic, the same input can produce different outputs across runs or programs. This creates understandable concerns in a domain where reproducibility is fundamental.

To manage these limitations, our approach combines several safeguards. First, humans remain in the loop throughout the workflow. Human review is especially important at the specification stage and before any generated output is considered suitable for production use. Second, prompt engineering and structured system instructions are used to guide model behavior more precisely. Third, model randomness can be constrained through configuration settings, where available. For example, parameters such as temperature and top-p directly affect the stochasticity of token

selection during response generation, while a fixed random seed may improve reproducibility where supported. Some models also expose reasoning effort settings, which control the computational effort devoted to a task and may improve consistency on complex reasoning problems. Fourth, RAG is used to ground outputs in relevant documentation and reduce unsupported improvisation. Finally, the workflow uses a hybrid design in which generative components are paired with deterministic and rule-based checks. These safeguards do not eliminate model limitations entirely, but they make them more manageable in practice.

With these foundations in place, future work may focus on extending the current workflow into a reusable framework applicable across statistical programs. A common prompt framework could be developed to accommodate program-specific requirements and support broader adoption, and the workflow could be evaluated and implemented across additional statistical programs. In addition, while the current workflow primarily supports index calculation, it could be extended to other stages of the statistical production pipeline, such as data ingestion, outlier detection, and analysis. These efforts aim to create more robust agentic systems that can support a wider range of official statistics production activities while enhancing accuracy, reproducibility, and scalability.

Commented [SW1]: More specific future work provided

Commented [H(2R1]: in the sentence that says other stages of statistical production, we can list some examples afterwards such as outlier detection.

6 Conclusion

While this work is still evolving, the project suggests a potential role for generative AI in supporting official statistics. At this stage, its primary value appears to be in reducing repetitive implementation tasks and helping translate approved methodological specifications into reproducible code more efficiently. Used in this way, generative AI may complement existing statistical production processes while methodological decisions and accountability remain with analysts and subject-matter experts.

In summary, this work contributes to ongoing modernization efforts in price index production by exploring a structured approach to supporting the development of reproducible analytical pipelines. By combining generative AI with human oversight, validation, and approved statistical specifications, the multi-agent workflow has the potential to reduce some routine coding effort while maintaining transparency, reproducibility, and methodological rigor. Although further evaluation is needed, the approach demonstrates how AI-assisted workflows can be incorporated into official statistics in a controlled and reproducible manner and may offer useful insights for other statistical programs considering similar applications.

References

Ha, Xin, and Steve Martin. 2025. “Modernizing Official Statistics: Transition to Open-Source Systems for Production of Producer Price Indexes.” Conference paper presented at the 39th Meeting of the Voorburg Group on Service Statistics, Copenhagen, Denmark.

Martin, Steve. 2024. “piar: Price Index Aggregation in R.” Journal of Open Source Software, 9 (101), 6781, https://doi.org/10.21105/joss.06781.