Assignment 1 - Introduction to Simulation with Variance Estimation

Konstantinos Vakalopoulos 12223236

2023-10-12

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){
  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, x[1]),
      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

Method	Variance
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. Furthermore, the two simulated data sets, x1, x2, from the slides, are going to be used not only for the computational performance but the comparison using the of the 4 algorithms with the gold standard using the operator "==" and the functions identical() and all.equal().

First we create the 2 simulated data sets (x1, x2)

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

and then we are going to install and import the library microbenchmark

```
# install.packages("microbenchmark") (Uncomment for installation)
library(microbenchmark)
```

We calculate the computational performance of the 2 data sets and we also visualize it using boxplots.

```
# First data set Comparison
mb_1 <- microbenchmark(
  "Gold Standard" = gold_standard(x1),
  "Precise" = precise(x1),
  "Excel" = excel(x1),
  "Shift" = shift(x1, x1[1]),
  "Online" = online(x1),
  times = 100
)
```

```
# Second data set Comparison
mb_2 <- microbenchmark(
  "Gold Standard" = gold_standard(x2),
  "Precise" = precise(x2),
  "Excel" = excel(x2),
  "Shift" = shift(x2,x2[1]),
  "Online" = online(x2),
  times = 100
)
```

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

Table 2: Computational Performance of the first data set

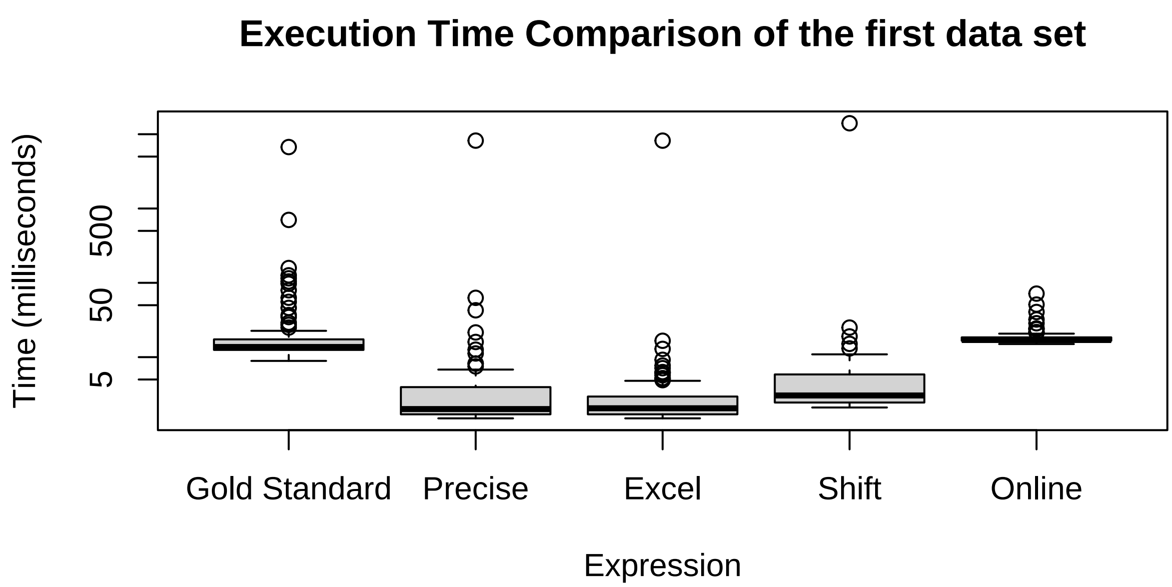
expr	min	lq	mean	median	uq	max	neval
Gold Standard	8.9	12.50	95.609	13.70	17.35	6717.3	100
Precise	1.5	1.70	86.162	2.00	3.95	8202.3	100
Excel	1.5	1.70	84.813	2.05	2.95	8192.1	100
Shift	2.1	2.45	145.089	3.05	5.85	14047.7	100
Online	15.0	16.60	18.877	17.00	18.45	71.6	100

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

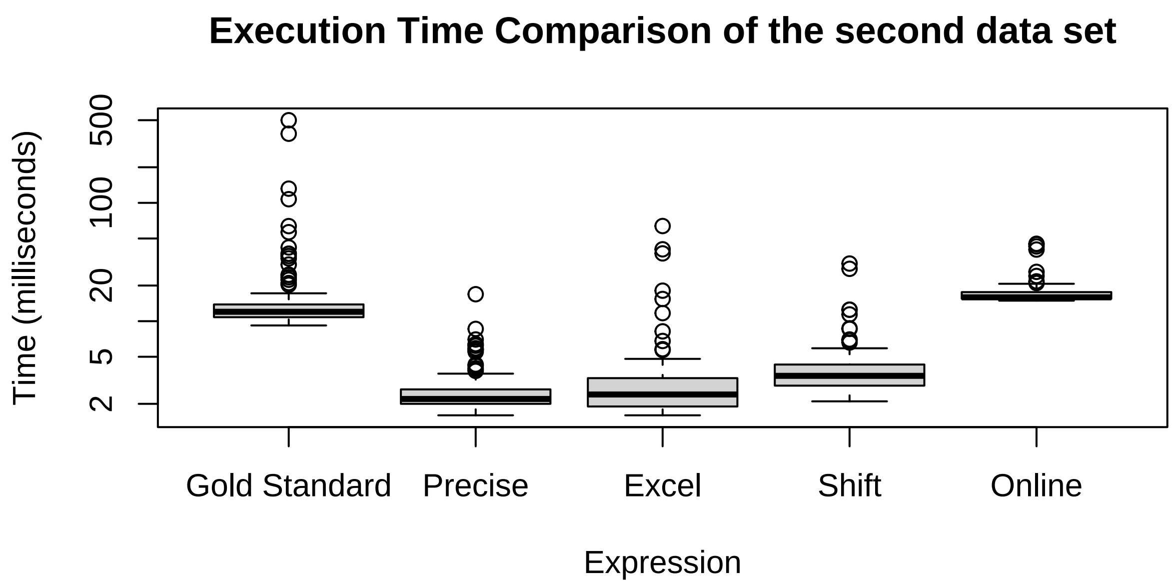
Table 3: Computational Performance of the second data set

expr	min	lq	mean	median	uq	max	neval
Gold Standard	9.2	10.80	25.534	12.00	13.85	499.7	100
Precise	1.6	2.00	2.898	2.20	2.65	16.9	100
Excel	1.6	1.90	4.382	2.40	3.30	63.7	100
Shift	2.1	2.85	4.457	3.45	4.30	30.7	100
Online	14.9	15.45	17.650	15.90	17.60	45.1	100

```r
#Graphically summarize
boxplot(mb_1, main="Execution Time Comparison of the first data set",
        ylab="Time (milliseconds)")
```



```
boxplot(mb_2, main="Execution Time Comparison of the second data set",
        ylab="Time (milliseconds)")
```



According to the tables, 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 boxplots above. Overall, equivalent results, with small differences, are observed in both data sets.

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
gold<- gold_standard(x1)
rest <- c(precise(x1),excel(x1),shift(x1, x1[1]),online(x1))

gold == rest
```

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

```
# 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
gold<- gold_standard(x2)
rest <- c(precise(x2),excel(x2),shift(x2, x2[1]),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.

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.

Through simulations, we are going to investigate the scale invariance property for different values. The shift algorithm is used because this algorithm contains the modification that is done in each value of the data set. Therefore, it will be proven the fact that the mean is performing best as mentioned with the condition number using again the library microbenchmark to compute the computational performance of the shift algorithm for different values of the the scale invariance property. Additionally, we compare the results using the operator "==" and the functions identical() and all.equal(). In the simulations, both x1 and x2 data sets from task 2 are used.

As for the scale invariance property values, our goal is to use values closer to the mean (e.g. median, 1st and 3rd quartile, min, max) and values noticeably higher from the mean.

```
scale_invariance_property_x1 <- c(-1e10,summary(x1),1e10)
scale_invariance_property_x2 <- c(-1e10,summary(x2),1e10)
```

The next step is to write two functions computational_comparison() and equality_comparison(). The computational_comparison() function takes as input the x data set and the scale invariance property values and returns a list with a vector of the variances for each scale invariance property value and the microbenchmark. The second function, equality_comparison(), takes as input the vector of the variances and return 3 matrices that are related to the three equality comparisons, the operator "==" and the functions identical() and all.equal().

```
computational_comparison <- function(x, scale_invariance_property){

  shifted_variance <- c()

  for (c in scale_invariance_property){
    # Calculation of the variance
    shifted_variance <- c(shifted_variance, shift(x,c))
  }
  # Computational performance
  mb <- microbenchmark(
    "Shift -1e10" = shift(x,scale_invariance_property[1]),
    "Shift Min" = shift(x,scale_invariance_property[2]),
    "Shift 1st Qu" = shift(x,scale_invariance_property[3]),
    "Shift Median" = shift(x,scale_invariance_property[4]),
    "Shift Mean" = shift(x,scale_invariance_property[5]),
    "Shift 3rd Qu" = shift(x,scale_invariance_property[6]),
    "Shift Max" = shift(x,scale_invariance_property[7]),
    "Shift 1e10" = shift(x,scale_invariance_property[8]),
    times = 100
  )
  return(list(shifted_variance = shifted_variance,mb = mb))
}
```

```r
equality_comparison <- function(shifted_variance){
  # Initialize empty matrices to store comparison results
  equal_matrix <- matrix(FALSE, nrow = length(shifted_variance),
                         ncol = length(shifted_variance))
  identical_matrix <- matrix(FALSE, nrow = length(shifted_variance),
                             ncol = length(shifted_variance))
  all_equal_matrix <- matrix(FALSE, nrow = length(shifted_variance),
                             ncol = length(shifted_variance))

  # Perform pairwise comparisons using nested loops
  for (i in 1:length(shifted_variance)) {
    for (j in 1:length(shifted_variance)) {
      # Using == operator
      equal_matrix[i, j] <- shifted_variance[i] == shifted_variance[j]

      # Using identical()
      identical_matrix[i, j] <- identical(shifted_variance[i], shifted_variance[j])

      # Using all.equal()
      all_equal_matrix[i, j] <- isTRUE(all.equal(shifted_variance[i], shifted_variance[j]))
    }
  }
  return(list(equal_matrix = equal_matrix,
              identical_matrix = identical_matrix,
              all_equal_matrix = all_equal_matrix))
}
```

Regarding the first data set:

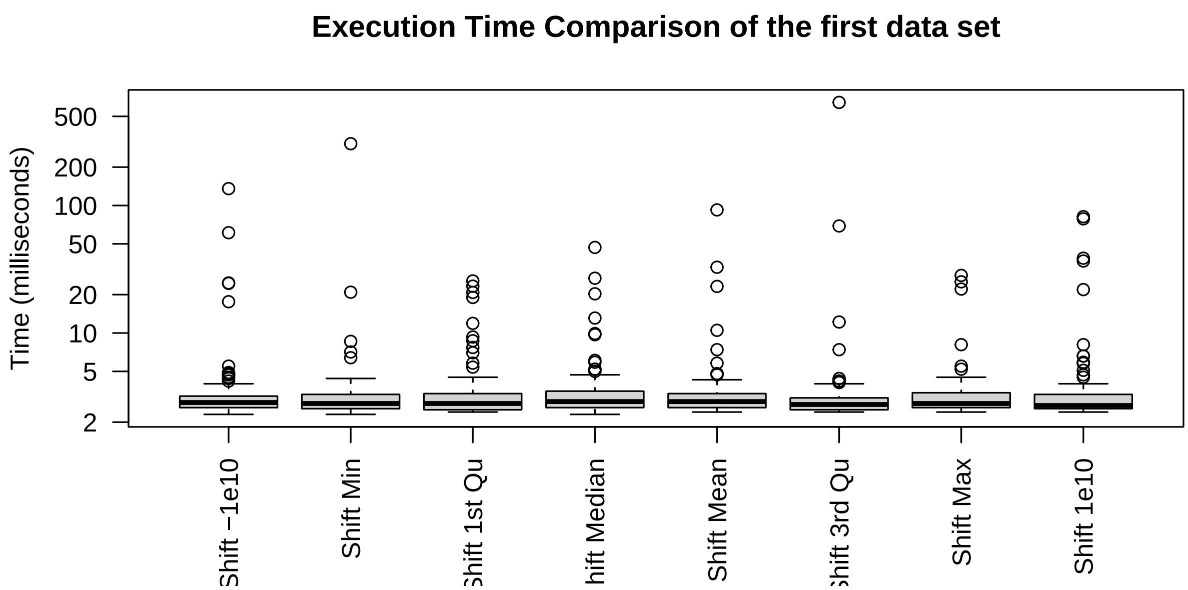
```r
results_x1 <- computational_comparison(x1,scale_invariance_property_x1)

kable(summary(results_x1$mb), format = "markdown",
      caption = "Computational Performance of the shift
                algoritm for the first data set")
```

Table 4: Computational Performance of the shift algoritm for the first data set

expr	min	lq	mean	median	uq	max	neval
Shift -1e10	2.3	2.60	5.496	2.85	3.20	135.6	100
Shift Min	2.3	2.55	6.277	2.80	3.30	305.2	100
Shift 1st Qu	2.4	2.50	4.026	2.80	3.35	25.6	100
Shift Median	2.3	2.60	4.198	2.90	3.50	46.9	100
Shift Mean	2.4	2.60	4.529	2.90	3.35	92.4	100
Shift 3rd Qu	2.4	2.50	10.100	2.75	3.10	643.6	100
Shift Max	2.4	2.60	3.738	2.80	3.40	28.3	100
Shift 1e10	2.4	2.55	5.534	2.70	3.30	81.7	100

```r
#Graphically summarize
boxplot(results_x1$mb, main="Execution Time Comparison of the first data set", xlab = "",
        ylab="Time (milliseconds)",las = 2)
```



Regarding the second data set:

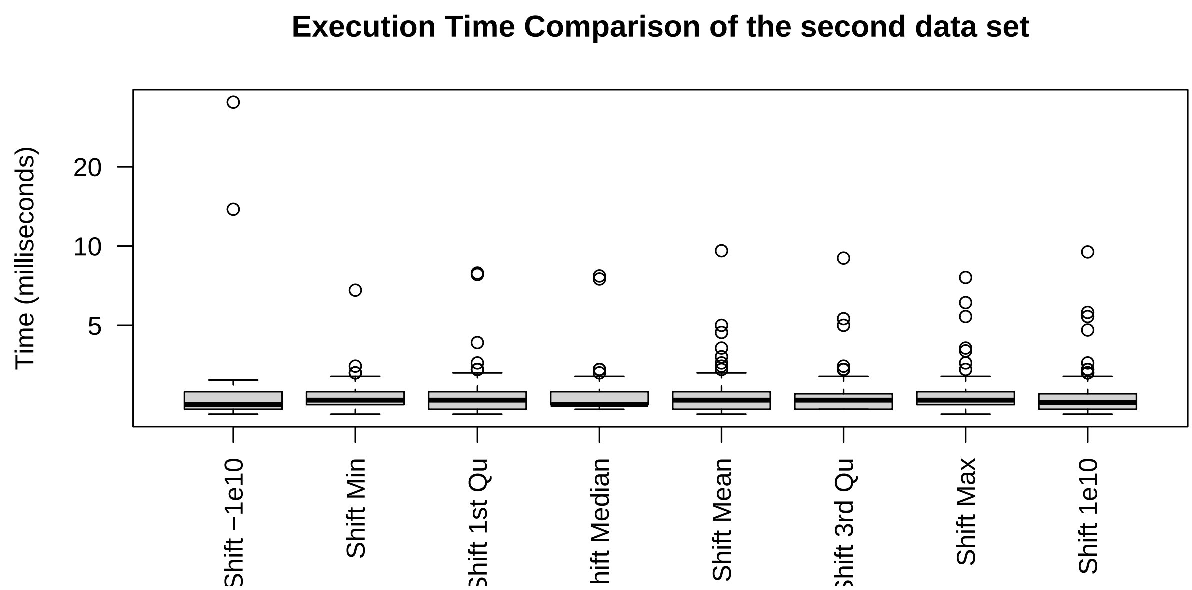
```
results_x2 <- computational_comparison(x2,scale_invariance_property_x2)

kable(summary(results_x2$mb), format = "markdown",
      caption = "Computational Performance of the shift
                algoritm for the second data set")
```

Table 5: Computational Performance of the shift algoritm for the second data set

expr	min	lq	mean	median	uq	max	neval
Shift -1e10	2.3	2.4	3.024	2.50	2.80	35.2	100
Shift Min	2.3	2.5	2.697	2.60	2.80	6.8	100
Shift 1st Qu	2.3	2.4	2.769	2.60	2.80	7.9	100
Shift Median	2.4	2.5	2.719	2.50	2.80	7.7	100
Shift Mean	2.3	2.4	2.769	2.60	2.80	9.6	100
Shift 3rd Qu	2.4	2.4	2.743	2.60	2.75	9.0	100
Shift Max	2.3	2.5	2.782	2.60	2.80	7.6	100
Shift 1e10	2.3	2.4	2.775	2.55	2.75	9.5	100

```
#Graphically summarize
boxplot(results_x2$mb, main="Execution Time Comparison of the second data set", xlab = "",
        ylab="Time (milliseconds)",las = 2)
```



According to the above tables and boxplots, all the results are equivalent. This leads to the fact that it cannot be proven that the mean is the optimal scale invariance property. However, this can be relative, as we are referring to computational performance, which can vary from one computer to another. More specifically, the values can be influenced by the hardware of each computer.

Subsequently, the equality of the variance values for each data set will be checked. The "equal_matrix" indicates the comparison using the "==" operator, the "identical_matrix" indicates the comparison using the function identical() and the "all_equal_matrix" indicates the comparison using the function all.equal().

Regarding the first data set:

```r
comparisons_x1 <- equality_comparison(results_x1$shifted_variance)
cap <- c("Operator ==", "Function identical()", "Function all.equal()")
names <- c("Shift -1e10","Shift Min","Shift 1st Qu","Shift Median",
           "Shift Mean","Shift 3rd Qu","Shift Max", "Shift 1e10")

for (i in 1:3){
  frame <- data.frame(comparisons_x1[[i]])
  colnames(frame) <- names
  rownames(frame) <- names
  print(cap[i])
  cat("\n")
  print(frame)
  cat("\n")
}
```

```
## [1] "Operator =="
##
##              Shift -1e10 Shift Min Shift 1st Qu Shift Median Shift Mean
## Shift -1e10         TRUE     FALSE        FALSE        FALSE      FALSE
## Shift Min          FALSE      TRUE        FALSE        FALSE      FALSE
## Shift 1st Qu       FALSE     FALSE         TRUE        FALSE      FALSE
## Shift Median       FALSE     FALSE        FALSE         TRUE       TRUE
## Shift Mean         FALSE     FALSE        FALSE         TRUE       TRUE
```

```
## Shift 3rd Qu       FALSE      TRUE        FALSE         FALSE      FALSE
## Shift Max          FALSE     FALSE        FALSE         FALSE      FALSE
## Shift 1e10         FALSE     FALSE        FALSE         FALSE      FALSE
##              Shift 3rd Qu Shift Max Shift 1e10
## Shift -1e10         FALSE     FALSE      FALSE
## Shift Min            TRUE     FALSE      FALSE
## Shift 1st Qu        FALSE     FALSE      FALSE
## Shift Median        FALSE     FALSE      FALSE
## Shift Mean          FALSE     FALSE      FALSE
## Shift 3rd Qu         TRUE     FALSE      FALSE
## Shift Max           FALSE      TRUE      FALSE
## Shift 1e10          FALSE     FALSE       TRUE
##
## [1] "Function identical()"
##
##              Shift -1e10 Shift Min Shift 1st Qu Shift Median Shift Mean
## Shift -1e10         TRUE     FALSE        FALSE         FALSE      FALSE
## Shift Min          FALSE      TRUE        FALSE         FALSE      FALSE
## Shift 1st Qu       FALSE     FALSE         TRUE         FALSE      FALSE
## Shift Median       FALSE     FALSE        FALSE          TRUE       TRUE
## Shift Mean         FALSE     FALSE        FALSE          TRUE       TRUE
## Shift 3rd Qu       FALSE      TRUE        FALSE         FALSE      FALSE
## Shift Max          FALSE     FALSE        FALSE         FALSE      FALSE
## Shift 1e10         FALSE     FALSE        FALSE         FALSE      FALSE
##              Shift 3rd Qu Shift Max Shift 1e10
## Shift -1e10         FALSE     FALSE      FALSE
## Shift Min            TRUE     FALSE      FALSE
## Shift 1st Qu        FALSE     FALSE      FALSE
## Shift Median        FALSE     FALSE      FALSE
## Shift Mean          FALSE     FALSE      FALSE
## Shift 3rd Qu         TRUE     FALSE      FALSE
## Shift Max           FALSE      TRUE      FALSE
## Shift 1e10          FALSE     FALSE       TRUE
##
## [1] "Function all.equal()"
##
##              Shift -1e10 Shift Min Shift 1st Qu Shift Median Shift Mean
## Shift -1e10         TRUE     FALSE        FALSE         FALSE      FALSE
## Shift Min          FALSE      TRUE         TRUE          TRUE       TRUE
## Shift 1st Qu       FALSE      TRUE         TRUE          TRUE       TRUE
## Shift Median       FALSE      TRUE         TRUE          TRUE       TRUE
## Shift Mean         FALSE      TRUE         TRUE          TRUE       TRUE
## Shift 3rd Qu       FALSE      TRUE         TRUE          TRUE       TRUE
## Shift Max          FALSE      TRUE         TRUE          TRUE       TRUE
## Shift 1e10         FALSE     FALSE        FALSE         FALSE      FALSE
##              Shift 3rd Qu Shift Max Shift 1e10
## Shift -1e10         FALSE     FALSE      FALSE
## Shift Min            TRUE      TRUE      FALSE
## Shift 1st Qu         TRUE      TRUE      FALSE
## Shift Median         TRUE      TRUE      FALSE
## Shift Mean           TRUE      TRUE      FALSE
## Shift 3rd Qu         TRUE      TRUE      FALSE
## Shift Max            TRUE      TRUE      FALSE
## Shift 1e10          FALSE     FALSE       TRUE
```

Regarding the second data set:

```r
comparisons_x2 <- equality_comparison(results_x2$shifted_variance)

for (i in 1:3){
  frame <- data.frame(comparisons_x2[[i]])
  colnames(frame) <- names
  rownames(frame) <- names
  print(cap[i])
  cat("\n")
  print(frame)
  cat("\n")
}
```

```
## [1] "Operator =="
##
##                Shift -1e10 Shift Min Shift 1st Qu Shift Median Shift Mean
## Shift -1e10           TRUE     FALSE        FALSE        FALSE      FALSE
## Shift Min            FALSE      TRUE        FALSE        FALSE      FALSE
## Shift 1st Qu         FALSE     FALSE         TRUE         TRUE       TRUE
## Shift Median         FALSE     FALSE         TRUE         TRUE       TRUE
## Shift Mean           FALSE     FALSE         TRUE         TRUE       TRUE
## Shift 3rd Qu         FALSE     FALSE         TRUE         TRUE       TRUE
## Shift Max            FALSE      TRUE        FALSE        FALSE      FALSE
## Shift 1e10            TRUE     FALSE        FALSE        FALSE      FALSE
##                Shift 3rd Qu Shift Max Shift 1e10
## Shift -1e10           FALSE     FALSE       TRUE
## Shift Min             FALSE      TRUE      FALSE
## Shift 1st Qu           TRUE     FALSE      FALSE
## Shift Median           TRUE     FALSE      FALSE
## Shift Mean             TRUE     FALSE      FALSE
## Shift 3rd Qu           TRUE     FALSE      FALSE
## Shift Max             FALSE      TRUE      FALSE
## Shift 1e10            FALSE     FALSE       TRUE
##
## [1] "Function identical()"
##
##                Shift -1e10 Shift Min Shift 1st Qu Shift Median Shift Mean
## Shift -1e10           TRUE     FALSE        FALSE        FALSE      FALSE
## Shift Min            FALSE      TRUE        FALSE        FALSE      FALSE
## Shift 1st Qu         FALSE     FALSE         TRUE         TRUE       TRUE
## Shift Median         FALSE     FALSE         TRUE         TRUE       TRUE
## Shift Mean           FALSE     FALSE         TRUE         TRUE       TRUE
## Shift 3rd Qu         FALSE     FALSE         TRUE         TRUE       TRUE
## Shift Max            FALSE      TRUE        FALSE        FALSE      FALSE
## Shift 1e10            TRUE     FALSE        FALSE        FALSE      FALSE
##                Shift 3rd Qu Shift Max Shift 1e10
## Shift -1e10           FALSE     FALSE       TRUE
## Shift Min             FALSE      TRUE      FALSE
## Shift 1st Qu           TRUE     FALSE      FALSE
## Shift Median           TRUE     FALSE      FALSE
## Shift Mean             TRUE     FALSE      FALSE
## Shift 3rd Qu           TRUE     FALSE      FALSE
## Shift Max             FALSE      TRUE      FALSE
```

```
## Shift 1e10          FALSE     FALSE       TRUE
##
## [1] "Function all.equal()"
##
##              Shift -1e10 Shift Min Shift 1st Qu Shift Median Shift Mean
## Shift -1e10         TRUE     FALSE        FALSE        FALSE      FALSE
## Shift Min          FALSE      TRUE         TRUE         TRUE       TRUE
## Shift 1st Qu       FALSE      TRUE         TRUE         TRUE       TRUE
## Shift Median       FALSE      TRUE         TRUE         TRUE       TRUE
## Shift Mean         FALSE      TRUE         TRUE         TRUE       TRUE
## Shift 3rd Qu       FALSE      TRUE         TRUE         TRUE       TRUE
## Shift Max          FALSE      TRUE         TRUE         TRUE       TRUE
## Shift 1e10          TRUE     FALSE        FALSE        FALSE      FALSE
##              Shift 3rd Qu Shift Max Shift 1e10
## Shift -1e10         FALSE     FALSE       TRUE
## Shift Min            TRUE      TRUE      FALSE
## Shift 1st Qu         TRUE      TRUE      FALSE
## Shift Median         TRUE      TRUE      FALSE
## Shift Mean           TRUE      TRUE      FALSE
## Shift 3rd Qu         TRUE      TRUE      FALSE
## Shift Max            TRUE      TRUE      FALSE
## Shift 1e10          FALSE     FALSE       TRUE
```

Task 4

Compare condition numbers for the 2 simulated data sets and a third one where the requirement is not fulfilled, as described during the lecture.

First, we generate a third data set that the requirement is not fulfilled. According to the lecture notes the requirement is: S is "small" and $\overline{x}$ nonzero, where $\sum_{i=1}^{n}(x_i - \overline{x})^2$. Therefore, the x3 data set will be:

```r
set.seed(12223236)
x3 <- rnorm(100, mean = 1, sd = 10000)
```

Afterwards, the formula from slide 20 is used for the three data sets regarding the calculation of the condition numbers. As for the scale invariance property values, the values from the previous task will be used.

```r
scale_invariance_property_x3 <- c(-1e10,summary(x3),1e10)
```

The formula is: $\tilde{\kappa} = \sqrt{1 + \frac{n}{S}(\overline{x} - c)^2}$.

```r
condition_number <- function(x,c){
  return(sqrt(1+(length(x)*(mean(x)-c)^2)/(sum((x-mean(x))^2))))
}
```

The condition numbers for each data set are calculated, The next table shows the results.

```r
data <- list(x1,x2,x3)
invariance <- list(scale_invariance_property_x1,
                   scale_invariance_property_x2,
                   scale_invariance_property_x3)
```

```r
result_condition_numbers <- data.frame()
for (i in 1:3){
  vec_condition_numbers <- c()
  for (c in invariance[[i]]){
    vec_condition_numbers <- c(vec_condition_numbers,condition_number(data[[i]],c))
  }
  result_condition_numbers <- rbind(result_condition_numbers,vec_condition_numbers)
}
colnames(result_condition_numbers) <- names
rownames(result_condition_numbers) <- c("x1", "x2", "x3")
```

```r
kable(result_condition_numbers, format = "markdown",
      caption = "Condition numbers for all three data sets")
```

Table 6: Condition numbers for all three data sets

	Shift -1e10	Shift Min	Shift 1st Qu	Shift Median	Shift Mean	Shift 3rd Qu	Shift Max	Shift 1e10
x1	11618501038	2.363237	1.246644	1.005776	1	1.200576	2.77146	11618501038
x2	11619662888	2.363237	1.246644	1.005776	1	1.200576	2.77146	11617339187
x3	1161850	2.363237	1.246644	1.005776	1	1.200576	2.77146	1161850

It is obvious from the table and from the formula that the optimal condition number, which is 1, is when the scale invariance property is equal to the mean of the data. Moreover, it is worth mentioning that for invariance property values closer to the mean, the condition numbers are closer to 1, as well. For values noticeable higher than the mean, the condition numbers are increased significantly.