STA-510 Statistical modelling and simulation

Mandatory exercise 1

Christian Stigen
UiS, September 21st, 2017

How to run the code

All the code for this exercise can be found in `assignment-1.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-1.R");
            problem1b()' > problem1b.out
```

to produce the textual output file `problem1b.out` and any plots in `problem1b.pdf`. All plots and R output was generated automatically at the same time as this document.

Problem 1 (a)

The probability density function is given in [1] as

$$f(x) = \begin{cases} \frac{1}{\beta} e^{-\frac{x}{\beta}} & x > 0 \\ 0 & \text{elsewhere} \end{cases}$$

We then have that

$$\mathrm{P}(X \leqslant x) = \int_{-\infty}^{x} f(t)\,\mathrm{d}t = \left[-e^{-\frac{t}{\beta}} \right]_0^x = 1 - e^{-\frac{x}{\beta}}$$

Problem 1 (a i)

$$\mathrm{P}(X > 4000) = 1 - \mathrm{P}(X \leqslant 4000) = e^{-\frac{4000}{5000}} = 0.44932896\ldots$$

Problem 1 (a ii)

$$P(4000 \leqslant X \leqslant 6000) = P(X \leqslant 6000) - P(X \leqslant 4000) = e^{-\frac{4000}{5000}} - e^{-\frac{6000}{5000}} \approx 0.148$$

Note that for continuous density functions, $P(X = x) = 0$. Therefore I have not been diligent in the use of the less-than-or-equal symbols here.

Problem 1 (b)

Output from R using `pexp`:

```
P(X > 4000) = 0.449329
P(4000 <= X <= 6000) = 0.1481348
```

Problem 1 (c)

Finding $P(X > 4000)$ with simulations. Output from R:

```
P(X > 4000) 10 simulations: 0.3
P(X > 4000) 100 simulations: 0.5
P(X > 4000) 10000 simulations: 0.44
P(X > 4000) 100000 simulations: 0.4486
```

We can clearly see that as we increase the number of simulations, the probability converges to the exact value of $0.4493\cdots$. However, it does converge somewhat slowly.

Problem 1 (d)

Using simulations to approximate the probability that the sum of the lifetimes of two light bulbs is more than 10 thousand hours. Output from R:

```
P(X + X > 10000) 10 simulations: 0.2
P(X + X > 10000) 100 simulations: 0.35
P(X + X > 10000) 1000 simulations: 0.4
P(X + X > 10000) 10000 simulations: 0.4039
```

```
P(X + X > 10000) 100000 simulations: 0.40695
P(X + X > 10000) 1000000 simulations: 0.405142
```

Problem 1 (e)

Using the `pgamma` function in R gives the output:

```
Exact P(X + X > 10000) = 0.4060058
```

This is the exact solution for problem 1 (d). As we can see, our simulation seems to give a good approximation.

Problem 2 (a)

For each round, we draw seven random numbers from 1–34. We count how many draws contain at least two consecutive number (c) and divide by how many total draws we have made (n). The probability will then be

$$\mathrm{P}(\text{contains two or more consecutive numbers}) = \frac{c}{n}$$

We have

$$\begin{aligned} e &> z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \\ \left(\frac{e}{z_{\alpha/2}}\right)^2 &> \frac{\hat{p}(1-\hat{p})}{n} \\ n &\geqslant \left\lceil \hat{p}(1-\hat{p})\frac{z_{\alpha/2}^2}{e^2} \right\rceil \end{aligned}$$

where $\lceil \ldots \rceil$ is the ceiling function and e is the margin of error.

I wrote a test simulation that seems to indicate that the $\hat{p} = 0.779$. The z-value must be 1.96 for a 95% interval ($z_{0.025} = 1.96$), and we have the margin of error $e = 0.01$, which gives us

$$n \geqslant 6614$$

That suggests that we should have *at least* 6614 samples.

Problem 2 (b)

Simulating the probability that seven out of 34 numbers contain at least two consecutive numbers was done in R with the output:

```
n=10: 0.8
n=100: 0.793
n=1000: 0.768
n=10000: 0.7752
95% certain of error at most 0.01 n=6614: 0.7821288
```

Problem 3 (a)

We want to sample x-values whose distribution is given by a probability density function (PDF) is $f(x)$.

We will use the *inverse transform method* [2]: Since the values given by the CDF is in the range 0–1, we can select u by drawing a number from a uniform pseudo-random number generator. We then use the inverse CDF function $g^{-1}(u)$ to find which x-value this corresponds to. This requires that it is possible to invert the CDF.

The CDF is given by

$$g(x) = \int_{-\infty}^{x} f(x)\,\mathrm{d}t = \left[-e^{-\frac{t^2}{2\theta^2}}\right]_0^x = 1 - e^{-\frac{x^2}{2\theta^2}}, \text{ where } x > 0 \text{ and } \theta > 0$$

We now need to find $g^{-1}(u)$ for

$$\begin{aligned} g(g^{-1}(u)) &= 1 - e^{-\frac{g^{-1}(u)^2}{2\theta^2}} = u \\ e^{\frac{g^{-1}(u)^2}{2\theta^2}} &= \frac{1}{1-u} \\ g^{-1}(u)^2 &= 2\theta^2 \log\left(\frac{1}{1-u}\right) \\ g^{-1}(u) &= \pm\theta\sqrt{2\log\left(\frac{1}{1-u}\right)} \end{aligned}$$

Note that since $u < 1$ then $\log(1-u) < 1$, meaning the following two equations are exactly the same:

$$\theta\sqrt{-2\log(1-u)} = \theta\sqrt{2\log\left(\frac{1}{1-u}\right)} \text{ where } u \in [0,1\rangle$$

The left expression is how `rrayleigh` has been implemented in the `VGAM` package [3]. While it avoids one division, one could imagine that it would be numerically more stable. I made a test program using IEEE 754 floats [4] (which is what R uses), and it doesn't seem to be more stable.

As to the $\pm$ operator, the whole expression returns a value for x, and since we know $x > 0$ we can remove the minus sign. We then get

$$x(u) = g^{-1}(u) = \theta\sqrt{2\log\left(\frac{1}{1-u}\right)} \text{ where } x > 0\,, \theta > 0\,, u \in [0,1\rangle$$

The algorithm to generate samples from the Rayleigh distribution is given in algorithm 1. Note that in the actual implementation I made for the Rayleigh sampling, I used the `runif` function to draw uniform floating point samples. This function does *not* include its extreme values, meaning you only get $\langle 0,1\rangle$ in the default case. This is explained in the help for `runif`. I originally used `sample(1:10^decimals-1, n) / 10^decimals` to generate $[0,1\rangle$, but that was painstakingly slow compared to `runif`.

Algorithm 1 Generates n samples from the Rayleigh distribution

```
1: function RAYLEIGH($n, \theta$)
2:     samples ← vector of size $n$
3:     for $i \leftarrow 1$ to $n$ do
4:         $u \leftarrow \mathrm{uniform}(0,1)$                ▷ Random float $u \in [0,1\rangle$
5:         samples$[i] \leftarrow \theta\sqrt{2\log(1/1-u)}$
6:     return samples
```

Problem 3 (b)

A histogram of Rayleigh samples is given in figure 1 on the next page, while a comparison between the expected value, variance and their sampled counterparts is given in table 1 on the following page.

	Expected	Sampled
Mean	2.230899	2.229622
Variance	1.359889	1.358519

Table 1: Theoretical and sampled metrics for problem 3 (b). “Expected Mean” is actually the *expected value*.

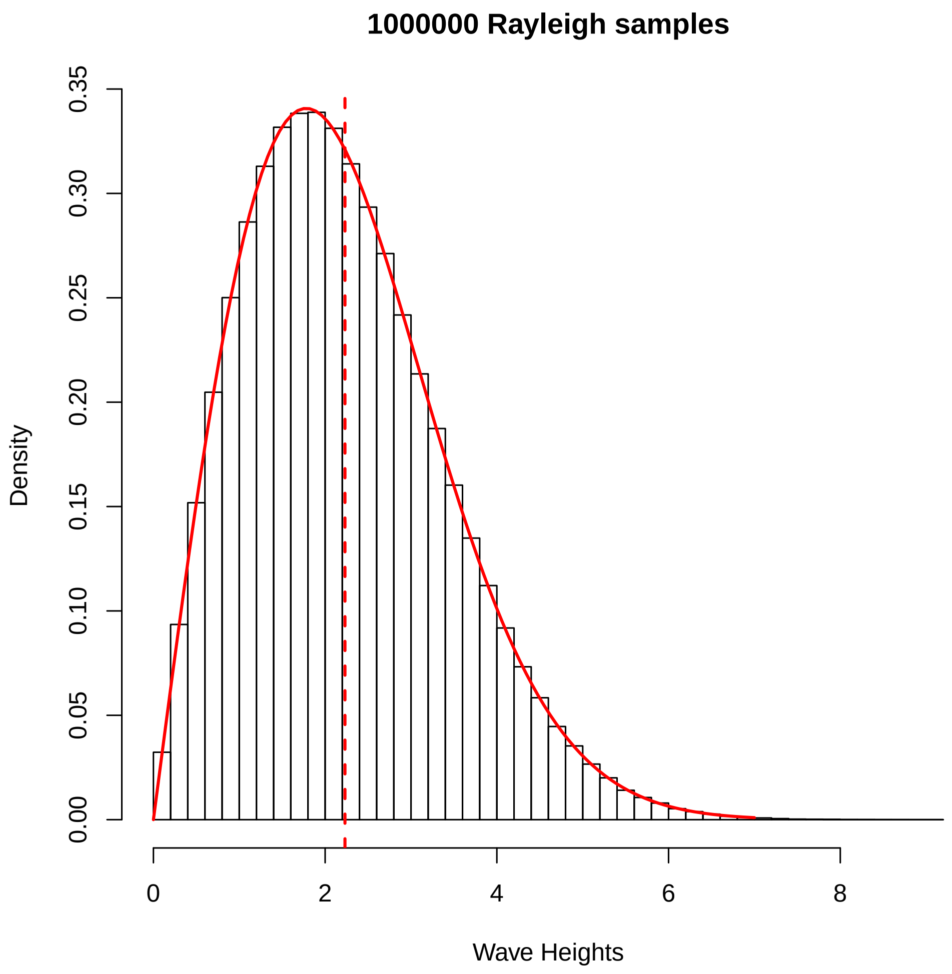

Figure 1: Rayleigh samples for problem 3 (b) with $\theta = 1.78$. The expected value and PDF are indicated by the red, dotted lines.

Problem 3 (c)

For one simulation: Set $\theta = 1.3$ and take 200 Rayleigh samples. Return 0 if all waves are less than or equal to 5 meters, 1 if not.

We then divide the total of a large number of such simulations with the number of simulations made.

Problem 3 (d)

Implemented the simulation described in problem 3 (c): The approximate probability that highest wave is higher than 5 meters. Output form R:

```
100 simulations, wave > 5 prob: 0.11
1000 simulations, wave > 5 prob: 0.116
10000 simulations, wave > 5 prob: 0.1101
100000 simulations, wave > 5 prob: 0.11531
1000000 simulations, wave > 5 prob: 0.115981
```

Problem 3 (e)

Acceptance-rejection sampling was first presented in [5] by the famous Jon von Neumann, and is a beautiful piece that every one should read.

Neumann argues that a good way to sample from a given distribution is to take two uniformly sampled numbers, then feed one of them into a computationally simpler function that is proportional to the desired PDF. By comparing with a scale factor (α in the article, c in the lecure notes), one can then throw away the numbers if they are higher than the comparison operation. He also shows how his algorithm enables to embed simpler mathematical expressions directly in the code, if applicable.

In [6] we see that the *inverse* of the triangle distribution does not have a closed form, and therefore cannot be used with the inverse transform method. The paper presents multiple alternatives, such as the MINIMAX-method, but also presents a novel method of calculating the triangle PDF in one line. However, I have used the triangle PDF that I developed in my code.

For details on how I have implemented this, see algorithm 2 on page 8 and the implementation in `assignment-1.R`. The number of expected iterations until acceptance will, with $c = 1.333\ldots$, we would expect around cn iterations until acceptance for n simulations.

Algorithm 2 Generates n samples from the triangle distribution

```
1:  function PTRIANGLE($x, a, b, c$)
2:      if $x \leqslant c$ then
3:          return $\frac{2}{(b-a)(c-a)}(x-a)$
4:      else
5:          return $\frac{2}{(b-a)(b-c)}(b-x)$
6:  function RTRIANGLE($n, a, b, c$)
7:      $c \leftarrow 2/(b-a)$                          ▷ Constant so that $\forall t : \frac{f(t)}{g(t)} \leqslant c$
8:      $f(x) \leftarrow$ PTRIANGLE($x, a, b, c$)        ▷ Partial application
9:      samples $\leftarrow$ numeric vector of size $n$
10:     for $i \leftarrow 1$ to $n$ do
11:         repeat
12:             $u \leftarrow$ uniform$(0, 1)$
13:             $y \leftarrow$ uniform$(a, b)$                  ▷ $y \in [a, b]$
14:         until $u \leqslant \frac{f(y)}{c}$
15:         samples$[i] \leftarrow y$
16:     return samples
```

Problem 3 (f)

For a plot of algorithm 2, see figure 2 on the following page.

Problem 3 (g)

The probability that a *single* wave is smaller or equal to y is given by the CDF in problem 3 (a) on page 4:

$$\mathrm{P}(X \leqslant y) = 1 - e^{-\frac{y^2}{2\theta^2}}$$

The probability that m waves are smaller than or equal to y will then be

$$\mathrm{P}(Y \leqslant y) = \mathrm{P}(X \leqslant y)^m = \left(1 - e^{-\frac{y^2}{2\theta^2}}\right)^m$$

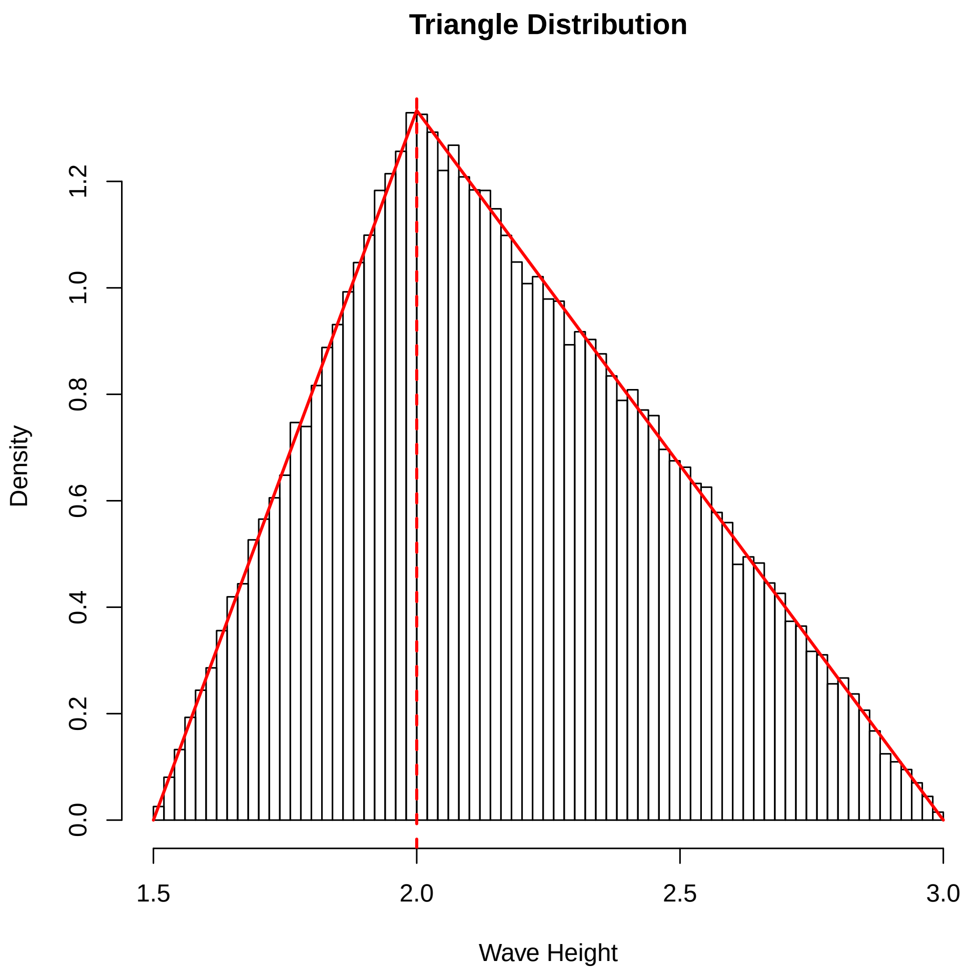

Figure 2: Acceptance-rejection sampling from the triangle distribution with $a = 1.5$, $b = 3$ and $c = 2$ for problem 3 (f).

because each wave is independent of another (although likely untrue in real life).

Now, the reverse would be that *at least one* wave is higher than y. We can express that as

$$\mathrm{P}(Y > y) = 1 - \mathrm{P}(Y \leqslant y) = 1 - \left(1 - e^{-\frac{y^2}{2\theta^2}}\right)^m$$

■

The exact calculation with $\theta = 1.3$ and $m = 200$ gives

$$\mathrm{P}(Y > 5) = 1 - \left(1 - e^{\frac{-y^2}{2\theta^2}}\right)^m = 1 - \left(1 - e^{\frac{-5^2}{2\cdot 1.3^2}}\right)^{200} = 0.11549135704657076\ldots$$

This corresponds quite well with the output in problem 3 (d) on page 7, although I do notice that my algorithm requires a very large number of samples to start converging to this value.

Problem 3 (h)

Output from R:

```
Generated 100000 maximum height samples.

Mean probability of critical wave: 0.3260729
Standard deviation of critical wave probability: 0.3499587

Minimum:       3.112135e-05
1st Quartile: 0.026802
Median:        0.1596781
Mean:          0.3260729
3rd Quartile: 0.6133412
Maximum:       0.9999844

95% confidence interval [0.3246387, 0.3275071]

We require that the confidence interval extreme is below 0.1
Recommendation: DO NOT PROCEED
```

Problem 3 (i)

We could sample m from a distribution — for example the normal distribution, or perhaps the triangle distribution based on input from the meteorologist and the weather forecast.

The next step would then be to repeat the experiment with different values for m. An automatic recommendation could be generated for each experiment (i.e., a binomial output) and summed together and divided by the number of experiments. Or we could aggregate confidence intervals for each experiment and then find out which interval would be most likely. This could be done by, for example, finding overlapping regions (like in interval arithmetic) and reporting on the most frequent. Or one could turn that into a frequency distribution of some kind and look at that.

I feel that the best would be to base everything on actual, observable data, like the weather forecast, consult with the opinion of the meteorologist, consider details about the operation and facility (e.g., tear and wear of the installation) and use that to make a more informed opinion.

References

[1] R. E. Walpole, R. H. Myers, S. L. Myers, and K. Ye, *Probability and statistics for engineers and scientists*, 5th ed. Macmillan New York, 1993.

[2] Wikipedia, "Inverse transform sampling — Wikipedia, The Free Encyclopedia," 2017, [Online; accessed 19-September-2017]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Inverse_transform_sampling&oldid=791760634

[3] T. W. Yee, "VGAM — Vector Generalized Linear and Additive Models," 2017, [Online; accessed 20-September-2017]. [Online]. Available: https://github.com/cran/VGAM/blob/2237f825c6696322aece806e1b53580538687c7b/R/family.univariate.R#L7306-L7310

[4] *IEEE standard for binary floating-point arithmetic*. New York: Institute of Electrical and Electronics Engineers, 1985, note: Standard 754–1985.

[5] J. Von Neumann, "Various techniques used in connection with random digits, paper no. 13 in in "monte carlo method"," *NBS Applied Mathematics Series*, no. 12, 1961.

[6] W. E. Stein and M. F. Keblis, "A new method to simulate the triangular distribution," *Mathematical and Computer Modelling*, vol. 49, no. 5, pp. 1143 – 1147, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0895717708002665