MATH5835M Statistical Computing
Exercise Sheet 3 (answers)

https://www1.maths.leeds.ac.uk/~voss/2023/MATH5835M/
Jochen Voss, J.Voss@leeds.ac.uk
2023/24, semester 1

Answer 5. We have

$$\begin{aligned}
\mathbb{E}\bigl(Z_N^{\mathrm{CV}}\bigr) &= \mathbb{E}\Bigl(\frac{1}{N}\sum_{j=1}^{N}\bigl(f(X_j)-g(X_j)\bigr)+\mathbb{E}\bigl(g(X)\bigr)\Bigr) \\
&= \mathbb{E}\Bigl(\frac{1}{N}\sum_{j=1}^{N}\bigl(f(X_j)-g(X_j)\bigr)\Bigr)+\mathbb{E}\bigl(g(X)\bigr) \\
&= \frac{1}{N}\sum_{j=1}^{N}\mathbb{E}\Bigl(\bigl(f(X_j)-g(X_j)\bigr)\Bigr)+\mathbb{E}\bigl(g(X)\bigr) \\
&= \mathbb{E}\bigl(f(X)\bigr)-\mathbb{E}\bigl(g(X)\bigr)+\mathbb{E}\bigl(g(X)\bigr) \\
&= \mathbb{E}\bigl(f(X)\bigr),
\end{aligned}$$

and thus Z_N^{CV} is unbiased. For an unbiased estimator, the MSE equals the variance, and thus

$$\begin{aligned}
\mathrm{MSE}\bigl(Z_N^{\mathrm{CV}}\bigr) &= \mathrm{Var}\bigl(Z_N^{\mathrm{CV}}\bigr) \\
&= \mathrm{Var}\Bigl(\frac{1}{N}\sum_{j=1}^{N}\bigl(f(X_j)-g(X_j)\bigr)+\mathbb{E}\bigl(g(X)\bigr)\Bigr) \\
&= \mathrm{Var}\Bigl(\frac{1}{N}\sum_{j=1}^{N}\bigl(f(X_j)-g(X_j)\bigr)\Bigr) \\
&= \frac{1}{N}\,\mathrm{Var}\bigl(f(X)-g(X)\bigr),
\end{aligned}$$

using independence of the X_j as in lectures.

Answer 6. Since we need to repeat the operation of computing the cosine 20 times, we use a `for` loop in R:

```
X <- 0
for (i in 1:20) {
    X <- cos(X)
}
print(X)
```

The output of these commands is `0.7390851`.

Answer 7. We can generate the 1000 samples as follows:

```
U <- runif(1000, 0, 1)
X <- 1 / U^1.5
```

The naïve approach to generate a histogram of X results in the output shown in the top panel of figure 1:

```
hist(X, main=NULL, col="gray80", border="gray20")
```

This histogram is useless, since there is only one bar visible.

To find out what went wrong, we first have a closer look at the data:

```
> summary(X)
    Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
   1.004     1.537     2.862    92.150     7.802 21660.000
```

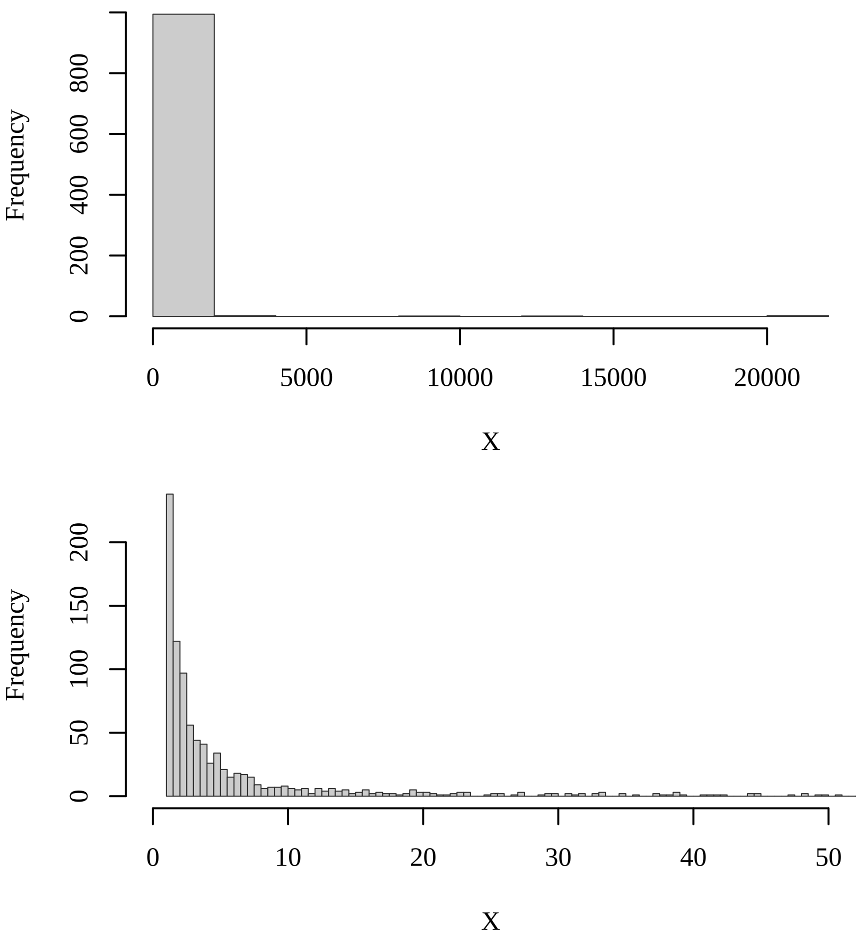

Figure 1. The two histograms from question 10. The top panel uses the automatic settings from R. The bottom panel uses our adjusted settings.

The summary output shows the problem clearly: the distribution of X is extremely skewed. The largest value (which determines the range of values on the horizontal axis of the histogram) is 21660.000, but 75% of the values are below 7.802 which explains why the leftmost bar is the only one which is clearly visible.

To work around these problems, two changes are required: First, we need to force `hist` to use more bars so that the different values in the range from 0 to around 8 can be distinguished. And secondly, we need to restrict the horizontal range of the plot to a smaller interval (omitting the very few very large values), so that the main bulk of the data becomes visible:

```
hist(X, breaks=50000, xlim=c(0, 50),
     main=NULL, col="gray80", border="gray20")
```

The resulting plot is shown in the bottom panel of figure 1

An alternative approach would be to use a logarithmic scale on the x-axis of the plot.

Answer 8. To find the mistake in the given R function, we first add a series of print commands to the function, to see at which step the function deviates from our expectations:

```
SomethingWrong <- function(x) {
  n <- length(x)
  cat("x =", x, ", n =", n, "\n")
  sum <- 0
  for (i in 1:n-1) {
    cat("i =", i, ", sum =", sum, "\n")
    sum <- sum + (x[i+1] - x[i])^2
```

```
  }
  return(sum)
}
```

This results in the following output:

```
> SomethingWrong(c(1,2,3))
x = 1 2 3 ,n = 3
i = 0 , sum = 0
i = 1 , sum =
i = 2 , sum =
numeric(0)
```

The first line of the output is what we expect: the input data is `1 2 3` and the length of the input data is `3`. The second line already shows a problem: we asked R to use the values $1, \ldots, n-1$ for `i`, but the first iteration of the loop has `i = 0`. This is the cause of the problem, since then the loop evaluates `x[0]` which is not the value we intended to use.

To find out why `i` took the value 0, we inspect the loop statement: we have `i in 1:n-1` and we know that at this point `n` equals 3. Thus the range of `i` is `1:3-1`:

```
> 1:3-1
[1] 0 1 2
> 1:2
[1] 1 2
> 1:(3-1)
[1] 1 2
```

The experiments shown above clearly point out the problem: R interprets `1:n-1` as $(1, 2, \ldots, n)-1$, *i.e.* the computer subtracts 1 from every value in the sequence `1:n`. This is not what we intended, and we can fix this by introducing brackets around `n-1`. The following version of the function fixes the error:

```
SomethingWrong <- function(x) {
  n <- length(x)
  sum <- 0
  for (i in 1:(n-1)) {
    sum <- sum + (x[i+1] - x[i])^2
  }
  return(sum)
}
```

Finally, we test the new version of the function:

```
> SomethingWrong(c(1,2,3))
[1] 2
```

Since this is the expected result, we can assume that we have correctly identified and fixed the problem.