Case Study 1

AKSTA Statistical Computing

The .Rmd and .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 .PDF is not in a decent form.

Ratio of Fibonacci numbers

1. Write two different R functions which return the sequence $r_n = F_{n+1}/F_n$ where F_n is the nth Fibonacci number, once using `for` and once using `while`.
2. Benchmark the two functions for $n = 100$ and $n = 1000$ (you can use package **microbenchmark** or package **bench** for this purpose). Which function is faster?
3. Plot the sequence for $n = 100$. For which n it starts to converge? What is this number?

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

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

Using R, show that golden ratio satisfies the Fibonacci-like relationship (go up to $n = 1000$):

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

Use once `==` and once `all.equal` to compare the two sides of the equation. What do you observe? If there are any differences, what can be the reason?

Game of craps

The game of craps is played as follows. First, you roll two six-sided dice; let x be the sum of the dice on the first roll. If $x = 7$ or 11 you win, otherwise you keep rolling until either you get x again, in which case you also win, or until you get a 7 or 11, in which case you 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.

Readable and efficient code

Read over the code below and perform the following:

- Wrap it into a function `foobar0` which has arguments `x` and `z` and which returns the vector `x` at the end of the following code.
- Rewrite this into a function `foobar` which is easier to read, by reducing repetitive code. E.g. `foobar` might call a function to check the input, and another function to perform the three lines of computation. Also, use a function such as `paste` to create the error messages.
- Check that the two versions produce the same output using the function `all.equal`.

```
set.seed(1)
x <- rnorm(100)
z <- rnorm(100)
if (sum(x >= .001) < 1) {
  stop("step 1 requires 1 observation(s) with value >= .001")
}
fit <- lm(x ~ z)
r <- fit$residuals
x <- sin(r) + .01
if (sum(x >= .002) < 2) {
  stop("step 2 requires 2 observation(s) with value >= .002")
}
fit <- lm(x ~ z)
r <- fit$residuals
x <- 2 * sin(r) + .02
if (sum(x >= .003) < 3) {
  stop("step 3 requires 3 observation(s) with value >= .003")
}
fit <- lm(x ~ z)
r <- fit$residuals
x <- 3 * sin(r) + .03
if (sum(x >= .004) < 4) {
  stop("step 4 requires 4 observation(s) with value >= .004")
}
fit <- lm(x ~ z)
r <- fit$residuals
x <- 4 * sin(r) + .04
x
```