Modernizing Production Systems: PPIs Using Reproducible Analytical Pipelines

39th Meeting of Voorburg Group of Service Statistics

Xin Ha

Producer Prices Division

Statistics Canada

September 23, 2025

Delivering insight through data for a better Canada

Outline

Background

- Our legacy corporate system was rigid, hard to maintain, and limited in transparency and scalability.
- In 2021, Producer Prices Division started to move away from corporate system to make price indexes towards a pipeline model.
  - Today, this is how we make nearly all our indexes.
- The pipeline model works well and results in better price statistics, but there are some challenges.
- The Reproducible Analytical Pipeline (RAP) framework gives us a way to make big improvements to our current workflow without much extra work.

Initial Pipeline Model

- A collection of R + Python scripts read prices, processes them (e.g., remove outliers), make elementary indexes, aggregate with some weights, write the index to disk.
- Scripts are version controlled on GitLab, usually with some automated tests.
- Documentation on how to execute a pipeline.

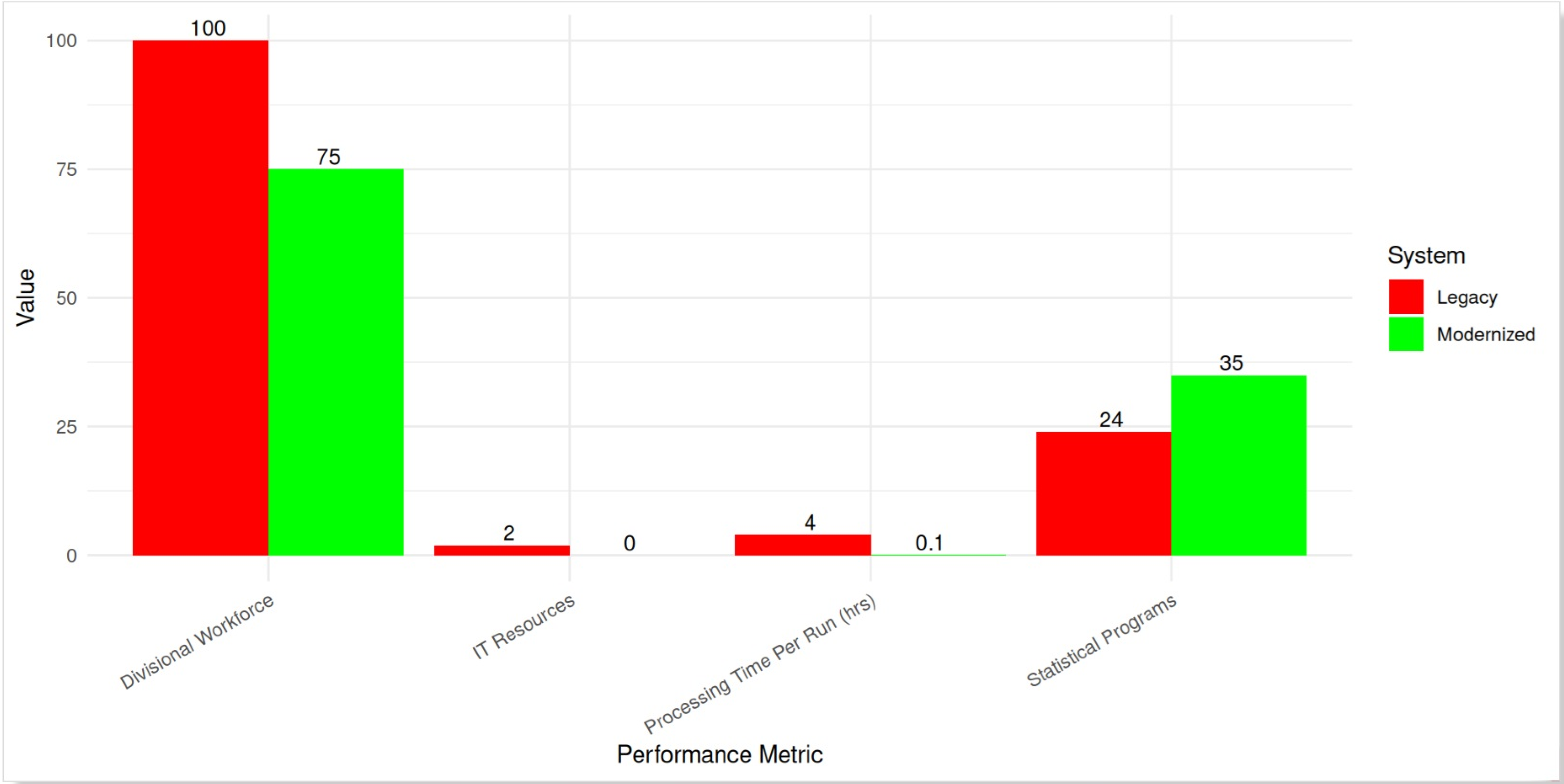
Initial Pipeline Model: Advantages

- Flexible.
- Transparent.
- Easier than corporate systems.

Initial Pipeline Model: Areas for Improvement

- Software environment drift can lead to inconsistencies in output over time or across users.
- Lack of structure allows programs and data to become complex.
- Manually executing a pipeline is less than ideal.

Initial Pipeline Model: Performance Metrics



Legacy vs. Modernized PPI System Performance

What is a Reproducible Analytical Pipeline (RAP)?

- Reproducible Analytical Pipelines is a set of tools, principles, and techniques to help you improve your analytical processes.
- “Reproducible” means someone else can rerun your analysis and get the same results.
  - Crucial for official statistics.
- With RAP, you’ll leverage open-source tools to make your work more efficient, more reusable, and less error-prone.
- A RAP is not just code, it’s a workflow mindset (modular, testable, documented, automated).

Why a RAP?

- Comes out of NHS / UK public service.
  - RAP Community of Practice.
- Popular model in the open science world.
  - e.g., The Turing Way.
- Natural evolution of current workflow for making price indexes.
- Why RAPs matter for official statistics:
  - Ensures statistical reproducibility.
  - Quality: reduced errors.
  - Facilitates peer review, auditing, and transparency.
  - Efficiency: faster updates, reruns, scaling.
  - Helps with workforce turnover. Pipelines are more maintainable.
  - Positions the agency for future innovation (cloud, APIs).

RAP: Ingredients

1. Environment management.
    - Create a reproducible software environment so the same tools are used.
2. Version control (version => reference period).
    - Keep track of the version of code/scripts to make an index.
    - Keep track of the version of data to make an index.
3. Pipeline orchestration.
    - Automate executing scripts to build index.
4. Modular design (each step in the pipeline should be independent and testable).
5. Continuous integration/testing (e.g., GitLab CI/CD).

RAP: Software stack

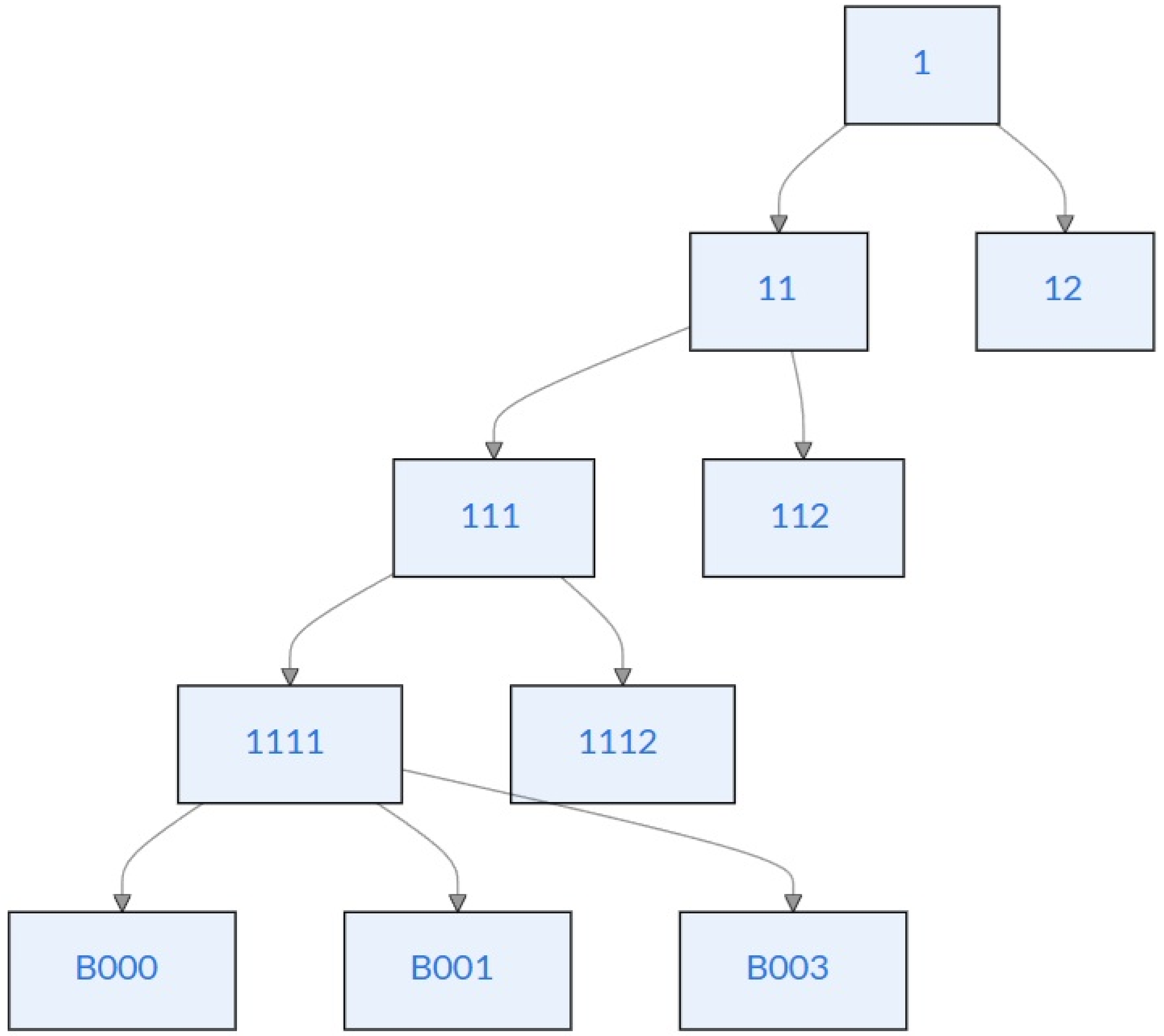
- Lots of ways to implement a RAP.
- Start with R + Python for computational tools.
  - Add git + gitlab for version control, collaboration, testing.
- Use conda for environment management.
- Use DVC to version data with git and orchestrate work as a (targets) pipeline
  - DVC not only versions data but can also help orchestrate pipeline steps through dependency graphs.

Lessons Learned

- Barriers:
  - Learning curve for some tools (Git, Conda, DVC)
  - Resistance to change
- Enablers:
  - Community of practice and shared codebases
  - Training and documentation
  - Management support for modernization

Example

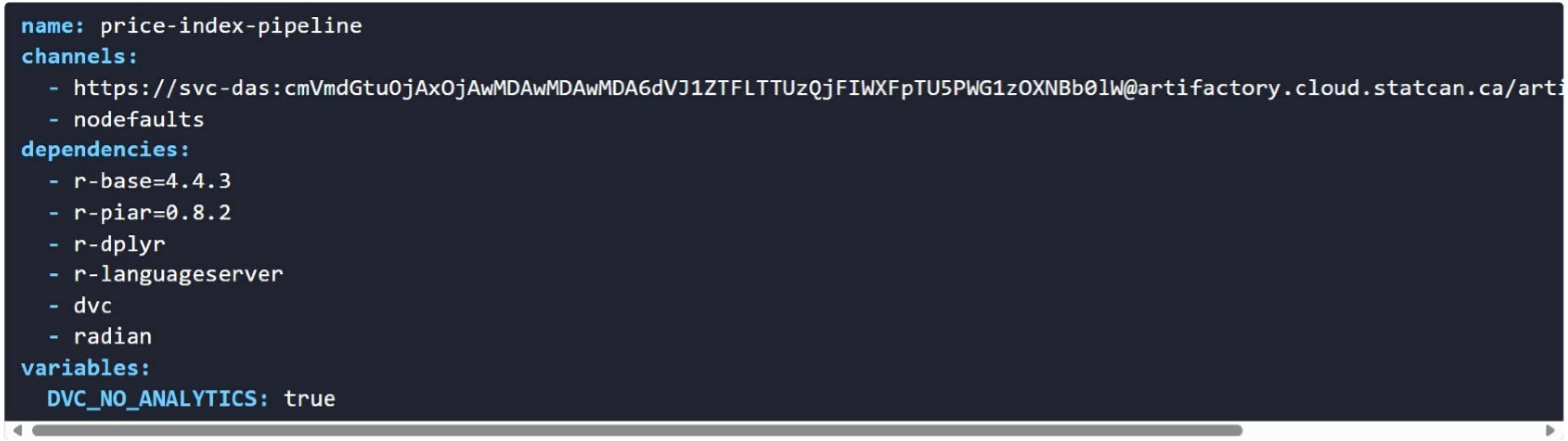
- Make a standard industry index aggregated according to an industry classification and based on data from a sample of 1,000 businesses over 10 years.



Index method

- Businesses are the elementary aggregates.
- Most price data come from a survey that collects prices (Jevons index).
  - Businesses in two subsectors (4-digit) are regulated and there's administrative data with prices and quantities (Törnqvist index).
  - The index for all businesses in one subsector (1255) is imputed from a different index series.
- Business indexes are aggregated with fixed revenue weights.

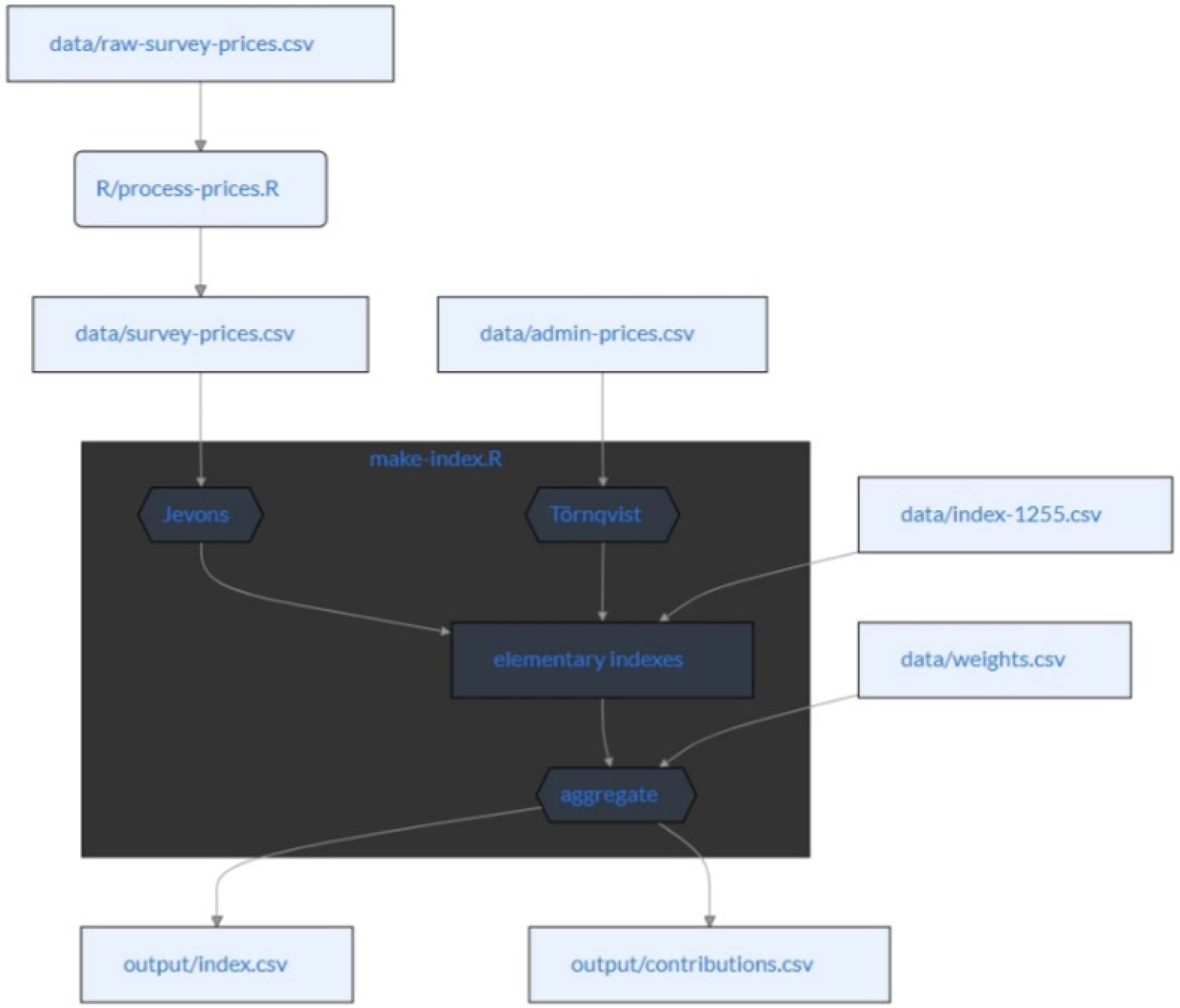
Software Environment



```yaml
name: price-index-pipeline
channels:
  - https://svc-das:cmVmdGtuOjAxOjAwMDAwMDAwMDA6dVJ1ZTFLTTUzQjFIWXFpTU5PWG1zOXNBb0lW@artifactory.cloud.statcan.ca/arti
  - nodefaults
dependencies:
  - r-base=4.4.3
  - r-piar=0.8.2
  - r-dplyr
  - r-languageserver
  - dvc
  - radian
variables:
  DVC_NO_ANALYTICS: true
```

Pipeline Workflow

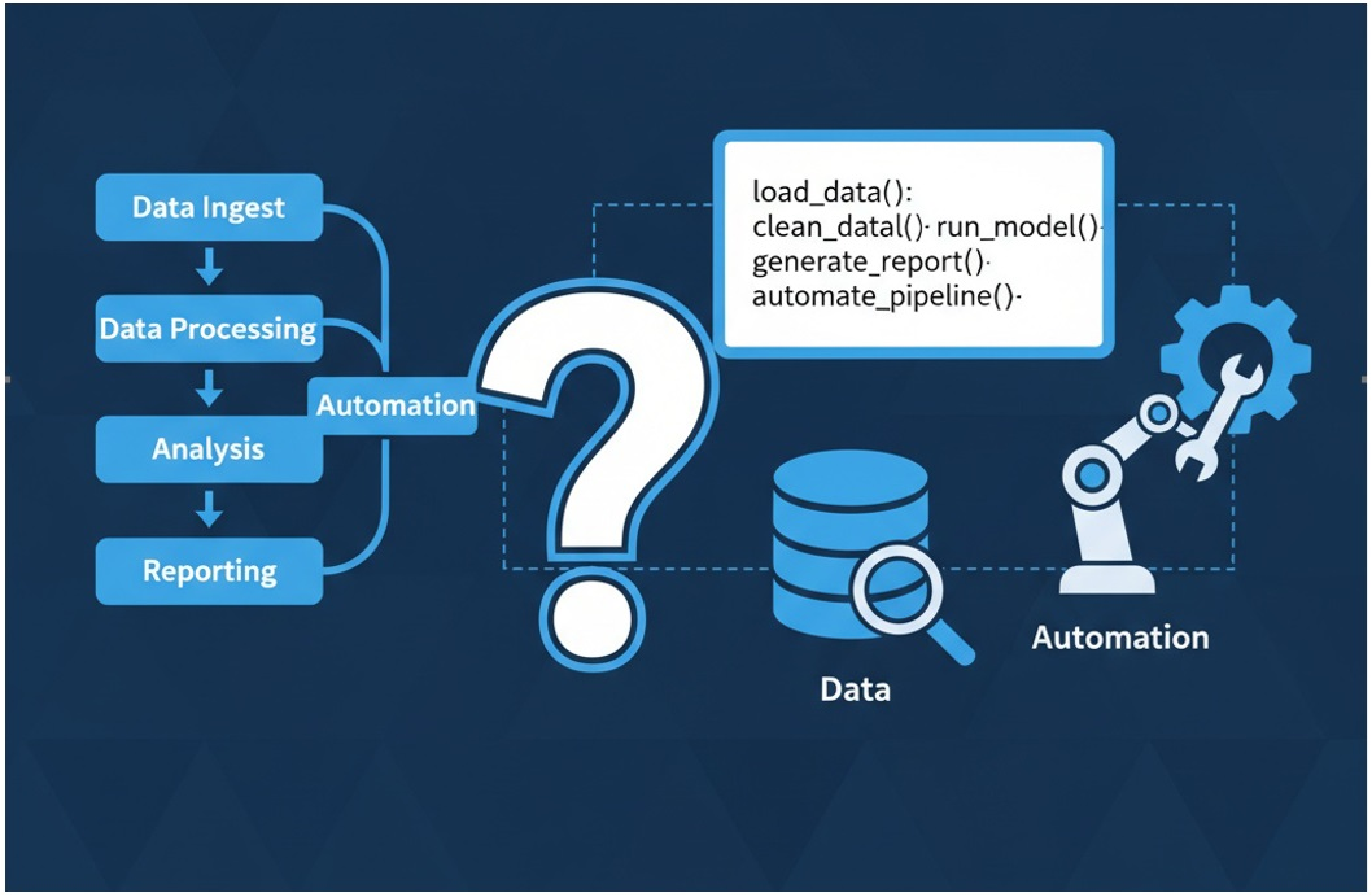
```
stages:
  process-prices:
    cmd: Rscript R/process-prices.R
    deps:
      - R/process-prices.R
      - data/raw-survey-prices.csv
    outs:
      - data/survey-prices.csv
  make-index:
    cmd: Rscript R/make-index.R
    deps:
      - R/make-index.R
      - data/survey-prices.csv
      - data/admin-prices.csv
      - data/index-1255.csv
      - data/weights.csv
    outs:
      - output/index.csv
      - output/contributions.csv
```



Making the index

```
conda activate price-index-pipeline
time dvc repro

Running stage 'process-prices':

> Rscript R/process-prices.R
Updating lock file 'dvc.lock'

Verifying data sources in stage: 'data/raw-survey-prices.csv.dvc'
Verifying data sources in stage: 'data/admin-prices.csv.dvc'
Verifying data sources in stage: 'data/index-1255.csv.dvc'
Verifying data sources in stage: 'data/weights.csv.dvc'

Running stage 'make-index':

> Rscript R/make-index.R
Updating lock file 'dvc.lock'
Use `dvc push` to send your updates to remote storage.
real    15.186s
```

Questions


Data Ingest
Data Processing
Analysis
Reporting
Automation
load_data():
clean_datal()· run_model()·
generate_report()·
automate_pipeline()·
Data
Automation