Advanced Programming 2025

PyMORT: Longevity Bond Pricing & Mortality Modeling

Final Project Report

Pierre-Antoine Le Quellec
HEC Lausanne – University of Lausanne
pierre-antoine.lequellec@unil.ch
Student ID: 22438071

November 11, 2025

Abstract

PyMORT develops a Python-based longevity bond pricing engine addressing the growing challenge of *longevity risk*—the financial risk that people live longer than expected, thereby increasing liabilities for pension funds and insurers. The project aims to model the stochastic evolution of mortality rates, forecast survival probabilities, and price securities that transfer this risk to financial markets.

Our methodology combines actuarial modeling and quantitative finance. Mortality dynamics are estimated using the **Lee-Carter** and **Cairns-Blake-Dowd (CBD)** models, fitted to real mortality data from the *Human Mortality Database*. Future mortality is projected through stochastic simulations, incorporating uncertainty and risk-neutral adjustments via a market price of longevity risk parameter. Using these forecasts, we value longevity-linked instruments—notably longevity bonds, survivor swaps, and mortality forwards—through expected discounted cash flows under the risk-neutral measure.

Key results include realistic mortality projections consistent with published studies and internally validated bond prices. The main contribution is a modular, open-source package that integrates actuarial modeling, risk-neutral valuation, and Monte Carlo simulation into a reproducible framework, offering researchers and practitioners a transparent tool for pricing and hedging longevity risk.

Keywords: Data Science, Python, Machine Learning, Longevity risk, Mortality modeling, Lee–Carter, Cairns–Blake–Dowd, Risk-neutral valuation, Longevity bonds, Survivor swaps, Quantitative finance.

Contents

1 Introduction

1.1 Background and Motivation

Over the past decades, the continuous increase in life expectancy has significantly reshaped the landscape of pension systems and insurance markets. This demographic shift introduces a new form of financial uncertainty known as **longevity risk**—the risk that individuals live longer than anticipated. While the extension of human life is a social achievement, it poses a financial challenge for institutions responsible for life-long payments such as pension funds, annuity providers, and governments. The longer beneficiaries live, the higher the cumulative liabilities become. To hedge against this risk, financial markets have developed a class of **longevity-linked securities**, whose cashflows depend on realized survival rates. These instruments create a bridge between actuarial science and financial engineering, enabling the transfer of longevity risk from pension funds to investors.

1.2 Problem Statement

The valuation of longevity-linked securities requires accurate **models of future mortality dynamics**. Mortality data are inherently complex—age-dependent, non-stationary, and affected by medical progress or sudden shocks such as pandemics. Designing models that capture both the **age structure** and the **temporal evolution** of mortality, while remaining stable and interpretable, remains an open problem in quantitative risk management. Furthermore, financial pricing demands translating real-world mortality forecasts into a **risk-neutral framework**, accounting for the market's perception of longevity risk. Building such a framework from raw demographic data is both statistically and computationally challenging.

1.3 Objectives and Goals

The aim of this project is to develop **PyMORT**, a Python-based longevity bond pricing engine that combines **actuarial modeling** and **financial mathematics**. More specifically, PyMORT will:

- Fit and calibrate stochastic mortality models, including *Lee–Carter* and *Cairns–Blake–Dowd*;
- Generate stochastic forecasts of mortality and survival probabilities with quantified uncertainty;
- Implement a **risk-neutral valuation** framework for longevity bonds and related derivatives;
- Provide a **command-line interface** with modular components for fitting, forecasting, pricing, and hedging;
- Ensure high code quality through type checking, testing coverage, and continuous integration.

Through these goals, PYMORT aims to serve as both an educational and practical tool for understanding how demographic dynamics translate into financial risk and asset pricing.

1.4 Report Organization

The remainder of this report is structured as follows:

- **Section 2 – Literature Review / Related Work** surveys the main mortality modeling approaches and existing longevity-linked instruments, including the Lee–Carter and Cairns–Blake–Dowd frameworks.
- **Section 3 – Methodology** describes the datasets used, data preprocessing steps, and the implementation of the stochastic mortality and pricing models within the PYMORT architecture.
  - **3.1 Data Description** outlines the Human Mortality Database and its key variables.
  - **3.2 Approach** details the statistical models and valuation methods.
  - **3.3 Implementation** presents the system design and major Python components.
- **Section 4 – Results** reports experimental outcomes, including model calibration, forecast accuracy, and bond pricing results, supported by tables and visualizations.
- **Section 5 – Discussion** interprets the findings, highlighting the main challenges, limitations, and lessons learned.
- **Section 6 – Conclusion and Future Work** summarizes the contributions and outlines potential extensions such as multi-population modeling, market calibration, and integration of stochastic interest rates.

2 Literature Review / Related Work

Discuss relevant prior work, existing solutions, or theoretical background. For data science projects, this might include:

- Previous approaches to similar problems
- Relevant algorithms or methodologies
- Datasets used in related studies

3 Methodology

3.1 Data Description

Describe your dataset(s):

- Source and collection method
- Size and characteristics
- Features/variables
- Data quality issues

3.2 Approach

Detail your technical approach:

- Algorithms used
- Data preprocessing steps
- Model architecture (if applicable)
- Evaluation metrics

3.3 Implementation

Discuss the implementation details:

- Programming languages and libraries
- System architecture
- Key code components

Example code snippet:

```python
def preprocess_data(df):
    """
    Preprocess the input dataframe.

    Args:
        df: Input pandas DataFrame

    Returns:
        Preprocessed DataFrame
    """
    # Remove missing values
    df = df.dropna()

    # Normalize numerical features
    scaler = StandardScaler()
    df[numerical_cols] = scaler.fit_transform(df[numerical_cols])

    return df
```

Listing 1: Example data preprocessing function

4 Results

Present your findings with appropriate visualizations and tables.

4.1 Experimental Setup

Describe your experimental environment:

- Hardware specifications
- Software versions
- Hyperparameters

4.2 Performance Evaluation

Table 1: Model Performance Metrics

Model	**Accuracy**	**Precision**	**Recall**
Baseline	0.75	0.72	0.78
Your Model	0.85	0.83	0.87

4.3 Visualizations

Include relevant plots and figures. For example:

Figure 1: Your results visualization

5 Discussion

Analyze and interpret your results:

- What worked well?
- What were the challenges?
- How do your results compare to expectations?
- Limitations of your approach

6 Conclusion and Future Work

6.1 Summary

Summarize your key findings and contributions.

6.2 Future Directions

Suggest potential improvements or extensions:

- Methodological improvements
- Additional experiments
- Real-world applications

References

1. Author, A. (2024). *Title of Article*. Journal Name, 10(2), 123-145.
2. Smith, B. & Jones, C. (2023). *Book Title*. Publisher.
3. Dataset Source. (2024). Dataset Name. Available at: `https://example.com`

A Additional Figures

Include supplementary figures or tables that support but aren't essential to the main narrative.

B Code Repository

GitHub Repository: `https://github.com/palqc/PYMORT`
Provide information about:

- Repository structure
- Installation instructions
- How to reproduce results