509 HW1

Zheng Cui

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
1 + 1
```

```
[1] 2
```

You can add options to executable code like this

```
[1] 4
```

The `echo: false` option disables the printing of code (only output is displayed).

1.1 When you are done with your homework, always turn in both the source Quarto and the compiled HTML or PDF files.

2.1 Write a function f1(x, y) to subtract numeric vector y from numeric vector x into a third vector z using a for-loop. Return z at the end of the function.

```
#define f1(x,y) using a for-loop
f1 <- function(x, y) {
  z <- numeric(length(x))

  for (i in 1:length(x)) {
    z[i] <- x[i] - y[i]
  }
  return(z)
}

#input the x and y vectors
x <- c(1, 2, 3, 4, 5)
y <- c(2, 4, 6, 8, 10)

z <- f1(x, y)  # Using the f1 function

print(z)     # Print z
```

```
[1] -1 -2 -3 -4 -5
```

2.2 Write a function f2(x, y) to achieve the same goal using a vectorized operation without a loop.

```
f2 <- function(x,y){
  z <- x - y      #Using vector elements minus the vector y element
  return(z)
}

#input the x and y vectors
x <- c(1, 2, 3, 4, 5)
y <- c(2, 4, 6, 8, 10)

z <- f2(x, y) # Using the f2 function
```

```r
print(z)    # Print z
```

```
[1] -1 -2 -3 -4 -5
```

2.3 Use what you have learned in class to create two vectors of variable lengths, such as 1,000,000, as input to the two functions. Record the runtime of the function by system.time(f1(x, y))[1]. Examine if the two functions return the same vector using function testthat::expect_equal().

```r
library(testthat)
#  #for loop method
f1 <- function(x, y) {
  z <- numeric(length(x))

  for (i in 1:length(x)) {
    z[i] <- x[i] - y[i]
  }
  return(z)
}
#vectorized operation
f2 <- function(x,y){
  z <- x - y
  return(z)
}
#  two vectors   length 1,000,000.
x <- c(1:1000000 )
y <- c(2:1000001)

# Record the runtime of the `for loop method.
runtime_f1 <- system.time(f1(x, y))[1] * 1000


# Record the runtime of the `vectorized operation.
runtime_f2 <- system.time(f2(x, y))[1] * 1000

expect_equal(f1(x, y), f2(x, y))

cat("Runtime of f1:", runtime_f1, "milliseconds\n")
```

```
Runtime of f1: 60 milliseconds
```

```r
cat("Runtime of f2:", runtime_f2, "milliseconds\n")
```

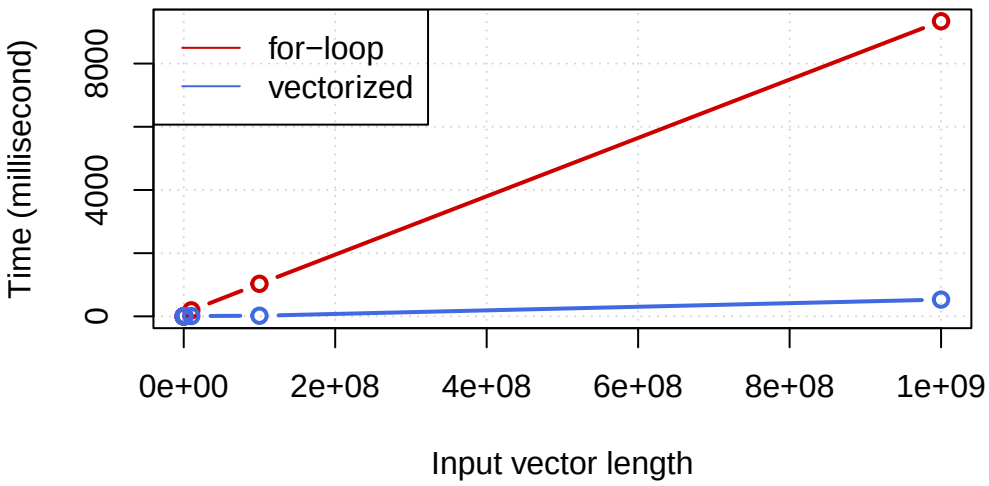
```
Runtime of f2: 0 milliseconds
```

2.4 Use the following plot function to visualize the runtime difference of the two functions on input pairs of varying lengths. Here, ns contains the input lengths of different trials; t.loop is a vector of runtime for the for-loop solution for each trial; t.vectorized is a vector of runtime for the vectorized solution for each trial.

```r
plot.runtime <- function(ns, t.loop, t.vectorized)
{
  plot(ns, t.loop, ylim=c(0, max(t.loop)), col="red3",
       type="b", panel.first=grid(), lwd=2,
       xlab="Input vector length",
       ylab="Time (millisecond)")
  lines(ns, t.vectorized, col="royalblue", type="b", lwd=2)
  legend("topleft", c("for-loop", "vectorized"),
         col=c("red3", "royalblue"), lwd=1)
}

# test data
input_lengths <- c(10, 100, 1000, 10000, 100000, 1000000, 10000000, 100000000, 1000000000)
runtime_for_loop <- c(0, 0, 0, 0, 0, 10, 190, 1030, 9340)   # Runtime of the for loop solu
runtime_vectorized <- c(0, 0, 0, 0, 0, 0, 10, 20, 530)     # Run time of vectorized soluti

# Using plot.runtime
plot.runtime(input_lengths, runtime_for_loop, runtime_vectorized)
```



2.5 Project the runtime that would be taken for vectors of 10 billion elements for each method on your computer. How much time will you have saved using the vectorized operation instead of a for-loop?

Answer: As the figure above shows, 10 billion elements may take a long time, which may take more than 100s in the for-loop, while vectorized operation may take more than 10s (depending on the speed of the computer), The for-loop may take more than 10 times as long anyway, so you can save more than 10 times as much time using the vectorized operation.

2.6 Conclude what you will do in the future when working on two vectors. Will it generalize to multiplying two vectors?

Answer: Based on the test data and the diagram, I will use the vectorized operation to handle the two vectors. The time required for a for-loop is too long, so I think it will definitely generalize to multiplying two vectors.

3.1 Please explain in your own words how the GRanges class is defined.

Answer:GRanges is a data structure for storing genomic intervals and their associated annotations. It is widely used in bioinformatics research. The GRanges class is a container for the genomic locations and their associated annotations. GenomicRanges are the storage method

for genome coordinates used in Bioconductor projects. Genomicranges are based on IRanges, Currently provides support for BSgenome, Rsamtools, ShortRead, rtracklayer, GenomicFeatures, GenomicAlignments, VariantAnnotations, etc A GRanges object is a vector, where each element represents a genomic interval and its associated annotations. A genomic interval is a contiguous sequence of any length on the genome. This can include genes, transcripts, repetitive sequences, SNPs, etc.

https://bioconductor.org/packages/release/bioc/vignettes/GenomicRanges/inst/doc/GenomicRangesIntroduct

3.2 Give two or more types of biological data that involve intervals. Describe the biological meaning of each type and why their data involve intervals.

Answer: ChIP-seq data is a type of genomic data that measures the binding of proteins to DNA. The intervals in ChIP-seq data are the protein-binding sites. The biological meaning of ChIP-seq data is that it can be used to identify the genes and regulatory elements that are regulated by specific proteins.

ChIP-seq data involves intervals because it is based on the immunoprecipitation of DNA fragments that are bound to a specific protein. The DNA fragments are then sequenced, and the resulting data can be used to identify the genomic intervals where the protein is bound.

REFERENCES: https://en.wikipedia.org/wiki/ChIP_sequencing

Gene expression data: This data measures the levels of RNA transcripts for genes in a cell or tissue. The intervals in gene expression data are the genes themselves. The biological meaning of gene expression data is that it can be used to understand how genes are regulated and how they are expressed in different cell types and tissues.

REFERENCES: https://en.wikipedia.org/wiki/Gene_expression