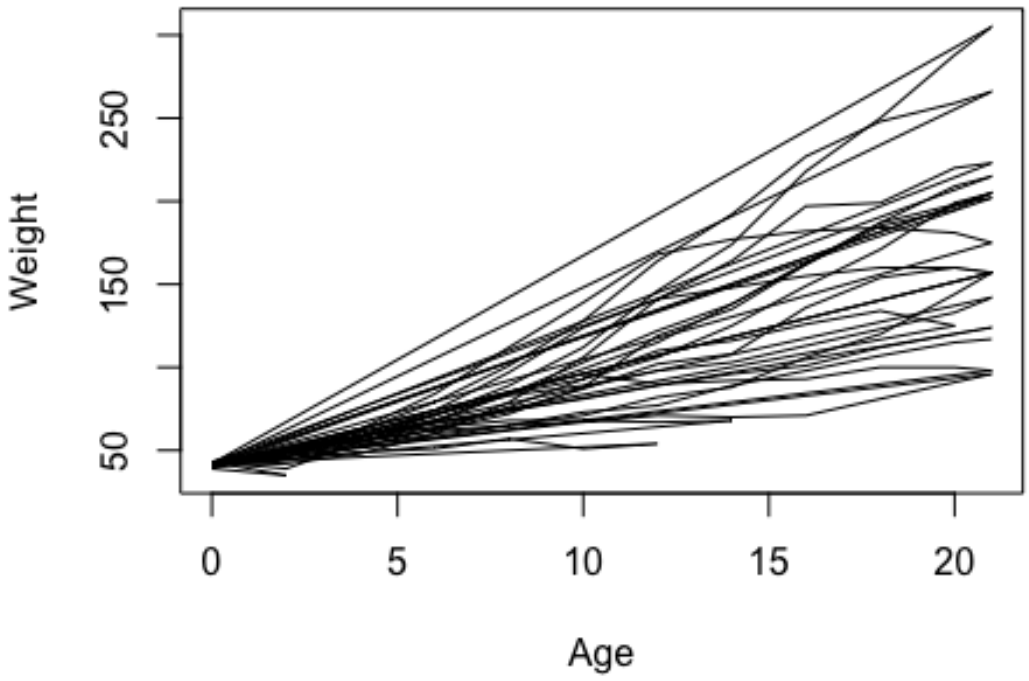
Assignment 2

1. Type 'ChickWeight' in R, then a dataset of 50 chicks' weight data will appear. Each chick was measured multiple times so you can see the growth of chicks. Draw growth curves of chicks with Diet = 1 in one pane (there are 20 chicks with Diet 1, so there will be 20 growth curves in one pane). The horizontal axis is Time (age) and the vertical axis is weight.

```
CW <- ChickWeight[ChickWeight$Diet == 1,]

plot.new()
axis(1, at= c(0:25))
axis(2, at= c(0:350))

plot(CW$Time, CW$weight, type="l", ylab="Weight", xlab = "Age")
```



2. We want to see how fast the Taylor expansion of sin x at x = 0 converges to the true value when x = 2π (all angles are expressed in radian).

(a) What is the exact value of sin 2π? (Answer only. No justification is needed.)

```
#Sin(2pi) = 0
```

(b) Approximate sin2π by a finite sum in (1) for n = 1. Do for n = 2 also.

```
library(pracma)
approxByFiniteSum <- function(nValue)
{
  return(polyval(taylor(f= sin, x0= 0, n= nValue), 2*pi))
}

print(approxByFiniteSum(1))

## [1] 6.283185

print(approxByFiniteSum(3))

## [1] -35.05851
```

(c) Using the 'while' or 'repeat' loop, find the smallest n such that the approximation error of the Taylor polynomial is less than 0.000001.

```
e <- 1
n <- 1
i <- 0
s <- 0
x <- 2*pi
j <- 0
repeat
{
  i <- i + 1
  j <- (-1)^(i+1)*(x^n)/factorial(n)
  n <- n + 2
  s <- s + j
  e <- abs(sin(x)-s)
  if(e < .000001)
  {
    break
  }
}
 print(i)

## [1] 13
```

3. Let x be a numeric vector with n observations ($n \in N$).

(a) Create a function to (i) resample n observations from x with replacement and then (ii) calculate the sample mean, for 100 times. The input is x, and the output is a numeric vector with 100 sample means.

```
sampleMean <- function(X)
{
  n <- length(X)
  D <- matrix(0,ncol= n , nrow= 100)
  for(i in 1:100)
  {
    D[i,] <- sample(X, n, replace= T)
  }
  return(apply(D,1,mean))
}
```

(b) Apply the function in (a) for the following x 8.5, 7.5, 9.6, 13.6, 7.5, 7.7, 6.1, 5.5, 8.9, 8.6

```
SM <- sampleMean(c(8.5, 7.5, 9.6, 13.6, 7.5, 7.7, 6.1, 5.5, 8.9, 8.6))
print(SM)
```

```
##   [1] 9.53 9.24 9.08 8.30 7.99 9.29 8.38 7.99 7.35 8.62 7.95 8.62 7.67 7.63 6.87
##  [16] 8.01 8.27 7.79 8.46 8.18 7.38 8.10 7.77 9.27 8.04 9.56 8.06 8.41 8.56 8.25
##  [31] 7.41 8.01 8.29 9.34 8.92 8.53 8.38 7.55 8.29 8.81 9.25 9.67 7.69 8.54 9.06
##  [46] 8.70 7.85 7.46 8.39 8.41 8.22 8.98 8.87 7.55 8.38 7.33 7.44 7.37 8.16 6.73
##  [61] 8.00 7.38 8.43 7.70 8.19 8.41 7.57 9.20 8.29 9.47 8.57 7.90 8.99 8.02 8.36
##  [76] 7.32 8.35 7.93 8.22 8.69 8.53 8.48 8.07 8.76 7.88 6.83 8.82 9.52 7.98 8.19
##  [91] 8.94 8.04 7.73 8.75 8.38 7.03 8.17 7.33 9.37 8.77
```

(c) What is the estimated standard deviation of the sample mean of x, based on (b)?

```
print(sd(SM))
```

```
## [1] 0.6558657
```

4. By using R, create the first 15 lines of the left 8 columns in the normal distribution table at: https://www.math.arizona.edu/~rsims/ma464/standardnormaltable.pdf. Namely, cre- ate a matrix which has (i) 15 lines and 8 columns, (ii) normal probabilities to the nearest 100,000-th, and (iii) appropriate row and column names (you do not have to create the letter "z" on the upper left of the table). (Use a normal probability function and 'for' loops. No credits are given if you just typed all 120 numbers by yourself!) Note: The expression of the numbers do not have to be exactly the same as the original table. For example, '-3.0' can be '-3'; ".00" can be '0'; '.12345' can be '0.12345'.)

```
D <- numeric(120)
count <- 0

for (i in -40:-26)
{
  for(j in 1:8)
  {
    D[1+count]<-round(pnorm(((i+1)/10)-((j-1)/100)),5)
    count<-count+1
  }
}

SND <- matrix(D,15,8,TRUE)
colnames(SND) <- (0:7)/100
rownames(SND) <- (-39:-25)/10
print(SND)
```

```
##            0    0.01    0.02    0.03    0.04    0.05    0.06    0.07
## -3.9 0.00005 0.00005 0.00004 0.00004 0.00004 0.00004 0.00004 0.00004
## -3.8 0.00007 0.00007 0.00007 0.00006 0.00006 0.00006 0.00006 0.00005
## -3.7 0.00011 0.00010 0.00010 0.00010 0.00009 0.00009 0.00008 0.00008
## -3.6 0.00016 0.00015 0.00015 0.00014 0.00014 0.00013 0.00013 0.00012
## -3.5 0.00023 0.00022 0.00022 0.00021 0.00020 0.00019 0.00019 0.00018
## -3.4 0.00034 0.00032 0.00031 0.00030 0.00029 0.00028 0.00027 0.00026
## -3.3 0.00048 0.00047 0.00045 0.00043 0.00042 0.00040 0.00039 0.00038
## -3.2 0.00069 0.00066 0.00064 0.00062 0.00060 0.00058 0.00056 0.00054
## -3.1 0.00097 0.00094 0.00090 0.00087 0.00084 0.00082 0.00079 0.00076
## -3   0.00135 0.00131 0.00126 0.00122 0.00118 0.00114 0.00111 0.00107
## -2.9 0.00187 0.00181 0.00175 0.00169 0.00164 0.00159 0.00154 0.00149
## -2.8 0.00256 0.00248 0.00240 0.00233 0.00226 0.00219 0.00212 0.00205
## -2.7 0.00347 0.00336 0.00326 0.00317 0.00307 0.00298 0.00289 0.00280
## -2.6 0.00466 0.00453 0.00440 0.00427 0.00415 0.00402 0.00391 0.00379
## -2.5 0.00621 0.00604 0.00587 0.00570 0.00554 0.00539 0.00523 0.00508
```