STM4PSD – Workshop 6 Solutions

1. Some possible answers are given. There are many correct approaches; verify that you get the same result in each case.

   (a) i. `pexp(1, rate=2)` or `pexp(1, 2)`

   ii. The answer is the same as in (i), because for continuous random variables, strict and non-strict inequalities are interchangeable.

   iii. `1 - pexp(1, rate=2)` or `pexp(1, rate=2, lower.tail=FALSE)`

   iv. `pexp(3, rate=2) - pexp(1, rate=2)`

   (b) i. `pgamma(2, shape=4, scale=1)` or `pgamma(2, 4, 1)`

   ii. The answer is the same as in (i), because for continuous random variables, strict and non-strict inequalities are interchangeable.

   iii. `1 - pgamma(2, shape=4, scale=1)` or `pgamma(2, shape=4, scale=1, lower.tail=FALSE)`

   iv. `pgamma(4, shape=4, scale=1) - pgamma(2, shape=4, scale=1)`

2. -

3.
```
f <- function(x) {
    ifelse(x >= 1 & x <= 1.5,
        4*x - 4,
        ifelse(x > 1.5 & x <= 2,
            8 - 4*x,
    0))
}
```

   Alternatively:

```
f <- function(x) {
    if (x >= 1 & x <= 1.5) { 4*x - 4 }
    else if (x > 1.5 & x <= 2) { 8 - 4*x }
    else { 0 }
}
f <- Vectorize(f)
```

4.
```
# P(X <= 1.4):
integrate(f, lower=-Inf, upper=1.4)
# P(X >= 1.7):
integrate(f, lower=1.7, upper=Inf)
# P(1.4 <= X <= 1.6):
integrate(f, lower=1.4, upper=1.6)
# P(X <= 1.6):
integrate(f, lower=-Inf, upper=1.6)
```

   Note that they are not all exact (e.g. the fourth command results in $P(X \leqslant 1.6) = 0.6799983$) but they are correct to a reasonable degree of accuracy.

5.
```
used.for.ev <- function(x) { x * f(x) }
integrate(used.for.ev, lower=-Inf, upper=Inf)
```

   which returns 1.5.

   Alternatively:

```
integrate(function(x) { x * f(x) }, lower=-Inf, upper=Inf)
```

6.
```
expected.value <- function() {
    integrate(used.for.ev, lower=-Inf, upper=Inf)$value
}
```

```
used.for.variance <- function(x) {
    mean <- expected.value()
    (x - mean)^2 * f(x)
}
integrate(used.for.variance, lower=-Inf, upper=Inf)
```

which returns 0.04166708.

Alternatively,

```
mean <- integrate(function(x) { x * f(x) }, lower=-Inf, upper=Inf)$value
integrate(function(x) { (x - mean)^2 * f(x) }, lower=-Inf, upper=Inf)
```

7.
```
my.normal <- function(x, mu, sigma) {
    1/(sigma * sqrt(2*pi)) * exp(-0.5 * ((x-mu)/sigma)^2)
}
```

8. This approach uses anonymous functions The indentation and line breaks used in the integrate function are used because of space limitations on the page, and are not required in your answers.

```
my.ev <- function(mu, sigma) {
    integrate(function(x) { x*my.normal(x, mu, sigma) },
              lower=-Inf,
              upper=Inf)$value
}
my.var <- function(mu, sigma) {
    ev <- my.ev(mu,sigma)
    integrate(function(x) { (x-ev)^2*my.normal(x, mu, sigma) },
              lower=-Inf,
              upper=Inf)$value
}
```

For an alternative approach, you can also define functions within other functions

```
my.ev <- function(mu, sigma) {
    temp.function <- function(x) { x*my.normal(x, mu, sigma) }
    integrate(temp.function, lower=-Inf, upper=Inf)$value
}
my.var <- function(mu, sigma) {
    ev <- my.ev(mu,sigma)
    temp.function <- function(x) { (x-ev)^2*my.normal(x, mu, sigma) }
    integrate(temp.function, lower=-Inf, upper=Inf)$value
}
```

Of course, since we know the mean mean and variance for a normal distribution can be expressed in terms of the parameters, a simpler approach would be:

```
my.ev <- function(mu, sigma) { mu }
my.var <- function(mu, sigma) { sigma^2 }
```

But the purpose of the exercise is to practice using the integrate function!

9. -