LAB 6

DATA 101 – MAKING PREDICTIONS USING DATA

IF YOU WANT TO FOLLOW ALONG

- If you want to follow along for the teaching part of this lab, you can grab the data on canvas. You can use the code below to import it into R.

```
covidBC <- read.table("covidBC.csv",header=TRUE, sep=",")
```

This data was collected from BCCDC:

http://www.bccdc.ca/health-info/diseases-conditions/covid-19/data

ASSIGNMENT 3 FEEDBACK

- Decimal number accuracy

DECIMAL NUMBER ACCURACY

- Two programs that subtract very large, nearly identical numbers will result in the largest amount of error.

Program A:

```
x <- 1000000
y <- 999999
A <- x^4 - y^4
```

Program B:

```
x <- 1000000
y <- 999999
B <- (x^2+y^2)*(x+y)*(x-y)
```

DECIMAL NUMBER ACCURACY

- Two programs that subtract very large, nearly identical numbers will result in the largest amount of error.

Program A:

```
x <- 1000000
y <- 999999
A <- x^4 - y^4
```

Program B:

```
x <- 1000000
y <- 999999
B <- (x^2+y^2)*(x+y)*(x-y)
```

- In A, both operands are large enough that they cannot be perfectly stored in the computer, leading to potential error. In B, the numbers are far smaller, and even though there are more calculations, those calculations are a lot smaller.

DECIMAL NUMBER ACCURACY

- Two programs that subtract very large, nearly identical numbers will result in the largest amount of error.

Program A:

```
x <- 1000000
y <- 999999
A <- x^4 - y^4
```

Program B:

```
x <- 1000000
y <- 999999
B <- (x^2+y^2)*(x+y)*(x-y)
```

- In A, both operands are large enough that they cannot be perfectly stored in the computer, leading to potential error. In B, the numbers are far smaller, and even though there are mor calculations, those calculations are a lot smaller.
- Only at the very end does a large calculation happen, and even then, it's smaller than A. This delays the point where the error happens till the end of the calculation.

DECIMAL NUMBER ACCURACY

- In question 5, it was the same situation. Two very large, very similar numbers being subtracted.

Suppose $x = 100$ and $y = 99$. Find the **true** value of (2 points)

$$x^{16}\left[\frac{(x^8 - y^8)}{(196059601)(19801)(199)} - 1\right].$$

DECIMAL NUMBER ACCURACY

- In question 5, it was the same situation. Two very large, very similar numbers being subtracted.

Suppose $x = 100$ and $y = 99$. Find the **true** value of (2 points)

$$x^{16}\left[\frac{(x^8 - y^8)}{(196059601)(19801)(199)} - 1\right].$$

- Both the top and bottom are equal to the same number, but the numerator has too much error for the division to equal 1.
- Doing a binary expansion fixes the issue and allows for the equation to equal 0.

ASSIGNMENT 3 FEEDBACK

- Decimal number accuracy
- Barplots & dotcharts

BARPLOTS & DOTCHARTS

- Basic barplots want either vectors or a matrix of one row. You can provide labels using colnames()
- Basic dotcharts want either vectors or a matrix of one column. You can provide labels using rownames()

BARPLOTS

- Basic barplots want either vectors or a matrix of one row. You can provide labels using colnames()

```
income <- matrix(c(93.07, 66.79, 70.16, 67.22, 75.24, 72.96), nrow = 1)
colnames(income)
       <- c("Ottawa", "Toronto", "London", "Windsor", "Sudbury", "ThunerBay")
barplot(income,
       ylab = "Income (In 1000s)",
       ylim = c(0,100),
       main = "2009 Family Income in Six Cities")
```

BARPLOTS

- Basic barplots want either vectors or a matrix of one row. You can provide labels using colnames()

```
income <- matrix(c(93.07, 66.79, 70.16, 67.22, 75.24, 72.96), nrow = 1)
colnames(income)
       <- c("Ottawa", "Toronto", "London", "Windsor", "Sudbury", "ThunerBay")
barplot(income,
       ylab = "Income (In 1000s)",
       ylim = c(0,100),
       main = "2009 Family Income in Six Cities")

income <- c(93.07, 66.79, 70.16, 67.22, 75.24, 72.96)
names(income)
       <- c("Ottawa", "Toronto", "London", "Windsor", "Sudbury", "ThunderBay")
barplot(income, main = "2009 Family Income in Six Cities",
    xlab = "Cities",
    ylab = "Income (in $1000s)",
    ylim = c(0, 100))
```

DOTCHARTS

- Basic dotcharts want either vectors or a matrix of one column. You can provide labels using rownames()

```
income <- matrix(c(93.07, 66.79, 70.16, 67.22, 75.24, 72.96), nrow = 6)
colnames(income) <- ("Cities in Ontario")
rownames(income)
       <- c("Ottawa", "Toronto", "London", "Windsor", "Sudbury", "Thunder Bay")
dotchart(income, xlab = "Income (In 1000s)", xlim = c(0,100),
       main = "2009 Family Income in Six Cities")
```

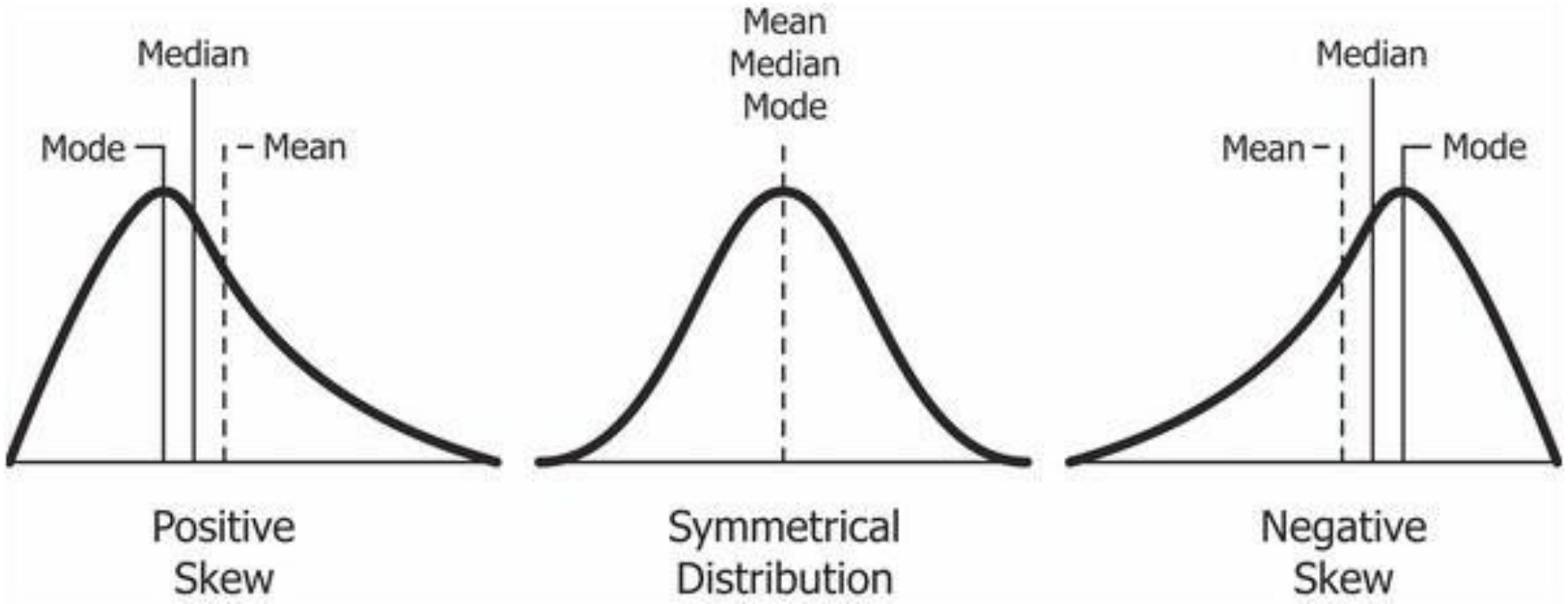
DOTCHARTS

- Basic dotcharts want either vectors or a matrix of one column. You can provide labels using rownames()

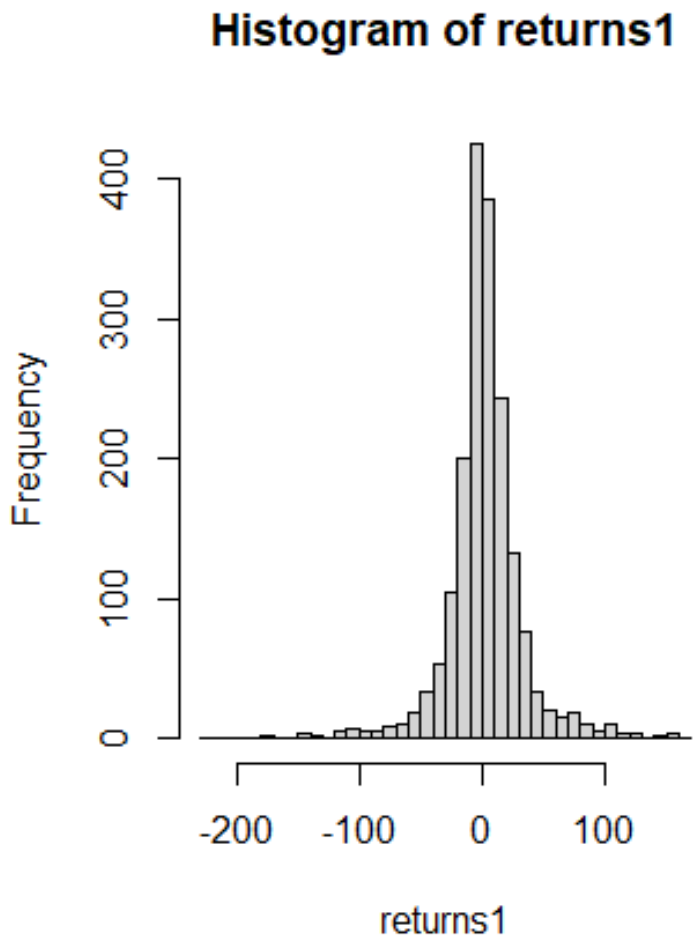
```
income <- matrix(c(93.07, 66.79, 70.16, 67.22, 75.24, 72.96), nrow = 6)
colnames(income) <- ("Cities in Ontario")
rownames(income)
      <- c("Ottawa", "Toronto", "London", "Windsor", "Sudbury", "Thunder Bay")
dotchart(income, xlab = "Income (In 1000s)", xlim = c(0,100),
      main = "2009 Family Income in Six Cities")

income <- c(93.07, 66.79, 70.16, 67.22, 75.24, 72.96)
names(income)
      <- c("Ottawa", "Toronto", "London", "Windsor", "Sudbury", "ThunderBay")
dotchart(income, xlab = "Income (In 1000s)", xlim = c(0,100),
      main = "2009 Family Income in Six Cities")
```

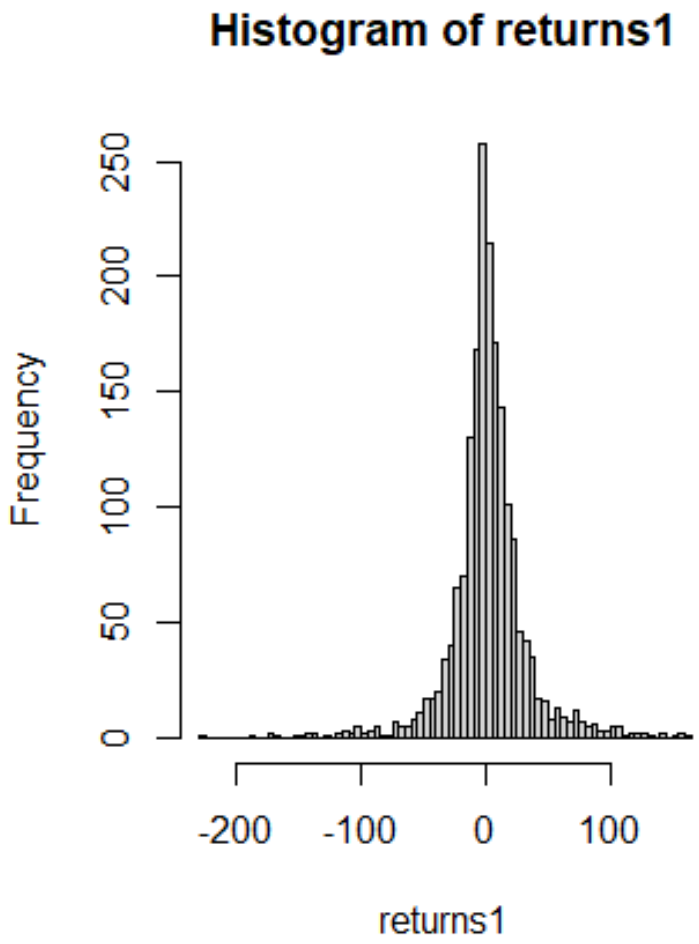
NOTE: If using vectors, both barplots and dotcharts use the same data preparation code.

GRAPH SKEWS


Median
Mode
Mean
Positive Skew
Mean
Median
Mode
Symmetrical Distribution
Median
Mean
Mode
Negative Skew

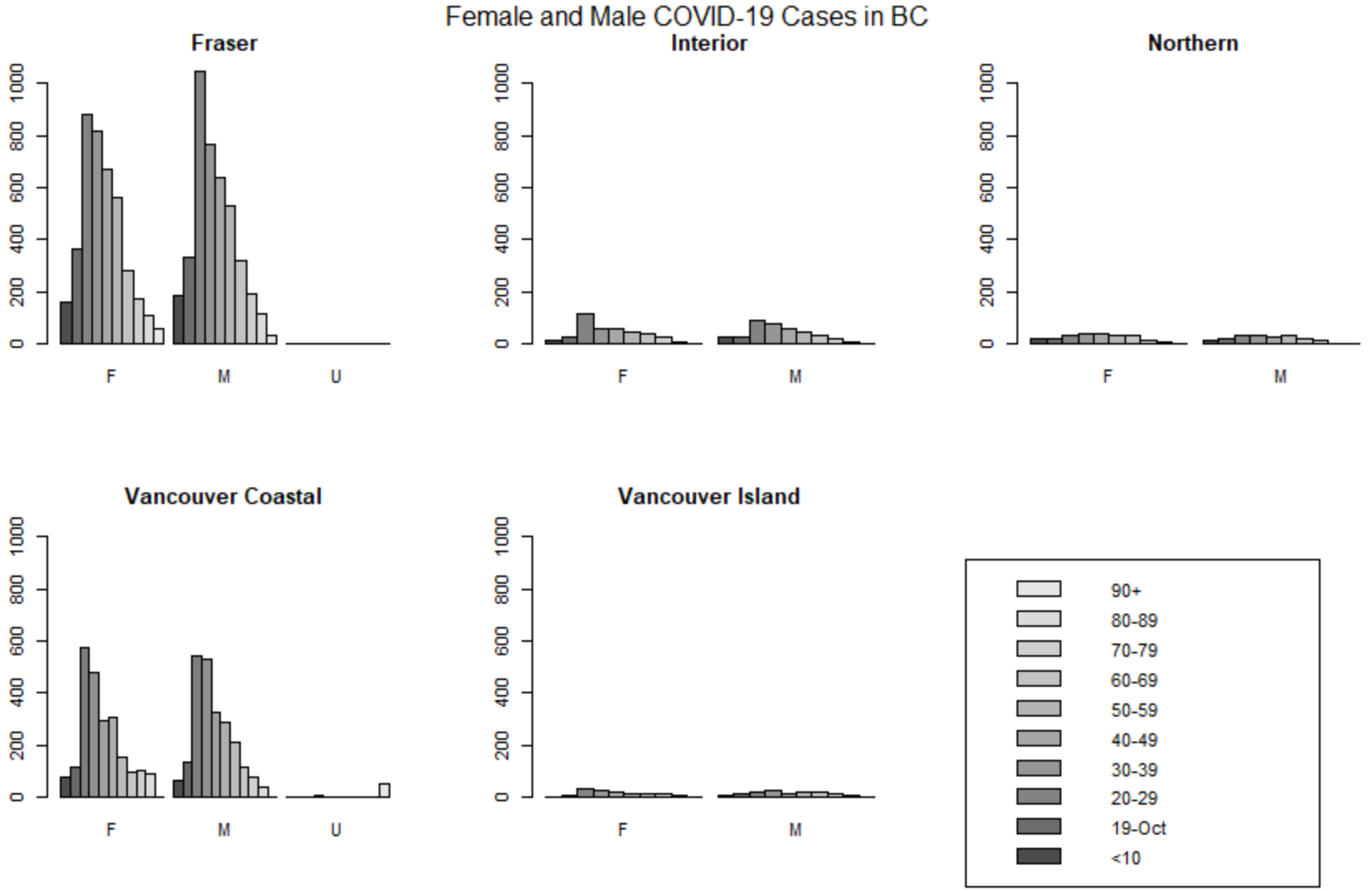
GRAPH SKEWS


Histogram of returns1
Frequency
0
100
200
300
400
-200
-100
0
100
returns1


Histogram of returns1
Frequency
0
50
100
150
200
250
-200
-100
0
100
returns1

LAB 6 -LOOPS

GOAL


Female and Male COVID-19 Cases in BC
Fraser
Interior
Northern
Vancouver Coastal
Vancouver Island
0
200
400
600
800
1000
F
M
U
90+
80-89
70-79
60-69
50-59
40-49
30-39
20-29
19-Oct
<10

DATA

The data frame contains:

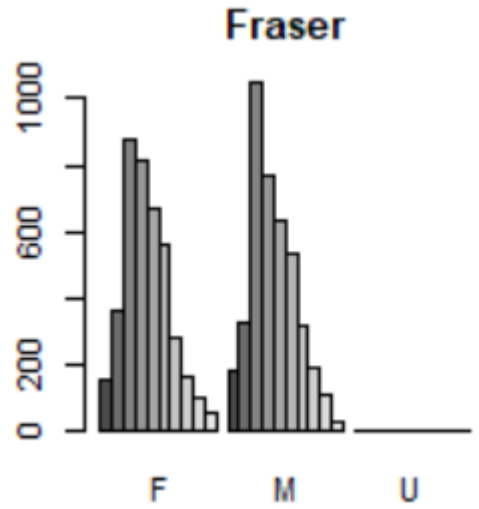
- The reported date;
- The health authority (regions inside BC);
- The sex of the patient;
- The age group of the patient;
- And the method of reporting, either lab diagnosed or epidemically linked (outbreaks).

	Reported_Date	HA	Sex	Age_Group	Classification_Reported
1	2020-02-20	Fraser	F	30-39	Lab-diagnosed
2	2020-02-21	Fraser	M	40-49	Lab-diagnosed
3	2020-03-03	Fraser	M	50-59	Lab-diagnosed
4	2020-03-05	Fraser	M	19-Oct	Lab-diagnosed
5	2020-03-05	Fraser	M	50-59	Lab-diagnosed
6	2020-03-05	Fraser	F	50-59	Lab-diagnosed
7	2020-03-06	Fraser	F	60-69	Lab-diagnosed
8	2020-03-06	Fraser	M	60-69	Lab-diagnosed
9	2020-03-06	Fraser	F	50-59	Lab-diagnosed
10	2020-03-06	Fraser	M	60-69	Lab-diagnosed

DATA PREPARATION

To make the graphs, we'll need to the following:

- A vector of health authorities (for the loop to make each graph);
- The data divided up for each health authority;
- The number of health authorities;
- Plot multiple plots in the same area;

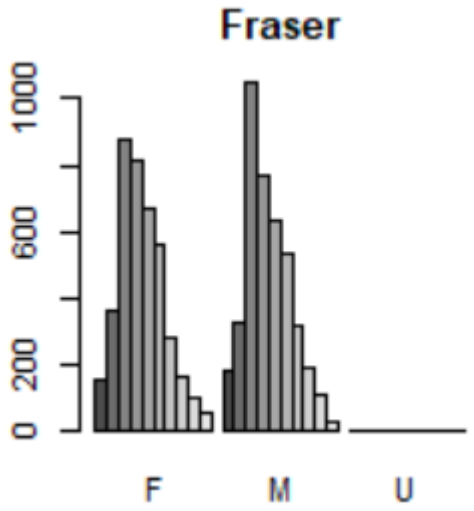


Just a note, the data we're using is more up to date than when the assignment was written, so our dataset is significantly larger.

DATA PREPARATION

To make the graphs, we'll need to the following:

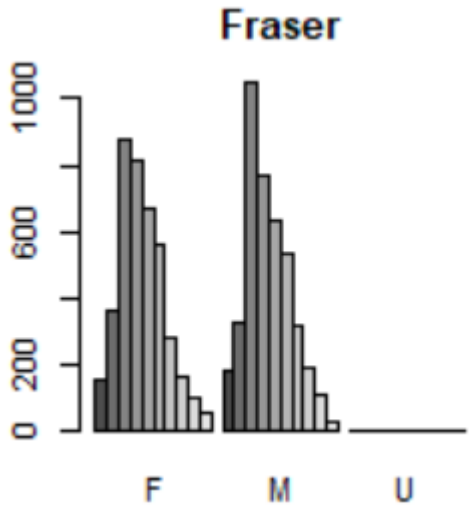
- A vector of health authorities;
  - `areas <- levels(factor(covidBC$HA))`



DATA PREPARATION

To make the graphs, we'll need to the following:

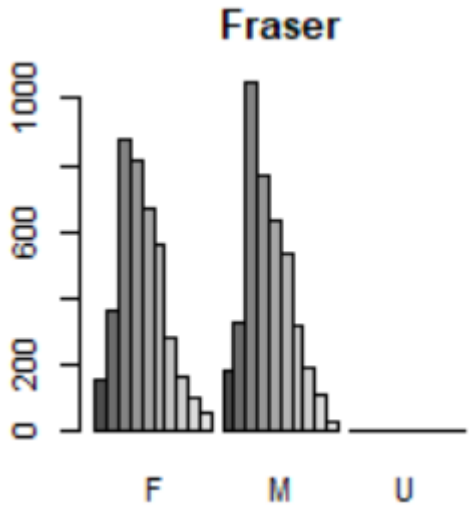
- A vector of health authorities;
  - `areas <- levels(factor(covidBC$HA))`
- The data divided up for each health authority;
  - `covidArea <- split(covidBC, covidBC$HA)`



DATA PREPARATION

To make the graphs, we'll need to the following:

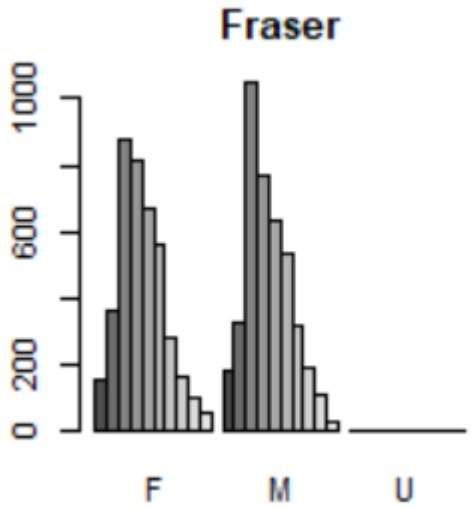
- A vector of health authorities;
  - `areas <- levels(factor(covidBC$HA))`
- The data divided up for each health authority;
  - `covidArea <- split(covidBC, covidBC$HA)`
- The number of health authorities;
  - `n <- length(areas)`
- Plot multiple plots in the same area;
  - `par(mfrow=c(2, 3))`



DATA PREPARATION

To make the graphs, we'll need to the following:

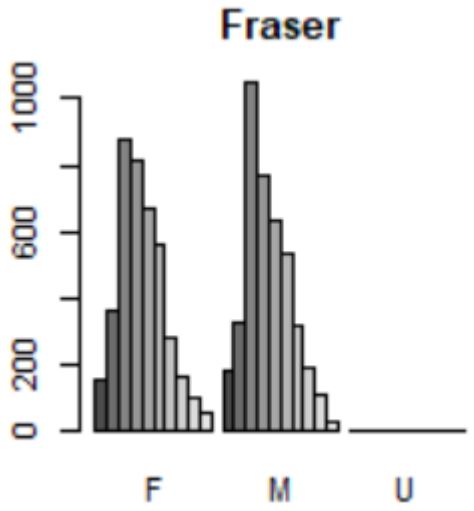
- A vector of health authorities;
  - `areas <- levels(factor(covidBC$HA))`
- The data divided up for each health authority;
  - `covidArea <- split(covidBC, covidBC$HA)`
- The number of health authorities;
  - `n <- length(areas)`



DATA PREPARATION

To make the graphs, we'll need to the following:

- A vector of health authorities;
  - `areas <- levels(factor(covidBC$HA))`
- The data divided up for each health authority;
  - `covidArea <- split(covidBC, covidBC$HA)`
- The number of health authorities;
  - `n <- length(areas)`
- Plot multiple plots in the same area;
  - `par(mfrow=c(2, 3))`



FOR LOOP

To make the graphs, we'll need to the following:

```
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, 1000))
}
```

FOR LOOP

To make the graphs, we'll need to the following:

```
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, 1000))
}
cov_table <- with(
      covidArea[[1]],
      table(Age_Group, Sex))
```

Age_Group	Sex F	Sex M	Sex U
<10	156	186	0
19-Oct	363	330	0
20-29	882	1045	1
30-39	815	768	0
40-49	669	636	1
50-59	559	533	0
60-69	283	319	1
70-79	170	192	0
80-89	107	115	0
90+	59	29	0

FOR LOOP

To make the graphs, we'll need to the following:

```
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, 1000))
}
```

```
cov_table <- with(
      covidArea[[1]],
      table(Age_Group, Sex))
```

Age_Group	Sex F	Sex M	Sex U
<10	156	186	0
19-Oct	363	330	0
20-29	882	1045	1
30-39	815	768	0
40-49	669	636	1
50-59	559	533	0
60-69	283	319	1
70-79	170	192	0
80-89	107	115	0
90+	59	29	0

```
cov_table <- with(
      covidArea[[2]],
      table(Age_Group, Sex))
```

Age_Group	Sex F	Sex M
<10	9	22
19-Oct	24	25
20-29	117	86
30-39	55	74
40-49	55	58
50-59	44	42
60-69	37	34
70-79	25	17
80-89	7	8
90+	1	1

LEGEND

To make the legend, we basically make an empty bar plot. No axis, not bars, no column names.

Remove column names

- `colnames(cov_table) <- NULL`

LEGEND

To make the legend, we basically make an empty bar plot. No axis, not bars, no column names.

Remove column names

- `colnames(cov_table) <- NULL`

Make all data empty

- `cov_table <- cov_table*NA`

LEGEND

To make the legend, we basically make an empty bar plot. No axis, not bars, no column names.

Remove column names

- `colnames(cov_table) <- NULL`

Make all data empty

- `cov_table <- cov_table*NA`

Plot with a legend but not axis

- `barplot(cov_table, legend=TRUE, ylim=c(0, 1000), axes=FALSE)`

LEGEND

To make the legend, we basically make an empty bar plot. No axis, not bars, no column names.

Remove column names

- `colnames(cov_table) <- NULL`

Make all data empty

- `cov_table <- cov_table*NA`

Plot with a legend but not axis

- `barplot(cov_table, legend=TRUE, ylim=c(0, 1000), axes=FALSE)`

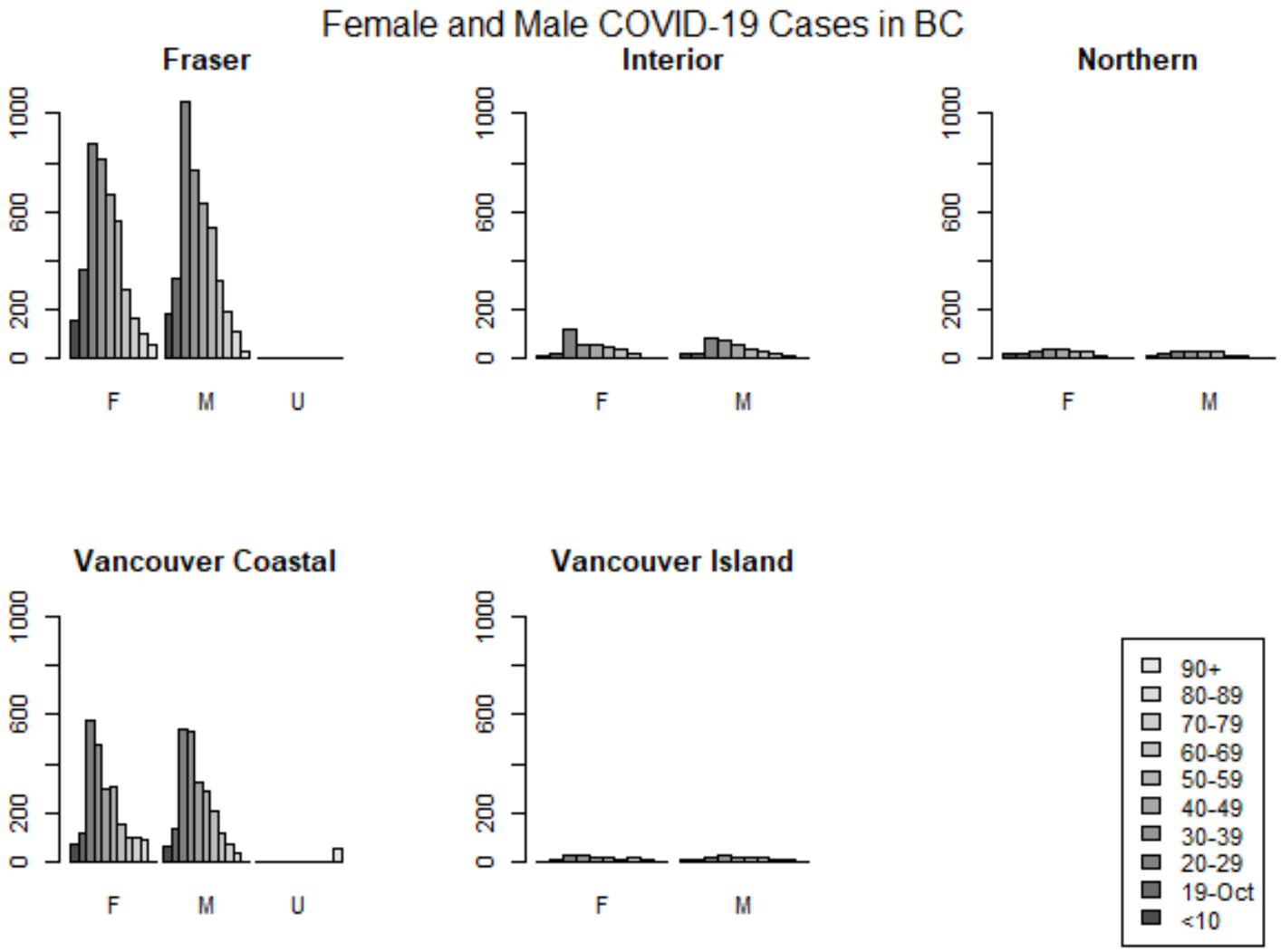
Add titles

- `mtext(side=3, line = -1.5, "Female and Male COVID-19 Cases in BC", outer = TRUE)`

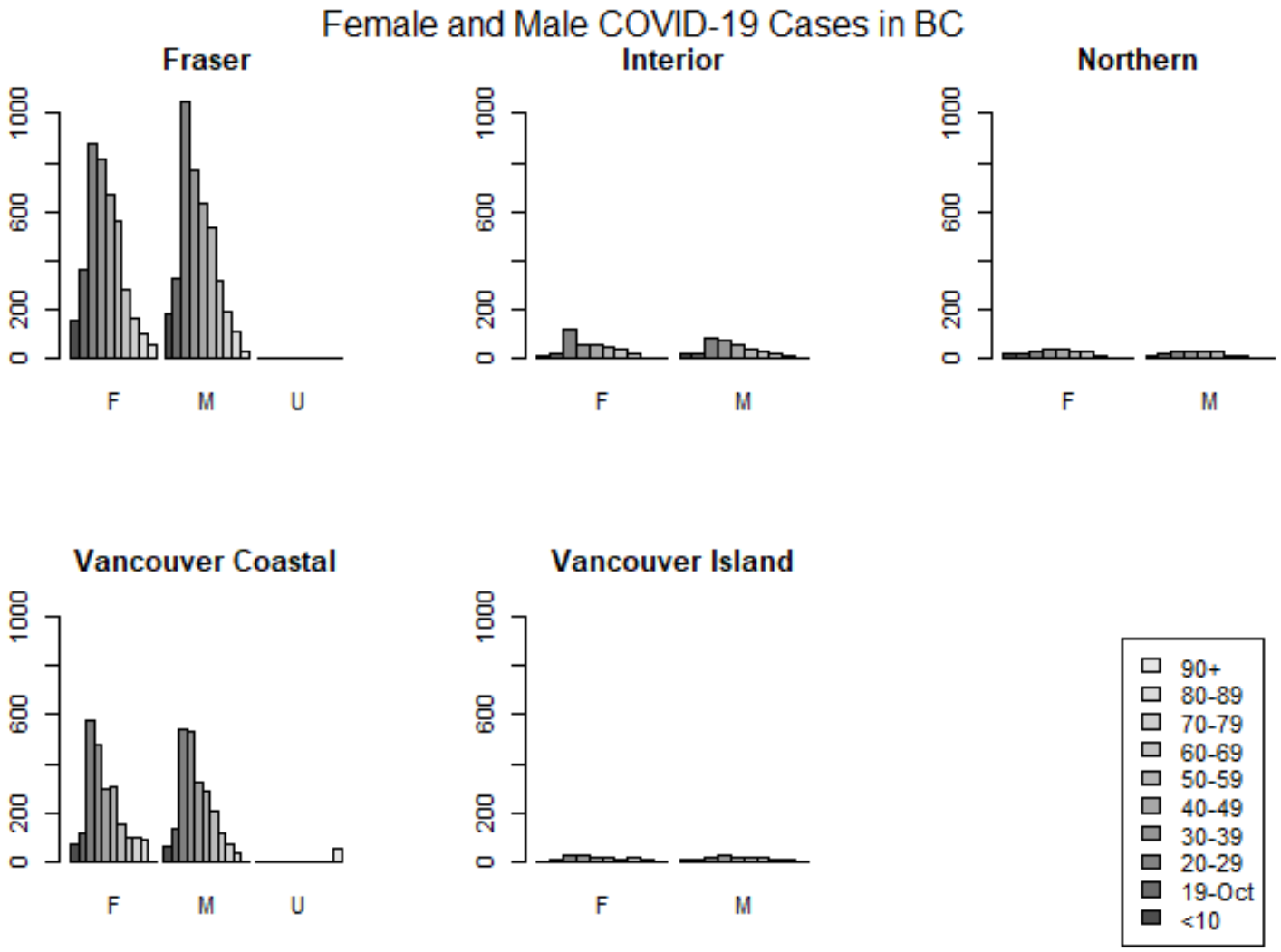
ALL TOGETHER

```
covidBC <- read.table("covidBC.csv",header=TRUE, sep=",")


areas <- levels(factor(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, 1000))
}


colnames(cov_table) <- NULL
cov_table <- cov_table*NA
barplot(cov_table, legend=TRUE, ylim=c(0, 1000), axes=FALSE)
mtext(side=3, line = -1.5, "Female and Male COVID-19 Cases in BC", outer = TRUE)
```

ALL TOGETHER



ALL TOGETHER



LAB 6 –LOGISTIC EQUATIONS

THE LOGISTIC EQUATION

The logistic equation is:

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

Where:

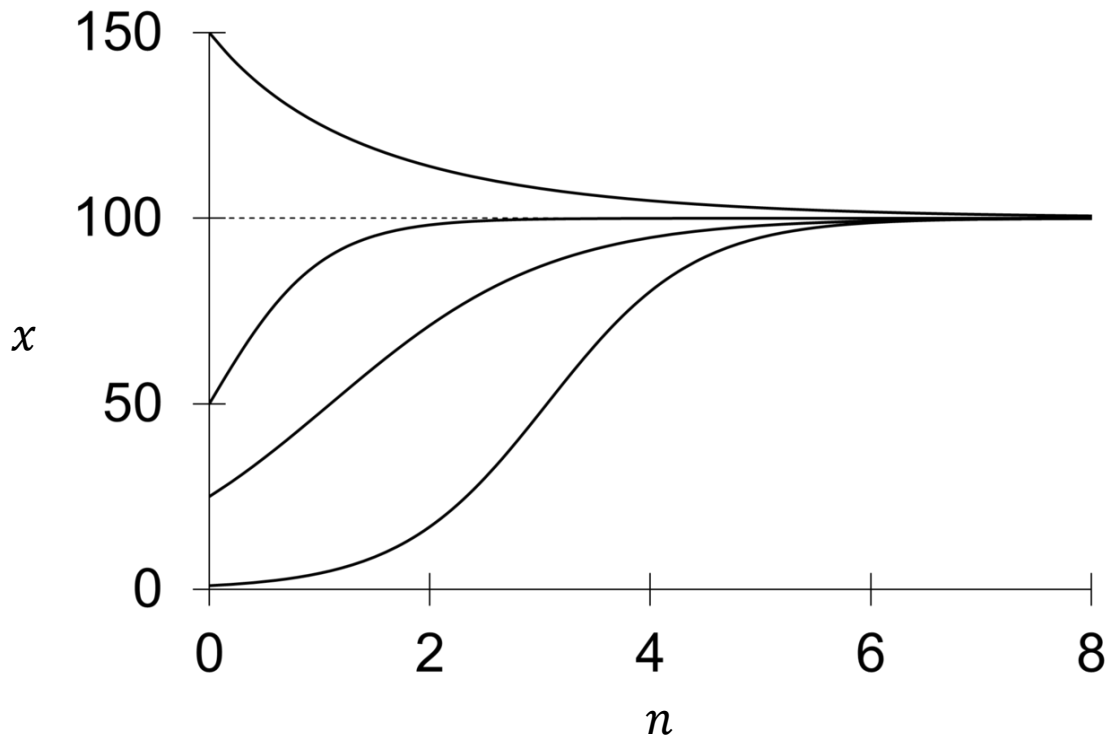
- R is a constant representing the carrying capacity (The maximum population a species in a given environment) for a population.
- x_n is the proportion of the carrying capacity occupied by the population at time n.
- x_{n+1} is the proportion at the next time n + 1.

Note that if $x_n < 1$, the population will grow, and if it's $x_n > 1$, it's population will decrease.

THE LOGISTIC EQUATION

The logistic equation is:

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



LOGISTIC EQU. IN R

Suppose $R = 1.5$, $x_0 = 0.1$. Calculate $x_1, x_2, x_3 \ldots x_{100}$

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

LOGISTIC EQU. IN R

Suppose $R = 1.5$, $x_0 = 0.1$. Calculate $x_1, x_2, x_3 \ldots x_{100}$

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

```
n <- 100
R <- 1.5
x <- 0.1

population <- numeric(n)
```

LOGISTIC EQU. IN R

Suppose $R = 1.5$, $x_0 = 0.1$. Calculate $x_1, x_2, x_3 \ldots x_{100}$

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

```
n <- 100
R <- 1.5
x <- 0.1

population <- numeric(n)


for (i in 1:n) {
   x <- R*x*(1-x) # logistic equation
   population[i] <- x # save the value Xn+1
}
```

LOGISTIC EQU. IN R

Suppose $R = 1.5$, $x_0 = 0.1$. Calculate $x_1, x_2, x_3 \ldots x_{100}$

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

```
n <- 100
R <- 1.5
x <- 0.1
population <- numeric(n)


for (i in 1:n) {
   x <- R*x*(1-x) # logistic equation
   population[i] <- x # save the value Xn+1
}


population <- ts(population) # turn it into a time-series object
plot(population, ylim=c(0, 1)) # plot
```

LOGISTIC EQU. IN R

Suppose $R = 1.5$, $x_0 = 0.1$. Calculate $x_1, x_2, x_3 \ldots x_{100}$

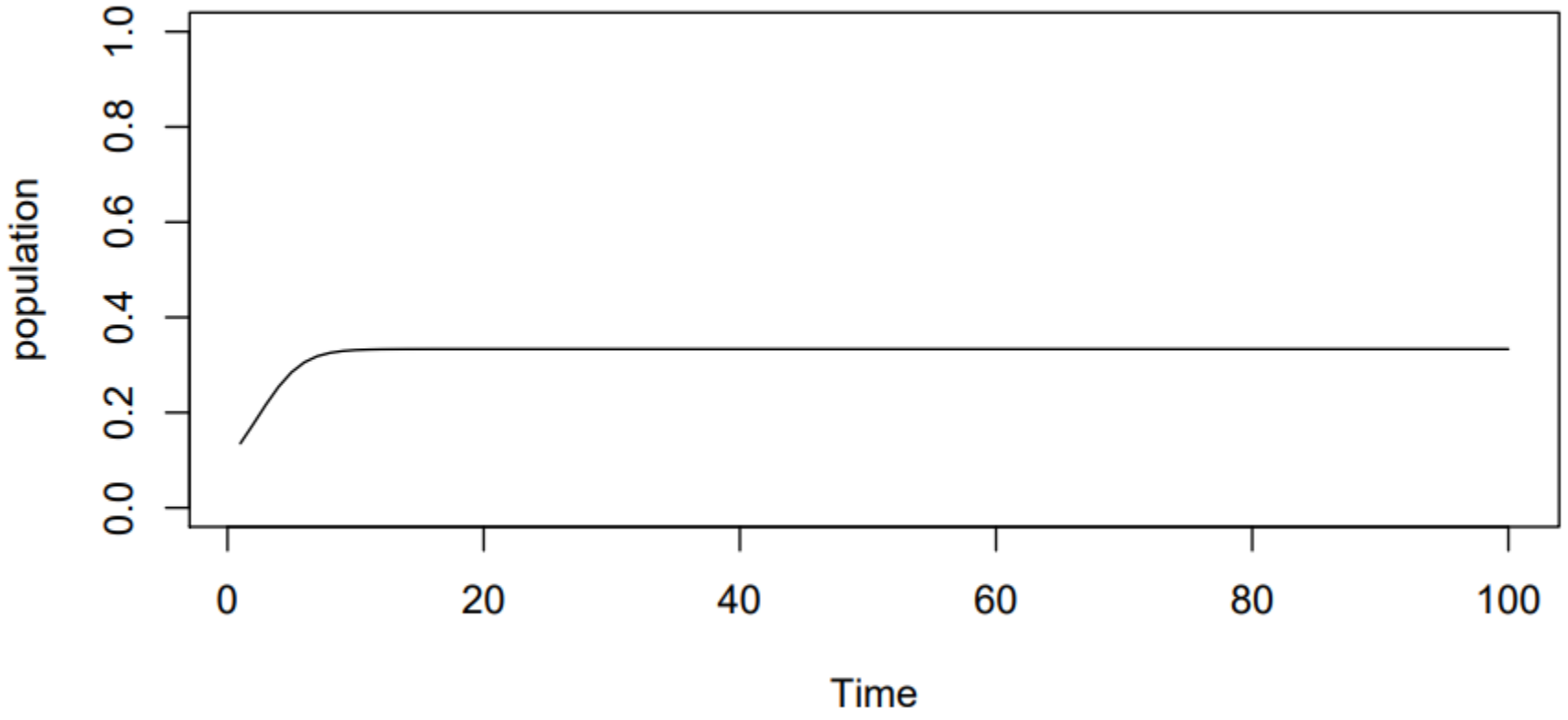
$$x_{n+1} = Rx_n(1 - x_n)$$

```
n <- 100
R <- 1.5
x <- 0.1
population <- numeric(n)


for (i in 1:n) {
   x <- R*x*(1-x) # logistic equation
   population[i] <- x # save the value Xn+1
}


population <- ts(population) # turn it into a time-series object
plot(population, ylim=c(0, 1)) # plot
```

LOGISTIC EQU. IN R

Suppose $R = 1.5$, $x_0 = 0.1$. Calculate $x_1, x_2, x_3 \ldots x_{100}$

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



LOGISTIC EQU. IN R

Do the same thing but for $R = 2.5, 3, 3.25, 3.5$, $x_0 = 0.1$.

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

LOGISTIC EQU. IN R

Do the same thing but for $R = 2.5, 3, 3.25, 3.5$, $x_0 = 0.1$.

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

```
n <- 100; x <- 0.1; population <- numeric(n);

Rvalues <- c(2.5, 3, 3.25, 3.5)
par(mfrow=c(2,2))
```

LOGISTIC EQU. IN R

Do the same thing but for $R = 2.5, 3, 3.25, 3.5$, $x_0 = 0.1$.

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

```
n <- 100; x <- 0.1; population <- numeric(n);

Rvalues <- c(2.5, 3, 3.25, 3.5)
par(mfrow=c(2,2))


for (R in Rvalues) {


}
```

LOGISTIC EQU. IN R

Do the same thing but for $R = 2.5, 3, 3.25, 3.5$, $x_0 = 0.1$.

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

```
n <- 100; x <- 0.1; population <- numeric(n);

Rvalues <- c(2.5, 3, 3.25, 3.5)
par(mfrow=c(2,2))

for (R in Rvalues) {
   for (i in 1:n) {
      x <- R*x*(1-x) # logistic equation
      population[i] <- x # save the value Xn+1
   }
}
```

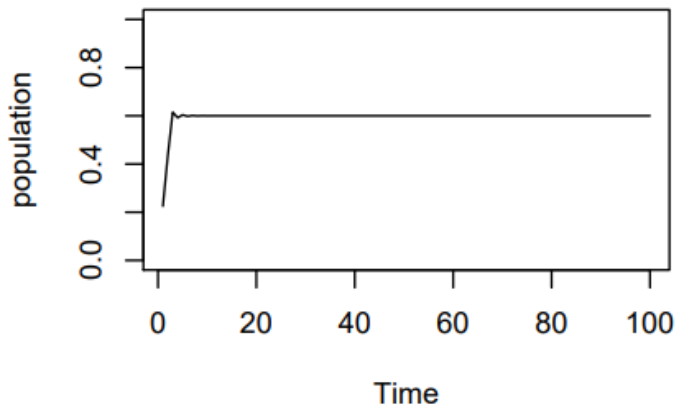
LOGISTIC EQU. IN R

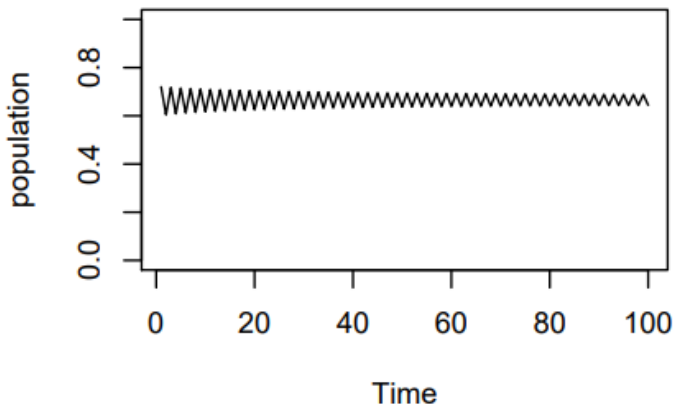
Do the same thing but for $R = 2.5, 3, 3.25, 3.5$, $x_0 = 0.1$.

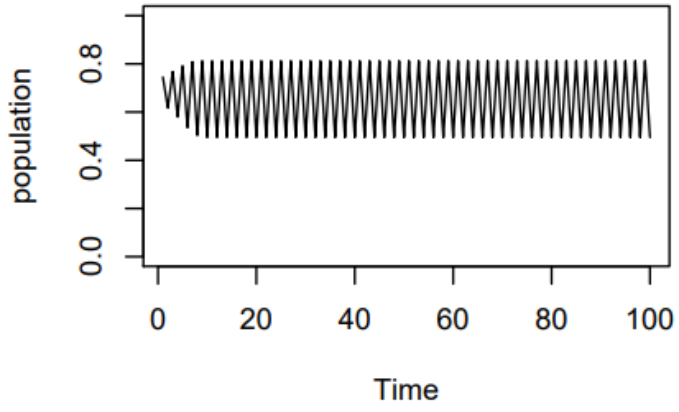
$$x_{n+1} = Rx_n(1 - x_n)$$

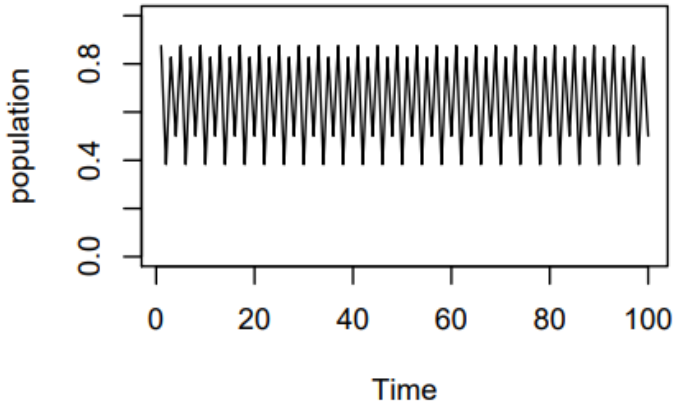
```
n <- 100; x <- 0.1; population <- numeric(n);

Rvalues <- c(2.5, 3, 3.25, 3.5)
par(mfrow=c(2,2))

for (R in Rvalues) {
   for (i in 1:n) {
      x <- R*x*(1-x) # logistic equation
      population[i] <- x # save the value Xn+1
   }

   population <- ts(population) # turn it into a time-series object
   plot(population, ylim=c(0, 1)) # plot
}
```

LOGISTIC EQU. IN R





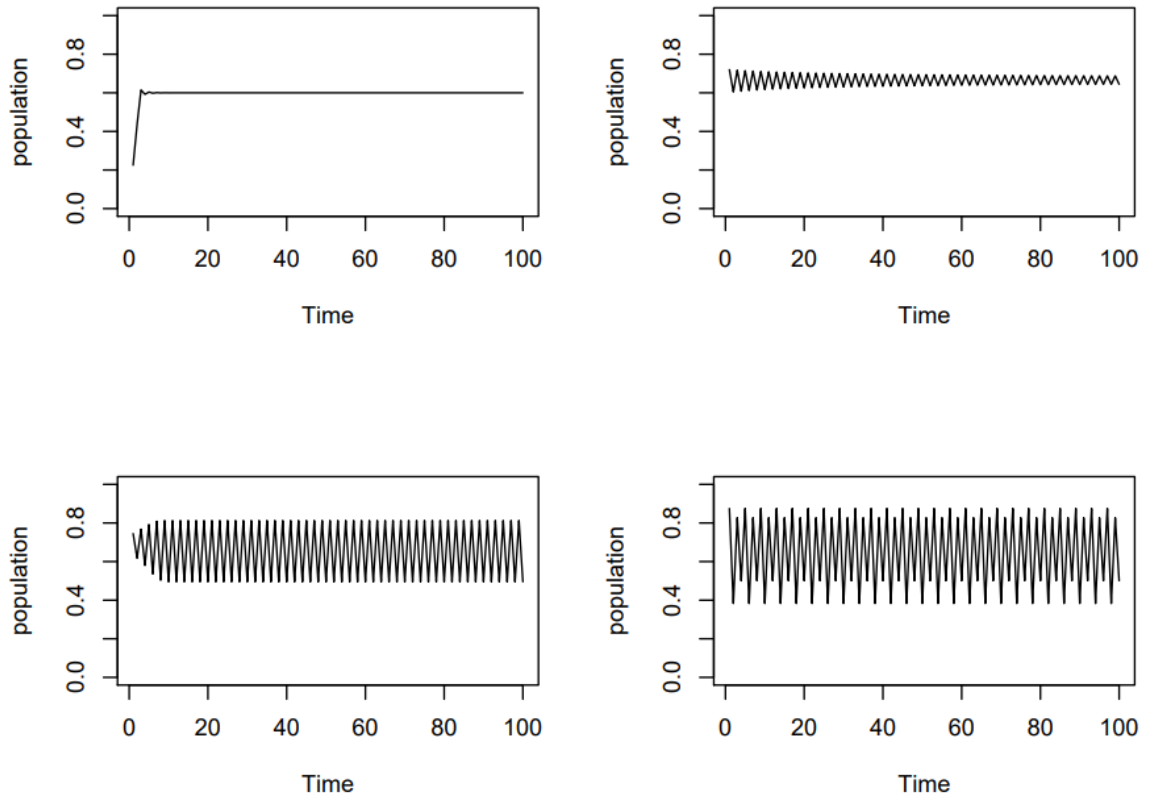




LAB 6 -IF STATMENTS

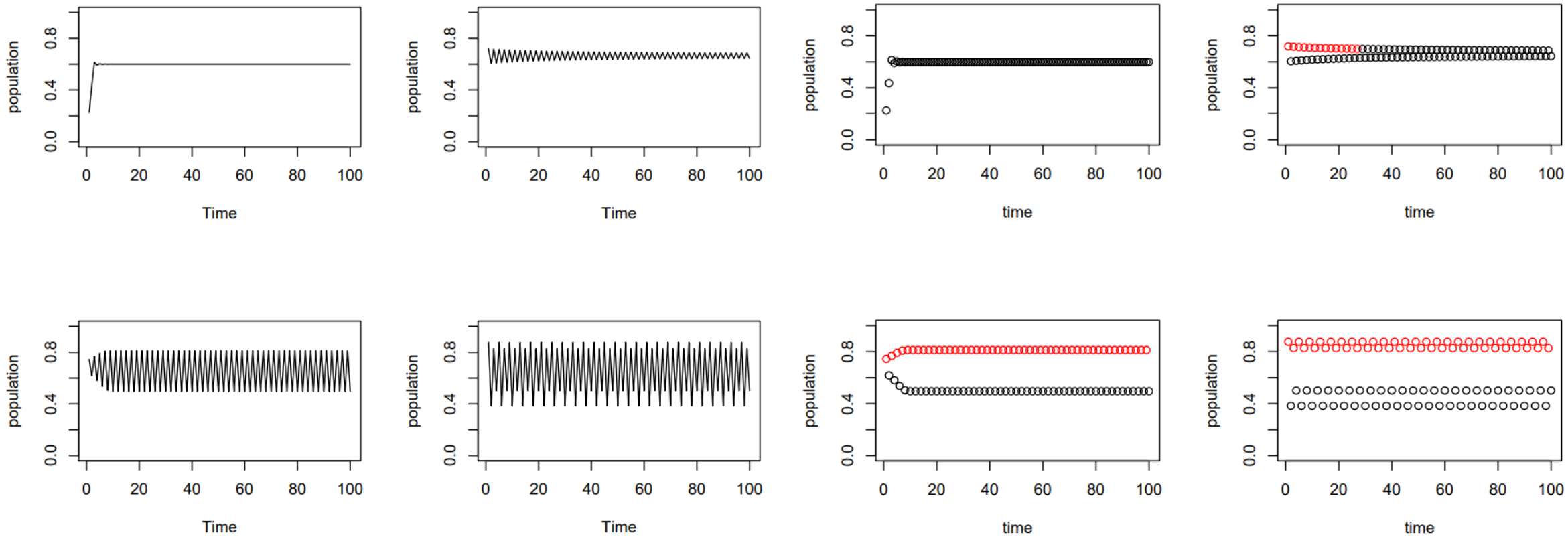
LOGISTIC EQU. IN R

We want to do the same thing as the previous example but remove the lines in between the points and color them red if they are above 0.7 and black if they are below.



LOGISTIC EQU. IN R

We want to do the same thing as the previous example but remove the lines in between the points and color them red if they are above 0.7 and black if they are below.



LOGISTIC EQU. IN R

```
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
      }
}
```

LOGISTIC EQU. IN R

```
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) }
      }
}
```

LOGISTIC EQU. IN R

```
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)
}
```