Exercises - Class 4: Functions

Sander Bossier

2023-10-22

1.

Harmonic numbers are sums of the reciprocals of the positive integers. More specifically, the n'th harmonic number is:

$$\sum_{i=1}^{n} \frac{1}{i}$$

Create a function `Harmonic(n)` which computes the nth harmonic number. The body of this function can consist of a single line. An example is:

```
Harmonic <- function(n) sum(1 / (1:n))

Harmonic(5)
```

```
## [1] 2.283333
```

2.

Harmonic numbers are recursive:

$$H_n = H_{n-1} + \frac{1}{n}$$

Write a recursive function rHarmonic(n) which computes the nth harmonic number recursively (in other words, the result should be the same as in the previous exercise).

```
rHarmonic <- function(n) {
  if (n == 1) {
    1
  }
  else{
    rHarmonic(n - 1) + (1 / n)
  }
}
rHarmonic(5)
```

```
## [1] 2.283333
```

3.

Compare the computation time of `Harmonic(n)` and `rHarmonic(n)` for the 1000th harmonic number (using R's core timing function).

```
system.time(
  Harmonic(1000)
)
```

```
##    user  system elapsed
##       0       0       0
```

```
system.time(
  rHarmonic(1000)
)
```

```
##    user  system elapsed
##    0.00    0.00    0.01
```

```
system.time(
  Harmonic(1e6)
)
```

```
##    user  system elapsed
##    0.00    0.00    0.02
```

4.

Create a binary infix operator **%fill%** which replaces all **NA** values in the vector on the left-hand side by the value specified on the right-hand side.

```
`%fill%` <- function(vec, value) {
  vec[is.na(vec)] <- value # or
  #replace(vec, is.na(vec1), value)
  vec
}

vec1 <- c(5, 55, NA, -9, -99, NA, NA, -5, -9, 999)
vec2 <- vec1 %fill% 0
vec2
```

```
##  [1]   5  55   0  -9 -99   0   0  -5  -9 999
```

5.

Create a replacement function fill() which replaces all NA values in a vector by the value specified on the right-hand side.

```
`fill<-` <- function(vec, value) {
  vec[is.na(vec)] <- value
  vec
}

vec1 <- c(5, 55, NA, -9, -99, NA, NA, -5, -9, 999)
fill(vec1) <- 0
vec1
```

```
##  [1]   5  55   0  -9 -99   0   0  -5  -9 999
```

6.

Compare the computation time of the %fill% operator and the fill() replacement function on the following large vector (you can copy-paste these two lines):

```
library(microbenchmark)
set.seed(123)
vec3 <- sample(c(5, -9, NA), size = 1e6, replace = TRUE)
```

Do this using both R's core function and the package microbenchmark (replace the NA's by 0).

```
system.time(
  tmp <- vec3 %fill% 0
)
```

```
##    user  system elapsed
##    0.00    0.00    0.03
```

```
system.time(
  fill(vec3) <- 0
)
```

```
##    user  system elapsed
##    0.00    0.00    0.02
```

```
microbenchmark(
  tmp <- vec3 %fill% 0,
  unit = "us"
)
```

```
## Unit: microseconds
##                  expr      min       lq     mean median       uq      max neval
##  tmp <- vec3 %fill% 0 3545.801 3722.901 4869.805 4041.9 5133.952 9220.201   100
```

```
microbenchmark(
  fill(vec3) <- 0,
  unit = "us"
)
```

```
## Unit: microseconds
##             expr      min       lq     mean   median       uq     max neval
##  fill(vec3) <- 0 3551.102 3719.851 4986.207 4208.801 5252.801 10261.7   100
```

7.

Create two functions **`full1()`** and **`full2()`** which both extract the non-NA elements from a vector. One of both functions needs to make use of the functional `Filter()` (see last week). Compare their computation times on the `vec3` object from the previous exercises. You can use both R's core function and the **microbenchmark** package but for the latter you can specify a number of `times` <= 100.

Note: There is even a way to combine the `Filter()` function with the `Negate()` function. If you manage to find that one, then you can compare all three functions.