lab01

Thijs Quast (thiqu264)

23-1-2019

Contents

Question 1 - Be careful when comparing

1.1

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

```
## [1] "Subtraction is wrong"
```

```
x1 <- 1
x2 <- 1/2

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

```
## [1] "Subtraction is correct"
```

Both snippets first assign a value to the variables x1 and x2. Afterwards x2 is subracted from x1. For the first snippet, one would expect that subracting 1/4 from 1/3 would result to 1/12. However, one appareantly R stores the values for 1/3 and 1/4, not exactly as the proposed values, therefore the computer does not recognize 1/3-1/4 as being exactly equal to 1/12. Particularly, the value 1/3 is a fraction which never ends if you write it as a decimal, e.g. 0.3333333333333, therefore when subtracing another value from this, although the values are very close 1/3(thus, 0.33333333) - 1/4(0.25) will not result in exactly the same value as 1/12. Also, because 1/12 is again a never ending number if one writes it as a decimal, e.g. 0.0833333333.

For the second snippet, the computer can store the values as exactly the values we want, therefore the condition is met and the snippet prints: "Subtraction is correct".

1.2

One way to solve the problem occuring in question 1.2 is to set a minimum number very close to zero. If we then subtract the condition to be met, from the initial subtraction, one expects to get a value of 0. But because the computer faces some difficulties with storing certain fractions, we can set as a condition for it to be smaller than or equal to a number very close to 0.

```
x1 <- 1/3
x2 <- 1/4
min <- 1*10^-10

if (x1-x2 - (1/12) <= min){
  print("Subtraction is correct")
  }else{
  print("Subtraction is wrong")
}
```

```
## [1] "Subtraction is correct"
```

If one now runs the code above, the if statement will return "Subtraction is correct".

Question 2 - Derivative

2.1

```
derivative <- function(x){
  error <- 10^-15
  numerator <- (x + error) - x
  denominator <- error
  final <- numerator/denominator
  return(final)
}
```

2.2

```
derivative(1)
```

```
## [1] 1.110223
```

```
derivative(100000)
```

```
## [1] 0
```

2.3

We obtained the values 1.110223 and 0. According to the provided formula, for both values we should obtain a value of 1. For the first value, due to underflow, the result is not exactly 1. For the second value, adding a very small number to a very large number, the machine will recognize it as just the large number, therefore the numberator will become 0 and thus the value of the derivative will become 0.

Question 3 - Variance

3.1

```
myvar <- function(x){
  n <- length(x)
  a <- sum(x*x)
  b <- (sum(x))^2
  c <- b/n
  d <- a - c
  e <- d/(n-1)
  return(e)
}
```

3.2

```
vector_x <- rnorm(n = 10000, mean = 10^8, sd = 1)
```

3.3

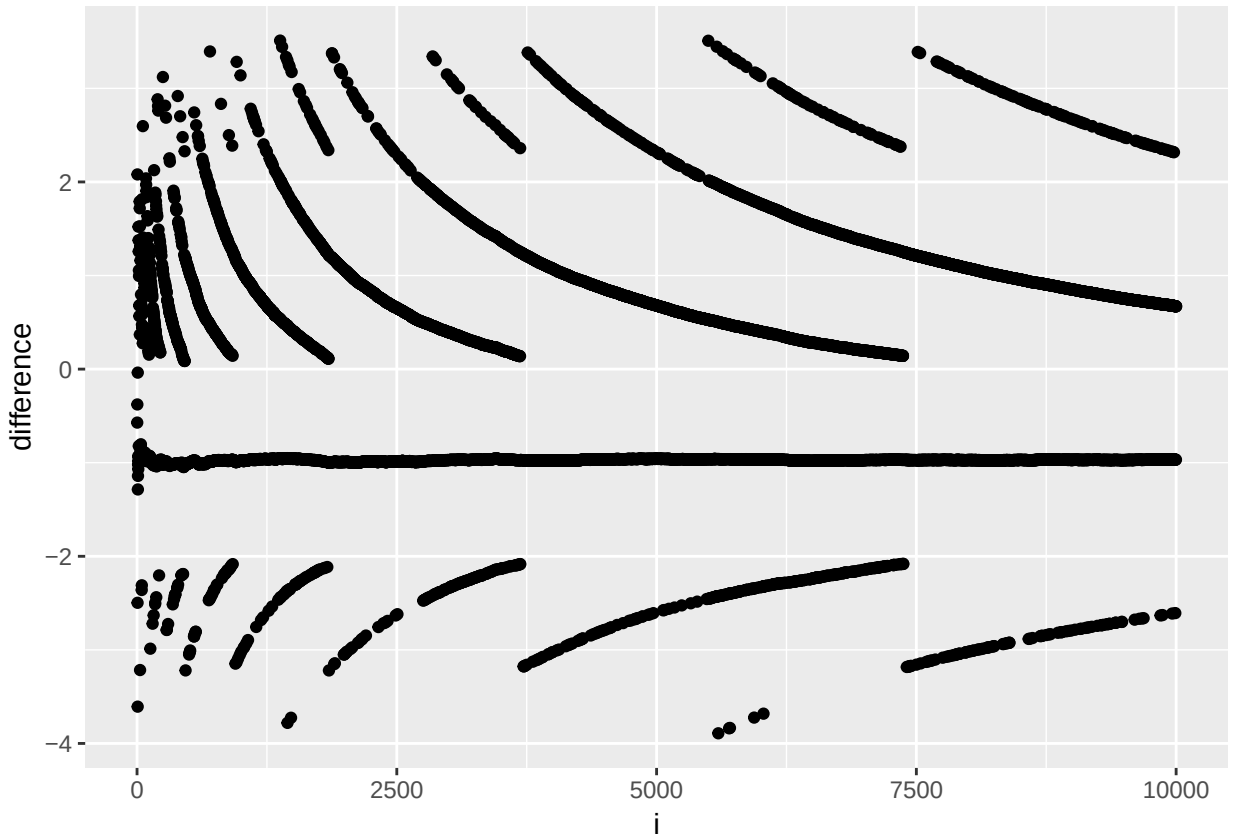
```
mat <- matrix(NA, nrow = length(vector_x), ncol = 1)
elements <- c(1:10000)

for (i in 1:10000){
  subvec <- vector_x[1:i]
  result <- myvar(subvec) - var(subvec)
  mat[i,] <- result
}

mat <- cbind(elements, mat)
df <- as.data.frame(mat)
colnames(df) <- c("i", "difference")
```

```
library(ggplot2)
plot <- ggplot(data = df, aes(x = i, y = difference)) + geom_point()
plot
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



If our function would work perfectly, the difference between the result of our function and the function provided by R would be 0. However, as seen from our plot, certain differences arise between the self written variance function and the function provided by R. Specifically, the values range from -4 to 4.

If our function would work correctly we would see a horizontal line at 0, the result is among other things a horizontal line at -1, meaning in many cases the computed value for variance is not correct. For certain values the difference is even bigger, which explains the differences up to -4 and 4.

3.4

```
myvar_improved <- function(x){
  n <- length(x)
  x_hat <- mean(x)
  variance <- (1/(n-1)) * sum((x-x_hat)^2)
  return(variance)
}
```

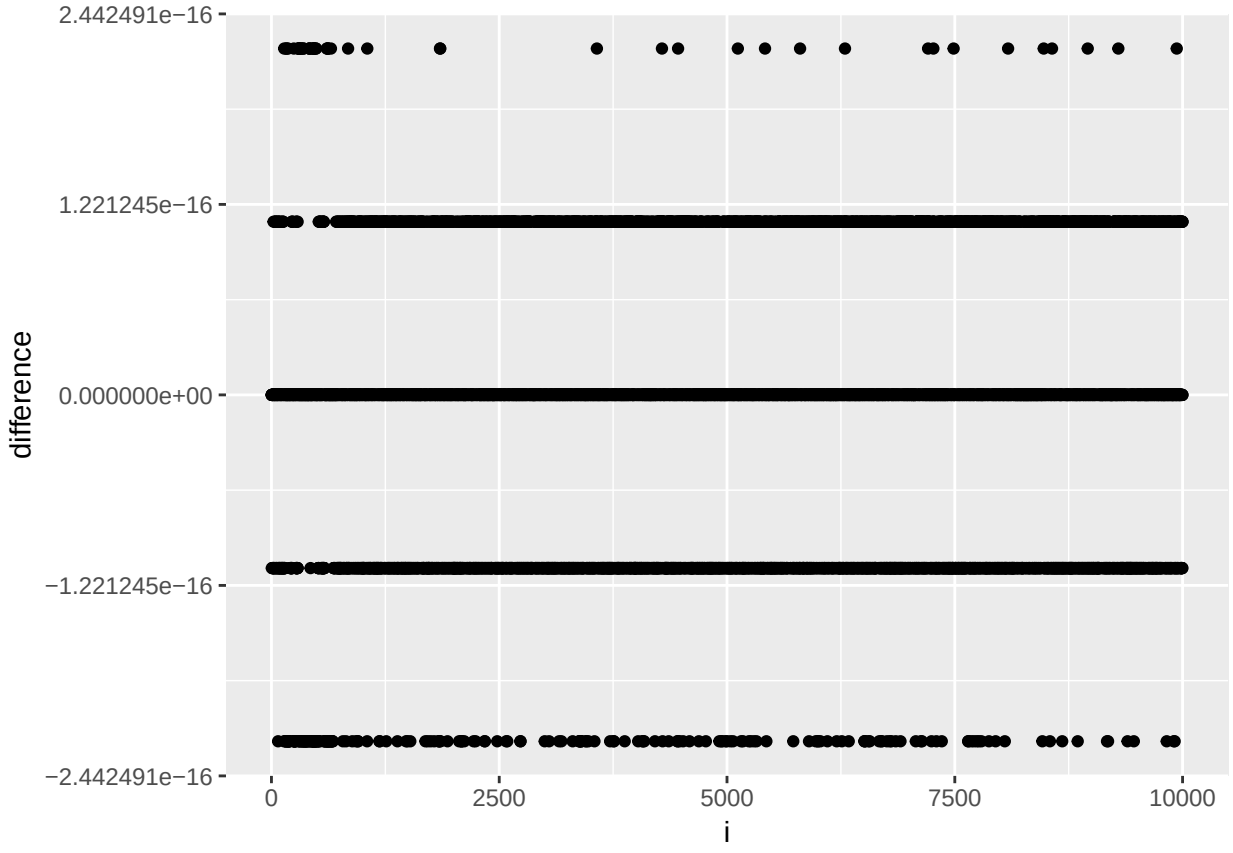
```
mat2 <- matrix(NA, nrow = length(vector_x), ncol = 1)
elements <- c(1:10000)

for (i in 1:10000){
  subvec2 <- vector_x[1:i]
  result <- myvar_improved(subvec2) - var(subvec2)
  mat2[i,] <- result
}

mat2 <- cbind(elements, mat2)
df2 <- as.data.frame(mat2)
colnames(df2) <- c("i", "difference")
```

```
library(ggplot2)
plot2 <- ggplot(data = df2, aes(x = i, y = difference)) + geom_point()
plot2
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



From the plot above one can see that the myvar_improved performs better. The differences between computed variances is really close to 0. As an alternation to the previously provide formula, we now have summed over the square difference between the value in the vector and the mean of the vector and multiplied by 1/n-1.

Question 4 - Linear Algebra

4.1

```
tecator <- read.csv("tecator.csv")
```

4.2

```
y_vector <- as.matrix(tecator[, "Protein"])
x_matrix <- as.matrix(subset(tecator, select = -c(Protein, Sample)))
```

```
A <- t(x_matrix) %*% x_matrix
small_b <- t(x_matrix) %*% y_vector
```

4.3

```
beta <- solve(A, small_b)
```

Running the solve function will return the following error message: "Error in solve.default(A, small_b) : system is computationally singular: reciprocal condition number = 7.13971e-17".

This error could be caused by the fact that the matrix is not invertible (singular), and therefor it is impossible to run a regression. Another explanation is that there is high occurance of multicollinearity.

(source: https://stats.stackexchange.com/questions/76488/ error-system-is-computationally-singular-when-running-a-glm)

(source: https://stats.stackexchange.com/questions/90020/ random-effects-model-with-plm-system-is-computationally-singular-error)

4.4

```
kappa(A)
```

```
## [1] 1.157834e+15
```

Following the ?kappa function in R, we get a very large number for the condition number. A large value for the condition number indicates that this matrix is close to being singular.

(source: http://www.cse.iitd.ernet.in/~dheerajb/CS210_lect07.pdf)

```
y_vector <- scale(y_vector)
x_matrix <- scale(x_matrix)

A <- t(x_matrix) %*% x_matrix
small_b <- t(x_matrix) %*% y_vector
```

```r
beta_scale <- solve(A, small_b)

kappa(A)
```

```
## [1] 490471520662
```

After scaling the variables, the kappa() value has become smaller. We assume this is because the matrix has now become less likely to be singular, which results in a lower value for the condition number of the matrix.

Appendix

```r
x1 <- 1/3
x2 <- 1/4
if (x1-x2 == 1/12){
  print("Subtraction is correct")
  }else{
  print("Subtraction is wrong")
}
x1 <- 1
x2 <- 1/2

if(x1 - x2 == 1/2){
  print("Subtraction is correct")
} else {
  print("Subtraction is wrong")
}
x1 <- 1/3
x2 <- 1/4
min <- 1*10^-10

if (x1-x2 - (1/12) <= min){
  print("Subtraction is correct")
  }else{
  print("Subtraction is wrong")
}
derivative <- function(x){
  error <- 10^-15
  numerator <- (x + error) - x
  denominator <- error
  final <- numerator/denominator
  return(final)
}
derivative(1)
derivative(100000)
myvar <- function(x){
  n <- length(x)
  a <- sum(x*x)
  b <- (sum(x))^2
  c <- b/n
  d <- a - c
  e <- d/(n-1)
  return(e)
```

```
}
vector_x <- rnorm(n = 10000, mean = 10^8, sd = 1)
mat <- matrix(NA, nrow = length(vector_x), ncol = 1)
elements <- c(1:10000)

for (i in 1:10000){
  subvec <- vector_x[1:i]
  result <- myvar(subvec) - var(subvec)
  mat[i,] <- result
}

mat <- cbind(elements, mat)
df <- as.data.frame(mat)
colnames(df) <- c("i", "difference")
library(ggplot2)
plot <- ggplot(data = df, aes(x = i, y = difference)) + geom_point()
plot
myvar_improved <- function(x){
  n <- length(x)
  x_hat <- mean(x)
  variance <- (1/(n-1)) * sum((x-x_hat)^2)
  return(variance)
}
mat2 <- matrix(NA, nrow = length(vector_x), ncol = 1)
elements <- c(1:10000)

for (i in 1:10000){
  subvec2 <- vector_x[1:i]
  result <- myvar_improved(subvec2) - var(subvec2)
  mat2[i,] <- result
}

mat2 <- cbind(elements, mat2)
df2 <- as.data.frame(mat2)
colnames(df2) <- c("i", "difference")
library(ggplot2)
plot2 <- ggplot(data = df2, aes(x = i, y = difference)) + geom_point()
plot2
tecator <- read.csv("tecator.csv")
y_vector <- as.matrix(tecator[, "Protein"])
x_matrix <- as.matrix(subset(tecator, select = -c(Protein, Sample)))
A <- t(x_matrix) %*% x_matrix
small_b <- t(x_matrix) %*% y_vector
beta <- solve(A, small_b)
kappa(A)
y_vector <- scale(y_vector)
x_matrix <- scale(x_matrix)

A <- t(x_matrix) %*% x_matrix
small_b <- t(x_matrix) %*% y_vector

beta_scale <- solve(A, small_b)
```

```
kappa(A)
```