Computational Statistics lab 1

Shwetha Vandagadde, Suhani Ariga and Hoda Fakharzadeh

11/4/2020

Question 1: Be careful when comparing

```
# ------------------------------------------------------------------------------
# A1
# ------------------------------------------------------------------------------
```

Two code snippets are mentioned in the question. Snippet 1 :

```
x1 = 1/3
x2 = 1/4
if ( x1-x2 == 1/12) {
print ("Subtsraction is correct")
}else {
print ( " Substraction is wrong" )
}
```

```
[1] " Substraction is wrong"
```

Snippet 2 :

```
x1 = 1
x2 = 1/2
if ( x1-x2 == 1/2) {
  print ("Subtsraction is correct")
}else {
  print ( " Substraction is wrong" )
}
```

```
[1] "Subtsraction is correct"
```

Analysis : In case of first snippet , 1/3 and 1/12 results in a number with infinite decimal part, ie to say that these fractions has no finite representation and so R rounds it up. So the resulting value of the subtraction is not exactly equal to each other. So snippet 1 gives out messsage "Substraction is wrong"

But in case of snippet 2, 1/2 has a finite representation and hence snippet 2 gives out correct answer.

To go around the problem with snippet 1 , we can use all.equal since it tests for 'near equity' of two objects. revised code for snippet 1

```r
x1 = 1/3
x2 = 1/4
if (isTRUE(all.equal((x1 - x2),1/12))) {
  print ("Subtsraction is correct")
}else {
  print ( " Substraction is wrong" )
}
```

```
[1] "Subtsraction is correct"
```

Question 2: Derivative

```r
# ------------------------------------------------------------------------------
# A2
# ------------------------------------------------------------------------------
```

Function to calculate derivative :

```r
f = function(x){x}

derivative_func = function(f=f,x){
  e = 10^-15
  f1 = (f(x + e) - f(x)) / e
  return(f1)
}
```

Evaluating function at x = 1 and x = 100000

```r
x1 = derivative_func(f,1)
x1
```

```
[1] 1.110223
```

```r
x100000 = derivative_func(f,100000)
x100000
```

```
[1] 0
```

Results obtained and true values

As f(x) = x , true value is e/e ie 1.

For x = 1 e is not considered to be very small when x = 1, so f(1+e) results in 1.0000000000000011 which is slighlty greater than 1, after this gets substracted with f(1), we get numerator slightly greater than denominator and when this gets divided by e due to underflow in r , we get the answer slightly bigger than 1.

For x = 100000 When x is 100000 , value of e is very small , this results in numerator being 0 due to underflow error. Hence we get derivative for this function for a large x as 0

Question 3: Variance

```
# ----------------------------------------------------------------------------
# A3
# ----------------------------------------------------------------------------
```

1 My variance function

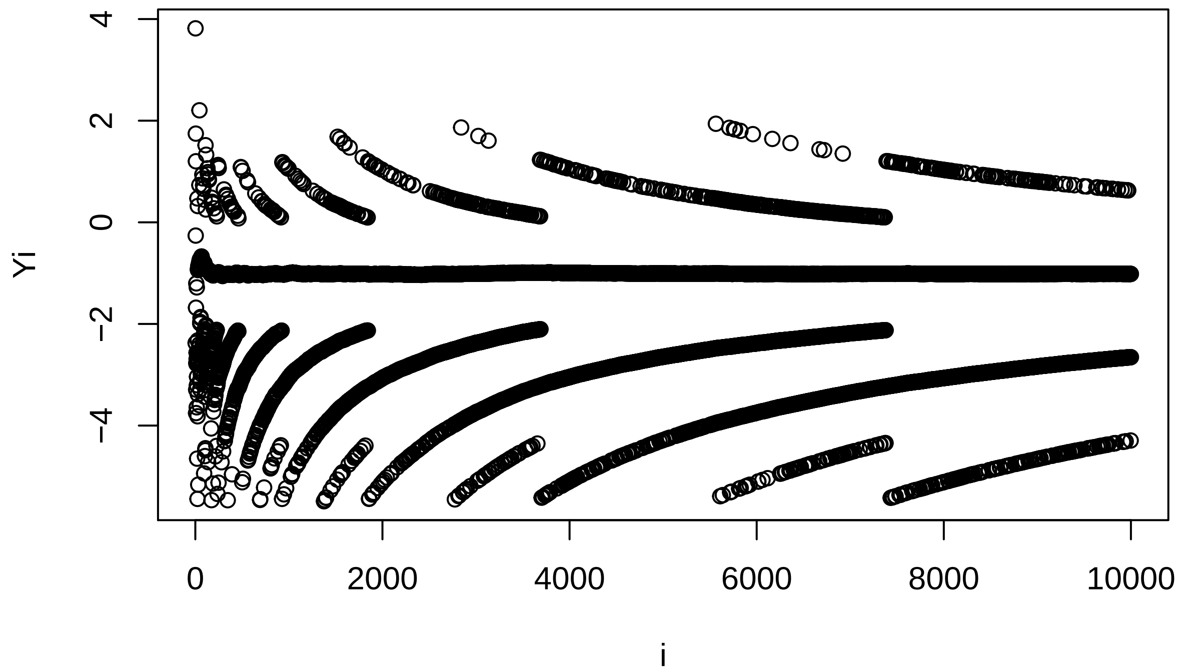
```
myvar = function(x){
  n = length(x)
  var = (1/(1-n)) * (sum(x^2) - ((1/n) * (sum(x))^2))
  return(var)
}
```

Generating x vector

```
x = rnorm(10000, mean = 10^8, sd = 1)
```

Plotting graph to compare the above variance function with var()

```
n = length(x)
i = 1:10000
Yi = 0
for( j in 1:n){
  Yi[j] = myvar(x[1:j]) - var(x[1:j])
}
plot(x = i , y = Yi)
```

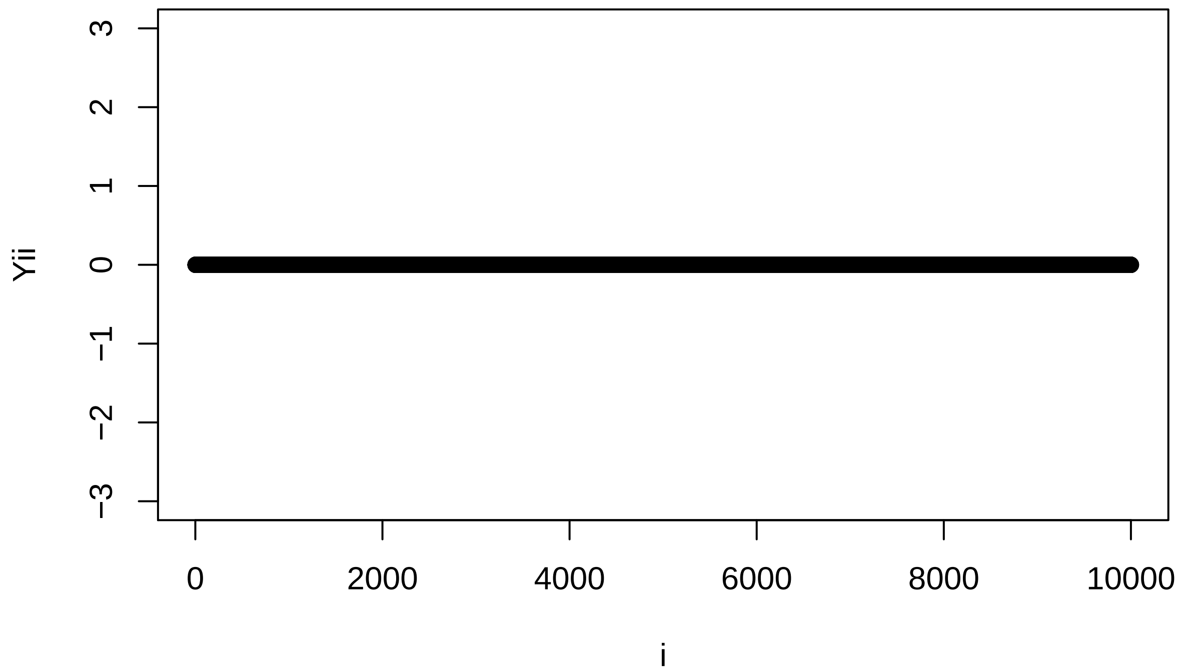


We can see there is difference between our variance function and var(). If there was no difference , we would have observed all the points to be at zero as there would be no difference. As the value of x increases , the summation of x^2 and x used in our formula will have overflow, hence the result is not accurate. This results in the difference of variances.

Better implementation variance estimator

Trying to implement formula of sample variance , var = sum(x - xbar)^2 / n-1 to improve the results.

```
better_var = function(x){
  len = length(x)
  x_bar = mean(x)
  v = (sum((x - x_bar)^2 ))/ (len-1)
  return(v)
}

n = length(x)
i = 1:10000
Yii = 0
j=0
for( j in 1:n){
  Yii[j] = better_var(x[1:j]) - var(x[1:j])
}

plot(x = i, y = Yii, ylim = c(-3,3))
```



This function gives better result as the difference plotted in the graph is concentrated at zero. This is because the x - xbar is done before summation is applied onto it , this tackles the overflow problem better than the function written before.

Question 4: Binomial Coefficient

```
# ---------------------------------------------------------------------------
# A4
# ---------------------------------------------------------------------------
```

A) prod(1:n)/(prod(1:k)∗prod(1:(n−k)))
B) prod((k+1):n)/prod(1:(n−k))

C) prod(((k+1):n)/(1:(n−k)))

1. For the above R expressions below values of n and k does not work properly
When k>n we will get answer -Inf/Inf instead of 0 for all expressions.

```
n = 8
k = 14
prod(1:n)/(prod(1:k)*prod(1:(n-k)))       # A expression
```

```
[1] Inf
```

```r
prod((k+1):n)/prod(1:(n-k))              # B expression
```

```
[1] Inf
```

```r
prod(((k+1):n)/(1:(n-k)))                # C expression
```

```
[1] Inf
```

```r
choose(n,k)
```

```
[1] 0
```

When k=n we will get answer Inf instead of 1 for all expressions.

```r
n = 8
k = 8
prod(1:n)/(prod(1:k)*prod(1:(n-k)))      # A expression
```

```
[1] Inf
```

```r
prod((k+1):n)/prod(1:(n-k))              # B expression
```

```
[1] Inf
```

```r
prod(((k+1):n)/(1:(n-k)))                # C expression
```

```
[1] Inf
```

```r
choose(n,k)
```

```
[1] 1
```

When k<0 we will get answer -Inf/Inf instead of 0 for expression A.

```r
n = 10
k = -12
prod(1:n)/(prod(1:k)*prod(1:(n-k)))      # A expression
```

```
[1] Inf
```

```r
prod((k+1):n)/prod(1:(n-k))              # B expression
```

```
[1] 0
```

```r
prod(((k+1):n)/(1:(n-k)))                # C expression
```

```
[1] 0
```

```r
choose(n,k)
```

```
[1] 0
```

When k=0 we will get answer Inf instead of 1 for expression A.

```r
n = 10
k = 0
prod(1:n)/(prod(1:k)*prod(1:(n-k)))      # A expression
```

```
[1] Inf
```

```r
prod((k+1):n)/prod(1:(n-k))              # B expression
```

```
[1] 1
```

```r
prod(((k+1):n)/(1:(n-k)))                # C expression
```

```
[1] 1
```

```r
choose(n,k)
```

```
[1] 1
```

When n=k=0 we will get answer NaN instead of 1 for all expressions.

```r
n = 0
k = 0
prod(1:n)/(prod(1:k)*prod(1:(n-k)))      # A expression
```

```
[1] NaN
```

```r
prod((k+1):n)/prod(1:(n-k))              # B expression
```

```
[1] NaN
```

```r
prod(((k+1):n)/(1:(n-k)))                # C expression
```

```
[1] NaN
```

```r
choose(n,k)
```

```
[1] 1
```

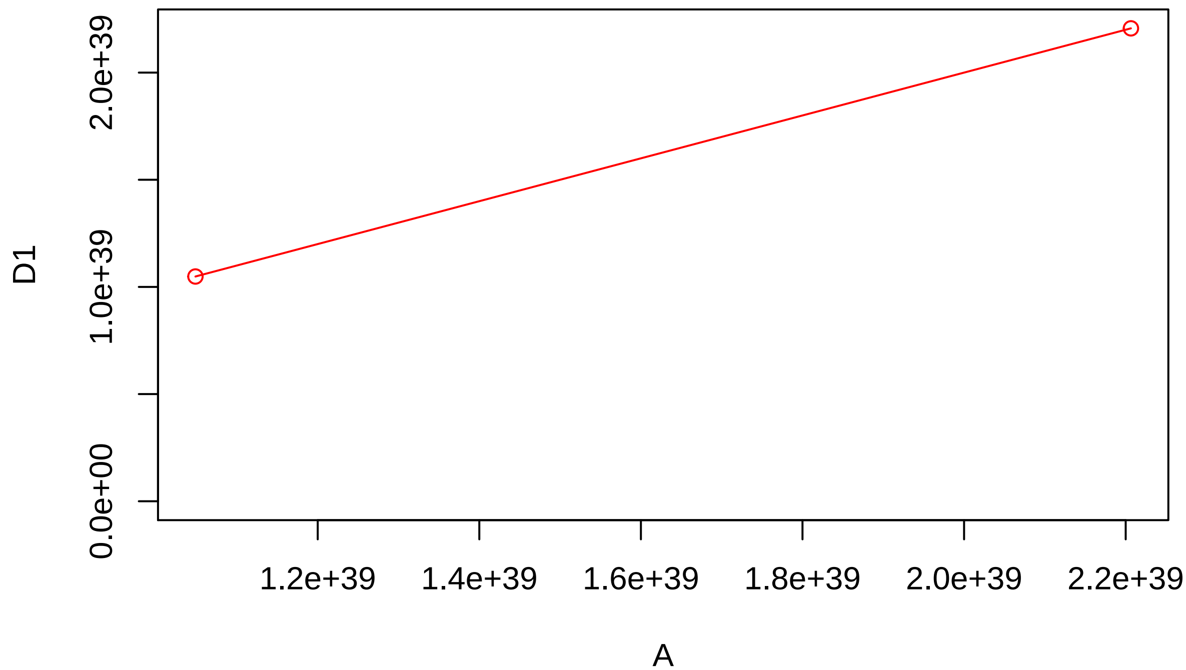
2. ## Comparing A, B and C for large values

```r
n = c(148, 148, 200, 320, 1500)
k = c(101, 100, 179, 191, 990)
i = 0
for(i in 1:5){
A = prod(1:n[i])/(prod(1:k[i])*prod(1:(n[i]-k[i])))
B = prod((k[i]+1):n[i])/prod(1:(n[i]-k[i]))
C = prod(((k[i]+1):n[i])/(1:(n[i]-k[i])))
D = choose(n[i], k[i])
cat("A expression    : ", A, fill = TRUE)
cat("B expression    : ", B, fill = TRUE)
cat("C expression    : ", C, fill = TRUE)
cat("choose function : ", D, fill = TRUE)
}
```

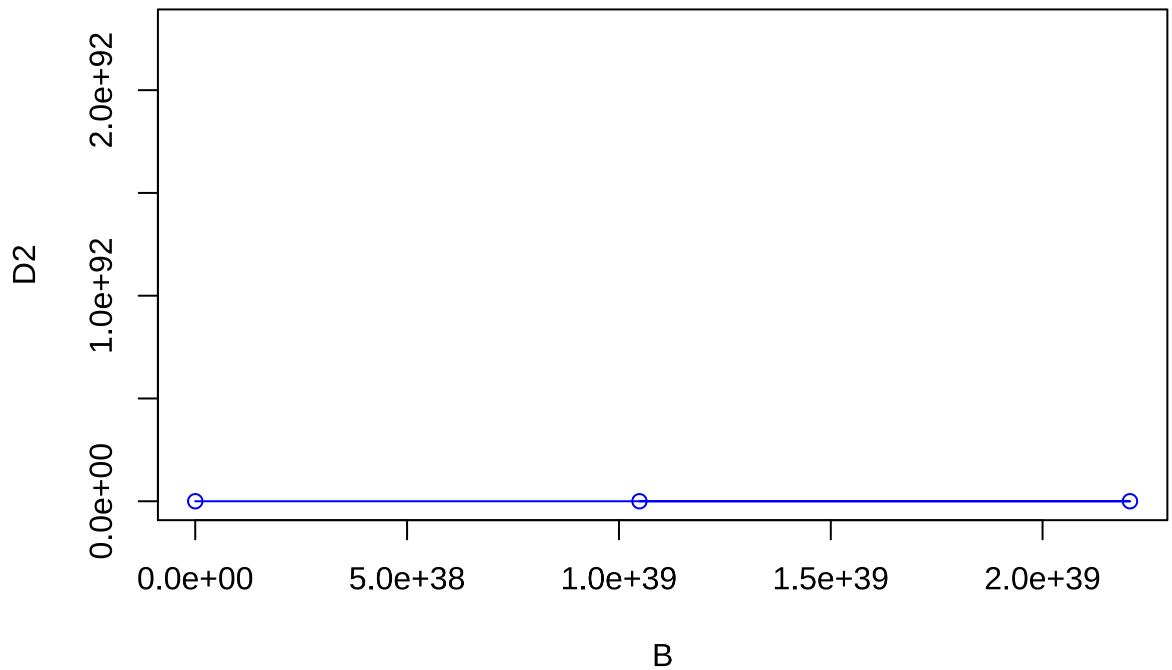
```
A expression    :  1.048632e+39
B expression    :  1.048632e+39
C expression    :  1.048632e+39
choose function :  1.048632e+39
A expression    :  2.206498e+39
B expression    :  2.206498e+39
C expression    :  2.206498e+39
choose function :  2.206498e+39
A expression    :  NaN
B expression    :  1.383075e+28
C expression    :  1.383075e+28
choose function :  1.383075e+28
A expression    :  NaN
B expression    :  Inf
C expression    :  2.300721e+92
choose function :  2.300721e+92
A expression    :  NaN
B expression    :  NaN
C expression    :  Inf
choose function :  Inf
```

Plotting the results from above expressions

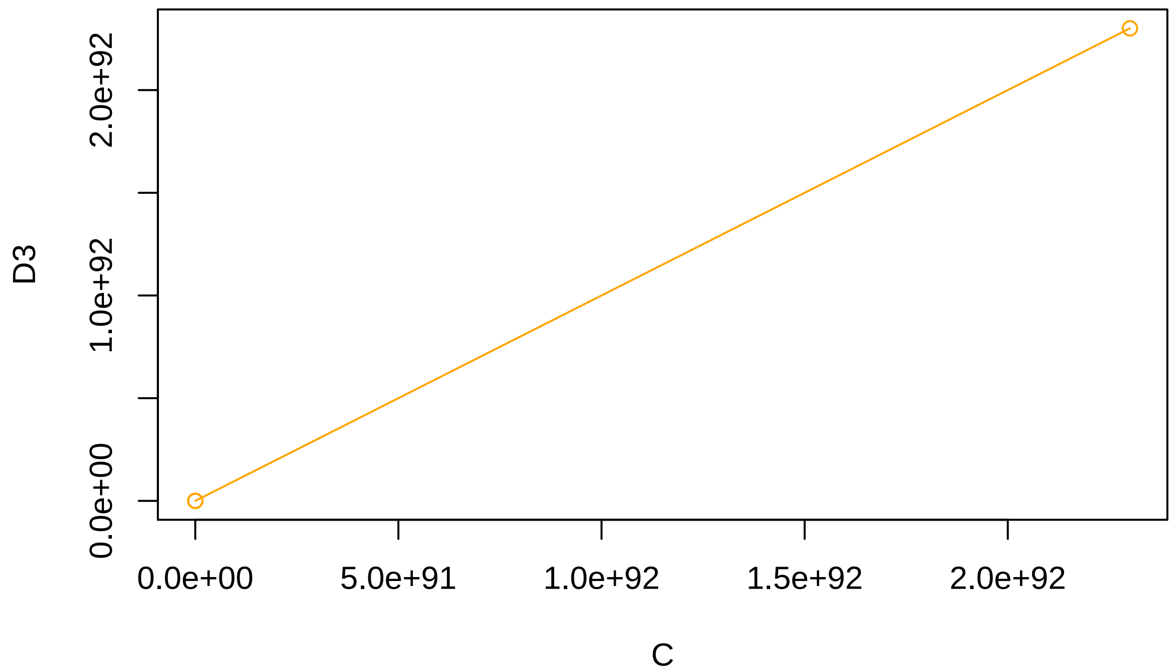
```r
A = c(1.048632e+39, 2.206498e+39, NaN)
D1 = c(1.048632e+39, 2.206498e+39, 1.383075e+28)
B = c(1.048632e+39, 2.206498e+39, 1.383075e+28, NaN)
D2 = c(1.048632e+39, 2.206498e+39, 1.383075e+28, 2.300721e+92)
C = c(1.048632e+39, 2.206498e+39, 1.383075e+28, 2.300721e+92, Inf)
D3 = c(1.048632e+39, 2.206498e+39, 1.383075e+28, 2.300721e+92, Inf)
plot(A, D1, type = "o", col = "red")
```




```r
plot(B, D2, type = "o", col = "blue")
```




```
plot(C, D3, type = "o", col = "orange")
```



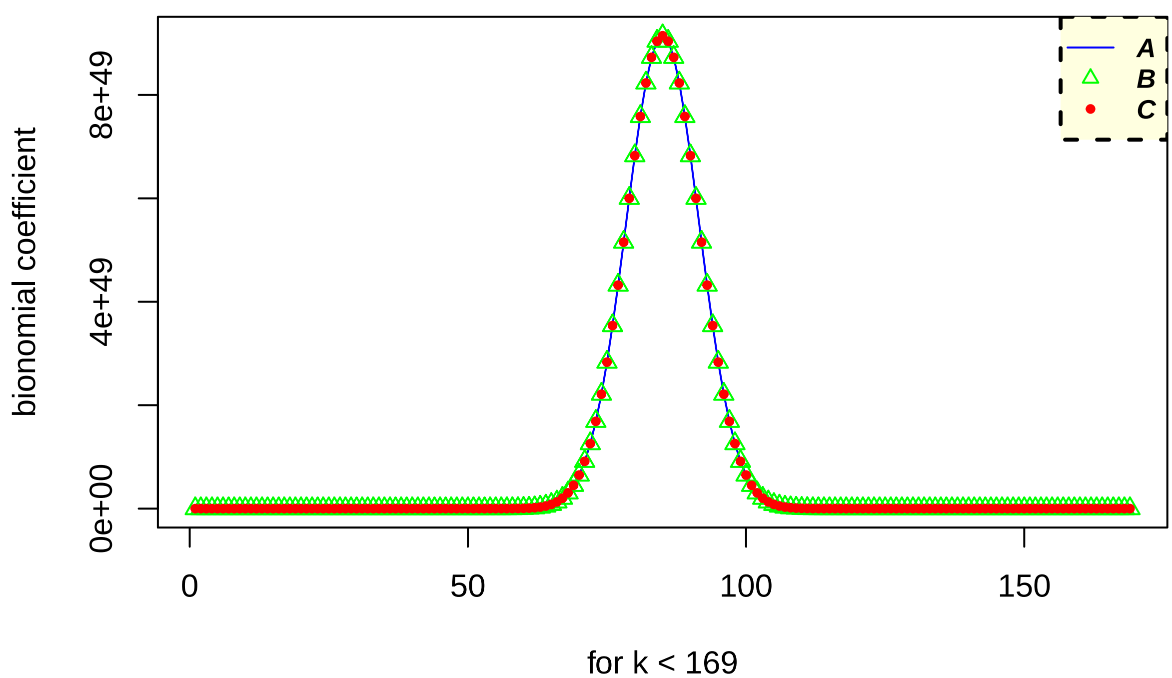
Here, D1, D2 and D3 are the results from choose function for A, B and C expression respectively. A expression throws overflow problem for n and k > 170 giving Inf/Inf = NaN. B expression throws overflow problem for n = 320 and k = 191 giving Inf/Inf = NaN. C expression gives same result as choose function(function to calculate binomial coefficient) and throws overflow problem for very large numbers.

Comparing A, B and C for small values

In the following plot we show the result for A, B and C when n = 173 and for k in range (1:170):

```
f1 <- function(n,k){
  fact1 <- prod(1:n)/(prod(1:k)*prod(1:(n-k)))
  return(fact1)
}                                         # A expression
f2 <- function(n,k){
  fact2 <- prod((k+1):n)/prod(1:(n-k))    # B expression
  return(fact2)

}
f3 <- function(n,k){
  fact3 <- return(fact3)
}                                         #C expression


N = 170
K = 169
fact1 <- vector()
```

```r
for(i in 1:K){
  fact1 <- append(fact1,f1(N,i))
}
plot(1:K,y = fact1,type = "l",col="blue",ylab = "bionomial coefficient",
     xlab = "for k < 169")

fact2 <- vector()
for(i in 1:K){
  fact2 <- append(fact2,f1(N,i))
}
points(1:K,y = fact2,type = "p",col="green",cex=1,pch = 2)

fact3 <- vector()
for(i in 1:K){
  fact3 <- append(fact3,f1(N,i))
}
points(1:K,y = fact2,type = "p",cex=0.8,pch = 20,col="red")
legend("topright",legend=c("A","B","C"),
       col=c("blue","green" ,"red"), lty=c(1,NA,NA),
       pch = c(NA,2,20),cex=0.8, text.font=4, box.lty=2,
       box.lwd=2, box.col="black",bg='lightyellow')
```



For small numbers, A, B and C expressions produces same set of results whereas C expression calculates binomial coefficient for large numbers compared to A and B expressions.

3. For the below value of n and k, A expression gives Nan which is a overflow problem whereas B and C expressions produces the result. The A expression computes product from 1 to n (i.e.199) which exceeds the range of R. Hence it throws overflow problem first among 3 expressions.

```r
n = 199
k = 197
prod(1:n)/(prod(1:k)*prod(1:(n-k)))      # A expression
```

```
[1] NaN
```

```r
prod((k+1):n)/prod(1:(n-k))             # B expression
```

```
[1] 19701
```

```r
prod(((k+1):n)/(1:(n-k)))               # C expression
```

```
[1] 19701
```

```r
choose(n,k)
```

```
[1] 19701
```

For the below value of n and k, A and B expressions gives NaN which is a overflow problem whereas C expression produces the result. The B expression computes product from k+1 to n and k is smaller. Hence it throws overflow problem.

```r
n = 199
k = 19
prod(1:n)/(prod(1:k)*prod(1:(n-k)))      # A expression
```

```
[1] NaN
```

```r
prod((k+1):n)/prod(1:(n-k))             # B expression
```

```
[1] NaN
```

```r
prod(((k+1):n)/(1:(n-k)))               # C expression
```

```
[1] 1.613588e+26
```

```r
choose(n,k)
```

```
[1] 1.613588e+26
```

For the below value of n and k, A and B expressions gives Nan and C expression gives Inf which is a overflow problem. Both C expression and choose function(R built in function for binomial coefficient) gives same result. The C expression first computes the division operation and then prod() is used for the computed value. Here, the overflow problem occurs for very large numbers.

```r
n = 10037
k = 997
prod(1:n)/(prod(1:k)*prod(1:(n-k)))       # A expression
```

```
[1] NaN
```

```r
prod((k+1):n)/prod(1:(n-k))               # B expression
```

```
[1] NaN
```

```r
prod(((k+1):n)/(1:(n-k)))                 # C expression
```

```
[1] Inf
```

```r
choose(n,k)
```

```
[1] Inf
```

Hence we can conclude that C expression calculates binomial coefficient for very large numbers compared to A and B expressions.

Appendix

```
knitr::opts_chunk$set(echo = TRUE, comment = NA)

# -----------------------------------------------------------------------------
# A1
# -----------------------------------------------------------------------------

x1 = 1/3
x2 = 1/4
if ( x1-x2 == 1/12) {
print ("Subtsraction is correct")
}else {
print ( " Substraction is wrong" )
}
x1 = 1
x2 = 1/2
if ( x1-x2 == 1/2) {
  print ("Subtsraction is correct")
}else {
  print ( " Substraction is wrong" )
}
x1 = 1/3
x2 = 1/4
if (isTRUE(all.equal((x1 - x2),1/12))) {
  print ("Subtsraction is correct")
}else {
  print ( " Substraction is wrong" )
}


# -----------------------------------------------------------------------------
# A2
# -----------------------------------------------------------------------------

f = function(x){x}

derivative_func = function(f=f,x){
  e = 10^-15
  f1 = (f(x + e) - f(x)) / e
  return(f1)
}
x1 = derivative_func(f,1)
x1
x100000 = derivative_func(f,100000)
x100000

# -----------------------------------------------------------------------------
# A3
# -----------------------------------------------------------------------------

myvar = function(x){
  n = length(x)
```

```r
  var = (1/(1-n)) * (sum(x^2) - ((1/n) * (sum(x))^2))
  return(var)
}
x = rnorm(10000, mean = 10^8, sd = 1)
n = length(x)
i = 1:10000
Yi = 0
for( j in 1:n){
  Yi[j] = myvar(x[1:j]) - var(x[1:j])
}
plot(x = i , y = Yi)

better_var = function(x){
  len = length(x)
  x_bar = mean(x)
  v = (sum((x - x_bar)^2 ))/ (len-1)
  return(v)
}

n = length(x)
i = 1:10000
Yii = 0
j=0
for( j in 1:n){
  Yii[j] = better_var(x[1:j]) - var(x[1:j])
}

plot(x = i, y = Yii, ylim = c(-3,3))


# -----------------------------------------------------------------------------
# A4
# -----------------------------------------------------------------------------

n = 8
k = 14
prod(1:n)/(prod(1:k)*prod(1:(n-k)))      # A expression
prod((k+1):n)/prod(1:(n-k))              # B expression
prod(((k+1):n)/(1:(n-k)))                # C expression
choose(n,k)
n = 8
k = 8
prod(1:n)/(prod(1:k)*prod(1:(n-k)))      # A expression
prod((k+1):n)/prod(1:(n-k))              # B expression
prod(((k+1):n)/(1:(n-k)))                # C expression
choose(n,k)
n = 10
k = -12
prod(1:n)/(prod(1:k)*prod(1:(n-k)))      # A expression
prod((k+1):n)/prod(1:(n-k))              # B expression
prod(((k+1):n)/(1:(n-k)))                # C expression
choose(n,k)
n = 10
```

```r
k = 0
prod(1:n)/(prod(1:k)*prod(1:(n-k)))      # A expression
prod((k+1):n)/prod(1:(n-k))              # B expression
prod(((k+1):n)/(1:(n-k)))                # C expression
choose(n,k)
n = 0
k = 0
prod(1:n)/(prod(1:k)*prod(1:(n-k)))      # A expression
prod((k+1):n)/prod(1:(n-k))              # B expression
prod(((k+1):n)/(1:(n-k)))                # C expression
choose(n,k)
n = c(148, 148, 200, 320, 1500)
k = c(101, 100, 179, 191, 990)
i = 0
for(i in 1:5){
A = prod(1:n[i])/(prod(1:k[i])*prod(1:(n[i]-k[i])))
B = prod((k[i]+1):n[i])/prod(1:(n[i]-k[i]))
C = prod(((k[i]+1):n[i])/(1:(n[i]-k[i])))
D = choose(n[i], k[i])
cat("A expression    : ", A, fill = TRUE)
cat("B expression    : ", B, fill = TRUE)
cat("C expression    : ", C, fill = TRUE)
cat("choose function : ", D, fill = TRUE)
}
A = c(1.048632e+39, 2.206498e+39, NaN)
D1 = c(1.048632e+39, 2.206498e+39, 1.383075e+28)
B = c(1.048632e+39, 2.206498e+39, 1.383075e+28, NaN)
D2 = c(1.048632e+39, 2.206498e+39, 1.383075e+28, 2.300721e+92)
C = c(1.048632e+39, 2.206498e+39, 1.383075e+28, 2.300721e+92, Inf)
D3 = c(1.048632e+39, 2.206498e+39, 1.383075e+28, 2.300721e+92, Inf)
plot(A, D1, type = "o", col = "red")
plot(B, D2, type = "o", col = "blue")
plot(C, D3, type = "o", col = "orange")
f1 <- function(n,k){
  fact1 <- prod(1:n)/(prod(1:k)*prod(1:(n-k)))
  return(fact1)
}                                         # A expression
f2 <- function(n,k){
  fact2 <- prod((k+1):n)/prod(1:(n-k))    # B expression
  return(fact2)

}
f3 <- function(n,k){
  fact3 <- return(fact3)
}                                         #C expression


N = 170
K = 169
fact1 <- vector()
for(i in 1:K){
  fact1 <- append(fact1,f1(N,i))
}
```

```r
plot(1:K,y = fact1,type = "l",col="blue",ylab = "bionomial coefficient",
     xlab = "for k < 169")

fact2 <- vector()
for(i in 1:K){
  fact2 <- append(fact2,f1(N,i))
}
points(1:K,y = fact2,type = "p",col="green",cex=1,pch = 2)

fact3 <- vector()
for(i in 1:K){
  fact3 <- append(fact3,f1(N,i))
}
points(1:K,y = fact2,type = "p",cex=0.8,pch = 20,col="red")
legend("topright",legend=c("A","B","C"),
       col=c("blue","green" ,"red"), lty=c(1,NA,NA),
       pch = c(NA,2,20),cex=0.8, text.font=4, box.lty=2,
       box.lwd=2, box.col="black",bg='lightyellow')

n = 199
k = 197
prod(1:n)/(prod(1:k)*prod(1:(n-k)))      # A expression
prod((k+1):n)/prod(1:(n-k))              # B expression
prod(((k+1):n)/(1:(n-k)))                # C expression
choose(n,k)
n = 199
k = 19
prod(1:n)/(prod(1:k)*prod(1:(n-k)))      # A expression
prod((k+1):n)/prod(1:(n-k))              # B expression
prod(((k+1):n)/(1:(n-k)))                # C expression
choose(n,k)
n = 10037
k = 997
prod(1:n)/(prod(1:k)*prod(1:(n-k)))      # A expression
prod((k+1):n)/prod(1:(n-k))              # B expression
prod(((k+1):n)/(1:(n-k)))                # C expression
choose(n,k)
```