Computational_statistics_Lab1

Hong Zhang (honzh073)

11/6/2021

Question 1: Be careful when comparing

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

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

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

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

1. Check the results of the snippets. Comment what is going on.

–**Answer:** The R code snippet 1 compare $\frac{1}{3}$ - $\frac{1}{4}$ == $\frac{1}{12}$, and output "Substration is wrong"; The R code snippet 2 compare 1 - $\frac{1}{2}$ == $\frac{1}{2}$, and output "Substration is correct";

One reason is that computer number system is not the same as real mathematical numbers. In the computer, fractions are stored as decimals, and there is a certain difference between finite decimal and repeating decimal. Forexample $\frac{1}{3}$ is a repeating decimal, stored as 0.3333333 in R and only seven digits after the decimal point are kept. So "$\frac{1}{3}$ - $\frac{1}{4}$ == $\frac{1}{12}$" in R will output FALSE.

On R code snippets 2, $\frac{1}{2}$ is a finite decimal stored as 0.5. Thus, "1 - $\frac{1}{2}$ == $\frac{1}{2}$" in R will output TRUE.

2. If there are any problems, suggest improvements.

In R code snippet 1 $\frac{1}{3}$ - $\frac{1}{4}$ == $\frac{1}{12}$, and output "Substration is wrong";

For two numbers that are circulating decimals, we can specify the number of digits that need to be compared. For example, in the R code snippet, I compare only the ten digits after the two decimal points , it is the same. The accuracy meets ordinary mathematical calculations.

In this way, the output of R code snippet 1 is "Substration is correct".

```r
x1 <- 1 / 3
x2 <- 1 / 4
# Compare whether the ten digits after the decimal point are equal
if (round(x1 - x2, 10) == round(1 / 12, 10)) {
  print("Substration is correct")
} else{
  print("Substration is wrong")
}
```

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

```r
# Use all.equal() function
# Test If Two Objects Are (Nearly) Equal
if (isTRUE(all.equal(x1 - x2, 1 / 12))) {
  print("Substration is correct")
} else{
  print("Substration is wrong")
}
```

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

Question 2: Derivative

1. Write your own R function to calculate the derivative of f(x) = x in this way with $\epsilon = 10^{-15}$.

```r
d <- function(x,epsilon){
  derivative = (x + epsilon - x) / epsilon
}
```

2. Evaluate your derivative function at x = 1 and x = 100000.

```r
result1 = d(1,10^-15)
result1
```

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

```r
result2 = d(100000,10^-15)
result2
```

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

3. What values did you obtain? What are the true values? Explain the reasons behind the discovered differences.

When x = 1, output 1.110223, x = 100000, outputs 0. The true values are both 1.

R uses double precision to save floating point numbers.

```r
typeof(10^-15)
```

```
## [1] "double"
```

A very small number, such as 1 plus a floating-point number with more than ten decimal places, the accuracy of R storing the result is reduced. But when a large number is added to a floating point-number with more than ten decimal places, R automatically omits the fractional part of the result.

This is a problem that arises when the R language is designed, and it is also related to its use of double-precision floating-point format.

Question 3: Variance

1. Write your own R function, myvar, to estimate the variance in this way.

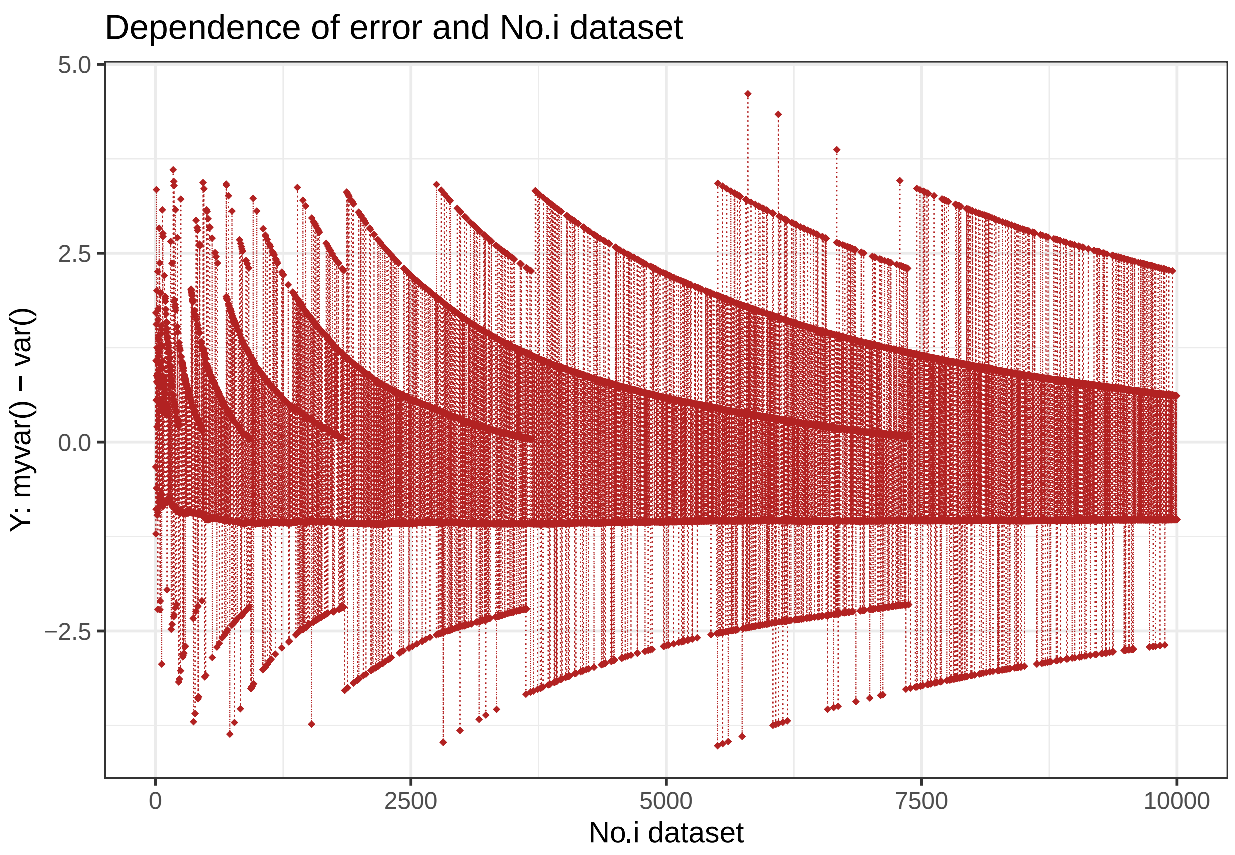
```r
myvar <- function(x) {
  (1 / (length(x) - 1))* ((sum(x ^ 2) - (1 / length(x)) * (sum(x) ^ 2)))
}
```

2. Generate a vector x = $(x_1 , . . . , x_{10000})$ with 10000 random numbers with mean 10^8 and variance 1.

```r
set.seed(1)
x = rnorm(10000, 10 ^ 8, 1)
```

3. plot

```r
library(ggplot2)
# myvar()
myvar <- function(x) {
  (1 / (length(x) - 1))* ((sum(x ^ 2) - (1 / length(x)) * (sum(x) ^ 2)))
}
# dataset
set.seed(1)
x = rnorm(10000, 10 ^ 8, 1)
# error, mavar() - var()
y = c()
for (i in 1:length(x)) {
  Y = myvar(x[1:i]) - var(x[1:i])
  y = append(y, Y)
}
# plot
table = data.frame(i = c(1:10000), Y = y)
ggplot(table, aes(x = i, y = Y)) +
  geom_point(color = "firebrick ",
             shape = "diamond",
             size = 1,
             na.rm = TRUE,) +
  geom_line(color = "firebrick ",
            linetype = "dotted",
            size = .2,
            na.rm = TRUE) +
  labs(x = "No.i dataset", y = "Y: myvar() - var()") +
  ggtitle("Dependence of error and No.i dataset") +
  theme(plot.title = element_text(hjust = 0.5))+
  theme_bw()
```



Draw conclusions from this plot

1. The plot shows a downward trend and this trend is accompanied by a cyclical phenomenon.
2. The diagram is symmetrical and the axis of symmetry is the "conspicuous straight line" in the middle.

How well does your function work?

The overall difference between myvar() method and var() method is -1.024866. This function is valid, but unstable.

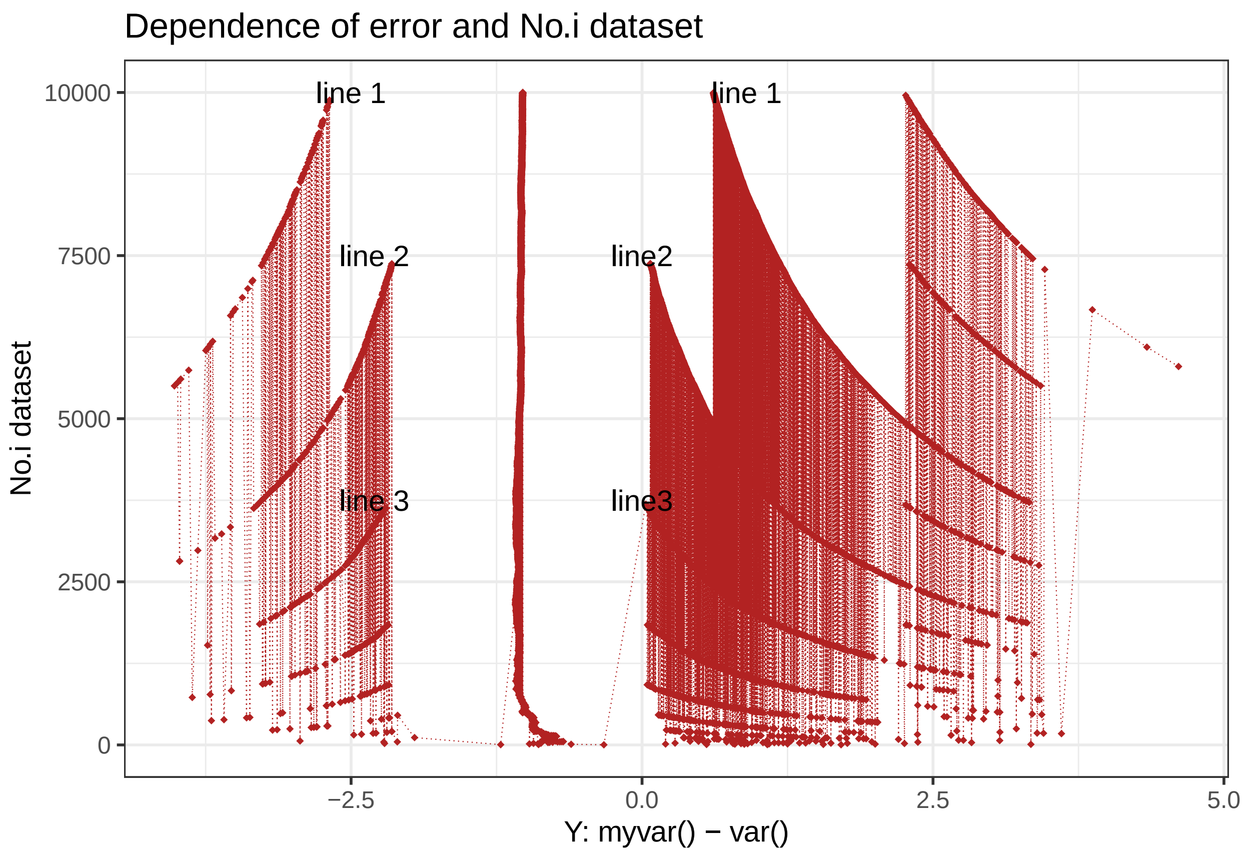
```
error = myvar(x[1:10000]) - var(x[1:10000])
error
```

```
## [1] -1.024866
```

Can you explain the behaviour?

If we rotate the plot by 90 degrees, this symmetry becomes apparent.

```
annotation <- data.frame(
  x = c(-2.5,-2.3,-2.3, 0.9,0,0),
  y = c(10000,7500,3750,10000,7500,3750),
  label = c("line 1", "line 2", "line 3","line 1","line2","line3")
)
ggplot(table, aes(y = i, x = Y)) +
  geom_point(color = "firebrick ",
             shape = "diamond",
             size = 1,
             na.rm = TRUE,) +
  geom_line(color = "firebrick ",
            linetype = "dotted",
            size = .2,
            na.rm = TRUE) +
  labs(y = "No.i dataset", x = "Y: myvar() - var()") +
  ggtitle("Dependence of error and No.i dataset") +
  theme(plot.title = element_text(hjust = 0.5))+
  theme_bw()+
  geom_text(data=annotation, aes( x=x, y=y, label=label))
```



We can find:

The symmetry axis approximates a straight line, which consists of a series of real points;

The lines 1, 2 and 3 marked in the figure are consistent with a normal distribution curve;

These curves that fit the characteristics of a normal distribution do not occur consecutively, they overlap with each other.

Explanation:

Why do the differences produced by the myvar() method and the var() method conform to a normal distribution? This is because we are processing data in the same way for these subsets, so the final difference (Y_i) will also conform to a normal distribution, such as the normal distribution curve line 1,2,3 in the graph.

1.The generation of normal distribution curves line 1,2,3 etc. and the overlap between them.

x is a data set generated by random numbers using the rnorm() function, which fits a normal distribution. For a data set (which is vector x) that is normally distributed, equally spaced sampling produces data samples that are also normally distributed.

For each subset X_i, they do not conform to the principle of equally spaced sampling, and each subset has a relatively large overlap, so that the normal distribution curves overlap in the plot, and do not appear one curve after the other.

2.Disappearance of peaks such as line 1,2,3 of the normal distribution curve.

We also found that the peaks of the normal distribution curve all disappeared because the regression properties of the sample were magnified as it was taken closer to the middle. The closer the sample is to the middle, the closer it is to the mean in the normal distribution, and the more it disappears into the regression line, along with the entire peak.

3.The creation of a symmetrical axis, can it be called a regression line?

If we calculate the error (Y_i) for the entire data set:

```
error = myvar(x[1:10000]) - var(x[1:10000])
error
```

```
## [1] -1.024866
```

The error between the two methods is -1.024866, when we calculate the variance of this data set using myvar() and var(), respectively.

This value is the mean of the normal distribution, and all subsets that fit the normal distribution have this mean, which is the reason for the creation of the axis of symmetry. If we zoom in on this line, see figure 1:

All these Y_i fluctuating up and down around the value -1.024866.

To summarise:

The cyclical downward trend mentioned earlier is actually characteristic of a normal distribution curve, generated by a series of subsets that conform to a normal distribution;

The symmetric axis is the mean of the normal distribution.

4. How can you better implement a variance estimator? Find and implement a formula that will give the same results as var()?

```
myvar2 <- function(x) {
  u = (1 / length(x)) * sum(x)
  variance = sum((x - u) ^ 2) / (length(x) - 1)
  return(variance)
}
myvar2(x)
```

```
## [1] 1.024866
```

```
var(x)
```

```
## [1] 1.024866
```

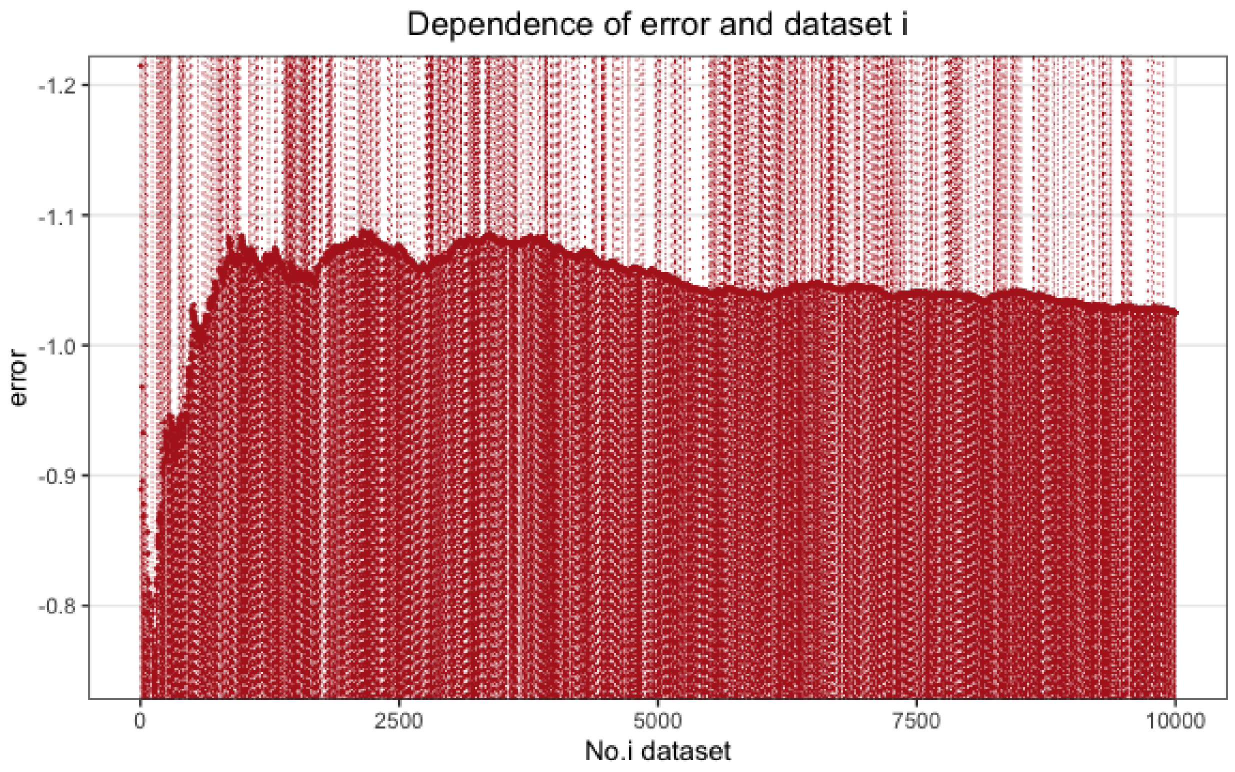
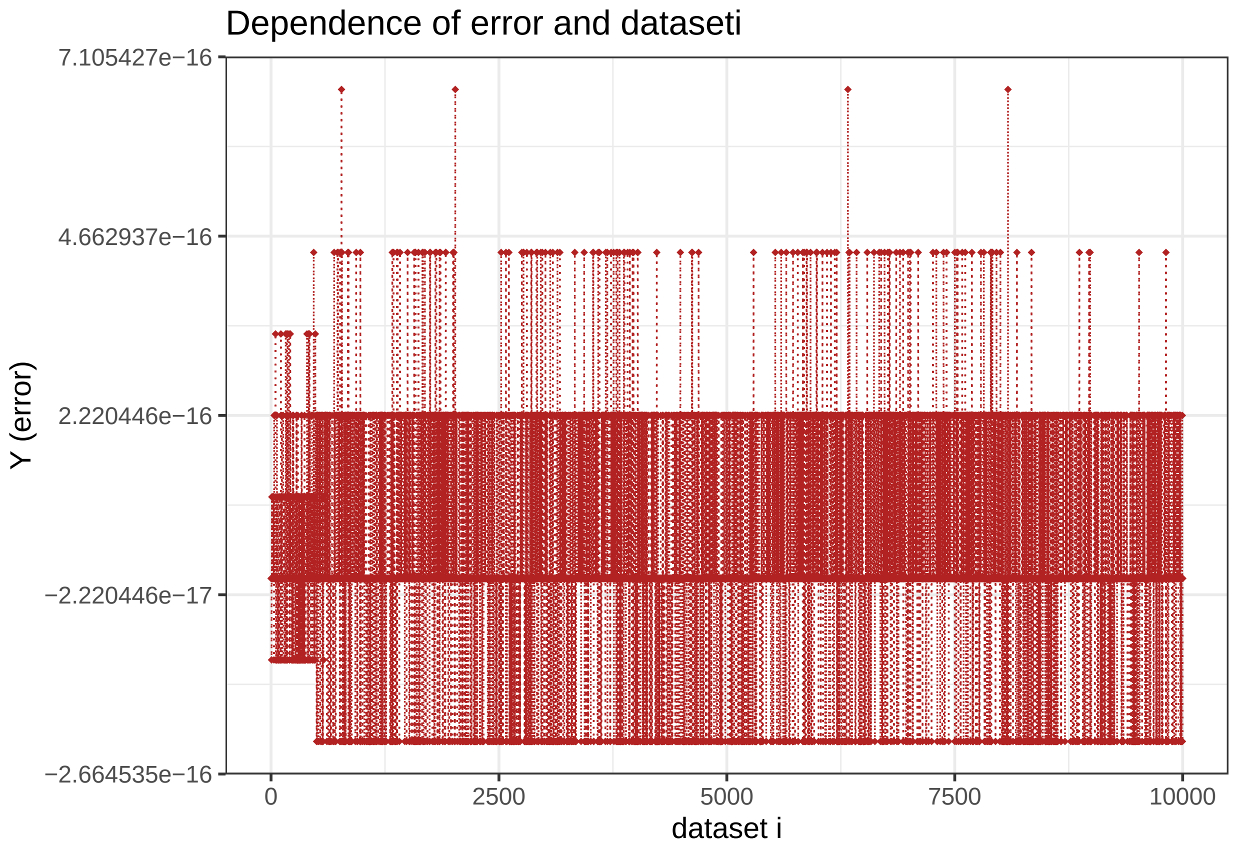


Figure 1: Zoom in.

new function

myvar2 <- function(x) { u = (1 / length(x)) * sum(x) variance = sum((x - u) ^ 2) / (length(x) - 1) return(variance) }

```
## x
set.seed(1)
x = rnorm(10000, 10 ^ 8, 1)
## y
y = c()
for (i in 1:length(x)) {
  Y = myvar2(x[1:i]) - var(x[1:i])
  y = append(y, Y)
}
i = seq(1, 10000, length.out = 10000)
table = data.frame(i = c(1:10000), Y = y)
## plot
ggplot(table, aes(x = i, y = Y)) +
  geom_point(color = "firebrick",
             shape = "diamond",
             size = 1,
             na.rm = TRUE) +
  geom_line(color = "firebrick",
            linetype = "dotted",
            size = .3,
            na.rm = TRUE) +
  labs(x = "dataset i", y = "Y (error)") +
  ggtitle("Dependence of error and dataseti") +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme_bw()
```


Dependence of error and dataseti
7.105427e−16
4.662937e−16
2.220446e−16
−2.220446e−17
−2.664535e−16
Y (error)
dataset i