STM4PSD – Workshop 4 Solutions

1. The following code will calculate each part in sequence.

```
# part (a): P(X = 10)
dbinom(10, size = 100, prob = 0.3)

# part (b): P(X <= 10)
pbinom(10, size = 100, prob = 0.3)

# part (c): P(X < 10) = P(X <= 9)
pbinom(9, size = 100, prob = 0.3)

# part (d): P(X > 10) = 1 - P(X <= 10)
1 - pbinom(10, size = 100, prob = 0.3)

# part (d): alternative approach:
#   using lower.tail = FALSE tells R to calculate P(X > a) instead.
pbinom(10, size = 100, prob = 0.3, lower.tail = FALSE)

# part (e): P(X >= 10) = 1 - P(X < 10) = 1 - P(X <= 9)
1 - pbinom(9, size = 100, prob = 0.3)

# part (e): alternative approach:
#   P(X >= 10) = P(X > 9), so instead:
pbinom(9, size = 100, prob = 0.3, lower.tail = FALSE)

# part (f): P(5 <= X < 10) = P(X < 10) - P(X < 5)
#                          = P(X <= 9) - P(X <= 4)
pbinom(9, size=100, prob=0.3) - pbinom(4, size=100, prob=0.3)

# part (f): alternative approach by summing probabilities from 5 to 9
sum(dbinom(5:9, size=100, prob=0.3))
```

2. (a) Calculate $P(X = 3)$ using `dbinom(3, 10, 0.3)`
   (b) Calculate $P(X \leqslant 1)$ using `pbinom(1, 10, 0.3)`
   (c) Calculate $P(X \geqslant 2) = 1 - P(X < 1)$ using `1 - pbinom(1, 10, 0.3)`
   (d) Calculate $P(X > 8) = 1 - P(X \leqslant 8)$ using `1 - pbinom(8, 10, 0.3)`
   or by using `pbinom(8, 10, 0.3, lower.tail=FALSE)`
   (e) Calculate $P(X \leqslant 9)$ using `pbinom(9, 10, 0.3)`
   (f) Calculate $P(2 \leqslant X \leqslant 9) = P(X \leqslant 9) - P(X < 2) = P(X \leqslant 9) - P(X \leqslant 1)$ using
   `pbinom(9, 10, 0.3) - pbinom(1, 10, 0.3)`

3. By reading the documentation for `dnbinom`, we can see the syntax for the negative binomial distribution.
   Note that `size` refers to the required number of successes, and `prob` refers to the probability of success.
   For $Y \sim \text{NB}(3, 0.15)$:
   (a) Calculate $P(Y = 3)$ using `dnbinom(3, size=3, prob=0.15)`
   (b) Calculate $P(Y \leqslant 1)$ using `pnbinom(1, size=3, prob=0.15)`
   (c) Calculate $P(Y \geqslant 2) = 1 - P(Y \leqslant 1)$ using `1 - pnbinom(1, size=3, prob=0.15)`
   (d) Calculate $P(1 < Y \leqslant 3) = P(Y \leqslant 3) - P(Y \leqslant 1)$ using
   `pnbinom(3, size=3, prob=0.15) - pbinom(1, size=3, prob=0.15)`

4. Note that the geometric distribution is implemented in R using the `dgeom` and `pgeom` functions.

(a) -

(b) For $X \sim \text{Geo}(1/3)$, we find $P(X < 6)$ using $P(X \leqslant 5)$:

```
pgeom(5, prob=1/3)
```

OR

For $Y \sim \text{Bin}(6, 1/3)$, we find $P(X \geqslant 1)$ using $1 - P(X = 0)$:

```
1 - dbinom(0, size=6, prob=1/3)
```

(c) -

(d) For $X \sim \text{Bin}(365, \frac{20}{365})$, we find $P(X < 10)$ using $P(X \leqslant 9)$:

```
dbinom(9, size=365, prob=20/365)
```

(e) -

(f) For $X \sim \text{Bin}(5, 0.1)$, we find $P(X \geqslant 1)$ using $1 - P(X = 0)$:

```
1 - dbinom(0, size=5, prob=0.1)
```

(g) For $X \sim \text{Bin}(1250, 0.01)$, we find $P(X \leqslant 20)$ directly:

```
pbinom(20, size=5, prob=0.1)
```

5.
```
binom.less <- function(q, size, prob) {
    pbinom(q-1, size=size, prob=prob)
}

binom.geq <- function(q, size, prob) {
    1 - binom.less(q, size=size, prob=prob)
}

binom.between <- function(a, b, size, prob) {
    pbinom(b, size=size, prob=prob) - binom.less(a, size=size, prob=prob)
}
```

6. You should notice that increasing `n` makes the graph appear smoother.

7. The following code will produce the desired plot.

```
g <- function(x) { 3/4*(x+1)*(1-x) }
curve(g, from=-3, to=3, lwd=3)
abline(h=0)
abline(v=0)
```

8. Note that `1:6 == 3` and `1:6 >= 3` perform element-wise comparison.

9. The expression `0:6` is the vector containing the numbers from 0 to 6: `c(0,1,2,3,4,5,6)`.

   The expression `(0:6)^2` squares the numbers from 0 to 6, resulting in the vector `c(0,1,4,9,16,25,36)`.

   The expression `0:6 == (0:6)^2` then performs element-wise comparison. Since the only numbers that equal their own square are 0 and 1, it shows `TRUE` for the first two entries, and `FALSE` for the remaining entries.

   An aside: note that the brackets in `(0:6)^2` are necessary. Without the brackets, it will first square the number 6, then populate the list of numbers, resulting in a vector containing numbers from 0 to 36.

10. –

11. You should see a warning in the first case, and an error in the second. The reason behind this is explained in the lab notes.

12. Here we have given two approaches, which will produce the same plot in each case.

```
# For this example, we have used the traditional ``if'' syntax,
# and then made sure to Vectorize the function before plotting it.
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)
curve(f, from=0, to=2, lwd=3)

# For this approach, we have used the ifelse function,
# which ensures the function is appropriately vectorised.
f <- function(x) {
    ifelse(x >= 1 & x <= 1.5,
           4*x - 4,
           ifelse(x > 1.5 & x <= 2, 8 - 4*x, 0))
}
curve(f, from=0, to=2, lwd=3)
```

The indentation/line breaks in the second approach are not a requirement, and used here just to highlight the syntax.