Intro to R Programming: Lab 2

Scalar summary functions

Task 1 Create a vector `x` using the command

```
x <- seq(1, 5, by=0.3)
```

1. Use the commands `mean` and `sd` to compute the mean and standard deviation of `x`.
2. Standardise `x` by subtracting the mean and dividing by the standard deviation. Store this result in a vector called `x.standardised`.
3. Check that the mean and standard deviation of `x.standardised` are 0 and 1, respectively.

Task 2 Create a vector of length 100 from the $\mathsf{N}(1, 1)$ distribution using

```
x <- rnorm(100, mean=1, sd=1)
```

and compute the test statistic of the one-sample t-test for the null hypothesis that $\mu = 0$:

$$t = \sqrt{n}\frac{\bar{x}}{\sqrt{s_x^2}}, \qquad \text{with } s_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Task 3 Use R to calculate (or approximate the following sums and products)

$$\sum_{i=0}^{9} 2^i, \text{ and } \sum_{i=1}^{\infty}\frac{1}{2^i}$$

Task 4 Use the code

```
x <- rnorm(100)
```

to generate a sample of size 100 from the $\mathsf{N}(0, 1)$ distribution.

1. Compute the mean and the median of `x`. Re-run the above line of code and re-compute the mean and the median five times. What can you observe?
2. Rather than generating a sample of size 100 from the standard normal distribution we will now generate a sample from the Cauchy distribution using

```
x <- rcauchy(100)
```

At first sight the Cauchy distribution is just like standard normal distribution symmetric around 0, i.e. it is centred around 0. Compute the mean and the median of `x`. Re-run the above line of code and re-compute the mean five times. What can you observe? Try to explain why this happens.

3. One way of "robustifying" the mean is to compute the so-called 20%-trimmed mean by removing the 10 smallest and the 10 largest observations and computing the mean of the remaining 80 observations. Compute the 20%-trimmed mean of both a sample from the standard normal distribution and the Cauchy distribution. Your code should automatically find the 10 smallest and largest values using the function `order` or `sort`.

Vectors and sequences

Task 5 Create a vector `u` containing all integers between 1 and 100. Replace all elements from `u` that are less than 55 by the number 0.

Task 6 Explain what values the logical vectors `c` and `d` will take before running the code in R.

```
a <- c(TRUE,FALSE)
b <- c(FALSE,FALSE)
c <- (a & !b)
d <- !(a | b)
```

Task 7 Create the following patterned sequences as efficiently as possible.

```
1 2 3 4 5 ... 10
1 2 3 1 2 3 1 2 3 1 2 3
1 1 2 2 3 3
1 2 3 4 ... 18 19 20 19 18 ... 4 3 2 1
1 4 9 16 25 36 49 64 81 100
1 2 4 8 16 32 64 128 256 512 1024
```

Matrices

Task 8 Create the following matrix and store it as `P`.

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 3 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 9 \end{pmatrix}$$

1. Print the first row and the second column of `P`.
2. Print the submatrix that consists of the first three rows and first two columns of `P`.
3. Compute the transpose and inverse of `P`.
4. Replace the first row of `P` by $(1, 2, 3, 4, 5)$.
5. Replace all non-zero entries of the matrix `P` by `1`.

Task 9 Matrix multiplication is associative, i.e. for two matrices $\mathbf{A}$ and $\mathbf{B}$ and a vector $\mathbf{x}$ we have that $(\mathbf{A} \cdot \mathbf{B}) \cdot \mathbf{x} = \mathbf{A} \cdot (\mathbf{B} \cdot \mathbf{x})$.

Create matrices `A` and `B`, and a vector `x` using the commands

```
A <- matrix(rnorm(9e6), nrow=3e3)     # Create matrix A with random entries (3000x3000)
B <- matrix(rnorm(9e6), nrow=3e3)     # Create matrix B with random entries (3000x3000)
x <- rnorm(3e3)                       # Create vector x with random entries (3000x1)
```

Because of the associativity of matrix multiplication both

`(A%*%B)%*%x`

and

`A%*%(B%*%x)`

give, except for rounding errors, the same answer. However the first command is much slower than the second command (3 seconds and 0.1 seconds on my computer), though both give the same answer. Explain why.

Task 10 Define a (symmetric and positive definite) matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 20 & 26 \\ 3 & 26 & 70 \end{bmatrix}$$

and a vector $\mathbf{b} = [4, 52, 31]$.

1. Use `solve` to compute $\mathbf{z} = \mathbf{A}^{-1}\mathbf{b}$, which is the solution to the system of equations $\mathbf{Az} = \mathbf{b}$:

$$\begin{array}{rcrcrcr} z_1 & + & 2z_2 & + & 3z_3 & = & 4 \\ 2z_2 & + & 20z_2 & + & 26z_3 & = & 52 \\ 3z_1 & + & 26z_2 & + & 70z_3 & = & 31 \end{array}$$

2. Use `chol` and `t` to compute the Choleski factor $\mathbf{L}$.
3. Verify that $\mathbf{LL}^{\mathrm{T}}$ is (except for rounding errors) identical to $\mathbf{A}$.
4. We can now exploit that $\mathbf{z} = \mathbf{A}^{-1}\mathbf{b} = (\mathbf{LL}^{\mathrm{T}})^{-1}\mathbf{b} = \mathbf{L}^{-\top}\underbrace{\mathbf{L}^{-1}\mathbf{b}}_{=\mathbf{v}}$ and use this so calculate $\mathbf{z}$.
5. Compute $\mathbf{v} = \mathbf{L}^{-1}\mathbf{b}$ using the function `solve`. Because of the lower-triangular structure of $\mathbf{L}$, this calculation can be done more efficiently using forwardsolving, `forwardsolve(L,b)`.
6. Compute $\mathbf{z} = \mathbf{L}^{-\top}\mathbf{v}$ using the functions `solve` and `t`. Because of the upper-triangular structure of $\mathbf{L}^{\top}$, this calculation can be done more efficiently using backsolving, `backsolve(t(L),v)`.
7. Compute the square of the product of the diagonal entries of $\mathbf{L}$ and compare it to the determinant of $\mathbf{A}$, which you can calculate using the function `det(A)`.