**The University of British Columbia**

Irving K. Barber School of Sciences

*DATA 101*

Lab Assignment 8 2020W1

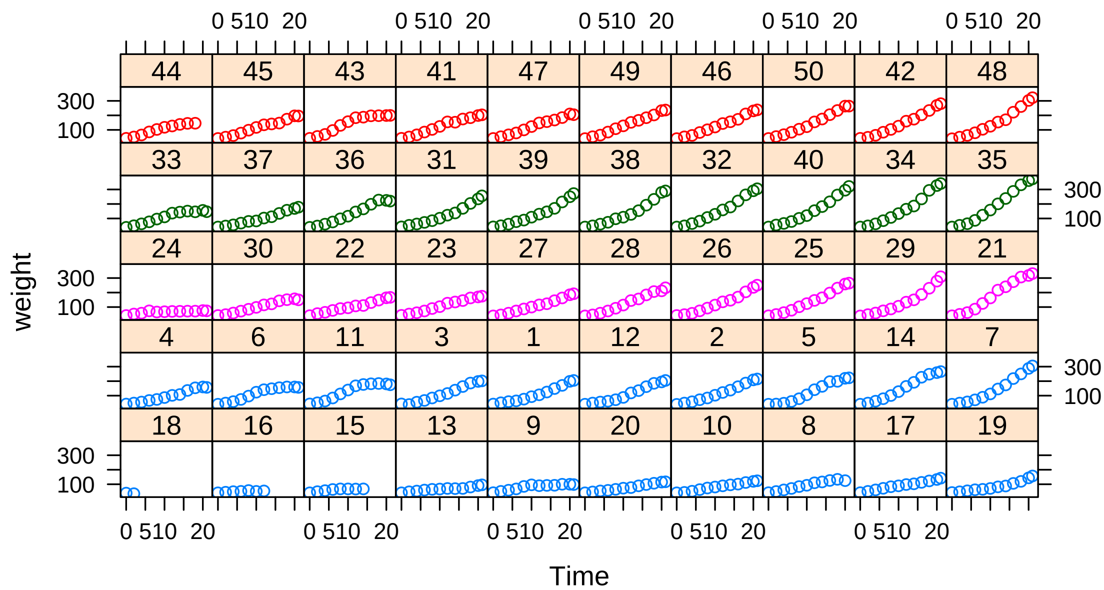
*Demonstration.* The TA will go through a set of examples with you.

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

1. Consider the data in `ChickWeight`. Construct conditioning scatter plots that relate weight to time, for each chick, and use `groups=Diet` in order to distinguish the different diets by different colours on the graphs.

**Solution:**

```
library(lattice)
xyplot(weight ~ Time|Chick, groups=Diet, data = ChickWeight)
```




2. Write a function which will evaluate polynomials of the form

$$P(x) = a_n x^{n-1} + a_{n-1} x^{n-2} + \cdots + a_2 x^2 + a_1$$

Your function should take $x$ and the vector of polynomial coefficients as arguments and it should return the value of the evaluated polynomial. Name this function as you want.

**Solution:** You may have your own way to do this. If you have different way to do this, please share with us.

- Method one:

```
directpolysinglex <- function(x, a){
         # a is a coefficients vector starting with c1
          n <- length(a)
          power <- (n-1):0
          xpower <- x^power
          p<-sum(a*xpower)
          return(p)
}
x<- 1
a <- c(2,6,1)
directpolysinglex(x,a)

## [1] 9

x<- 2
a <- c(2,6,1)
directpolysinglex(x,a)

## [1] 21

x<- 10000
a <- rep(10000, 1000)
directpolysinglex(x,a)

## [1] Inf
```

Now we check this function with input x that is a vector

```
x<- c(1,2)
a <- c(2,6,1)
directpolysinglex(x,a)

## Warning in x^power:  longer object length is not a multiple of shorter
object length

## [1] 15
```

The function mess up the solution.

```
directpolysinglex <- function(x, a){
         # a is a coefficients vector starting with c1
          if (length(x)!=1) {
                   print("Warning: The first argument has to be a single nu
          } else {
                   n <- length(a)
                  power <- (n-1):0
                  xpower <- x^power
                  p<-sum(a*xpower)
                 return(p)
         }
}
x<- c(1,2)
a <- c(2,6,1)
directpolysinglex(x,a)

## [1] "Warning: The first argument has to be a single number"
```

- method 2:

```
directpolyvectorx <- function(x, a){
        # a is a coefficients vector starting with c1
          n <- length(a)
          p<- 0  # initial set for P(x)
          for (i in 1:n) {
          p <- p+a[i]*x^(i-1)
          }
          return(p)
}
x<- 1
a <- c(2,6,1)
directpolyvectorx(x,a)

## [1] 9

x<- 2
a <- c(2,6,1)
directpolyvectorx(x,a)

## [1] 18

x<- 10000
a <- rep(10000, 1000)
directpolyvectorx(x,a)

## [1] Inf

#this one to check input x is vector
x<- c(1,2)
a <- c(2,6,1)
directpolyvectorx(x,a)

## [1]  9 18
```

3. Refer to the previous question. For moderate to large values of n, evaluation of a polynomial at $x$ can be done more efficiently using `Horner's rule`

   (a) Set the initial $P(x)$ as $P \leftarrow a_n$

   (b) For $i = n-1, n-2, \cdots, 2, 1$, update

$$P = P * x + a_i \tag{1}$$

That is

$$P = a_n x + a_{n-1}, \text{when } i = n - 1$$

$$P = (a_n x + a_{n-1})x + a_{n-2}, \text{when } i = n - 2$$

( this time P on the right of equation 3b is equal to $a_n x + a_{n-1}$) Then we have

$$P = a_n x^2 + a_{n-1} x + a_{n-2}, \text{when } i = n - 2$$

and so on

$$P = a_n x^{n-1} + a_{n-1} x^{n-2} + \cdots + a_2 x^2 + a_1, \text{ when } i = 1$$

(c) Return P. ( This is the computed value of $P(x)$.)

Write an R function which takes arguments $x$ and a vector of polynomial coefficients and which returns the value of the polynomial evaluated at x. Call the resulting function `hornerpoly()`. Ensure that your function returns an appropriate vector of values when x is a vector.

**Solution:**

```
hornerpoly <- function(x,a){
# a is a coefficients vector starting with c1
         n<-length(a)
          p<-a[n]
          for (i in (n-1):1){
          p<- p*x + a[i]
          }
          return(p)
}
x<- 1
a <- c(2,6,1)
hornerpoly(x,a)

## [1] 9

x<- 2
a <- c(2,6,1)
hornerpoly(x,a)

## [1] 18

x<- 10000
a <- rep(10000, 1000)
hornerpoly(x,a)

## [1] Inf

#this one to check input x is vector
x<- c(1,2)
a <- c(2,6,1)
hornerpoly(x,a)

## [1]  9 18
```