**Prof. Dr. Steffen Wagner**
Angewandte Statistik
Fachbereich II
Berliner Hochschule für Technik

# Exercise 08: Lists, *apply and simulations

Statistical Computing – WiSe 2022/23

## Recap: lists

1. Explain the difference between generic and atomic vectors.
2. Create a generic vector named `aList`, i.e. a list, with the following elements:

```
aList
```

```
## $aCharacter
## [1] "a"
##
## $aNumber
## [1] 1
##
## $aFactor
## [1] a b c
## Levels: c a b
##
## $aDataFrame
##   aCharacter aNumber someNumers
## 1          a       1          1
## 2          a       1          2
## 3          a       1          3
```

4. What is the result, or in other words the class, of the following expressions. Think first, check your results in R!

```
class(aList[[1]])
class(aList[["aNumber"]])
class(aList$x)
class(aList[4])
class(aList$aDataFrame)
class(aList$aDataFrame$aCharacter[1])
```

**Recap:** linear regression

5. Run the example of the help page for the function `lm()`. Which class has lm.D9. Basically an object of class `lm` is very similar to a list. Which elements does `lm.D9` contain? Access the elements of `lm.D9` by name and position to retrieve the residuals and store them in an object residuals. To confirm your results:

6. Specify which of the two variables `weight` or `group` is the explanatory and which is the dependent one.

7. For the OLS estimation the variable `group` is internally coded as binary 0/1 variable . The coding is as follows

$$group_{intern} = \begin{cases} 0 & \text{when } group = \text{'ctl'} \\ 1 & \text{when } group = \text{'trt'} \end{cases}.$$

How can you interprete the estimated coefficients `coef(lm.D9)` based on the information about the internal coding?

## The `*apply` family

The `*apply` can be seen as an alternative to loops whenever the individual outcomes per loop do *not* depend on each other. We will discuss the following functions

- `apply`
- `tapply` (see exercise sheet 05)
- `lapply` and `sapply`
- `mapply`

### `apply`: **applying a function to margins of an array or matrix**

1. Get the following matrix of 5 rows and call it 'm'.

```
m <- matrix(data = c(6, 17, 625, 5, -Inf, 100:104, 5, 9, 35, 79, 1, NA, 246:249), nrow = 5)
m
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    6  100    5   NA
## [2,]   17  101    9  246
## [3,]  625  102   35  247
## [4,]    5  103   79  248
## [5,] -Inf  104    1  249
```

2. Get the mean of each row and the sum of each column using `apply`.

```
# row means:
apply(m , MARGIN = 1, mean)
```

```
## [1]     NA  93.25 252.25 108.75   -Inf
```

```
# column sums:
apply(m , MARGIN = 2, sum)
```

```
## [1] -Inf  510  129   NA
```

3. Adjust your function calls so that missing values are ignored.

```
# row means:
apply(m , MARGIN = 1, mean, na.rm = TRUE)
```

```
## [1]  37.00  93.25 252.25 108.75   -Inf
```

```
# column sums:
apply(m , MARGIN = 2, sum, na.rm = TRUE)
```

```
## [1] -Inf  510  129  990
```

4. What additional argument can you use calculating the row means so that extreme values like `-Inf` is ignored. Check the help for `mean`.

```
# trimmed row means:
apply(m , MARGIN = 1, mean, na.rm = TRUE, trim = 0.3)
```

5. Sort the columns in ascending order so that your result looks like:

```r
# trimmed row means:
apply(m , MARGIN = 2, sort, na.last = TRUE)
```

```
##      [,1] [,2] [,3] [,4]
## [1,] -Inf  100    1  246
## [2,]    5  101    5  247
## [3,]    6  102    9  248
## [4,]   17  103   35  249
## [5,]  625  104   79   NA
```

6. Check the help for the data set `Titanic` and discuss the following code:

```r
## Higher survival rates in children?
apply(Titanic, MARGIN = c(3, 4), sum)
## Higher survival rates in females?
apply(Titanic, MARGIN = c(2, 4), sum)
```

## Using `lapply` on lists and data.frames

1. Calculate the `summary` for each element of the object `aList` created in exercise 1.

```
lapply(aList, summary)
```

2. Create a new list named `poisList` using `lapply`. The list shall consist of 4 vectors each containing 10 poisson distributed random variables drawn from distributions with parameters $\lambda_i \in \{1, 2, 5, 10\}$. Hints:

   - Make use of the function `set.seed()` to obtain reproducible results.
   - Check the help for `rpois` and think about argument matching in R.

```
set.seed(1234567890)
poisList <- lapply(c(1, 2, 5, 10), rpois, n = 10)
```

3. Calculate the mean and the variance for each element of `poisList` using `lapply`. What values do you expect?

```
lapply(poisList, mean)
lapply(poisList, var)
```

4. Extract the fourth element of each vector stored in `poisList` using an *anonymous* function.

```
lapply(poisList, function(x) x[4])
```

5. Explain the following statement: *"A data frame is a list, but not every list is a data frame."*

6. Load the dataset `mtcars` into the global environment and calculate the range of values for each variable contained in the data set using `lapply`.

```
data(mtcars)
lapply(mtcars, range)
```

7. Repeat the calculation of ranges using `sapply`. Discuss the differences with respect to the the class of the result.

```
sapply(mtcars, range)
```

### `mapply`

1. Use `mapply` to get a list of 6 elements. The list is a set of vectors containing the letters `A`, `B`, `C`, `D`, `E`, `F`. The lengths of those 6 elements decreases step by step from 6 to 1. Expected result:

```
mapply(rep, c("A", "B", "C", "D", "E", "F"), 6:1)
```

```
## $A
## [1] "A" "A" "A" "A" "A" "A"
##
## $B
## [1] "B" "B" "B" "B" "B"
##
## $C
## [1] "C" "C" "C" "C"
```

```
##
## $D
## [1] "D" "D" "D"
##
## $E
## [1] "E" "E"
##
## $F
## [1] "F"
```

## Random variables and the central limit theorem

This exercise illustrates the central limit theorem (CLT) using simulations of random variables. Consider i.i.d. random variables $X_1, \dots, X_n$ with

$$X_i \sim Uniform(1, 9).$$

The CLT states that

$$\lim_{n\to\infty} \overline{X} \overset{a}{\sim} N\left(\mu, \frac{\sigma^2}{n}\right)$$

with $\mu = E(X)$ and $\sigma^2 = \text{Var}(X)$.

a) Check https://en.wikipedia.org/wiki/Continuous_uniform_distribution and specify the CLT explicitly for the given situation. $\lim_{n\to\infty} \overline{X} \overset{a}{\sim} N\left(5, \frac{5.3333333}{n}\right)$

b) Illustrate the CLT by simulating $M = 100$ times $\overline{X}$ for a sample sizes $n = 5$. Proceed as follows:

- Draw 100 times 5 uniformly distributed observations using the function `runif`. Check `?runif`.
- Store the 500 values in a matrix with dimension $M \times n$.
- Use the function `rowMeans` to calculate $\overline{X}$ and store the result in a vector `xMeans`.
- Plot a histogramm of `xMeans` showing the density of the observed values..

```r
# Upper and lower boundary for the uniform distribution
a <- 1
b <- 9

# Sample size and number of samples
n <- 5
M <- 100
```

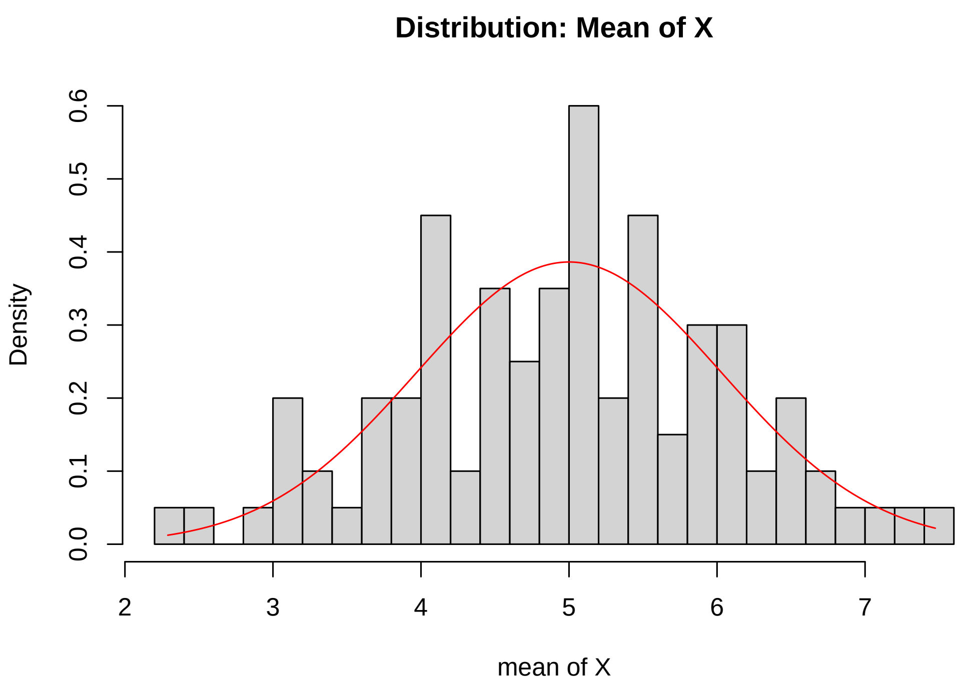
```r
# matrix with iid variables
m <- matrix(runif(n * M, min = a, max = b),
            nrow = M, ncol = n)
# calculation of means
xMeans <- rowMeans(m)
```

c) Plot the theoretical distribution according to the CLT on top of the histogramm. Proceed as follows:

- Create a vector `x` with 1000 values ranging from `min(meanX)` to `max(meanX)`. Use the function `seq()`.
- Calculate for each element of `x` the corresponding density of $N\left(\mu, \frac{\sigma^2}{n}\right)$ with the appropriate values for $\mu$, $sigma^2$ and $n$. Check the help for `dnrom`.
- Use the function `lines()` to plot the calculated densities.

```r
# visualisation
hist(xMeans,
     breaks = 20,
     probability = TRUE,
     xlab = "mean of X",
     main = "Distribution: Mean of X")
```

```r
# density plot
x <- seq(min(xMeans), max(xMeans), length.out = 1000)
dX <- dnorm(x, mean = (b+a)/2, sd = sqrt(1/12*(b-a)^2/n))
lines(x, dX, col = "red")
```



d) Transform your R code to a function named `cltDensity()` accepting the arguments `n` and `M` (with default value $M = 100$) so that you can easily make plots for other sample sizes $n$. Adjust the plot title of the histogram so that the sample size $n$ is printed. Check the help for `paste()`.

```r
cltDensity <- function(n, M = 100) {

  a <- 1
  b <- 9
  # matrix with iid variables
  m <- matrix(runif(n * M, min = a, max = b),
              nrow = M, ncol = n)
  # calculation of means
  xMeans <- rowMeans(m)

  # density plot
  x <- seq(min(xMeans), max(xMeans), length.out = 1000)
  dX <- dnorm(x, mean = (b+a)/2, sd = sqrt(1/12*(b-a)^2/n))

  # visualisation
  hist(xMeans,
       breaks = 20,
       probability = TRUE,
       xlab = "mean of X",
       ylim = range(pX),
       main = paste0("Distribution: Mean of X (sample size n=", n, ")"))
  lines(x, dX, col = "red")
}
cltDensity(100)
```

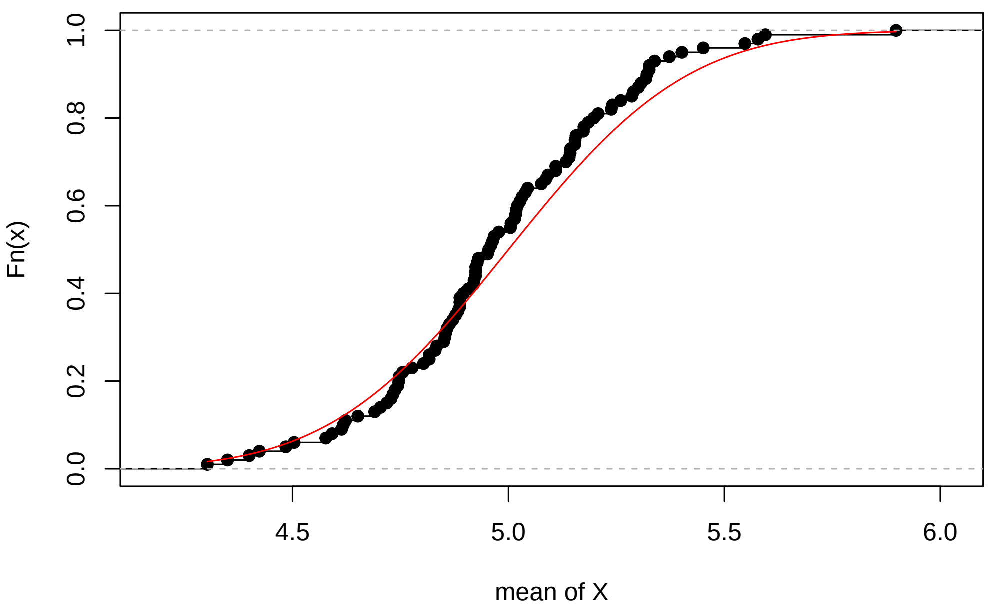
e) Play with the function `cltDensity` and check the results for different sample sizes $n \in \{1, 2, 5, 10, 20, 50, 100, 1000\}$.

f) Now write a second function `cltDistrFun` where the empirical distribution function and distribution function of the appropriate normal distribution is plotted. Your result should look comparable to

```r
    cltDistrFun <- function(n, M = 100) {

      a <- 1
      b <- 9
      # matrix with iid variables
      m <- matrix(runif(n * M, min = a, max = b),
                  nrow = M, ncol = n)
      # calculation of means
      xMeans <- rowMeans(m)

      # density plot
      x <- seq(min(xMeans), max(xMeans), length.out = 1000)
      pX <- pnorm(x, mean = (b+a)/2, sd = sqrt(1/12*(b-a)^2/n))


      # visualisation
      plot(ecdf(xMeans),
           xlab = "mean of X",
           main = paste0("Distribution: Mean of X (sample size n=", n, ")"))
      lines(x, pX, col = "red")
    }
```

```r
cltDistrFun(n = 50)
```

Distribution: Mean of X (sample size n=50)


Fn(x)
mean of X