# The University of British Columbia

Irving K. Barber School of Sciences

*DATA 101*

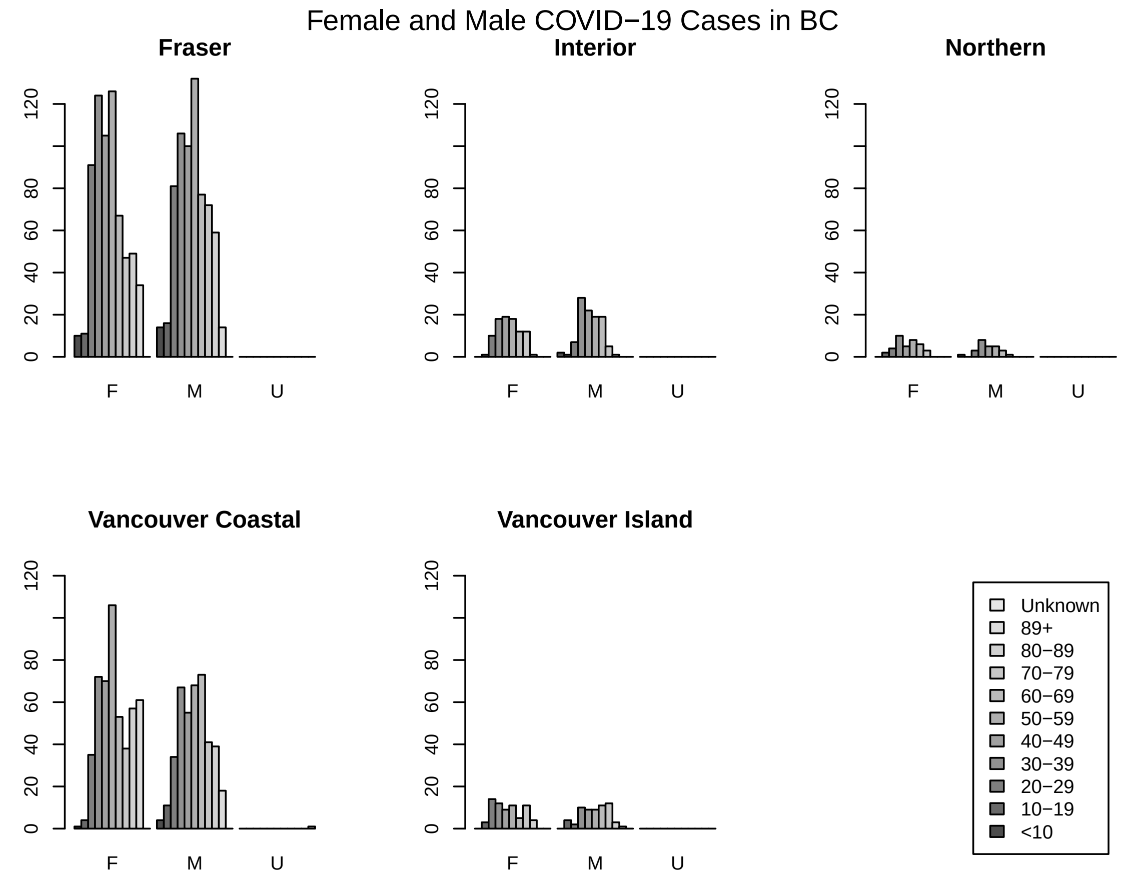
Lab 6

*Demonstration.* The TA will go through this set of examples with you.

1. The coronavirus has infected more than 2000 residents of BC. The data frame in *covidBC.R* contains information on the date infected, health area (`HA`), `Sex`, `Age_Group` as well as `Classification_Reported` for each infected patient.

   Split the data frame into a list of data frames, one for each health area. Within each health area, use the `table` function to construct a matrix of counts of `Age_Group` by `Sex`, and construct a bar plot of these counts, for each health area. (The same kind of code should be used as for the `VADeaths` bar plot studied in the BaseGraphics slides.) Ensure that the y-axis scale is the same for all five health areas. Finally, include a single legend in the remaining frame - with nothing else. The final output should appear as below.

```r
source("covidBC.R")
areas <- levels(covidBC$HA)
n <- length(areas)
covidArea <- split(covidBC, covidBC$HA)
par(mfrow=c(2, 3))
for (i in 1:n) {
      cov_table <-  with(covidArea[[i]],  table(Age_Group, Sex))
      barplot(cov_table, beside=TRUE, main=  areas[i], ylim=c(0, 125))
}
colnames(cov_table) <-  NULL
cov_table <- cov_table*NA
barplot(cov_table, legend=TRUE, ylim=c(0, 125), axes=FALSE)
mtext(side=3, line = -1.5, "Female and Male COVID-19 Cases in BC", outer = TRUE)
```



From the graph, identify the region where the fewest males were infected, and identify the region, sex and age group where there was the largest number of infections.

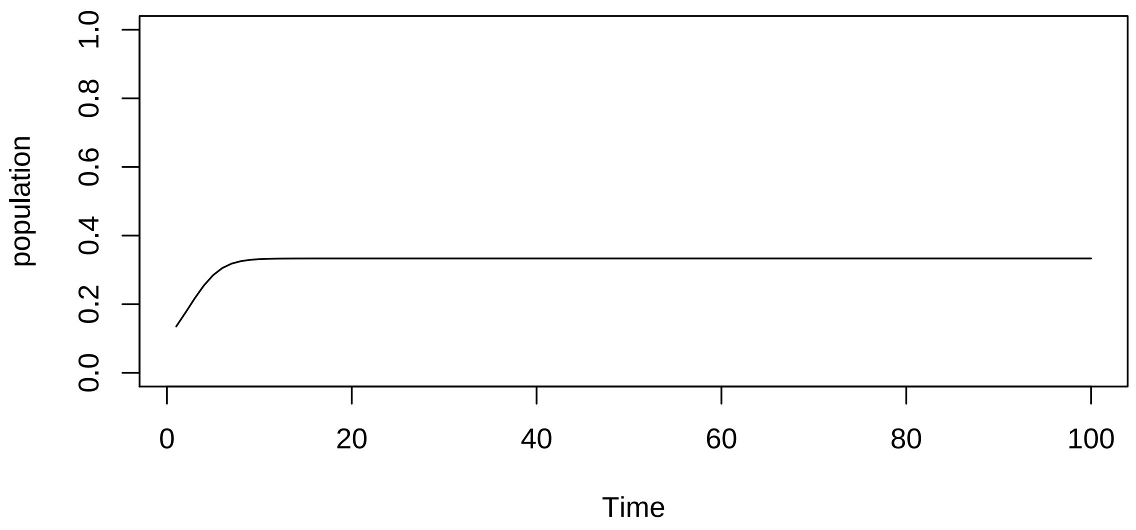
2. A simple, but reasonably realistic, model for population growth is based on something called the logistic equation:

$$x_{n+1} = Rx_n(1 - x_n)$$

where $R$ is a constant specifying the carrying capacity for the population, and $x_n$ is the proportion of the carrying capacity occupied by the population at time $n$, and $x_{n+1}$ is the proportion at the next time $n+1$. Note that when $x_n$ hits either 0 or 1, the population dies out.

Suppose the carrying capacity is $R = 1.5$, and $x_0 = 0.1$. Calculate $x_1, x_2, \ldots, x_{100}$ and create a time series object of length 100 called `population` which contains these values. Plot the time series object, ensuring that the scale on the vertical axis covers the interval from 0 to 1. Do the values stabilize at a constant value?

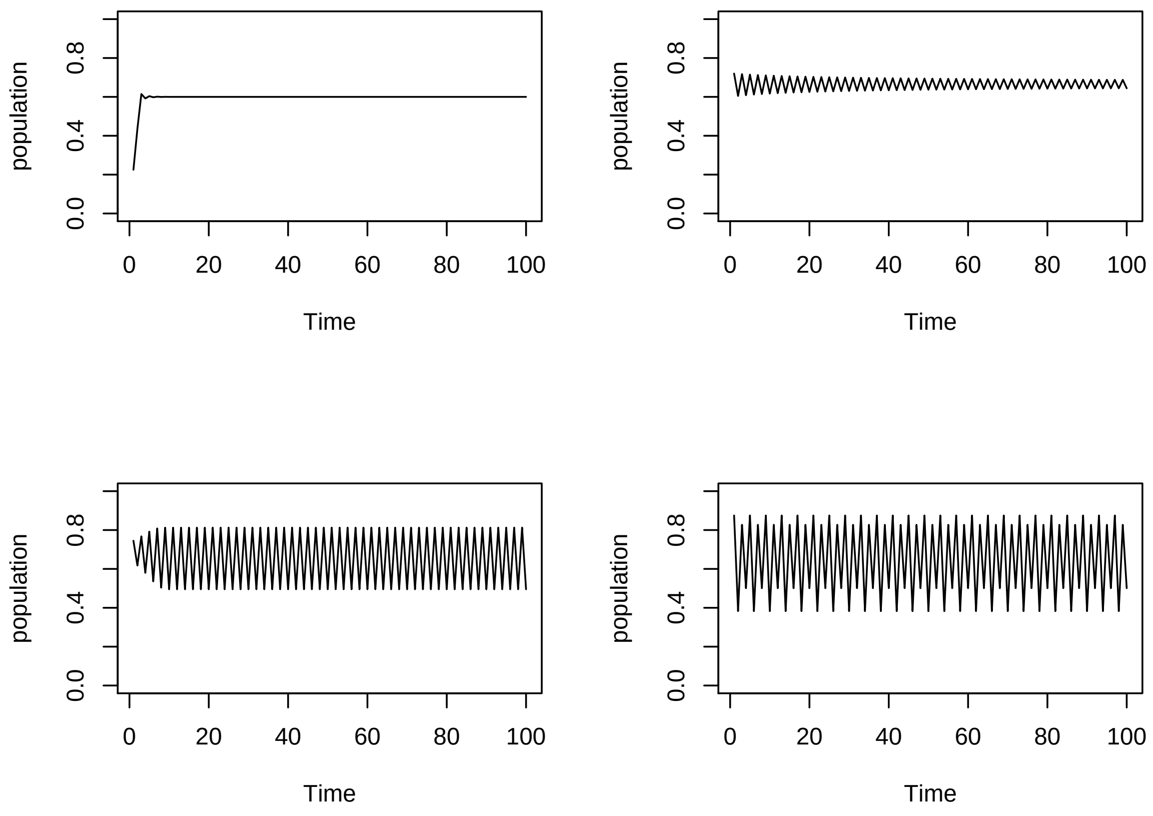
```
n <- 100; R <- 1.5; x <- 0.1
population <- numeric(n)
for (i in 1:n) {
  x <- R*x*(1-x)
  population[i] <- x
}
population <- ts(population)
plot(population, ylim=c(0, 1))
```



3. Repeat the preceding question with $R = 2.5$, $R = 3$, $R = 3.25$, and $R = 3.5$.

```
n <- 100; Rvalues <- c(2.5,  3, 3.25, 3.5); x <- 0.1
par(mfrow=c(2,2))
population <- numeric(n)
for (R in Rvalues) {
```

```
for (i in 1:n) {
  x <- R*x*(1-x)
  population[i] <- x
}
population <- ts(population)
plot(population, ylim=c(0, 1))
}
```

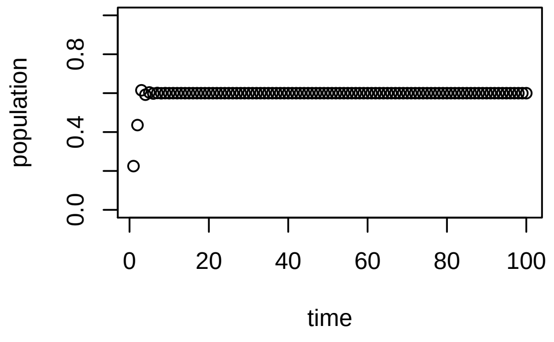


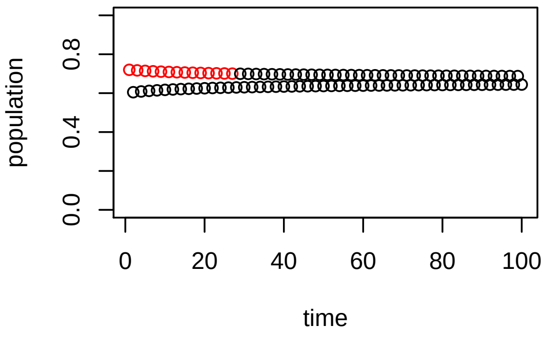
4. In this exercise, we will use the `if()` function inside our for loop to help distinguish the points in the above plots in order to get another view of the patterns. This time, we will plot the successive points and not join with lines but instead use different colours depending on whether the values are above or below 0.7.

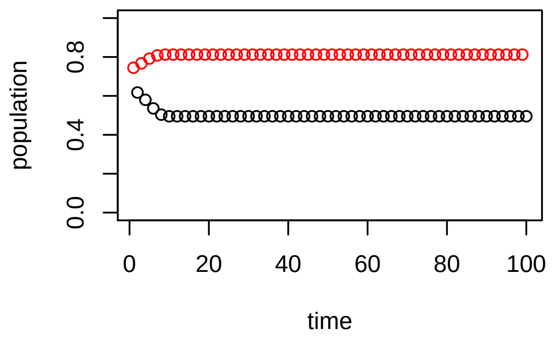
   The code below collects all indices where `x` is greater than 0.7 into a vector called `IH` and all other indices into a vector called `IL`. Then we create a new vector called `color` which has the same length as `population` but where the color codes (1, for black and 2, for red) are assigned to the `IL` and `IH` elements, respectively. We use the `color` vector in our call to `plot` so that we can clearly see the points that are above and below 0.7.

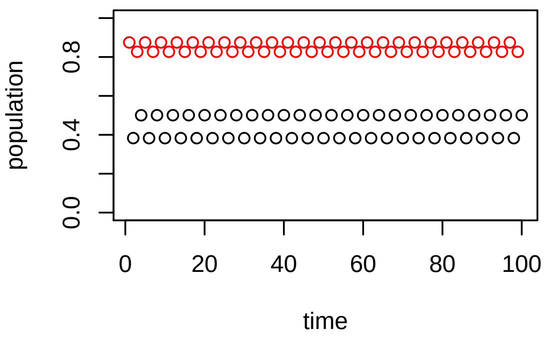
```
n <- 100; Rvalues <- c(2.5,  3, 3.25, 3.5); x <- 0.1
par(mfrow=c(2,2))
```

```
population <- numeric(n)
for (R in Rvalues) {
IH <- NULL
IL <- NULL
for (i in 1:n) {
  x <- R*x*(1-x)
  population[i] <- x
  if (x >0.7){
      IH <- c(IH, i)} else{
      IL <- c(IL,i)
      }
}

color<-population
color[IH]<-2
color[IL]<-1
time <- 1:n
plot(population~time, ylim=c(0, 1), col=color)
}
```









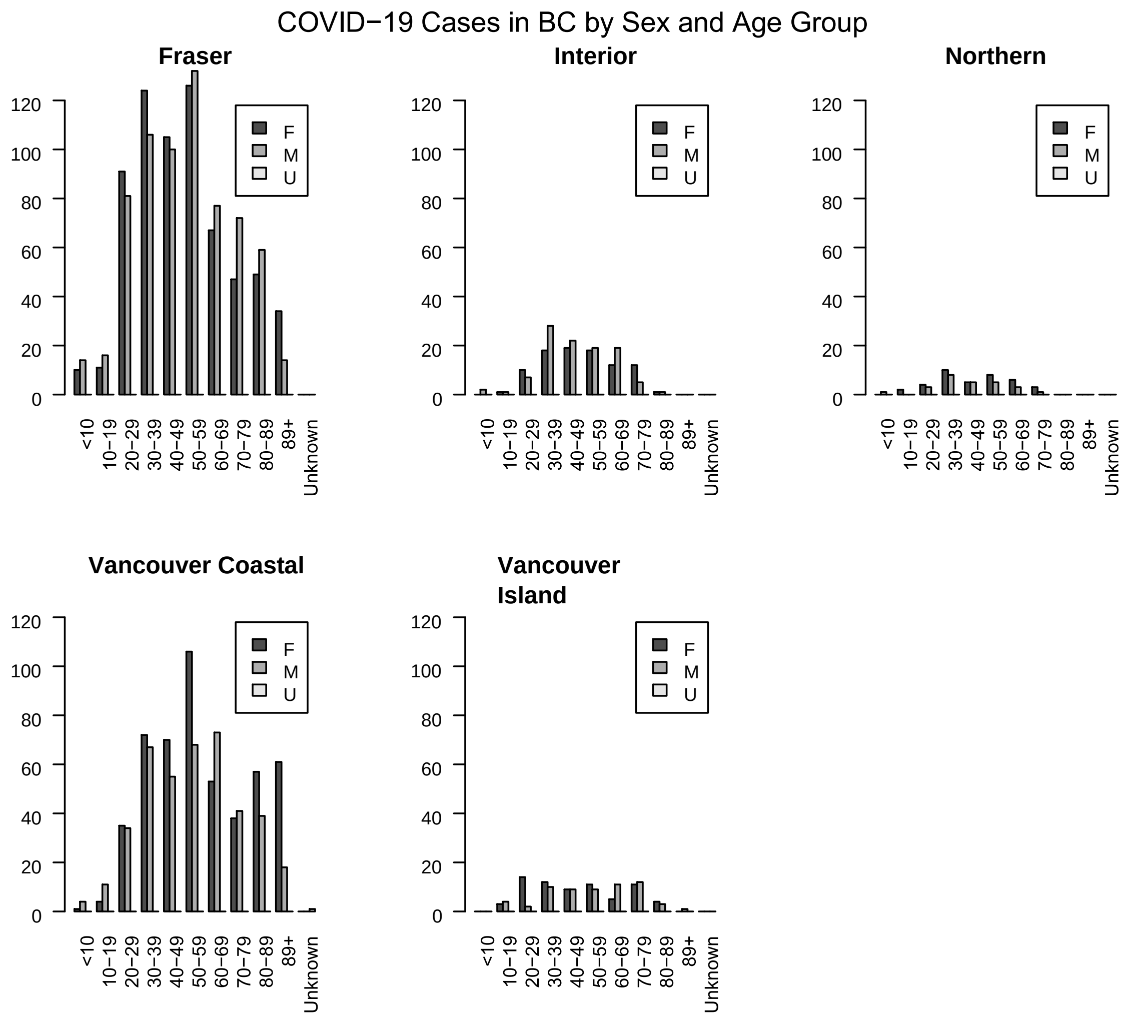
```
color

##   [1] 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2
##  [36] 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
##  [71] 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
```

*The population stabilizes when $R = 2.5$ but not for the other values here. The population seems to oscillate between two values when $R = 3$ and among more values as $R$ increases.*

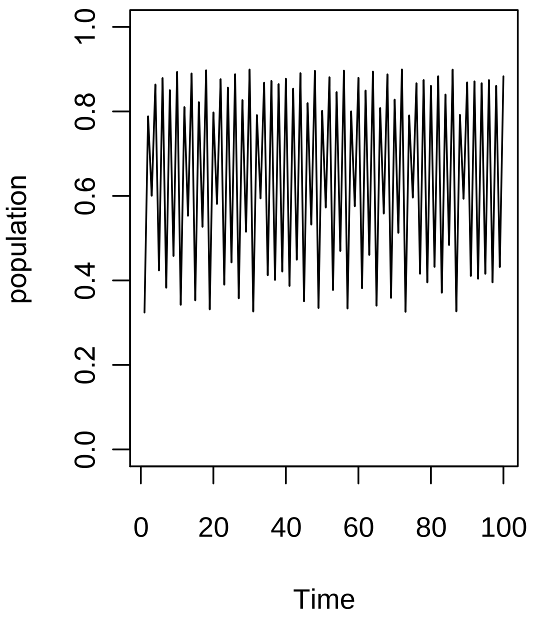
## Exercises for Submission

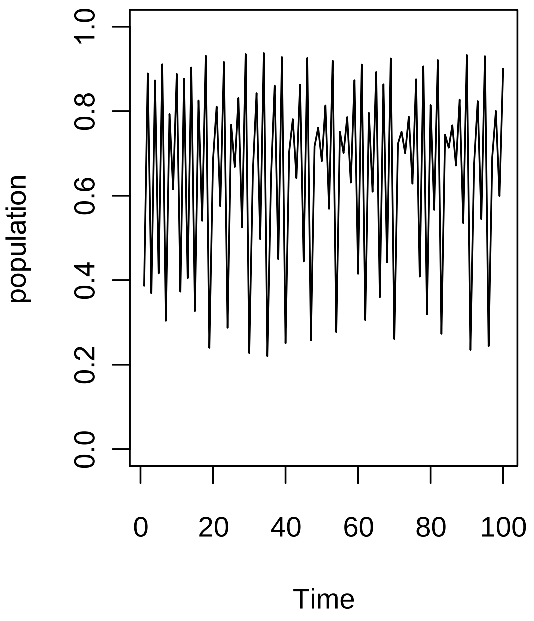
In each question below, write out (or type) the required lines of R code, if needed, together with the answer to the question.

1. Modify the code from the demonstration to create the set of bar plots below. Again, use a for loop to run through the health areas.



2. Repeat question 2 from the demonstration with $R = 3.6$ and $R = 3.7$. What do you observe about these populations?





*There is definitely no stabilizing in either of these cases. In fact, this is an example of what is called 'chaos'.*