

# Sample Answer Lab 01

---


**Handout date:** Wednesday, August 28, 2019

**Due date:** Wednesday, September 11, 2019 at the beginning of the lecture as hardcopy

*This lab counts 4 % toward your total grade*

**Objectives:** In this lab

[a] you will practice your understanding of the structure of mathematical equations and subsequently properly typesetting these equations,

[b] you will practice your  skills.

**Format of answer:** Your answers (graphs and verbal description) should be handed in as **hard-copy** in **one** document. Add a running title into the header of the document with the following information: **your name, GIS6301-Lab01** and **page numbers**. Label each answer properly starting with its task number. Maintain the sequence of questions. Format any code and computer output properly before inserting it into the document with your answer. R-code and text output need to be in a **monospaced** font (i.e., fixed-pitch font) such as Courier New so proper spacing and alignments are preserved. Excessive, but irrelevant, output will lead to a deduction of your accumulated points.

## Task 1: Equation Editor Exercise (1 point)

Comments on common mistakes:

- Underbrace should cover the whole equation instead of only sigma sign.
- "i" from 1 to n, use "i=1" instead of "i-1" under the sigma sign.

Typeset the derivation of the computational equation of the variance equation in line (7) from its definitional counterpart in line (1). Make sure that your formatting follows exactly the one given in the screen shot below. In particular, focus on [a] the change from the *italics* equation mode to the upright text mode in equation (1), [b] proper nesting of the employed mathematical templates, [c] the alignment of all equation at the "="-sign, and [d] the use of underbraces  $\underbrace{\quad}_{bottom}$  in equations (3) and (4).

You do not need to replicate the *explanations* and *line numbers* in red. Deviations from the formatting in the screen shot will lead to a loss of partial points.

$$\begin{aligned}
 S_x^2 &= \frac{1}{n-1} \cdot \sum_{i=1}^n (x_i - \bar{x})^2 \text{ with } \bar{x} = \frac{1}{n} \cdot \sum_{i=1}^n x_i \\
 &= \frac{1}{n-1} \cdot \sum_{i=1}^n (x_i^2 - 2 \cdot x_i \cdot \bar{x} + \bar{x}^2)
 \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{n-1} \cdot \sum_{i=1}^n x_i^2 - \frac{1}{n-1} \cdot \sum_{i=1}^n 2 \cdot x_i \cdot \bar{x} + \frac{1}{n-1} \cdot \underbrace{\sum_{i=1}^n \bar{x}^2}_{=n \cdot \bar{x}^2} \\
&= \frac{1}{n-1} \cdot \sum_{i=1}^n x_i^2 - \frac{1}{n-1} \cdot 2 \cdot \bar{x} \cdot \underbrace{\sum_{i=1}^n x_i}_{=n \cdot \bar{x}} + \frac{n}{n-1} \cdot \bar{x}^2 \\
&= \frac{1}{n-1} \cdot \sum_{i=1}^n x_i^2 - \frac{n}{n-1} \cdot 2 \cdot \bar{x}^2 + \frac{n}{n-1} \cdot \bar{x}^2 \\
&= \frac{1}{n-1} \cdot \sum_{i=1}^n x_i^2 - \frac{n}{n-1} \cdot \bar{x}^2 \\
&= \frac{1}{n-1} \cdot \left( \sum_{i=1}^n x_i^2 - n \cdot \bar{x}^2 \right)
\end{aligned}$$

Both formulas will be used in Task 3 to demonstrate speed and numerical stability issues of the definitional and the computational algorithms.

## Task 2: Summation rules applied in Task 1 (1 point)

### Comments on common mistakes:

- Please justify your answer

Appendix 3a in Burt, Barber & Rigby (pp 148-149) reviews the use of the **sigma** summation-notation

The transformations in Task 1 make use of some algebraic rules. Assign the relevant rules to the equation lines. The learning objective here is that you understand the algebraic transformation

List of algebraic rules:

1. None of the given rules has been applied
2.  $\sum_{i=1}^n c = n \cdot c$  with  $c$  being a constant
3.  $\sum_{i=1}^n c \cdot x_i = c \cdot \sum_{i=1}^n x_i$  with  $c$  being a constant
4.  $\sum_{i=1}^n (x_i + y_i) = \sum_{i=1}^n x_i + \sum_{i=1}^n y_i$
5. A transformation of the arithmetic mean:  $\bar{x} = \frac{1}{n} \cdot \sum_{i=1}^n x_i \Leftrightarrow n \cdot \bar{x} = \sum_{i=1}^n x_i$
6.  $a \cdot x + b \cdot x = (a + b) \cdot x$
7.  $(a - b)^2 = (a - b) \cdot (a - b) = a^2 - 2 \cdot a \cdot b + b^2$

Which algebraic rules were applied to get to from one equation to the next? Justify your answer.

Eq. (1)  $\rightarrow$  Eq. (2): rule 7 applied.

Eq. (2)  $\rightarrow$  Eq. (3): rule 4 applied, it combines the  $\Sigma$  terms.


Eq. (3)  $\rightarrow$  Eq. (4): rule 3 applied

Eq. (4)  $\rightarrow$  Eq. (5): rule 5 applied

Eq. (5)  $\rightarrow$  Eq. (6): rule 1 applied, it only adds and subtracts a term, do not apply any rule.

Eq. (6)  $\rightarrow$  Eq. (7): rule 6 applied, term  $\frac{1}{n-1}$  is distributed to the two terms in parentheses.

### Task 3: Computational Properties of Different Implementations of the Sample Variance Estimator (2 points)

You find below the  code for calculating the *definitional expression* and the *computational expression* of the variance including some code which tests these functions:

```
rm(list = ls(all = TRUE))
library(compiler)
enableJIT(0)
options(digits = 6)

VarComExpress <- function(x) {

  ## Computational expression of teh sample variance
  x<- as.vector(x)
  n<- length(x)
  meanLoop <- 0 ; squareLoop <- 0

  for (i in 1:n) {

    meanLoop <- meanLoop + x[i]
    squareLoop <- squareLoop + x[i]^2

  }
  xMean <- meanLoop / n
  xVar <- (squareLoop - n * xMean ^2) / (n-1)
  return (xVar)

} #end:: VarComExpress

VarDefExpress<- function(x) {

  ## Definitional expression of the sample variance

  n<- length(x)
  meanLoop <- 0
  VarLoop <- 0
```

```

    for (i in 1:n) {
      meanLoop <- meanLoop + x[i]
    }
    xMean <- meanLoop /n
    for (i in 1:n) {
      VarLoop <- VarLoop + (x[i]- xMean)^2
    }
    xVar<- VarLoop/(n-1)
    return(xVar)
} #end:: VarDefExpress

## generate random vector
inputMean <- 100
nCount <- 1e7
x <- rnorm (nCount, mean = inputMean, sd = 1)
print(object.size(x), units = "Mb")

## Plot histogram with density curve
summary(x, digits =12)
hist(x, freq = FALSE)
lines(density(x), lwd=2, col="red")

## evaluate definitional, computational and R's internal variance functions
timeSys <- system.time( varSys <- var(x) ) [3]
timeDef <- system.time( varDef <- VarDefExpress(x) ) [3]
timeCom <- system.time( varCom <- VarComExpress(x) ) [3]

## Report results
data.frame("Method" = c("Sys:", "Def:", "Com:"),
          "Var"=c(varSys, varDef, varCom),
          "Time"=c(timeSys, timeDef, timeCom))

```

[a] Enter this code properly formatted into an  script. Show your properly formatted script. (1 point)

You can study loops, control statements and user functions in greater depth by looking at Lander's *Chapter 8: Writing R functions*, *Chapter 9: Control Statements* and *Chapter 10: Loops, the Un-R Way to Iterate*.

[b] Why is it required to initialize the variables **meanLoop**, **squareLoop** and **varLoop** with zero in the functions **VarDefExpress ( )** and **VarComExpress ( )**? (0.1 points)

In the for-loops the terms **varLoop**, **meanLoop** and **squareLoop** are computed **recursively** at each iteration by adding new numbers to the previous sums of **varLoop**, **meanLoop** and **squareLoop**. Therefore, at the very first iteration the terms need be initialized by the neutral value for summations, which is zero. Otherwise, what ever meaningless numbers are in the memory cell associate with **varLoop**, **meanLoop** and **squareLoop** would become part of the summations.

[c] What is the **rnorm ( )** function doing? (0.1 points)

`rnorm()` is to generate random values that follow a normal distribution with specified mean and variance. Due to the nature of randomness, everyone will get different random numbers

[d] What is the `system.time()` function doing? (0.1 point)

`System.time()` is to return the CPU time used when a sequence of code executes. That time is decomposed into the time used by the operating system to perform general tasks and the time used by the R code.

[e] Run the script once with the variable `inputMean` set to 100 (line 43) and report the run times and calculated variances of all three variance functions in a table. (0.1 points)

Tips:

1. The variance of both definitional expression and computational expression should be closer to one.
2. The definitional expression should spend more time than the computational expression.

Method	Var	Time
Sys:	1.00022	0.03
Def:	1.00022	10.58
Com:	1.00022	8.13

[f] Run the script again with the variable `inputMean` set to 100,000,000 (line 43) and report the run times and calculated variances of all three variance functions in a table. (0.1 points)

Tips:

1. As mean increase, variance of computational expression deviates from one substantially
2. The definitional expression should spend more time than the computational expression.

Method	Var	Time
Sys:	1.00046	0.03
Def:	1.00046	9.98
Com:	1457.94022	8.08

[g] Why will the *computational expression* have a speed advantage? Think in terms of how many times the functions have to pass over the data. (0.25 point)

In the definitional expression, we need to pass through the dataset twice, once to calculate iteratively the mean and subsequently to iteratively calculate the sum of the squared differences. However, the computational expression only needs to pass through the data once to simultaneously calculate the sum and the sum of squares. Because of the higher number of data passes in the definitional variance function it takes longer than the computational expression.

[h] In the *computational expression* both variables `xMean`, which was derived from `meanLoop` in line 16, and `squareLoop` are of different magnitudes on the floating point scale. Why does this have the potential to lead to rounding errors when combining both variables in line 17? (0.25 point)

The culprit is the large input numbers, which - once squared - induce a numerical scale shift. Each scale shift leads to rounding errors because a scale with a larger exponent loses numerical accuracy. These rounding errors accumulate in the summation unless the errors are properly balanced. Therefore, there will be a substantial difference in rounding errors between the terms  $\sum_{i=1}^n x_i^2$  and  $\bar{x}^2$ . The term  $\bar{x}^2$  only experience one rounding error.