

Autonomous AI-driven discovery of actuarial reserving models: lessons from self-improving code

by S Mdaka and S Liebenberg

Presented at the Actuarial Society of South Africa's 2026 Convention
Cape Town International Convention Centre, 7–8 October 2026

ABSTRACT

Large language models can now read, modify and evaluate code autonomously. This paper applies that capability to actuarial reserving. Following Karpathy's autoresearch framework, the same self-improving loop is adapted to loss reserving: an artificial intelligence agent iterates through reserving-model configurations on the Casualty Actuarial Society Loss Reserve Database, varying development patterns, tail factors, ultimate-claims methods and credibility blends, scored by actual-versus-expected error on a held-out diagonal. The loop can be applied to any method the underlying library implements, including methods practitioners rarely have time to test, and the agent records why each did or did not help. A complete run of 56 experiments under a frozen, unit-tested evaluation harness is reported, with a case-study run that selects on one-year reserve stability instead (the actuary directing the research chooses the objective), and a validation on untouched diagonals and nine years of actual subsequent development against a pre-specified practitioner-style default. The agent converged, in minutes, on a per-line structure a practising actuary would recognise, with out-of-sample accuracy better than the uniform chain-ladder control. Comparable accuracy is attainable by mechanical search and by sensible untuned defaults, so the agent's demonstrated value is the evidence base it produces (every kept and discarded experiment logged with its hypothesis and rationale) and what the runs reveal about evaluation design. An autonomous loop optimises its metric exactly as implemented: both

metrics tried rewarded deleting tail factors an actuary would keep, so the research brief and the metric must be designed to withstand optimisation.

KEYWORDS

Loss reserving; generative artificial intelligence; autonomous experimentation; chain ladder; model governance

CONTACT DETAILS

S Mdaka, Deloitte; E-mail: smdaka@deloitte.co.za

S Liebenberg, Deloitte; E-mail: mailto:sliebenberg@deloitte.co.za

1. INTRODUCTION

1.1 The actuarial profession has always adapted its tools to the technology of the era. From paper-based calculations and actuarial handbooks, to spreadsheet-driven analysis, to programming-powered predictive and stochastic models, each generation of actuaries has adopted the computational tools available to them. We argue that generative artificial intelligence (AI), specifically large language models (LLMs) capable of writing and modifying code, represents the next such inflection point.

1.2 In March 2026, Andrej Karpathy, Director of AI at Tesla from 2017 to 2022 and a founding member of OpenAI, released autoresearch (Karpathy, unpublished): a small project that gives an AI coding agent a real machine-learning training script and lets it experiment autonomously. The agent modifies the code, runs a training experiment, checks if the result improved, keeps or reverts the change, and repeats, indefinitely. Left running overnight, it conducts approximately 100 experiments, producing a log of discoveries and (often) a meaningfully better model by morning.

1.3 This paper asks whether the same pattern can be applied to actuarial reserving. A scoping note on the word ‘autonomous’: the agent automates the analytical steps in the actuarial control cycle, including method assessment (proposing a method, stating why it should suit a line, and testing it), assumption testing, and reserve estimation; specification of the objective, monitoring of the process, and judgement over the outcome remain human throughout, and section 7 develops what that division of labour requires before any statutory reliance could be placed on the result.

1.4 Section 6 presents evidence that the autoresearch framework can be applied to reserving: a complete 56-experiment autonomous run that arrived at a documented, per-line method selection a practising actuary can interrogate, validated in aggregate against nine years of subsequent development, together with two failure modes (a metric exploit

and a winner's curse) that define the governance conditions under which the pattern is useful. Unlike neural-network training, where the search space is opaque and improvements are difficult to interpret, reserving models are built from well-understood actuarial building blocks: development factors, tail extrapolations, a priori loss ratios, and credibility-weighted blends. When an AI agent finds that a particular combination of these building blocks improves on the baseline, the result can be stated in standard reserving terms and assessed by practising actuaries. This builds on earlier automated selection among traditional reserving methods using out-of-sample actual-versus-expected performance (Balona & Richman, 2020), adding an open-ended coding agent to that work's predefined search grid.

1.5 Position and research questions

1.5.1 This paper applies an autonomous LLM coding agent to classical reserving-method selection through an empirical keep-or-discard loop. Our contribution adds to the automated method selection of Balona & Richman (2020): alongside a predefined search grid, an open-ended coding agent can hypothesise, implement, and test method variants of its own choosing, producing a natural-language audit trail of every accepted and rejected experiment. (Throughout, 'discovery' refers to this empirical keep-or-discard record produced under a frozen metric within a human-written brief, not to ideas originating outside that brief; section 7.5 develops the distinction.) Distinctively, we report the paradigm's observed failure modes as results in their own right (a converged agent exploiting a weakness in its selection metric, and a winner's curse that leaves the selection scores unable to rank the discovered models out of sample), since these are what a governance framework for agentic reserving must provide for.

1.5.2 The research questions are:

- Q1: Can an autonomous keep-or-discard loop, under a frozen evaluation protocol, find reserving-model configurations that improve out-of-sample accuracy?
- Q2: Is the agent's value that it finds a more accurate model than a sensible practitioner default would?
- Q3: What does the agent demonstrably add, and where does it fail? The design pairs one complete frozen-protocol run (section 6) with validation on data the agent never touched: earlier untouched diagonals, nine years of actual run-off, and a comparison against a pre-specified practitioner-style default (section 6.7).

1.5.3 In brief, the answers are as follows:

- Q1: The loop finds real but modest out-of-sample improvement over its uniform control
- Q2: Its value is not accuracy unattainable by other means: a sensible untuned practitioner-style default reaches comparable accuracy on nine years of actual run-off.
- Q3: What it demonstrably adds is a working-paper record, written as the run happened, for each experiment: the hypothesis, the implemented model change, and the resulting score, including rejected changes. It exposed two weaknesses in the evaluation design:

adaptive selection produced a winner's curse, and both metrics rewarded actuarially unsound tail deletion. In the stability-metric case study, the agent also investigated a suspected denominator exploit, which aggregate testing largely disproved (section 6.8). For reserving practice, the practical import is a documented, reproducible evidence base for method selection that supports the statutory actuary (in South Africa, the Head of Actuarial Function, or HAF (Actuarial Society of South Africa, 2022)) in determining and documenting technical provisions, and provides evidence for external audit where those provisions fall within the audit scope (section 7).

1.5.4 This paper proceeds as follows. Section 2 explains the technological developments that make autonomous AI experimentation possible today. Section 3 describes Karpathy's autoresearch framework and its core design principles. Section 4 explains why actuarial reserving is a particularly well-suited domain for this approach. Section 5 details our implementation. Section 6 presents results from autonomous experimentation. Section 7 discusses implications for the reserving profession. Section 8 concludes.

2. WHY NOW? THE CAPABILITIES OF GENERATIVE AI THAT ENABLE THIS

2.1 The rise of code-capable language models

The key enabler of autonomous AI research is not any single model but the convergence of several capabilities that matured between the years 2023 and 2026.

2.1.1 *Code understanding and generation*

Modern LLMs, including OpenAI's ChatGPT, Anthropic's Claude, and Google's Gemini model families, can read an entire codebase, understand its structure, and generate syntactically and semantically correct modifications. This is qualitatively different from earlier code-completion tools: these models can condition modifications on the stated purpose of the code, not only on its syntax.

2.1.2 *Long context windows*

Models now routinely support context windows (the amount of text a model can read at once) of 200,000 to 1,000,000 tokens (the word fragments into which text is split), allowing an agent to hold an entire small project in memory simultaneously: the data pipeline, the model definition, the evaluation harness (the fixed code that loads the data and scores each model), the experiment history, and the instructions.

2.1.3 *Tool use and agentic loops*

The 'agentic' paradigm, in which an LLM is given access to tools (file editing, terminal commands, web browsing) and executes multi-step plans, has moved from research prototype to production capability. Agentic coding products such as OpenAI's

Codex and Anthropic's Claude Code, alongside open-source agent frameworks, allow models to operate autonomously for extended periods.

2.1.4 *Bounded recursive self-improvement*

In a practical, systems-level sense, autoresearch is a bounded form of recursive self-improvement: an AI coding agent iteratively modifies an AI training script, measures the result, and carries accepted changes into subsequent experiments (Karpathy, unpublished). The agent does not change its own model weights or harness; the recursive improvement occurs in the code-and-evaluation loop. Similar systems now operate in frontier research: Google DeepMind's AlphaEvolve combines LLM-generated code changes with automated evaluators and evolutionary selection, and has improved code used in Gemini training (Novikov et al., unpublished). The same bounded pattern can improve reserving-model code while a fixed evaluation protocol preserves comparability across experiments.

2.2 The coding agent as autonomous researcher

What distinguishes the current generation of AI from earlier automation is the ability to operate in an open-ended fashion. Previous approaches to automated model selection (grid search, Bayesian optimisation, automated machine learning) search over a predefined parameter space. An LLM-based agent, by contrast, can in principle:

- restructure code (not merely tune parameters);
- compose existing building blocks in novel ways;
- read documentation and apply techniques represented in its training data;
- reason about why an experiment succeeded or failed and adjust strategy accordingly; and
- generate natural-language descriptions of each experiment for human review.

This makes the agent more akin to a junior researcher than to an optimiser of hyperparameters (the settings fixed before a model is fitted). We stress 'in principle': our own runs exercised only a fraction of these capabilities (every accepted change was a configuration edit within a human-enumerated space; see section 5.4 and Appendix D), section 6.7 shows what the agent's demonstrated contribution is and is not, and section 7.4 sets out the less-used methods the same loop could be applied to next.

3. KARPATY'S AUTORESEARCH: THE BLUEPRINT

3.1 The core idea

The autoresearch project (Karpathy, unpublished) is deliberately minimal: its training code is a simplified implementation of Karpathy's nanochat¹ that runs on a single

¹ nanochat is Karpathy's compact, open-source implementation of a ChatGPT-style language model, covering training and chat in a single small codebase.

graphics processing unit (GPU), and its autonomous experiment loop is built around three core files, each with an assigned editor (Table 1).

TABLE 1. The three files of Karpathy’s autoresearch and who edits each

File	Purpose	Who edits it
prepare.py	Data downloading, tokeniser training, data loading, evaluation metric	Nobody (fixed)
train.py	GPT model definition, optimiser, training loop	The AI agent
program.md	Instructions for the agent: experiment protocol, logging format, decision rules	The human

3.2 The reference implementation targets a single GPU.² The training script runs for a fixed five-minute budget of elapsed time. The evaluation metric is `val_bpb` (validation bits per byte), the model’s compression performance on held-out text: a single number, for which lower is better. The constraint of fixed wall-clock time means every experiment is directly comparable regardless of what the agent changes.

3.3 The experiment loop

3.3.1 The agent operates in an infinite loop.

LOOP FOREVER:

- 1. Modify `train.py` with an experimental idea
- 2. `git commit`
- 3. Run the experiment (5 minutes)
- 4. Read the evaluation metric
- 5. If improved → keep the commit (advance the branch)
If not improved → reset to the pre-experiment commit
- 6. Log results to a TSV file

3.3.2 When an experiment fails to improve the metric, the agent resets the branch (the version-control line of development to which each saved snapshot is added) back to the state it was in before the experiment, discarding the change. Successful experiments advance the branch, so each new experiment builds on the current best configuration.

3.3.3 The agent is explicitly instructed never to stop. It runs autonomously until the human returns, potentially after sleeping for eight hours, giving the agent time for approximately 100 experiments.

2 For readers less familiar with the machine-learning terms: a ‘tokeniser’ converts raw text into the integer tokens a language model consumes, and ‘GPT’ (generative pre-trained transformer) is the transformer-based language-model architecture being trained.

3.4 Key design principles

Several design principles make autoresearch effective:

- **Single metric:** There is exactly one number to optimise. This eliminates ambiguity in the keep-or-discard decision.
- **Fixed budget:** The time constraint means the agent cannot ‘cheat’ by simply training longer. It must find genuinely better configurations.
- **Single-file scope:** The agent modifies only `train.py`. This keeps changes reviewable and reduces the risk that the agent will break the evaluation harness.
- **Simplicity criterion:** The instructions explicitly state that simpler code is preferred: ‘A 0.001 `val_bpb` improvement that adds 20 lines of hacky code? Probably not worth it. A 0.001 `val_bpb` improvement from deleting code? Definitely keep.’ In other words, a marginal improvement does not justify brittle complexity; an equally small improvement achieved by simplifying the system is valuable.
- **Interpretable log:** Every experiment is logged with a human-readable description, creating an audit trail that is itself a research artefact. In Karpathy’s original project each log row records the commit hash,³ the `val_bpb` metric, peak memory,⁴ a keep, discard or crash status, and a natural-language description of the experiment.

3.5 What Karpathy called ‘programming the program’

A notable feature of autoresearch is the shift in the human’s role. As Karpathy (unpublished) puts it, ‘you’re not touching any of the Python files like you normally would as a researcher. Instead, you are programming the `program.md` Markdown files that provide context to the AI agents and set up your autonomous research org.’ The human does not write the code that trains the model; the human writes `program.md`, a Markdown document that instructs the AI agent on how to do research, and iterates on that document over time to improve the rate of research progress. The human directs the research rather than implementing it.

4. WHY ACTUARIAL RESERVING IS IDEAL FOR THIS APPROACH

The reserving problem addressed here is the standard one: estimating the ultimate cost of claims that have occurred but are not yet fully settled, from the claims triangle indexed by origin period and development period, whose unobserved lower-right portion the models project. We assume familiarity with development triangle methods and refer readers to Friedland (2010) for the estimation techniques and to England & Verrall (2002) and Wüthrich & Merz (2008) for their stochastic treatment; this paper applies the standard reserving toolkit rather than explaining its underlying methods in detail.

3 A ‘commit hash’ is the unique identifier that version-control software assigns to each saved snapshot of the code, so that a logged result can be traced to the exact code that produced it.

4 ‘Peak memory’ is the largest amount of graphics-processor memory the training run used, a binding resource constraint in that setting.

4.1 A rich but finite method space

4.1.1 The standard actuarial reserving toolkit includes chain-ladder and expected-loss methods (Friedland, 2010), together with stochastic approaches that quantify prediction uncertainty (Table 2) (England & Verrall, 2002; Wüthrich & Merz, 2008).

TABLE 2. The classical reserving methods in the agent’s search space

Method	Key idea
Chain Ladder	Future development follows the same pattern as historical development
Bornhuetter–Ferguson (BF)	Blend actual emergence with a prior expectation of ultimate losses
Cape Cod	Like BF, but derive the prior expectation from the data itself
Benktander	Iterated BF that interpolates between BF and Chain Ladder

- 4.1.2 Each method has tuneable components:
- **development factor selection:** the historical periods included in the average, volume-weighted versus simple averaging, and the treatment of outlier link ratios;
 - **tail-factor estimation:** the extrapolation beyond the observed development horizon (exponential or inverse-power curves, the Bondy method, or no tail);
 - **a priori selection:** the expected loss ratio assumed by the expected-loss methods; and
 - **method blending:** per-origin-year weighting between methods.

4.1.3 This creates a large but structured search space whose elements correspond to recognised actuarial methods and parameters (Friedland, 2010), in contrast to the neural-architecture space of Karpathy’s original application.

4.2 The interpretability advantage

- 4.2.1 A key difference between the reserving application and the original LLM-training application by Karpathy is the interpretability of the search space. When an agent discovers that a different learning-rate schedule⁵ improves `val_bp`, it is difficult to extract a general principle that transfers beyond that specific set-up. When an agent discovers that excluding the highest of the three most recent link ratios, and estimating the ultimate with the Cape Cod method, reduces the commercial-auto actual-versus-expected error from 1.75% to 0.63% relative to the uniform chain-ladder baseline (section 6.4), the change is expressed entirely in recognised actuarial quantities (Friedland, 2010). A practising actuary can examine why the selection works:
- averages over recent periods respond to calendar-year trends in the development pattern;

5 The ‘learning rate’ controls how large a step the training procedure takes each time it updates a neural network’s parameters; a ‘schedule’ varies that step size over the course of training.

- excluding the highest link ratio in each development period reduces the influence of unusually large development factors; and
- an expected loss ratio estimated from the data is more stable than a fixed assumption.

4.2.2 Actuaries can then test the finding on their own books and use it to refine their professional judgement. The experiment log thereby records, for each experiment, a hypothesis, a result, and an interpretation in standard reserving terms.

4.3 Fast evaluation, and where the time and cost go

Karpathy's LLM experiments take five minutes each (with start-up and compilation overhead added), yielding roughly 12 experiments per hour. Actuarial reserving experiments, by contrast, evaluate in one to three seconds on standard hardware. The primary run completed 56 documented experiments in about 12 minutes, including about 2.6 minutes of model-evaluation time. Most of the elapsed time was therefore spent selecting, implementing and documenting experiments. A mechanical sweep is faster and cheaper per configuration, but a human could not test and document as many variants in the same time. The agent's advantage here is the hypothesis and rationale recorded for each experiment, not search speed (sections 5.4 and 6.7).

4.4 Why reserving specifically?

While the autoresearch pattern could be applied to many actuarial problems (pricing, capital modelling, mortality forecasting), we chose reserving for several reasons:

- **Well-established ground truth:** By using historical triangles where later diagonals have emerged, we have a direct measure of prediction accuracy (actual versus expected).
- **Standardised data:** The Casualty Actuarial Society Loss Reserve Database (CLRD) provides a common benchmark across multiple lines of business (Casualty Actuarial Society, 2025).
- **Mature software ecosystem:** The `chainladder-python` library implements the full suite of standard methods through an application programming interface (API) compatible with the scikit-learn machine-learning library (Casualty Actuarial Society, 2026).
- **High professional relevance:** Reserving is a core statutory task for every short-term (general) insurer, and the technical-provision estimate is central to the reserving actuary's responsibilities. Improvements here have a direct business effect.
- **Regulatory context:** In South Africa, the HAF must express an opinion on the insurer's technical provisions under the Insurance Act, 2017,⁶ and the Prudential Authority's Solvency Assessment and Management (SAM) prudential standards. Where those provisions fall within an audit or regulatory-assurance engagement, the external auditor

6 Insurance Act, No. 18 of 2017

must also obtain sufficient appropriate evidence of the reported amounts. A documented, reproducible evidence base for method selection supports both responsibilities, while the actuarial and audit opinions remain distinct.

Several of these reasons apply equally to other actuarial domains. Pricing in particular shares the ground truth, tooling, relevance, and regulatory arguments, and at the most fundamental level a chain-ladder projection can itself be cast as a generalised linear model, (GLM) (England & Verrall, 2002), so reserving-method selection is a machine-learning problem in the same sense pricing is. Two features nonetheless make reserving the natural first demonstration. The set of widely used, well-performing reserving methods is small and enumerable, so the search space is structured and its elements are recognised methods rather than arbitrary model classes; the public availability of completed triangles permits genuine out-of-time validation of every candidate (section 6.7), which pricing data rarely afford. Finally, because the technical-provision estimate has a statutory role (reason 5), the resulting evidence base has a direct professional use.

5. OUR IMPLEMENTATION

5.1 Architecture overview

5.1.1 Following the autoresearch blueprint, our project consists of three files (Table 3).

TABLE 3. The three files of the reserving implementation and their autoresearch analogues

File	Purpose	Analogue in Karpathy’s project
<code>prepare.py</code>	Data loading, evaluation metric (read-only)	<code>prepare.py</code>
<code>reserve.py</code>	Reserving-model configuration (agent-editable)	<code>train.py</code>
<code>program.md</code>	Agent instructions with actuarial context	<code>program.md</code>

The `prepare.py` file is fixed and provides:

- data loading from the `chainladder-python` library;
- a standardised actual-versus-expected evaluation metric; and
- aggregation across six lines of business.

5.1.2 The read-only status of `prepare.py` is a convention, not a mechanism. The agent-edited `reserve.py` runs in the same process, imports the same library, and nothing but the agent’s instructions in `program.md` prevents it from reloading the public dataset and reading the ‘held-out’ diagonal directly. The recorded code-change history shows no run ever did this, but for production use on real data, the held-out data must be isolated from the agent by construction (a separate scoring process, an integrity-verified evaluation harness, a held-out store the agent cannot reach), not by comment. The evaluation harness is version-controlled and unit-tested. The tests confirm that the

intended held-out data are excluded from model fitting and that a model failure results in rejection, rather than an artificially favourable score. These tests protect against coding defects; they do not prevent an agent from deliberately accessing the held-out data.

5.1.3 The `reserve.py` file contains everything the agent can modify:

- development pattern configuration (averaging method, number of periods, outlier treatment);
- tail-factor selection (curve type, fitting parameters);
- ultimate-claims method choice (chain ladder, BF, Benktander, Cape Cod), which, following the `chainladder` library, we call ‘IBNR methods’ (incurred but not reported), although on a paid triangle the projected ultimate minus paid-to-date is the total unpaid-claims liability, not IBNR alone;
- method-specific hyperparameters (a priori loss ratios, trend, decay); and
- the `build_model()` function itself⁷, so that the agent can restructure the entire reserving pipeline.

5.2 The evaluation metric

5.2.1 Our scoring metric is the weighted mean absolute percentage error (wMAPE) of actual versus expected cumulative paid positions on the held-out diagonal, computed per line of business and averaged with equal weight across the six lines. We call this held-out calendar-year diagonal the ‘selection diagonal’, because every keep-or-discard decision is scored against it. The evaluation procedure is:

- **Hold out the latest diagonal:** for each line-of-business (LOB) triangle, we remove the most recent calendar period’s data (the latest diagonal). For the CLRD this leaves a 9×9 training triangle valued at year-end 1996; nine origin years per LOB are scored (the valuation-year origin has no training data and cannot be projected).
- **Fit the model:** we fit the reserving model on the truncated triangle.
- **Project forward:** from the fitted model’s projection of the completed triangle, we extract the model’s expectation for what the held-out diagonal should contain.
- **Compare:** we compute the absolute difference between the model’s projection and the actual emerged values for each origin year.
- **Score:** the wMAPE summarises this into a single number:

$$\text{wMAPE} = \frac{\sum_i |A_i - E_i|}{\sum_i |A_i|},$$

where A_i is the actual cumulative paid loss for origin year i at the held-out valuation date and E_i is the model’s expectation of that cumulative position.

⁷ `build_model()` is the function in `reserve.py` that assembles the reserving pipeline (development factor selection, tail extrapolation, and ultimate-claims method) from the configuration settings. Because the agent may rewrite the function itself, it is not limited to changing parameter values.

5.2.2 This metric was chosen for several reasons:

- It is scale-independent (percentage-based), allowing fair comparison across LOBs of different sizes.
- It penalises both over- and under-estimation equally, though we note that the actuarial loss function is asymmetric in severity in practice (material under-reserving overstates solvency and attracts supervisory consequences when it unwinds, while over-reserving immediately depresses reported own funds, and both directions depart from the unbiased best estimate that SAM and International Financial Reporting Standard 17 (IFRS 17) require), so a production deployment could substitute an asymmetric loss.
- It is weighted by the magnitude of actual losses, so errors on large origin years matter more.
- It is directly aligned with the actuarial concept of actual-versus-expected analysis, a standard component of reserve reviews.

5.2.3 Both A_i and E_i are cumulative paid positions, and the distinction between the cumulative and incremental bases matters for reading every number in this paper. First, most of each cumulative position was already observed at fit time and is reproduced exactly by every model, so only the final year's emergence is being predicted. Second, errors expressed as a percentage of the cumulative base are therefore mechanically small (on this dataset the cumulative basis error is roughly one-sixth to one-thirteenth of the corresponding incremental basis error per line, and about one-ninth of it at the composite score, the equal-weighted average across the six lines of business), so the wMAPE levels reported here must not be read as reserve estimation errors. Third, the absolute numerator $|A_i - E_i|$ is unaffected (the paid-to-date amounts cancel, so it equals the projected emergence error); it is only the denominator that shrinks the ratio. The same compression applies to the actual-to-expected (A/E) ratios: an A/E of 0.985 on the cumulative position corresponds to over-projecting the year's emergence by roughly 13%, and the baseline's product liability cumulative A/E of 0.892 corresponds to an emergence A/E near 0.5. Where sections 6.1, 6.7 and 6.8 below quote 'emergence A/E', that is the genuinely incremental ratio; the two bases must not be compared directly. We also track per-line metrics, including the root-mean-square error, the unweighted mean absolute percentage error and the aggregate actual-to-expected ratio, but the composite score the agent optimises is a single number: the equal-weighted average of the six per-LOB wMAPEs (Appendix B).

5.3 The dataset

5.3.1 We use the CLRD, a well-known actuarial benchmark dataset included in the `chainladder-python` library (Casualty Actuarial Society, 2025; 2026). It contains:

- 775 triangles, one per company and line of business, from 376 US insurers;
- six lines of business: commercial auto, medical malpractice, other liability, private

- passenger auto, product liability, and workers' compensation (the database's short labels for these lines, `comauto`, `medmal`, `othliab`, `ppauto`, `prodliab` and `wkcomp`, appear only in the configuration listings and experiment log of the appendices);
- ten origin years by ten development periods (annual intervals); and
 - six financial columns: Incurred Loss, Cumulative Paid Loss, Bulk Loss, Earned Premium (Direct, Ceded, Net).

5.3.2 We group by line of business and evaluate on `CumPaidLoss` (cumulative paid losses), using `EarnedPremDIR` (direct earned premium) as the exposure base for methods that require it (BF, Cape Cod). The diversity of LOBs is important: short-tail lines (Private Passenger Auto, Commercial Auto) develop quickly and are well-suited to simple chain-ladder methods, while long-tail lines (Medical Malpractice, Product Liability) are more challenging and may benefit from expected-loss methods, heavier tail factors, or different development factor selections in the chain-ladder family (shorter averaging periods, simple rather than volume-weighted averages, or outlier-trimmed link ratios) relative to the all-period volume-weighted default; the run's Findings 2 and 3 (section 6.6) bear this out.

5.3.2 Grouping across 376 insurers is an analytical choice with consequences: the industry aggregate is far smoother than any single insurer's book, which strengthens the evaluation signal per diagonal but means the results demonstrate the framework rather than any entity-level recommendation. Section 6.7 reports how the aggregate-discovered model transfers to the largest single writers.

5.4 The agent's search space

5.4.1 The baseline configuration in `reserve.py` uses volume-weighted development factors, an exponential tail curve, and the basic chain ladder. It was chosen as a uniform yardstick rather than as a representation of practice, in which method choice is routinely tailored by line (section 6.6, Finding 1). From there, the agent can explore four groups of dimensions.

- 5.4.2 The development pattern dimensions are:
- averaging method: volume, simple, regression (as implemented by `chainladder-python`; Casualty Actuarial Society, 2026);
 - number of recent periods included in the link-ratio average (the 'factor window'): all available periods, or only the most recent three, five or seven;
 - outlier removal ('trimming'): excluding the highest or lowest observed link ratio, or both, in each development period before averaging; and
 - alternative estimators: `ClarkLDF` (parametric curve fitting), `TweedieGLM` (generalised linear model), `DevelopmentML` (machine-learning interface).

5.4.3 The tail-factor dimensions are:

- curve type: exponential, inverse power;
- Bondy method with varying attachment ages;
- constant tail factor; and
- no tail at all.

5.4.4 The ultimate-claims (IBNR) method dimensions are:

- basic chain ladder;
- BF with varying a priori loss ratios;
- Benktander with varying iteration counts (interpolating between BF and the chain ladder); and
- Cape Cod with varying trend and decay parameters.

5.4.5 The structural dimensions are:

- per-LOB model specialisation (different configurations for different lines);
- ensemble methods via `VotingChainladder` (blending multiple methods with per-origin-year weights); and
- pipeline composition (chaining the library's transformation steps, which scikit-learn calls 'transformers', in different orders).

5.4.6 This is the space available to the agent; the space actually searched was narrower: all the runs to date explored the classical development, tail and method dimensions and their per-LOB composition (some 4,320 configurations per line across these dimensions), while the machine-learning-style estimators (`ClarkLDF`, `TweedieGLM`, `DevelopmentML`) and `VotingChainladder` ensembles remain untouched. That narrowness is a product of the run design rather than the method space: a 56-experiment greedy budget (each change accepted or rejected solely on whether it improves the current score, without regard to later experiments), a classical starting point, and an instruction file whose priority ordering places the classical dimensions first (Appendix D). In the explored dimensions the space is small enough that a mechanical grid can search it exhaustively in minutes; the interesting question this paper tests is not whether an agent can search faster than a simple programmed loop (it cannot) but what a hypothesis-driven, self-documenting searcher adds and where it fails. The untouched estimators are where the agent's reach exceeds a grid's: a grid enumerates parameter values in methods a human has already implemented, whereas the agent can implement a further method from the library, fit it, and record why it did or did not help. A run that iterates over the whole available space, seeded from the primary run's final configuration and with the brief re-prioritised towards the less-used methods, is the natural next study (section 7.4).

5.5 The experiment loop

5.5.1 The agent follows the experiment loop below, which preserves the loop semantics of Karpathy's autoresearch (section 3.3):

LOOP:

1. Review current state and experiment history
2. Choose a modification to `reserve.py` and state the hypothesis
3. `git commit` with descriptive message
4. Run: `python reserve.py > run.log 2>&1`
5. Extract: `grep "^composite_score:" run.log (+ per-line detail)`
6. Write `experiment_journal.md` entry: hypothesis, change, result
7. Log to `results.tsv`
8. If improved → keep (advance branch)
If not → `git reset --hard HEAD~1` (revert)
9. Repeat

5.5.2 Each experiment's model evaluation takes one to three seconds; end-to-end throughput is governed by the agent's own deliberation (§4.3). The `results.tsv` log captures every experiment, successful or not, creating a complete record of the agent's search; a companion journal, `experiment_journal.md`, records the agent's hypothesis, configuration change, and result for each experiment, with interpretations consolidated in its running summary.

5.5.3 Relative to Karpathy's original loop (commit, run, read a single metric, keep if improved else revert, log, repeat), the loop swaps in a reserving-specific metric and log schema, and makes the journal entry a mandatory step of every iteration. Where the original optimises `val_bpb` and logs peak memory, our log records the composite wMAPE and the average unweighted mean absolute percentage error against actual emerged losses (see section 5.2), reflecting the quantities of actuarial interest.

5.6 The instruction file

Our `program.md` file provides the agent with:

- **Actuarial context:** an explanation of triangles, development factors, IBNR, and the standard methods, giving the agent domain knowledge to reason about why certain changes might help.
- **The chainladder API reference:** a summary of available classes, their parameters, and how they compose, so the agent knows what tools it has.
- **Experiment-strategy guidance:** a prioritised list of ideas to explore, from simple parameter tuning to advanced ensemble methods.
- **Decision rules:** rules for when to keep, discard or debug an experiment, the simplicity criterion, and the instruction never to stop.

The quality of `program.md` is itself a research variable: better instructions lead to more productive experimentation. This mirrors Karpathy’s observation that the human’s job is to ‘program the program’.

6. RESULTS

6.1 The run at a glance

6.1.1 The paper’s evidence is one complete autonomous run under a frozen, unit-tested evaluation harness: 56 experiments, whose trajectory this section narrates and whose final model section 6.7 validates against the database’s true subsequent development. A separate case-study run swaps the selection metric from accuracy to one-year stability (the claims development result, CDR) and shows how strongly the metric choice shapes what the loop discovers (section 6.8). The complete experiment logs and journals for both runs are preserved as working papers. Section 6.7 then validates the final model on data the agent never touched (earlier diagonals and nine years of actual run-off) and places it beside a pre-specified practitioner-style default, so that the reader can judge whether the documented search arrived at a model a practising actuary would recognise. Table 4 summarises the two runs against the uniform baseline.

TABLE 4. The primary run and the stability-metric case study at a glance

Run	Experiments (kept)*	Selection wMAPE (1997 diagonal, final)	True run-off wMAPE (45 unobserved cells)**	Emergence A/E (1997 → age 120)
Uniform chain-ladder baseline	—	0.0363	4.93%	0.913
Primary run	56 (15)	0.0224	4.24%	1.030
Stability-metric (CDR) run	15 (5)	0.0362***	6.77%	0.978
*Counting experiments after each run’s baseline evaluation.				
**Mean per-line error against actual development to the end of calendar year 2006 (section 6.7).				
***The CDR run selects on the mean weighted claims development result (wCDR), the one-year stability measure of section 6.8, which it improved from 0.210 to 0.083; its accuracy-metric score is shown for comparability only.				

6.1.2 The primary run began from the standard chain-ladder baseline and reduced the composite score from 3.63% to 2.24%: a 38.2% relative improvement, measured on the held-out diagonal used for the keep-or-discard decisions (sections 6.7 and 7.6 quantify how much of that survives on data the agent never touched).

6.2 Baseline performance

6.2.1 Our baseline configuration (volume-weighted chain ladder on all development periods with an exponential tail curve) establishes the starting point (Table 5).

TABLE 5. Baseline accuracy by line of business on the 1997 selection diagonal

Line of Business	wMAPE	A/E (cumulative paid position)
Commercial auto	1.75%	0.985
Medical malpractice	2.11%	0.987
Other liability	1.02%	0.996
Private passenger auto	1.18%	0.988
Product liability	12.99%	0.892
Workers' compensation	2.71%	0.974
Average (composite)	3.63%	—
The A/E column is on the cumulative-paid basis of section 5.2 and is compressed towards 1 accordingly; every ratio below 1 means the model projected more emergence for the year than actually arrived.		

6.2.2 Two features of Table 5 are relevant to the analysis that follows. First, other liability is already well served by the plain chain ladder at 1.0%. Second, product liability dominates the composite. Every A/E ratio at baseline is at or below 1: the uniform chain ladder over-projected the 1997 calendar year's emergence on every line.

6.3 The trajectory of the search

6.3.1 The primary run's 56 experiments fall into four thematic phases (the phases are groupings made after the fact; a few experiments span them). Of the 56 experiments, 15 were kept and 41 were discarded, a 27% keep rate. Table 6 shows every kept experiment and the most instructive discards; the complete log (the baseline and all 56 experiments) is Table 14 in Appendix C. The run journal records the hypothesis and composite outcome for every experiment; from Experiment 6 onwards it also records per-LOB results, with interpretations consolidated in its running summary.

TABLE 6. Kept experiments and the most instructive discards of the primary run

#	Change	Score	Δ versus best	Status	Key finding
0	Baseline (volume-weighted chain ladder, exponential tail)	0.03627	—	keep	Starting point
1	Cape Cod on all lines	0.03222	−0.00405	keep	The premium-based prior reduces the over-projection
2	Cape Cod for product liability only	0.02996	−0.00226	keep	Product liability is the dominant gain
3	BF (a priori 0.70) for product liability	0.03473	+0.00477	discard	Fixed a priori too rigid
5	Cape Cod for commercial auto, other liability and product liability	0.02806	−0.00190	keep	Cape Cod retained for the two liability lines and for commercial auto, which carries third-party liability
6	Product liability: drop highest link ratio	0.02641	−0.00165	keep	Excluding the top link ratio reduces the projection

9	Three-period factor window on all lines	0.02501	−0.00140	keep	Shorter averaging periods improve all lines
13	Medical malpractice: all periods	0.02593	+0.00092	discard	All-period average rejected for medical malpractice
14	Workers’ compensation: drop highest	0.02438	−0.00063	keep	The exclusion improves a second line
17	Commercial auto: drop highest	0.02419	−0.00020	keep	Third line trimmed
19	Private passenger auto: drop highest	0.02380	−0.00039	keep	Fourth line trimmed
21	Medical malpractice: drop highest	0.02350	−0.00030	keep	Five of six lines now trimmed
24	Simple averaging on all lines	0.02343	−0.00008	keep	Simple average of recent ratios, highest excluded
29	Product liability: no tail	0.02337	−0.00006	keep	Metric exploit (Finding 6)
36	Workers’ compensation: Cape Cod	0.02506	+0.00170	discard	Exposure methods do not help workers’ compensation
38	No tail on any line	0.02386	+0.00050	discard	Metric retains tail sensitivity via old origins
39	Other liability: five-period window	0.02289	−0.00048	keep	A longer averaging period preferred for other liability
43	Other liability: seven-period window	0.02249	−0.00040	keep	... settling at seven periods (all periods slightly worse)
47	Product liability: four-period window	0.02248	−0.00002	keep	Negligible change
52	Medical malpractice: Cape Cod	0.02648	+0.00400	discard	Exposure methods again worsen medical malpractice
53	Other liability: Cape Cod decay 0.8	0.02243	−0.00005	keep	Negligible change; final state

6.3.2 Phase 1: Method structure (Experiments 1 to 5)

Because the baseline over-projected every line, the exposure-based methods improved the score immediately: Cape Cod applied globally improved the composite (Experiment 1), and per-line probing then localised the gain to Cape Cod for commercial auto, other liability and product liability, with the chain ladder elsewhere. A fixed a priori BF and four Benktander blends for product liability were all inferior to Cape Cod (Experiments 3, 4, 7, 11, 31 in Appendix C): because the industry-aggregate loss ratio drifted across the era, the data-driven a priori outperformed every fixed one we tried. This per-line method assignment became the final model’s structural core, with the caveat that the idea is suggested in the agent’s own instruction file (Appendix D).

6.3.3 Phase 2: Robust development factors (Experiments 6 to 28)

The run’s central discovery is that one-sided trimming of link ratios is beneficial. Dropping the highest link ratio per column was kept for product liability (Experiment 6), then workers’ compensation, commercial auto, private passenger auto, and medical malpractice (14, 17, 19, 21): five of six lines in all, while it was tested and discarded twice for other liability. Combined with the kept short factor window ($n = 3$, Experiment 9) and the switch

to simple averaging (Experiment 24), the resulting selection is a simple average of the three most recent link ratios in each development period, excluding the highest: a robust response to calendar-year trends in the development pattern, trading a small downward bias for reduced sensitivity to upward outliers among the link ratios, whose frequency increased over calendar time in these data. Reverting medical malpractice to an average over all available periods was tested and discarded twice (Experiments 13, 28): its trimmed short-period factors track the line's calendar-year trend better than its long history.

6.3.4 Phase 3: Tails, and a metric exploit (Experiments 22, 29, 37, 38, 54)

The inverse-power tail was probed and discarded (Experiment 22). Experiment 29 is more instructive: removing product liability's tail factor entirely was kept, improving the score by a within-noise margin. This change has no actuarial justification: it improves the score only because the metric is sensitive to the tail assumption solely through the oldest origin year's final cell. It is an 'exploit' in the sense of a change that improves the measured score while worsening the underlying model. Removing all tails worsened the score (Experiment 38), so the insensitivity is line-specific; and the CDR-metric case study kept the same class of deletion (globally, where this run's was selective) for its own mechanistic reason (section 6.8), showing that the result follows from how each metric is constructed, rather than being a quirk of the primary run. We flag Experiment 29 as adopted-by-protocol but actuarially unsound (no production basis should book a no-tail product liability model), and we quantify what it does to reserves in sections 6.4 and 6.7. Its significance lies in what it shows about greedy selection (section 5.4) and metric design, rather than in the score improvement itself (Finding 6).

6.3.5 Phase 4: Fine-tuning (Experiments 39 to 56)

Other liability alone benefited from a longer factor window, settling at seven periods (Experiments 39 and 43); using all available periods was slightly worse (Experiment 44). Product liability moved to a four-period average, and other liability's Cape Cod method placed modestly more weight on recent origin years (decay 0.8); both changes moved the score by amounts too small to carry meaning. The final probes were all discarded: Cape Cod for medical malpractice (Experiment 52, the largest deterioration of the second half of the run), a two-period window for commercial auto, two-sided trimming, and Bondy tails. With no further improvement from these probes, the run was stopped.

6.4 Final model: baseline versus optimised

6.4.1 The primary run's final model is set out in Table 7.

TABLE 7. Components of the baseline and of the final agent-discovered model

Component	Baseline	Final (agent-discovered)
Development averaging	Volume-weighted, all periods	Simple average of the three most recent periods (other liability: seven; product liability: four)
Outlier treatment	None	Drop highest link ratio per column (all lines except other liability)
Tail factor	Exponential curve	Exponential curve (product liability: none; flagged, Finding 6)
Commercial auto method	Chain ladder	Cape Cod (pooled a priori, decay 1.0)
Medical malpractice method	Chain ladder	Chain ladder
Other liability method	Chain ladder	Cape Cod (decay 0.8)
Private passenger auto method	Chain ladder	Chain ladder
Product liability method	Chain ladder	Cape Cod (pooled a priori, decay 1.0)
Workers' compensation method	Chain ladder	Chain ladder

6.4.2 Table 8 gives the per-line comparison on the selection diagonal.

6.4.3 Every line improved (Figure 1), and, more meaningfully, every A/E ratio moved towards 1: the discovered model's lower projected development corrected the baseline chain ladder's uniform over-projection of the 1997 calendar year. Product liability remains the hardest line at 8.50%, and section 6.7's run-off test shows that, for this line specifically, correcting the 1997 diagonal was the wrong lesson.

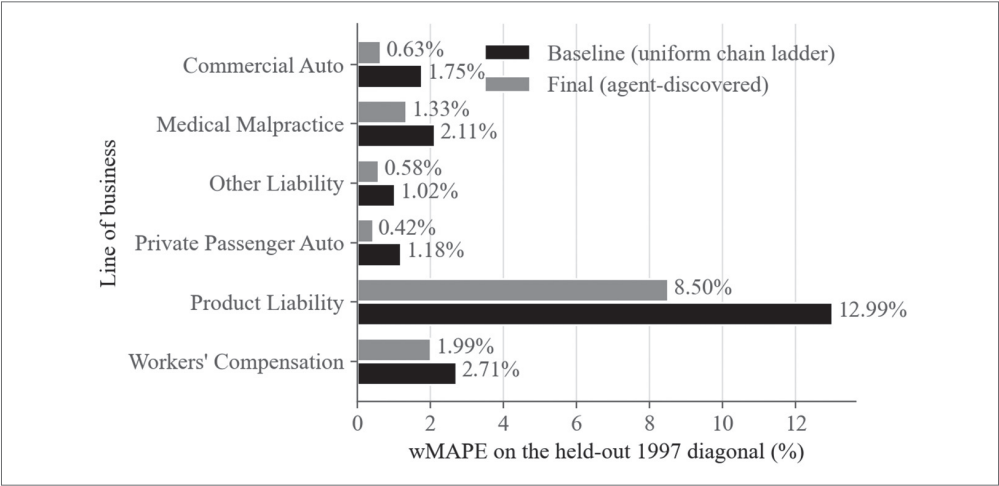


FIGURE 1. Accuracy by line of business on the held-out 1997 diagonal, uniform chain-ladder baseline versus the agent's final model

6.4.4 In reserve terms, the effect on the deployment basis (models fitted on the full upper triangle, projecting ultimates) is shown in Table 9. Amounts in Tables 9 and 10 and in the discussion that follows are in US dollars, the currency of the source database.

TABLE 8. Accuracy by line of business on the selection diagonal, baseline and final model

Line of business	Baseline wMAPE	Final wMAPE	Improvement	Baseline A/E*	Final A/E*
Commercial auto	1.75%	0.63%	64%	0.985	0.996
Medical malpractice	2.11%	1.33%	37%	0.987	1.004
Other liability	1.02%	0.58%	43%	0.996	0.999
Private passenger auto	1.18%	0.42%	64%	0.988	0.996
Product liability	12.99%	8.50%	35%	0.892	0.967
Workers' compensation	2.71%	1.99%	27%	0.974	0.981
Composite	3.63%	2.24%	38.2%	—	—
*Cumulative-paid basis, per section 5.2.					

TABLE 9. Projected ultimates and unpaid claims from the full 1997 upper triangle (US\$ thousands)

Line of business	Baseline ultimate	Final ultimate	Baseline reserve	Final reserve
Commercial auto	8,223,311	8,021,288	1,773,749	1,571,726
Medical malpractice	3,459,288	3,257,110	1,374,954	1,172,776
Other liability	4,910,829	4,964,540	1,688,859	1,742,570
Private passenger auto	121,058,842	119,178,618	17,235,278	15,355,054
Product liability	1,377,220	1,164,031	549,510	336,321
Workers' compensation	13,969,979	13,494,732	2,940,659	2,465,412
Total (US\$ thousands)	152,999,469	150,080,319	25,563,009	22,643,859

6.4.5 Aggregate emergence over the completed triangle

Fitting each model to the 1997 upper triangle, we compared its projected cumulative emergence from the 1997 diagonal to age 120 with the development observed in the completed lower triangle. Because the CLRD ends at age 120, this is not an observed ultimate; development beyond that age is unavailable (Table 10).

TABLE 10. Projected and actual emergence from the 1997 diagonal to age 120 (US\$ thousands)

Line of business	Projected emergence, baseline	Projected emergence, final	Actual emergence	Emergence A/E, baseline	Emergence A/E, final
Commercial auto	1,743,193	1,546,956	1,580,311	0.907	1.022
Medical malpractice	1,330,331	1,137,277	1,158,410	0.871	1.019
Other liability	1,640,597	1,694,680	1,667,562	1.016	0.984
Private passenger auto	17,138,459	15,261,940	15,618,034	0.911	1.023
Product liability	531,649	336,321	532,558	1.002	1.584
Workers' compensation	2,777,813	2,334,399	2,416,210	0.870	1.035
Total (US\$ thousands)	25,162,042	22,311,573	22,973,085	0.913	1.030
(Emergence from the 1997 diagonal to age 120, per the completed lower triangles; A/E = actual ÷ projected emergence.)					

6.4.6 The discovered model reduces the booked total unpaid-claims liability by 11.4% relative to the uniform baseline. On the development observed to age 120, its aggregate emergence A/E is 1.030, compared with 0.913 for the uniform baseline: most of the reduction was borne out by subsequent development.

6.4.7 Product liability is the important exception. The baseline was almost exactly unbiased over the subsequent nine years (emergence A/E 1.002), whereas the discovered model projected US\$336 million against US\$533 million of actual emergence. Its 39% reduction in unpaid claims for this line is therefore not supported by the subsequent development. The unusually poor 1997 selection result for the baseline on this line (its worst per-line score in Table 5) led the agent in the wrong direction: a clear instance of the risk of selecting against an unusual diagonal.

6.4.8 This check does not assess development beyond age 120. The product liability no-tail configuration therefore remains unsuitable as a reserving basis, even though its additional understatement cannot be observed in this dataset.

6.5 Score progression

6.5.1 The composite score trajectory over the primary run's 56 experiments is shown in Figure 2.

6.5.2 The pattern is the familiar one of diminishing returns: the method structure and the first trims delivered most of the gain; everything after Experiment 29 (the flagged no-tail keep, Finding 6) was refinement, and the last two kept changes moved the score by negligible amounts.

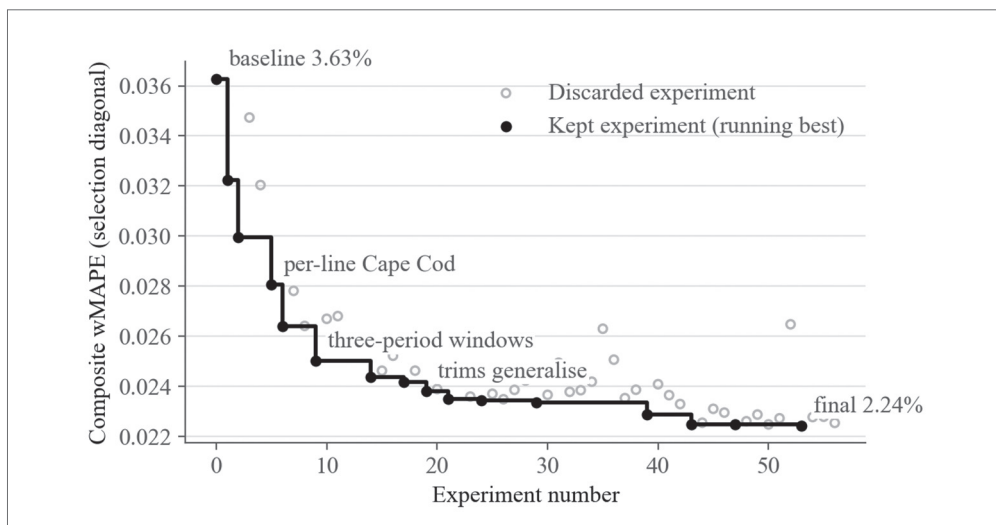


FIGURE 2. The primary run's 56 experiments, with the step line tracing the running best (kept experiments) and open grey markers the discarded attempts

6.6 Key actuarial findings

6.6.1 The experiment log, and particularly the discarded experiments, yielded several observations of independent actuarial interest. We label them ‘findings’ for readability, with the standing caveat that a greedy loop’s findings are conditional on the path that produced them (section 7.5):

- **Finding 1: No single method dominates; the per-line method assignment is the structural discovery.** The run ended with Cape Cod on commercial auto, other liability, and product liability, and the chain ladder on medical malpractice, private passenger auto, and workers’ compensation. This validates the universal actuarial practice of tailoring methods to line characteristics. One caveat: the idea was pre-seeded. The agent’s instruction file explicitly suggests per-LOB strategies (Appendix D), so what the run demonstrates is reliable execution and calibration of the idea, rather than an idea the agent originated.
- **Finding 2: Robust trimmed factors on short windows.** The estimator the run converged on (drop the highest of the three most recent link ratios per column and simple-average the rest) is a robust response to calendar-year trends in the development pattern: it trades a small downward bias for reduced sensitivity to upward outliers among the link ratios, and it suited every line except other liability. For a 10×10 industry aggregate this is a non-obvious selection actuaries can interrogate and test on their own data (section 6.7 shows how much of the selection-diagonal gain survives on untouched data).
- **Finding 3: Factor-window choice was line-specific.** Most lines retained a three-period window, while other liability settled on seven periods and medical malpractice rejected the full history. In this run, tail length did not explain those choices. The result is a reminder to test factor windows by line rather than applying one rule across the portfolio.
- **Finding 4: Cape Cod’s selection-diagonal advantage did not validate for product liability.** On the 1997 selection diagonal, the Cape Cod variant reduced product liability’s apparent over-projection more than the tested BF and Benktander alternatives (Experiments 3, 4, 7, 11, 31). Subsequent run-off gave a different result: its lower projection under-provided actual emergence by 37%, while the uniform chain-ladder baseline was nearly unbiased (sections 6.4 and 6.7). This is an example of why a promising selection result remains a candidate for actuarial review, rather than a booking recommendation.
- **Finding 5: Untuned exposure-based methods did not improve medical malpractice.** Cape Cod worsened the selection score when tested (Experiment 52), and the stability-metric run also finished with the chain ladder for this line (section 6.8). This is evidence of the untuned variants examined here, not of premium-based methods generally; line-specific calibration remains untested. (CLRD losses are net of reinsurance while our exposure base is direct premium, a basis mismatch discussed in section 7.5 that particularly affects this line.)
- **Finding 6: The loop kept an actuarially implausible no-tail model because the selection metric barely tested the tail.** Removing the product liability tail marginally improved the selection score (Experiment 29), though it reduced the line’s reserve estimate. The selection

diagonal tests that assumption only weakly (only the oldest origin year’s final cell reflects it), so the apparent improvement was not a basis for adoption. A simple plausibility rule on the search space, requiring a tail on the long-tail lines, would have blocked the change before it was scored. This illustrates the central governance requirement: the metric and the permitted search space must be reviewed before an autonomous loop is run.

— **Finding 7: Failed experiments are as informative as successful ones.** Of the run’s 41 discards: exposure methods failed workers’ compensation (Experiment 36) and medical malpractice (52); two-period averaging windows ($n = 2$) failed at every probe (15, 16, 35, 51), though the repeat runs showed the same windows can survive in different surrounding configurations, the state-dependence of ¶7.5.6 made concrete; two-sided trims over-trimmed (33, 49); and weighting recent origin years more heavily in the Cape Cod a priori (the decay parameter) mostly worsened the score (12, 30, 34, 55, 56). These negative results narrow the viable space; they are also the content a reviewing actuary needs to challenge the final selection.

6.7 Beyond the selection diagonal: rolling hold-outs, true run-off, and a practitioner benchmark

The checks in this section are validation, not a further round of selection and not a contest between search methods. They ask whether the model produced by the documented search was broadly sensible on data the agent never touched, and where it was not. Alongside the uniform control, which is a deliberately simple yardstick rather than a representation of practice (section 5.4), the rolling hold-out and run-off checks include one pre-specified practitioner-style default: Cape Cod on every line with standard untuned parameters (all-period volume-weighted development, exponential tail, a pooled a priori with decay 0.6), specified before scoring.

6.7.1 Rolling hold-outs

Scoring the same configurations on earlier, untouched diagonals shows how much of each headline is selection-set optimism (Table 11).

TABLE 11. Composite wMAPE on the selection diagonal and on two earlier untouched diagonals

Configuration	1997 diagonal (selection set for the run)	1996 diagonal	1995 diagonal
Uniform chain-ladder baseline	0.0363	0.0350	0.0270
Practitioner default: all-lines Cape Cod	0.0283	0.0350	0.0264
Primary final	0.0224	0.0338	0.0243

6.7.2 The final model’s 38% improvement on its selection diagonal shrinks to roughly 3% on 1996 and 10% on 1995: consistent with the winner’s curse expected from

repeated adaptive selection against the same 54 cells, and the practitioner default’s selection edge over the baseline disappears on 1996 altogether. Improvements measured on a single diagonal, including those reported here, overstate what generalises; and very small selection-score differences, such as those between the run’s last kept experiments, should not be over-interpreted at all, particularly after repeated adaptive search. (Mechanical searches of the same space, reported in the working papers, reach far stronger selection scores than the agent and no better out-of-sample accuracy; we do not pursue that comparison here, because the question this paper asks is not which searcher performs better.)

6.7.3 *True run-off: validation against actual subsequent development*

The CLRD was built for retrospective testing: the Casualty Actuarial Society publishes the completed triangles for these same companies, with actual development to the end of calendar year 2006. We aggregated them per line (the upper triangles reconcile cell-for-cell with the harness’s data), fitted each candidate on the full 1997 upper triangle (the deployment scenario), and compared projections against the development that actually emerged (Table 12).

TABLE 12. Selection score, true run-off error and aggregate emergence for the three candidates

Configuration	Selection wMAPE (1997)	Mean per-line run-off wMAPE, all 45 unobserved cells	Aggregate emergence A/E (1997 diagonal → age 120)
Uniform chain-ladder baseline	0.0363	4.93%	0.913 (over-projects by about 9%)
Practitioner default: all-lines Cape Cod	0.0283	4.11%	0.907 (over-projects by about 10%)
Primary final	0.0224	4.24%	1.030 (under-projects by about 3%)

6.7.4 Figure 3 plots the selection score against the true run-off error for the three candidates.

6.7.5 Three conclusions follow (Figure 3). First, the agent’s model was broadly sensible on subsequent development: it outperformed the deliberately simple control on nine years of actual run-off (4.24% against 4.93%) and largely removed that control’s 9% aggregate over-projection bias. Second, a sensible fixed actuarial default performed similarly or slightly better: the untuned all-lines Cape Cod reached 4.11% on run-off (4.06% with decay 0.8), although its aggregate over-projection bias remains essentially the baseline’s (A/E 0.907). The agent’s converged structure, Cape Cod on the liability lines and commercial auto with the chain ladder elsewhere, is itself a version of that practitioner consensus, discovered and calibrated rather than assumed. Third, the aggregate hides a per-line failure that the selection diagonal missed: on product liability the baseline’s projection of nine years of actual emergence was almost exactly unbiased (A/E 1.002) while the final model under-provided it by 37% (per-line run-off wMAPE 13.5% against the baseline’s 10.6%), and the final model’s healthy aggregate A/E of 1.030 netted opposite-signed line

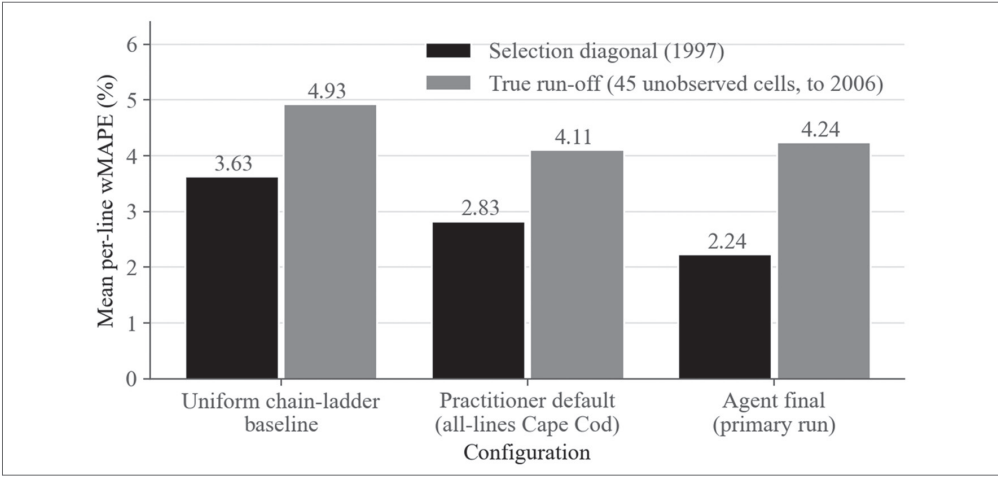


FIGURE 3. Selection-diagonal score versus true run-off error for the uniform baseline, a pre-specified practitioner-style default and the agent’s final model

biases (section 6.4). This is why a human reviewer must not treat a selection result as a booking recommendation. One limitation applies to the test itself: it evaluates only to age 120, where tails have no effect, so the harm of the flagged no-tail exploit is invisible even to this, the strongest validation the data permit; metrics that penalise implausible tail behaviour explicitly are needed for that. The practice implication is direct: the agentic loop should be used to generate and document candidate models, and the selection among them should then be made on data the agent never touched, ideally true run-off read per line and not only in aggregate, with the agent’s journal as working papers.

6.7.6 Two further pre-specified practitioner variants make a calibration point by contrast: a flat 0.70 BF a priori on the long-tail lines performs worse than the baseline on these data (run-off 9.4%), while the identical basis with a priori loss ratios calibrated from training data recovers to 5.8%. The gap between uncalibrated and calibrated practice far exceeds the gap between calibrated practice and any optimised candidate. As context of a different kind, the industry’s own booked year-end-1997 positions, tested against the same subsequent development in the retrospective tradition of Meyers & Shi (2011), exceeded the age-120 paid outturn by 2% to 3% in aggregate, with the conservatism concentrated in the long-tail lines; booked incurred includes development beyond age 120 that neither the models nor the paid actuals capture, so this brackets rather than ranks the candidates. What the agentic route adds over booking a sensible default is the evidence file, not necessarily a more accurate model.

6.7.7 *Repeatability*

The evaluator is deterministic (any logged configuration re-scores to six decimal places from an independent code path), while the discovery trajectory is stochastic. The

protocol imposes no experiment cap (the loop runs until interrupted); the primary run was stopped after its final probes were all discarded (section 6.3), and four further runs under the same frozen protocol were stopped at 13 to 20 experiments in the same outcome band, with no drift towards extreme configurations. Those runs reached selection composites of 0.0216 to 0.0220 against the primary run's 0.0224, and every run outperformed the uniform baseline on true run-off (3.93% to 4.66% versus 4.93%), with emergence A/E of 1.03 to 1.05. At the method level, all five trajectories converged to the same per-line assignment (Cape Cod on the liability lines and commercial auto, chain ladder elsewhere), with parameter-level detail varying by trajectory (run journals): repeatable in structure, though no individual trajectory is exactly re-executable (section 7.5). The procedure's product is accordingly best read as a set of documented candidates from which the final model is selected using untouched data, rather than as a single run assumed to identify the best model.

6.7.8 Entity-level transfer

We also tested the baseline and discovered model on the five largest writers in each line. The discovered model performed better on the selection diagonal for 23 of the 30 triangles, but its performance did not transfer reliably to the nine-year run-off data. It outperformed the baseline for only 17 triangles, and its median nine-year run-off wMAPE across the 30 triangles was 10.8%, compared with 8.5% for the baseline. The aggregate results therefore demonstrate the framework but do not support applying the selected configurations to individual insurers. Testing the loop directly on an insurer's own data remains future work.

6.8 Case study: the selection metric is a design choice

6.8.1 A natural alternative to selecting on accuracy against the next diagonal is to select on the one-year stability of the ultimates: the CDR of Merz & Wüthrich (2008), used as a backtesting criterion by Balona & Richman (2020), and the natural quantity under the SAM one-year risk horizon. To test how strongly the metric choice shapes the discovered model, we re-ran the loop with the selection metric changed accordingly.

6.8.2 CDR selection (the stability-metric run)

The selection metric became the mean, over the two observable calendar-year transitions, of the wCDR per LOB, equal-weighted across lines:

$$\text{wCDR} = \frac{\sum_i |U_i(t) - U_i(t+1)|}{\sum_i \ddot{R}_i(t)},$$

where $U_i(t)$ is the model's estimated ultimate for origin year i at valuation year t and $R_i(t)$ is its held reserve for that origin year at the same date; the metric measures how far the

model's ultimates move, relative to held reserves, when one more diagonal arrives. The direction of the outcome was expected (the CDR rewards stability, and the accuracy–stability trade-off is inherent to the criterion), so the case study's purpose was to measure the size and structure of that trade-off, using a second objective as a control experiment on the loop. Three run-level hypotheses were stated in the run journal before the run began (per-experiment hypotheses continued under the standard protocol), and all three held: fixed-a-priori BF, which is inferior under the accuracy metric, dominates under the stability metric (mean wCDR from 0.210 to 0.101 in one move); the short recency windows that drive the accuracy-optimal models hurt stability (rejected at every probe); and the converged CDR-optimal model (BF with a 0.60 a priori on every line except medical malpractice, full-history simple factors, and no tail) shares almost no architecture with the accuracy-optimal models. The case-study run converged in 15 experiments (five kept, its final three probes all discarded): its purpose is to demonstrate that the discovered model is a function of the metric choice, not to re-optimize exhaustively under a second objective. Two invariants stand out: medical malpractice rejected exposure-based methods under both metrics (a further confirmation of Finding 5), and both selection metrics rewarded tail deletion. The accuracy metric is barely sensitive to the tail assumption (only the final observed cell of the oldest origin year reflects it), while under the stability metric the tail's year-to-year refit movement flows straight into every ultimate, so deleting it mechanically stabilises them. In both cases, the metric favoured the same actuarially unsound model change for a different reason.

6.8.3 The cross-metric scorecard (Table 13) makes the trade-off concrete. The CDR-optimal model is the most stable and the least accurate model produced in this study, which is the direction the criterion guarantees. What the definition does not supply is the structure and size of the trade-off, and three measurements carry the section's findings.

6.8.4 Table 13 shows an asymmetric trade-off.

6.8.5 The accuracy-selected model also improved stability, reducing mean wCDR from 0.210 to 0.135. By contrast, the stability-selected model had worse run-off accuracy than the baseline (6.77% compared with 4.93%), despite having aggregate emergence close to expectation (A/E 0.978). Stability can therefore be achieved by making estimates less responsive to new observations.

TABLE 13. Cross-metric scorecard for the accuracy-selected and stability-selected models

Model	Selection wMAPE (1997)	Mean wCDR	True run-off wMAPE (45 cells)	Emergence A/E
Uniform chain-ladder baseline	0.0363	0.210	4.93%	0.913
Primary final	0.0224	0.135	4.24%	1.030
CDR-optimal	0.0362	0.083	6.77%	0.978

6.8.6 During the run, the agent tested whether changes in wCDR were caused simply by changes in the reserve denominator. Three experiments and a subsequent controlled sweep did not support that explanation at the aggregate level. The lowest wCDR occurred at an a priori loss ratio of 0.60, close to the data-implied ratio of 0.57, rather than at the upper end of the tested range. Product liability was an exception: its best result occurred at the upper boundary of 0.90 despite a data-implied ratio of 0.50, suggesting that the denominator influenced the result for that line.

6.8.7 These results support selecting models using out-of-sample accuracy while imposing a stability constraint, or reporting both measures. The evidence is limited, however, because only two calendar-year transitions were available and the hypotheses were recorded internally by the same agent conducting the run.

7. DISCUSSION: WHAT DOES THIS MEAN FOR RESERVING ACTUARIES?

7.1 The actuary as research director

7.1.1 The principal conceptual shift is in the actuary's role. In the traditional workflow, the actuary:

- selects a method (chain ladder, BF, Cape Cod);
- selects parameters (which periods, which averaging);
- runs the model;
- reviews the output; and
- exercises judgement to adjust.

7.1.2 In the autoresearch workflow, the actuary:

- defines the evaluation metric (what 'good' means);
- defines the search space (what methods and parameters are acceptable);
- writes the agent instructions (how to explore);
- reviews the experiment log (what worked and why); and
- exercises judgement on the final selection.

7.1.3 The actuary's role shifts from implementation to direction. The mechanical work of running hundreds of model variations is delegated to the agent. The actuarial judgement (defining what to optimise, what constraints to impose, and how to interpret results) remains with the human.

7.1.4 Before a HAF, or another actuary responsible for the technical opinion provision, could rely on an agent-selected method for technical provisions, or before an external auditor could obtain sufficient appropriate audit evidence over amounts derived from that method, we would regard the following as minimum conditions, mapped onto the actuarial control cycle:

- (1) a locked, independently tested evaluation harness with the held-out data isolated from the agent by construction rather than by convention (section 5.1 explains why);

- (2) a versioned, reproducible run record identifying the agent model, its instructions at a commit hash, the environment, and the full experiment log;
- (3) selection-set and confirmation-set performance with uncertainty attached, including validation against actual run-off where the data allow;
- (4) independent peer review of the metric choice and of the kept experiments;
- (5) a documented rationale for accepting or overriding the agent's selection; and
- (6) independent review and change control of the agent instruction file, since Appendix D shows that it shapes the adopted model structure and delimits the search space.

Accountability for the actuarial opinion remains with the responsible actuary, while the external auditor remains responsible for any audit opinion covering the reported technical provisions; neither responsibility can be transferred to the agent. The agent's journal is best understood as working papers of unusual completeness.

7.2 Enhanced method selection evidence

7.2.1 One of the most common challenges in reserve reviews is justifying method selection: why the chain ladder rather than BF, volume-weighted rather than simple averaging, or one tail curve rather than another. The autoresearch experiment log provides empirical evidence for method selection. Instead of 'we selected the chain ladder because it is the standard approach', the actuary can say 'we evaluated 56 configurations across six lines of business; a trimmed three-period chain ladder produced the lowest actual-versus-expected error on held-out data for medical malpractice, private passenger auto and workers' compensation, while Cape Cod served the liability lines and commercial auto. Because those errors were measured on the same diagonal used to select the model, they overstate the improvement. Checked against the nine years of development that subsequently emerged, the selected model still outperformed the uniform baseline on mean per-line error, though by a smaller margin than the selection figures suggest, and with one line (product liability) where the baseline proved better. The full experiment log, including 41 discarded experiments that demonstrate why alternatives were rejected, and the per-line detail of that check, are attached as working papers.'

7.2.2 This does not replace actuarial judgement: the actuary must still assess whether the held-out evaluation is representative, whether the winning method is appropriate for the current environment, and whether qualitative factors override the quantitative ranking. But it provides a stronger quantitative foundation for that judgement. The benefit is not unique to an LLM agent: any logged search, including a deterministic sweep, produces a record; what the agent adds to the record is the stated hypothesis and negative-result narrative a reviewer can challenge.

7.3 Continuous learning and adaptation

7.3.1 Method reviews are costly, so their depth and frequency are budget-limited: practice ranges from annual method reviews to the quarterly reviews several large South African non-life insurers already run, while the technical provisions themselves are produced quarterly under SAM, typically with actual-versus-expected monitoring in each cycle. The autoresearch pattern allows method-level re-evaluation at the cadence of the valuation itself, closing the gap where reviews are less frequent and deepening the evidence base per review where they are not: as new diagonals emerge, the agent can re-run the experiment loop to test whether the previously selected method still performs best.

7.3.2 This is particularly relevant for lines of business undergoing structural change, where historical development patterns can stop being predictive and the best-performing method can change between valuations. Recent South African examples include social inflation in liability lines, the Covid-19 contingent business interruption judgments, catastrophe reinsurance repricing after the April 2022 KwaZulu-Natal floods, and Road Accident Fund developments affecting motor books. In such conditions, re-testing method performance against emerging experience at each valuation identifies a deteriorating method sooner than a less frequent review cycle would. We stress, however, that the current implementation does not detect structural change: it minimises error on a small hold-out, which cannot distinguish regime change from noise, and unconstrained re-selection each period would risk spurious method churn; our own repeat runs, while agreeing on method structure, converged to different parameter-level configurations on identical data (section 6.7's repeatability paragraph), differences a per-period re-selection rule would book as method changes. Turning the loop into a monitoring design, with rolling hold-outs, stability constraints, and significance thresholds before a method change is accepted, is future work.

7.4 Extending the method space: the methods an actuary rarely has time to test

7.4.1 The runs reported here stayed within the classical toolkit: development factor selection, tail extrapolation, and the chain ladder, BF, Benktander and Cape Cod methods. That is where most reserving practice also stays, and not because the alternatives are unknown. The same library implements several methods that appear in the literature and the syllabus but seldom in a valuation file: Clark's parametric growth-curve models, which fit a loglogistic or Weibull development curve by maximum likelihood and supply their own tail and standard errors (ClarkLDF; Clark, 2003); the Munich chain ladder, which corrects paid and incurred projections towards each other using the observed relationship between paid-to-incurred ratios and subsequent development (MunichAdjustment; Quarg & Mack, 2004); the case-outstanding development technique, which projects unpaid claims from the run-off of case reserves rather than from paid development alone (CaseOutstanding; Friedland, 2010); GLMs fitted directly to the incremental triangle, of which the over-dispersed Poisson model is the best-known member (TweedieGLM;

England & Verrall, 2002); and machine-learning interfaces and voting ensembles that combine methods with per-origin weights (DevelopmentML, VotingChainladder). The CLRD carries incurred as well as paid losses, so the data support all of them. Each requires time to implement correctly, understanding to parameterise, and an audit of why it behaves as it does on the book in question. Under a valuation deadline those costs are rarely paid, so the methods go untested and the method-selection file records a choice among three or four familiar candidates.

7.4.2 This is where the agentic loop changes the economics of the exercise rather than its accuracy. The agent can add a further method to the same evaluation harness, fit it under the same frozen protocol, score it against the same held-out data, and write down the hypothesis, the result, and its reading of why the method did or did not suit the line, all in the minutes an experiment takes (§4.3). A method that fails is recorded as a documented negative result rather than left as an untested option; a method that helps arrives with a stated actuarial reason the reviewing actuary can challenge. The evidence file thereby covers ground a manual review could not afford to cover, and the actuary's understanding of the book is extended by the methods that were previously too expensive to try. The actuary remains the research director throughout: the brief decides which methods are in scope, and the metric decides what counts as helping.

7.4.3 Two cautions carry over from the classical results. First, the wider the space, the more the metric matters: a paid-only, one-step accuracy metric barely tests the tail (Finding 6) and cannot see the paid-versus-incurred information that the Munich chain ladder and the case-outstanding method are designed to exploit, so a run over the wider space needs a metric that scores what those methods change, for example emergence over several diagonals and against incurred as well as paid. Second, the more flexible the method, the larger the winner's curse (section 6.7): a GLM or an ensemble tuned to one diagonal fits it better than a chain ladder does and generalises no better, so selection among the candidates belongs on untouched data, as in section 6.7. Our own runs did not reach these methods (section 5.4), so this paper makes no claim about how they perform on these data. A run that starts from the primary run's final configuration, with the research brief re-prioritised towards the less-used methods and a multi-diagonal, paid-and-incurred metric, is the natural next study, and the one in which the agent's reach beyond a parameter grid would be tested directly.

7.5 Limitations and caveats

We emphasise several important limitations.

7.5.1 *Overfitting to the hold-out*

Optimising for a single held-out diagonal risks selecting a method that happened to perform well in that specific period but may not generalise. With every keep-or-discard decision scored against the same 54 hold-out cells, the final headline is adaptively

optimistic, a winner's curse of model selection, quantified in section 6.7: the run's 38% selection-diagonal improvement shrinks to roughly 3% to 10% on earlier untouched diagonals. Robustness checks across multiple hold-out periods are essential; very small differences in the score of a particular selection should not be treated as improvements at all; and headline improvements should be confirmed on data the agent never touched (for this database, the published lower triangles).

7.5.2 *The metric is not the objective*

Minimising wMAPE on historical data is a proxy for the true objective: producing accurate reserve estimates for the current period. Environmental changes, regulatory shifts, and portfolio composition changes may make historical performance an imperfect guide.

7.5.3 *Judgement cannot be automated*

The AI agent can identify which method produces the lowest error on historical data. It cannot assess whether the current environment is similar to history, whether there are qualitative factors that should override the quantitative ranking, or whether the winning method produces reserves that are defensible to regulators and auditors. Supplying that context explicitly, for example through the agent's instruction file, can help the agent reflect known environmental changes and qualitative constraints in what it proposes; the assessment itself remains the actuary's.

7.5.4 *Transparency and auditability*

While the experiment log is interpretable, the agent's reasoning about why it chose each experiment is not always transparent. It remains unclear what constitutes adequate human review of dozens of machine-generated experiments.

7.5.5 *Data quality*

The agent optimises within the data it is given. Data errors, late-reporting changes, or portfolio transfers that distort the triangle lead the agent to optimise for a flawed picture.

7.5.6 *Path-dependence of greedy search*

The keep-or-discard loop is a greedy single trajectory: each 'finding' is conditional on the configuration state in which it was tested. Repeat runs of the same protocol on identical data (§6.7.8) converged to the same per-line method structure with parameter-level detail varying by trajectory, landing in a selection band of 0.0216 to 0.0224 and beating the baseline on every readout; they also showed the interaction risk plainly: the two-period averaging windows the primary run rejected on their own survived elsewhere in different surrounding configurations. Findings from any single trajectory are observations, not general rules; run-level repetition is as necessary here as in any stochastic optimisation, and the selection among candidates belongs to independent data (section 6.7).

7.5.7 *Training data contamination and carried-over knowledge*

The CLRD is public, is distributed with `chainladder-python`, and has been used in published benchmark studies. The language model may therefore have encountered the data, including the 1997 selection diagonal, during pre-training. This cannot alter the reported scores calculated by the frozen evaluation harness without a language model in the scoring process. It could, however, have influenced which experiments the agent chose.

7.5.7.1 We found no evidence that the agent recalled or used CLRD-specific results. None of its 56 recorded hypotheses cited published findings about the dataset; the agent discarded 41 experiments, achieved weaker selection scores than mechanical searches of the same space, and generalised no better than a fixed practitioner default. More general influence from pre-training cannot be ruled out, because familiarity with methods that tend to work on claims triangles is indistinguishable from the actuarial knowledge the approach is intended to use. For this reason, the paper does not claim that the agent originated the ideas it tested. Here, ‘discovery’ means the documented process of proposing and testing them under a frozen metric and a human-written brief (section 1.5, Appendix D).

7.5.7.2 The primary run also began with project-specific context: the agent had read the earlier development history, and its instruction file explicitly encouraged different strategies by line of business. Tailoring methods by line is a conventional actuarial idea, but the run should therefore not be credited with originating it. This project-specific context is distinct from possible pre-training contamination; it means that the reported accuracy-metric runs are best understood as tests and calibration of a seeded strategy, not independent replications of the discovery process.

7.5.8 *Exposure-basis mismatch*

CLRD losses are net of reinsurance, while our exposure base is direct earned premium; the implied a priori loss ratios of the premium-based methods therefore embed year- and line-varying cession distortions. This particularly affects the medical-malpractice result (Finding 5), where cession rates varied materially across the era. A sensitivity re-run with net earned premium moved the primary final’s composite from 0.0224 to 0.0228, confirming that the basis choice affects the result; matching the basis remains the theoretically correct choice, and it is a one-line harness change for future runs.

7.5.9 *Point estimates only*

The framework optimises central estimate accuracy and produces no ranges, prediction intervals, or distributional diagnostics, although Mack’s standard errors (Mack, 1993) and bootstrap predictive distributions are available in the same library. For a jurisdiction whose statutory basis requires a risk margin (SAM) and whose accounting basis requires a risk adjustment (IFRS 17), that is a substantive gap: a method selected purely on central

estimate accuracy may not have the best-behaved error distribution, and the actuarial loss function is asymmetric besides. Scoring candidates additionally on distributional calibration against emerged actuals is the natural extension.

7.6 Broader applications

While this paper focuses on reserving, the autoresearch pattern is applicable to other actuarial problems:

- **Pricing:** autonomous exploration of GLM specifications, territory definitions, and rating-factor interactions;
- **Capital modelling:** exploring copula structures, distribution fits, and dependence assumptions;
- **Mortality and longevity:** exploring mortality improvement models, basis-risk adjustments, and experience analysis configurations; and
- **Reinsurance optimisation:** searching over programme structures, attachment points, and treaty terms.

Each of these domains shares the key requirements: a well-defined metric, a structured search space, and fast evaluation.

8. CONCLUSION

8.1 We have demonstrated that the autonomous AI research paradigm, an AI agent iterating on code, evaluating results, and keeping improvements, can be meaningfully applied to actuarial reserving. In one complete run of 56 experiments, executed in about 12 minutes of commit-to-commit time under a frozen, unit-tested evaluation harness, the agent arrived at a per-line method selection a practising actuary would recognise (Cape Cod on the liability lines and commercial auto, a trimmed short-window chain ladder elsewhere), with each step recorded as a hypothesis, a change, and a result. The discovered model was broadly sensible on nine years of actual subsequent development: better than the uniform chain-ladder control, and comparable to a sensible untuned practitioner default. The selection score overstated what transfers, however, and on the single line that drove the search, product liability, the run-off data sided with the baseline (sections 6.4 and 6.7). The run's distinctive product is accordingly not the model but the evidence file (every hypothesis, calibration, and negative result documented) and the metric governance findings, and that is the basis on which we commend the paradigm. The framework rests on four elements that allow a documented method selection search (56 hypothesis-driven experiments here) to be executed and logged in minutes:

- modern LLM capabilities (code understanding, long context, tool use, agentic execution);
- a well-structured actuarial problem (finite method space, interpretable building blocks, clear evaluation metric);

- mature software infrastructure (`chainladder-python`, scikit-learn API conventions); and
- fast evaluation (seconds per experiment, enabling rapid iteration).

8.2 The experiment log, documenting what was tried, what worked, and what did not, is itself a valuable research artefact. The run produced seven key actuarial observations (section 6.6): among them the per-line method assignment, the trimmed short-window estimator as a robust answer to calendar-year trends, and medical malpractice’s rejection of untuned exposure methods, each one interrogable against the run journal. The failed experiments were as informative as the successful ones, and the two failure modes the run itself surfaced (a converged agent exploiting the metric’s insensitivity to the tail assumption, and selection gains that largely did not transfer to data the agent never touched) teach the paper’s central lesson: an autonomous loop optimises the metric as implemented, so metric design, error bars, and confirmation on untouched data must come before any agentic reserving deployment, not after it.

8.3 We do not suggest that AI will replace the reserving actuary. The actuary’s role in defining objectives, imposing constraints, interpreting results, and exercising professional judgement remains essential. What changes is the scale of exploration that supports that judgement. The actuary who previously evaluated three to five method variations now has evidence from dozens or hundreds, including methods that were previously too costly to test (section 7.4). The method selection that was previously based on convention and limited testing is now grounded in systematic, documented experimentation.

8.4 The tools for autonomous AI-assisted actuarial research are already available, the software ecosystem is mature, and the framework requires modest implementation effort (three short files built on an open-source library). The profession’s task now is to develop best practices for integrating these tools into workflows, standards, and governance.

8.5 The front page of Karpathy’s (unpublished) repository closes with a satirical passage: ‘Research is now entirely the domain of autonomous swarms of AI agents ... This repo is the story of how it all began.’ The contribution available to reserving actuaries today is more modest and more useful: a documented, independently verifiable method selection evidence file, produced in minutes, under governance disciplines the profession already knows how to apply.

REFERENCES

- Actuarial Society of South Africa (ASSA) (2022). APN 403: Head of Actuarial Function for South African Insurers, version 2. ASSA, Cape Town. https://www.actuarialsociety.org.za/wp-content/uploads/2025/07/APN-403-V2_-HEAD-OF-ACTUARIAL-FUNCTION-FOR-SOUTH-AFRICAN-INSURERS-3.pdf (Retrieved 16 July 2026)
- Balona, C & Richman, R (2020). The actuary and IBNR techniques: a machine learning approach. *SSRN Electronic Journal*. <https://dx.doi.org/10.2139/ssrn.3697256>
- Casualty Actuarial Society (2025). Loss reserving data pulled from NAIC Schedule P. Casualty Actuarial Society, Arlington, <https://www.casact.org/publications-research/research/research-resources/loss-reserving-data-pulled-naic-schedule-p> (Retrieved 4 July 2026)
- Casualty Actuarial Society (2026). Chainladder-python documentation. Casualty Actuarial Society, Arlington, <https://chainladder-python.readthedocs.io/> (Retrieved 4 July 2026)
- Clark, DR (2003). LDF curve-fitting and stochastic reserving: a maximum likelihood approach. *Casualty Actuarial Society Forum*, Fall 2003, 41–92
- England, PD & Verrall, RJ (2002). Stochastic claims reserving in general insurance. *British Actuarial Journal* 8(3), 443–518. <https://dx.doi.org/10.1017/S1357321700003809>
- Friedland, J (2010). *Estimating Unpaid Claims Using Basic Techniques*, version 3. Casualty Actuarial Society, Arlington, https://www.casact.org/sites/default/files/database/studynotes_friedland_estimating.pdf (Retrieved 4 July 2026)
- Karpathy, A (unpublished). Autoresearch. GitHub repository, 2026, <https://github.com/karpathy/autoresearch> (Retrieved 4 July 2026)
- Mack, T (1993). Distribution-free calculation of the standard error of chain ladder reserve estimates. *ASTIN Bulletin* 23(2), 213–25 <https://dx.doi.org/10.2143/AST.23.2.2005092>
- Merz, M & Wüthrich, MV (2008). Modelling the claims development result for solvency purposes. *Casualty Actuarial Society E-Forum*, Fall 2008, 542–68
- Meyers, G & Shi, P (2011). The retrospective testing of stochastic loss reserve models. *Casualty Actuarial Society E-Forum*, Summer 2011, 1–37, <https://www.casact.org/abstract/retrospective-testing-stochastic-loss-reserve-models> (Retrieved 16 July 2026)
- Novikov, A, Vü, N, Eisenberger, M, Dupont, E, Huang, P-S, Wagner, AZ, Shirobokov, S, Kozlovskii, B, Ruiz, FJR, Mehrabian, A, Kumar, MP, See, A, Chaudhuri, S, Holland, G, Davies, A, Nowozin, S, Kohli, P & Balog, M (unpublished). AlphaEvolve: a coding agent for scientific and algorithmic discovery. arXiv:2506.13131, 2025, <https://arxiv.org/abs/2506.13131>. <https://dx.doi.org/10.48550/arXiv.2506.13131> (Retrieved 15 July 2026).
- Quarg, G & Mack, T (2004). Munich chain ladder: a reserving method that reduces the gap between IBNR projections based on paid losses and IBNR projections based on incurred losses. *Blätter der DGVFM* 26(4), 597–630 <https://dx.doi.org/10.1007/BF02808969>
- Wüthrich, MV & Merz, M (2008). *Stochastic Claims Reserving Methods in Insurance*. John Wiley & Sons, Chichester

APPENDIX A

MODEL CONFIGURATIONS

Baseline (Experiment 0):

```
DEV_N_PERIODS = - 1          # All available periods
DEV_AVERAGE = "volume"      # Volume-weighted averaging
# drop_high / drop_low: if True, drop the highest / lowest
# link ratio in each development column before averaging
DEV_DROP_HIGH = False
DEV_DROP_LOW = False
TAIL_TYPE = "curve"
TAIL_CURVE = "exponential"
METHOD = "chainladder"
```

Final agent-discovered model of the primary run (Experiment 53; the run's remaining probes were all discarded, so this is the converged state):

```
DEV_N_PERIODS = 3           # 3 recent periods (global default)
DEV_AVERAGE = "simple"      # Simple average of link ratios
DEV_DROP_HIGH = False      # (set per line below)
DEV_DROP_LOW = False
TAIL_TYPE = "curve"
TAIL_CURVE = "exponential" # Exponential tail curve (global)
METHOD = "chainladder"     # global default (medmal, ppauto, wkcomp)
CC_TREND = 0.0
CC_DECAY = 1.0 # Cape Cod a priori pooling (1.0 = all years pooled)
# Per-LOB method selection
LOB_METHOD_OVERRIDES = {
    "comauto" : "capecod" , # Cape Cod, pooled a priori (decay=1.0)
    "othliab" : "capecod" , # Cape Cod, decay=0.8 (below)
    "prodliab" : "capecod" , # Cape Cod, pooled a priori (decay=1.0)
}
# Per-LOB development overrides
LOB_DEV_OVERRIDES = {
    "prodliab" : { "drop_high" : True , "n_periods" : 4 },
    "wkcomp" : { "drop_high" : True },
    "comauto" : { "drop_high" : True },
    "ppauto" : { "drop_high" : True },
    "medmal" : { "drop_high" : True },
    "othliab" : { "n_periods" : 7 }, # long window, no trim
}
```

```

# Per-LOB IBNR parameter overrides
LOB_IBNR_OVERRIDES = {
    "othliab" : { "cc_decay" : 0.8 }, # late refinement (section 6.3)
}
# Per-LOB tail overrides
LOB_TAIL_OVERRIDES = {
    "prodliab" : "none" ,           # FLAGGED metric exploit (Finding 6);
                                   # not a recommended reserving basis
}

```

APPENDIX B

EVALUATION METRIC DEFINITION

B.1 The composite score S is the average wMAPE across the six lines of business:

$$S = \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} \text{wMAPE}_{\ell},$$

where $\mathcal{L}$ is the set of six lines of business (section 5.3). One wMAPE is first calculated for each line by summing the absolute actual-versus-expected errors across its nine scored origin years and dividing by the line's total absolute actual losses across those years:

$$\text{wMAPE}_{\ell} = \frac{\sum_{i \in \mathcal{I}} |A_{\ell,i} - E_{\ell,i}|}{\sum_{i \in \mathcal{I}} |A_{\ell,i}|}$$

B.2 Here, $\mathcal{I}$ is the set of nine scored origin years, those strictly before the valuation year: the valuation-year origin has no training data and cannot be projected (section 5.2). The actual values $A_{\ell,i}$ are the cumulative paid losses at the held-out valuation date. The expected values $E_{\ell,i}$ are the model's projected cumulative paid positions for that diagonal, from a fit to the truncated triangle (see the cumulative basis caveat in section 5.2). Thus, over- and under-projection in different origin years do not offset each other. If any line fails to produce finite expected values for its scored origins, the composite is non-finite and the experiment is discarded. The composite is an equal-weighted average across the six LOBs: each line is treated as an equally important test problem for method research, regardless of its claim amount. This is a deliberate choice with consequences (an amount-weighted composite would be dominated by private passenger auto and would rank configurations differently), and readers should note that the equal weighting steered the agent's effort towards the highest-percentage-error line (product liability) even though it is economically the smallest.

APPENDIX C

COMPLETE EXPERIMENT LOG OF THE PRIMARY RUN

Table 14 lists every experiment, kept or discarded, as recorded in the run's `results.tsv` log; the agent's hypothesis, configuration change, and result for each (with per-LOB results from Experiment 6 onwards) are in its `experiment_journal.md`. Experiments 8, 20 and 25 as run differed from their stated intent: one changed a parameter that the line's selected method does not use, so the change had no effect, and two unintentionally replaced, rather than added to, settings kept from earlier experiments. Each was re-tested cleanly afterwards (Experiment 8 as Experiment 11, 20 as 21, and 25 as 26 and again as 46) and all rows are retained here. Rows are quoted from the log, and use the database's short line-of-business labels (section 5.3); two descriptions (Experiments 5 and 52) are lightly edited to remove internal shorthand.

TABLE 14. Complete experiment log of the primary run

#	Score	Status	Description
0	0.036275	keep	baseline (volume-weighted chain ladder, exponential tail curve)
1	0.032224	keep	CapeCod IBNR method (all LOBs)
2	0.029964	keep	per-LOB: CapeCod for prodliab only
3	0.034734	discard	per-LOB: BF (apriori 0.70) for prodliab only
4	0.032038	discard	per-LOB: Benktander (n=2, apriori 0.70) for prodliab
5	0.028065	keep	per-LOB: CapeCod for comauto+othliab+prodliab
6	0.026412	keep	prodliab drop_high=True
7	0.027812	discard	prodliab Benktander apriori=0.50, n_iters=3
8	0.026412	discard	prodliab Benktander apriori=0.60, n_iters=3
9	0.025010	keep	n_periods=3 globally
10	0.026696	discard	n_periods=5 globally
11	0.026804	discard	prodliab Benktander apriori=0.60 n_iters=3 (drop_high retained)
12	0.025557	discard	CapeCod decay=0.8 (comauto/othliab, and prodliab if still CC [Cape Cod])
13	0.025925	discard	medmal uses all development periods
14	0.024383	keep	wkcomp drop_high=True
15	0.024623	discard	ppauto n_periods=2
16	0.025222	discard	wkcomp n_periods=2
17	0.024186	keep	comauto drop_high=True
18	0.024626	discard	othliab drop_high=True
19	0.023800	keep	ppauto drop_high=True
20	0.023890	discard	medmal drop_high=True
21	0.023504	keep	medmal drop_high=True (added to the four kept trims)
22	0.024349	discard	prodliab tail = inverse_power (per-LOB override)

23	0.023590	discard	regression averaging globally
24	0.023426	keep	simple averaging globally
25	0.023701	discard	prodliab n_periods=5
26	0.023483	discard	prodliab n_periods=5 (all five trims retained)
27	0.023859	discard	othliab drop_high=True (added under simple averaging)
28	0.024236	discard	medmal n_periods=-1 (trim retained)
29	0.023368	keep	prodliab tail = none
30	0.023660	discard	CapeCod decay=0.9
31	0.024940	discard	prodliab Benktander apriori=0.55, n_iters=2 (trims retained)
32	0.023780	discard	ppauto n_periods=4 (trim retained)
33	0.023844	discard	prodliab drop_low=True added (two-sided trim)
34	0.024185	discard	prodliab per-LOB CapeCod decay=0.6
35	0.026299	discard	prodliab n_periods=2, no trim
36	0.025064	discard	wkcomp CapeCod
37	0.023528	discard	wkcomp tail = none
38	0.023864	discard	tail = none for ALL lines
39	0.022889	keep	othliab n_periods=5
40	0.024084	discard	medmal n_periods=5 (trim retained)
41	0.023647	discard	wkcomp n_periods=4 (trim retained)
42	0.023292	discard	prodliab volume averaging (per-LOB, global stays simple)
43	0.022491	keep	othliab n_periods=7
44	0.022546	discard	othliab n_periods=-1 (all periods)
45	0.023101	discard	comauto n_periods=5 (trim retained)
46	0.022951	discard	prodliab n_periods=5 with trim (retest in the new state)
47	0.022476	keep	prodliab n_periods=4 with trim
48	0.022604	discard	othliab volume averaging (per-LOB)
49	0.022872	discard	wkcomp drop_low added (two-sided trim)
50	0.022478	discard	ppauto volume averaging (per-LOB)
51	0.022723	discard	comauto n_periods=2 (trim off, ultra-recency)
52	0.026476	discard	medmal CapeCod (retest)
53	0.022427	keep	othliab per-LOB CapeCod decay=0.8
54	0.022771	discard	prodliab TailBondy [Bondy tail] (instead of no tail)
55	0.022786	discard	othliab CapeCod decay=0.6
56	0.022539	discard	comauto per-LOB CapeCod decay=0.8

APPENDIX D

THE AGENT'S INSTRUCTIONS AND THE PRE-SEEDING DISCLOSURE

Readers assessing what the agent 'discovered' versus what it was told should know that the agent's instruction file, `program.md`, pre-seeds several of the ideas the runs adopted. In particular, it states verbatim:

Key insight for experimentation: Different lines of business may benefit from different methods. Long-tail lines (medmal, othliab, prodliab) may need different treatment than short-tail lines (ppauto, comauto). The agent can and should build per-LOB strategies.

and its 'Ideas to explore (in rough priority order)' section enumerates the averaging methods, `n_periods` values, `drop_high` and `drop_low` trimming, tail-factor alternatives (including no tail), and method switching that constitute most of every run's search space. (The file also fixes the hard boundaries of the search: only the model-configuration script may be edited, the data-and-metric harness is read-only, and no new dependencies may be installed.) Finding 1's per-line method assignment is therefore a pre-seeded hypothesis that the runs validated and calibrated; the specific kept configurations (which lines take Cape Cod, the trimmed three-period simple average, other liability's seven-period window) are parameter-level outcomes the instructions did not contain. All 56 primary-run experiments fall within the instruction file's idea categories; the run's contribution is the empirical keep-or-discard record and its documentation, not an idea outside the human-written brief.