Evolutionary Genetics

LV 25600-01 | Lecture with exercises | 4KP

HS2020

Loading R packages: What is the difference between the following R commands?

```
(A) install.packages("dplyr")

(B) load("dplyr")

(C) source("dplyr")

(D) require("dplyr")

(E) library("dplyr")
```

Bioinformatics - R

```
# Install R packages from repositories (or local files)

install.packages("dplyr")

install.packages(c("dplyr", "plyr")

install.packages("lattice", repos="http://cran.r-project.org")

install.packages("local.R.Package", lib.loc="/my/local/R/library")
```

Bioinformatics - R

```
## Save&Reload datasets written with the function save

# save object x
save(x, file = "test.RData")

# save all objects
save(list = ls(all = TRUE), file= "all.rda")

# load dataset(s)
load("test.RData", envir = parent.frame(), verbose = FALSE)
```

save() writes an external representation of objects to the specified file. The objects can be read back from the file at a later date by using the function **load**.

save.image() is just a short-cut for save my current workspace,
i.e., save(list = ls(all.names = TRUE), file = ".RData", envir = .GlobalEnv).

It is good practice to create separate **R scripts** that you can use to **store sets of related functions**. You can then call those functions using the **source() function**, at the top of your script. R will then load those functions into memory and you can use them!

```
source("my_awesome_functions.R")
```

The **library()** and **require()** can be used to attach and **load add-on packages which are already installed**.

The **library**() by default returns an error if the requested package does not exist.

The **require**() is designed to be used **inside functions** as it gives a warning message and returns a logical value say, FALSE if the requested package is not found and TRUE if the package is loaded.

It is better to use the library() as it gives the error message if the package is not found during the package loading time. This will indeed avoid unnecessary headaches of tracking down the errors caused while attempting to use the library routines which are not installed.

What should you do first when you are starting a new R session?

First

```
## clean/reset environment
rm(list = ls())
```

Last

```
## Session-Log
session.log <- sessionInfo()
write("session.log")
```

What is the difference between the two R commands?

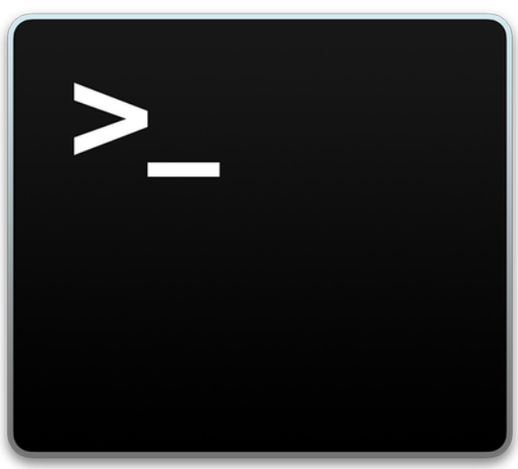
```
(A) brewer.pal.info["Blues",]
(B) RColorBrewer::brewer.pal.info["Blues",]
```

What is the difference between the two R commands?

```
# Use a particular function from a specific package
RColorBrewer::brewer.pal.info["Blues",]
# The function work without loading the packege
```

```
# The function only works if the package is installed and
loaded
library("RColorBrewer")
brewer.pal.info["Blues",]
```

Bioinformatics - R

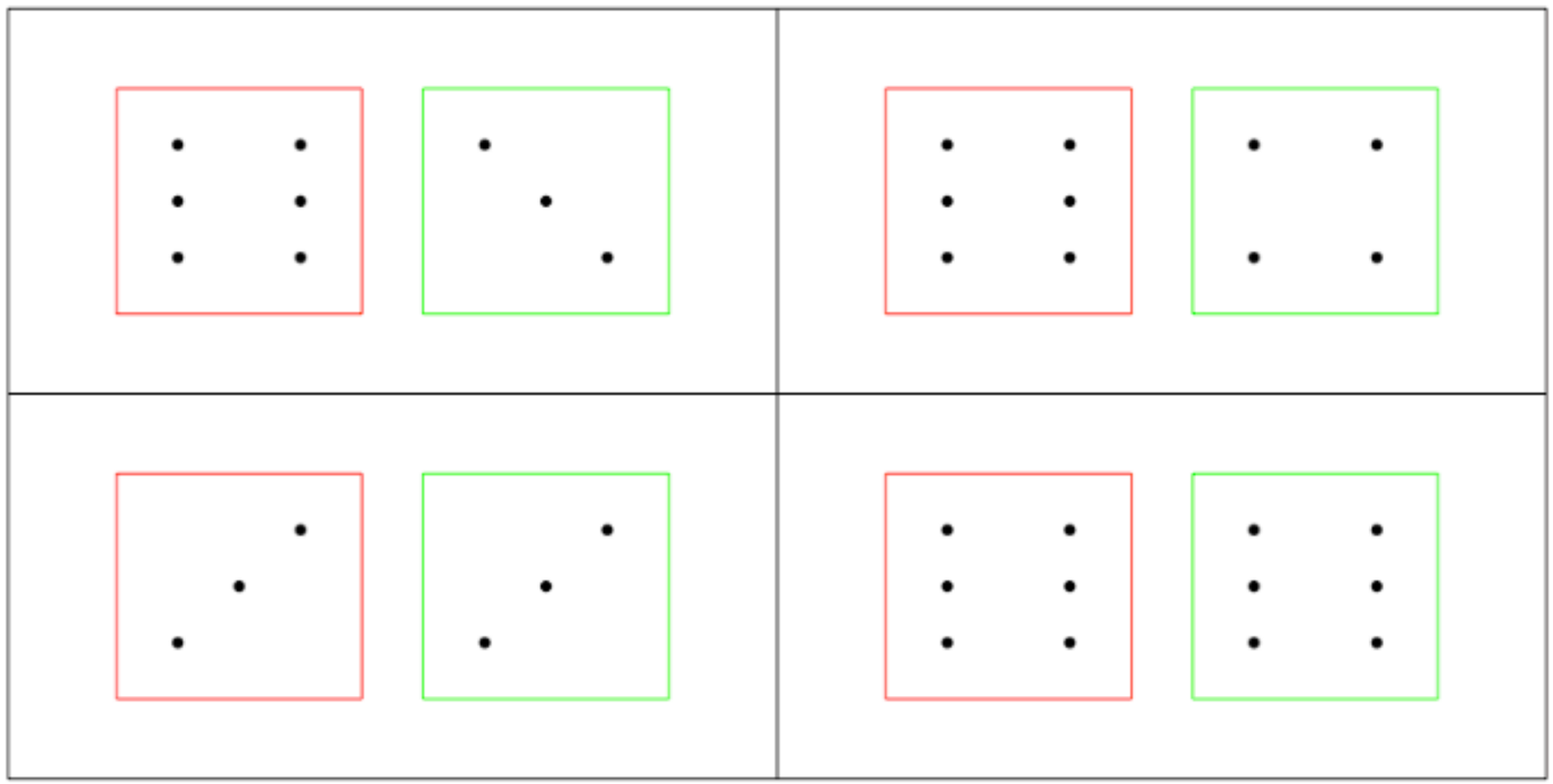


Rscript

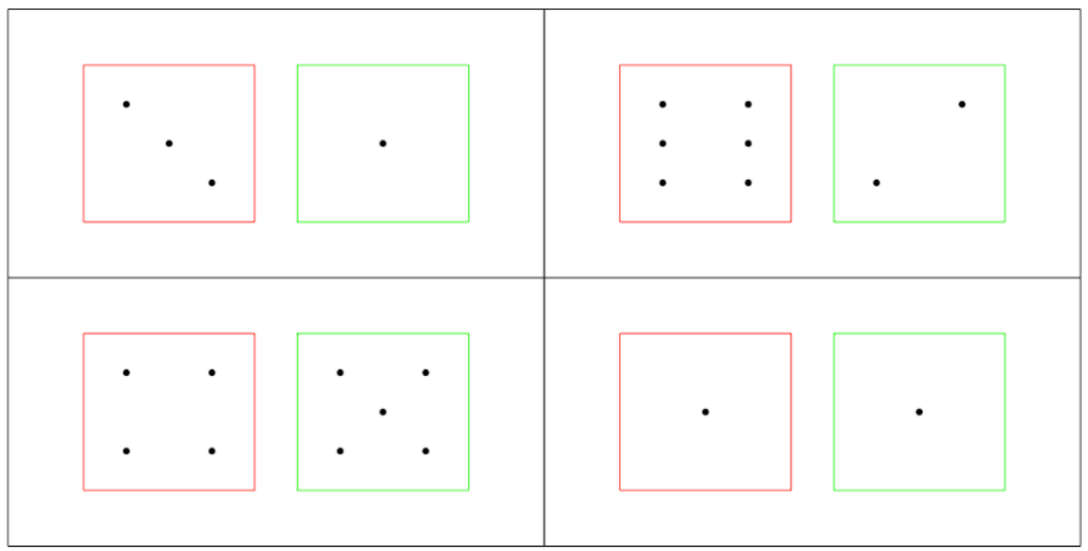
```
#!/usr/bin/env Rscript
# usage: Rscript Boxplot.R 'table.txt'

## Define Import
args = commandArgs(trailingOnly = TRUE)

## Load data
d <- read.table(args[1], sep = ";", header = TRUE)

pdf(OUT, paper = "a4")
  boxplot(d$length ~ d$groups, names = d$groups)
dev.off()

## END
print(paste(date(), "Boxplot finished", sep = " "))
```

What can I do to make random objects reproducable in R?

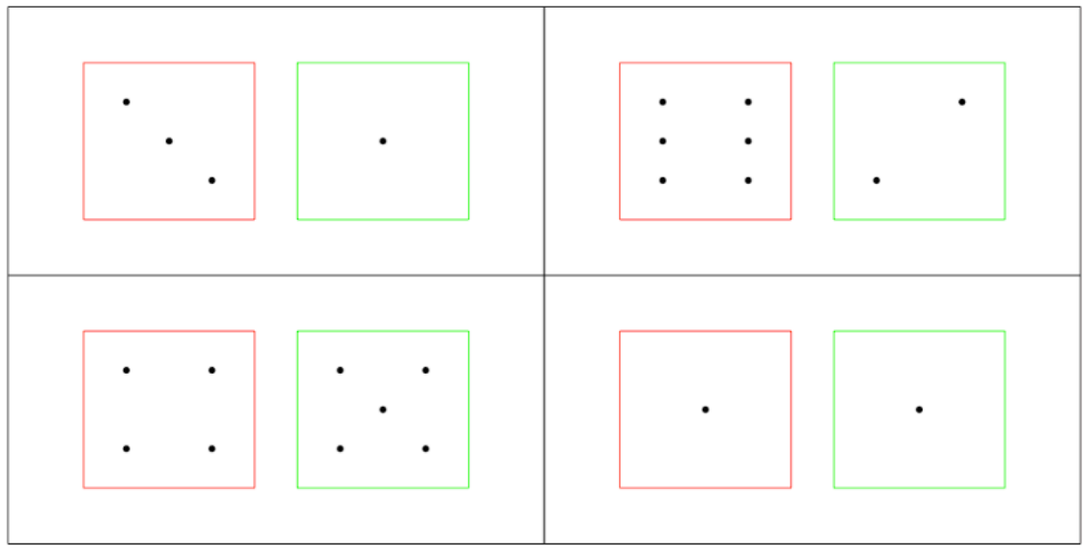
```
TeachingDemos:dice(4, 2, plot.it = TRUE)
```



Bioinformatics - R



```
set.seed(123)
dice(4, 2, plot.it = TRUE)
```



```
set.seed(123)
dice(4, 2, plot.it = TRUE)
```

What can I do to make random objects reproducable in R?

```
# Generate random number(s)
> rnorm(3)
# -0.2141846  0.2189062  0.2942180
> rnorm(3)
#  1.4912150 -1.4957131 -0.5814747
```

```
# Generate random number(s)
set.seed(191029); rnorm(3)
# -0.8421856 -0.4601290  0.9407093

# Next set using the same seed
set.seed(191029); rnorm(3)
# -0.8421856 -0.4601290  0.9407093
```

Nonparametric Resampling

bootstrap(data, mean)

```
Call:
bootstrap(data = CLEC, statistic = mean, seed = 19)
Replications: 10000

Summary Statistics:
     Observed       SE     Mean         Bias
mean 16.50913 3.96866 16.54073 0.03160109
```

bootstrap(data, mean)

```
Call:
bootstrap(data = CLEC, statistic = mean)
Replications: 10000

Summary Statistics:
     Observed        SE     Mean          Bias
mean 16.50913 3.985572 16.47043 -0.03869548
```

Nonparametric Resampling

```
bootstrap(data, mean, seed = 191029)

Call:
bootstrap(data = CLEC, statistic = mean, seed = 191029)
Replications: 10000

Summary Statistics:
     Observed        SE     Mean         Bias
mean 16.50913 3.957816 16.47054 -0.03859261
```


```
bootstrap(data, mean, seed = 191029)

Call:
bootstrap(data = CLEC, statistic = mean, seed = 191029)
Replications: 10000

Summary Statistics:
     Observed        SE     Mean         Bias
mean 16.50913 3.957816 16.47054 -0.03859261
```

```
x <- matrix( rnorm( 50000 * 900),
       nrow = 50000,
       ncol =   900)

t(x) %*% x    # version A

crossprod(x) # version B
```

```
system.time(t(x) %*% x)
   user  system elapsed
 27.053   0.148  27.277
```

```
system.time(crossprod(x))
   user  system elapsed
 21.166   0.020  21.209
```

Δ 6.07s

Bioinformatics - R

```
ex1 <- function(n) {
  for(i in 1:n) {
  x <- mean(rt(100000, df = 4))
  cat(paste("...", i, sep = ""))
  }
  print("END")
}

ex2 <- function(n) {
  for(i in 1:n) {
    d <- rt(100000, df = 4)
    x <- mean(d)
    cat(paste("...", i, sep = ""))
  }
  print("END")
}

system.time(ex1(500))
system.time(ex2(500))
```

```
n=100: -0.014s
n=200: -0.056s
n=300: -0.073s
n=400: -0.083s
n=500: -0.157s
```

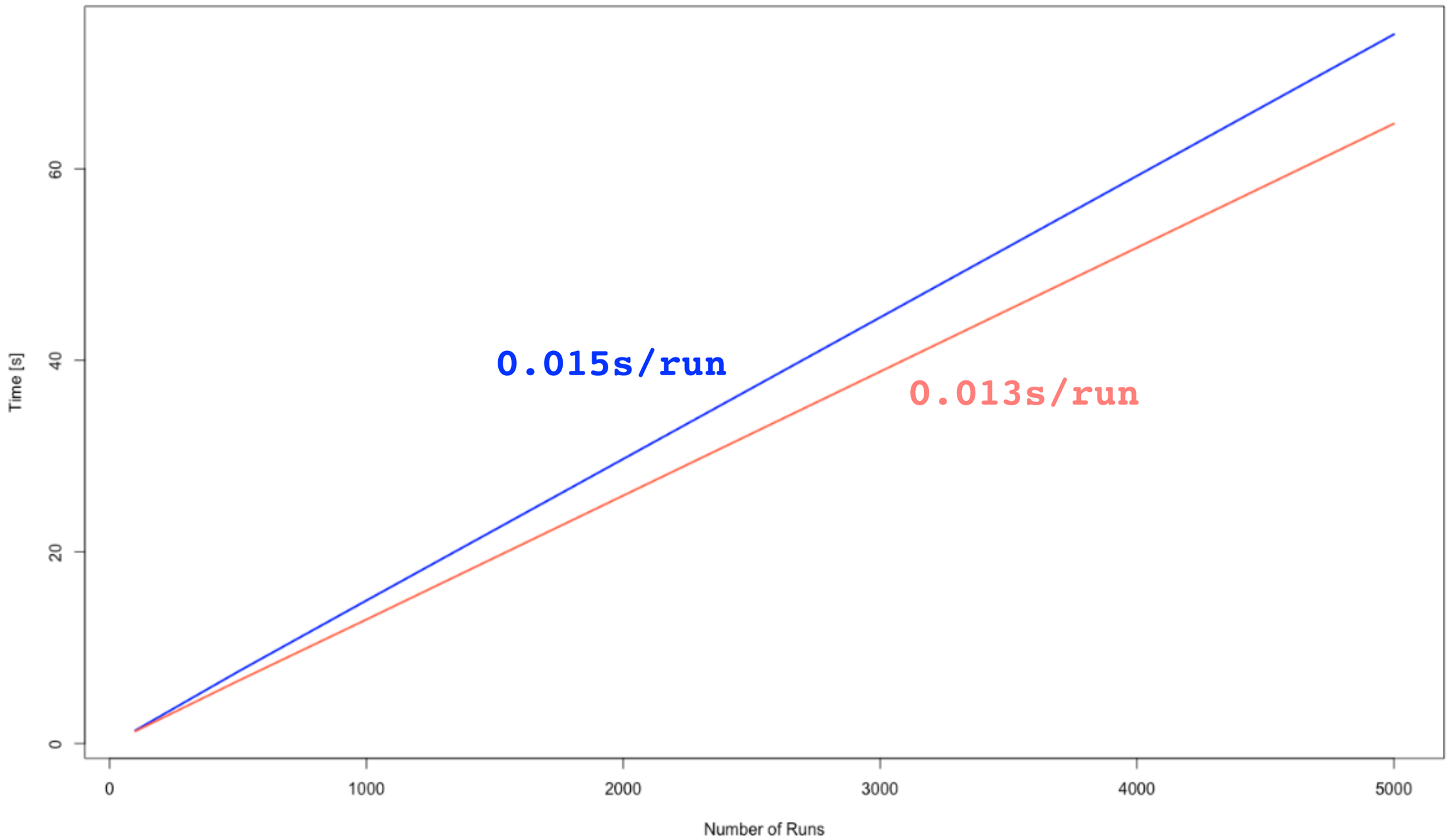
Bioinformatics - R

```
ex1 <- function(n) {
  for(i in 1:n) {
  x <- mean(rt(100000, df = 4))
  cat(paste("...", i, sep = ""))
  }
  print("END")
}
```

```
ex3 <- function(n) {
  for(i in 1:n) {
    x <- mean(rt(100000, df = 4))
  }
}

system.time(ex1(500))
system.time(ex3(500))
```

```
n= 100: -0.069s
n= 500: -0.649s
n=1000: -1.771s
```

Bioinformatics - R



Bioinformatics - R

Task: Write a function for each solution and meassure the execution time.

```
# Solution A (loop)
n <- 10^8
A <- 0
for (j in c(1:n)) {A <- j + A}
A
```

```
# Solution B (function)
n <- 10^8
sum(1:n)
```

Bioinformatics - R

```
ex1 <- function(n) {
  A <- 0
  for (j in c(1:n)) {A <- j + A}
  print(A)
}

ex2 <- function(n) {
  print(sum(1:n))
}

system.time(ex1(10^8)); system.time(ex2(10^8))
```

```
[1] 5e+15
   user  system elapsed
  2.237   0.203   2.447
[1] 5e+15
   user  system elapsed
  0.001   0.000   0.000
```