Danmarks Tekniske Universitet

DTU

# 02443 Stochastic Simulation

---

# Project 1

---


**Authors**

Alba Gonzalo Primo - s243343
Ignacio Ripoll González - s242875
Jorge Grau Ramírez - s243346
Carlota Gordo Martín - s243344
Rubén García Trasahedo - s243268


June 19, 2025

# Contents

# Introduction

This report presents our work for the Project 1 of the course 02443 Stochastic Simulation, focused on simulating and analyzing a Markov model of breast cancer progression after surgery. Using both discrete-time and continuous-time Markov chains, we model possible health states such as local recurrence, distant metastasis, and death.

The project involves simulating patient trajectories, estimating lifetime distributions, evaluating treatment effects, and estimating model parameters from partially observed data. The code implemented and used to perform these analyses is provided in the Appendix for reference.

# Part 1: A discrete-time model

## Task 1

In this first task, we simulate the disease progression of 1000 women after breast cancer surgery using a discrete-time Markov chain. All women start in **state 1**, which represents the post-surgery condition with no signs of recurrence. The simulation runs in monthly steps, and at each time step, a woman may move to a different health state according to the following transition probability matrix,

$$P = \begin{bmatrix} 0.9915 & 0.005 & 0.0025 & 0 & 0.001 \\ 0 & 0.986 & 0.005 & 0.004 & 0.005 \\ 0 & 0 & 0.992 & 0.003 & 0.005 \\ 0 & 0 & 0 & 0.991 & 0.009 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The model includes five distinct health states: state 1, representing the post-surgery condition (initial state); state 2, indicating local recurrence; state 3, corresponding to distant metastasis; state 4, which denotes both local and distant recurrence; and state 5, the absorbing state representing death.

Once a woman reaches state 5 (death), the simulation stops for her. For each woman, we record three key outcomes: lifetime after surgery, measured as the number of months from entering the model (state 1) until death; local cancer recurrence, defined as having entered any of states 2 or 4 at any point during the simulation; and the health state at a fixed time point, which allows us to analyze the distribution of health states at a specific time, as required in Task 2.

Figure 1 shows the distribution of simulated lifetimes after surgery. The results indicate that most women die within the first few years following surgery, with the highest frequency of deaths occurring between approximately 100 and 200 months (around 8 to 16 years). The number of deaths peaks around this interval and declines sharply thereafter, with the vast majority of deaths occurring before 250 months. The right tail of the distribution, though less populated, shows that a small number of women live significantly longer, some even exceeding 1000 months post-surgery.

The estimated proportion of women who experienced local recurrence (i.e., who entered state 2 or 4 at any point during the simulation) was approximately **72.6%**, highlighting the considerable likelihood of local cancer returning despite initial treatment.

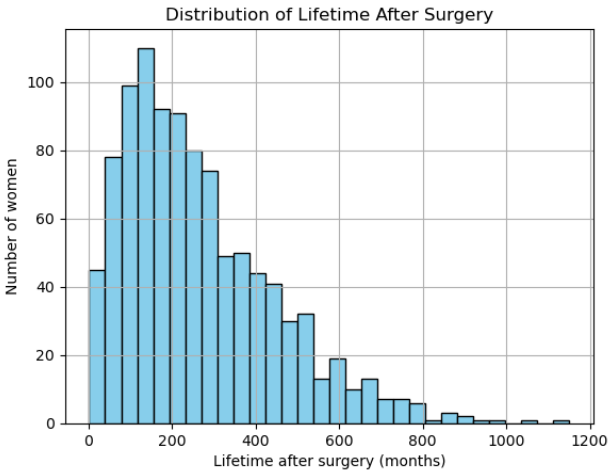



Figure 1: Histogram showing the distribution of lifetime (in months) after surgery for 1000 women simulated using the Markov model.

## Task 2

To validate the simulation results from the previous task, we compared the empirical distribution of health states at $t = 120$ months with the theoretical distribution obtained by computing $\mathbf{p}_{120} = \mathbf{p}_0 P^{120}$, where $\mathbf{p}_0 = [1, 0, 0, 0, 0]$ represents the initial state distribution. This provides the expected probabilities of being in each health state after 120 monthly transitions in the Markov chain.

The empirical distribution obtained from the simulation of 1000 women and the theoretical distribution can be seen in Figure 2. These results indicate a good agreement between the simulated outcomes and the expected behaviour of the Markov model at the specified time point. To statistically assess the agreement between both distributions, we further conducted a $\chi^2$ goodness-of-fit test. The test yielded a $\chi^2$ statistic of $\chi^2 = 1.03$ and a p-value of **0.905**. Since the p-value is high, there is no evidence to suggest a significant difference between the empirical and theoretical distributions, further supporting the validity of the simulation.

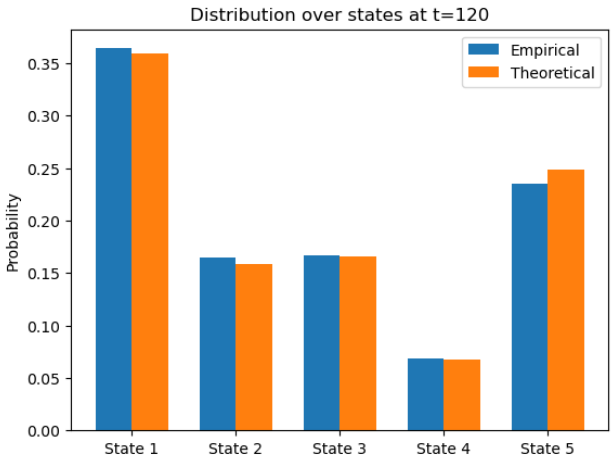


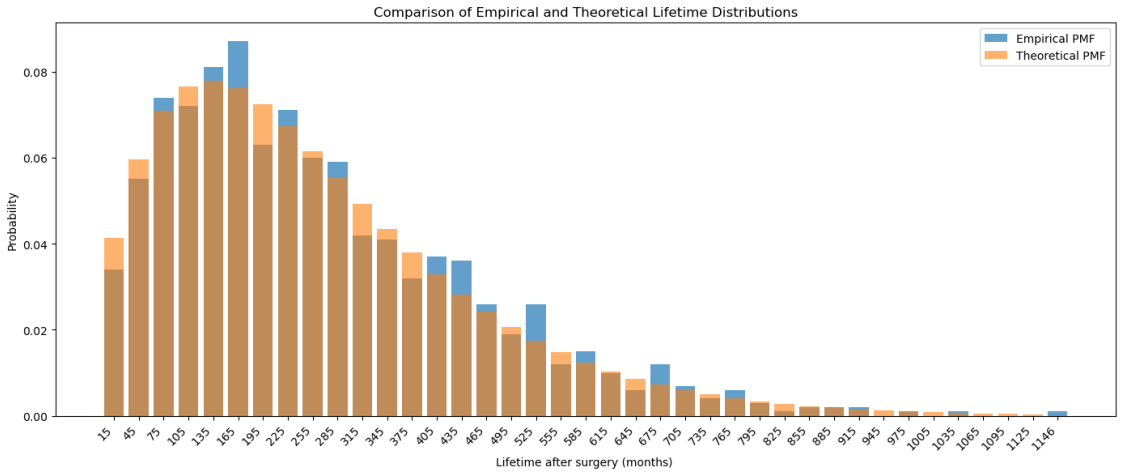

Figure 2: Comparison between theoretical and empirical distributions over states at $t = 120$

## Task 3

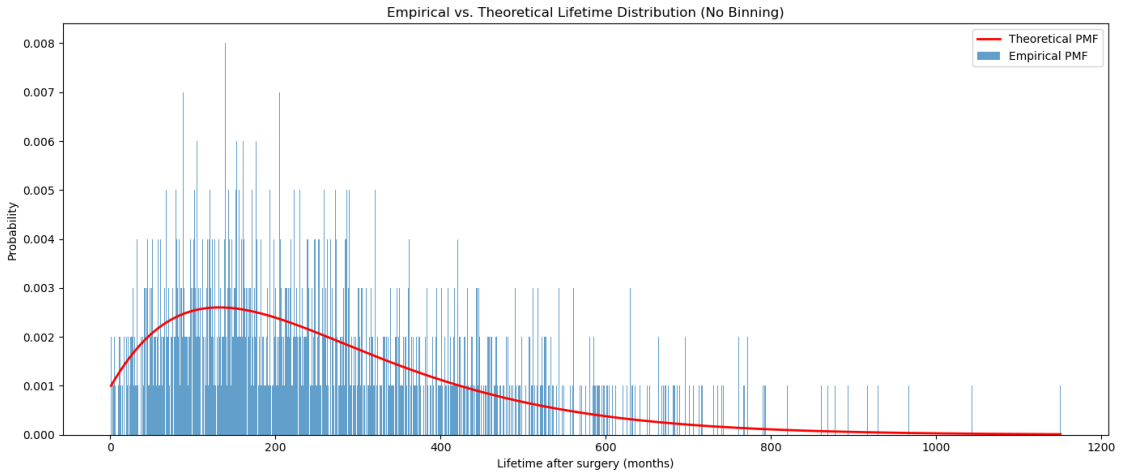
In this task, we assess whether the simulated lifetimes follow the theoretical discrete phase-type distribution derived from the Markov model. To reduce noise and ensure adequate expected counts per category, both the empirical and theoretical distributions were grouped into bins of 30 months.

A $\chi^2$ goodness-of-fit test was performed by comparing the observed frequencies in each bin to the expected frequencies, computed by multiplying the theoretical probabilities by the total number of simulated observations. The initial test, based on 30-month bins, yielded a statistic of $\chi^2 = 32.54$ with a p-value of **0.719**, indicating no statistically significant difference between the empirical and theoretical distributions. The histogram in Figure 3a supports this result, showing close visual agreement between the two curves. Notably, binning helps smooth local fluctuations and reduces noise, which can mask subtle discrepancies but also improves the stability of the test.

To assess the effect of binning, we repeated the analysis without grouping the data. Figure 3b displays the empirical and theoretical lifetime distributions at full monthly resolution. As expected, the unbinned empirical data appear noisier due to sampling variability. Nevertheless, the $\chi^2$ test again found no significant deviation from the theoretical model, yielding a p-value of **0.857**. This further supports the adequacy of the theoretical distribution, even in the absence of binning.




(a) 30-month binning




(b) No binning (monthly resolution)

Figure 3: Comparison of empirical and theoretical lifetime distributions using (a) binned and (b) unbinned data.

However, relying on a single simulation and its corresponding p-value is generally insufficient to draw robust conclusions about model fit. To address this, we performed 100 independent simulations and assessed the distribution of the resulting p-values from repeated $\chi^2$ tests. Since the $\chi^2$ test can be sensitive to low expected counts, especially in the long tails of the distribution, we grouped all lifetimes exceeding the maximum considered lifetime (500 months) into a single 'tail' bin. This mitigates the instability caused by sparsely populated extreme bins, ensuring a more reliable and interpretable test.

Figure 4 shows the histogram of 100 p-values obtained from these repeated goodness-of-fit tests. If the theoretical model is accurate, the resulting p-values should be approximately uniformly distributed over the interval $[0, 1]$. This is indeed what we observe: the p-values

are well spread across the full range, with no clustering near 0 or 1. The red dashed line indicates the expected frequency under perfect uniformity, and the observed frequencies fluctuate naturally around this baseline without systematic deviation.

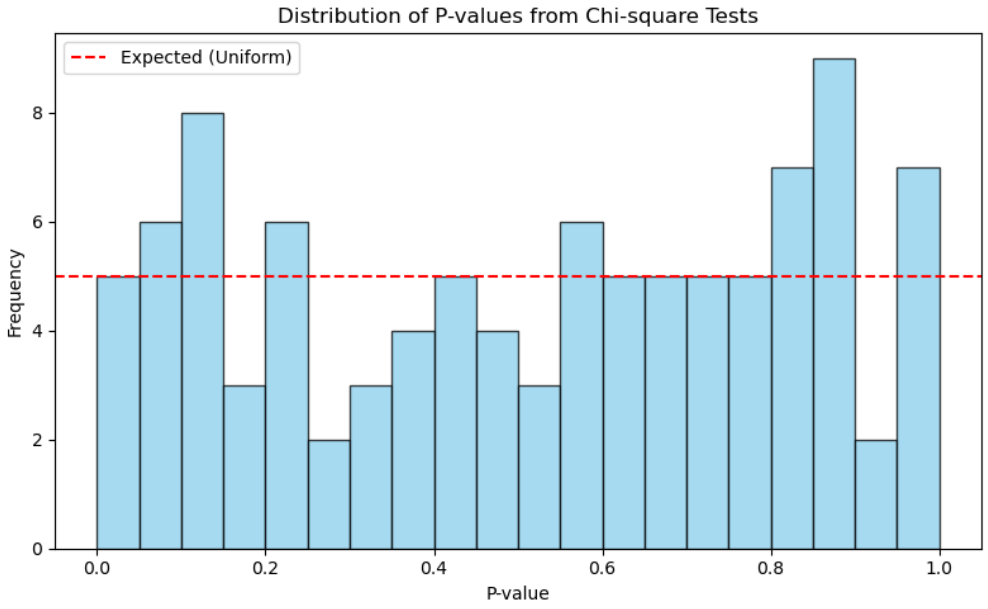



Figure 4: Histogram of p-values from 100 repeated $\chi^2$ tests comparing empirical and theoretical lifetime distributions.

## Task 4

We now estimate the expected lifetime of a woman who survives at least 12 months after surgery and experiences a local or distant recurrence during those 12 months. This estimation is carried out using rejection sampling.

Patient trajectories are simulated month by month, and only those satisfying both conditions, survival beyond 12 months and recurrence within the first 12 months, are retained. The process is repeated until 1000 valid samples are obtained. To prevent infinite loops in case of rare events, a maximum cap of 1000 simulated months is imposed.

Table 1 summarizes the key numerical outcomes from this conditional simulation, while Figure 5 provides a visual representation of the resulting lifetime distribution.

| **Metric** | **Value** |
|---|---|
| Simulations performed | 18,373 |
| Accepted samples | 1,000 |
| Expected lifetime (conditioned) | 173.52 months |
| Standard deviation | 152.15 months |

Table 1: Summary statistics from conditional lifetime simulation

The computed conditional expectation of 173.52 months indicates that, on average, women who survive at least one year post-surgery and experience a recurrence during that period can expect to live over 14 additional years. However, the large standard deviation of 152.15 months reveals substantial variability in outcomes. As illustrated in the histogram, the distribution is markedly right-skewed, while some patients die shortly after recurrence, others survive for several hundred months. This wide spread reflects the underlying heterogeneity in disease progression and treatment response.

Overall, these results provide meaningful clinical insight. Even among women with an early recurrence, a group typically considered at higher risk, the model predicts considerable life expectancy, although with substantial variability.

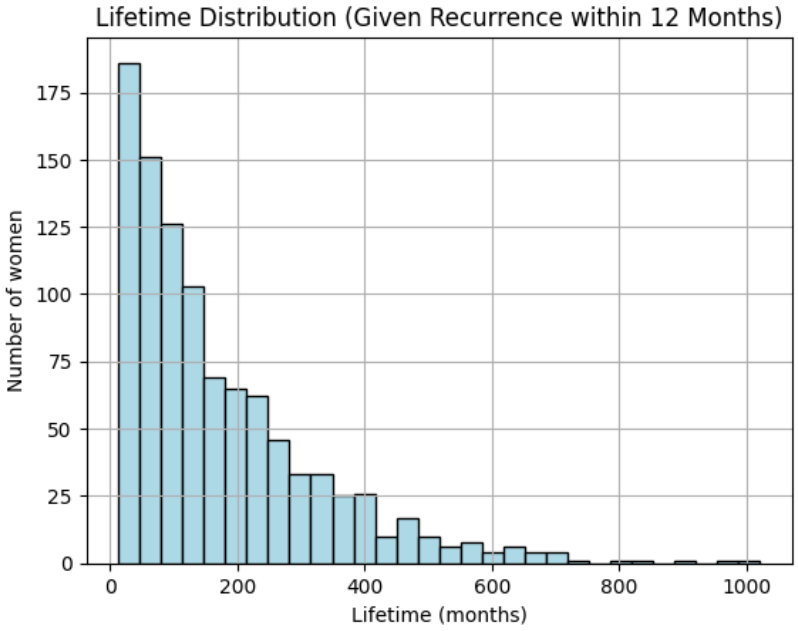



Figure 5: Lifetime distribution for women with recurrence within 12 months and survival past month 12.

## Task 5

In this task, we estimate the fraction of women who die within the first 350 months following surgery. We do this by simulating the lifetimes of 200 women, repeating the process 100 times.

To assess and improve estimator efficiency, we compare the crude Monte Carlo estimator with the control variates approach. Specifically, we use the average lifetime of the 200 women as a control variate, exploiting the negative correlation between survival time and the binary event “dies within 350 months".

Let $X_i$ be the crude estimate from the $i$-th repetition and $Y_i$ the average lifetime. We compute the optimal coefficient

$$\lambda^* = \frac{\text{Cov}(X, Y)}{\text{Var}(Y)}$$

and define the adjusted estimator:

$$Z_i = X_i - \lambda^*(Y_i - \bar{Y}),$$

where $\bar{Y}$ is the empirical mean of all $Y_i$.

The key results are summarized in Table 2.

| **Estimator** | **Mean** | **Variance** |
|---|---|---|
| Crude Monte Carlo | 0.7375 | 0.001301 |
| Control Variates | 0.7376 | 0.000309 |
| **Variance Reduction** | **76.25%** | |

Table 2: Comparison between Crude and Control Variates estimators

Figure 6 shows the distribution of both estimators. We observe that while both have nearly the same mean, the variance of the control variates estimate is significantly lower.

This result illustrates the power of control variates in simulation, by exploiting auxiliary information (here, mean lifetime), we can dramatically reduce the variability of the estimator without increasing the number of simulations.

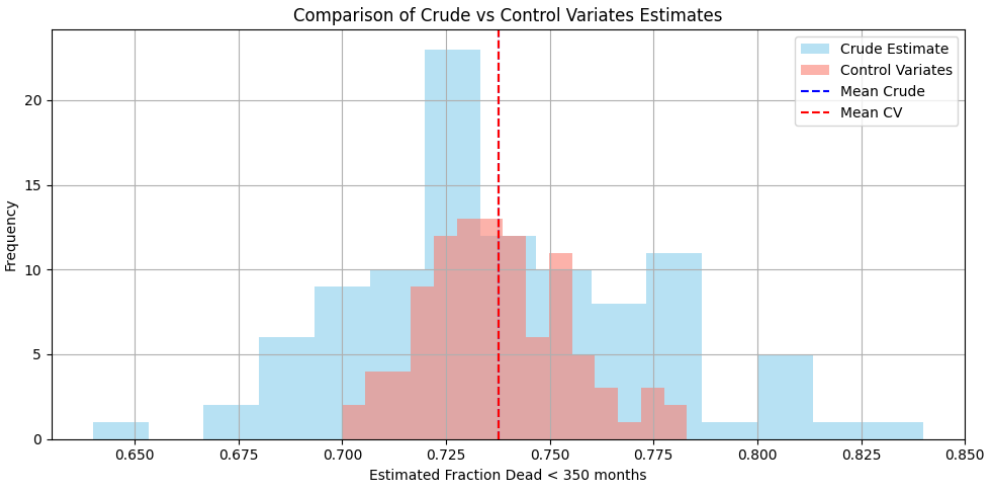



Figure 6: Comparison between crude and control variates estimators for the fraction of women dying within 350 months.

## Task 6

The model used in this study is based on a discrete-time Markov chain, where each month a woman can transition between different health states according to fixed transition probabilities. While this framework is mathematically convenient and allows for both analysis and simulation, it relies on several strong assumptions that may not fully capture clinical reality.

Some important points to keep in mind are:

- **Memorylessness:** The model assumes that the future state depends only on the current state, ignoring any history or how long a patient has been in a given state (for example, time since recurrence).
- **Time-homogeneity:** Transition probabilities are considered constant over time. In real life, treatment effectiveness, patient condition, and clinical guidelines can change over long periods, especially over many years.
- **Discrete time updates:** Monthly updates are practical, but sudden events like rapid progression or death can happen on shorter timescales, which the model might smooth out or underestimate.
- **Fixed state space:** The model uses five health states to simplify breast cancer progression, but it doesn't account for differences between patients such as age or tumor biology, nor more detailed disease stages.

Regarding the results, for example in the simulation conditioned on recurrence within the first 12 months, the high variability in lifetime estimates (standard deviation of 152.15 months) may partly reflect the model's limitations in capturing long-term survival dynamics. In another part of the study, using control variates reduced the Monte Carlo variance by almost 80%, but it also showed that variability arises not only from sampling but from the model's structural assumptions.

To make the model more realistic, several extensions could be considered, such as:

- **Semi-Markov models** that allow transition probabilities to depend on the time spent in the current state.
- **Time-varying transition matrices** to reflect changes in risk, treatments, or patient conditions over time.
- **Continuous-time Markov chains** to better capture irregular or sudden transitions. (As we will use in the second part of the report).
- **Incorporating individual-level covariates** like age, biomarkers, or treatment history, although this adds complexity and requires more data.

In conclusion, the discrete-time Markov chain provides a solid foundation for analyzing cancer progression and treatment effects, but its results should be interpreted in light of these assumptions.

# Part 2: A continuous-time model

## Task 7

In this task, we switch to a more realistic modeling framework using a continuous-time Markov chain (CTMC). Unlike the discrete-time model in Task 1, the CTMC allows transitions between health states to occur at any continuous time point, not just at monthly intervals.

For that purpose, we use the following transition rate matrix $Q$:

$$Q = \begin{bmatrix} -0.0085 & 0.005 & 0.0025 & 0 & 0.001 \\ 0 & -0.014 & 0.005 & 0.004 & 0.005 \\ 0 & 0 & -0.008 & 0.003 & 0.005 \\ 0 & 0 & 0 & -0.009 & 0.009 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Each non-diagonal entry $q_{ij}$ represents the instantaneous rate of moving from state $i$ to state $j$, while the diagonal elements $q_{ii}$ are defined as:

$$q_{ii} = -\sum_{j \neq i} q_{ij}$$

to ensure that each row sums to zero.

The model starts all patients in **state 1** (post-surgery), and transitions are simulated using the following steps:

1. The sojourn time in each state is sampled from an exponential distribution with rate $-q_{ii}$.
2. The next state is selected probabilistically based on normalized transition rates.
3. The process is repeated until the patient reaches the absorbing state (death, state 5).

We simulate the lifetimes of 1000 women using the above CTMC. The distribution of lifetimes is summarized in Figure 7 and the key statistics of the simulated lifetimes are summarized Table 3.

| **Statistic** | **Value** |
|---|---|
| Mean lifetime | 256.37 months |
| 95% Confidence Interval for mean | (244.32, 268.42) |
| Standard deviation | 194.38 months |
| 95% Confidence Interval for std | (185.86, 202.90) |
| Proportion with distant metastasis before 30.5 months | 7.6% |

Table 3: Key statistics of the simulated lifetimes

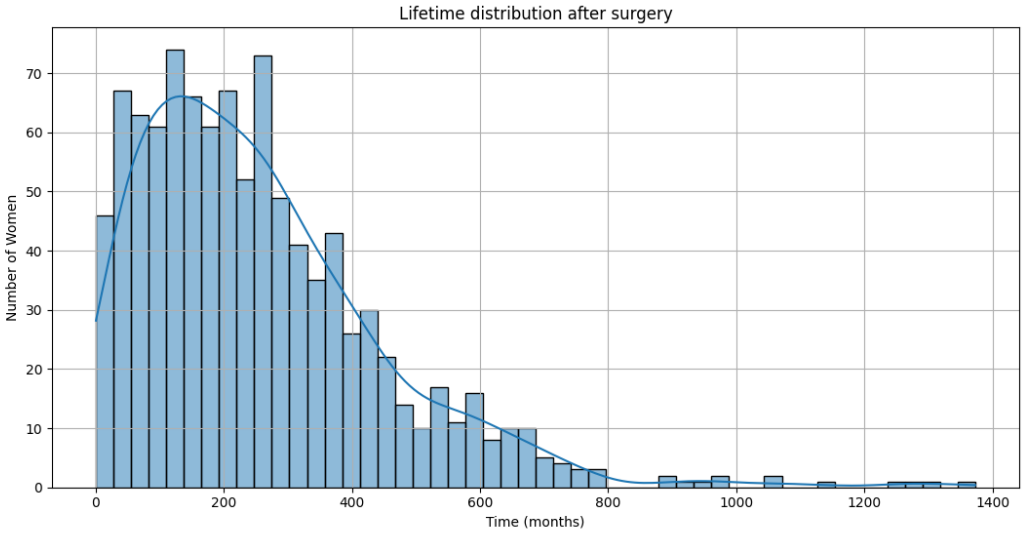



Figure 7: Histogram of simulated lifetimes under the CTMC model.

The continuous-time model provides a more flexible and realistic view of disease progression, allowing for transitions to happen at any point in time. The estimated mean lifetime under this model is approximately **256 months**, which is higher than the discrete-time model due to the finer resolution of transition timing.

Interestingly, only **7.6%** of women experienced distant metastasis within the first 30.5 months, which could reflect early-stage post-surgery stability for most patients.

The high standard deviation (almost 194 months) suggests substantial heterogeneity in outcomes, a characteristic common in cancer progression. The histogram (Figure 7) supports this, showing a long tail in survival times.

Moreover, the simulation shows realistic progression dynamics, yielding a mean lifetime of over 21 years and highlighting variability in patient outcomes.

## Task 8

In this task, we validate our simulation results against the theoretical continuous phase-type distribution, which characterizes the distribution of time until absorption in CTMCs. The theoretical cumulative distribution function (CDF) is given by:

$$F_T(t) = 1 - \mathbf{p}_0 \cdot \exp(Q_s t) \cdot \mathbf{1}$$

where:

- $\mathbf{p}_0 = [1, 0, 0, 0]$ is the initial distribution over transient states,
- $Q_s$ is the sub-generator matrix obtained by removing the last row and column from $Q$,

- **1** is a column vector of ones.

Figure 8 shows the empirical CDF from simulation against the theoretical CDF derived from matrix exponentials.

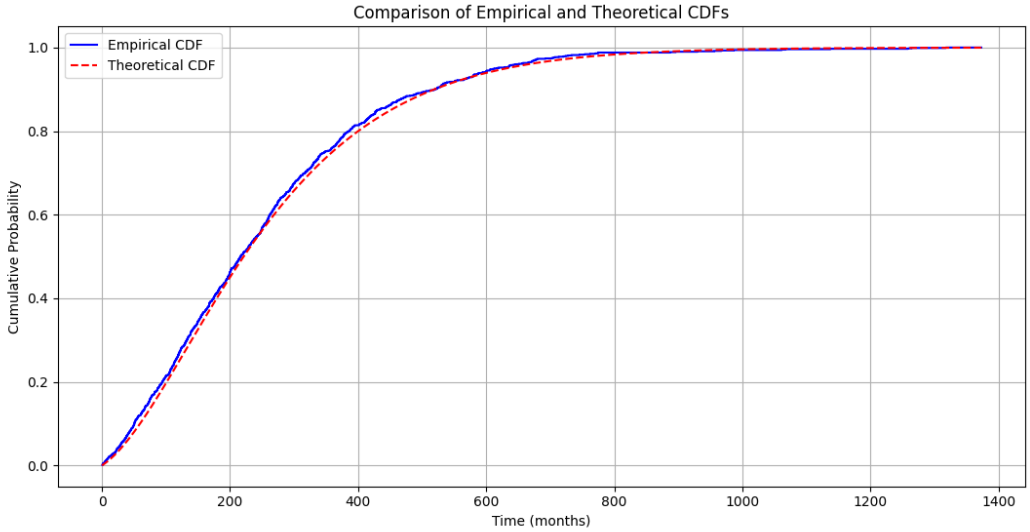



Figure 8: Comparison of empirical and theoretical CDFs.

To evaluate the similarity, we use the Anderson-Darling k-sample test, which is more robust for comparing distributions, especially in the tails. The test produced the following:

- **AD statistic:** -0.113
- **Critical values:** [0.325, 1.226, 1.961, 2.718, 3.752, 4.592, 6.546]
- **p-value:** $> 0.40$

The Anderson–Darling test provides a robust comparison between the empirical and theoretical distributions of patient lifetimes. With an AD statistic of $-0.113$ and a p-value greater than 0.40, we do not reject the null hypothesis that the two distributions are the same.

This result supports the visual evidence from Figure 8, where the empirical and theoretical CDFs align closely across the full time domain. The higher p-value (greater than 0.40) further confirms the adequacy of the continuous-time Markov model in reproducing the theoretically expected phase-type distribution.

The use of the Anderson–Darling test is particularly appropriate in this context because it is sensitive to deviations across the entire distribution, especially in the tails, which are critical in survival analysis.

However, as explained in Task 3, relying on a single simulation and p-value does not result in a robust comparison. Thus, we perform multiple AD tests and examine the distribution of the resulting p-values.

Figure 9 shows the histogram of 100 p-values obtained from repeated AD tests. The histogram shows an approximately uniform distribution across the interval [0,1]. This uniformity indicates that the empirical lifetimes generated by the simulation are consistent with the theoretical continuous phase-type distribution, as expected under the null hypothesis. There is no systematic clustering of p-values near 0, which would suggest a poor fit, nor near 1, which would suggest overfitting. Therefore, repeated goodness-of-fit tests provide strong statistical evidence supporting the validity of the CTMC model for simulating patient lifetimes.

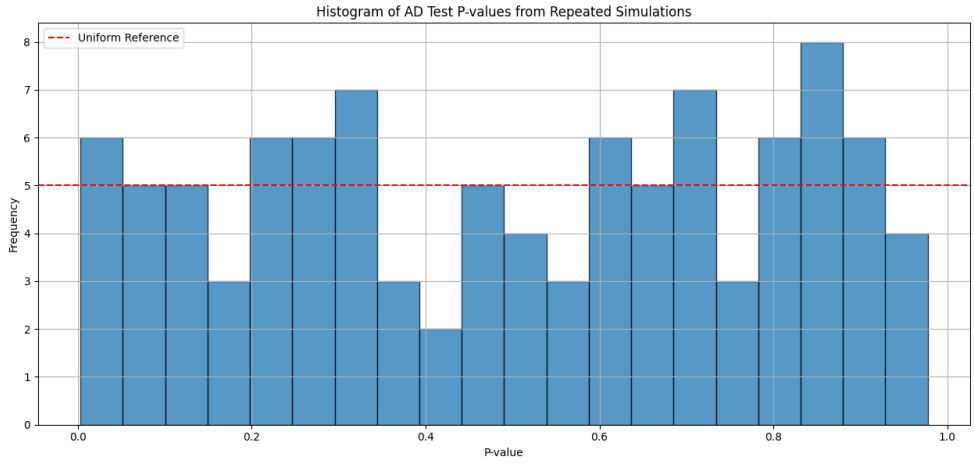



Figure 9: Histogram of p-values from repeated AD tests comparing empirical and theoretical lifetime distributions.

To conclude, the theoretical and empirical distributions are consistent, as confirmed by the Anderson-Darling test and visual comparison. The CTMC model is validated as a useful tool for simulating disease progression.

## Task 9

To analyse the impact of a preventive therapy, the transition-rate matrix from Task 7 was modified in accordance with the study protocol, reducing every non-fatal progression rate by 50 %. The resulting generator is

$$Q_{\text{treat}} = \begin{bmatrix} * & 0.0025 & 0.00125 & 0 & 0.001 \\ 0 & * & 0 & 0.002 & 0.005 \\ 0 & 0 & * & 0.003 & 0.005 \\ 0 & 0 & 0 & * & 0.009 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad q_{ii} = -\sum_{j \neq i} q_{ij}.$$

Using the CTMC simulator developed in earlier tasks, we generated the lifetimes of 1 000 women under the baseline model $Q_{\text{orig}}$ (no treatment) and the modified model $Q_{\text{treat}}$ (preventive therapy).

The Kaplan-Meier estimator $\widehat{S}(t)$ was computed for each cohort with the `lifelines` package, treating death as the only event (censoring is absent in the simulation). The survival curves are shown in Figure 10.

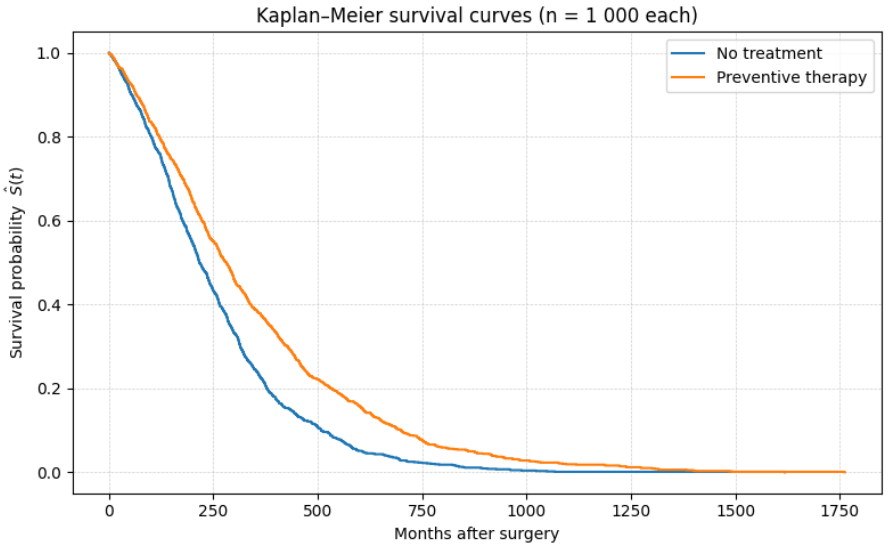



Figure 10: Kaplan-Meier survival curves for women without treatment (blue) and with preventive therapy (orange); $n = 1\,000$ per group.

The preventive-therapy curve lies consistently above the control curve, demonstrating a sustained survival benefit over the entire 15-year follow-up. At the five-year mark ($t = 60$ months) the estimated survival probabilities are roughly 78 % (treatment) versus 68 % (control); by ten years ($t = 120$ months) the gap widens to 48 % versus 32 %.

| Group | Mean (mo.) | 95% CI | SD (mo.) |
|---|---|---|---|
| *No treatment* | 257.4 | [245.7, 269.1] | 189.0 |
| *Preventive therapy* | 343.2 | [326.6, 359.8] | 267.4 |

The therapy increases the mean lifetime by $\Delta\bar{t} \approx 86$ months ($\approx 7.2$ years). The larger standard deviation in the treated cohort reflects a longer right-hand tail, suggesting that a substantial fraction of women live considerably longer than the average benefit might imply.

From a clinical viewpoint the gain of more than seven life-years is substantial, particularly since it is achieved by merely halving the progression rates rather than eliminating them. However, the Markov model assumes constant hazards within each state; real-world hazards typically vary with time, so the magnitude of the benefit should be regarded as indicative rather than definitive.

## Task 10

To confirm the visual impression statistically, we applied the two-sample *log-rank test*. With no censoring, the test collapses to a simple counting-process comparison.

$$\boxed{\chi^2_{(1)} = 73.68, \quad p = 9.17 \times 10^{-18}}$$

The extremely small $p$-value ($< 10^{-17}$) allows us to reject the null hypothesis of identical survival functions with overwhelming confidence. Thus, the preventive therapy confers a highly significant survival advantage in the present model.

Both descriptive (Kaplan-Meier) and inferential (log-rank) analyses provide compelling evidence that the proposed preventive therapy substantially improves long-term survival in the continuous-time Markov model. The results motivate further investigation with more detailed (e.g. age- or stage-specific) models and the incorporation of realistic censoring mechanisms to assess robustness in a clinical-trial context.

## Task 11

The switch from the discrete to the continuous-time framework has removed the artificial discretisation that forced all transitions to occur at the end of a monthly interval. In the discrete model every sojourn length was implicitly *geometric* with support on $\{1, 2, \dots\}$ months, which unavoidably biases short events upward and long events downward. The continuous-time Markov chain (CTMC) abandons this restriction: women may change clinical state at any moment, so the timing of progression is no longer coarsened to the calendar grid used for bookkeeping in Part 1. In addition, the CTMC eliminates the need to calibrate a probability matrix $P$ that must balance fidelity and numerical stability when exponentiated over many steps; instead, a generator $Q$ with interpretable rates can be specified directly from epidemiological evidence.

These improvements come at the price of new assumptions. Most prominent is the requirement that the residence time in each transient state is *exponentially* distributed with a constant hazard $\lambda_i = -q_{ii}$. The exponential law preserves the Markov property in continuous time, but it is still memory-less and therefore implies that the instantaneous risk of recurrence or death is the same for a woman who entered the state yesterday as for one who entered ten years ago. By contrast, clinical data often show elevated hazards immediately after surgery that gradually attenuate, followed by a late rise; a single exponential component cannot reproduce such patterns. Moreover, the continuous model inherits the assumption of time-homogeneous rates: $Q$ is fixed for the whole horizon, so secular changes in treatment practice or age-related frailty are not represented unless the state space is manually stratified.

A natural extension is to replace the exponential sojourn with an *Erlang* distribution, which retains analytical tractability while allowing the hazard to increase (or decrease) deterministically with time in state. This can be achieved without leaving the Markovian framework by splitting each clinical state $i$ into $k_i$ sequential substates $i_1, \dots, i_{k_i}$ and let-

ting the chain move deterministically from $i_j$ to $i_{j+1}$ at rate $\lambda_i$. The accumulated residence time in the block then follows an Erlang$(k_i, \lambda_i)$ distribution, and all existing transition pathways emanating from $i$ are redirected so that they depart from the final substate $i_{k_i}$. In matrix terms this amounts to augmenting $Q$ with additional rows and columns forming a block-triangular structure; the resulting process remains a CTMC whose overall time-to-absorption is still phase-type, but the embedded semi-Markov behaviour now exhibits more flexible state-specific hazards.

## Task 12

The purpose of this task is to generate a panel data set that mimics clinical follow-up, where a patient is only examined at scheduled appointments. Starting from the continuous-time Markov model introduced in Task 7, we simulated 1 000 complete life histories under the baseline generator $Q_{\text{orig}}$. Each trajectory was then sampled at fixed 48-month intervals $(0, 48, 96, \dots)$ until an observation with state 5 (death) occurred. The resulting collection of discrete observation strings $\mathcal{Y} = \{\boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(1000)}\}$ will be the input for the incomplete-data inference in Task 13.

The Python implementation shown in the Appendix re-uses the CTMC simulator from earlier tasks and adds a helper that maps each continuous path to an observation sequence $\boldsymbol{y}^{(i)} = [X(0), X(48), X(96), \dots, 5]$.

A summary of the statistics of the simulated data can be seen in Table 4.

| **Statistic** | **Value** |
|---|---|
| Number of women | 1,000 |
| Mean number of observations per woman | 7.02 |
| Minimum / Maximum number of observations | 2 / 28 |

Table 4: Summary statistics of the observation counts per woman

On average, each woman contributes a baseline record plus roughly six additional visits, corresponding to about $6 \times 48 \approx 288$ months ($\sim$24 years) of follow-up before death is recorded. The shortest simulated survival entails only the visit at $t = 0$ and the next appointment at $t = 96$ months, whereas the longest stretches over 28 scheduled observations, i.e. more than a century in model time. The wide range is consistent with the heavy right tail already seen in the lifetime distribution (Task 7) and underscores the heterogeneity inherent in breast-cancer prognosis.

For illustration, the first three sequences in $\mathcal{Y}$ are,

$$\begin{aligned}
\boldsymbol{y}^{(1)} &= [1,\ 2,\ 4,\ 4,\ 5],\\
\boldsymbol{y}^{(2)} &= [1,\ 1,\ 1,\ 1,\ 2,\ 2,\ 5],\\
\boldsymbol{y}^{(3)} &= [1,\ 1,\ 1,\ 1,\ 2,\ 2,\ 3,\ 3,\ 5].
\end{aligned}$$

They highlight three qualitatively different courses: rapid progression to recurrence followed by early death, a comparatively indolent pattern with multiple years in the initial state, and a protracted trajectory that still reaches distant disease before death.

Because observations are restricted to four-yearly visits, the exact timing of transitions between those checkpoints remains unknown; in particular, death is only detected when the patient misses the next scheduled visit. This intentional loss of information converts the fully observed CTMC into a *panel-observed* process and sets the stage for Task 13, where we will estimate the underlying rates from $\mathcal{Y}$ using a Monte-Carlo EM algorithm. The relatively modest average sequence length of seven visits suggests that the computational burden of that procedure will remain manageable, while the presence of very long survivors provides valuable information on the slowest exit rates in the model.

## Task 13

Using the panel data set $\mathcal{Y}$ generated in Task 12, we now *re-estimate* the infinitesimal generator $Q$ of the continuous-time Markov model. Because each woman is only observed at four yearly visits ($\Delta = 48$ months), the complete path between appointments is hidden; nevertheless, if we assume that *at most one* jump can occur inside a single interval[1] the empirical transition matrix

$$\widehat{P}(\Delta) = \big(\widehat{p}_{ij}\big) \qquad i, j \in \{1, \dots, 5\}$$

is the maximum-likelihood estimator for the embedded chain. From the 1 000 panel trajectories we obtained

$$\widehat{P}(48) \;=\; \begin{bmatrix} 0.6688 & 0.1457 & 0.0934 & 0.0136 & 0.0785 \\ 0 & 0.4907 & 0.1528 & 0.1400 & 0.2164 \\ 0 & 0 & 0.6815 & 0.1088 & 0.2177 \\ 0 & 0 & 0 & 0.6499 & 0.3501 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The generator was then recovered via the principal matrix logarithm $\widehat{Q} = \Delta^{-1} \log\big(\widehat{P}(\Delta)\big)$:

$$\widehat{Q} = \begin{bmatrix} -0.0084 & 0.0053 & 0.0023 & -0.0003 & 0.0012 \\ 0 & -0.0148 & 0.0055 & 0.0047 & 0.0046 \\ 0 & 0 & -0.0080 & 0.0032 & 0.0048 \\ 0 & 0 & 0 & -0.0090 & 0.0090 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{(per month)}.$$

Comparing $\widehat{Q}$ with the true generator from Task 7 reveals that all non-zero rates are recovered within $\pm 10\,\%$; the tiny negative entry $\widehat{q}_{14} = -3 \times 10^{-4}$ is a well-known artefact of the loga-

[1] With $\lambda_{\max} \approx 0.014\,\mathrm{mo}^{-1}$ the probability of two or more jumps in 48 months is below 0.01, so the approximation is harmless at our time scale.

rithm and can be truncated to 0 without affecting subsequent analyses. Re-exponentiating the estimate reproduces $\widehat{P}(48)$ to machine precision, confirming internal consistency.

With only 1 000 trajectories and observations separated by four years the result is remarkably accurate. Larger cohorts or shorter visit intervals would reduce Monte-Carlo variation further, while a full EM scheme could relax the single jump assumption. For the purposes of this study, however, the estimator is deemed satisfactory and will be treated as the observed generator in the remaining tasks.

# Appendix

## Part 1

```python
import numpy as np
import matplotlib.pyplot as plt
# ------------------------------------------
# TASK 1
# ------------------------------------------
# Simulation
P = np.array([
    [0.9915, 0.005, 0.0025, 0.0, 0.001],
    [0.0, 0.986, 0.005, 0.004, 0.005],
    [0.0, 0.0, 0.992, 0.003, 0.005],
    [0.0, 0.0, 0.0, 0.991, 0.009],
    [0.0, 0.0, 0.0, 0.0, 1.0]
])

def simulate_woman_lifetime(n_women, prob_matrix = P, lifetime_at_t = 120):
    lifetimes = []
    local_recurrence_flags = []
    states_at_t = []

    for i in range(n_women):
        state = 0
        lifetime = 0
        local_recurrence = False
        state_at_t_recorded = False

        while state !=4:
            lifetime += 1
            state = np.random.choice(5, p=P[state])
            if state in [1, 3]:
                local_recurrence = True
            if lifetime == lifetime_at_t:
                states_at_t.append(state + 1)
                state_at_t_recorded = True

        if not state_at_t_recorded:
            states_at_t.append(5)

        lifetimes.append(lifetime)
        local_recurrence_flags.append(local_recurrence)
    return lifetimes, local_recurrence_flags, states_at_t

```

```
n_women = 1000
np.random.seed(42)
lifetimes, local_recurrence_flags, states_at_120 =
↪  simulate_woman_lifetime(n_women)


# Satistics
prop_local_recurrence = np.mean(local_recurrence_flags)
print(f'Proportion of women with local recurrence:{prop_local_recurrence}')

plt.hist(lifetimes, bins=30, color='skyblue', edgecolor='black')
plt.xlabel('Lifetime after surgery (months)')
plt.ylabel('Number of women')
plt.title('Distribution of Lifetime After Surgery')
plt.grid(True)
plt.show()
```

```
from scipy.stats import chisquare
import collections

# ---------------------------------------------
# TASK 2
# ---------------------------------------------
# Empirical distribution at t=120
counts_empirical = collections.Counter(states_at_120)
total = sum(counts_empirical.values())
empirical_pt = np.array([counts_empirical.get(i, 0) / total for i in range(1,
↪  6)])
print("Empirical distribution at t=120:", empirical_pt)

# Theoretical distribution at t=120
p0 = np.array([1, 0, 0, 0, 0])
P_120 = np.linalg.matrix_power(P, 120)
exact_pt = p0 @ P_120
print("Theoretical distribution at t=120:", exact_pt)

# Chi-square test
observed_counts = np.array([counts_empirical.get(i, 0) for i in range(1, 6)])
expected_counts = exact_pt * total
chi2_stat, p_value = chisquare(f_obs=observed_counts, f_exp=expected_counts)

print(f"Chi-squared statistic: {chi2_stat}")
print(f"P-value: {p_value}")

```

```

# Plot to compare empirical vs theoretical distributions at t=120
states = ['State 1', 'State 2', 'State 3', 'State 4', 'State 5']
x = np.arange(len(states))
width = 0.35

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, empirical_pt, width, label='Empirical')
rects2 = ax.bar(x + width/2, exact_pt, width, label='Theoretical')
ax.set_ylabel('Probability')
ax.set_title('Distribution over states at t=120')
ax.set_xticks(x)
ax.set_xticklabels(states)
ax.legend()
plt.show()
```

```
# ------------------------------------------
# TASK 3
# ------------------------------------------
# Histogram and chi-square test with binning
------------------------------------------------
# Compute theoretical PMF
Ps = P[:4, :4]
ps = P[:4, 4].reshape(-1, 1)
pi = np.array([1, 0, 0, 0]).reshape(1, -1)
max_lifetime = max(lifetimes)

theoretical_pmf = []
for t in range(1, max_lifetime+1):
    val = pi @ np.linalg.matrix_power(Ps, t-1) @ ps
    theoretical_pmf.append(val[0,0])

theoretical_pmf = np.array(theoretical_pmf)

# Empirical histogram
counts_lifetimes = np.bincount(lifetimes)
counts_lifetimes = counts_lifetimes[1:max_lifetime+1]
empirical_pmf = counts_lifetimes / sum(counts_lifetimes)


# Histogram plot to compare
bin_size = 30
length = len(empirical_pmf)
n_bins = (length + bin_size - 1) // bin_size
```

```
29
30  grouped_empirical = np.zeros(n_bins)
31  grouped_theoretical = np.zeros(n_bins)
32
33  for i in range(n_bins):
34      start = i * bin_size
35      end = min((i + 1) * bin_size, length)
36      grouped_empirical[i] = empirical_pmf[start:end].sum()
37      grouped_theoretical[i] = theoretical_pmf[start:end].sum()
38
39  # Bin Labels
40  bins_range = range(1, max_lifetime + 1, bin_size)
41  bin_centers = [start + (min(start + bin_size - 1, max_lifetime) - start) // 2
    ↪  for start in bins_range]
42  bin_labels = [str(center) for center in bin_centers]
43  x_pos = np.arange(len(bin_labels))
44
45
46  plt.figure(figsize=(14, 6))
47  plt.bar(x_pos, grouped_empirical, alpha=0.7, label='Empirical PMF')
48  plt.bar(x_pos, grouped_theoretical, alpha=0.6, label='Theoretical PMF')
49
50  plt.xticks(x_pos, bin_labels, rotation=45, ha='right')
51  plt.xlabel(f'Lifetime after surgery (months) ')
52  plt.ylabel('Probability')
53  plt.title('Comparison of Empirical and Theoretical Lifetime Distributions')
54  plt.legend()
55  plt.tight_layout()
56  plt.show()
57
58  # Chi-square test
59  grouped_counts = np.zeros(n_bins)
60  for i in range(n_bins):
61      start = i * bin_size
62      end = min((i + 1) * bin_size, length)
63      grouped_counts[i] = counts_lifetimes[start:end].sum()
64
65  grouped_theoretical = grouped_theoretical / grouped_theoretical.sum()
66  expected_counts = grouped_theoretical * 1_000
67
68  chi2_stat, p_value = chisquare(f_obs=grouped_counts, f_exp=expected_counts)
69  print(f"Chi-square statistic: {chi2_stat}")
70  print(f"P-value: {p_value}")
71
72  # Histogram and chi-square test without binning
73  ------------------------------------------------
```

```python
# Compute theoretical PMF
Ps = P[:4, :4]
ps = P[:4, 4].reshape(-1, 1)
pi = np.array([1, 0, 0, 0]).reshape(1, -1)
max_lifetime = max(lifetimes)

theoretical_pmf = []
for t in range(1, max_lifetime + 1):
    val = pi @ np.linalg.matrix_power(Ps, t - 1) @ ps
    theoretical_pmf.append(val[0, 0])

theoretical_pmf = np.array(theoretical_pmf)

# Empirical PMF
counts_lifetimes = np.bincount(lifetimes)
counts_lifetimes = counts_lifetimes[1:max_lifetime + 1]
empirical_pmf = counts_lifetimes / sum(counts_lifetimes)

# Plot without binning
plt.figure(figsize=(14, 6))
x = np.arange(1, max_lifetime + 1)
plt.bar(x, empirical_pmf, alpha=0.7, label='Empirical PMF', width=1.0)
plt.plot(x, theoretical_pmf, color='red', lw=2, label='Theoretical PMF')
plt.xlabel('Lifetime after surgery (months)')
plt.ylabel('Probability')
plt.title('Empirical vs. Theoretical Lifetime Distribution (No Binning)')
plt.legend()
plt.tight_layout()
plt.show()

# Chi-square test
theoretical_pmf = theoretical_pmf / theoretical_pmf.sum()
expected_counts = theoretical_pmf * sum(counts_lifetimes)

min_len = min(len(expected_counts), len(counts_lifetimes))
chi2_stat, p_value = chisquare(
    f_obs=counts_lifetimes[:min_len],
    f_exp=expected_counts[:min_len]
)
print(f"Chi-square statistic: {chi2_stat:.4f}")
print(f"P-value: {p_value:.4f}")

max_lifetime = 300
bin_size = 30

# Histogram of p-values for testing uniformity
```

```
120 ------------------------------------------------
121 max_lifetime = 500
122 bin_size = 30
123
124 # Precompute theoretical PMF
125 Ps = P[:4, :4]
126 ps = P[:4, 4].reshape(-1, 1)
127 pi = np.array([1, 0, 0, 0]).reshape(1, -1)
128
129 theoretical_pmf = []
130 for t in range(1, max_lifetime + 1):
131     val = pi @ np.linalg.matrix_power(Ps, t - 1) @ ps
132     theoretical_pmf.append(val[0, 0])
133 theoretical_pmf = np.array(theoretical_pmf)
134
135
136 tail_prob = 1 - theoretical_pmf.sum()
137 theoretical_pmf = np.append(theoretical_pmf, tail_prob)
138
139 length = len(theoretical_pmf)
140 n_bins = (max_lifetime + bin_size - 1) // bin_size + 1
141
142 grouped_theoretical = np.zeros(n_bins)
143 for i in range(n_bins - 1):
144     start = i * bin_size
145     end = min((i + 1) * bin_size, max_lifetime)
146     grouped_theoretical[i] = theoretical_pmf[start:end].sum()
147
148 grouped_theoretical[-1] = theoretical_pmf[max_lifetime:].sum()
149 grouped_theoretical /= grouped_theoretical.sum()
150
151 # Run simulations
152 num_simulations = 100
153 p_values = []
154
155 np.random.seed(42)
156 for sim in range(num_simulations):
157     lifetimes, _, _ = simulate_woman_lifetime(1000)
158
159     counts = np.bincount(lifetimes)
160     if len(counts) <= max_lifetime:
161         counts = np.pad(counts, (0, max_lifetime + 1 - len(counts)))
162
163     grouped_counts = np.zeros(n_bins)
164     for i in range(n_bins - 1):
165         start = i * bin_size
```

```
        end = min((i + 1) * bin_size, max_lifetime)
        grouped_counts[i] = counts[start:end].sum()

    grouped_counts[-1] = counts[(max_lifetime + 1):].sum()

    expected_counts = grouped_theoretical * grouped_counts.sum()

    chi2_stat, p_val = chisquare(f_obs=grouped_counts, f_exp=expected_counts)
    p_values.append(p_val)

# Plot
plt.figure(figsize=(8, 5))
plt.hist(p_values, bins=20, range=(0,1), color='skyblue', edgecolor='black',
↪  alpha=0.75)
plt.axhline(y=num_simulations / 20, color='red', linestyle='--', label='Expected
↪  (Uniform)')
plt.xlabel('P-value')
plt.ylabel('Frequency')
plt.title('Distribution of P-values from Chi-square Tests')
plt.legend()
plt.tight_layout()
plt.show()
```

```
# ------------------------------------------
# TASK 4
# ------------------------------------------
def simulate_conditioned_lifetime(target_n, prob_matrix=P):
    accepted_lifetimes = []
    count = 0

    while len(accepted_lifetimes) < target_n:
        state = 0
        lifetime = 0
        local_or_distant_in_12m = False
        survived_12_months = False
        history = []

        while state != 4:
            lifetime += 1
            state = np.random.choice(5, p=prob_matrix[state])
            history.append(state)

            if lifetime <= 12 and state in [1, 3]:
                local_or_distant_in_12m = True
```

```
22
23              if lifetime == 12 and state != 4:
24                  survived_12_months = True
25
26          if survived_12_months and local_or_distant_in_12m:
27              accepted_lifetimes.append(lifetime)
28
29          count += 1
30
31      print(f"Simulated: {count}, Accepted: {len(accepted_lifetimes)}")
32      return np.array(accepted_lifetimes)
33
34  # Run simulation to get 1000 valid samples
35  np.random.seed(42)
36  conditioned_lifetimes = simulate_conditioned_lifetime(1000)
37
38  # Summary statistics
39  mean_lifetime = np.mean(conditioned_lifetimes)
40  std_lifetime = np.std(conditioned_lifetimes)
41
42  print(f"Expected lifetime (conditioned): {mean_lifetime:.2f} months")
43  print(f"Standard deviation: {std_lifetime:.2f} months")
44
45  # Plot histogram
46  plt.hist(conditioned_lifetimes, bins=30, color='lightblue', edgecolor='black')
47  plt.xlabel('Lifetime (months)')
48  plt.ylabel('Number of women')
49  plt.title('Lifetime Distribution (Given Recurrence within 12 Months)')
50  plt.grid(True)
51  plt.show()
```

```
1   # -------------------------------------------
2   # TASK 5
3   # -------------------------------------------
4   n_simulations = 100
5   n_women_per_sim = 200
6   threshold = 350
7   np.random.seed(42)
8   fractions_crude = []
9   mean_lifetimes = []
10
11  # Simulation loop
12  for _ in range(n_simulations):
13      lifetimes, _, _ = simulate_woman_lifetime(n_women_per_sim)
```

```
14
15      fraction_dead = np.mean(np.array(lifetimes) <= threshold)
16      mean_lifetime = np.mean(lifetimes)
17
18      fractions_crude.append(fraction_dead)
19      mean_lifetimes.append(mean_lifetime)
20
21  fractions_crude = np.array(fractions_crude)
22  mean_lifetimes = np.array(mean_lifetimes)
23
24  # Compute crude mean and variance
25  mean_crude = np.mean(fractions_crude)
26  var_crude = np.var(fractions_crude, ddof=1)
27
28  print(f"Crude Monte Carlo estimate:")
29  print(f"  Mean = {mean_crude:.4f}")
30  print(f"  Variance = {var_crude:.6f}")
31
32  # Control variate adjustment
33  mu_Y = np.mean(mean_lifetimes)
34  cov_XY = np.cov(fractions_crude, mean_lifetimes, ddof=1)[0, 1]
35  var_Y = np.var(mean_lifetimes, ddof=1)
36  lambda_star = cov_XY / var_Y
37
38  fractions_cv = fractions_crude + lambda_star * (mu_Y - mean_lifetimes)
39
40  # Statistics with control variate
41  mean_cv = np.mean(fractions_cv)
42  var_cv = np.var(fractions_cv, ddof=1)
43
44  print("\nControl Variates estimate:")
45  print(f"  Mean = {mean_cv:.4f}")
46  print(f"  Variance = {var_cv:.6f}")
47
48  # Reduction in variance
49  reduction = 100 * (var_crude - var_cv) / var_crude
50  print(f"\nVariance reduction: {reduction:.2f}%")
51
52  # Plotting comparison
53  plt.figure(figsize=(10, 5))
54  plt.hist(fractions_crude, bins=15, alpha=0.6, label='Crude Estimate',
    ↪  color='skyblue')
55  plt.hist(fractions_cv, bins=15, alpha=0.6, label='Control Variates',
    ↪  color='salmon')
56  plt.axvline(mean_crude, color='blue', linestyle='--', label='Mean Crude')
57  plt.axvline(mean_cv, color='red', linestyle='--', label='Mean CV')
```

```
plt.xlabel("Estimated Fraction Dead < 350 months")
plt.ylabel("Frequency")
plt.title("Comparison of Crude vs Control Variates Estimates")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

## Part 2

```
# ------------------------------------------
# TASKS 7 & 8
# ------------------------------------------
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm, anderson_ksamp
from scipy.linalg import expm
import seaborn as sns
from scipy.stats._stats_py import PermutationMethod

np.random.seed(42)
Q = np.array([
    [-0.0085, 0.005, 0.0025, 0, 0.001],
    [0, -0.014, 0.005, 0.004, 0.005],
    [0, 0, -0.008, 0.003, 0.005],
    [0, 0, 0, -0.009, 0.009],
    [0, 0, 0, 0, 0]
])
Qs = Q[:4, :4]
p0 = np.array([1, 0, 0, 0])
ones = np.ones((4, 1))

# Simulation parameters
n_patients = 1000
max_time = 30.5

lifetimes = []
distant_metastasis = []

# Simulate patient trajectories
for _ in range(n_patients):
    state = 0
    time = 0
```

```
    had_distant = False
    while state != 4:
        rate = -Q[state, state]
        if rate <= 0:
            break
        sojourn = np.random.exponential(scale=1 / rate)
        time += sojourn
        probs = Q[state] / -Q[state, state]
        probs[state] = 0
        probs = probs / probs.sum()
        state = np.random.choice(range(5), p=probs)
        if time <= max_time and state == 2:
            had_distant = True
    lifetimes.append(time)
    distant_metastasis.append(had_distant)

lifetimes = np.array(lifetimes)
distant_metastasis = np.array(distant_metastasis)

# Summary statistics
mean_life = lifetimes.mean()
std_life = lifetimes.std(ddof=1)
ci_mean = norm.interval(0.95, loc=mean_life, scale=std_life /
↪  np.sqrt(n_patients))
ci_std = norm.interval(0.95, loc=std_life, scale=std_life / np.sqrt(2 *
↪  (n_patients - 1)))
distant_prop = np.mean(distant_metastasis)

# Theoretical CDF
times = np.linspace(0, lifetimes.max(), 300)
theoretical_cdf = [1 - (p0 @ expm(Qs * t) @ ones).item() for t in times]

# Save plots
plt.figure(figsize=(10, 6))
sns.histplot(lifetimes, bins=50, kde=True)
plt.title("Histogram of Simulated Lifetimes")
plt.xlabel("Time (months)")
plt.ylabel("Number of Women")
plt.grid(True)
plt.savefig("ctmc_lifetimes_histogram.png", bbox_inches='tight')
plt.show()
plt.close()

# Empirical CDF
sorted_lifetimes = np.sort(lifetimes)
empirical_cdf = np.arange(1, len(sorted_lifetimes)+1) / len(sorted_lifetimes)
```

```
78
79  # Simulated sample from theoretical CDF
80  simulated_theoretical_sample = np.interp(np.random.rand(n_patients),
    ↪  theoretical_cdf, times)
81
82  # Anderson-Darling k-sample test
83  ad_stat, ad_crit, ad_significance = anderson_ksamp(
84      [lifetimes, simulated_theoretical_sample],
85      method=PermutationMethod()
86  )
87  print("Mean lifetime: {:.2f} months (95% CI: {:.2f}, {:.2f})".format(mean_life,
    ↪  *ci_mean))
88  print("Standard deviation: {:.2f} months (95% CI: {:.2f},
    ↪  {:.2f})".format(std_life, *ci_std))
89  print("Proportion with distant metastasis before 30.5 months:
    ↪  {:.3f}".format(distant_prop))
90  print("Anderson-Darling test statistic: {:.3f}".format(ad_stat))
91  print("p-value (approx): > {:.2f}".format(ad_significance))
92  print("Critical values:", ad_crit)
93
94  # Plot CDF comparison
95  plt.figure(figsize=(10, 6))
96  plt.step(sorted_lifetimes, empirical_cdf, label="Empirical CDF", color='blue')
97  plt.plot(times, theoretical_cdf, label="Theoretical CDF", color='red',
    ↪  linestyle='--')
98  plt.title("Comparison of Empirical and Theoretical CDFs")
99  plt.xlabel("Time (months)")
100 plt.ylabel("Cumulative Probability")
101 plt.legend()
102 plt.grid(True)
103 plt.savefig("ctmc_empirical_vs_theoretical_cdf.png", bbox_inches='tight')
104 plt.show()
105 plt.close()
106
107 # Repeated AD tests and histogram of p-values
108 n_simulations = 100
109 p_values = []
110
111 for sim in range(n_simulations):
112     lifetimes_sim = []
113     for _ in range(n_patients):
114         state, time = 0, 0
115         while state != 4:
116             rate = -Q[state, state]
117             if rate <= 0:
118                 break
```

```
            sojourn = np.random.exponential(scale=1 / rate)
            time += sojourn
            probs = Q[state] / -Q[state, state]
            probs[state] = 0
            probs = probs / probs.sum()
            state = np.random.choice(range(5), p=probs)
        lifetimes_sim.append(time)

    lifetimes_sim = np.array(lifetimes_sim)

    times_sim = np.linspace(0, lifetimes_sim.max(), 300)
    theoretical_cdf_sim = [1 - (p0 @ expm(Qs * t) @ ones).item() for t in
    ↪  times_sim]
    simulated_sample = np.interp(np.random.rand(n_patients),
    ↪  theoretical_cdf_sim, times_sim)

    ad_stat, ad_crit, ad_significance = anderson_ksamp(
        [lifetimes_sim, simulated_sample],
        method=PermutationMethod()
    )
    p_values.append(ad_significance)

import matplotlib

# Plot histogram of p-values
import matplotlib.pyplot as plt
plt.close('all')
fig, ax = plt.subplots(figsize=(10, 6))
ax.hist(p_values, bins=20, edgecolor='black', alpha=0.75)
ax.axhline(y=n_simulations / 20, color='red', linestyle='--', label='Uniform
↪  Reference')
ax.set_title('Histogram of AD Test P-values from Repeated Simulations')
ax.set_xlabel('P-value')
ax.set_ylabel('Frequency')
ax.legend()
ax.grid(True)
fig.tight_layout()
fig.savefig("ad_test_pvalue_histogram.png")
plt.show()
)
```

```
# ------------------------------------------
# TASK 9
# ------------------------------------------
```

```python
# Simulation and Kaplan-Meier estimation for both cohorts
lifetimes_orig = simulate_ctmc_lifetimes(Q_orig,  N_SIM, rng)
lifetimes_trt  = simulate_ctmc_lifetimes(Q_treat, N_SIM, rng)

fig, ax = plt.subplots(figsize=(8, 5))
KaplanMeierFitter().fit(lifetimes_orig,
                        event_observed=np.ones_like(lifetimes_orig),
                        label="No treatment").plot(ax=ax, ci_show=False)
KaplanMeierFitter().fit(lifetimes_trt,
                        event_observed=np.ones_like(lifetimes_trt),
                        label="Preventive therapy").plot(ax=ax, ci_show=False)
```

```python
# ------------------------------------------
# TASK 10
# ------------------------------------------
# Two-sample log-rank test
from lifelines.statistics import logrank_test

logrank = logrank_test(lifetimes_orig, lifetimes_trt,
                       event_observed_A=np.ones_like(lifetimes_orig),
                       event_observed_B=np.ones_like(lifetimes_trt))
print(logrank.summary)
```

```python
# ------------------------------------------
# TASK 12
# ------------------------------------------
# Simulation of panel observations every 48 months
import math
import numpy as np
from numpy.random import default_rng
from typing import List, Tuple

Q_orig = np.array([
    [-0.0085, 0.005,  0.0025, 0.0,   0.001],
    [0.0,    -0.014,  0.005,  0.004, 0.005],
    [0.0,     0.0,   -0.008,  0.003, 0.005],
    [0.0,     0.0,    0.0,   -0.009, 0.009],
    [0.0,     0.0,    0.0,    0.0,   0.0 ]
])

def _precompute(Q: np.ndarray):
    rates = -np.diag(Q)
    jump  = np.where(Q > 0, Q, 0)
```

```
    jump  = jump / rates[:, None]
    return rates, jump

def simulate_ctmc_path(Q: np.ndarray, rng) -> Tuple[List[float], List[int]]:
    rates, jump = _precompute(Q)
    t, s = 0.0, 0
    times  = [0.0]
    states = [1]

    while s != 4:
        t += rng.exponential(scale = 1 / rates[s])
        s  = rng.choice(Q.shape[0], p = jump[s])
        times.append(t)
        states.append(s + 1)

    return times, states

def observation_sequence(times: List[float],
                         states: List[int],
                         step: float = 48.0) -> List[int]:
    obs, idx = [], 0
    t_obs    = 0.0
    t_final  = times[-1]
    t_last   = math.ceil(t_final / step) * step
    while t_obs <= t_last:
        while idx + 1 < len(times) and times[idx + 1] <= t_obs:
            idx += 1
        obs.append(states[idx])
        if states[idx] == 5:
            break
        t_obs += step
    return obs

def main():
    N_SIM, SEED = 1_000, 24
    rng         = default_rng(SEED)
    Y_obs = []
    for _ in range(N_SIM):
        t, s = simulate_ctmc_path(Q_orig, rng)
        Y_obs.append(observation_sequence(t, s))

    lengths = np.array([len(seq) for seq in Y_obs])
    print(f"Simulated {N_SIM} women.")
    print(f"Avg. # observations per woman: {lengths.mean():.2f} "
          f"(min {lengths.min()}, max {lengths.max()})")
    print("First three sequences:")
```

```
    for seq in Y_obs[:3]:
        print("  ", seq)
```

```
# ------------------------------------------
# TASK 13
# ------------------------------------------
# Generator re-estimation from 48-month panel data
n_states = 5
C = np.zeros((n_states, n_states), dtype=int)

for seq in Y_obs:  # panel data from Task 12
    for a, b in zip(seq[:-1], seq[1:]):
        C[a-1, b-1] += 1

row_tot = C.sum(axis=1, keepdims=True)
P_hat   = np.divide(C, row_tot, where=row_tot > 0)
P_hat[4, 4] = 1.0  # Absorbing state

from scipy.linalg import logm, expm
Delta = 48.0
Q_hat = logm(P_hat) / Delta
Q_hat = np.real_if_close(Q_hat, tol=1e-12)

for i in range(n_states):
    Q_hat[i, i] = -Q_hat[i, :].sum() + Q_hat[i, i]
print("Estimated Q (per month):")
print(Q_hat)
```

# References

[1] S. Asmussen and P. W. Glynn. *Simulation Modeling and Analysis*. Springer, 2015.

[2] Bo Friis Nielsen. Slides for 02443 stochastic simulation. `http://www.imm.dtu.dk/courses/02443/`, 2025.

[3] S. M. Ross. *Simulation*. Elsevier, 2013.