Case Study 1

AKSTA Statistical Computing

The .Rmd and .html (alternatively the .pdf) should be uploaded in TUWEL by the deadline. Refrain from using explanatory comments in the R code chunks but write them as text instead. Points will be deducted if the submitted file is not in a decent form.

1. Ratio of Fibonacci numbers

a.

Write two different R functions which return the sequence $r_i = F_{i+1}/F_i$ for $i = 1, \ldots, n$ where F_i is the ith Fibonacci number, once using `for` and once using `while`.

b.

Benchmark the two functions for $n = 200$ and $n = 2000$ (you can use package **microbenchmark** or package **bench** for this purpose). Which function is faster?

c.

Plot the sequence for $n = 100$. For which value n it starts to converge? What is the number that it converges to?

2. Gamma function

a.

Write a function to compute the following for n a positive integer using the `gamma` function in base R.

$$\rho_n = \frac{\Gamma((n-1)/2)}{\Gamma(1/2)\Gamma((n-2)/2)}$$

b.

Try $n = 2000$. What do you observe? Why do you think the observed behavior happens?

c.

Write an implementation which can also deal with large values of $n > 1000$.

d.

Plot $\rho_n/\sqrt{n}$ for different values of n. Can you guess the limit of $\rho_n/\sqrt{n}$ for $n \to \infty$?

3. The golden ratio

Two positive numbers x and y with $x > y$ are said to be in the golden ratio if the ratio between the larger number and the smaller number is the same as the ratio between their sum and the larger number:

$$\frac{x}{y} = \frac{x+y}{x}$$

The golden ratio $\Phi = x/y$ can be computed as a solution to $\Phi^2 - \Phi - 1 = 0$ and is

$$\Phi = \frac{\sqrt{5}+1}{2}$$

and it satisfies the Fibonacci-like relationship:

$$\Phi^{n+1} = \Phi^n + \Phi^{n-1}$$

a.

Write an R function which computes Φ^{n+1} using the recursion above (go up to $n = 1000$) .

b.

Write a function which computes this Φ^{n+1} by simply using the power operator `^`.

c.

Use once `==` and once `all.equal` to compare Φ^{n+1} obtained in a. vs the one obtained in b. for $n = 12, 60, 120, 300$. What do you observe? If there are any differences, what can be the reason?

4. Game of craps

A variation of game of craps is played as follows. The player rolls a die and if they roll a 6, they immediately wins. If a 1 is rolled, they immediately loses. If a 2, 3, 4, or 5 is rolled, then this number becomes their "point number.'' The player then has up to three more rolls of the die to roll their point number again. If they do, they win otherwise they lose. Write a program in R to simulate a game of craps. Explain the steps of your program. If the code is not explained, no points will be earned.

5. Readable and efficient code

Read over the code below.

a.

Explain (in text) what the code does.

b.

Explain (in text) what you would change to make the code more readable.

c.

Change the code according to a. and wrap it in a function. This function should have at most 10 lines (without adding commands to more lines such as `x <- 1; y <- 2`. Such commands will count as 2 lines!). Check that the function called on the same input outputs the same as the provided code.

```r
set.seed(1)
x <- rnorm(1000)
y <- 2 + x + rnorm(1000)
df <- data.frame(x, y)

cat("Step", 1, "\n")
fit1 <- lm(y ~ x, data = df[-(1:250),])
p1 <- predict(fit1, newdata = df[(1:250),])
r <- sqrt(mean((p1 - df[(1:250),"y"])^2))

cat("Step", 2, "\n")
fit2 <- lm(y ~ x, data = df[-(251:500),])
p2 <- predict(fit2, newdata = df[(251:500),])
r <- c(r, sqrt(mean((p2 - df[(251:500),"y"])^2)))

cat("Step", 3, "\n")
fit3 <- lm(y ~ x, data = df[-(501:750),])
p3 <- predict(fit3, newdata = df[(501:750),])
r <- c(r, sqrt(mean((p3 - df[(501:750),"y"])^2)))

cat("Step", 4, "\n")
fit4 <- lm(y ~ x, data = df[-(751:1000),])
p4 <- predict(fit4, newdata = df[(751:1000),])
r <- c(r, sqrt(mean((p4 - df[(751:1000),"y"])^2)))
r
```

6. Measuring and improving performance

Have a look at the code of the function below. It is a function for performing a Kruskal Wallis test, a robust non-parametric method for testing whether samples come from the same distribution. (Note: we assume no missing values are present in `x`).

```r
kwtest <- function (x, g, ...)
{
    if (is.list(x)) {
        if (length(x) < 2L)
            stop("'x' must be a list with at least 2 elements")
        if (!missing(g))
            warning("'x' is a list, so ignoring argument 'g'")
        if (!all(sapply(x, is.numeric)))
            warning("some elements of 'x' are not numeric and will be coerced to numeric")
        k <- length(x)
        l <- lengths(x)
        if (any(l == 0L))
            stop("all groups must contain data")
        g <- factor(rep.int(seq_len(k), l))
        x <- unlist(x)
    }
    else {
        if (length(x) != length(g))
            stop("'x' and 'g' must have the same length")
        g <- factor(g)
        k <- nlevels(g)
```

```
        if (k < 2L)
            stop("all observations are in the same group")
    }
    n <- length(x)
    if (n < 2L)
        stop("not enough observations")
    r <- rank(x)
    TIES <- table(x)
    STATISTIC <- sum(tapply(r, g, sum)^2/tapply(r, g, length))
    STATISTIC <- ((12 * STATISTIC/(n * (n + 1)) - 3 * (n + 1))/(1 -
        sum(TIES^3 - TIES)/(n^3 - n)))
    PARAMETER <- k - 1L
    PVAL <- pchisq(STATISTIC, PARAMETER, lower.tail = FALSE)
    names(STATISTIC) <- "Kruskal-Wallis chi-squared"
    names(PARAMETER) <- "df"
    RVAL <- list(statistic = STATISTIC, parameter = PARAMETER,
        p.value = PVAL, method = "Kruskal-Wallis rank sum test")
    return(RVAL)
}
```

a.

Write a pseudo code outlining what the function does.

b.

For example data, call the function in two ways: once using `x` as a list and once using `x` as a vector with a corresponding `g` argument. Ensure that the two different function calls return the same thing by aligning the inputs.

c.

Make a faster version of `kwtest()` that only computes the Kruskal-Wallis **test statistic** when the input is a numeric variable `x` and a variable `g` which gives the group membership. You can try simplifying the function above or by coding from the mathematical definition (see https://en.wikipedia.org/wiki/Kruskal%E2%80%93Wallis_one-way_analysis_of_variance). This function should also perform some checks to ensure the correctness of the inputs (use `kwtest()` as inspiration).

d.

Consider the following scenario. You have samples available from multiple experiments $m = 1000$ where you collect the numerical values for the quantity of interest `x` and the group membership for n individuals. The first 20 individuals in each sample belong to group 1, the following 20 individuals in each sample belong to group 2, the last 10 individuals in each sample belong to group 3. Use the following code to simulate such a data structure:

```
set.seed(1234)
m <- 1000 # number of repetitions
n <- 50   # number of individuals
X <- matrix(rt(m * n, df = 10), nrow = m)
grp <- rep(1:3, c(20, 20, 10))
```

Write a function which performs the Kruskal-Wallis test using the function `stats:::kruskal.test.default()` for m repeated experiments and returns a vector of m test statistics. The input of this function are a matrix `X` with m rows and n columns and a vector `g` which gives the grouping.

e.

Write a function which performs the Kruskal-Wallis test using the function in point c. for m repeated experiments and returns a vector of m test statistics. The input of this function are a matrix `X` with m rows and n columns and a vector `g` which gives the grouping.

f.

Compare the performance of the two approaches using a benchmarking package on the data generated above. Comment on the results.

g.

Now consider vectorizing the function in point c. Compare this approach to the other two. Comment on the results.