Exercice n°1

Variance calculation

Grégoire de Lambertye

2022-10-10

Variance calculation

The aim of this first exercice is to approache the difficulties of computer simulation and to get used to R and R Markdown. In order to illustrate these problems we will use the variance calculation through 4 different algorithms and the "var" function provided by R.

As starting point, we will use theese lines

```
library(microbenchmark)#Allows the use of the microbenchmark library

set.seed(11220221)#Create random data
x1 <- rnorm(100)
x2 <- rnorm(100, mean=1000000)
x3 <- rnorm(100, mean=10)
```

Algorithme n°1: (two-pass algorithme)

The first algorithm follows the traditional variance formula: $s_n^2 = \frac{1}{(n-1)} \sum_{i=1}^{n} (x_i - x_n)^2$. It needs to read all the data twice, once to calculate the mean and once to calculate the variance.

```
precise <- function(x) {
  sum <- 0
  n <- length(x)

  #First pass: mean calculation
  for (i in x) {
    sum <- sum + i
  }
  mean <- sum/n

  variance <- 0
  #Second pass: variance calculation
  for(i in x) {
    variance <- variance + (i - mean)^2
  }
  variance <- variance/(n-1)
  return(variance)
}
```

Algorithme n°2: (one-pass algorithme)

The second algorithm use the Variance Decomposition princips : $s_n^2 = \frac{1}{(n-1)}(\sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} x_i)^2)$. This allows the algorithm to read the data only once.

```
excel <- function(x) {
  P1 <- 0
  P2 <- 0
  n <- length(x)
  variance <- 0

  for (i in x) {
    P1 <- P1 + i^2
    P2 <- P2 + i
  }
  P2 <- (P2^2)/n
  variance <- (P1-P2)/(n-1)
  return(variance)
}
```

Algorithme n°3: (shifted one-pass algorithme)

The thrid algorithm works with the Scale Invariance property : $s_x^2 = s_{x-c}^2$ with c a constant. That gives us the following formula :

Consider what would be a good value for c ?

Considering the compututation pinciples of a computer, it would be interesting to work with small number (i.e: approaching 0) so giving c the median value should be interessting.

```
shifted <- function(x, c=x[1]) {
  P1 <- 0
  P2 <- 0
  n <- length(x)
  variance <- 0

  for (i in x) {
    P1 <- P1 + (i-c)^2
    P2 <- P2 + i-c
  }
  P2 <- (P2^2)/n
  variance <- (P1-P2)/(n-1)
  return(variance)
}
```

Algorithme n°4: (online algorithme)

The last algortihm is based on the online calulation of the variance :

```r
online <- function(x) {
  #initalisation
  n <- 2
  mean <- (x[1]+x[2])/2
  variance <- (x[1]-mean)^2 + (x[2]-mean)^2

  for (i in 3:length(x)) {
    n <- n+1
    variance <- ((n-2)/(n-1)) * variance + ((x[i]-mean)^2/n)
    mean <- mean + (x[i]-mean)/n
  }
  return(variance)
}
```

Comparison

To facilitate the comparison between the different algorithms we will use a wrapper function that call every algorithm

```r
variances <- function(x){
  return(c(precise(x), excel(x), shifted(x), online(x),var(x)))
}
```

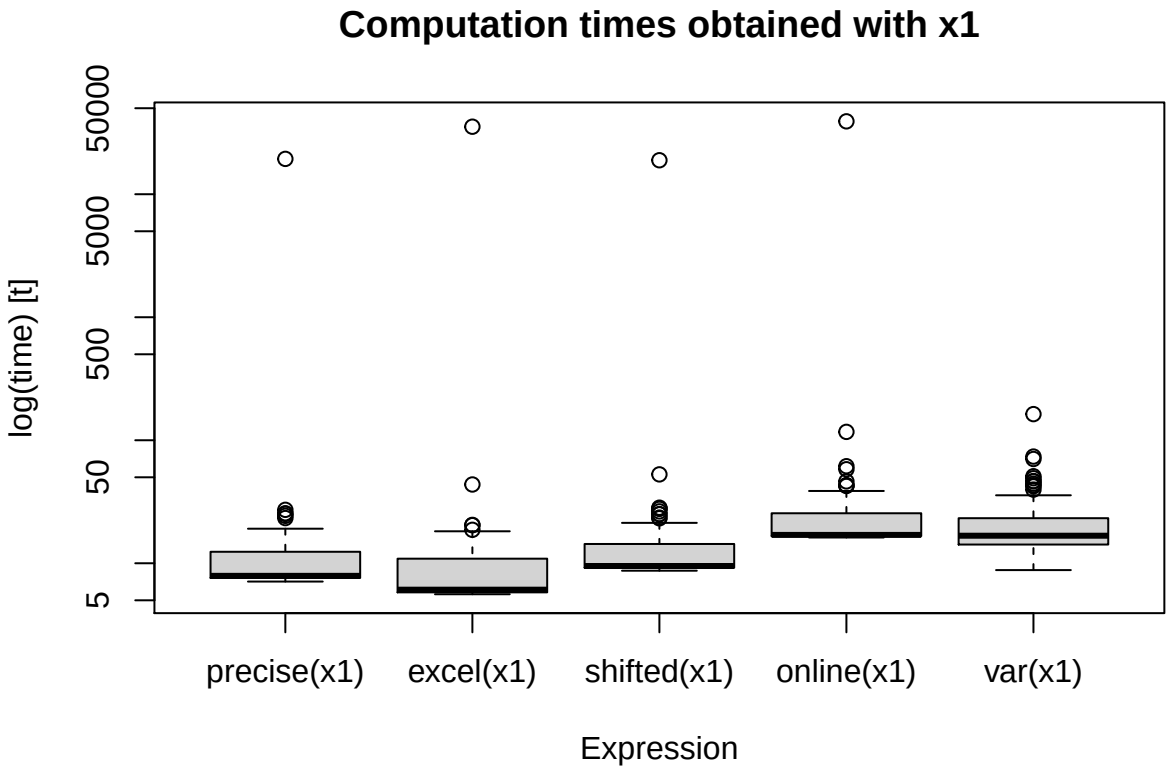
Computation time

Let's focus on the computation time, we will run each algoritm 100 times thank to the microbenchmark function using the x1 dataset.

```r
micro <- microbenchmark(precise(x1), excel(x1), shifted(x1), online(x1),var(x1),  times=100)
knitr::kable(summary(micro))
```

expr	min	lq	mean	median	uq	max	neval	cld
precise(x1)	7.101	7.6010	203.90702	7.9010	12.4010	19375.000	100	a
excel(x1)	5.601	5.8020	362.37009	6.1005	10.9005	35382.902	100	a
shifted(x1)	8.701	9.2010	201.49504	9.5015	14.3510	18882.902	100	a
online(x1)	16.200	16.6505	413.64994	17.0015	25.5010	39113.601	100	a
var(x1)	8.801	14.2010	22.38293	16.7505	23.2515	163.101	100	a

```r
boxplot(micro, main="Computation times obtained with x1")
```



Thank to the boxplot it clearly appears that the excel algorithm is the speediest one and the online one is the worth.

Would you know another way in R to compare computing times?

Recording computing time in R can also be done with the system time :

```
start_time <- Sys.time()
invisible(excel(x1))
end_time <- Sys.time()
computation_time = end_time-start_time
print(computation_time)
```

```
## Time difference of 0.002999067 secs
```

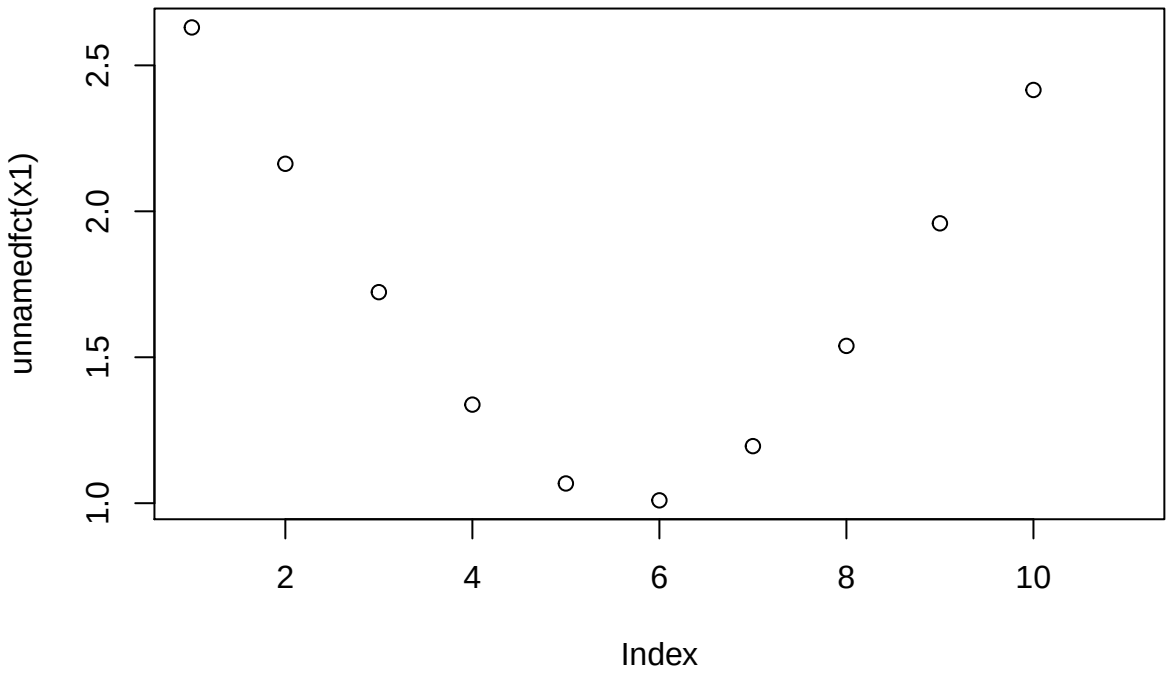
Scale invariance property

Thanks to the scale invariance property, we can assume that with c a constant. We can investigate this property with the shifted algorithm by changing the c-value.

```
condition_number <- function(mean, n , S){
  return(sqrt(1+(mean^2*n)/S))
}
```

```
unnamedfct <- function(x){
  minimum <- min(x)
  maximum <- max(x)
  c_list <- seq(from=minimum, to=maximum, length.out=10)
  condition_numb <- c(0:10)
  for(i in 0:length(c_list)+1){
    mean <- mean(x) - c_list[i]
    n <- 100
    S <- shifted(x,c_list[i])*(n-1)
    condition_numb[i] <- condition_number(mean, n ,S)
  }
  return(condition_numb)
}

plot(unnamedfct(x1))
```



We will examine the result obtained by each algorithm on the same two datasets we have set up earlier.

```
library(xtable)

res <- matrix(c(variances(x1),variances(x2)), ncol=5, byrow=TRUE)
res <- as.table(res)
col_name <- c("precise", "excel", "shifted", "online", "var")
raw_name <- c("x1","x2")
rownames(res) <- raw_name
```

```
colnames(res) <- col_name
knitr::kable(res, caption = "Variance calulation")
```

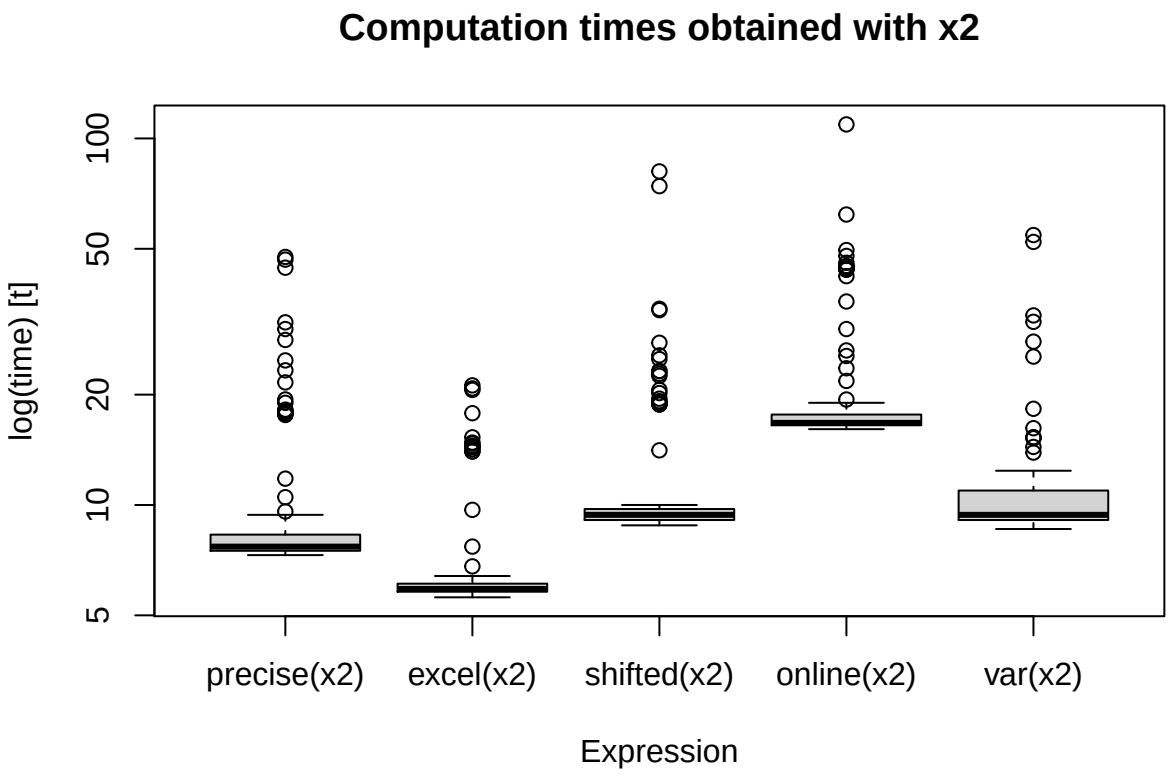
Table 2: Variance calulation

	precise	excel	shifted	online	var
x1	1.1244184	1.1244184	1.1244184	1.1244184	1.1244184
x2	0.9419366	0.9416035	0.9419366	0.9419366	0.9419366

```
microx2 <- microbenchmark(precise(x2), excel(x2), shifted(x2), online(x2),var(x2),  times=100)
knitr::kable(summary(microx2))
```

expr	min	lq	mean	median	uq	max	neval	cld
precise(x2)	7.300	7.501	11.09798	7.7000	8.3015	47.501	100	b
excel(x2)	5.600	5.800	7.36499	5.9010	6.1010	21.200	100	a
shifted(x2)	8.801	9.101	13.16203	9.4010	9.7510	81.301	100	b
online(x2)	16.102	16.501	22.43604	16.7515	17.6515	109.201	100	c
var(x2)	8.601	9.101	11.78299	9.4010	10.9515	54.500	100	b

```
boxplot(microx2, main="Computation times obtained with x2")
```



To compare the result we will use a function that return a binary matrix with a 1 if 2 vector members are the same and a 0 if they differs.

```r
equal_matrix <- function(tab, raw_col_name=c(1:length(tab)),tolerance=1e-05){
  res <- matrix(0, nrow = length(tab), ncol = length(tab))
  for (i in 1:length(tab)){
    for (j in 1:length(tab)){
      comp <- all.equal(tab[i],tab[j],tolerance)
      if(comp == TRUE){
        res[i,j] <- 1
      }
    }
  }
  rownames(res) <- raw_col_name
  colnames(res) <- raw_col_name
  return(res)
}
```

X1 results:

Table 4: X1 result comparison

	precise	excel	shifted	online	var
precise	1	1	1	1	1
excel	1	1	1	1	1
shifted	1	1	1	1	1
online	1	1	1	1	1
var	1	1	1	1	1

X2 results:

Table 5: X2 result comparison

	precise	excel	shifted	online	var
precise	1	0	1	1	1
excel	0	1	0	0	0
shifted	1	0	1	1	1
online	1	0	1	1	1
var	1	0	1	1	1

Usually the result gaven by the excel algorithm differs to the other for x2. It seems to make mistake with big numbers

Condition number

similaire to the derivative, this number allows to Let's assume S is small and use $k = mean * \sqrt{\frac{n}{S}} = \frac{mean}{s_n}$