Assignment 1 - Introduction to Simulation with Variance Estimation

Konstantinos Vakalopoulos 12223236

2023-10-11

Reproduce the example from the first lecture and document your results using R Markdown.

Task 1

Compare the 4 algorithms against R's 'var' function as a gold standard regarding the quality of their estimates.

- Implement all variants of variance calculation as functions.
- Write a wrapper function which calls the different variants.

Below, all the 5 algorithms for the variance calculation are presented as functions. Algorithm 0 is the gold standard (var function from R), algorithm 1 is the normal calculation of variance, algorithm 2 is the excel implementation, algorithm 3 is the shift algorithm and algorithm 4 is the online implementation.

```
##Algorithm 0##
gold_standard <- function(x){
  return(var(x))
}
```

```
##Algorithm 1##
precise <- function(x){
  sample_mean <- sum(x)/length(x)
  variance <- sum((x-sample_mean)^2)/(length(x)-1)
  return(variance)
}
```

```
##Algorithm 2##
excel <- function(x){
  P1 <- sum(x^2)
  P2 <- (sum(x))^2/length(x)
  variance <- (P1-P2)/(length(x)-1)
  return(variance)
}
```

```
##Algorithm 3##
shift <- function(x){
  c <- x[1]
  P1 <- sum((x-c)^2)
  P2 <- (sum(x-c))^2/length(x)
  variance <- (P1-P2)/(length(x)-1)
  return(variance)
}
```

```
##Algorithm 4##
online <- function(x){
  # Calculate the mean and variance for the first and second element
  sample_mean <- (x[1]+x[2])/2
  variance <- ((x[1]-sample_mean)^2+(x[2]-sample_mean)^2)/(2-1)

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

Finally, the variance functions are being called using a wrapper function. We pass the data set x to the wrapper function, which was created using the command rnorm() and the seed was set to 12223236 (student ID).

```
wrapper_function <- function(x){
  # Create a data frame for the variance result
  results <- data.frame(
    Method = c("Gold Standard", "Precise",
               "Excel", "Shift", "Online"),
    Variance = c(
      gold_standard(x),
      precise(x),
      excel(x),
      shift(x),
      online(x)
    )
  )
  return(results)
}

set.seed(12223236)
x <- rnorm(100)
# Compare variance calculation methods
comparison_results <- wrapper_function(x)
```

The library(knitr) is used for good representation of the tables

```
library(knitr) # this library is used for good representation of the tables
kable(comparison_results, format = "markdown", caption = "Variances")
```

Table 1: Variances

Method	Variance
Gold Standard	0.7482808
Precise	0.7482808
Excel	0.7482808
Shift	0.7482808
Online	0.7482808

Task 2

Compare the computational performance of the 4 algorithms against R's 'var' function as a gold standard and summarise them in tables and graphically.

For this task, library microbenchmark was installed and used in order to compare computational performance of the 4 algorithms against R's 'var' function.

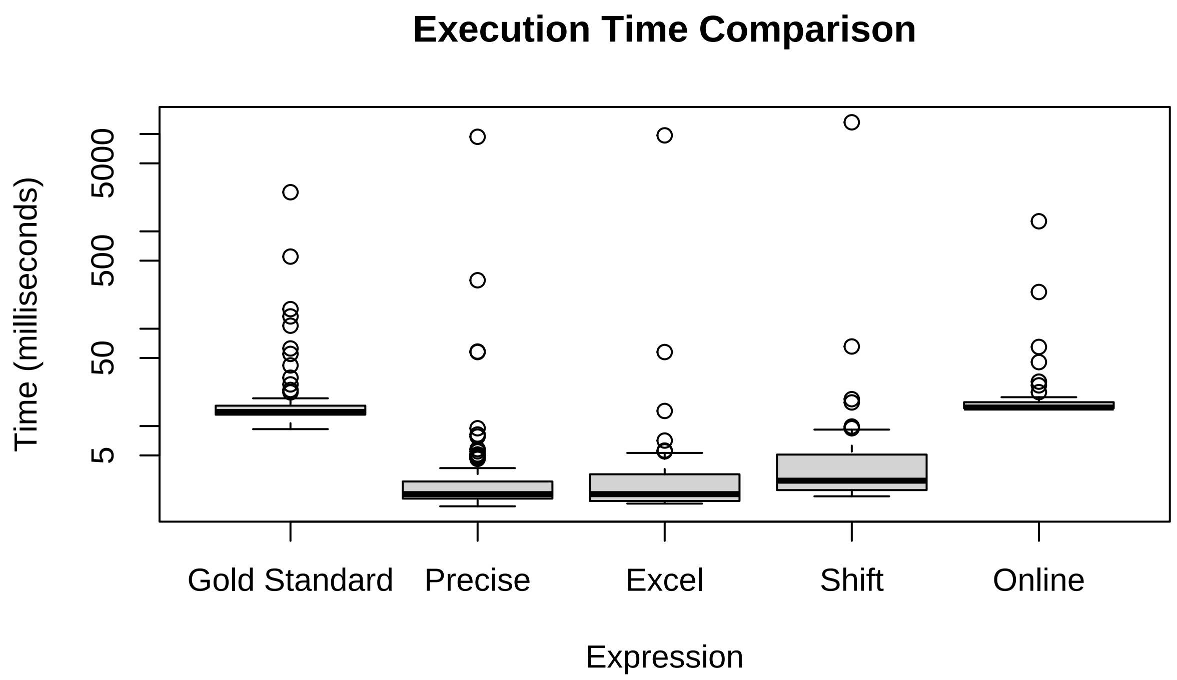
```
# install.packages("microbenchmark")
library(microbenchmark)

# Table Summarize
mb <- microbenchmark(
  "Gold Standard" = gold_standard(x),
  "Precise" = precise(x),
  "Excel" = excel(x),
  "Shift" = shift(x),
  "Online" = online(x),
  times = 100
)
kable(summary(mb), format = "markdown", caption = "Computational Performance")
```

Table 2: Computational Performance

expr	min	lq	mean	median	uq	max	neval
Gold Standard	9.3	13.1	49.527	13.95	16.2	2524.7	100
Precise	1.5	1.8	100.462	2.00	2.7	9365.9	100
Excel	1.6	1.7	100.092	2.00	3.2	9690.3	100
Shift	1.9	2.2	136.391	2.75	5.1	13195.6	100
Online	14.8	15.2	31.936	15.50	17.6	1270.8	100

According to the table, the var function from R and the online implementation perform the worst. The performance of other 3 algorithms (precise, excel, shift) is roughly equivalent. The same is observed in the boxplot below.

```
#Graphically summarize
boxplot(mb, main="Execution Time Comparison", ylab="Time (milliseconds)")
```



Task 3

Investigate the scale invariance property for different values and argue why the mean is performing best as mentioned with the condition number.

- Compare the results according to the instructions provided by Comparison I and Comparison II of the slides.
- Provide your results in table format and graphical format comparable to the example in the slides.

First we create the 2 simulated data sets (x1, x2), as we did in the task 1.

```
set.seed(12223236)
x1 <- rnorm(100)
set.seed(12223236)
x2 <- rnorm(100, mean=1000000)
```

Consequently, we compare the results of the 4 algorithms with the gold standard using the operator "==" and the functions identical() and all.equal(). Regarding the x1 data set:

```
# First data set Comparison 1
gold<- gold_standard(x1)
rest <- c(precise(x1),excel(x1),shift(x1),online(x1))

gold == rest
```

```
## [1]  TRUE FALSE FALSE  TRUE
```

```r
# Function identical is used
for (v in rest){
  print(identical(gold,v))
}
```

```
## [1] TRUE
## [1] FALSE
## [1] FALSE
## [1] TRUE
```

```r
# Function all.equal is used
for (v in rest){
  print(all.equal(gold,v))
}
```

```
## [1] TRUE
## [1] TRUE
## [1] TRUE
## [1] TRUE
```

According to the results, the gold standard variance is not equal with excel and shift implementation using the operator "==" and the identical function. On the other hand, the all.equal() function proves that all the calculations are equal.

Regarding the x2 data set:

```r
# Second data set Comparison 1
gold<- gold_standard(x2)
rest <- c(precise(x2),excel(x2),shift(x2),online(x2))

gold == rest
```

```
## [1]  TRUE FALSE  TRUE FALSE
```

```r
# Function identical is used
for (v in rest){
  print(identical(gold,v))
}
```

```
## [1] TRUE
## [1] FALSE
## [1] TRUE
## [1] FALSE
```

```r
# Function all.equal is used
for (v in rest){
  print(all.equal(gold,v))
}
```

```
## [1] TRUE
## [1] "Mean relative difference: 0.0001883182"
## [1] TRUE
## [1] TRUE
```

According to the results, the gold standard variance is not equal with excel and online implementation using the operator "==" and the identical function. On the other hand, the all.equal() function proves that all the calculations are equal except the excel implementation where the Mean relative difference is 0.0001883182.

In this part of this exercise, we calculate the computational performance of the 2 data sets and we also visualize it using boxplots.

```r
# First data set Comparison 2
mb_1 <- microbenchmark(
  "Gold Standard" = gold_standard(x1),
  "Precise" = precise(x1),
  "Excel" = excel(x1),
  "Shift" = shift(x1),
  "Online" = online(x1),
  times = 100
)
```

```r
# Second data set Comparison 2
mb_2 <- microbenchmark(
  "Gold Standard" = gold_standard(x2),
  "Precise" = precise(x2),
  "Excel" = excel(x2),
  "Shift" = shift(x2),
  "Online" = online(x2),
  times = 100
)
```

```r
kable(summary(mb_1), format = "markdown", caption = "Computational Performance of the first data set")
```

Table 3: Computational Performance of the first data set

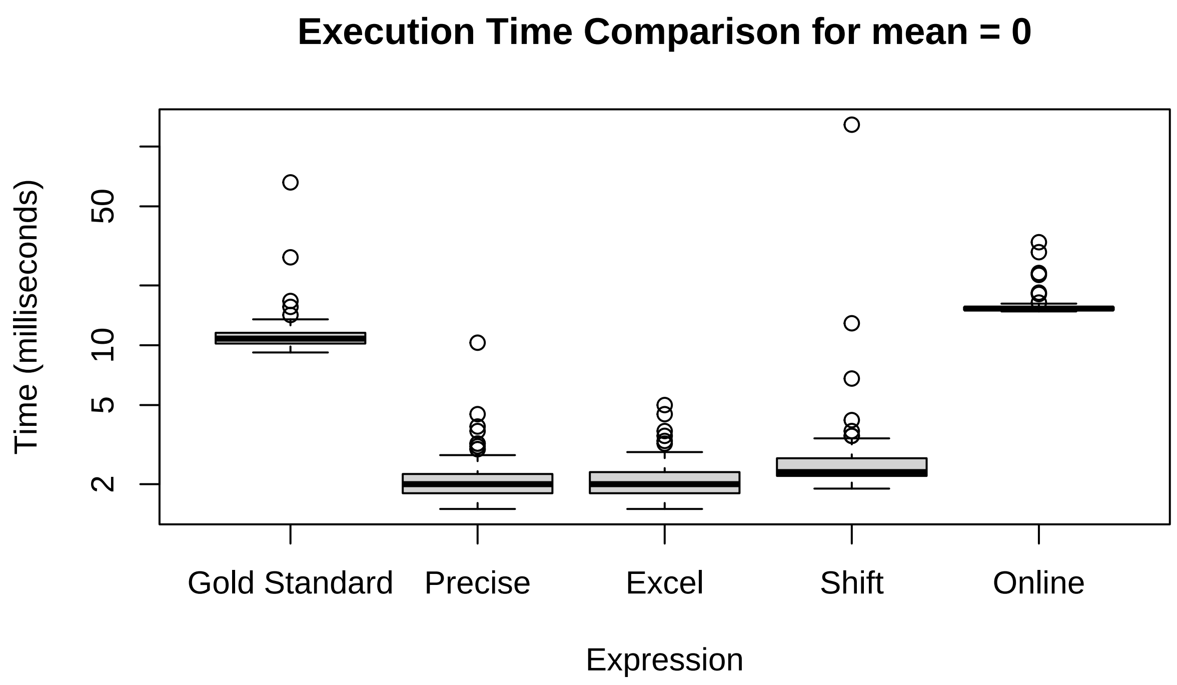
expr	min	lq	mean	median	uq	max	neval
Gold Standard	9.2	10.20	11.771	10.8	11.55	66.0	100
Precise	1.5	1.80	2.188	2.0	2.25	10.3	100
Excel	1.5	1.80	2.125	2.0	2.30	5.0	100
Shift	1.9	2.20	3.867	2.3	2.70	128.8	100
Online	14.8	15.05	15.833	15.3	15.55	33.0	100

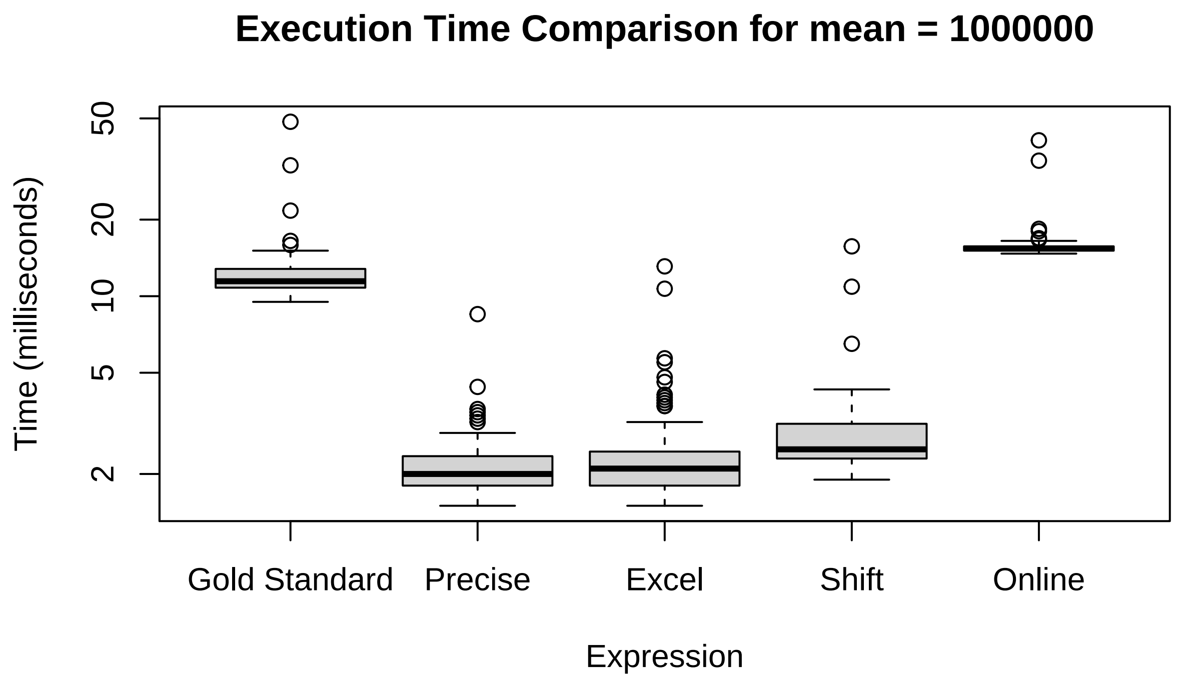
```r
kable(summary(mb_2), format = "markdown", caption = "Computational Performance of the second data set")
```

Table 4: Computational Performance of the second data set

expr	min	lq	mean	median	uq	max	neval
Gold Standard	9.5	10.8	12.508	11.45	12.80	48.5	100
Precise	1.5	1.8	2.216	2.00	2.35	8.5	100
Excel	1.5	1.8	2.504	2.10	2.45	13.1	100
Shift	1.9	2.3	2.911	2.50	3.15	15.7	100
Online	14.7	15.1	15.951	15.40	15.70	41.0	100

```r
#Graphically summarize
boxplot(mb_1, main="Execution Time Comparison for mean = 0", ylab="Time (milliseconds)")
boxplot(mb_2, main="Execution Time Comparison for mean = 1000000", ylab="Time (milliseconds)")
```


Execution Time Comparison for mean = 0
Time (milliseconds)
Gold Standard
Precise
Excel
Shift
Online
Expression


Execution Time Comparison for mean = 1000000
Time (milliseconds)
Gold Standard
Precise
Excel
Shift
Online
Expression