STA-510 Statistical modelling and simulation

Mandatory exercise 2

Christian Stigen
UiS, October 19th, 2017

How to run the code

All the code for this exercise can be found in `assignment-2.R`. I have made a single function for each problem. For example, to see the output of problem 1 (d), execute the function `problem1d()`. From the UNIX command-line, you can run

```
Rscript -e 'pdf("problem1b.pdf");
            source("assignment-2.R");
            problem1b()' > problem1b.out
```

to produce the textual output file `problem1b.out` and any plots in `problem1b.pdf`. In R Studio, you may want to reset the display with `par(mfrow=c(1,1))` before each call. All plots and R output was generated automatically at the same time as this document.

Problem 1 (a)

Because

$$\rho_{ab} = \frac{\text{Cov}(X_a, X_b)}{\sqrt{\text{Var}(X_a)\text{Var}(X_b)}} = \frac{\sigma_{ab}}{\sigma_a \sigma_b}$$

we have that $\sigma_{ab} = \rho_{ab}\sigma_a\sigma_b$ and $\sigma_{aa} = 1$. Furthermore, by definition $\text{Cov}(X, X) = \text{E}(X^2) - \text{E}(X)^2 = \text{Var}(X)$, so that $\rho_{aa} = 1$ and $\sigma_{aa} = \text{Var}(X_a) = \sigma_a^2$.

The **covariance matrix** is then

$$\Sigma(\mathbf{X}) = \begin{bmatrix} \rho_{11} & \rho_{12} & \rho_{13} \\ \rho_{21} & \rho_{22} & \rho_{23} \\ \rho_{31} & \rho_{32} & \rho_{33} \end{bmatrix} = \begin{bmatrix} 900 & -0.8 \cdot 30 \cdot 10 & 0.2 \cdot 30 \cdot 4 \\ \rho_{12} & 100 & -0.3 \cdot 10 \cdot 4 \\ \rho_{13} & \rho_{23} & 16 \end{bmatrix} = \begin{bmatrix} 900 & -240 & 24 \\ -240 & 100 & -12 \\ 24 & -12 & 16 \end{bmatrix}$$

while the **expectation vector** is

$$\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3)^T = (90, 48, 18)^T$$

Problem 1 (b)

Output from R using `rmvnorm` [1]:

```
P(X1 > 100, X2 > 50, X3 > 20) = 0.006092 (1e+06 simulations)
P(X1 < 100, X2 > 50, X3 < 15) = 0.123114 (1e+06 simulations)
P(X1 + 2X2 + 5X3 > 300)       = 0.188951 (1e+06 simulations)
```

We could also use the function `pmvnorm` to calculate the first two:

```
Alternative P(X1 > 100, X2 > 50, X3 > 20) = 0.006153271
Alternative P(X1 < 100, X2 > 50, X3 < 15) = 0.1229014
```

Problem 1 (c)

A top-tier character demands that both X_1 and X_2 are high at the same time. Therefore, we should have the highest probability of getting a top-tier character if they are positively correlated, so that a high X_1 implies a high X_2 and vice-versa.

(i) $\rho_{12} = -0.8$**,** $\rho_{13} = \rho_{23} = 0$

$$\Sigma(\mathbf{X}) = \begin{bmatrix} \rho_{11} & \rho_{12} & \rho_{13} \\ \rho_{21} & \rho_{22} & \rho_{23} \\ \rho_{31} & \rho_{32} & \rho_{33} \end{bmatrix} = \begin{bmatrix} 900 & -0.8 \cdot 30 \cdot 10 & 0 \\ -0.8 \cdot 10 \cdot 30 & 100 & 0 \\ 0 & 0 & 16 \end{bmatrix} = \begin{bmatrix} 900 & -240 & 0 \\ -240 & 100 & 0 \\ 0 & 0 & 16 \end{bmatrix}$$

This scenario should have the *least probability* of a top-tier character, because X_1 and X_2 are correlated negatively: When one is high, the other tends to be low.

(ii) $\rho_{12} = 0$**,** $\rho_{13} = \rho_{23} = 0$

$$\Sigma(\mathbf{X}) = \begin{bmatrix} \rho_{11} & \rho_{12} & \rho_{13} \\ \rho_{21} & \rho_{22} & \rho_{23} \\ \rho_{31} & \rho_{32} & \rho_{33} \end{bmatrix} = \begin{bmatrix} 900 & 0 \cdot 30 \cdot 10 & 0 \\ 0 \cdot 10 \cdot 30 & 100 & 0 \\ 0 & 0 & 16 \end{bmatrix} = \begin{bmatrix} 900 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 16 \end{bmatrix}$$

(iii) $\rho_{12} = 0.8$, $\rho_{13} = \rho_{23} = 0$

$$\Sigma(\mathbf{X}) = \begin{bmatrix} \rho_{11} & \rho_{12} & \rho_{13} \\ \rho_{21} & \rho_{22} & \rho_{23} \\ \rho_{31} & \rho_{32} & \rho_{33} \end{bmatrix} = \begin{bmatrix} 900 & 0.8 \cdot 30 \cdot 10 & 0 \\ 0.8 \cdot 10 \cdot 30 & 100 & 0 \\ 0 & 0 & 16 \end{bmatrix} = \begin{bmatrix} 900 & 240 & 0 \\ 240 & 100 & 0 \\ 0 & 0 & 16 \end{bmatrix}$$

This scenario should have the *highest probability* of a top-tier character, because the two values are positively correlated. That is, they both tend to be either high or low at the same time.

Problem 1 (d)

Output from R using `rmvnorm`:

```
Results for 1e+06 simulations
(i)   P(X1 > 100, X2 > 50, X3 > 20) = 0.008932
(ii)  P(X1 > 100, X2 > 50, X3 > 20) = 0.048027
(iii) P(X1 > 100, X2 > 50, X3 > 20) = 0.090955
```

As we can see from the output, we seem to have reasoned correctly in problem 1 (c) about which scenarios should have the highest and lowest probability of producing a top-tier character.

Problem 2 (a)

The transform method in the lectures seem to first have been presented by Çinlar in [2, p. 96]. While I do not have the book, the key theorem is reproduced in [3]. I have quoted that verbatim as theorem 1 below.

Theorem 1. *Let $\Lambda(t), t \geqslant 0$ be a positive-valued, continuous, nondecreasing function. Then the random variables $T_1, T_2, \ldots$ are event times corresponding to a nonhomogeneous Poisson process with expectation function $\Lambda(t)$ if and only if $\Lambda(T_1), \Lambda(T_2), \ldots$ are the event times corresponding to a homogeneous Poisson process with rate one.*

In other words, it relates a *homogeneous* Poisson process with a *constant rate of one* to a non-homogeneous Poisson process. We can actually go backwards by sampling event times from the homogeneous, rate 1 Poisson process, then go backwards through Λ^{-1} to then generate samples from the non-homogeneous process. See algorithm 1 on the next page.

In algorithm 1 on the following page, Λ^{-1} is the inverted rate function, while rpois is a function for generating numbers from a homogeneous Poisson. The algorithm operates in a for-loop. In the implementation, we will operate on vectors.

Note that we transform the domain $[a, b]$ to HPP-space by using Λ.

I have also tested Çinlar's method from [3] and found it to be equivalent to algorithm 1, although more compressed. The details are in algorithm 2 on the next page. It also seems to use an algorithm by Donald Knuth [4] to sample numbers from the Poisson distribution. A plot using Çinlar's method is given in plot 2 on page 8.

Finally, I have compared all of this with the code from the lecturer using the thinning method, shown in plot 3 on page 9.

Finding Λ and Λ^{-1}. In practice, this means that the rate function $\Lambda(t)$ must be readily reversible. That may not always be the case, but suffices for the current task. We will start by finding it.

Algorithm 1 Generates n numbers for the non-homogeneous Poisson process (NHPP)

```
1: function RHPP($\lambda, a, b$)                          ▷ Homogeneous Poisson process sampler
2:     $n \leftarrow 3(b-a)\lambda$                          ▷ Number of samples, arbitrary 3
3:     $w \leftarrow a + \text{cumsum}(\text{rexp}(n, \text{rate} = \lambda))$   ▷ Cum. sum of samples from exp. distribution
4:     return $w$ so that $a \leqslant w \leqslant b$
5: function RNHPP($n, \Lambda^{-1}$)                       ▷ Non-homogeneous Poisson process sampler
6:     $w \leftarrow$ RHPP($1, \Lambda(a), \Lambda(b)$)      ▷ Transforms $a$ and $b$ to HPP-space
7:     $s \leftarrow$ vector of length $|w|$
8:     for $i \leftarrow 1$ to $|w|$ do
9:         $s_i \leftarrow \Lambda^{-1}(w_i)$
10:    return $s$
```

Algorithm 2 Çinlar's method for NHPP

```
1: function CINLAR($n$)
2:     $s \leftarrow 0$
3:     $t \leftarrow$ vector of length $n$
4:     for $i \leftarrow 1$ to $n$ do
5:         $u \leftarrow U[0, 1]$                            ▷ Uniform sample
6:         $s \leftarrow s - \log(u)$
7:         $t_i \leftarrow \Lambda^{-1}(s)$
8:     return $t$
```

The intensity is given by $\lambda(t) = 14t^{0.4}$. We then have that

$$\Lambda(t) = \int_0^t \lambda(u)\,\mathrm{d}u = 14\int_0^t u^{0.4}\,\mathrm{d}u = \frac{14}{1.4}\left[u^{1.4}\right]_0^t = 10t^{1.4} = 10t^{\frac{7}{5}}$$

If we set $w = \Lambda(t)$, its inverse $\Lambda^{-1}(w) = t$ is given by

$$\begin{aligned} w &= 10t^{\frac{7}{5}} \\ w^{\frac{5}{7}} &= 10^{\frac{5}{7}}t \\ \Lambda^{-1}(w) = t &= 10^{-\frac{5}{7}}w^{\frac{5}{7}} \end{aligned}$$

Sampling from the entire period $t \in [0, 5]$ We simply use the same strategy as used in the lecture notes (and accompanying code) and put this functionality in the homogeneous Poisson process sampler. We calculate the expected number of events, and multiply that with a number (three in the submitted code). We then filter away values that are outside this range.

Problem 2 (b)

See the plot in figure 1 on the following page. Not shown here is a comparison with the thinning method as given by the lecturer. In my tests, the two plots looked similar, with equivalent domain and range.

Problem 2 (c)

In both [3, 5] we see that $\Lambda(t)$ is the expected number of failures in the period $[0, t]$, or $\Lambda(t) = \mathrm{E}\left[N(t)\right]$. So the expected number of failures during the first year is simply

$$\Lambda(1) = 10$$

The expected number of failures during the last year, i.e. from year four to five, will then be

$$\Lambda(5) - \Lambda(4) = 25.53865$$

For the other calculations, I have used R with the following strategy: I transformed t from NHPP-space to HPP space using the Λ function, then I calculated the probabilities using the definition of the Poisson distribution.

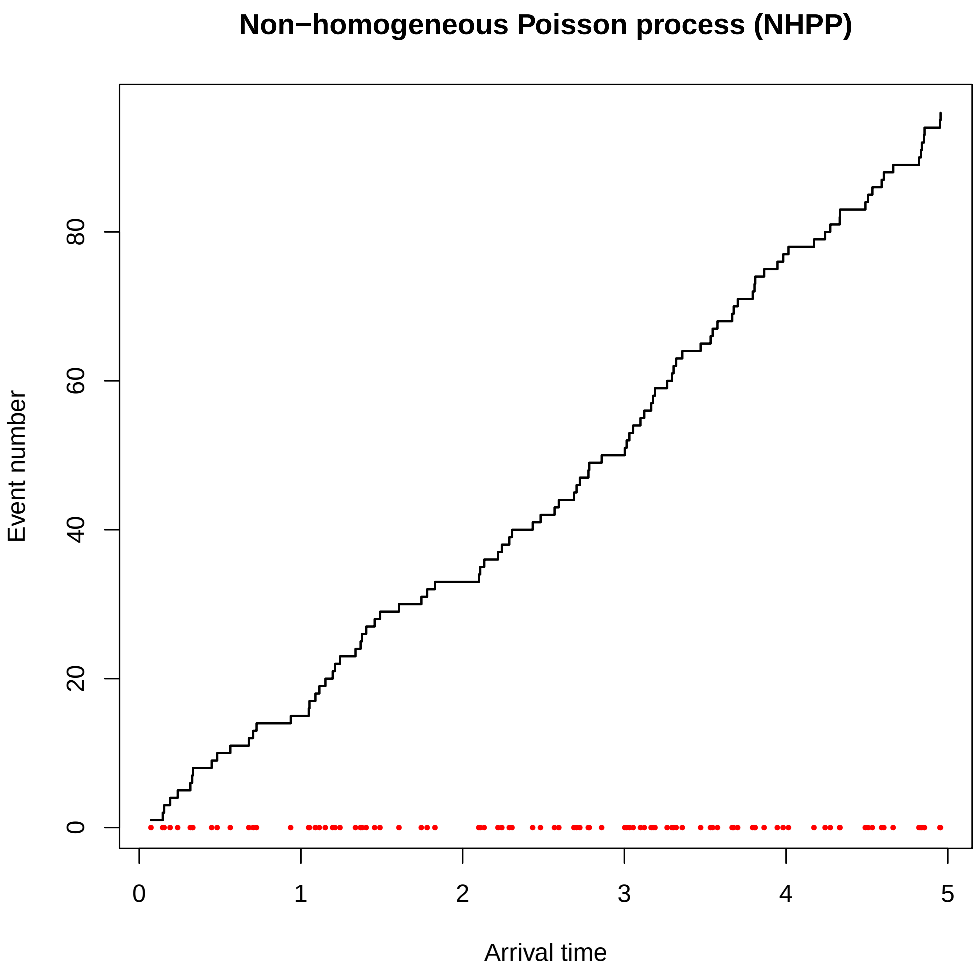

Figure 1: Plot for problem 2 (b).

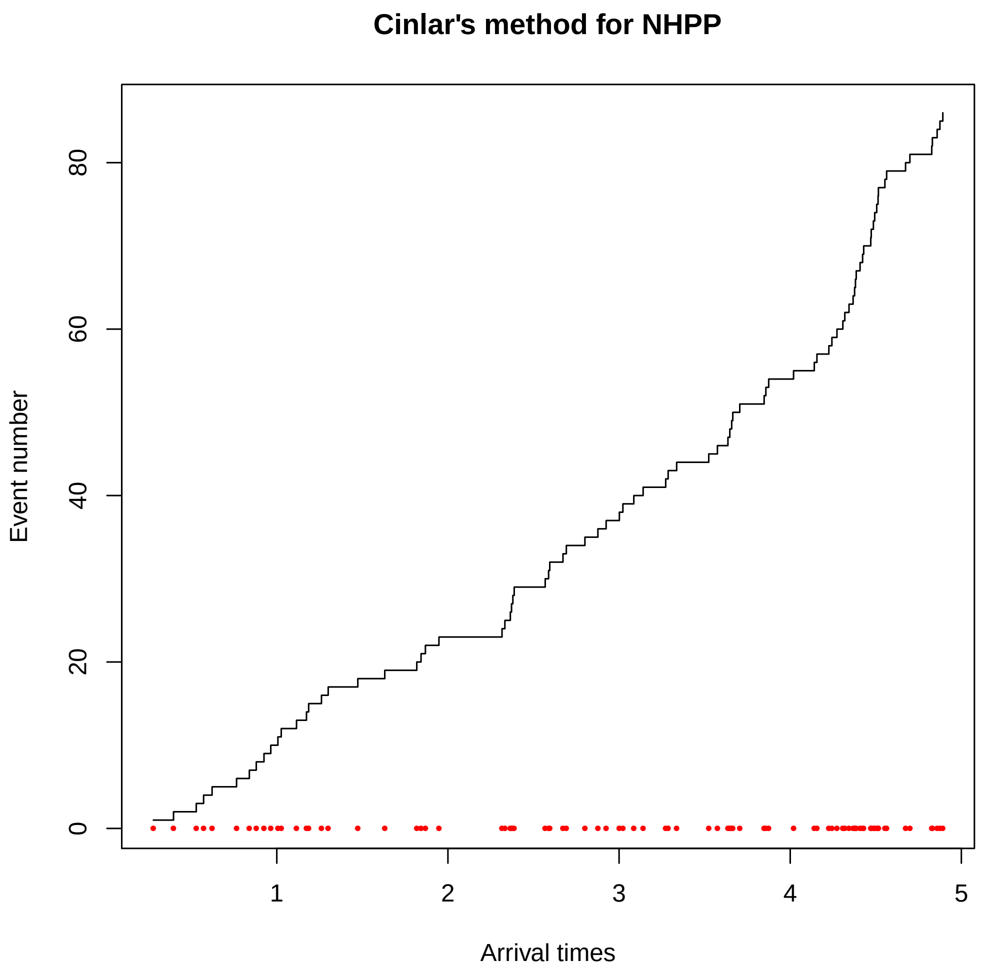

Figure 2: Plot of Çinlar’s algorithm for non-homogeneous Poisson process (NHPP), for comparison.

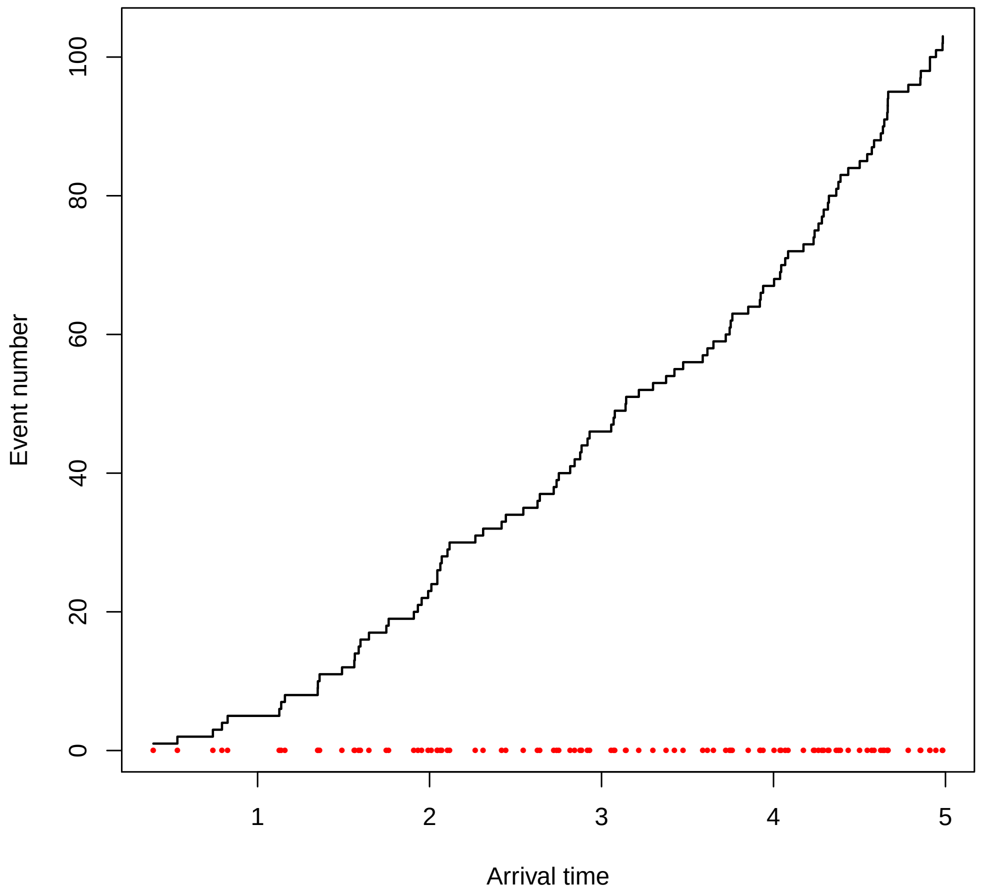

Figure 3: Plot of NHPP using the thinning method for comparison (lecturer's code)

Please see the code for problem 2c. The output is:

```
Exact P(more than 10 failures first year) = 0.4169602
Exact P(more than 20 failures last year) = 0.7909019
```

Problem 2 (d)

Output from R:

```
Expected failures last year: 10.0153 (10000 runs)
P(failures > 10 first year) = 0.4183 (10000 runs)

Expected failures last year: 25.5791 (10000 runs)
P(failures > 20 last year)  = 0.8404 (10000 runs)
```

Problem 2 (e)

The NHPP process gives us arrival times (i.e., points in time when a pump fails). For each of those we can draw a sample from the gamme distribution with shape $\alpha = 2$ and scale $\beta = 0.01$.

With that, we can do a number of things. We can calculate the total repair time for the entire five year period. This aggregate should obviously be less than five years, or else we will not be able to repair all the pumps within the period.

If we run the entire simulation several times, we can calculate the mean total repair time over the five year period.

Problem 2 (f)

See the plot in figure 4 on page 12. From the plot, we can see that the repair time definitely is less than five years.

Output from R:

```
Mean yearly, total repair time over 10000 runs = 1.903412
Ratio of mean(repair_time) / 5 years = 0.3806824
```

While the total repair time of ≈ 1.9 years is less than five, it means that on average, $\frac{1.9}{5} = 0.38$ or 38% of the equipment is in an unusable state of repair. That leaves only a capacity of $1 - 0.38 = 0.62$ of the equipment in service. In other words, the pumping capacity is almost half because of repairs.

My recommendation is to try to decrease the repair time, or buy better pumps.

Problem 2 (g)

Unfortunately, I have not completed this task.

How I would have attempted to solve it: Use multivariate analysis for the failure and repair times. If I can get an expression for the two, I should be able to calculate probabilities.

Problem 2 (h)

Unfortunately, I have not completed this task.

Suggested answer: Perhaps because our system works in whole years, it is more cumbersome to calculate transitions because all equations we use take the exact year as an argument.

Problem 2 (i)

Unfortunately, I have not completed this task.

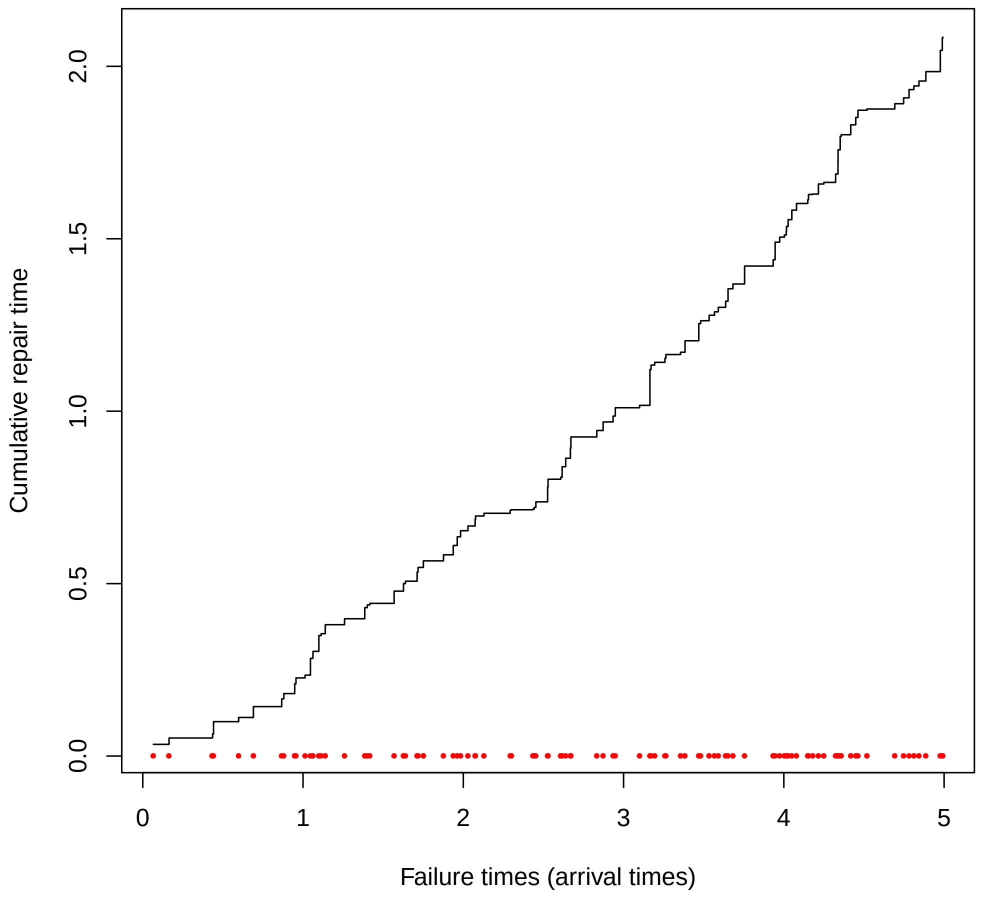

Figure 4: Plot of cumulative repair times for problem 2 (f).

Problem 3 (a)

The basic idea of Monte Carlo integration is to approximate an integral by sampling. By drawing random numbers in the plane, we can easily determine if the point falls above or below the integrand line by inserting the random x value in the integrand function. The integral, or area under the curve, should then be approximated with the ratio of points that fall above and below the line.

Approximating the integral with crude Monte Carlo integration. This will be a win in situations where the integral is impossible, hard or computationally expensive to perform — this applies especially for multi-dimensional integrals, where Monte Carlo integration scales linearly while true integration scales polynomially.

The *crude Monte Carlo integration* (CMC) is given by

$$\hat{\theta}_{\text{CMC}} = \frac{b-a}{n}\sum_{i=1}^{n} g(X_i) = (b-a)\overline{g(X)}$$

Here, $f(x) = \frac{1}{b-a}$ is the density function, and n is the number of samples drawn from $X \sim U[a,b]$. By the law of large numbers, $\hat{\theta}_{\text{CMC}} \to \theta$ as $n \to \infty$. The derivation of the above equation is given in the lecture notes.

Estimating the number of simulations. To have a $(1-\alpha)100\%$ probability that $\hat{\theta}_{\text{CMC}}$ is at most a margin e from θ we need use the empirial standard deviation equation.

$$SD(\hat{\theta}_{\text{CMC}}) = \frac{(b-a)\sigma_{g(X)}}{\sqrt{n}}$$

which gives us

$$n > \frac{z_{\alpha/2}^2 (b-a)^2 \sigma_{g(x)}^2}{e^2} = \sigma_{g(x)}^2 \left(\frac{z_{\alpha/2}(b-a)}{e}\right)^2$$

We can approximate $\sigma_{g(X)}^2$ as well with

$$\widehat{\sigma_{g(X)}^2} = \frac{1}{1-n}\sum_{i=1}^{n}\left(g(X_i) - \overline{g(X)}\right)^2$$

where $X \sim [a,b]$.

This gives us a probabilistic lower bound for n so that $\hat{\theta_{\text{CMC}}}$ deviates no more than $e = 100$ from the true θ in $(1-\alpha)100 = 95\%$ of the time. Of course, this value will depend heavily on accuracy of $\widehat{\sigma_{g(X)}}$ and thus on the number of samples used to approximate it.

I have implemented this estimation in the code for problem 3 (b) on this page.

Problem 3 (b)

Output from R:

```
Approximation of sigma.hat

  sigma.hat       = 89672.01 (10000 runs)
  ceiling(n.hat) = 19842 (alpha=0.05 e=100)

Approximation and exact value of integral

  theta.hat  = 10893.26 (19842 runs)
  theta      = 10800 (from Wolfram Alpha)
  difference = 93.25802

Over 1000 runs, how many times did our theta.hat fall within
e = 100 of the actual theta ?

  empirical.alpha = 0.94
```

In the above program, we also tested the choice of $\hat{n}$ (`n.hat`, the number of simulations to find a good $\hat{\theta}$) over a number of runs. According to confidence intervals, $100(1-\alpha)\%$ of the time, $\hat{\theta}$ should fall within $e = 100$ of the actual θ.

Problem 3 (c)

The *hit or miss Monte Carlo integration* (HM) is done by choosing a constant c so that $0 \leqslant g(x) \leqslant c$ for $x \in [a, b]$. We then draw random points in this area with $X \sim U[a, b]$ and $Y \sim U[0, c]$ and keep track of how many hit inside the area delimited by $g(x)$ and $y = 0$.

This should then enable us to approximate $\theta = \int_a^b g(x)\,\mathrm{d}x$. The number of hits should be equal to $\mathrm{E}\left[\mathrm{I}(Y \leqslant g(x))\right]$ and the rectangular area is given by $c(b-a)$, or

$$\theta = \int_a^b g(x)\,\mathrm{d}x = c(b-a)\,\mathrm{E}\left[\mathrm{I}(Y \leqslant g(x))\right]$$

or

$$\hat{\theta}_{\text{HM}} = \frac{c(b-a)}{n}\sum_{i=1}^{n} \mathrm{I}(Y_i \leqslant g(X_i))$$

Choosing a suitable constant c. We need to find a suitable $c > \lambda(t)$ for $t \in [a, b]$. This can be done in R itself (where we use $g(x) = \lambda(t)$) by using

```
t <- optimize(g, interval=c(a, b), maximum=TRUE)
c <- ceiling(g(t$maximum))
```

This gives us $c = 1052$.

Estimating the number of simulations. The estimated standard error for HM is given by

$$\widehat{\text{SD}}(\hat{\theta}_{\text{HM}}) = c(b-a)\frac{\sqrt{\hat{p}(1-\hat{p})}}{\sqrt{n}}$$

where $p = \mathrm{P}(Y \leqslant g(X))$ is estimated with

$$\hat{p} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{I}(Y_i \leqslant g(X_i))$$

That means we can find the number of n simulations required to be less than e away from the true θ with a $(1-\alpha)$ confidence interval by

$$\begin{aligned} z_{\alpha/2}\,\widehat{\text{SD}}(\hat{\theta}_{\text{HM}}) &< e \\ z_{\alpha/2}\,c(b-a)\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} &< e \\ n &> \hat{p}(1-\hat{p})\left(\frac{z_{\alpha/2}\,c(b-a)}{e}\right)^2 \end{aligned}$$

We can now estimate $\hat{p}$ using using $\alpha = (1-0.95) = 0.05$ and $e = 100$. Output from R:

```
Maximum value of g(x) between [0, 24]
  c = 1052 (ceiling at t = 18.13871)

Estimation of p
  p.hat = 0.4256 (over 10000 runs)

Estimation of n for HM Monte Carlo
  e = 100
  alpha = 0.05
  ceiling(n.hat) = 59865 (number of runs for HM)
```

As we can see from the output[1], we will require around 60 thousand runs to estimate $\hat{\theta}_{\text{HM}}$ with the given confidence. That is much larger than what we found for the crude Monte Carlo (CMC), which required 18–20 thousand runs. The reason is that in HM, the underlying information is compressed into a strictly binary outcome (i.e., zero or one), which throws away much of the information. We then take the mean of the binary values, losing information like, for example, how far away the sampled point was from $\lambda(t)$. Thus, getting a good mean of the binary values simply requires many more of them.

Finally, the assignment asks for an *upper* bound on n. This may be a semantic issue, but I would believe that what I have calculated is a *lower* bound: The least n that gives us the desired confidence interval.

Problem 3 (d)

Output from R:

```
Estimation of theta.hat.hm with 95% conf. int. for e = 100
  theta.hat.hm = 10789.73 (60000 runs)
  theta.exact  = 10800
  difference   = 10.2672

Over 1000 runs, how many times did our theta.hat.hm fall within
e = 100 of the actual theta ?

  empirical.alpha = 0.955
```

[1]The R output is always run whenever I reproduce this PDF document. Therefore, each run will always be slightly different.

References

[1] X. Mi, T. Miwa, and T. Hothorn, “mvtnorm: New numerical algorithm for multivariate normal probabilities,” *The R Journal*, vol. 1, no. 1, pp. 37–39, 2009.

[2] E. Cinlar, *Introduction to stochastic processes*. Courier Corporation, 2013.

[3] R. Pasupathy, “Generating Nonhomogeneous Poisson Processes,” *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[4] D. E. Knuth, “Seminumerical Algorithms, volume 2 of The Art of Computer Programming,” *Reading, MA*, 1969.

[5] M. L. Rizzo, *Statistical Computing with R*. CRC Press, 2007.